

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| И. о. зав. кафедрой | | «КБИС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | О.А. Сафарьян |
| подпись | | И.О. Фамилия |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | |

**КУРСОВАЯ РАБОТА**

Тема: «ПРОГРАММНОЕ СРЕДСТВО ОПРЕДЕЛЕНИЯ АВТОРСТВА ТЕКСТОВ»

Дисциплина (модуль): «Теория кодирования, сжатия и восстановления информации»

Специальность: 10.05.01 Компьютерная безопасность

Специализация: Математические методы защиты информации

Обозначение курсовой работы ТКСВИ 630000.000 Группа ВКБ42

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Михайлов

подпись, дата

Курсовая работа защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ доцент, Р.В. Егорова

подпись, дата

Ростов-на-Дону

2024



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

|  |  |  |
| --- | --- | --- |
|  | | |
| И. о. зав. кафедрой | | «КБИС» |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | О.А. Сафарьян |
| подпись | |  |
| «\_\_\_» | \_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | |

**ЗАДАНИЕ**

на выполнение курсовой работы

Тема: «ПРОГРАММНОЕ СРЕДСТВО ОПРЕДЕЛЕНИЯ АВТОРСТВА ТЕКСТОВ»

Дисциплина (модуль): Теория кодирования, сжатия и восстановления информации

Обучающийся: Михайлов Александр Сергеевич

Обозначение курсового проекта (работы) ТКСВИ.630000.000 Группа: ВКБ42

Срок представления проекта (работы) к защите «\_\_\_» \_\_\_\_\_\_\_\_\_ 2024 г.

Исходные данные для выполнения курсового проекта (работы):

1. Задание на выполнение курсовой работы;
2. Федеральный закон № 149-ФЗ «Об информации, информационных технологиях и о защите информации»;
3. Теоритические основы для определения авторства текстов;
4. Модель подхода к определению авторства (общая структурная схема).

|  |
| --- |
| Содержание курсовой работы |
| Введение: |
| В данном разделе рассматривается актуальность темы курсовой работы, её цель, задачи, которые необходимо решить в ходе выполнения, а также определение объекта и предмета данной работы |
| Наименование и краткое содержание разделов: |
| 1. В данном разделе рассматривается понятие существующих методов определения авторства текстов, анализ преимуществ и недостатков существующих решений и формулировка требований к программному средству. |
| 1. В данном разделе рассматривается выбор и обоснование метода определения авторства, происходит анализ, выбор и обоснование технологического стека, язык программирования, библиотеки и фреймворки, средства разработки и тестирования, разработка программного кода, тестирование и отладка программного средства, а также оценка производительности и эффективности. |
| 1. В данном разделе рассматриваются результаты выполнения работы программного средства, демонстрация выполнения программного кода и полученные результаты. |
|  |
| Заключение:  Выводы по работе. |

|  |  |  |
| --- | --- | --- |
| Руководитель проекта (работы) | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | доцент Р.В. Егорова |
| Задание принял к исполнению | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  подпись, дата | А. С. Михайлов |

Содержание

[Введение 5](#_Toc184321629)

[1 Основные подходы к определению авторства текстов 7](#_Toc184321630)

[1.1 Общая информация по определению авторства текстов 7](#_Toc184321631)

[1.2 Методы статистического анализа 7](#_Toc184321632)

[1.3 Лингвистический анализ 9](#_Toc184321633)

[2 Разработка программного средства 10](#_Toc184321634)

[2.1 Обоснование выбора языка и среды разработки 10](#_Toc184321635)

[2.2 Архитектура программного средства 11](#_Toc184321636)

[2.3 Основные методы и классы программы 12](#_Toc184321637)

[2.3.1 Класс «Form1» 12](#_Toc184321638)

[3 Демонстрация программного средства 14](#_Toc184321639)

[Заключение 17](#_Toc184321640)

[Перечень использованных информационных ресурсов 18](#_Toc184321641)

[Приложение А Листинг кода 19](#_Toc184321642)

# **Введение**

В настоящее время, когда в силу цифровизации большинства областей человеческой деятельности значительная часть контента создается и распространяется в цифровом виде, важность вопросов аутентификации текстов и установления авторства над ними становится все более актуальной. Вместе с увеличением объема информации и доступности технологий увеличивается вероятность плагиата и несанкционированного использования текстов, призванных обеспечить защиту интересов авторов и правообладателей. Программа определения авторства текстов является программным обеспечением, предназначенным для анализа текстов и установления возможного авторства с использованием статистических методов и приемов лингвистики.

Сначала определение авторства текстов было основано на стилистике и лексикографии, и в XIX веке лингвисты и литературоведы пытались истолковать анонимные или псевдонимные произведения. После развития компьютерных и информационных технологий стали применяться более тонкие и точные методы анализа текстов. Из-за обработки больших объемов данных развития различные алгоритмы определения авторства текстов, включая статистику, начали использовать лингвистический подход.

Такие методы позволяют не только определить вероятного автора текста, но также выявить плагиат, который сравнивается с базами данных уже опубликованных работ. Множество разнообразных алгоритмов и программных средств определения авторства текстов различаются по принципу работы, точности определения и области применения. Одним из самых известных методов подобного анализа является анализ частотности употребления слов или фраз, поскольку он может отражать уникальные членения и связок слов авторов. Некоторые другие методы – анализ синтаксических конструкций, морфологические признаки и даже анализ.

Объектом курсовой работы является программное средство определения авторства текстов.

Предметом данной работы является программная реализация алгоритма определения авторства текстов на основе статистического анализа частотности слов и фраз, а также использование лингвистического метода для повышения точности определения.

Таким образом целью курсовой работы является реализация программного средства для авторства текстов с использованием статистических методов.

Цель определила следующие задачи:

1. рассмотреть теорию определения авторства текстов и существующие методы анализа;
2. проанализировать и описать реализацию алгоритма определения авторства на основе частности слов и фраз;
3. реализовать программное средство для определения авторства текстов текста с использованием выше перечисленных алгоритмов.

# **1 Основные подходы к определению авторства текстов**

## **Общая информация по определению авторства текстов**

Определение авторства текстов представляет собой сложную задачу, которая требует комплексного подхода, включающего в себя анализ лингвистических, статистических и стилистических особенностей текста. В современном мире, где большая часть информации создается и распространяется в цифровом, вопросы аутентификации и определения авторства становятся все более актуальными.

Первые попытки определения авторства текстов были предприняты еще в XIX веке, когда лингвисты и литературоведы пытались установить авторство анонимных или псевдонимных произведений на основе стилистических особенностей и лексических предпочтений. Однако, с появлением компьютеров и развитием информационных технологий, стало возможным применять более сложные и точные методы анализа текстов.

Современные методы определения авторства текстов включают в себя использование статистических методов и лингвистического анализа. Эти методы позволяют не только определить вероятного автора текста, но и выявить плагиат, сравнивая текст с базами данных уже опубликованных работ ошибок. [1]

## **Методы статистического анализа**

Определение авторства текстов с помощью статистического анализа основывается на предположении, что каждый автор имеет уникальный (письменный) почерк, который проявляется на опыте использования родного языка. Сам статистический метод позволяет выявить эти особенности и сравнить их с особенностями текстов, авторство которых известно.

К основным методам, которые мы будем использовать, можно отнести:

Статистический анализ частности:

1. Частность слов: подсчет частоты встречаемости отдельных слов в тексте
2. Частность биграмм и триграмм: подсчет частоты встречаемых пар и троек последовательных слов
3. Частность морфологических форм: подсчет частоты встречаемости различных форм слов
4. Частность синтаксических конструкций: подсчет частоты встречаемости различных синтактических структур [2]

Также данный метод позволяет проводить анализы тематических и семантических моделей, проводить определение основных тем, затрагиваемых в тексте, сравнивать их, строить связи между словами и выявлять особенности, которые характерны для стиля написания.

Стоит отметить, что методы основаны на количественных показателях, что снижает влияние субъективных факторов, также эти методы применимы к большим текстовым объёмам и подвластны компьютерной реализации.

Зависимость от языка, жанра и наличие возможных ошибок – одни из недостатков данного метода. [3]

## **Лингвистический анализ**

Лингвистический анализ основан на изучении стилистических особенностей языка, характерных для конкретного автора. В отличие от статистического метода, который фокусируется на количественных показателях, лингвистический анализ исследует качественные характеристики текста.

К основным методам стилистического анализа следует отнести анализы лексики (изучение использования автором слов, идиомов, фразеологизмов, метафор и других лексических единиц), синтаксиса (изучение структуры предложения, использования конструкций, порядка слов и пунктуации), стилистических фигур (выявление использованных автором тропов, сравнений, антитез, градаций) и интонации

Семантический анализ – более сложный метод, который ориентирован на сравнение и проверку не только тематики текста, но и контекста, приводятся анализ семантических полей, группировка слов по признакам семантики и анализ их соотношения в тексте.

Помимо выше перечисленных методов, также имеются фразеологический (изучение автором использованных словосочетаний и выражений, а также их образа) и диахронический (анализ изменения стиля в разные периоды творчества и влияние на других авторов) анализы.

Из преимуществ лингвистического стиля можно выделить глубокое понимание стиля, которое позволяет выявить уникальные особенности языка автора, которые не всегда показаны в прошлом методе. Также существует возможность интерпретации в зависимости от эпохи и личностных качеств автора и комплексное исследование – сочетание статистического метода для повышения точности определения авторства.

Несмотря на все преимущества и особенности, существует ряд недостатков, такие как: субъективность исследования текстов, трудоемкость работы, требующие больших как умственных, так и материальных затрат. [4]

# **Разработка программного средства**

## **Обоснование выбора языка и среды разработки**

В качестве языка разработки был выбран C#.

C# является языком с Си-подобным синтаксисом и близок в этом отношении к C++ и Java. Поэтому, овладеть C# существенно легче, если есть навыки работы на C++ или Java [5].

C# является объектно-ориентированным и поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Объектноориентированный подход позволяет решить задачи по построению крупных, но в тоже время гибких, масштабируемых и расширяемых приложений. C# в настоящий момент активно развивается, и с каждой новой версией появляется все больше интересных функциональностей, такие как лямбда-выражения, динамическое связывание, асинхронные методы и т.д. [5].

В качестве среды разработки для реализации курсовой работы был выбран Visual Studio 2022.

Visual Studio представляет собой мощную среду разработки, которая объединяет весь цикл создания программного обеспечения в одном месте. Она включает в себя интегрированную среду разработки (IDE), которая позволяет писать, редактировать, отлаживать и компилировать код. Кроме того, Visual Studio предлагает широкий набор компиляторов, расширений и встроенных библиотек.

При разработке приложений на платформе .NET, Visual Studio 2022 предоставляет гибкие инструменты для работы с графическим интерфейсом. Вы можете как вручную редактировать XML-файлы, так и использовать мощный графический конструктор для создания интерфейса.

Visual Studio 2022 также упрощает поиск и установку библиотек с различных источников благодаря системе NuGet. Это позволяет разработчикам легко интегрировать сторонние решения в свои проекты.

В целом, Visual Studio отличается высокой функциональностью, гибкостью и интуитивно понятным интерфейсом. Она позволяет эффективно находить и исправлять ошибки в коде, делая процесс разработки более продуктивным.

## **Архитектура программного средства**

Программа построена на основе монолитной архитектуры, которая предполагает наличие единого большого программного блока, в данном случае класса, отвечающего за все основные вычислительные процессы и обеспечивающего работу программы.,

Такой подход оправдан для простых программных средств, ориентированных на выполнение ограниченного набора задач в рамках одной программной среды.

В контексте курсового проекта монолитная архитектура является оптимальным выбором по следующим причинам:

1. Ограниченный набор задач: Программа выполняет четко определенные задачи: сжатие, кодирование, распаковку и декодирование.
2. Простота реализации: Каждая из этих задач может быть реализована в рамках одного или нескольких методов одного класса, без необходимости создания отдельных классов для каждой задачи.
3. Удобство структурирования кода: Наличие всех функций в одном классе упрощает структурирование и читаемость кода, так как не требуется подключать отдельные функции для выполнения различных задач.

Таким образом, монолитная архитектура обеспечивает простоту реализации, эффективность и удобство использования для курсового проекта, ориентированного на выполнение конкретных задач.

## **Основные методы и классы программы**

В программе используется одна дополнительная библиотека: Microsoft.Office.Interop.Word – для считывания текста из файлов с расширением .docx и .doc. В самой же программе присутствует только один единый класс «Form1» в котором описан весь функционал.

### **2.3.1 Класс «Form1»**

Основной класс, в котором происходят все вычисления и процессы

программы. Здесь находятся функции для определения авторства текстов путем использования статистического и лингвистического анализов.

button1\_Click – ункция, вызываемая при нажатии первой кнопки в графическом интерфейсе Windows Forms. Открывает диалоговое окно для выбора двух файлов для сравнения. Если выбрано менее двух файлов, выводит сообщение об ошибке. В противном случае считывает содержимое файлов и вызывает функцию CompareTexts для сравнения текстов..

CompareTexts – основная функция, которая выполняет сравнение двух текстов. Вызывает функции предварительной обработки текстов, извлечения признаков и вычисления косинусного сходства.

PreprocessText – функция, выполняющая предварительную обработку текста. Приводит текст к нижнему регистру, удаляет пунктуацию и лишние пробелы, а также выполняет лемматизацию или стемминг слов (опционально).

ExtractFeatures – функция, извлекающая признаки из текста. Выполняет частотный анализ слов, биграмм и триграмм, а также (опционально) извлекает стилистические признаки.

CosineSimilarity– основная функция, вычисляющая косинусное сходство между двумя векторами признаков. Используется для оценки сходства текстов.

GetNGrams – вспомогательная функция, которая генерирует n-граммы (биграммы или триграммы) из массива слов.

RemoveFromQueue – вспомогательная функция которая удаляет элементы из очереди..

В таблице 2.1 представлены методы класса «Form1»:

Таблица 2.1 – Методы класса «Form1»

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Входные параметры | Описание метода | Возвращаемое значение |
| button1\_Click | Файлы с расширением .docx или .txt | Считывание данных из файлов | Данные внутри listbox |
| CompareTexts | Строка файлов | Считывает признаков и косинусного сходства | Данные сходства, внутри listbox |
| PreprocessText | Обработанный текст | Выполняет лемматизацию и стемминг слов | Данные сходства, внутри listbox |
| ExtractFeatures | Частотный анализ текста | Извлекает стилистические признаки | Данные сходства, внутри listbox |
| CosineSimilarity | Косинусное сходство текстов | Вычисляет основное сходство на основе функций | Данные сходства, внутри listbox |
| GetNGrams | Текст n-граммы | Генерация n-грамм из массива слов | Данные сходства, внутри listbox |

# **Демонстрация программного средства**

При запуске программы пользователя будет встречать графический интерфейс программы, продемонстрированный на рисунке 3.1:

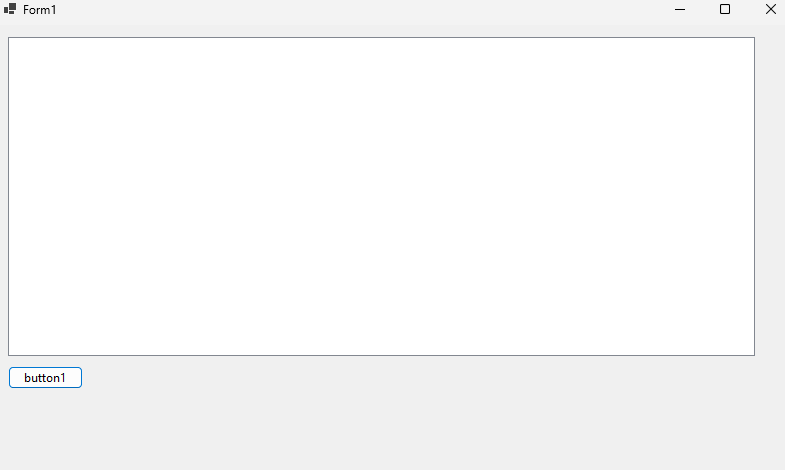


Рисунок 3.1 – Графический интерфейс программы.

Для определения сходств тестов, необходимо выбрать 2 файла, чтоб начать их сравнение. В данном случае текст считывается из текстового файла. На рисунке 3.2 показан процесс выбора текстового файла для загрузки данных в окно ввода.

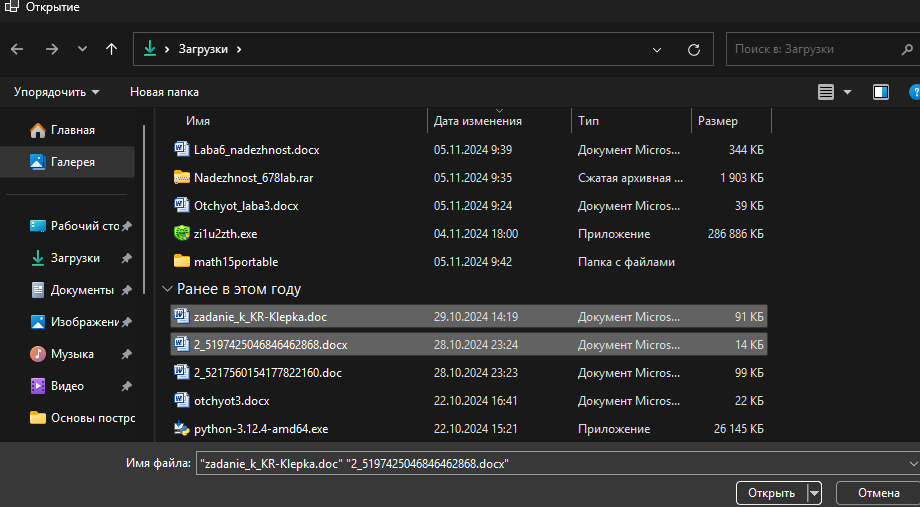


Рисунок 3.2 – Загрузка данных из текстового файла.

После ввода данных по нажатию кнопки будет проводиться сравнение двух тестов, используя методы статического и лингвистического анализов, а полученные вычисления выводятся на экран. На рисунке 3.3 продемонстрирован процесс сравнения авторства.

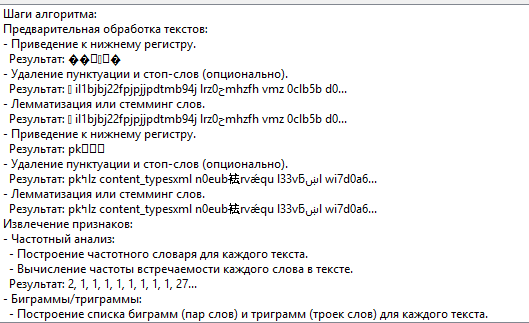


Рисунок 3.3 – Демонстрация процесса кодирования.

В окне вывода приводятся все возможные сравнения, в том числе сравнения как отдельных слов, так и триграмм и биграмм. В самом конце выводится коэффициент сходство, число, которое показывает, насколько выбранные файлы схожи друг с другом. На рисунке 3.4 продемонстрирована получившаяся, после всех процессов строка.

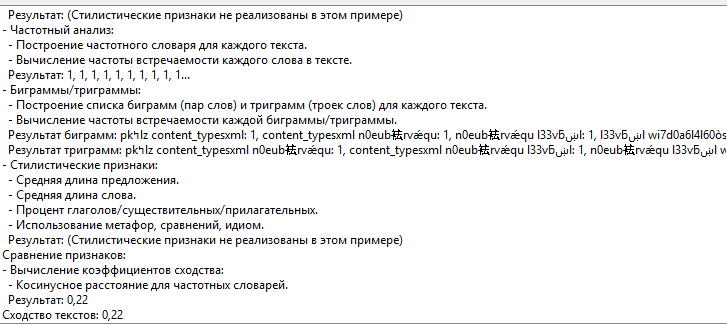


Рисунок 3.4 – Результат сходства текстов.

# **Заключение**

В рамках данной курсовой работы было разработано программное средство для определения авторства текстов с использованием статистических и лингвистических методов.

Также в ходе курсовой работы были выполнены следующие поставленные задачи:

1. Изучена теория определения авторства текстов и существующие исторические аспекты определения авторства, начиная с ручного анализа стилистики и лексикографии в XIX веке, и заканчивая современными компьютерными методами, основанными на статистике.
2. Проанализирован и детально изучен и описан алгоритм, основанный на анализе частотности употребления слов и фраз, который позволяет выявить уникальные особенности стиля автора.
3. Описана реализация алгоритма статистического и лингвистического методов.
4. Реализовано программа, которая позволяет анализировать тексты на основе частотности слов и фраз, а также использует лингвистические методы для повышения точности определения авторства.

В результате выполнения курсовой работы была достигнута поставленная цель, заключающаяся в разработке программного средства для определения авторства текстов с использованием статистических методов. Разработанное программное обеспечение может быть использовано для выявления плагиата, установления авторства анонимных или псевдонимных текстов, а также для исследования стилистических особенностей авторов.

# **Перечень использованных информационных ресурсов**

1. Holmes, D. I. The Evolution of Stylometry in Humanities Scholarship / D. I. Holmes // Literary and Linguistic Computing. – 1998. – Vol. 13, No. 3. – P. 111. – EDN JITRPL. .- Текст : Непосредственный.
2. Анализ качества производства многодвигательных систем с использованием методов графико-вероятностных методов статистического анализа / В. И. Кадошников, С. В. Куликов, О. С. Чекашкина, И. Д. Кадошникова // Процессы и оборудование металлургического производства : Межрегиональный сборник научных трудов / под редакцией С. И. Платова. Том Выпуск 8. – Магнитогорск : Магнитогорский государственный технический университет им. Г.И. Носова, 2009. – С. 161-166. – EDN SBSXYP.- Текст : Непосредственный.
3. Жданович, Е. С. Статистический анализ текста и задача определения авторства / Е. С. Жданович // 75-я научная конференция студентов и аспирантов Белорусского государственного университета : Материалы конференции: в 3 частях, Минск, 14–23 мая 2018 года. Том Часть 1. – Минск: Белорусский государственный университет, 2018. – С. 19-22. – EDN YPRHZJ.– Текст : Непосредственный.
4. Гудкова, С. А. Современные методы лингвистического анализа текстов: речевая агрессия и политический дискурс / С. А. Гудкова. – Тольятти : Анна, 2021. – 88 с. – ISBN 978-5-6045366-7-4. – EDN OICCSQ. – Текст : Непосредственный.
5. Александрова И. Л. Программирование на языке C#. Учебное пособие / И.Л. Александрова, Д.Н. Тумаков. – Издание 2-ое, исправленное и дополненное. – Казань: Казанский государственный университет, 2017. – 112 с. - ISBN 978-5-00019-914-5 – Текст : Непосредственный.

# **Приложение А Листинг кода**

*public Form1()*

*{*

*InitializeComponent();*

*}*

*private void button1\_Click(object sender, EventArgs e)*

*{*

*OpenFileDialog openFileDialog = new OpenFileDialog();*

*openFileDialog.Multiselect = true;*

*if (openFileDialog.ShowDialog() == DialogResult.OK)*

*{*

*if (openFileDialog.FileNames.Length < 2)*

*{*

*MessageBox.Show("Пожалуйста, выберите два файла для сравнения.");*

*return;*

*}*

*string textA = File.ReadAllText(openFileDialog.FileNames[0]);*

*string textB = File.ReadAllText(openFileDialog.FileNames[1]);*

*listBoxResults.Items.Clear();*

*listBoxResults.Items.Add("Шаги алгоритма:");*

*double similarity = CompareTexts(textA, textB);*

*listBoxResults.Items.Add($"Сходство текстов: {similarity:F2}");*

*}*

*}*

*private double CompareTexts(string textA, string textB)*

*{*

*listBoxResults.Items.Add("Предварительная обработка текстов:");*

*textA = PreprocessText(textA);*

*textB = PreprocessText(textB);*

*listBoxResults.Items.Add("Извлечение признаков:");*

*var featuresA = ExtractFeatures(textA);*

*var featuresB = ExtractFeatures(textB);*

*listBoxResults.Items.Add("Сравнение признаков:");*

*return CosineSimilarity(featuresA, featuresB);*

*}*

*private string PreprocessText(string text)*

*{*

*listBoxResults.Items.Add("- Приведение к нижнему регистру.");*

*text = text.ToLower();*

*listBoxResults.Items.Add($" Результат: {text.Substring(0, Math.Min(text.Length, 50))}...");*

*listBoxResults.Items.Add("- Удаление пунктуации и стоп-слов (опционально).");*

*text = Regex.Replace(text, @"[^\w\s]", "");*

*text = Regex.Replace(text, @"\s+", " ");*

*listBoxResults.Items.Add($" Результат: {text.Substring(0, Math.Min(text.Length, 50))}...");*

*listBoxResults.Items.Add("- Лемматизация или стемминг слов.");*

*// Здесь можно добавить лемматизацию или стемминг, если это необходимо.*

*listBoxResults.Items.Add($" Результат: {text.Substring(0, Math.Min(text.Length, 50))}...");*

*return text;*

*}*

*private int[] ExtractFeatures(string text)*

*{*

*listBoxResults.Items.Add("- Частотный анализ:");*

*listBoxResults.Items.Add(" - Построение частотного словаря для каждого текста.");*

*listBoxResults.Items.Add(" - Вычисление частоты встречаемости каждого слова в тексте.");*

*var words = text.Split(' ');*

*var wordCounts = words.GroupBy(w => w).ToDictionary(g => g.Key, g => g.Count());*

*var allWords = words.Distinct().OrderBy(w => w).ToArray();*

*var featureVector = new int[allWords.Length];*

*for (int i = 0; i < allWords.Length; i++)*

*{*

*featureVector[i] = wordCounts.ContainsKey(allWords[i]) ? wordCounts[allWords[i]] : 0;*

*}*

*listBoxResults.Items.Add($" Результат: {string.Join(", ", featureVector.Take(10))}...");*

*listBoxResults.Items.Add("- Биграммы/триграммы:");*

*listBoxResults.Items.Add(" - Построение списка биграмм (пар слов) и триграмм (троек слов) для каждого текста.");*

*listBoxResults.Items.Add(" - Вычисление частоты встречаемости каждой биграммы/триграммы.");*

*var bigrams = GetNGrams(words, 2);*

*var trigrams = GetNGrams(words, 3);*

*var bigramCounts = bigrams.GroupBy(b => b).ToDictionary(g => g.Key, g => g.Count());*

*var trigramCounts = trigrams.GroupBy(t => t).ToDictionary(g => g.Key, g => g.Count());*

*listBoxResults.Items.Add($" Результат биграмм: {string.Join(", ", bigramCounts.Take(10).Select(b => $"{b.Key}: {b.Value}"))}...");*

*listBoxResults.Items.Add($" Результат триграмм: {string.Join(", ", trigramCounts.Take(10).Select(t => $"{t.Key}: {t.Value}"))}...");*

*listBoxResults.Items.Add("- Стилистические признаки:");*

*listBoxResults.Items.Add(" - Средняя длина предложения.");*

*listBoxResults.Items.Add(" - Средняя длина слова.");*

*listBoxResults.Items.Add(" - Процент глаголов/существительных/прилагательных.");*

*listBoxResults.Items.Add(" - Использование метафор, сравнений, идиом.");*

*// Здесь можно добавить обработку стилистических признаков, если это необходимо.*

*listBoxResults.Items.Add(" Результат: (Стилистические признаки не реализованы в этом примере)");*

*return featureVector;*

*}*

*private double CosineSimilarity(int[] vectorA, int[] vectorB)*

*{*

*listBoxResults.Items.Add("- Вычисление коэффициентов сходства:");*

*listBoxResults.Items.Add(" - Косинусное расстояние для частотных словарей.");*

*// Выравнивание векторов по длине*

*int maxLength = Math.Max(vectorA.Length, vectorB.Length);*

*Array.Resize(ref vectorA, maxLength);*

*Array.Resize(ref vectorB, maxLength);*

*double dotProduct = 0.0;*

*double magnitudeA = 0.0;*

*double magnitudeB = 0.0;*

*for (int i = 0; i < maxLength; i++)*

*{*

*dotProduct += vectorA[i] \* vectorB[i];*

*magnitudeA += vectorA[i] \* vectorA[i];*

*magnitudeB += vectorB[i] \* vectorB[i];*

*}*

*magnitudeA = Math.Sqrt(magnitudeA);*

*magnitudeB = Math.Sqrt(magnitudeB);*

*if (magnitudeA == 0 || magnitudeB == 0)*

*return 0.0;*

*double similarity = dotProduct / (magnitudeA \* magnitudeB);*

*listBoxResults.Items.Add($" Результат: {similarity:F2}");*

*return similarity;*

*}*

*private string[] GetNGrams(string[] words, int n)*

*{*

*var ngrams = new string[words.Length - n + 1];*

*for (int i = 0; i < words.Length - n + 1; i++)*

*{*

*ngrams[i] = string.Join(" ", words.Skip(i).Take(n));*

*}*

*return ngrams;*

*}*

*}*