«Утверждаю»

доцент кафедры ИТИС

\_\_\_\_\_\_\_\_\_\_\_\_\_(Хуснутдинов Р.М.)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_202\_\_г.

**Справочная документация**

для программного продукта

«**Анализатор текста**»

Казань, 2025

Код программы

import os

import docx

from difflib import SequenceMatcher

def read\_txt\_file(filepath):

    with open(filepath, 'r', encoding='utf-8') as file:

        return file.read()

def read\_doc\_file(filepath):

    doc = docx.Document(filepath)

    return '\n'.join([para.text for para in doc.paragraphs])

def read\_document(filepath):

    if filepath.endswith('.txt'):

        return read\_txt\_file(filepath)

    elif filepath.endswith('.doc') or filepath.endswith('.docx'):

        return read\_doc\_file(filepath)

    else:

        return ''

def get\_documents\_from\_folder(folder\_path):

    documents = {}

    for filename in os.listdir(folder\_path):

        if filename.endswith('.txt') or filename.endswith('.doc') or filename.endswith('.docx'):

            full\_path = os.path.join(folder\_path, filename)

            content = read\_document(full\_path)

            documents[filename] = content

    return documents

def compare\_texts(text1, text2):

    matcher = SequenceMatcher(None, text1, text2)

    return matcher.ratio()

def analyze\_original\_for\_plagiarism(documents, original\_filename='original.txt'):

    if original\_filename not in documents:

        print(f"Файл {original\_filename} не найден.")

        return {}

    original\_text = documents[original\_filename]

    max\_similarity = 0.0

    source\_document = None

    similarities = {}

    for filename, text in documents.items():

        if filename == original\_filename:

            continue

        similarity = compare\_texts(original\_text, text)

        similarities[filename] = similarity

        if similarity > max\_similarity:

            max\_similarity = similarity

            source\_document = filename

    originality = 1 - max\_similarity

    result = {

        'Процент оригинальности original.txt': round(originality \* 100, 2),

        'Наибольшее совпадение с': source\_document,

        'Процент совпадения': round(max\_similarity \* 100, 2),

        'Подробности': {doc: round(sim \* 100, 2) for doc, sim in similarities.items()}

    }

    return result

def save\_report(result, output\_file='plagiarism\_report.txt'):

    with open(output\_file, 'w', encoding='utf-8') as f:

        f.write(f'Проверка документа: original.txt\n')

        f.write(f'Оригинальность: {result["Процент оригинальности original.txt"]}%\n')

        f.write(f'Наибольшее совпадение с: {result["Наибольшее совпадение с"]}\n')

        f.write(f'Процент совпадения: {result["Процент совпадения"]}%\n')

        f.write('\nДетальное сравнение с каждым документом:\n')

        for doc, percent in result['Подробности'].items():

            f.write(f'- {doc}: {percent}% совпадения\n')

def main():

    folder\_path = input("Введите путь к папке с документами: ")

    documents = get\_documents\_from\_folder(folder\_path)

    result = analyze\_original\_for\_plagiarism(documents, original\_filename='original.txt')

    print(f"\nПроверка original.txt на заимствование:")

    print(f'Оригинальность: {result["Процент оригинальности original.txt"]}%')

    print(f'Наибольшее совпадение с: {result["Наибольшее совпадение с"]}')

    print(f'Процент совпадения: {result["Процент совпадения"]}%\n')

    print("Детальное сравнение:")

    for doc, percent in result['Подробности'].items():

        print(f'- {doc}: {percent}%')

    save\_report(result)

    print("\nОтчет сохранен в файл plagiarism\_report.txt")

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**Расчетно-пояснительная записка**

**Общее описание программы**

Программа «Антиплагиат» предназначена для обработки текстовых документов и выявления заимствований. Она использует алгоритмы обработки естественного языка (NLP) и методы вычисления схожести текстов для точного анализа их оригинальности.

Приложение реализовано на языке Python с использованием библиотек re, hashlib, pandas, docx и других инструментов для анализа текста и вывода результатов.

**Основные модули**

* Модуль чтения файлов – загружает текстовые данные из файлов .txt и .docx.
* Модуль обработки текста – анализирует тексты и вычисляет коэффициент схожести.
* Главный модуль – управляет процессом анализа и выводит результаты.

**Структура функций**

* read\_text\_file(file\_path) – загружает текст из .txt файла.
* read\_docx\_file(file\_path) – извлекает текст из .docx документа.
* calculate\_similarity(text1, text2) – вычисляет коэффициент схожести двух текстов.
* save\_results\_to\_file(results, output\_file) – сохраняет результаты анализа в .csv файл.
* main() – объединяет все модули и управляет процессом обработки текста.

**Логика работы программы**

1. Пользователь загружает текстовый файл (.txt или .docx).
2. Программа сравнивает текст с другими файлами из определенной папки.
3. Вычисляется коэффициент схожести.
4. Результаты отображаются на экране и могут быть сохранены в файл.

**Техническое задание**

Полное техническое задание приведено в документе «Техническое задание.docx».

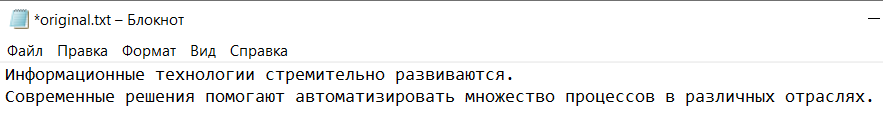
Руководство пользователя

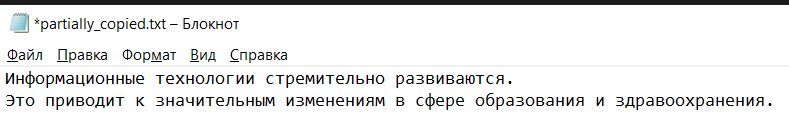
1. Выберите и загрузите файл (.txt или .docx).
2. Дождитесь завершения анализа.
3. Ознакомьтесь с результатами (коэффициент схожести и выделенные совпадения).

**Концептуальная модель предметной области**

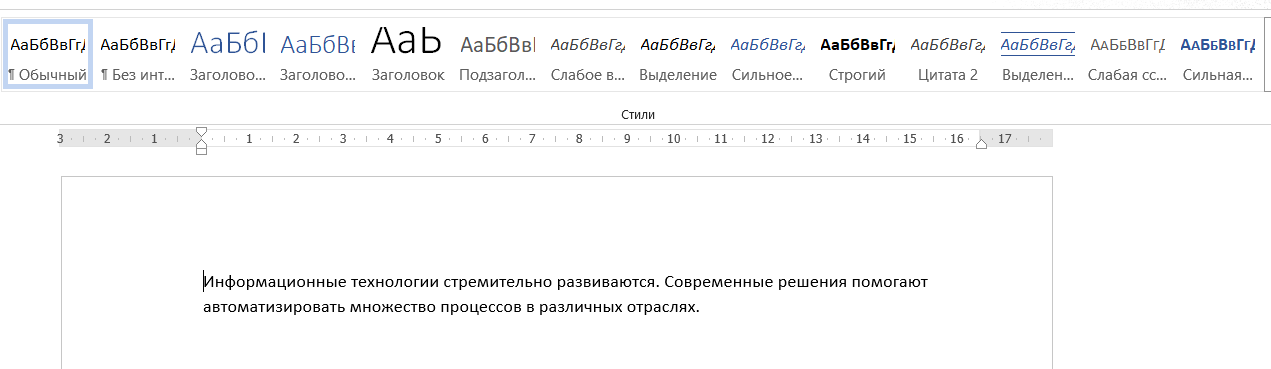
Начальные данные

Оригинальный текст в файле original.txt

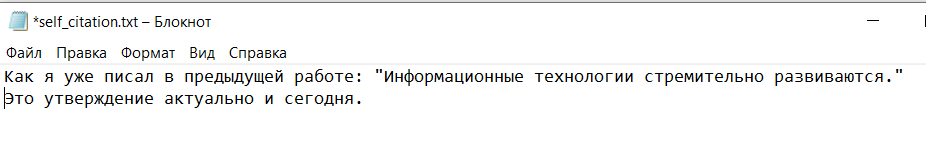


Частично скопированный текст в файле partially\_copied.txt

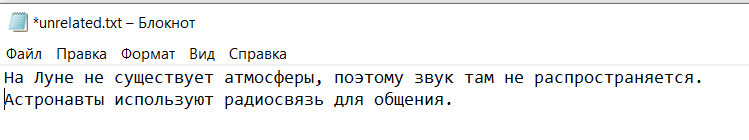
Полностью позаимствованный текст из файла full\_copy.docx



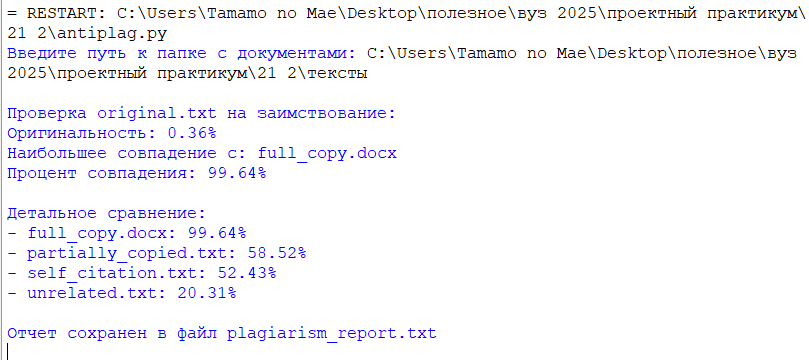
Цитирование из файла self\_citation.txt



Текст не относящийся к оригинальному тексту в файлеunrelated.txt



Проверим результат выполнения программы



Результат в файле

