**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**на курсовую работу**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема** «Информационная система преобразования текста из формата jpg в формат doc»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2361-01 ТЗ-08

Листов 38

Руководитель разработки:

к.т.н., доцент

*Шишкин В.В.*

« » 2023 г.

Исполнитель:

студент гр. ИСТбд-21

*Аглиуллин Р.Р.*

« » 2023 г.

**2023**

Содержание

Аннотация……...…………………………………………………………03

Техническое задание…………………………………………………...04

Пояснительная записка...……………………………………………...08

Руководство пользователя……………………………….…………...25

Текст программы…..…………………………………………………....41

**Аннотация**

Курсовая работа посвящена разработке и исследованию информационной системы, предназначенной для преобразования текстовой информации из формата изображения jpg в текстовый формат doc. Работа включает в себя анализ существующих методов распознавания текста на изображениях, а также проектирование и реализацию программного комплекса, способного автоматически извлекать текст из изображений и сохранять его в формате документа Microsoft Word. В процессе исследования уделяется внимание эффективности и точности распознавания, а также возможным областям применения полученной информации. Работа имеет практическую значимость для автоматизации процесса конвертации текстов из изображений в редактируемый формат документов.

Развитие современных технологий и увеличение объема цифровых данных требуют эффективных средств обработки информации. В работе рассматриваются основные методы распознавания текста в изображениях, а также принципы и алгоритмы преобразования изображений в текстовые документы.

Ключевые моменты исследования включают в себя анализ существующих подходов к оптическому распознаванию символов (OCR). В ходе работы реализован прототип информационной системы, позволяющий пользователям конвертировать текст из изображений в редактируемый формат документа (doc).

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

**на курсовую работу**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема** «Информационная система преобразования текста из формата jpg в формат doc»

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2361-01 ТЗ-02

Листов 4

Исполнитель:

студент гр. ИСТбд-21

*Аглиуллин Р.Р.*

« » 2023 г.

**2023**

**Введение**

Информационная система преобразования текста из формата jpg в формат doc.

Приложение должно соответствовать следующим требованиям, приведенным ниже.

Пользователи должны иметь возможность загружать изображения в формате JPG, которые содержат текст, в приложение. Приложение должно использовать технологию оптического распознавания символов (OCR), чтобы извлекать текстовую информацию из загруженных изображений. Распознанный текст должен быть преобразован в формат документа DOC (например, Microsoft Word) с возможностью редактирования.

Приложение должно предоставлять возможность сохранения сконвертированных документов в формате DOC и загрузки их позже для редактирования.

**1. Основания для разработки**

В качестве оснований для разработки учебный план направления 09.03.02 «Информационные системы и технологии» и распоряжение по факультету о закреплении тем курсовых работ (проектов) за студентами 2 курса ФИСТ направления 09.03.02 “Информационные системы и технологии” (профиль Информационные системы и технологии) по дисциплине “Алгоритмы и структуры данных”.

**2. Требования к программе или программному изделию**

**2.1. Функциональное назначение**

Требуется разработать однопользовательское десктопное приложение для автоматического преобразования текста из формата jpg в формат doc.

**2.2 Требования к функциональным характеристикам**

2.2.1 Требования к структуре приложения

Приложение должно быть разработано в виде одного модуля с дополнительными информационными файлами при необходимости.

2.2.2 Требования к составу функций приложения

В приложении должны быть реализованы в графическом режиме следующие основные функции:

- отрисовка интерфейса;

- взаимодействие с пользователем;

- прием изображения jpg;

- распознавание текста;

- преобразование в формат doc;

- сохранение и загрузка;

- информирование пользователя о результате работы программы.

2.2.2 Требования к организации информационного обеспечения, входных и выходных данных

В приложении должен быть реализован графический интерфейс взаимодействия с пользователем. Файлы формата jpg должны загружаться пользователем через графический интерфейс. Преобразованные файлы должны сохраняться в любом месте, на усмотрение пользователя, через графический интерфейс.

**2.3 Требования к надёжности**

Поддержка непрерывной и стабильной работы компьютера.

**2.4 Требования к информационной и программной совместимости**

Рекомендуется к использованию на Windows 7, 10.

При создании программы используются библиотеки “Pillow”, “pyTesseract” (Tesseract OCR), “tkinter”, “docx”.

Разработка ведётся в “PyCharm Community Edition 2023.2.3” на версии языка программирования Python 3.9.

**2.5 Требования к маркировке и упаковке**

Определяются заданием на курсовую работу.

**2.6 Требования к транспортированию и хранению**

2.6.1 Условия транспортирования

Требования к условиям транспортирования не предъявляются.

2.6 2 Условия хранения

Хранение производится в открытом доступе в отдельном репозитории на GitHub до окончания сроков хранения.

2.6 3 Сроки хранения

Срок хранения – до июля 2024 года**.**

**3. Требования к программной документации**

Определяются заданием на курсовую работу.

**4. Стадии и этапы разработки**

Определяются заданием на курсовую работу.

**5. Порядок контроля и приёмки**

Определяются заданием на курсовую работу.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

**По дисциплине «Алгоритмы и структуры данных»**

**Тема** «Информационная система преобразования текста из формата jpg в формат doc»

**Пояснительная записка**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2361-01 ПЗ-06

Листов 17

**Исполнитель**:

студент гр. ИСТбд-21

*Аглиуллин Р.Р.*

« » 2023 г.

**2023**

**Введение**

В ходе курсовой работы было разработано приложение под наименованием «Информационная система преобразования текста из формата jpg в формат doc». Условное обозначение - «Преобразование текста из формата jpg в формат doc».

Подход для реализации данного приложения, в общем виде:

* Загрузка изображения jpg;
* Распознавание текста с использованием библиотеки Tesseract OCR (PyTesseract);
* Сохранение обработанного текста в формате doc.

Для загрузки изображения в формате jpg используется диалоговое окно, сохраняющее путь к файлу и впоследствии открывающее его с помощью библиотеки Pillow, наиболее популярной для работы с растровой графикой. Так как нам требуется лишь держать jpg в памяти, эта библиотека является вполне исчерпывающей.

Для распознавания текста в данном приложении было выбрано использование библиотеки Tesseract OCR (PyTesseract), позволяющей распознавать текст на различных изображениях при помощи нейросетевых алгоритмов. Данная библиотека является наиболее удачной в силу большого количества документаций, написанных по ней, а также простоты ее использования в большинстве ситуаций. В совокупности с этими преимуществами Tesseract является достаточно точным инструментом для распознавания текста, имеющим возможность обучения.

Формат doc является одним из самых простых среди текстовых форматов, что позволяет открыть его на большинстве компьютеров.

Реализованное приложение представляет собой информационную систему преобразования текста из jpg в формат doc, включающую в себя:

* Интерфейс для взаимодействия с пользователем;
* Прием изображения в формате jpg;
* Распознавание текста;
* Преобразование в формат doc;
* Сохранение и загрузку файлов;
* Информирование пользователя о работе программы.

**1. Проектная часть**

**1.1 Постановка задачи на разработку приложения**

Определяется заданием на курсовую работу. Детализируется в разработанном техническом задании (приложение 1).

**1.2 Математические методы**

В качестве математической модели выступают два формата jpg и doc.

Изображение в формате jpg представляется как трехмерный массив пикселей, где каждый пиксель содержит информацию о цвете. Таким образом, Pillow принимает изображение. Обозначим изображение как I, размер изображения M \* N, где M – высота изображения, а N – ширина.

Тогда элемент изображения можно представить следующим образом:

I (i, j) = (R, G, B),

где:

i и j – координаты пикселя в изображении;

R, G, B – значения красного, зеленого и синего цветов соответственно.

Таким образом, каждый элемент массива I содержит тройку значений, представляющих цветовой спектр для данного пикселя. Эти значения лежат в диапазоне от 0 до 255 для каждого цвета в 8-битном представлении.

Но данное представление дает нам слишком много информации в случае распознавание и анализа текста на изображении, поэтому оно должно быть преобразовано в оттенки серого.

Для преобразования в оттенки серого применяется формула, учитывающая вклад каждого цветового канала в общую яркость.

Для каждого пикселя (i, j) цветного изображения I с компонентами (R, G, B) формула для преобразования в оттенки серого выглядит следующим образом:

Gray (i, j) = 0,299 \* R + 0,587 \* G + 0,114 \* B

Весовые коэффициенты перед компонентами соответствуют интенсивности каждого цвета. В результате мы получаем интенсивность серого для данного пикселя. Что проделывается для всего изображения.

Для того чтобы перейти к распознаванию текста необходимо провести дополнительную обработку изображения, а именно подразумевается его нормализация и очистка от зашумляющих элементов.

Формула для предварительной обработки выглядит следующим образом:

A (i, j) = Gray (i, j) \* R,

где R = {r1, r2, … ri} – множество грамматических правил, применяемых к обесцвеченному изображению.

Таким образом, мы получаем изображение размера M\*N, как и изначальное, но исключив всю избыточную и зашумляющую информацию.

Когда изображение обработано, мы переходим к сегментации. Сегментация представляет собой разделение изображения на однородные области. Существует множество методов сегментации. Метод на основе поиска компонент связности является одним из самых действенных в отношении изображений, содержащих текстовую информацию.

Сегментированное изображение задается парой – изображение и разметка на классы. S = (A (i, j), Q). Разметкой является матрица Q размерами M\*N. Ее элементы – числа (1, …, 8).

Введем понятие компоненты связности. Зададим граф смежности между пикселями с 8-связной системой соседства. Назовем окрестностью пикселя множество его соседей. Если один пиксель лежит в окрестности другого, то эти пиксели смежны. Пусть имеется множество пикселей V. V называется связным множеством, если любой пары пикселей p1, p2 найдется последовательность пикселей (p1, q1, q2, …, qm, p2), в которой пиксели (qk, qk+1), ∀k ∈ {1, 2, …, m - 1} и (p1, q1), (p1, qm) смежны.

Таким образом, множество Q распадается однозначным образом на множество непересекающихся связных множеств V1, …, Vn, которые называются компонентами связности множества Q.

Каждая компонента связности представляет из себя пару (B, M): бинарную маску Bn\*m и метку класса M. Получаем, что:

В итоге сегментированное изображение – это:

.

На практике сегментированное изображение может отличаться от некоей истинной сегментации для конкретного изображения.

Кроме того, стоит отметить другие возможные методы сегментации, выбор конкретного метода зависит от поставленной задачи, в некоторых случаях может потребоваться и совмещение нескольких методов.

Автоматические методы сегментации изображения:

По порогу (Thresholding):

Бинаризация изображения на основе яркости пикселей. Пиксели, превышающие порог, считаются одним классом, а остальные - другим.

Методы обнаружения границ (Edge Detection):

Использование операторов обнаружения границ, таких как Sobel, Canny, или Laplacian, для выделения контуров текста.

Методы кластеризации (Clustering):

Применение алгоритмов кластеризации, таких как k-средних, для группировки пикселей на изображении.

Семантическая сегментация (Semantic Segmentation):

Использование глубоких нейронных сетей, таких как U-Net, FCN (Fully Convolutional Network), для присвоения каждому пикселю на изображении определенного класса (текст, фон и т.д.).

Ручные методы сегментации изображения:

Ручная разметка (Manual Annotation):

Вручную выделение областей текста с использованием графических инструментов или программ для редактирования изображений.

Маркировка областей (Region Labeling):

Разметка областей, содержащих текст, с последующим использованием этой информации для выделения текста на изображении.

Интерактивная сегментация (Interactive Segmentation):

Применение методов сегментации с возможностью взаимодействия с пользователем, чтобы уточнить результаты.

Семантическая разметка (Semantic Labeling):

Ручное присвоение меток классов различным областям изображения (например, текст, фон, объекты) для обучения модели с учителем.

Выбор конкретного метода, как уже было сказано, зависит от характера данных, задачи и требований точности. В большинстве случаев эффективным подходом является комбинация нескольких методов для достижения оптимальных результатов. В нашем случае мы используем, предварительно обученную модель Tesseract. Она использует двухпроходный процесс сегментации. Во время первого прохода он пытается создать сегментацию на основе изображения, выделяя блоки текста, а затем разбивая блоки на строки и слова. Однако, если результаты неудовлетворительны, он переходит ко второму проходу, включающему в себя более тщательную сегментацию с использованием дополнительных методов, таких как адаптивная бинаризация и анализ границ.

Теперь с уже сегментированным изображением можно перейти к распознаванию. Распознавание – отнесение предъявляемых объектов к определенным классам с помощью применения известных правил классификации. Для того, чтобы система могла выполнять данный функционал, необходимо провести ее обучение, используя большую выборку примеров, представляющую обучающую выборку объектов распознавания.

Для распознавания символов текстовых изображений используется модель нейронной сети, в нашем случае Tesseract OCR. В общем виде, модель работает следующим образом.

Дано множество M объектов w. Объекты задаются значениями некоторых признаков xi, i = 1, …, N, наборы которых одинаковы для всех объектов. Совокупность признаков объекта w определяет, некоторым образом, его описание I(w) = (x1(w), x2(w), …, xN(w)). Признаки могут выражаться значениями из набора возможных вариантов языка .

На всем множестве M существует разбиение на подмножества (классы объектов):

Разбиение на классы может быть задано полностью или определяться некоторой априорной информацией о классах – например, характеристическим описанием входящих в них объектов.

Задача распознавания состоит в том, чтобы для каждого данного объекта w по его описанию I(w) и априорной (обучающей) информации вычислить значения предикатов

Процедура, строящая информационный вектор в данном случае, выражает алгоритм принятия решения об отнесении объекта w к тому или иному классу и называется «решающей функцией».

Альтернативой ручному вводу в систему описаний объектов и параметров решающей функции является обучение системы. Оно представляет собой процедуру самонастройки системы распознавания на основе воспринимаемой информации и может происходить как при подготовке системы, так и в процессе ее работы по мере «накопления опыта».

Вид обучения, в котором системе представляется набор образцов распознаваемых объектов с указанием их принадлежности с классам, называется обучением с учителем. Набор образцовых объектов называется обучающей выборкой. Необученная система производит распознавание предлагаемых объектов и сравнивает свои результаты с правильными ответами, поступающими от учителя. По результатам сравнения система корректирует параметры решающей функции.

Итак, обучающая выборка в задаче распознавания является априорной информацией о множестве распознаваемых объектов и представляет описание всех классов объектов:

Ее составляют описания предложенных учителем объектов с указанием их принадлежности классам, т.е. можно определить ее как совокупность описаний объектов:

,

где объекты принадлежат классу , объекты – классу и т.д.

Таким образом, обучающая выборка представляет собой таблицу, строки которой помечены названиями объектов , а столбцы – названиями признаков . Элементами таблицы являются значения признаков объектов Строки таблицы сгруппированы по классам

Для обучения сети необходимо иметь множество пар векторов {xs, ds}, s = 1 … S, где {xs}= {x1, …, xs} – множество входных векторов x, {ds} = {d1, …, ds} – множество эталонов выходных векторов. Совокупность пар {xs, ds} образует обучающее множество.

Количество элементов S в обучающем множестве должно быть достаточным для обучения сети, чтобы алгоритм смог сформировать набор параметров сети, выдающий в результате нужное отображение. Количество пар в обучающем множестве не имеет четких значений. Однако если элементов слишком много или мало, сеть не обучится и не решит поставленную задачу.

Для оценки качества обучения чаще всего выбирают суммарную квадратичную ошибку:

,

где j – число нейронов в выходном слое,

ds – известное решение задачи xs,

ys – решение, полученное на выходе обученной нейронной сети.

Данная оценка помогает подогнать входные данные алгоритма до оптимальных для повышения его качества.

Таким образом после того, как символы распознаются они объединяются в последовательность, представляющую текстовую строку, некоторый результат распознавания.

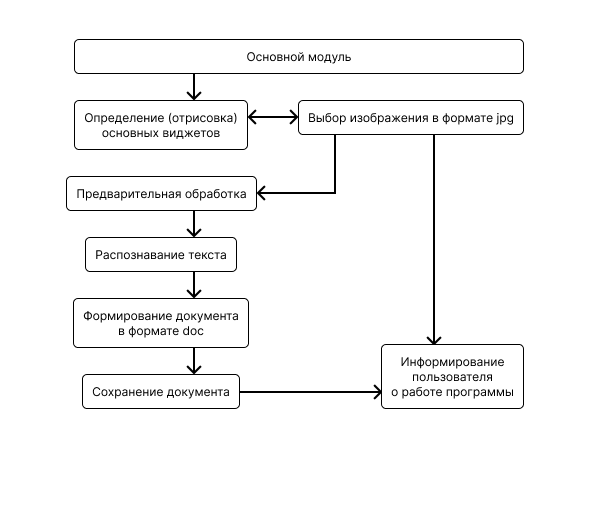
Из данной текстовый строки мы формируем DOC-документ. Для этого используется библиотека “python-docx”, представляющая средства для создания и редактирования документов.

Мы создаем пустой объект документа, в который добавляем текст с помощью метода add\_paragraph(), имеющего стандартное форматирование. После чего, мы можем сохранять документ в необходимом нам формате DOC.

**1.3 Архитектура и алгоритмы**

1.3.1. Архитектура

Приложение имеет следующую архитектуру.



1.3.2. Алгоритм отрисовки интерфейса

1. Создание окна приложения.
2. Добавление элементов управления, таких как кнопки для загрузки изображения, кнопки для запуска преобразования, поля для отображения текста и другие необходимые элементы интерфейса.
3. Настройка обработчиков событий для элементов управления, таких как кнопки загрузки и запуска преобразования.
4. Отображение интерфейса приложения.

1.3.3 Алгоритм взаимодействия с пользователем

1. Ожидание действий пользователя.
2. Обработка событий, таких как нажатие кнопок.
3. Выполнение соответствующих действий, таких как загрузка изображения, запуск распознавания текста, сохранение документа.
4. Вывод информации пользователю, включая результаты преобразования и сообщения об ошибках.

1.3.4 Алгоритм приема изображений JPEG

1. Ожидание действия пользователя для выбора изображения через файловый диалог и нажатия соответствующей кнопки.
2. Загрузка выбранного изображения в приложение.
3. Проверка формата изображения на соответствие JPEG.
4. Сохранение изображения внутри приложения.

1.3.5 Алгоритм распознавания текста

1. Применение методов OCR (Tesseract OCR) для распознавания текста в загруженном изображении.
2. Обработка распознанного текста.
3. Сохранение распознанного текста в некоторой переменной для дальнейшего преобразования в формат DOC.

1.3.6 Алгоритм преобразования в формат DOC

1. Получение распознанного текста внутри некоторой переменной.
2. Преобразование текста и его структуры в формат, совместимый с DOC.
3. Сохранение полученного документа внутри некоторой переменной.
   * 1. Алгоритм сохранения полученного документа

После получения итоговой переменной, хранящей полученный документ, происходит ее сохранение в формате DOC-документа в выбранное место на компьютере пользователя.

1.3.8 Алгоритм информирования пользователя о результате работы программы

Вывод сообщений об успешном или неуспешном преобразовании, а также о возможных ошибках, если они возникнут. Сообщения могут быть выведены на интерфейсе в виде диалоговых окон или текстовых уведомлений, а также в формате текстового поля внутри интерфейса.

* 1. **Тестирование**

Интеллектуальная карта приложения

****

1.4.1 Описание отчета о тестировании

В данном отчете представлены результаты тестирования программы на основе разработанных test-case и чек-листов, написанных unit-тестов и статического тестирования документации и программного кода. Описаны проведенные тесты, их результаты и обнаруженные дефекты.

1.4.2 Цель тестирования

Целью тестирования является проверка соответствия ПО предъявляемым требованиям, а также выявление возможных багов. По результатам тестирования следует исправление выявленных багов.

1.4.3 Методика тестирования

Тестирование проводилось с использованием следующих методов:

– Статическое тестирование: анализ и проверка кода без его запуска, выявление ошибок в технической документации;

– Ручное тестирование: запуск пользовательских сценариев программы с различными входными данными и проверка корректности полученных результатов.

– Автоматизированное тестирование: написание и запуск набора unit-тестов отдельных модулей программы.

1.4.4 Проведенные тесты

В ходе тестирования были проведены следующие тесты:

1) Ручное тестирование

Написаны и проведены следующие тест-кейсы и чек-листы:

TK1. Выбор файла

Шаги:

1. Запустить приложение

2. Нажать на кнопку «Выбрать».

3. В открывшемся диалоговом окне выбрать файл.

Ожидаемый результат:

Сохранение пути файла в памяти ПК и появления сообщения “Выбрано: \*путь\_к\_файлу.jpg\*”.

Фактический результат:

Путь к файлу сохранился. На экране появилось сообщение “Выбрано: \*путь\_к\_файлу.jpg\*”.

ТК2. Замена выбранного файла

1. Запустить приложение

2. Нажать на кнопку «Выбрать».

3. В открывшемся диалоговом окне выбрать файл.

4. Снова нажать на кнопку «Выбрать».

5. В открывшемся диалоговом окне выбрать другой файл.

Ожидаемый результат:

Сохранение пути файла в памяти ПК и появления сообщения “Выбрано: \*путь\_к\_файлу\_1.jpg\*”. После чего замена сохраненного пути файла на другой и появление нового сообщения “Выбрано: \*путь\_к\_файлу\_2.jpg\*”.

Фактический результат:

Путь к файлу сохранился и появилось сообщение “Выбрано: \*путь\_к\_файлу\_1.jpg\*”. После чего заменился сохраненный путь к файлу на другой и появилось новое сообщение “Выбрано: \*путь\_к\_файлу\_2.jpg\*”.

ТК3. Отмена выбора файла

1. Запустить приложение

2. Нажать на кнопку «Выбрать».

3. В открывшемся диалоговом окне, ничего не выбирая, закрыть его.

Ожидаемый результат:

Появление сообщения о том, что файл не выбран.

Фактический результат:

Появление сообщения о том, что файл не выбран.

ТК4. Преобразование текста из изображения в doc

1. Запустить приложение

2. Нажать на кнопку «Выбрать».

3. В открывшемся диалоговом окне выбрать файл.

4. Нажать на кнопку «Преобразовать».

5. В открывшемся диалоговом окне выбрать место сохранения и название файла.

Ожидаемый результат:

Преобразование выбранного файла в текстовый документ формата doc с возможностью редактирования, а также появление сообщения о том, что результат сохранен в \*путь\_к\_файлу.doc\*

Фактический результат:

Преобразование выбранного файла в текстовый документ формата doc с возможностью редактирования, а также появление сообщения о том, что результат сохранен в \*путь\_к\_файлу.doc\*

TK5. Отмена преобразования текста из изображения в doc

1. Запустить приложение

2. Нажать на кнопку «Выбрать».

3. В открывшемся диалоговом окне выбрать файл.

4. Нажать на кнопку «Преобразовать».

5. В открывшемся диалоговом окне не выбирать место сохранения и название файла, а просто закрыть его.

Ожидаемый результат:

Сообщение не должно измениться, все так же должно быть написано о том, какой файл выбран и по какому пути, не должны выходить различного рода ошибки.

Фактический результат:

Сообщение не изменилось, ошибок не возникло.

Чек-лист для проверки распознавания текста в различных файлах

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Описание | Пример | | Ожидание | | Результат | |
| Проверка преобразования и распознавания текста различных файлов | | | | | | |
| Файл №1 | | Простой текстовый файл с одним абзацем | | Сообщение об успешном сохранении файла и в целом достаточно точная работа распознавания. | | Сообщение об успешном сохранении файла и в целом достаточно точная работа распознавания. |
| Файл №2 | | Текстовый файл с несколькими абзацами | |  | |
| Файл №3 | | Текстовый файл со сложной структурой, но последовательной | |  | |
| Файл №4 | | Скан документа со сложной структурой | | Сообщение об успешном сохранении файла и в целом достаточно точная работа распознавания. | | Сообщение об успешном сохранении файла, но плохая работа распознавания. |
| Файл №5 | | Фото без текста | | Сообщение об успешном сохранении файла, пустота в файле. | | Сообщение об успешном сохранении файла, мусор в файле. |

Автоматизированное тестирование

Для курсовой работы было составлено 4 unit-теста. Тесты покрывают следующие функции приложения: browse\_file() (проверка для выбора файла, а также в случае его не выбора), convert\_to\_doc() (проверка для перевода файла в формат документа и в случае отмены его сохранения).

Все unit-тесты прошли успешную проверку.

1.4.5 Выводы

На основе проведенных тестов сделаны следующие выводы:

– Программа успешно прошла все тесты и работает корректно.

– Обнаружены и исправлены следующие дефекты: дефектов нет.

– Рекомендации по дальнейшему развитию программы: добавление возможности ручной сегментации изображения, улучшение стандартных методов сегментации, работа с фильтрами для повышения точности.

**2. Источники, использованные при разработке**

1. К.А. Бобров, В.Д. Шульман, К.П. Власов. АНАЛИЗ ТЕХНОЛОГИЙ РАСПОЗНАВАНИЯ ТЕКСТА ИЗ ИЗОБРАЖЕНИЯ // Международный журнал гуманитарных и естественных наук. 2022. №3-2. URL: https://cyberleninka.ru/article/n/analiz-tehnologiy-raspoznavaniya-teksta-iz-izobrazheniya (дата обращения: 10.12.2023).

2. Матушев А.А., Лобанов Ф.Н. Технологии перевода технической документации с бумажных носителей в электронный вид // БРНИ. 2014. №2 (11). URL: https://cyberleninka.ru/article/n/tehnologii-perevoda-tehnicheskoy-dokumentatsii-s-bumazhnyh-nositeley-v-elektronnyy-vid (дата обращения: 10.12.2023).

3. Гуськова, А. М. Математическое моделирование систем распознавания изображений, содержащих текстовую информацию, на основе нейронных сетей / А. М. Гуськова. — Текст : непосредственный // Молодой ученый. — 2015. — № 18 (98). — С. 7-10. — URL: https://moluch.ru/archive/98/21912/ (дата обращения: 16.12.2023).

4. Койнова, Т. А. Алгоритмы распознавания символов / Т. А. Койнова. — Текст : непосредственный // Молодой ученый. — 2022. — № 18 (413). — С. 73-76. — URL: https://moluch.ru/archive/413/91060/ (дата обращения: 16.12.2023).

5. Шагалова Полина Анатольевна, Ерофеева Анастасия Дмитриевна, Орлова Марина Михайловна, Чистякова Юлия Сергеевна, Соколова Элеонора Станиславовна. Исследование алгоритмов предобработки изображений для повышения эффективности распознавания медицинских снимков // Труды НГТУ им. Р. Е. Алексеева. 2020. №1 (128). URL: https://cyberleninka.ru/article/n/issledovanie-algoritmov-predobrabotki-izobrazheniy-dlya-povysheniya-effektivnosti-raspoznavaniya-meditsinskih-snimkov (дата обращения: 10.12.2023).

6. Болотова Юлия Александровна, Спицын Владимир Григорьевич, Осина Полина Максимовна. Обзор алгоритмов детектирования текстовых областей на изображениях и видеозаписях // КО. 2017. №3. URL: https://cyberleninka.ru/article/n/obzor-algoritmov-detektirovaniya-tekstovyh-oblastey-na-izobrazheniyah-i-videozapisyah (дата обращения: 10.12.2023).

7. Годунов Анатолий Иванович, Баланян Сергей Товмасович, Егоров Павел Сергеевич. СЕГМЕНТАЦИЯ ИЗОБРАЖЕНИЙ И РАСПОЗНАВАНИЕ ОБЪЕКТОВ НА ОСНОВЕ ТЕХНОЛОГИИ СВЕРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ // НиКСС. 2021. №3 (35). URL: https://cyberleninka.ru/article/n/segmentatsiya-izobrazheniy-i-raspoznavanie-obektov-na-osnove-tehnologii-svertochnyh-neyronnyh-setey (дата обращения: 10.12.2023).

8. Сегментация изображения текста. – [Электронный ресурс]. – Режим доступа: http://mechanoid.su/cv-text-image-segmentator.html (дата обращения: 10.12.2023).

9. Сегментация текстовых строк документов на символы с помощью сверточных и рекуррентных нейронных сетей. – [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/smartengines/articles/328000 (дата обращения: 10.12.2023).

10. Предобработка изображений для OCR. – [Электронный ресурс]. – Режим доступа: https://vc.ru/u/1627433-nazhmutdin-gumuev/696910-predobrabotka-izobrazheniy-dlya-ocr?ysclid=lq4lcrf76x414322535 (дата обращения: 10.12.2023).

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Кафедра «Измерительно-вычислительные комплексы»

**Курсовая работа**

**По дисциплине «Алгоритмы и структуры данных»**

**Тема** «Информационная система преобразования текста из формата jpg в формат doc»

**Руководство программиста**

**Инв. № подл.**

**Подп. и дата**

**Взам. инв. №**

**Инв. № дубл.**

**Подп. и дата**

Р.02069337.22/2361-01 РП-03

Листов 12

Исполнитель:

студент гр. ИСТбд-21

*Аглиуллин Р.Р.*

« » 2023 г.

**2023**

**1. Назначение и условия применения программы**

**1.1 Назначение и функции, выполняемые приложением**

Однопользовательское десктопное приложение для автоматического преобразования текста из формата jpg в формат doc.

Реализованное приложение представляет собой информационную систему преобразования текста из jpg в формат doc, включающую в себя:

* Интерфейс для взаимодействия с пользователем;
* Прием изображения в формате jpg;
* Распознавание текста;
* Преобразование в формат doc;
* Сохранение и загрузку файлов;
* Информирование пользователя о работе программы.

**1.2 Условия, необходимые для использования приложения**

Рекомендуется к использованию на Windows 7, 10.

При создании программы используются библиотеки “Pillow”, “pyTesseract” (Tesseract OCR), “tkinter”, “docx”.

Разработка ведётся в “PyCharm Community Edition 2023.2.3” на версии языка программирования Python 3.9.

**2. Характеристики программы**

**2.1 Характеристики приложения**

Количество значимых строк кода: 59 строк

Используемые библиотеки:

* sys: для работы с системными параметрами.
* os: для взаимодействия с операционной системой.
* tkinter: для создания графического интерфейса.
* filedialog из tkinter: для диалогового окна выбора файла.
* PIL (Python Imaging Library): для работы с изображениями.
* pytesseract: для распознавания текста на изображении.
* docx из python-docx: для создания и редактирования файлов формата Word.

Структуры данных:

Класс JpegToDocConverter с методами \_\_init\_\_, create\_widgets, browse\_file, convert\_to\_doc, preprocess\_image.

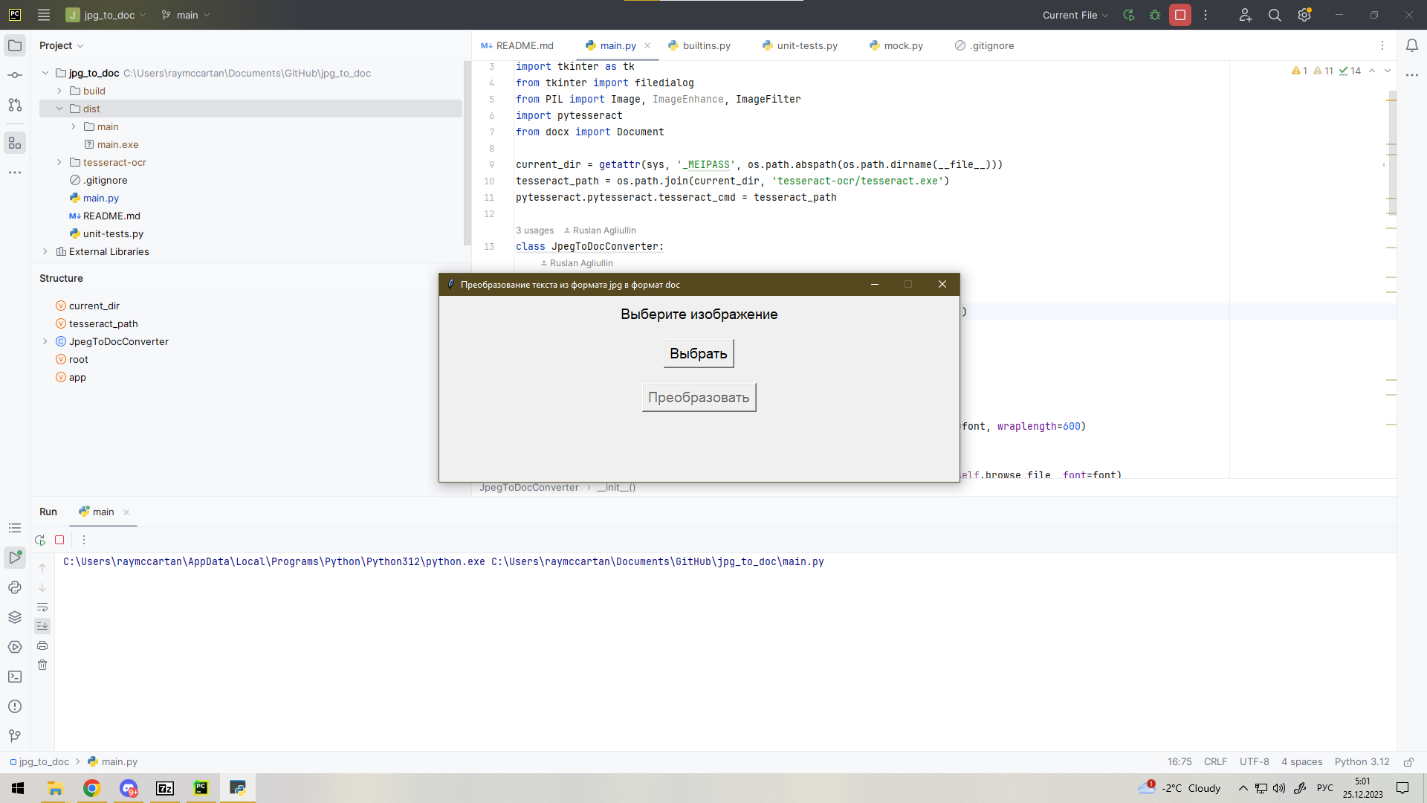
Используется переменная file\_path для хранения пути к выбранному файлу.

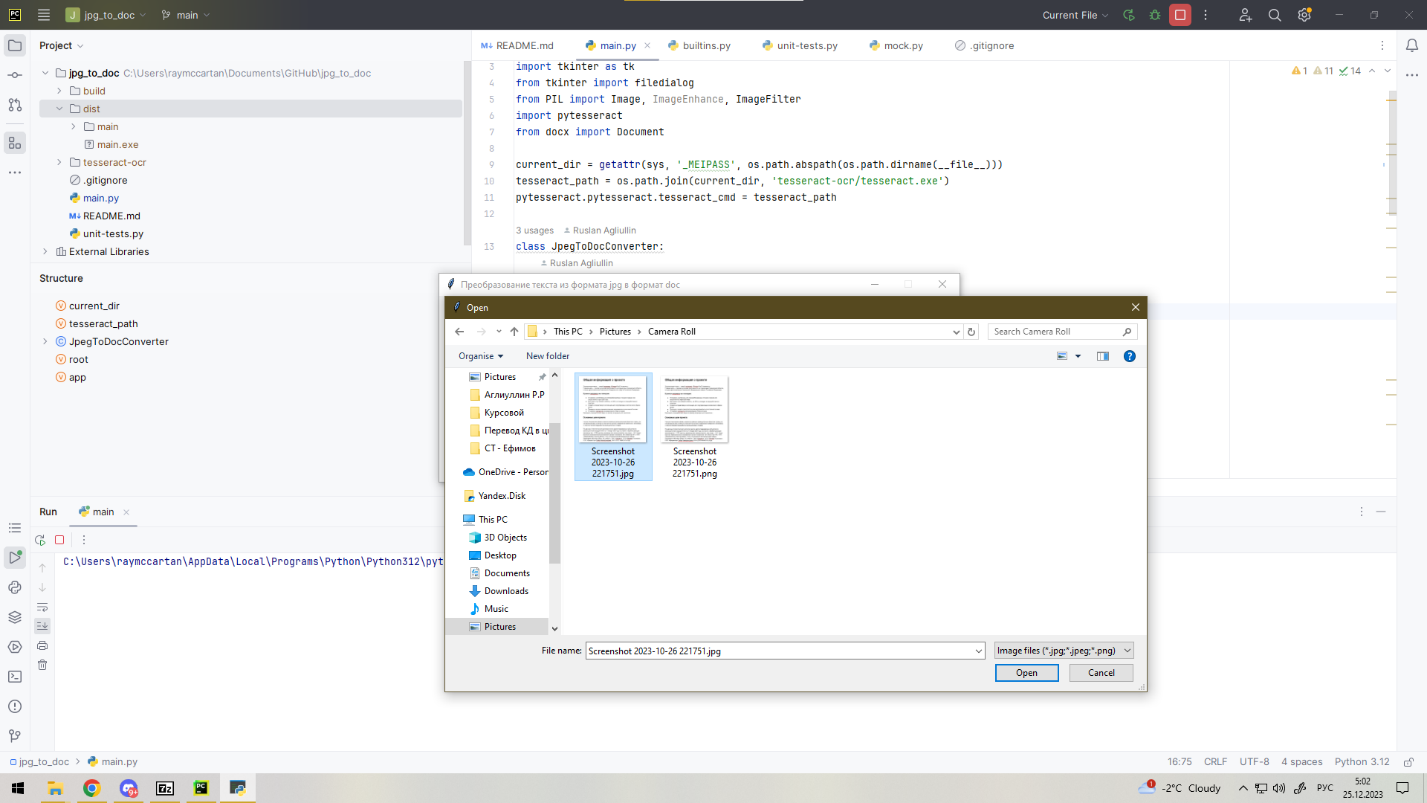
Используется флаг file\_saved для отслеживания сохранения результата.

Используется 7 алгоритмов, а именно отрисовка интерфейса, взаимодействия с пользователем, приема изображения, распознавания текста, преобразования в формат doc, сохранения полученного документа, информирования пользователя о результате работы программы.

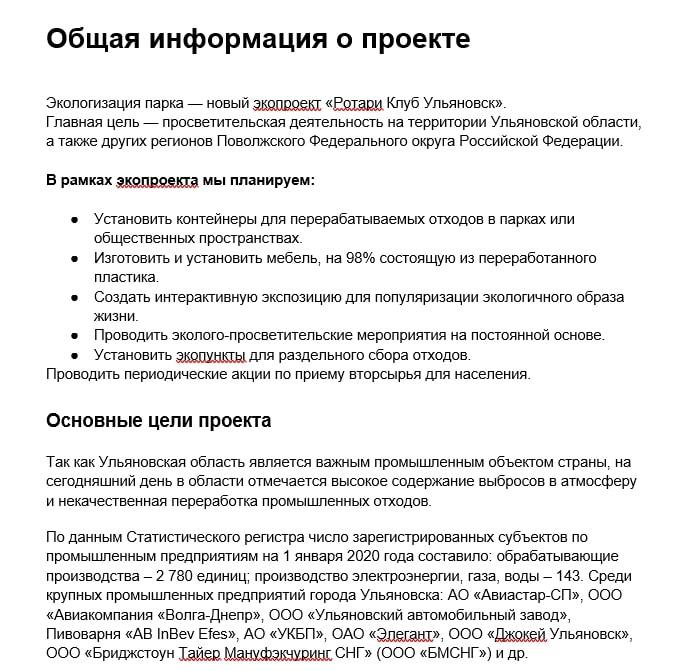
Работа приложения:

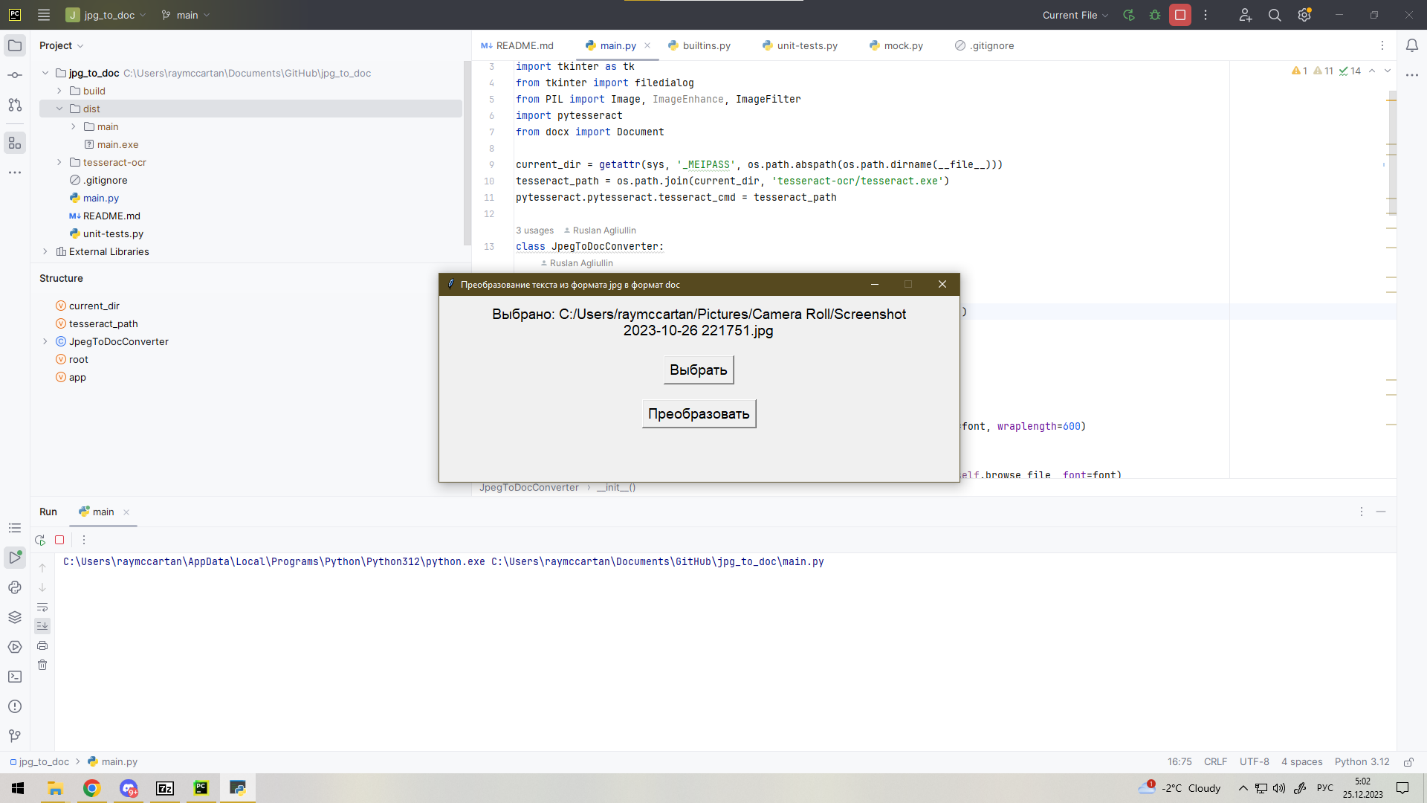
Пользователь выбирает изображение через диалоговое окно.





Ниже представлено изображение, загружаемое в программу.



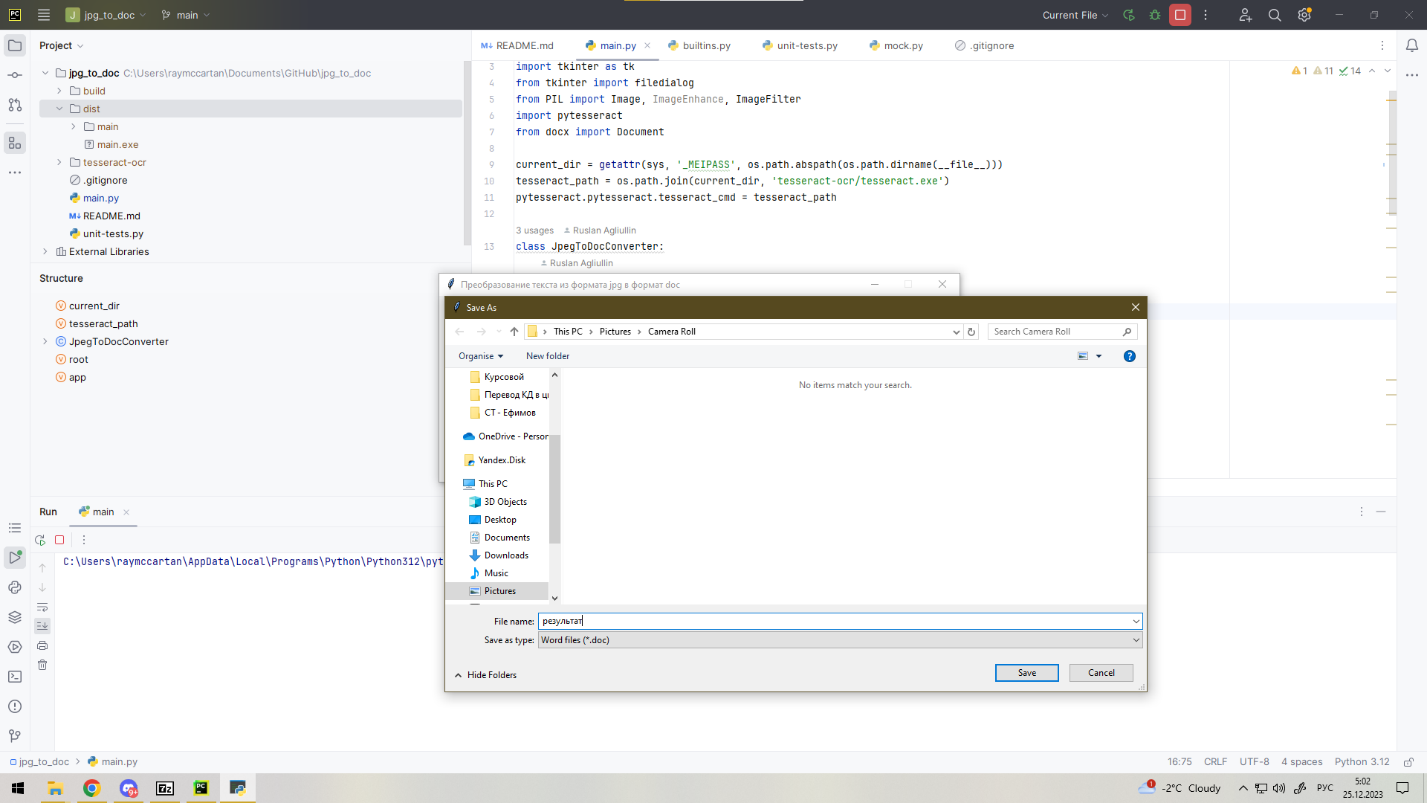


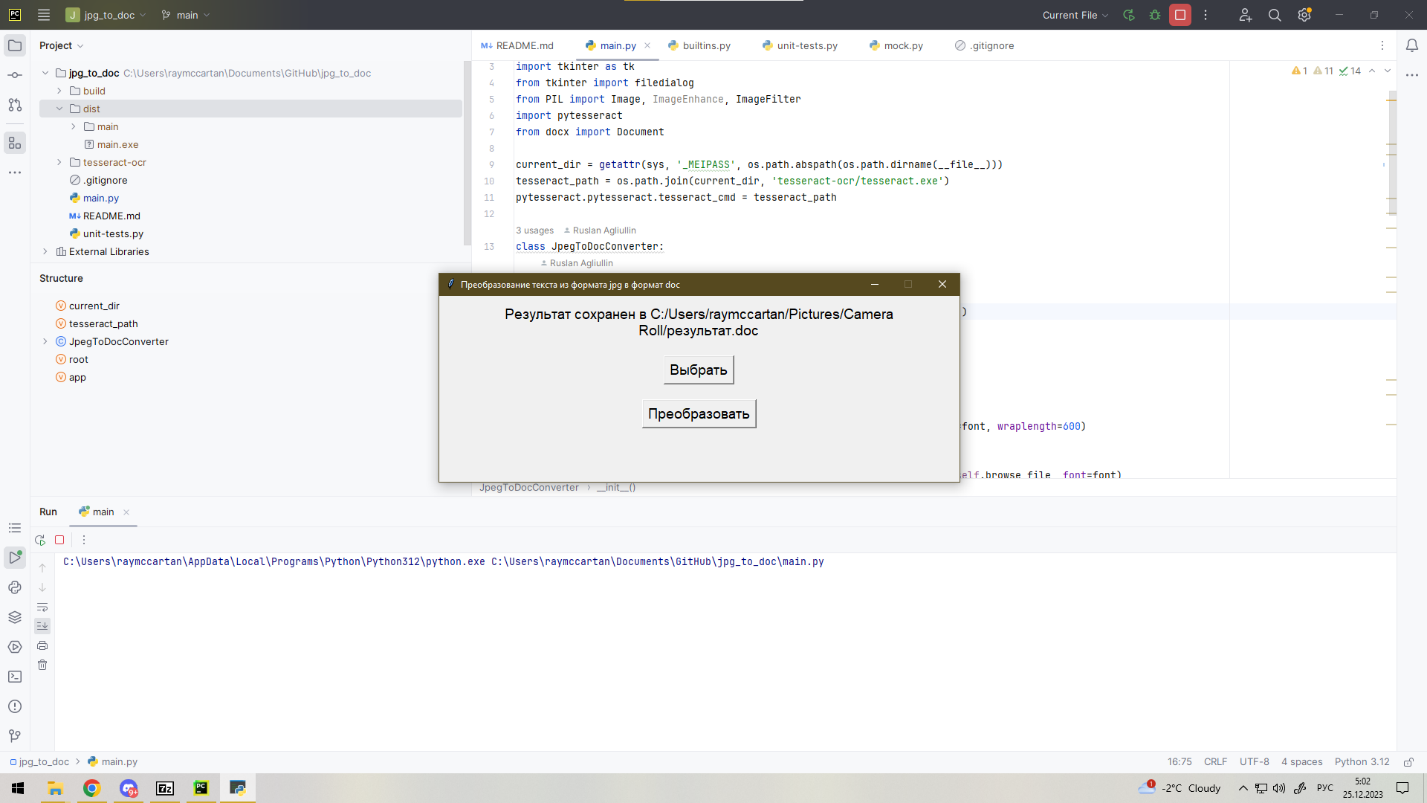
Изображение подвергается предварительной обработке (преобразование в оттенки серого и размытие).

Текст изображения распознается с использованием Tesseract OCR.

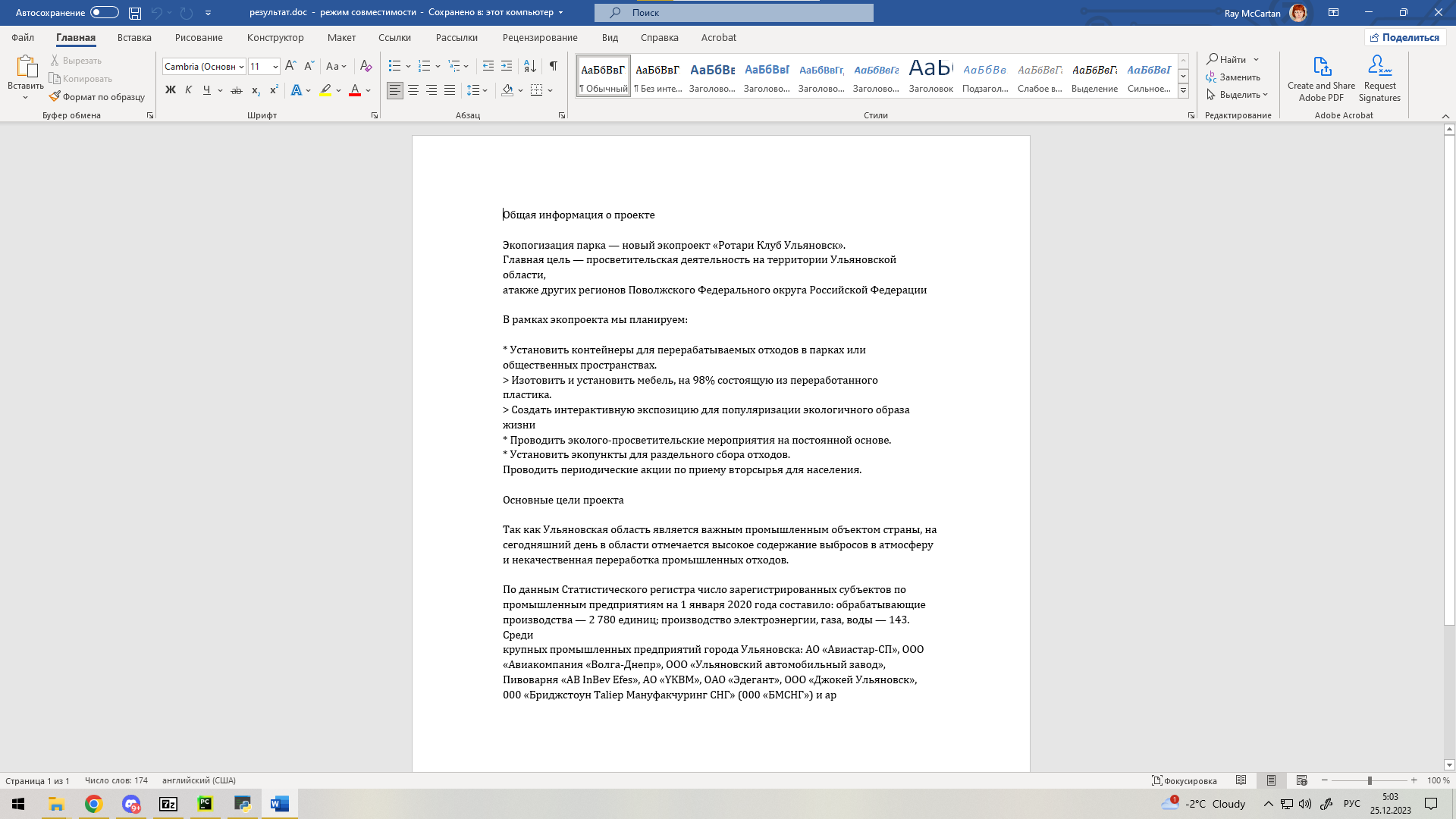
Распознанный текст добавляется в новый документ Word.

Пользователю предоставляется возможность выбрать путь для сохранения документа Word.





Полученный результат распознавания:



Интерфейс приложения представляет:

Окно с текстом для информирования пользователя о том или ином результате, а также представления подсказок.

Кнопка "Выбрать" для выбора изображения.

Кнопка "Преобразовать" для запуска процесса преобразования и сохранения.

Средства контроля корректности ввода/вывода:

Используется диалоговое окно для выбора изображения и сохранения документа.

Кнопка "Преобразовать" блокируется до выбора файла.

Выводится информация о результате каждой операции.

Общая реализация компиляции программы:

Библиотеки:

* sys: Предоставляет функции для взаимодействия с интерпретатором Python.
* os: Предоставляет функции для взаимодействия с операционной системой.
* tkinter: Библиотека для создания графического интерфейса пользователя (GUI).
* filedialog из tkinter: Позволяет создавать диалоговые окна для выбора файлов.
* PIL (Pillow): Библиотека для работы с изображениями.
* pytesseract: Интерфейс к Tesseract OCR для распознавания текста на изображениях.
* docx: Библиотека для создания и редактирования файлов формата Word (.docx).

Инициализация переменных и настройка Tesseract OCR:

* Определяется текущий каталог (current\_dir) с использованием атрибута \_MEIPASS для поддержки запуска программы из запакованного исполняемого файла.
* Формируется путь к исполняемому файлу Tesseract OCR.

Класс JpegToDocConverter:

* Инициализирует GUI с использованием библиотеки tkinter.
* Создает виджеты (метку, кнопки) для выбора файла и преобразования изображения в документ.
* Реализует методы browse\_file для выбора файла и convert\_to\_doc для преобразования изображения в документ.

Методы класса:

* browse\_file: Открывает диалоговое окно выбора файла с поддержкой изображений. При выборе файла обновляет метку и активирует кнопку преобразования.
* convert\_to\_doc: Преобразует изображение в текст, затем создает документ Word (.doc) и сохраняет результат.

Создание экземпляра класса и запуск GUI:

* Создается экземпляр класса JpegToDocConverter с заданным размером окна и запускается GUI.

Программа предназначена для выбора изображения, преобразования его в текст с использованием Tesseract OCR, и сохранения результата в файл Word (.doc).

Для компиляции данной программы с использованием PyInstaller, выполняем следующие шаги:

Выполняем следующую команду PyInstaller для компиляции программы:

pyinstaller main.py

Эта команда создаст каталог dist в нашем текущем рабочем каталоге, а внутри него будет находиться скомпилированный исполняемый файл.

Для корректной работы программы также переносим каталог tesseract-ocr в каталог \_internal, который создается вместе с каталогом dist.

Таким образом, программа запускается при помощи main.exe с использованием каталога \_internal.

**2.2 Особенности реализации приложения**

В данной программе используются следующие структуры данных:

Массивы и строки:

В коде присутствуют строки (например, 'Выберите изображение', 'Выбрано: {file\_path}', и т.д.), но они представляют собой базовые строки и не требуют особых структур данных для их хранения.

Объекты и классы:

Используется класс JpegToDocConverter для организации GUI-интерфейса и функционала программы. Внутри класса хранятся переменные, такие как self.root, self.label, self.browse\_button, self.convert\_button, self.file\_saved, и т.д.

Другие структуры данных:

Используется объект Image из библиотеки PIL (Pillow) для представления изображения.

Для взаимодействия с файловой системой используются функции из модулей os и filedialog.

Используется класс Document из библиотеки python-docx для создания и работы с документами Word.

* JpegToDocConverter - класс, представляющий GUI приложение.
* create\_widgets - метод для создания виджетов интерфейса.
* browse\_file - метод для выбора изображения через диалоговое окно.
* convert\_to\_doc - метод для конвертации изображения в текст и сохранения результата в файл формата docx.
* preprocess\_image - метод для предварительной обработки изображения перед распознаванием текста.

**Обоснование выбора:**

Tkinter (GUI): Tkinter - это стандартная библиотека для создания графического интерфейса в Python. Она проста в использовании и хорошо интегрирована с языком.

PIL (Pillow): Используется для работы с изображениями. Pillow предоставляет простой и удобный интерфейс для манипуляций с изображениями.

pytesseract: Библиотека для взаимодействия с Tesseract OCR, предоставляющая возможность распознавания текста на изображениях.

python-docx: Используется для создания и работы с документами Word. Позволяет легко добавлять содержимое в документ.

**Возможные альтернативы:**

Tkinter:

Возможно использование других библиотек для создания GUI, таких как PyQt или Kivy, но Tkinter обычно является стандартным и легким выбором.

Pillow:

Возможно использование других библиотек для работы с изображениями, таких как OpenCV, но Pillow обладает простым синтаксисом и хорошо подходит для базовых манипуляций с изображениями.

pytesseract:

Если требуется более сложная обработка изображений, можно рассмотреть использование других библиотек, таких как OpenCV.

python-docx:

Для более сложных задач работы с документами Word можно рассмотреть использование библиотеки python-docx-template.

Выбор данных библиотек и структур данных обусловлен простотой использования, легкостью интеграции и достаточным функционалом для поставленной задачи.

**3. Обращение к программе**

В программе используются следующие алгоритмы.

3.1. Алгоритм отрисовки интерфейса

1. Создание окна приложения.
2. Добавление элементов управления, таких как кнопки для загрузки изображения, кнопки для запуска преобразования, поля для отображения текста и другие необходимые элементы интерфейса.
3. Настройка обработчиков событий для элементов управления, таких как кнопки загрузки и запуска преобразования.
4. Отображение интерфейса приложения.

3.2 Алгоритм взаимодействия с пользователем

1. Ожидание действий пользователя.
2. Обработка событий, таких как нажатие кнопок.
3. Выполнение соответствующих действий, таких как загрузка изображения, запуск распознавания текста, сохранение документа.
4. Вывод информации пользователю, включая результаты преобразования и сообщения об ошибках.

3.3 Алгоритм приема изображений JPEG

1. Ожидание действия пользователя для выбора изображения через файловый диалог и нажатия соответствующей кнопки.
2. Загрузка выбранного изображения в приложение.
3. Проверка формата изображения на соответствие JPEG.
4. Сохранение изображения внутри приложения.

3.4 Алгоритм распознавания текста

1. Применение методов OCR (Tesseract OCR) для распознавания текста в загруженном изображении.
2. Обработка распознанного текста.
3. Сохранение распознанного текста в некоторой переменной для дальнейшего преобразования в формат DOC.

3.5 Алгоритм преобразования в формат DOC

1. Получение распознанного текста внутри некоторой переменной.
2. Преобразование текста и его структуры в формат, совместимый с DOC.
3. Сохранение полученного документа внутри некоторой переменной.

3.6 Алгоритм сохранения полученного документа

После получения итоговой переменной, хранящей полученный документ, происходит ее сохранение в формате DOC-документа в выбранное место на компьютере пользователя.

3.7 Алгоритм информирования пользователя о результате работы программы

Вывод сообщений об успешном или неуспешном преобразовании, а также о возможных ошибках, если они возникнут. Сообщения могут быть выведены на интерфейсе в виде диалоговых окон или текстовых уведомлений, а также в формате текстового поля внутри интерфейса.

Реализация данных алгоритмов возможна при помощи следующих библиотек.

sys и os:

sys используется для доступа к некоторым переменным и функциям, специфичным для интерпретатора Python. В данном коде sys используется для определения текущего рабочего каталога и объединения путей.

os используется для взаимодействия с операционной системой, таким образом, программа может правильно находить пути к файлам и директориям, независимо от того, запущена она из исходного кода или скомпилированного исполняемого файла.

tkinter и filedialog:

tkinter используется для создания графического интерфейса пользователя (GUI). В данной программе используется для создания окна приложения, кнопок, меток и диалогового окна выбора файла.

filedialog является модулем внутри tkinter и используется для создания диалоговых окон для выбора файлов. В данной программе filedialog используется для выбора изображения и сохранения результирующего файла.

* Tk() - создает основное окно приложения.
* Label - создает текстовый или изображенческий виджет.
* Button - создает кнопку, реагирующую на события мыши.
* filedialog - предоставляет диалоговые окна для выбора и сохранения файлов.

PIL (Pillow):

Pillow (PIL fork) - это библиотека для работы с изображениями. В данной программе Image из Pillow используется для открытия, обработки и сохранения изображений.

* Image - класс из библиотеки Pillow, предоставляющей базовые функции обработки изображений.

Основные методы и возможности класса Image включают в себя:

1. Image.open(file) - открывает изображение из файла.
2. Image.save(file) - сохраняет изображение в файл.
3. Image.convert(mode) - изменяет режим цветности изображения (например, в черно-белый).
4. Image.filter(filter) - применяет фильтр к изображению.
5. Image.crop(box) - обрезает изображение по указанной области.
6. Image.resize(size) - изменяет размер изображения.
7. Image.rotate(angle) - поворачивает изображение на указанный угол.

Таким образом, Image из Pillow предоставляет широкий спектр функциональности для работы с изображениями, что полезно при предварительной обработке изображений в задачах, таких как распознавание текста.

* ImageEnhance - класс, предоставляющий методы для улучшения изображений.

Он предоставляет классы для улучшения различных характеристик изображения, таких как яркость, контраст и т. д.

Примеры классов из ImageEnhance включают:

1. ImageEnhance.Brightness - улучшение яркости.
2. ImageEnhance.Contrast - улучшение контраста.
3. ImageEnhance.Color - улучшение насыщенности цвета.
4. ImageEnhance.Sharpness - улучшение резкости.

Для использования этих классов, вы создаете экземпляр класса и вызываете метод enhance(factor), где factor - это коэффициент улучшения.

* ImageFilter - класс для применения различных фильтров к изображениям.

Примеры фильтров, доступных в ImageFilter:

1. BLUR: Размытие изображения.
2. CONTOUR: Подчеркивание контуров объектов.
3. DETAIL: Увеличение деталей на изображении.
4. EDGE\_ENHANCE: Улучшение рёбер.
5. EMBOSS: Выделение выпуклых и вогнутых частей изображения.
6. FIND\_EDGES: Поиск границ объектов.
7. SHARPEN: Улучшение резкости.

ImageEnhance и ImageFilter используются для улучшения и фильтрации изображения соответственно. В данной программе ImageFilter.GaussianBlur применяется для размытия изображения.

pytesseract:

pytesseract используется для взаимодействия с Tesseract OCR (Optical Character Recognition). В данной программе pytesseract применяется для извлечения текста с предварительно обработанного изображения.

text = pytesseract.image\_to\_string(preprocessed\_image, lang='rus+eng')

Этот код выполняет следующее:

1. preprocessed\_image - это изображение, которое было предварительно обработано с использованием метода preprocess\_image. В данном случае, изображение было конвертировано в черно-белый формат и применен фильтр размытия.
2. lang='rus+eng' - указывает языковые параметры для распознавания текста. В данном случае, используются языки "русский" и "английский". Это важно, чтобы Tesseract мог правильно интерпретировать символы на изображении.
3. pytesseract.image\_to\_string - это метод, который используется для распознавания текста на изображении. Ему передается предварительно обработанное изображение (preprocessed\_image), и он возвращает строку, содержащую распознанный текст.

Краткое описание основных внутренних механизмов Tesseract:

Бинаризация: Преобразование изображения в черно-белый формат, где каждый пиксель либо черный, либо белый. Это упрощает дальнейшую обработку.

Фильтры: Применение различных фильтров для улучшения качества изображения, уменьшения шума и повышения контраста.

Разбиение изображения: Выделение областей, которые могут содержать текст. Это может быть выполнено с использованием методов компонент связности или других техник.

Машинное обучение: Tesseract использует обученные модели для распознавания отдельных символов. Эти модели обучаются на больших наборах данных, содержащих изображения символов и соответствующие им текстовые метки.

Контекстная информация: Tesseract также учитывает контекст и взаимное расположение символов для повышения точности распознавания.

Языковые модели: Tesseract поддерживает множество языков, и для каждого языка может быть использована своя модель для учета особенностей этого языка.

Словари: Для уточнения результатов Tesseract может использовать словари со словами и частотными данными.

Коррекция ошибок: После распознавания текста Tesseract может попытаться исправить возможные ошибки с использованием словарей или других методов.

Текстовый вывод: Результаты распознавания текста возвращаются в виде текстовой строки.

Таким образом, весь этот блок кода берет изображение, предварительно обрабатывает его, затем использует Tesseract OCR для извлечения текста. Полученный текст затем используется для создания нового документа Word (docx.Document) в методе convert\_to\_doc.

docx из библиотеки python-docx:

python-docx используется для создания и работы с документами Microsoft Word в формате .doc. В данной программе Document из этой библиотеки используется для создания нового документа и добавления извлеченного текста.

Document - класс из библиотеки python-docx, предназначенной для создания и редактирования файлов формата docx (Microsoft Word).

getattr(sys, '\_MEIPASS', os.path.abspath(os.path.dirname(\_\_file\_\_))) - используется для определения текущего рабочего каталога, особенно в среде, где приложение может быть упаковано (как исполняемый файл).

**4. Сообщения**

В коде сообщения об ошибках/результатах контроля корректности ввода/вывода могут быть следующими:

При выборе файла:

* "Файл не выбран" - если пользователь не выбрал файл при нажатии кнопки "Выбрать".
* "Выбрано: [путь\_к\_файлу]" - если файл успешно выбран.

При преобразовании в документ:

* "Выберите изображение" - если пользователь не выбрал файл изображения перед нажатием кнопки "Преобразовать".
* "Результат сохранен в [путь\_к\_файлу]" - если преобразование в документ выполнено успешно и файл сохранен.

В коде также предусмотрено отключение кнопки "Преобразовать" до момента выбора файла и включение ее только после успешного выбора файла.

**Текст программы**

Исходный код приложения «Информационная система преобразования текста из формата jpg в формат doc»

|  |
| --- |
| import sys import os import tkinter as tk from tkinter import filedialog from PIL import Image, ImageEnhance, ImageFilter import pytesseract from docx import Document  current\_dir = getattr(sys, '\_MEIPASS', os.path.abspath(os.path.dirname(\_\_file\_\_))) tesseract\_path = os.path.join(current\_dir, 'tesseract-ocr/tesseract.exe') pytesseract.pytesseract.tesseract\_cmd = tesseract\_path  class JpegToDocConverter:  def \_\_init\_\_(self, root):  self.root = root  self.root.title("Преобразование текста из формата jpg в формат doc")  self.create\_widgets()   def create\_widgets(self):  font = ('Arial', 14)   self.label = tk.Label(self.root, text="Выберите изображение", font=font, wraplength=600)  self.label.pack(pady=10)   self.browse\_button = tk.Button(self.root, text="Выбрать", command=self.browse\_file, font=font)  self.browse\_button.pack(pady=10)   self.convert\_button = tk.Button(self.root, text="Преобразовать", command=self.convert\_to\_doc, font=font)  self.convert\_button.pack(pady=10)  self.convert\_button.config(state=tk.DISABLED)   self.file\_saved = False   def browse\_file(self):  file\_path = filedialog.askopenfilename(filetypes=[("Image files", "\*.jpg;\*.jpeg;\*.png")])  if file\_path:  self.file\_path = file\_path  self.label.config(text=f"Выбрано: {file\_path}")  self.convert\_button.config(state=tk.NORMAL)  self.file\_saved = False  else:  self.label.config(text="Файл не выбран")  self.convert\_button.config(state=tk.DISABLED)   def convert\_to\_doc(self):  if hasattr(self, 'file\_path'):  preprocessed\_image = self.preprocess\_image(self.file\_path)  text = pytesseract.image\_to\_string(preprocessed\_image, lang='rus+eng')   doc = Document()  doc.add\_paragraph(text)   save\_path = filedialog.asksaveasfilename(defaultextension=".doc", filetypes=[("Word files", "\*.doc")])  if save\_path:  doc.save(save\_path)  self.label.config(text=f"Результат сохранен в {save\_path}")  self.file\_saved = True   def preprocess\_image(self, image\_path):  image = Image.open(image\_path)  image = image.convert("L")  image = image.filter(ImageFilter.GaussianBlur(radius=0.3))  return image  root = tk.Tk() root.geometry('700x250+200+100') root.resizable(False, False) app = JpegToDocConverter(root) root.mainloop() |