**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.

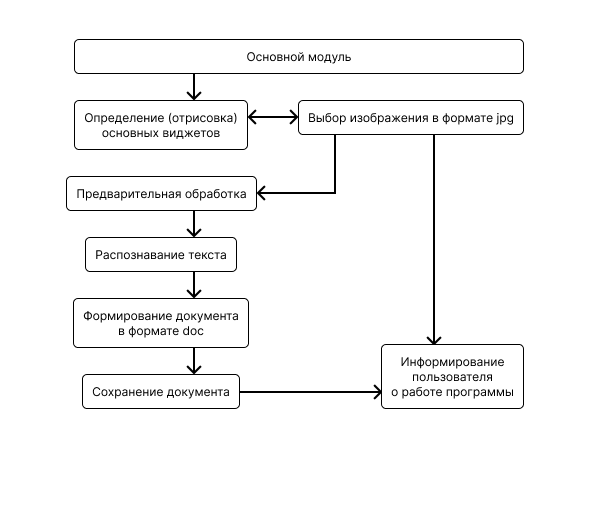
Из данной текстовый строки мы формируем DOC-документ. Для этого используется библиотека “python-docx”, представляющая средства для создания и редактирования документов.

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).