Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Институт цифровых технологий, электроники и физики

Кафедра вычислительной техники и электроники (ВТиЭ)

Отчёт по:

ПРОЕКТНО-ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

Студент группы: 505 Э. Н. Плотицына

Руководитель работы: ст. препод. И. А. Шмаков

Барнаул 2021 г.

РЕФЕРАТ

Полный объём работы составляет 29 страниц, включая 0 рисунков и 0 таблиц.

Цель работы (формулировка задания):

Создать приложение для считывания текста с изображения и занесения его в таблицу. Программа нацелена на работу с изображениями гербарных образцов, имеющих этикетку с сопровождающей информацией. Распределение информации об образце должно быть максимально простым в использовании. В приложении должны присутствовать как заполнение, так и изменение уже существующих табличных данных. Программа должна являться полностью завершенным программным продуктом.

Постановка задачи:

Написать приложение удобное в использовании. На вход программы поступает изображение или .csw файл. Программа считывает текст с изображения и выводит его в виде списка. У пользователя приложение есть возможность редактировать и распределить данных по категориям. При сохранении на выход программы поступает .csv файл, подготовленный к последующей загрузки в электронную базу данных «Virtual Herbarium ALTB». Цель программы помочь ускорить процесс заполнения электронной библиотеки. Время затраченное на анализ изображения программой зависит от характеристик компьютера.

Ключевые слова: .

Отчёт оформлена с помощью системы компьютерной вёрстки TEX и его расширения XETEX из дистрибутива *TeX Live*.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Написанная программа должна помочь сэкономить время и служить для автоматизации вывода и обработки данных с изображения. В качестве изображения используется фотография образца гербария с имеющейся на ней информационной этикеткой. Технология оптического распознавания переводит изображение в формат, доступный для редактирования. Программа находит буквы, объединяет их в слова и предложения, воссоздавая текст.

Остается только распределить полученную информацию в соответствии с ее значением. То есть распределить по категориям: ’Страна:’, ’Семейство:’, ’Род:’, ’Название вида:’, ’Внутренний номер:’, ’Коллектор/-ы:’, ’Место сбора:’, ’Экология:’, ’Высота н.у.м.:’, ’Координаты сбора:’, ’Дата сбора:’, ’Автор определения:’, ’Типовой статус:’.

Оптическое распознавание символов (англ. optical character recognition, OCR) — механический или электронный перевод изображений рукописного, машинописного или печатного текста в текстовые данные, использующиеся для представления символов в компьютере (например, в текстовом редакторе). Распознавание широко применяется для преобразования книг и документов в электронный вид.

Точное распознавание латинских символов в печатном тексте в настоящее время возможно, только если доступны чёткие изображения, такие, как сканированные печатные документы. Точность при такой постановке задачи превышает 99 %, абсолютная точность может быть достигнута только путём последующего редактирования человеком. Проблемы распознавания рукописного «печатного» и стандартного рукописного текста, а также печатных текстов других форматов (особенно с очень большим числом символов) в настоящее время являются предметом активных исследований. [2]

1. ОБЩИЕ СВЕДЕНИЯ

1.1. Обозначение и наименование программы

Полное наименование - «Оптическое распознавание символов для «Virtual Herbarium ALTB» ».

Краткое наименование - « OCR Virtual Herbarium ALTB».

1.2. Программное обеспечение, необходимое для функционирования программы

Клиентская часть:

—операционная система Windows, MacOs, Linux, и т.д.

Программные средства внешних систем:

—язык программирования Python3.6 и выше;

—модули: OpenCV-python v. 4.5; EasyOCR v. 1.4.1; Pillow v. 8.2.0;

1.3. Язык программирования, на котором написана программа и используемые модули

Программа написана на языке Python3.6.9.

Для создания кроссплатформенной программы с графическим интерфейсом используется поставляемая с Python библиотека tkinter на основе Tcl/Tk.

Для возможности выполнять структурный анализ файлов CSV используется одноименный модуль csv. С его помощью происходит открытие/создание электронной таблицы для внесения в нее изменений.

Для извлечения текста из изображения и построение рамки вокруг него на фото используется EasyOCR. EasyOCR - это пакет Python, который позволяет преобразовывать изображение в текст.

Для загрузки изображений из файлов, создания новых изображений и отображения в интерфейсе программы используется библиотека Pillow. Библиотека изображений Python, или PIL (Python Imaging Library) нужна для обработки графики в Python.

2. ФУНКЦИОНАЛЬНОЕ НАЗНАЧЕНИЕ

2.1. Классы решаемых задач

Классы решаемых задач:

—задачи извлечения информации;

—задачи визуализации информации;

—задачи обработки информации.

2.2. Назначение программы

Функциональное назначение:

— извлечение данных с изображений;

— наглядное графическое и табличное представление полученной информации;

— распределение, редактирование информации;

— сохранение информации записанной в формате .csv .

Эксплуатационное назначение:

— для использования в научных библиотеках.

3. ОПИСАНИЕ ЛОГИЧЕСКОЙ СТРУКТУРЫ

3.1. Используемые методы

3.1.1. Tkinter

class tkinter.ttk.Treeview

column(column, option=None, **kw): Изменение параметров для указанного column.

heading(column, option=None, **kw): Изменение параметров заголовка для указанного column.

delete(*items): Удалить все указанные items и все их потомки.

insert(parent, index, iid=None, **kw): Создает новый элемент и возвращает идентификатор вновь созданного элемента. parent - это идентификатор родительского элемента или пустая строка для создания нового элемента верхнего уровня. index - целое число или значение «END», указывающее место в списке для вставки нового элемента.

item(item, option=None, **kw): Изменение параметров для указанного item.

selection(): Возвращает кортеж выбранных элементов.

class tkinter.ttk.Style Класс используемый для управления базой данных стилей.

configure(style, query_opt=None, **kw): Задает значение по умолчанию для указанных параметров в style.

map(style, query_opt=None, **kw): Задание динамических значения указанных параметров в style.

xscrollcommand - Используется для связи с горизонтальными полосами прокрутки. yscrollcommand - Используется для связи с вертикальными полосами прокрутки.

Модули, обеспечивающие поддержку Tk, включают:

tkinter.filedialog Общие диалоговые окна, позволяющие пользователю указать файл для открытия или сохранения.

asksaveasfile

askopenfilename

tkinter.messagebox

showinfo Доступ к стандартным диалоговым окнам Tk. [5]

3.1.2. OpenCV

cv.imread(filename[, flags]): Функция imread загружает изображение из указанного файла и возвращает его. Если изображение не может быть прочитано (из-за отсутствия файла, неправильных разрешений, неподдерживаемого или недопустимого формата), функция возвращает пустую матрицу.

cv.cvtColor(src, code[, dst[, dstCn]]): Функция преобразует входное изображение из одного цветового пространства в другое. В случае преобразования цветового пространства в цветовое пространство RGB, порядок каналов должен быть указан явно (RGB или BGR).

cv.rectangle(img, pt1, pt2, color[, thickness[, lineType[, shift]]]): Функция cv.rectangle рисует контур прямоугольника или прямоугольник с заливкой, два противоположных угла которого - pt1 и pt2. Возвращает измененное изображение.

cv.putText(img, text, org, fontFace, fontScale, color[, thickness[, lineType[, bottomLeftOrigin]]]): Функция cv.putText отображает указанную текстовую строку в изображении. [3]

3.1.3. Pillow

PIL.Image.open(fp, mode='r', formats=None): Открывает и идентифицирует указанный файл изображения.

PIL.ImageTk.PhotoImage(image=None, size=None, **kw): Преобразует изображение в фотоизображение, совместимое с Tkinter. [4]

3.1.4. Csv

csv.writer(csvfile, dialect='excel', **fmtparams) Метод csv.writer возвращает объект записи, преобразует данные в строки с разделителями для данного CSV-файла.

csvwriter.writerow(row) Метод writerow записывает строку данных в указанный CSV-файл.

csv.reader(csvfile, dialect='excel', **fmtparams) Метод csv.reader считывает значения по строкам в данном CSV-файле. [1]

3.1.5. EasyOCR

reader.readtext Функция чтения текста. При запуске данного метода в виде списка возвращаются проанализированные строки, координаты текстовой области, а также уровень уверенности. [**EasyOCR**]

3.2. Описание функций составных частей программы

3.2.1. Функции главной программы

1. def func_fill_tree(value=’-’):
2. def select_all():
3. def func_add_record():
4. def func_remove_all():
5. def func_select(event):
6. def func_update():
7. def func_save():
8. def func_insert():
9. def select_file(choice):
10. def func_view_photo(img):
11. def func_view_csv(filename):
12. def func_scan_img(filename_img):

3.2.2. Функции подмодуля программы

1. def main(filename):
2. class InfoString:
   2.1. def __init__(self, bbox, text):
   2.2. def truncate(self, img):
   2.3. @classmethod def text(cls):

3.3. Описание связей между составными частями программы

В главную программу импортирован модуль подпрограммы. Название подпрограммы является её идентификатором, через который получается доступ к атрибутам, определенным внутри неё.

Доступ к атрибутам модуля осуществляется с помощью точечной нотации. Так для вызова подпрограммы из главной используется конструкция:

info_list, img = module.main(filename_img);

На вход функции main() подпрограммы module поступает filename_img, содержащий в себе полный путь для доступа к файлу с изображением.

На выход из функции поступает info_list и img, которое записываются в одноименные переменные в главной программе. Теперь info_list содержит в себе список из строк, полученных с изображения, а img обработанное изображение.

3.4. Связи программы с другими программами

С программой используется встроенное средство выбора файлов.

В средстве выбора файлов отображаются сведения, позволяющие сориентировать пользователя и обеспечить ему привычное взаимодействие с системой при открытии и сохранении файлов.

Программа совместима с операционной системой Windows и Linux, при условии установленных модулей используемых в ней.

4. ВХОДНЫЕ ДАННЫЕ И ВЫХОДНЫЕ ДАННЫЕ

4.1. Входные данные

На вход могут поступать файлы в формате .jpg/ .png изображений и .csv табличных данных.

Изображение поступающее на вход программы должно быть хорошего разрешения для наиболее точного получения реального текста.

В файле с табличными данными .csv должно быть использовано разделение столбцов данных по ”,”.

4.2. Выходные данные

Выходные данные, представляют собой текстовые файлы в формате .csv.

Тип файла, имеющий расширение CSV, содержит информацию, которую могут импортировать в базу данных, так же подобный текстовый файл может носить данные, необходимые для организации таблиц.

5. ИНТЕРФЕЙС ПРОГРАММЫ

Функциональность и интерфейс пользователя для разрабатываемого программного продукта:

Пользовательский интерфейс представляет собой совокупность программных и аппаратных средств, обеспечивающих диалог пользователя и компьютера.

Для пользователя доступно шесть кнопок, пять из которых активны с запуска приложения. Кнопка ”Сканировать изображение” появляется после открытия через файловый диалог изображения.

Функции кнопок:

”Открыть изображение” - выбрать путь расположения файла и открыть его через файловый диалог(для изображения);

”Откры́ыть .csv файл” - выбрать путь расположения файла и открыть его через файловый диалог(для .csv);

”Взять выбранные элементы” - выбрать выделенные строки из списка и переместить данные в поле ”Данные” для редактирования;

”Обновить” - обновит содержимое столбца ”Данные” изменяемой строки таблицы;

”Сканировать изображение” - запуск считывания текста с изображения;

”Сохранить” - сохранить информацию из таблицы в .csv файл.

Последовательность действий для пользователя:

При редактировании .csv файла без необходимости сканирования изображения:

1) Открыть файл с табличными данными в формате .csv;

2) Выбрать двойным щелчком мыши из таблицы строку с категорией, которую необходимо изменить;

3) В поле ”Данные” отредактировать или добавить информацию об образце;

4) Использовать кнопку ”Обновить” для переноса текста в таблицу с данными;

5) Повторить при необходимости п.2-4;

6) Использовать кнопку ”Сохранить”, выбрать имя и расположения нового файла.

При сканировании изображения:

1) Открыть изображение в формате .jpg/.png;

2) Использовать кнопку ”Сканировать изображение”, и подождать когда процесс закончится;

3) Выбрать двойным щелчком мыши из таблицы строку с категорией, которую необходимо изменить;

4) Выбрать один или несколько элементов из списка со считанной информацией;

5) Использовать кнопку ”Взять выбранные элементы”, после чего они помещаются в поле для редактирования ”Данные”;

6) В поле ”Данные” отредактировать или добавить информацию об образце;

7) Использовать кнопку ”Обновить” для переноса текста в таблицу с данными;

8) Повторить при необходимости п.3-7;

9) Использовать кнопку ”Сохранить”, выбрать имя и расположения нового файла.

ЗАКЛЮЧЕНИЕ

Написанная программа сокращает время затрачиваемое на занесение данных о гербарных образцах в электронную базу данных. Упрощает процесс занесения информации в таблицу, делая его интерактивным и быстрым.

Данную программу можно использовать в научных библиотеках, в учебных заведениях или в собственных целях.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. [Электронный ресурс] CSV File Reading and Writing. — URL: https://docs.python.org/3/library/csv.html.
2. [Электронный ресурс] OCR - Википедия. — URL: https://ru.wikipedia.org/wiki/.
3. [Электронный ресурс] OpenCV - Image file reading and writing. — URL: https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56.
4. [Электронный ресурс] Reference — Pillow. — URL: https://pillow.readthedocs.io/en/stable/reference/index.html.
5. [Электронный ресурс] Tk — Commands. — URL: https://www.tcl.tk/man/tcl8.6/TkCmd/contents.html.

5.1. Код программы OCR Virtual Herbarium ALTB.py

```
from tkinter import *
from tkinter import ttk
import module
import csv
from PIL import Image, ImageTk
from tkinter import filedialog as fd
from tkinter.messagebox import showinfo

text = ['Страна:', 'Семейство:', 'Род:', 'Название
 ↪  вида:', 'Внутренний номер:', 'Коллектор/-ы:',
 ↪  'Место сбора:',
        'Экология:', 'Высота н.у.м.:', 'Координаты
         ↪  сбора:', 'Дата сбора:', 'Автор
         ↪  определения:', 'Типовой статус:']


def func_fill_tree(value='-'):
    count = 0
    for i in text:
        my_tree.insert(parent='', index=END,
         ↪  iid=str(count), text='',
         ↪  values=(str(count), i, value))
        count += 1


def select_all():
    result = entryDataText.get()
    for item in my_listbox.curselection():
        result = result + str(my_listbox.get(item)) + '
         ↪  '
    entryDataText.set(result)
```

```
def func_add_record():
    global count
    my_tree.insert(parent='', index=END,
     ↪  iid=str(count), text='', values=(count,
     ↪  name_box.get(), data_box.get()))
    count += 1
    name_box.delete(0, END)
    data_box.delete(0, END)


def func_remove_all():
    global count
    for record in my_tree.get_children():
        my_tree.delete(record)
        count = 0


def func_select(event):
    id_box.delete(0, END)
    name_box.delete(0, END)
    data_box.delete(0, END)
    # номер строки
    selected = my_tree.focus()
    # значение
    values = my_tree.item(selected, 'values')
    entryIdText.set(values[0])
    entryNameText.set(values[1])
    entryDataText.set(values[2])


def func_update():
    if entryIdText.get() and entryNameText.get():
        selected = my_tree.focus()
```

```
            my_tree.item(selected, text="",
             ↪  values=(entryIdText.get(),
             ↪  entryNameText.get(), entryDataText.get()))
        entryIdText.set("")
        entryNameText.set("")
        data_box.delete(0, END)


def func_save():
    file_name = fd.asksaveasfile(mode='w',
     ↪  defaultextension=".csv")
    if file_name is None:
        return
    with open(file_name.name, 'w', newline="") as
     ↪  text_file:
        writer = csv.writer(text_file)
        writer.writerow(func_insert())


def func_insert():
    insert_arr = []
    for items in my_tree.get_children():
        values = my_tree.item(items, 'values')
        insert_arr.append(values[2])
    return insert_arr


def select_file(choice):
    if choice:
        filetypes = (('images files', '*.jpg'),
                     ('All files', '*.*'))
        filename_img = fd.askopenfilename(
            title='Open a file',
            initialdir='/',
```

```
            filetypes=filetypes)
        if filename_img:
            img = Image.open(filename_img)
        else:
            img = None
        if img is None:
            showinfo(title='Could not open or find the
             ↪  image:',
                     message=filename_img)
        else:
            showinfo(title='Selected File',
                     message=filename_img)
            func_view_photo(img)
            ttk.Button(my_frame_but_2,
             ↪  text="Сканировать изображение",
                       command=lambda:
                        ↪  func_scan_img(filename_img)).gri
                        ↪  column=0)
            func_fill_tree()
    else:
        filetypes = (('images files', '*.csv'),
                     ('All files', '*.*'))
        filename_csv = fd.askopenfilename(
            title='Open a file',
            initialdir='/',
            filetypes=filetypes)
        showinfo(title='Selected File',
                 message=filename_csv)
        if filename_csv:
            func_view_csv(filename_csv)


def func_view_photo(img):
```

```
    img = img.resize((int(new_width/2)-150,
     ↪  int(new_height)-100), Image.ANTIALIAS)
    img = ImageTk.PhotoImage(img)
    panel.config(image=img)
    panel.image = img


def func_view_csv(filename):
    with open(filename, 'r') as f:
        reader = csv.reader(f, delimiter=',')
        read_list = list(reader)
        read_list2 = read_list[0]
        count = 0
        func_remove_all()
        for i in text:
            my_tree.insert(parent='', index=END,
             ↪  iid=str(count), text='',
             ↪  values=(str(count), i,
             ↪  read_list2[count]))
            count += 1


def func_scan_img(filename_img):
    info_list, img = final1_module.main(filename_img)
    img_into = Image.fromarray(img)
    func_view_photo(img_into)
    for item in info_list:
        my_listbox.insert(END, item)


ws = Tk()
ws.title('OCR')
ws.config(background='#AFE1AF')
width_wind = ws.winfo_screenwidth()
```

```
height_wind = ws.winfo_screenheight()
ws.resizable(width=False, height=False)

new_width = int(width_wind/1.3)
new_height = int(height_wind/1.5)

style = ttk.Style()
style.configure("Treeview",
                background="silver",
                foreground="black",
                rowheight=25,
                fieldbackground="silver")
style.configure('new.TFrame', background='#AFE1AF')
style.configure('new.TLabel', foreground="black",
 ↪  background="#AFE1AF")
# style.configure('.', font=('Helvetica', 12))
style.map("Treeview", background=[('selected',
 ↪  'green')])
style.map('TButton',
          foreground=[('!active', 'black'),
                      ('pressed', 'white'),
                      ('active', 'black')],
          background=[('!active', '#008000'),
                      ('pressed', '#006400'),
                      ('active', 'white')]
          )

panel = ttk.Label(ws, style='new.TLabel',
 ↪  width=int(new_width/20)-25)
panel.grid(row=1, column=0, rowspan=4)

my_frame_text_before_tree = Frame(ws,
 ↪  background="white")
```

```
ttk.Label(my_frame_text_before_tree,
 ↪  style='new.TLabel',
          text="Выберите необходимую категорию для
           ↪  изменения:",
           ↪  width=int(new_width/20)).pack(side=LEFT)

my_frame_text_before_tree.grid(row=0, column=1,
 ↪  sticky=SW)

my_frame_tree = Frame(ws)
my_scrollbar_tree_y = Scrollbar(my_frame_tree,
 ↪  orient=VERTICAL)
my_scrollbar_tree_x = Scrollbar(my_frame_tree,
 ↪  orient=HORIZONTAL)

my_tree = ttk.Treeview(my_frame_tree, height=10)

my_tree['columns'] = ('ID', 'Категория', 'Данные')
my_tree.column('#0', width=0, stretch=NO)
my_tree.column('ID', anchor=W, width=50, minwidth=50)
my_tree.column('Категория', anchor=W, width=200,
 ↪  minwidth=200)
my_tree.column('Данные', anchor=W,
 ↪  width=int(new_width/2)-200,
 ↪  minwidth=int(new_width/2)-200)

my_tree.heading('#0', text='', anchor=CENTER)
my_tree.heading('ID', text='ID', anchor=CENTER)
my_tree.heading('Категория', text='Категория',
 ↪  anchor=CENTER)
my_tree.heading('Данные', text='Данные',
 ↪  anchor=CENTER)

my_tree.bind("<Double-1>", func_select)
```

```
my_scrollbar_tree_y.pack(side=RIGHT, fill=Y)
my_scrollbar_tree_x.pack(side=BOTTOM, fill=X)

my_tree.config(yscrollcommand=my_scrollbar_tree_y.set,
 ↪  xscrollcommand=my_scrollbar_tree_x.set)
my_scrollbar_tree_y.config(command=my_tree.yview)
my_scrollbar_tree_x.config(command=my_tree.xview)

my_tree.pack()

my_frame_tree.grid(row=1, column=1, sticky=NW)

my_frame_text_before_list = ttk.Frame(ws,
 ↪  style='new.TFrame', width=int(new_width/20))

ttk.Label(my_frame_text_before_list,
 ↪  style='new.TLabel',
          text="Укажите подходящие данные для занесения
           ↪  в выбранную категорию:").pack(side=TOP)
ttk.Label(my_frame_text_before_list,
 ↪  style='new.TLabel',
           text="(для множественного выбора
            ↪  +ctrl)").pack(side=LEFT)
ttk.Button(my_frame_text_before_list, text="Взять
 ↪  выбранные элементы",
 ↪  command=select_all).pack(side=RIGHT)

my_frame_text_before_list.grid(row=2, column=1,
 ↪  sticky=SW)

my_frame_listbox = ttk.Frame(ws, style='new.TFrame',
 ↪  width=int(new_width/2))
```

```
my_listbox = Listbox(my_frame_listbox,
 ↪  width=int(new_width/15), selectmode=EXTENDED)
my_listbox.grid(row=0, column=0)

my_scrollbar_list_y = Scrollbar(my_frame_listbox,
 ↪  orient=VERTICAL)
my_scrollbar_list_x = Scrollbar(my_frame_listbox,
 ↪  orient=HORIZONTAL)

my_scrollbar_list_y.grid(row=0, column=1, sticky=NS)
my_scrollbar_list_x.grid(row=1, column=0, sticky=EW)

my_listbox.config(yscrollcommand=my_scrollbar_list_y.set,
 ↪  xscrollcommand=my_scrollbar_list_x.set)
my_scrollbar_list_y.config(command=my_listbox.yview)
my_scrollbar_list_x.config(command=my_listbox.xview)
my_frame_listbox.grid(row=3, column=1, sticky=NW)

my_frame_text_before_box = ttk.Frame(ws,
 ↪  style='new.TFrame', width=new_width/2)
ttk.Label(my_frame_text_before_box,
 ↪  style='new.TLabel', text="Внесите при
 ↪  необходимости изменения в записи данных, \n \
после чего занесите их в таблицу:").pack(side=LEFT)
ttk.Button(my_frame_text_before_box, text="Обновить",
 ↪  command=func_update).pack(side=RIGHT, padx=10)
my_frame_text_before_box.grid(row=4, column=1,
 ↪  sticky=SW)

my_frame_box = ttk.Frame(ws, style='new.TFrame',
 ↪  width=new_width/2)

my_scrollbar_x2 = Scrollbar(my_frame_box,
 ↪  orient=HORIZONTAL)
```

```
ttk.Label(my_frame_box, style='new.TLabel',
 ↪  text="№").grid(row=0, column=0)

ttk.Label(my_frame_box, style='new.TLabel',
 ↪  text="Категория", width=20).grid(row=0, column=1)

ttk.Label(my_frame_box, style='new.TLabel',
 ↪  text="Данные",
 ↪  width=int(new_width/20)-30).grid(row=0, column=2)

entryIdText = StringVar()
entryNameText = StringVar()
entryDataText = StringVar()

id_box = Entry(my_frame_box, width=5,
 ↪  disabledbackground="white",
 ↪  disabledforeground="black", state='disabled',
                textvariable=entryIdText)
id_box.grid(row=1, column=0)

name_box = Entry(my_frame_box, width=20,
 ↪  disabledbackground="white",
 ↪  disabledforeground="black", state='disabled',
                  textvariable=entryNameText)
name_box.grid(row=1, column=1)

data_box = Entry(my_frame_box,
 ↪  width=int(new_width/20),
 ↪  textvariable=entryDataText)
data_box.grid(row=1, column=2)

my_scrollbar_x2.grid(row=2, column=2, sticky=EW)
data_box.config(xscrollcommand=my_scrollbar_x2.set)
```

```
my_scrollbar_x2.config(command=data_box.xview)

my_frame_box.grid(row=5, column=1, pady=0, sticky=NW)


my_frame_but_1 = ttk.Frame(ws, style='new.TFrame')

ttk.Button(my_frame_but_1, text="Открыть изображение",
 ↪  width=int(new_width/50),
            command=lambda:
             ↪  select_file(True)).grid(row=0,
             ↪  column=0)
ttk.Button(my_frame_but_1, text="Открыть .csv файл",
 ↪  width=int(new_width/50),
            command=lambda:
             ↪  select_file(False)).grid(row=0,
             ↪  column=1)

my_frame_but_1.grid(row=0, column=0)


my_frame_but_2 = ttk.Frame(ws, style='new.TFrame',
 ↪  width=new_width/2)

ttk.Button(my_frame_but_2, text='Сохранить',
 ↪  width=int(new_width/50),
 ↪  command=func_save).grid(row=0, column=1)

my_frame_but_2.grid(row=5, column=0)

ws.mainloop()
```

5.2. Код подпрограммы module.py

```
import easyocr
import cv2


class InfoString:
    info_string = []

    def __init__(self, bbox, text):
        InfoString.info_string.append(self)
        self.tl, self.tr, self.br, self.bl = bbox
        self.id_text = text
        self.crop = None

    def truncate(self, img):
        self.crop =
         ↪  img[int(self.tl[1]):int(self.br[1]),
         ↪  int(self.tl[0]):int(self.br[0])]

    @classmethod
    def text(cls):
        global info_list
        for obj in cls.info_string:
            info_list.append(obj.id_text)


def main(filename):
    global info_list
    info_list = []
    img = cv2.imread(filename)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    reader = easyocr.Reader(['ru', 'en'], gpu=True)
    result = reader.readtext(gray, width_ths=2)
```

```
    for (bbox, text, _) in result:
        s = InfoString(bbox, text)
        s.truncate(gray)

        # tl - top-left, br - bottom-right
        (tl, tr, br, bl) = bbox
        tl = (int(tl[0]), int(tl[1]))
        br = (int(br[0]), int(br[1]))
        cv2.rectangle(img, tl, br, (0, 255, 0), 2)
        cv2.putText(img, text, (tl[0], tl[1] - 10),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,
                     ↪  255, 0), 2)

    InfoString.text()
    return info_list, img
```