Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №7**

**Дисциплина: Алгоритмы цифровой обработки мультимедиа**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.М. Нагалевский

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А. А. Крамаренко

**Тема работы:** решения для TesseractOCR и EasyOCR на примере распознавания капчи.

**Ход работы:**

TesseractOCR — это библиотека оптического распознавания символов (OCR), которая разрабатывается Google. Она предоставляет средства для распознавания текста на изображениях. Проект начался в 1985 году в лаборатории искусственного интеллекта Hewlett Packard, а затем был активно развиваем и поддерживаем Google.

Принцип работы TesseractOCR включает в себя несколько этапов:

1. **Предварительная обработка изображения:** Прежде чем приступить к распознаванию текста, изображение подвергается предварительной обработке. Это включает в себя такие шаги, как бинаризация (преобразование изображения в черно-белое), устранение шума, настройка контраста и другие методы улучшения качества изображения.
2. **Разбиение изображения на блоки (сегментация):** Изображение разбивается на маленькие блоки или линии для более эффективного анализа. Это помогает улучшить точность распознавания, особенно когда текст на изображении имеет разные шрифты или ориентации.
3. **Выделение текстовых областей:** TesseractOCR старается выделить области, где находится текст. Это помогает сузить область поиска и ускорить процесс распознавания.
4. **Распознавание символов:** На этом этапе TesseractOCR анализирует каждый блок или линию изображения и пытается распознать символы, используя модели машинного обучения. Он опирается на обширные данные обучения, чтобы понять форму и структуру символов.
5. **Обработка результатов:** После распознавания TesseractOCR предоставляет текстовый вывод, который можно использовать в приложениях или системах для дальнейшей обработки.

TesseractOCR поддерживает множество языков и может работать с различными типами изображений, включая изображения с низким разрешением и плохим освещением. Он является открытым программным обеспечением и широко используется сообществом разработчиков для решения задач распознавания текста.

EasyOCR — это еще одна библиотека оптического распознавания символов (OCR), которая предоставляет простой интерфейс для выполнения задач распознавания текста. Она разработана на языке программирования Python и является надстройкой над TesseractOCR, предоставляя более удобный API для использования функционала Tesseract.

Рассмотрим некоторые ключевые характеристики EasyOCR:

1. **Простой интерфейс**: Как следует из названия, EasyOCR ориентирована на простоту использования. Она предоставляет удобный API для распознавания текста, что делает ее доступной для разработчиков с различным уровнем опыта.
2. **Многопоточность**: EasyOCR поддерживает многопоточность, что позволяет распознавать текст на нескольких изображениях параллельно. Это может значительно улучшить производительность при обработке больших объемов данных.
3. **Поддержка нескольких языков**: EasyOCR, также как и TesseractOCR, обладает широкой поддержкой языков. Это делает ее полезной для проектов, в которых требуется распознавание текста на различных языках.
4. **Модульность и конфигурируемость:** EasyOCR позволяет настраивать различные параметры распознавания, такие как язык, используемые модели, а также предоставляет возможность настройки предварительной обработки изображений.
5. **Использование глубокого обучения**: EasyOCR использует глубокие нейронные сети для более точного распознавания текста. Она может автоматически выбирать оптимальные параметры для каждого языка и типа изображения.

Принцип работы EasyOCR в основном основан на использовании TesseractOCR внутри, а также на дополнительных оптимизациях и удобствах, предоставляемых EasyOCR.

Общий принцип работы:

1. **Инициализация и конфигурация:** Разработчик создает экземпляр Reader с указанием языков, которые будут использоваться для распознавания текста. Этот шаг также может включать в себя конфигурацию дополнительных параметров, таких как использование глубокого обучения, параметры предварительной обработки изображений и другие.
2. **Подготовка изображения:** Перед тем как передать изображение на распознавание, EasyOCR может выполнить некоторые предварительные шаги обработки, такие как изменение размера изображения, улучшение контраста, бинаризация и другие методы для оптимизации качества распознавания.
3. **Разбиение изображения на блоки:** Аналогично TesseractOCR, EasyOCR может разбивать изображение на блоки или линии для более эффективного распознавания текста, особенно когда на изображении присутствует текст с разными шрифтами или ориентациями.
4. **Распознавание текста:** EasyOCR передает блоки изображения в TesseractOCR для распознавания текста. В этот момент используются глубокие нейронные сети и модели машинного обучения для анализа областей и определения символов.
5. **Обработка результатов:** Результаты распознавания возвращаются в виде текстового вывода, который может быть использован разработчиком в своем приложении или системе.

Теперь рассмотрим плюс и минусы каждой из моделей.

**TesseractOCR.**

Плюсы:

* Широкая поддержка языков: TesseractOCR поддерживает множество языков, что делает его мощным инструментом для распознавания текста на различных языках.
* Бесплатное и открытое ПО: TesseractOCR является открытым программным обеспечением, что означает, что он бесплатен для использования и может быть модифицирован в соответствии с потребностями пользователя.
* Обширные возможности настройки: Пользователи могут настраивать различные параметры и конфигурации для достижения оптимальных результатов в различных сценариях.

Минусы:

* Сложность использования: Интерфейс TesseractOCR может быть не таким простым для новичков. Настройка параметров и обработка результатов требуют понимания внутреннего механизма работы.
* Требуется предварительная обработка изображений: Наилучшие результаты достигаются при предварительной обработке изображений, что может быть вызовом для пользователей, не знакомых с областью обработки изображений.

**EasyOCR.**

Плюсы:

* Простой интерфейс: EasyOCR предоставляет более удобный и простой интерфейс для использования. Это упрощает интеграцию OCR в проекты, особенно для разработчиков, работающих с языком программирования Python.
* Многопоточность: Поддержка многопоточности в EasyOCR может улучшить производительность при обработке большого числа изображений параллельно.
* Использование глубокого обучения: EasyOCR использует глубокие нейронные сети, что способствует более точному распознаванию текста.

Минусы:

* Зависимость от TesseractOCR: EasyOCR внутренне использует TesseractOCR, и если программисту нужны последние обновления или определенные функции TesseractOCR, можно столкнуться с ограничениями.
* Ограниченная гибкость настройки: В сравнении с TesseractOCR, EasyOCR может предоставлять меньше параметров для настройки, что может быть ограничивающим в определенных случаях.

В общем, выбор между TesseractOCR и EasyOCR зависит от конкретных потребностей проекта и уровня опыта разработчика. TesseractOCR предоставляет большую гибкость и функциональность, тогда как EasyOCR обеспечивает более простой и удобный интерфейс.

После выполнение заданий и сбора итоговых данных, получилась следующая таблица сравнения работы двух моделей на предоставленном датасете (рисунок 1).

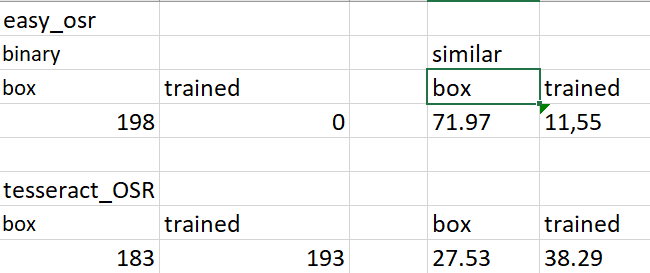
****

Рисунок 1 – сравнение моделей

**Листинг программ**

Файл task1-2.py

import csv

import pathlib

import cv2

import pytesseract

from pytesseract import Output

from difflib import SequenceMatcher

import statistics

pytesseract.pytesseract.tesseract\_cmd = r"C:/Program Files/Tesseract-OCR/tesseract.exe"

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

    path = pathlib.Path(\_\_file\_\_).parent / rel\_path

    return path

def test\_recognition(rec\_type, val\_type, dataset\_name, show\_img=False):

    output\_str = ""

    labels = {}

    images\_count = 0

    correct\_guesses = 0

    similarities = []

    with open(

        str(rel\_path(dataset\_name + "/labels.csv")), newline="", encoding="utf-8"

    ) as csvfile:

        reader = csv.reader(csvfile, delimiter=",", quotechar="'")

        for row in reader:

            labels[row[0]] = row[1]

    img\_files = list(

        pathlib.Path(str(rel\_path(dataset\_name + "/"))).glob("\*.jpg")

    )

    for img\_file in img\_files:

        img = cv2.imread(str(img\_file.resolve()), 0)

        groud\_truth = labels[img\_file.name]

        if rec\_type == "straight\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

        if rec\_type == "boxes\_recognition":

            h, w = img.shape

            boxes = pytesseract.image\_to\_boxes(img, lang="rus+eng")

            for box in boxes.splitlines():

                box\_data = box.split(" ")

                cv2.rectangle(

                    img,

                    (int(box\_data[1]), h - int(box\_data[2])),

                    (int(box\_data[3]), h - int(box\_data[4])),

                    (0, 255, 0),

                    2,

                )

            result = "".join([sym\_data.split(" ")[0] for sym\_data in boxes.split("\n")])

        result = "".join(result.splitlines())

        output\_str += f"{img\_file.name} | {groud\_truth} | {result}\n"

        if val\_type == "binary\_correct":

            if result.lower() == groud\_truth.lower():

                correct\_guesses += 1

        if val\_type == "similarity":

            similarity = SequenceMatcher(

                None, groud\_truth.lower(), result.lower()

            ).ratio()

            similarities.append(similarity)

        images\_count += 1

        print(result)

        if show\_img:

            cv2.imshow("capthca", img)

            cv2.waitKey()

    output\_str += "\n"

    if val\_type == "binary\_correct":

        output\_str += f"Угадано {correct\_guesses} / {images\_count} капч"

    if val\_type == "similarity":

        output\_str += (

            f"Средняя схожесть: {statistics.fmean(similarities) \* 100}%"

        )

    with open(

        str(

            rel\_path(

                "results\_" + val\_type + "\_" + rec\_type + "\_" + dataset\_name + ".txt"

            )

        ),

        "w",

        encoding="utf-8",

    ) as f:

        f.write(output\_str)

def main():

    test\_recognition(

        "straight\_recognition", "binary\_correct", "dataset", show\_img=False

    )

    test\_recognition("straight\_recognition", "similarity", "dataset", show\_img=False)

    test\_recognition(

        "straight\_recognition", "binary\_correct", "dataset2", show\_img=False

    )

    test\_recognition("straight\_recognition", "similarity", "dataset2", show\_img=False)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Файл task3-4.py

import csv

import pathlib

import cv2

import re

import pytesseract

from pytesseract import Output

from difflib import SequenceMatcher

import statistics

pytesseract.pytesseract.tesseract\_cmd = r"C:/Program Files/Tesseract-OCR/tesseract.exe"

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

    path = pathlib.Path(\_\_file\_\_).parent / rel\_path

    return path

def clear\_str(str):

    str = str.lower()

    str = re.sub(

        r"[0123456789\!\#\$\%\^\&\\*\(\)\\_\~@\`\n\/\|\,\"\<\°\„\?\.\«\’\‚\”\“\®\¥\>\`\'\—\™\‘\:\ \]\[\{\}\=\+\-\\]",

        " ",

        str,

    )

    return str

def test\_recognition(rec\_type, val\_type, dataset\_name, show\_img=False):

    output\_str = ""

    labels = {}

    images\_count = 0

    correct\_guesses = 0

    similarities = []

    dict\_for\_avg\_of\_aug\_dataset = {}

    with open(

        str(rel\_path(dataset\_name + "/labels.csv")), newline="", encoding="utf-8"

    ) as csvfile:

        reader = csv.reader(csvfile, delimiter=",", quotechar="'")

        for row in reader:

            labels[row[0]] = row[1]

    img\_files = list(

        pathlib.Path(str(rel\_path(dataset\_name + "/"))).glob("\*.jpg")

    )

    for img\_file in img\_files:

        img = cv2.imread(str(img\_file.resolve()), 0)

        groud\_truth = labels[img\_file.name]

        if rec\_type == "straight\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

        elif rec\_type == "boxes\_recognition":

            h, w = img.shape

            boxes = pytesseract.image\_to\_boxes(img, lang="rus+eng")

            for box in boxes.splitlines():

                box\_data = box.split(" ")

                cv2.rectangle(

                    img,

                    (int(box\_data[1]), h - int(box\_data[2])),

                    (int(box\_data[3]), h - int(box\_data[4])),

                    (0, 255, 0),

                    2,

                )

            result = "".join([sym\_data.split(" ")[0] for sym\_data in boxes.split("\n")])

        elif rec\_type == "filtered\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

            groud\_truth = clear\_str(groud\_truth)

            result = clear\_str(result)

        elif rec\_type == "avg\_of\_aug":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

            img\_name\_wo\_aug = img\_file.name.split("\_")[0]

            if img\_name\_wo\_aug in dict\_for\_avg\_of\_aug\_dataset:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug].append(result)

            else:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug] = [result]

        result = "".join(result.splitlines())

        output\_str += f"{img\_file.name} | {groud\_truth} | {result}\n"

        if val\_type == "binary\_correct":

            if (

                result.lower() in groud\_truth.lower()

                or groud\_truth.lower() in result.lower()

            ):

                correct\_guesses += 1

        elif val\_type == "similarity":

            similarity = SequenceMatcher(

                None, groud\_truth.lower(), result.lower()

            ).ratio()

            similarities.append(similarity)

        images\_count += 1

        print(result)

        if show\_img:

            cv2.imshow("capthca", img)

            cv2.waitKey()

    output\_str += "\n"

    if rec\_type == "avg\_of\_aug":

        images\_count = 0

        similarities = []

        correct\_guesses = 0

        for key in dict\_for\_avg\_of\_aug\_dataset.keys():

            results = dict\_for\_avg\_of\_aug\_dataset[key]

            if val\_type == "binary\_correct":

                for result in results:

                    if (

                        result.lower() in groud\_truth.lower()

                        or groud\_truth.lower() in result.lower()

                    ):

                        correct\_guesses += 1

                        break

            elif val\_type == "similarity":

                max\_similarity = 0

                for result in results:

                    similarity = SequenceMatcher(

                        None, groud\_truth.lower(), result.lower()

                    ).ratio()

                    max\_similarity = max(similarity, max\_similarity)

                similarities.append(max\_similarity)

            images\_count += 1

    if val\_type == "binary\_correct":

        output\_str += f"Угадано {correct\_guesses} / {images\_count} капч"

    elif val\_type == "similarity":

        output\_str += (

            f"Средняя схожесть: {statistics.fmean(similarities) \* 100}%"

        )

    with open(

        str(

            rel\_path(

                "results\_" + val\_type + "\_" + rec\_type + "\_" + dataset\_name + ".txt"

            )

        ),

        "w",

        encoding="utf-8",

    ) as f:

        f.write(output\_str)

def main():

    test\_recognition("avg\_of\_aug", "binary\_correct", "dataset2", show\_img=False)

    test\_recognition("avg\_of\_aug", "similarity", "dataset2", show\_img=False)

    test\_recognition(

        "filtered\_recognition", "binary\_correct", "dataset2", show\_img=False

    )

    test\_recognition("filtered\_recognition", "similarity", "dataset", show\_img=False)

    test\_recognition("filtered\_recognition", "similarity", "dataset2", show\_img=False)

    test\_recognition(

        "filtered\_recognition", "binary\_correct", "dataset", show\_img=False

    )

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Файл task5.py

import csv

import pathlib

import cv2

import re

import pytesseract

from pytesseract import Output

from difflib import SequenceMatcher

import statistics

pytesseract.pytesseract.tesseract\_cmd = r"C:/Program Files/Tesseract-OCR/tesseract.exe"

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

    path = pathlib.Path(\_\_file\_\_).parent / rel\_path

    return path

def clear\_str(str):

    str = str.lower()

    str = re.sub(

        r"[0123456789\!\#\$\%\^\&\\*\(\)\\_\~@\`\n\/\|\,\"\<\°\„\?\.\«\’\‚\”\“\®\¥\>\`\'\—\™\‘\:\ \]\[\{\}\=\+\-\\]",

        " ",

        str,

    )

    return str

def test\_recognition(rec\_type, val\_type, dataset\_name, show\_img=False):

    output\_str = ""

    labels = {}

    images\_count = 0

    correct\_guesses = 0

    similarities = []

    dict\_for\_avg\_of\_aug\_dataset = {}

    with open(

        str(rel\_path(dataset\_name + "/labels.csv")), newline="", encoding="utf-8"

    ) as csvfile:

        reader = csv.reader(csvfile, delimiter=",", quotechar="'")

        for row in reader:

            labels[row[0]] = row[1]

    img\_files = list(

        pathlib.Path(str(rel\_path(dataset\_name + "/"))).glob("\*.jpg")

    )

    for img\_file in img\_files:

        img = cv2.imread(str(img\_file.resolve()), 0)

        groud\_truth = labels[img\_file.name]

        if rec\_type == "straight\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

        elif rec\_type == "boxes\_recognition":

            h, w = img.shape

            boxes = pytesseract.image\_to\_boxes(img, lang="rus+eng")

            for box in boxes.splitlines():

                box\_data = box.split(" ")

                cv2.rectangle(

                    img,

                    (int(box\_data[1]), h - int(box\_data[2])),

                    (int(box\_data[3]), h - int(box\_data[4])),

                    (0, 255, 0),

                    2,

                )

            result = "".join([sym\_data.split(" ")[0] for sym\_data in boxes.split("\n")])

        elif rec\_type == "filtered\_recognition":

            result = pytesseract.image\_to\_string(img, lang="test4")

            groud\_truth = clear\_str(groud\_truth)

            result = clear\_str(result)

        elif rec\_type == "avg\_of\_aug":

            result = pytesseract.image\_to\_string(img, lang="test4")

            img\_name\_wo\_aug = img\_file.name.split("\_")[0]

            if img\_name\_wo\_aug in dict\_for\_avg\_of\_aug\_dataset:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug].append(result)

            else:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug] = [result]

        result = "".join(result.splitlines())

        output\_str += f"{img\_file.name} | {groud\_truth} | {result}\n"

        if val\_type == "binary\_correct":

            if (

                result.lower() in groud\_truth.lower()

                or groud\_truth.lower() in result.lower()

            ):

                correct\_guesses += 1

        elif val\_type == "similarity":

            similarity = SequenceMatcher(

                None, groud\_truth.lower(), result.lower()

            ).ratio()

            similarities.append(similarity)

        images\_count += 1

        print(result)

        if show\_img:

            cv2.imshow("capthca", img)

            cv2.waitKey()

    output\_str += "\n"

    if rec\_type == "avg\_of\_aug":

        images\_count = 0

        similarities = []

        correct\_guesses = 0

        for key in dict\_for\_avg\_of\_aug\_dataset.keys():

            results = dict\_for\_avg\_of\_aug\_dataset[key]

            if val\_type == "binary\_correct":

                for result in results:

                    if (

                        result.lower() in groud\_truth.lower()

                        or groud\_truth.lower() in result.lower()

                    ):

                        correct\_guesses += 1

                        break

            elif val\_type == "similarity":

                max\_similarity = 0

                for result in results:

                    similarity = SequenceMatcher(

                        None, groud\_truth.lower(), result.lower()

                    ).ratio()

                    max\_similarity = max(similarity, max\_similarity)

                similarities.append(max\_similarity)

            images\_count += 1

    if val\_type == "binary\_correct":

        output\_str += f"Угадано {correct\_guesses} / {images\_count} капч"

    elif val\_type == "similarity":

        output\_str += (

            f"Средняя схожесть: {statistics.fmean(similarities) \* 100}%"

        )

    with open(

        str(

            rel\_path(

                "results\_trained\_" + val\_type + "\_" + rec\_type + "\_" + dataset\_name + ".txt"

            )

        ),

        "w",

        encoding="utf-8",

    ) as f:

        f.write(output\_str)

def main():

    test\_recognition(

        "straight\_recognition", "binary\_correct", "dataset", show\_img=False

    )

    test\_recognition("straight\_recognition", "similarity", "dataset", show\_img=False)

    test\_recognition(

        "straight\_recognition", "binary\_correct", "dataset2", show\_img=False

    )

    test\_recognition("straight\_recognition", "similarity", "dataset2", show\_img=False)

    test\_recognition("avg\_of\_aug", "binary\_correct", "dataset2", show\_img=False)

    test\_recognition("avg\_of\_aug", "similarity", "dataset2", show\_img=False)

    test\_recognition(

        "filtered\_recognition", "binary\_correct", "dataset2", show\_img=False

    )

    test\_recognition("filtered\_recognition", "similarity", "dataset", show\_img=False)

    test\_recognition("filtered\_recognition", "similarity", "dataset2", show\_img=False)

    test\_recognition(

        "filtered\_recognition", "binary\_correct", "dataset", show\_img=False

    )

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Файл task6.py

import csv

import pathlib

import cv2

import re

import pytesseract

from pytesseract import Output

from difflib import SequenceMatcher

import statistics

pytesseract.pytesseract.tesseract\_cmd = r"C:/Program Files/Tesseract-OCR/tesseract.exe"

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

    path = pathlib.Path(\_\_file\_\_).parent / rel\_path

    return path

def clear\_str(str):

    str = str.lower()

    str = re.sub(

        r"[0123456789\!\#\$\%\^\&\\*\(\)\\_\~@\`\n\/\|\,\"\<\°\„\?\.\«\’\‚\”\“\®\¥\>\`\'\—\™\‘\:\ \]\[\{\}\=\+\-\\]",

        " ",

        str,

    )

    return str

def test\_recognition(rec\_type, val\_type, dataset\_name, show\_img=False):

    output\_str = ""

    labels = {}

    images\_count = 0

    correct\_guesses = 0

    similarities = []

    dict\_for\_avg\_of\_aug\_dataset = {}

    with open(

        str(rel\_path(dataset\_name + "/labels.csv")), newline="", encoding="utf-8"

    ) as csvfile:

        reader = csv.reader(csvfile, delimiter=",", quotechar="'")

        for row in reader:

            labels[row[0]] = row[1]

    img\_files = list(

        pathlib.Path(str(rel\_path(dataset\_name + "/"))).glob("\*.jpg")

    )

    for img\_file in img\_files:

        img = cv2.imread(str(img\_file.resolve()), 0)

        groud\_truth = labels[img\_file.name]

        if rec\_type == "straight\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

        elif rec\_type == "boxes\_recognition":

            h, w = img.shape

            boxes = pytesseract.image\_to\_boxes(img, lang="rus+eng")

            for box in boxes.splitlines():

                box\_data = box.split(" ")

                cv2.rectangle(

                    img,

                    (int(box\_data[1]), h - int(box\_data[2])),

                    (int(box\_data[3]), h - int(box\_data[4])),

                    (0, 255, 0),

                    2,

                )

            result = "".join([sym\_data.split(" ")[0] for sym\_data in boxes.split("\n")])

        elif rec\_type == "filtered\_recognition":

            result = pytesseract.image\_to\_string(img, lang="test4")

            groud\_truth = clear\_str(groud\_truth)

            result = clear\_str(result)

        elif rec\_type == "avg\_of\_aug":

            result = pytesseract.image\_to\_string(img, lang="test4")

            img\_name\_wo\_aug = img\_file.name.split("\_")[0]

            if img\_name\_wo\_aug in dict\_for\_avg\_of\_aug\_dataset:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug].append(result)

            else:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug] = [result]

        result = "".join(result.splitlines())

        output\_str += f"{img\_file.name} | {groud\_truth} | {result}\n"

        if val\_type == "binary\_correct":

            if (

                result.lower() in groud\_truth.lower()

                or groud\_truth.lower() in result.lower()

            ):

                correct\_guesses += 1

        elif val\_type == "similarity":

            similarity = SequenceMatcher(

                None, groud\_truth.lower(), result.lower()

            ).ratio()

            similarities.append(similarity)

        images\_count += 1

        print(result)

        if show\_img:

            cv2.imshow("capthca", img)

            cv2.waitKey()

    output\_str += "\n"

    if rec\_type == "avg\_of\_aug":

        images\_count = 0

        similarities = []

        correct\_guesses = 0

        for key in dict\_for\_avg\_of\_aug\_dataset.keys():

            results = dict\_for\_avg\_of\_aug\_dataset[key]

            if val\_type == "binary\_correct":

                for result in results:

                    if (

                        result.lower() in groud\_truth.lower()

                        or groud\_truth.lower() in result.lower()

                    ):

                        correct\_guesses += 1

                        break

            elif val\_type == "similarity":

                max\_similarity = 0

                for result in results:

                    similarity = SequenceMatcher(

                        None, groud\_truth.lower(), result.lower()

                    ).ratio()

                    max\_similarity = max(similarity, max\_similarity)

                similarities.append(max\_similarity)

            images\_count += 1

    if val\_type == "binary\_correct":

        output\_str += f"Угадано {correct\_guesses} / {images\_count} капч"

    elif val\_type == "similarity":

        output\_str += (

            f"Средняя схожесть: {statistics.fmean(similarities) \* 100}%"

        )

    with open(

        str(

            rel\_path("results\_trained\_" + val\_type + "\_" + rec\_type + "\_" + dataset\_name + ".txt"

            )

        ),

        "w",

        encoding="utf-8",

    ) as f:

        f.write(output\_str)

def main():

    test\_recognition("boxes\_recognition", "binary\_correct", "dataset", show\_img=False)

    test\_recognition("boxes\_recognition", "similarity", "dataset", show\_img=False)

    test\_recognition("boxes\_recognition", "binary\_correct", "dataset2", show\_img=False)

    test\_recognition("boxes\_recognition", "similarity", "dataset2", show\_img=False)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Файл task8.py

import csv

import pathlib

import cv2

import re

import pytesseract

import easyocr

from pytesseract import Output

from difflib import SequenceMatcher

import statistics

pytesseract.pytesseract.tesseract\_cmd = r"C:/Program Files/Tesseract-OCR/tesseract.exe"

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

    path = pathlib.Path(\_\_file\_\_).parent / rel\_path

    return path

def clear\_str(str):

    str = str.lower()

    str = re.sub(

        r"[0123456789\!\#\$\%\^\&\\*\(\)\\_\~@\`\n\/\|\,\"\<\°\„\?\.\«\’\‚\”\“\®\¥\>\`\'\—\™\‘\:\ \]\[\{\}\=\+\-\\]",

        " ",

        str,

    )

    return str

def test\_recognition(rec\_type, val\_type, dataset\_name, show\_img=False):

    output\_str = ""

    labels = {}

    images\_count = 0

    correct\_guesses = 0

    similarities = []

    dict\_for\_avg\_of\_aug\_dataset = {}

    with open(

        str(rel\_path(dataset\_name + "/labels.csv")), newline="", encoding="utf-8"

    ) as csvfile:

        reader = csv.reader(csvfile, delimiter=",", quotechar="'")

        for row in reader:

            labels[row[0]] = row[1]

    img\_files = list(

        pathlib.Path(str(rel\_path(dataset\_name + "/"))).glob("\*.jpg")

    )

    for img\_file in img\_files:

        img = cv2.imread(str(img\_file.resolve()), 0)

        groud\_truth = labels[img\_file.name]

        if rec\_type == "straight\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

        elif rec\_type == "boxes\_recognition":

            h, w = img.shape

            boxes = pytesseract.image\_to\_boxes(img, lang="rus+eng")

            for box in boxes.splitlines():

                box\_data = box.split(" ")

                cv2.rectangle(

                    img,

                    (int(box\_data[1]), h - int(box\_data[2])),

                    (int(box\_data[3]), h - int(box\_data[4])),

                    (0, 255, 0),

                    2,

                )

            result = "".join([sym\_data.split(" ")[0] for sym\_data in boxes.split("\n")])

        elif rec\_type == "filtered\_recognition":

            result = pytesseract.image\_to\_string(img, lang="test4")

            groud\_truth = clear\_str(groud\_truth)

            result = clear\_str(result)

        elif rec\_type == "avg\_of\_aug":

            result = pytesseract.image\_to\_string(img, lang="test4")

            img\_name\_wo\_aug = img\_file.name.split("\_")[0]

            if img\_name\_wo\_aug in dict\_for\_avg\_of\_aug\_dataset:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug].append(result)

            else:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug] = [result]

        elif rec\_type == "easy\_ocr":

            reader = easyocr.Reader(["en", "ru"])

            result\_easy = reader.readtext(img, detail=0, paragraph=True)

            result = ""

            for i in result\_easy:

                result += i

        result = "".join(result.splitlines())

        output\_str += f"{img\_file.name} | {groud\_truth} | {result}\n"

        if val\_type == "binary\_correct":

            if (

                result.lower() in groud\_truth.lower()

                or groud\_truth.lower() in result.lower()

            ):

                correct\_guesses += 1

        elif val\_type == "similarity":

            similarity = SequenceMatcher(

                None, groud\_truth.lower(), result.lower()

            ).ratio()

            similarities.append(similarity)

        images\_count += 1

        print(result)

        if show\_img:

            cv2.imshow("capthca", img)

            cv2.waitKey()

    output\_str += "\n"

    if rec\_type == "avg\_of\_aug":

        images\_count = 0

        similarities = []

        correct\_guesses = 0

        for key in dict\_for\_avg\_of\_aug\_dataset.keys():

            results = dict\_for\_avg\_of\_aug\_dataset[key]

            if val\_type == "binary\_correct":

                for result in results:

                    if (

                        result.lower() in groud\_truth.lower()

                        or groud\_truth.lower() in result.lower()

                    ):

                        correct\_guesses += 1

                        break

            elif val\_type == "similarity":

                max\_similarity = 0

                for result in results:

                    similarity = SequenceMatcher(

                        None, groud\_truth.lower(), result.lower()

                    ).ratio()

                    max\_similarity = max(similarity, max\_similarity)

                similarities.append(max\_similarity)

            images\_count += 1

    if val\_type == "binary\_correct":

        output\_str += f"Угадано {correct\_guesses} / {images\_count} капч"

    elif val\_type == "similarity":

        output\_str += (

            f"Средняя схожесть: {statistics.fmean(similarities) \* 100}%"

        )

    with open(

        str(

            rel\_path(

                "results\_" + val\_type + "\_" + rec\_type + "\_" + dataset\_name + ".txt"

            )

        ),

        "w",

        encoding="utf-8",

    ) as f:

        f.write(output\_str)

def main():

    test\_recognition("easy\_ocr", "binary\_correct", "dataset", show\_img=False)

    test\_recognition("easy\_ocr", "similarity", "dataset", show\_img=False)

    test\_recognition("easy\_ocr", "binary\_correct", "dataset2", show\_img=False)

    test\_recognition("easy\_ocr", "similarity", "dataset2", show\_img=False)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Файл task9.py

import csv

import pathlib

import cv2

import re

import pytesseract

import easyocr

from pytesseract import Output

from difflib import SequenceMatcher

import statistics

pytesseract.pytesseract.tesseract\_cmd = r"C:/Program Files/Tesseract-OCR/tesseract.exe"

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

    path = pathlib.Path(\_\_file\_\_).parent / rel\_path

    return path

def clear\_str(str):

    str = str.lower()

    str = re.sub(

        r"[0123456789\!\#\$\%\^\&\\*\(\)\\_\~@\`\n\/\|\,\"\<\°\„\?\.\«\’\‚\”\“\®\¥\>\`\'\—\™\‘\:\ \]\[\{\}\=\+\-\\]",

        " ",

        str,

    )

    return str

def test\_recognition(rec\_type, val\_type, dataset\_name, show\_img=False):

    output\_str = ""

    labels = {}

    images\_count = 0

    correct\_guesses = 0

    similarities = []

    dict\_for\_avg\_of\_aug\_dataset = {}

    with open(

        str(rel\_path(dataset\_name + "/labels.csv")), newline="", encoding="utf-8"

    ) as csvfile:

        reader = csv.reader(csvfile, delimiter=",", quotechar="'")

        for row in reader:

            labels[row[0]] = row[1]

    img\_files = list(

        pathlib.Path(str(rel\_path(dataset\_name + "/"))).glob("\*.jpg")

    )

    for img\_file in img\_files:

        img = cv2.imread(str(img\_file.resolve()), 0)

        groud\_truth = labels[img\_file.name]

        if rec\_type == "straight\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

        elif rec\_type == "boxes\_recognition":

            h, w = img.shape

            boxes = pytesseract.image\_to\_boxes(img, lang="rus+eng")

            for box in boxes.splitlines():

                box\_data = box.split(" ")

                cv2.rectangle(

                    img,

                    (int(box\_data[1]), h - int(box\_data[2])),

                    (int(box\_data[3]), h - int(box\_data[4])),

                    (0, 255, 0),

                    2,

                )

            result = "".join([sym\_data.split(" ")[0] for sym\_data in boxes.split("\n")])

        elif rec\_type == "filtered\_recognition":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

            groud\_truth = clear\_str(groud\_truth)

            result = clear\_str(result)

        elif rec\_type == "avg\_of\_aug":

            result = pytesseract.image\_to\_string(img, lang="rus+eng")

            img\_name\_wo\_aug = img\_file.name.split("\_")[0]

            if img\_name\_wo\_aug in dict\_for\_avg\_of\_aug\_dataset:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug].append(result)

            else:

                dict\_for\_avg\_of\_aug\_dataset[img\_name\_wo\_aug] = [result]

        elif rec\_type == "easy\_ocr":

            reader = easyocr.Reader(

                ["ru"],

                user\_network\_directory="train\_easy/user\_network",

                model\_storage\_directory="train\_easy/model",

                recog\_network="train\_easy",

            )

            result\_easy = reader.readtext(img, detail=0, paragraph=True)

            result = ""

            for i in result\_easy:

                result += i

        result = "".join(result.splitlines())

        output\_str += f"{img\_file.name} | {groud\_truth} | {result}\n"

        if val\_type == "binary\_correct":

            if (

                result.lower() in groud\_truth.lower()

                or groud\_truth.lower() in result.lower()

            ):

                correct\_guesses += 1

        elif val\_type == "similarity":

            similarity = SequenceMatcher(

                None, groud\_truth.lower(), result.lower()

            ).ratio()

            similarities.append(similarity)

        images\_count += 1

        print(result)

        if show\_img:

            cv2.imshow("capthca", img)

            cv2.waitKey()

    output\_str += "\n"

    if rec\_type == "avg\_of\_aug":

        images\_count = 0

        similarities = []

        correct\_guesses = 0

        for key in dict\_for\_avg\_of\_aug\_dataset.keys():

            results = dict\_for\_avg\_of\_aug\_dataset[key]

            if val\_type == "binary\_correct":

                for result in results:

                    if (

                        result.lower() in groud\_truth.lower()

                        or groud\_truth.lower() in result.lower()

                    ):

                        correct\_guesses += 1

                        break

            elif val\_type == "similarity":

                max\_similarity = 0

                for result in results:

                    similarity = SequenceMatcher(

                        None, groud\_truth.lower(), result.lower()

                    ).ratio()

                    max\_similarity = max(similarity, max\_similarity)

                similarities.append(max\_similarity)

            images\_count += 1

    if val\_type == "binary\_correct":

        output\_str += f"Угадано {correct\_guesses} / {images\_count} капч"

    elif val\_type == "similarity":

        output\_str += (

            f"Средняя схожесть: {statistics.fmean(similarities) \* 100}%"

        )

    with open(

        str(

            rel\_path(

                "results\_trained\_"+ val\_type + "\_" + rec\_type + "\_" + dataset\_name + ".txt"

            )

        ),

        "w",

        encoding="utf-8",

    ) as f:

        f.write(output\_str)

def main():

    test\_recognition("easy\_ocr", "binary\_correct", "dataset", show\_img=False)

    test\_recognition("easy\_ocr", "similarity", "dataset", show\_img=False)

    test\_recognition("easy\_ocr", "binary\_correct", "dataset2", show\_img=False)

    test\_recognition("easy\_ocr", "similarity", "dataset2", show\_img=False)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

Файл TesseractOCR\_test.py

import pytesseract

import cv2

pytesseract.pytesseract.tesseract\_cmd = r'C:\Program Files\Tesseract-OCR\tesseract.exe'

img = cv2.imread('dataset/6.jpg')

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

print(text)