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.А. Мазницкий

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 плюсы 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).

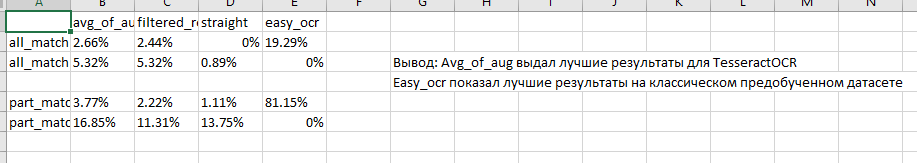


Рисунок 1 – сравнение моделей

**Листинг программ**

Файл 7\_1\_2.py

import os

import pytesseract

import easyocr

from PIL import Image

import glob

from calculate\_similarity\_score import calculate\_similarity\_score

# from augmentation import augment\_dataset

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

tesseract\_config = r'--oem 1 --psm 6'  # Настройки для TesseractOCR

reader = easyocr.Reader(['en', 'ru'])  # EasyOCR с поддержкой английского и русского

# Tesseract, запись результатов в файл аннотаций.

def annotate\_images(image\_paths, annotation\_file):

    annotations = {}

    for image\_path in image\_paths:

        img = Image.open(image\_path)

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

        annotations[image\_path] = text.strip()

    with open(annotation\_file, 'w', encoding='utf-8',errors='replace') as file:

        for image\_path, annotation in annotations.items():

            file.write(f"{image\_path}: {annotation}\n")

# tesseract

def straight\_recognition(image\_paths):

    predictions = {}

    for image\_path in image\_paths:

        img = Image.open(image\_path)

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

        predictions[image\_path] = text.strip()

    return predictions

# easyocr

def easyocr\_recognition(image\_paths):

    predictions = {}

    for image\_path in image\_paths:

        img = Image.open(image\_path)

        result = reader.readtext(image\_path)

        text = ' '.join([item[1] for item in result])

        predictions[image\_path] = text.strip()

    return predictions

def test\_recognition(rec\_type, val\_type, image\_paths, truth\_file, dpath):

    accuracy = 0  # Инициализация переменной accuracy

    if rec\_type == 'straight':

        predictions = straight\_recognition(image\_paths)

    elif rec\_type == 'easyocr':

        predictions = easyocr\_recognition(image\_paths)

    else:

        raise ValueError(f"Unsupported recognition type: {rec\_type}")

    ground\_truth = {}

    with open(truth\_file, 'r', encoding='utf-8') as file:

        for line in file:

            parts = line.split(':')

            if len(parts) >= 2:

                image\_path = parts[0].strip()

                true\_text = parts[1].strip()

                ground\_truth[image\_path] = true\_text

            else:

                # Обработка случая, когда в строке нет символа ':' или после ':' нет текста

                print(f"Invalid line format: {line}")

    # Оцениваем точность на основе указанного типа проверки

    if val\_type == 'full\_match':

        accuracy = evaluate\_accuracy\_wordwise(ground\_truth, predictions)

    if val\_type == "part\_match":

        accuracy = evaluate\_partial\_accuracy\_wordwise(ground\_truth, predictions, 0.7)

    # Сохраняем прогнозы в файл в кодировке UTF-8

    predictions\_file = f'{dpath}/{rec\_type}\_predictions.txt'

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

        for image\_path, prediction in predictions.items():

            file.write(f"{image\_path}: {prediction}\n")

        return accuracy

def evaluate\_accuracy\_wordwise(ground\_truth, predictions):

    correct = 0

    total = len(predictions)

    for image\_path, true\_text in ground\_truth.items():

        for img, txt in predictions.items():

            true\_words = set(true\_text.split())

            predicted\_words = set(txt.split())

            if true\_words == predicted\_words:

                correct += 1

    accuracy = correct / total

    return accuracy

# def evaluate\_accuracy\_wordwise(ground\_truth, predictions):

#     correct = 0

#     total = len(ground\_truth)

#     # total = len(predictions)

#

#     for image\_path, true\_text in ground\_truth.items():

#         # for img, txt in predictions.items():

#         predicted\_text = predictions.get(image\_path, '')

#         true\_words = set(true\_text.split())

#         predicted\_words = set(predicted\_text.split())

#         if true\_words == predicted\_words:

#             correct += 1

#

#     accuracy = correct / total

#     return accuracy

def evaluate\_partial\_accuracy\_wordwise(ground\_truth, predictions, threshold):

    correct = 0

    total = len(predictions)

    for image\_path, true\_text in ground\_truth.items():

        for img, txt in predictions.items():

            true\_words = set(true\_text.split())

            predicted\_words = set(txt.split())

            matched\_words = 0

            for true\_word in true\_words:

                for predicted\_word in predicted\_words:

                    if calculate\_similarity\_score(true\_word, predicted\_word) >= threshold:  # Пример использования порогового значения

                        matched\_words += 1

                        break

            if matched\_words == len(predicted\_words) and len(predicted\_words) == len(true\_words):

                correct += 1

    accuracy = correct / total

    return accuracy

def test\_augmented\_dataset(rec\_type, val\_type, augmented\_path, ground\_truth\_file, dpath):

        # Получяем список всех изображений в аугментированном датасете

        augmented\_images = glob.glob(os.path.join(augmented\_path, '\*.jpg'))

        # Тестируем распознавание на аугментированном датасете

        accuracy = test\_recognition(rec\_type, val\_type, augmented\_images, ground\_truth\_file, dpath)

        return accuracy

def main():

    image\_paths = ['capchi/1.jpg', 'capchi/2.jpg', 'capchi/3.jpg', 'capchi/4.jpg', 'capchi/5.jpg', 'capchi/6.jpg',

                   'capchi/7.jpg', 'capchi/8.jpg', 'capchi/9.jpg', 'capchi/10.jpg', 'capchi/11.jpg']

    true\_captcha\_txt = 'capchi/true\_text.txt'

    dpath = "capchi"

    print("\nPart Match")

    recognition\_type = 'easyocr'

    validation\_type = 'part\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    recognition\_type = 'straight'

    validation\_type = 'part\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    print("\nFull Match")

    recognition\_type = 'easyocr'

    validation\_type = 'full\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    recognition\_type = 'straight'

    validation\_type = 'full\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

def part2():

    # Путь к оригинальному датасету

    original\_dataset\_path = 'capchi'

    augmented\_dataset\_path = 'dataset2'

    ground\_truth\_file = 'dataset2/true\_text.txt'

    recognition\_type = 'easyocr'  # straight   easyocr

    validation\_type = 'full\_match'

    # # Аугментировать датасет

    # augment\_dataset(original\_dataset\_path, augmented\_dataset\_path)

    # Получяем список всех изображений в аугментированном датасете

    augmented\_images = glob.glob(os.path.join(augmented\_dataset\_path, '\*.jpg'))

    # Тестировать распознавание на аугментированном датасете

    accuracy\_augmented = test\_recognition(recognition\_type, validation\_type, augmented\_images,

                                                             ground\_truth\_file, augmented\_dataset\_path)

    print("\nFull Match")

    print(

        f"Точность для {recognition\_type} распознавание по дополненному набору данных: {accuracy\_augmented \* 100:.2f}%")

    recognition\_type = 'straight'  # straight   easyocr

    validation\_type = 'full\_match'

    # # Тестировать распознавание на аугментированном датасете

    accuracy\_augmented = test\_augmented\_dataset(recognition\_type, validation\_type, augmented\_dataset\_path,

                                                             ground\_truth\_file, augmented\_dataset\_path)

    print(

        f"Точность для {recognition\_type} распознавание по дополненному набору данных: {accuracy\_augmented \* 100:.2f}%")

    print("\nPart Match")

    recognition\_type = 'easyocr'

    validation\_type = 'part\_match'

    accuracy = test\_augmented\_dataset(recognition\_type, validation\_type, augmented\_dataset\_path,

                                                             ground\_truth\_file, augmented\_dataset\_path)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    recognition\_type = 'straight'

    validation\_type = 'part\_match'

    accuracy = test\_augmented\_dataset(recognition\_type, validation\_type, augmented\_dataset\_path,

                                                             ground\_truth\_file, augmented\_dataset\_path)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

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

#     main()

# main()

part2()

Файл 7\_3\_4.py

import os

import pathlib

import cv2

import pytesseract

import easyocr

from PIL import Image

import glob

from evaluate\_accuracy import evaluate\_accuracy\_wordwise, evaluate\_partial\_accuracy\_wordwise, \

    evaluate\_accuracy\_wordwise\_one, evaluate\_partial\_accuracy\_wordwise\_one

from clear\_str import clear\_str

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

tesseract\_config = r'--oem 1 --psm 6'  # Настройки для TesseractOCR

reader = easyocr.Reader(['en', 'ru'])  # EasyOCR с поддержкой английского и русского

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

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

    return path

# tesseract

def straight\_recognition(image\_paths):

    predictions = {}

    for image\_path in image\_paths:

        img = Image.open(image\_path)

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

        predictions[image\_path] = text.strip()

    return predictions

# easyocr

def easyocr\_recognition(image\_paths):

    predictions = {}

    for image\_path in image\_paths:

        img = Image.open(image\_path)

        result = reader.readtext(image\_path)

        text = ' '.join([item[1] for item in result])

        predictions[image\_path] = text.strip()

    return predictions

'''Новые методы rec\_type'''

def filtered\_recognition(img, groud\_truth):

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

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

    result = clear\_str(result)

    return result, groud\_truth

def avg\_of\_aug(img, img\_file, dict\_for\_avg\_of\_aug\_dataset):

    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]

'''-------------------------------------------------------------------'''

def test\_recognition(rec\_type, val\_type, image\_paths, truth\_file, dpath):

    accuracy = 0

    predictions = {}

    labels = {}

    groud\_truth = {}

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

    output\_str = ''

    images\_count = 0

    with open(truth\_file, 'r', encoding='utf-8') as file:

        for line in file:

            parts = line.split(':')

            if len(parts) >= 2:

                image\_path = parts[0].strip()

                true\_text = parts[1].strip()

                labels[image\_path] = true\_text

            else:

                # Обработка случая, когда в строке нет символа ':' или после ':' нет текста

                print(f"Invalid line format: {line}")

    img\_files = list(

        pathlib.Path(str(rel\_path(dpath))).glob("\*.jpg")

    )

    if rec\_type == 'straight':

        predictions = straight\_recognition(image\_paths)

    elif rec\_type == 'easyocr':

        predictions = easyocr\_recognition(image\_paths)

    elif rec\_type == 'filtered\_recognition' or rec\_type == 'avg\_of\_aug':

        for img\_file in img\_files:

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

            groud\_truth = labels[f'{dpath}/{img\_file.name}']

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

                result, groud\_truth = filtered\_recognition(img, groud\_truth)

            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 == 'full\_match':

                accuracy += evaluate\_accuracy\_wordwise\_one(groud\_truth.lower(), result.lower())

            elif val\_type == 'part\_match':

                accuracy += evaluate\_partial\_accuracy\_wordwise\_one(groud\_truth.lower(), result.lower(), 0.7)

            images\_count += 1

            print(result)

        output\_str += "\n"

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

            images\_count = 0

            similarities = []

            accuracy = 0

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

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

                if val\_type == "full\_match":

                    for result in results:

                        accuracy += evaluate\_accuracy\_wordwise\_one(groud\_truth.lower(), result.lower())

                elif val\_type == "part\_match":

                    accuracy += evaluate\_partial\_accuracy\_wordwise\_one(groud\_truth.lower(), result.lower(), 0.7)

                images\_count += 1

        if val\_type == "full\_match":

            output\_str += f"Статистика: угадано {accuracy}  капч"

        elif val\_type == "part\_match":

            output\_str += (

                f"Статистика: средняя схожесть: {accuracy / images\_count \* 100:.2f}%"

            )

        with open(

                str(

                    rel\_path(

                        "results\_" + val\_type + "\_" + rec\_type + "\_" + dpath + ".txt"

                    )

                ),

                "w",

                encoding="utf-8",

        ) as f:

            f.write(output\_str)

    else:

        raise ValueError(f"Unsupported recognition type: {rec\_type}")

    groud\_truth = labels

    # Оцениваем точность на основе указанного типа проверки

    if rec\_type == 'straight' or rec\_type == 'easyocr':

        if val\_type == 'full\_match':

            accuracy = evaluate\_accuracy\_wordwise(groud\_truth, predictions)

        if val\_type == "part\_match":

            accuracy = evaluate\_partial\_accuracy\_wordwise(groud\_truth, predictions, 0.7)

        # Сохраняем прогнозы в файл в кодировке UTF-8

        predictions\_file = f'{dpath}/{rec\_type}\_predictions.txt'

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

            for image\_path, prediction in predictions.items():

                file.write(f"{image\_path}: {prediction}\n")

            return accuracy

def main():

    image\_paths = ['capchi/1-.jpg', 'capchi/2-.jpg', 'capchi/3-.jpg', 'capchi/4-.jpg', 'capchi/5-.jpg', 'capchi/6-.jpg',

                   'capchi/7-.jpg', 'capchi/8-.jpg', 'capchi/9-.jpg', 'capchi/10-.jpg', 'capchi/11-.jpg']

    true\_captcha\_txt = 'capchi/true\_text.txt'

    dpath = "capchi"

    print("\nPart Match")

    recognition\_type = 'easyocr'

    validation\_type = 'part\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    recognition\_type = 'straight'

    validation\_type = 'part\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    print("\nFull Match")

    recognition\_type = 'easyocr'

    validation\_type = 'full\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    recognition\_type = 'straight'

    validation\_type = 'full\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, dpath)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

def part2():

    # Путь к оригинальному датасету

    original\_dataset\_path = 'capchi'

    augmented\_dataset\_path = 'dataset2'

    ground\_truth\_file = 'dataset2/true\_text.txt'

    recognition\_type = 'easyocr'  # straight   easyocr

    validation\_type = 'full\_match'

    # Получяем список всех изображений в аугментированном датасете

    augmented\_images = glob.glob(os.path.join(augmented\_dataset\_path, '\*.jpg'))

    # Тестировать распознавание на аугментированном датасете

    accuracy\_augmented = test\_recognition(recognition\_type, validation\_type, augmented\_images,

                                          ground\_truth\_file, augmented\_dataset\_path)

    print("\nFull Match")

    print(

        f"Точность для {recognition\_type} распознавание по дополненному набору данных: {accuracy\_augmented \* 100:.2f}%")

    recognition\_type = 'straight'  # straight   easyocr

    validation\_type = 'full\_match'

    # # Тестировать распознавание на аугментированном датасете

    accuracy\_augmented = test\_recognition(recognition\_type, validation\_type, augmented\_images,

                                          ground\_truth\_file, augmented\_dataset\_path)

    print(

        f"Точность для {recognition\_type} распознавание по дополненному набору данных: {accuracy\_augmented \* 100:.2f}%")

    print("\nPart Match")

    recognition\_type = 'easyocr'

    validation\_type = 'part\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, augmented\_images,

                                ground\_truth\_file, augmented\_dataset\_path)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

    recognition\_type = 'straight'

    validation\_type = 'part\_match'

    accuracy = test\_recognition(recognition\_type, validation\_type, augmented\_images,

                                ground\_truth\_file, augmented\_dataset\_path)

    print(f"Точность для {recognition\_type} распознавания: {accuracy \* 100:.2f}%")

def part34():

    recognition\_type = 'filtered\_recognition'

    recognition\_type2 = 'avg\_of\_aug'

    validation\_type = 'part\_match'

    validation\_type2 = 'full\_match'

    augmented\_dataset\_path = 'dataset2'

    ground\_truth\_file = 'dataset2/labels.txt'

    augmented\_images = glob.glob(os.path.join(augmented\_dataset\_path, '\*.jpg'))

    # test\_recognition(recognition\_type, validation\_type, augmented\_images, ground\_truth\_file, augmented\_dataset\_path)

    test\_recognition(recognition\_type2, validation\_type2, augmented\_images, ground\_truth\_file,

                     augmented\_dataset\_path)

# main()

# part2()

part34()

Файл 7\_8.py

import os

import pathlib

import cv2

import pytesseract

import easyocr

from PIL import Image

import glob

from evaluate\_accuracy import evaluate\_accuracy\_wordwise, evaluate\_partial\_accuracy\_wordwise, \

    evaluate\_accuracy\_wordwise\_one, evaluate\_partial\_accuracy\_wordwise\_one

from clear\_str import clear\_str

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

tesseract\_config = r'--oem 1 --psm 6'  # Настройки для TesseractOCR

reader = easyocr.Reader(['en', 'ru'])  # EasyOCR с поддержкой английского и русского

# Построить абсолютный путь до файла относительно местоположения скрипта

def rel\_path(rel\_path):

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

    return path

# tesseract

def straight\_recognition(image\_path):

    img = Image.open(image\_path)

    text = pytesseract.image\_to\_string(img, config=tesseract\_config, lang="train/test4")

    predictions = text.strip()

    return predictions

# easyocr

def easyocr\_recognition(image\_path):

    result = reader.readtext(image\_path, detail=70)

    text = ' '.join([item[1] for item in result])

    predictions = text.strip()

    return predictions

'''Новые методы rec\_type'''

def filtered\_recognition(img, groud\_truth):

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

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

    result = clear\_str(result)

    return result, groud\_truth

def avg\_of\_aug(img, img\_file, dict\_for\_avg\_of\_aug\_dataset):

    result = pytesseract.image\_to\_string(img, lang="train/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]

'''-------------------------------------------------------------------'''

def test\_recognition(rec\_type, val\_type, image\_paths, truth\_file, dpath):

    accuracy = 0

    # predictions = {}

    labels = {}

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

    output\_str = ''

    images\_count = 0

    with open(truth\_file, 'r', encoding='utf-8') as file:

        for line in file:

            parts = line.split(':')

            if len(parts) >= 2:

                image\_path = parts[0].strip()

                true\_text = parts[1].strip()

                labels[image\_path] = true\_text

            else:

                # Обработка случая, когда в строке нет символа ':' или после ':' нет текста

                print(f"Invalid line format: {line}")

    img\_files = list(

        pathlib.Path(str(rel\_path(dpath))).glob("\*.jpg")

    )

    if rec\_type == 'filtered\_recognition' or rec\_type == 'avg\_of\_aug' or 'straight' or rec\_type == 'easyocr':

        for img\_file in img\_files:

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

            groud\_truth = labels[f'{dpath}/{img\_file.name}']

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

                result, groud\_truth = filtered\_recognition(img, groud\_truth)

            elif rec\_type == 'easyocr':

                result = easyocr\_recognition(img)

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

                result = pytesseract.image\_to\_string(img, lang="train/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 == 'straight':

                result = straight\_recognition(img\_file)

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

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

            if val\_type == 'full\_match':

                accuracy += evaluate\_accuracy\_wordwise\_one(groud\_truth.lower(), result.lower())

            elif val\_type == 'part\_match':

                accuracy += evaluate\_partial\_accuracy\_wordwise\_one(groud\_truth.lower(), result.lower(), 0.7)

            images\_count += 1

            print(result)

        output\_str += "\n"

        if val\_type == "full\_match":

            output\_str += f"Точность для {rec\_type} распознавание по набору данных {dpath}: {accuracy/images\_count \* 100:.2f}%"

        elif val\_type == "part\_match":

            output\_str += (

                f"Точность для {rec\_type} распознавание по набору данных {dpath}: {accuracy / images\_count \* 100:.2f}%"

            )

        with open(

                str(

                    rel\_path(

                        "results\_" + val\_type + "\_" + rec\_type + "\_" + dpath + ".txt"

                    )

                ),

                "w",

                encoding="utf-8",

        ) as f:

            f.write(output\_str)

    else:

        raise ValueError(f"Unsupported recognition type: {rec\_type}")

    # groud\_truth = labels

    # # Оцениваем точность на основе указанного типа проверки

    # if rec\_type == 'straight' or rec\_type == 'easyocr':

    #     if val\_type == 'full\_match':

    #         accuracy = evaluate\_accuracy\_wordwise(groud\_truth, predictions)

    #     if val\_type == "part\_match":

    #         accuracy = evaluate\_partial\_accuracy\_wordwise(groud\_truth, predictions, 0.7)

    #     # Сохраняем прогнозы в файл в кодировке UTF-8

    # predictions\_file = f'{dpath}/{rec\_type}\_predictions\_on\_train.txt'

    # with open(predictions\_file, 'w', encoding='utf-8') as file:

    #     for image\_path, prediction in predictions.items():

    #         file.write(f"{image\_path}: {prediction}\n")

def on\_train():

    recognition\_type = 'easyocr'

    validation\_type = 'part\_match'

    validation\_type2 = 'full\_match'

    augmented\_dataset\_path = 'dataset2'

    original\_dataset\_path = 'capchi'

    ground\_truth\_file\_old = 'dataset2/true\_text.txt'

    true\_captcha\_txt = 'capchi/true\_text.txt'

    ground\_truth\_file = 'dataset2/labels.txt'

    augmented\_images = glob.glob(os.path.join(augmented\_dataset\_path, '\*.jpg'))

    image\_paths = glob.glob(os.path.join(original\_dataset\_path, '\*.jpg'))

    ########################################## dataset2 #############################################################

    # filtered\_recognition

    test\_recognition(recognition\_type, validation\_type, augmented\_images, ground\_truth\_file, augmented\_dataset\_path)

    test\_recognition(recognition\_type, validation\_type2, augmented\_images, ground\_truth\_file, augmented\_dataset\_path)

    ###################################### dataset1 (capchi) #######################################################

    test\_recognition(recognition\_type, validation\_type, image\_paths, true\_captcha\_txt, original\_dataset\_path)

    test\_recognition(recognition\_type, validation\_type2, image\_paths, true\_captcha\_txt, original\_dataset\_path)

on\_train()