Задание первого этапа конкурса OCR. Искусственный интеллект.

OpenCV + OCR. Извлекаем признаки

Описание:

2320 год, Вы и ваши друзья исследуете руины в одной из центральных зон прошлого.
Попадая в лабораторию в одном здании руин, вы замечаете, что одна из дверей должна вести в глубь лаборатории, но она заблокирована.
Рядом с дверью вы находите компьютер и флешку с разными наборами данных предыдущей цивилизации, просмотрев часть открытых данных, вы пришли к выводу, что данные представляют из себя определенные типы документов из прошлого, которые имеют определенные отличия, но, на ваш взгляд, все документы представляют из себя изображения с характерными параметрами.

Научимся работать с документами с помощью библиотек компьютерного зрения и оптического распознавания символов. Попробуем автоматизировать задачу быстрого просмотра документов, превзойти в скорости просмотра файлов перебор и просмотр файлов человеком. Для этого извлечём из файлов полезную информацию, на основе которой пользователь в ходе ручного просмотра файлов выделяет основные признаки документа.

На данном этапе абстрагируемся от типа документа. Представим, что все файлы имеют случайное название и лежат в одном каталоге на локальном диске компьютера. Все файлы являются сканами в форматах pdf или png, то есть заранее неизвестно даже по расширению, с каким из документов мы работаем. Все документы одностраничные, то есть нет необходимости создавать массив признаков для каждой страницы отдельно. Для того, чтобы научиться с помощью кода понимать, что за файл перед нами, для начала необходимо научиться извлекать базовые признаки документа, которые помогут отличить один документ от другого.

Задача:

Написать код и реализовать в коде функцию. Сигнатура и контракт метода реализации, то есть имя функции, перечень входных переменных и возвращаемое значение должны строго совпадать с примером из раздела ожидаемый результат.
Функция должна возвращать строго dict объект следующего вида с ключами и типами значений:

```
result = {
                    'red_areas_count': int,  # количество красных участков (штампы, печати и т.д.) на скане
                    'blue_areas_count': int,  # количество синих областей (подписи, печати, штампы) на скане
                    'text_main_title': str,  # текст главного заголовка страницы или ""
                    'text_block': str,  # текстовый блок параграфа страницы, только первые 10 слов, или ""
```

'table_cells_count': int, # уникальное количество ячеек (сумма количеств ячеек одной или более таблиц)
}

Подробнее об обучающей выборке:
Перед вами 15 изображений. Следующие утверждения верны для каждого документа:
1. На изображении могут присутствовать или отсутствовать синие и красные области.
2. Количество синих и красных областей специальным образом не задается, их может быть несколько каждого типа.
3. В файле могут присутствовать или отсутствовать отдельно параграфы текста, а также таблицы.
4. Все документы одностраничные. Также, вы находите подсказку, что для разблокирования двери необходимо ввести сколько синих и красных областей на каждом документе, текстовые блоки и количество уникальных ячеек и уточнения по расчёту:
5. Объединенные печать и подпись, печать и печать, печать и штамп одного оттенка (в оттенках синего или красного) являются одним контуром (подразумевается явное наложение или сильное пересечение).
6. Пересечение красной и синей печатей, штампов или подписи не является одним контуром участка.
7. Шаблоны форм, в которых есть одновременно слова, линии и спец разметка параграфами текста не считаются.
8. Заголовок может находиться в любой части документа. Заголовок всегда расположен примерно по середине страницы (отцентрирован относительно полей документа).
9. Под текстом заголовка для простоты валидации ответов подразумевается одна строка.
10. Из текста параграфа (если подобный присутствует) для валидации необходимо распознавать и сохранять только первые 10 слов, включая предлоги, союзы. Знаки препинания не учитываются.
11. Если на странице присутствует несколько параграфов с текстом, необходимо распознавать только первый параграф (подразумевается от верхней границы документа).
12. Необходимо правильно посчитать суммарное количество ячеек для всех таблиц на странице. На странице может быть несколько таблиц или отсутствовать вовсе. На введение кодов доступа у вас есть 10 часов и 3 попытки, исходя из чего Вы понимаете, разблокировать дверь можно только с помощью программного кода, который поможет сделать это быстрее, чем Вы будете делать это со своими друзьями вручную, ведь документов огромное количество.
Подробнее см. в json файлах разметки обучающей выборки.

Используемые технологии:
1. Язык программирования: Python, версия >= 3.7
2. Закрепленные в requirement.txt необходимые для работы библиотеки
3. Все библиотеки должны иметь возможность быть установленными через
pip3 install --no-cache-dir -q -r requirements.txt

Ожидаемый формат результата:
В результате выполнения задания ожидается получить:
1. .py файл с кодом, в котором задана функция

```
def extract_doc_features(filepath: str) -> dict:
    """
    Функция, которая будет вызвана для получения признаков документа, для которого задан:
    :param filepath: абсолютный путь до тестового файла на локальном компьютере (строго pdf или png).

    :return: возвращаемый словарь, совпадающий по составу и написанию ключей условию задачи
    """
    # ваша реализация функции ниже
    pass

    return result_dict
```

2. файл requirements.txt, который в виртуальной среде может быть создан командой pip3 freeze > requirements. txt

Файлы, отправленные в виде результата на конкурс, будут запущены в виртуальной автоматизированной среде на основе контейнеров docker. Docker контейнер будет отрезан от внешней сети Интернет, сделать запрос для расширения данных из кода будет невозможно.

На операционной системе, на которой будет проводиться валидация результатов выполнена установка библиотеки tesseract для решения задач OCR:

RUN apt-get install -y tesseract-ocr-rus.

Метрика качества:

p (precision) — точность определения признаков для одного документа, по 1 баллу за каждый верный признак, всего признаков — M,

p_i = TruePositive_i / M # точность определения признаков;

sum((p_i)^2) (sum of precisions) - сумма квадратов precisions по всем документам тестовой выборки.

Итоговая метрика: sum((p_i)^2)

Метрики признаков:

Количественные признаки с типом данных int или float — требуется абсолютное совпадение.

Текстовые признаки считаются верно заполненными при совпадении по Левенштейну до заданного уровня threshold от самой длинной из строк (Levenshtein.distance(s1, s2) / len(longest_str) < 1.0 - threshold.

Подробнее о метрике:

https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D0%B5_%D0%9B%D0%B5%D0%B2%D0%B5%D0%BD%D1%88%D1%82%D0%B5%D0%B9%D0%BD%D0%B0

Распределение баллов, пример подсчёта:
В настоящем задании не требуется дополнительное описание алгоритмов и решения.
Итого в тестовой выборке будет представлено N объектов.
Всего признаков для каждого объекта: M.
Максимальная точность для одного объекта: M / M = 1.

Смысл метрики — поощрять за большее кол-во правильных признаков в рамках одного документа.
То есть, есть, например, 2 документа (N), по 5 признаков (M) в каждом. Рассмотрим два случая:
1) Правильно для первого: 2 из 5 (p_1), для второго: 2 из 5 (p_2).
2) Правильно для первого: 4 из 5 (p_1), для второго: 0 из 5 (p_2).
Итоговые метрики:
1) (2 / 5)^2 + (2 / 5)^2 = 4 / 25 + 4 / 25 = 8 / 25
2) (4 / 5)^2 + (0 / 5)^2 = 16 / 25 + 0 / 25 = 16 / 25

Второй вариант оказывается выше в рейтинге, хотя формальное кол-во выявленных правильно признаков у них равно.