|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_ ***ИУ-КФ «Информатики и управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**КАФЕДРА** \_\_***ИУ5-КФ «Системы обработки информации»***\_\_\_\_\_\_\_\_\_\_

**НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА**

**«Исследование технологии Tesseract для решения задач распознавания текста»**

Студент гр.САПР.Б-71 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Тимченко Д.К. )

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Кириллов В.Ю. )

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка проекта \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2019

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc27048763)

[1. История развития и архитектура Tesseract 4](#_Toc27048764)

[2. Распознавание линий и слов 6](#_Toc27048765)

[3. Подбор базисной линии 7](#_Toc27048766)

[4. Статическая классификация символов 9](#_Toc27048767)

[5. Лингвистический анализ 11](#_Toc27048768)

[6. Дальнейшее развитие 12](#_Toc27048769)

[ЗАКЛЮЧЕНИЕ 14](#_Toc27048770)

ВВЕДЕНИЕ

Оптическое распознавание символов (англ. Optical Character Recognition – OCR) – это технология, которая позволяет преобразовывать различные типы документов, такие как отсканированные документы, PDF-файлы или фото с цифровой камеры, в редактируемые форматы с возможностью поиска[4].

Предположим, у вас есть бумажный документ, например, статья в журнале, брошюра или договор в формате PDF, присланный вам партнером по электронной почте. Очевидно, для того чтобы получить возможность редактировать документ, его недостаточно просто отсканировать. Единственное, что может сделать сканер, – это создать изображение документа, представляющее собой всего лишь совокупность черно-белых или цветных точек, то есть растровое изображение.

Для того чтобы копировать, извлекать и редактировать данные, вам понадобится программа для распознавания символов, которая сможет выделить в изображении буквы, составить их в слова, а затем объединить слова в предложения, что в дальнейшем позволит работать с содержимым исходного документа.

Решением поставленной задачи будет использование OCR движка Tesseract, позволяющего распознавать текст на сканированных копиях бумажных документов. Целью этой работы является исследование технологии Tesseract для решения такого типа задач.

Среди задач этого исследования можно выделить:

* Изучение истории развития Tesseract
* Изучение особенностей его ранней реализации
* Изучение его современных возможностей

1. История развития и архитектура Tesseract

Tesseract – это открытый OCR-движок, который был разработан в HP в 1984-1994 годах. Впервые о нем заявили в рамках исследования UNLV Annual Test of OCR Accuracy 1995 года, где он показал лучшие результаты, после чего разработка вновь была скрыта, пока в 2005 году компания Google не выкупила технологию и не сделала ее открытой.

Он был разработан в рамках аспирантской диссертации в HP Labs, Бристоль, и впоследствии был дополнительным программным обеспечением к производимым корпорацией сканнерам. Решение разработать такой продукт было принято из-за того, что тогда системы OCR только зарождались и не могли обработать ничего, кроме сканированных копий высочайшего качества[2].

Для разработки этого проекта объединились два подразделения компании: исследовательский модуль – HP Labs – и отдел производства сканнеров в Колорадо. Результатом их работы стал OCR-движок, показывавший на тот момент беспрецедентные показатели точности среди коммерческих проектов подобного рода, но отдельным продуктом он так и не стал. Следующим этапом работы над Tesseract была попытка его применения для сжатия графической информации. Работа сосредоточилась больше не над повышением точности, а над увеличением отказоустойчивости. Проект был разработан к концу 1994 года, после чего его развитие почти полностью завершилось. В конце 2005 года HP выпустили Tesseract как открытое программное обеспечение, дальнейшим развитием проекта занимается Google[1].

Так как HP имела собственный инструмент анализа разметки страницы, который уже использовался в их продуктах (и не был издан под открытой лицензией), Tesseract никогда не нуждался в собственной системе анализа страницы. Несмотря на это, при использовании Tesseract предполагается использовать в качестве входной информации бинарное изображение с оптически определенными текстовыми областями.

Обработка изображения сводилась к обычному конвейеру, однако некоторые шаги этого алгоритма были необычными для того времени. Первым шагом алгоритма являлся анализ связных компонентов, по завершению которого программа получала контуры таких компонентов. На тот момент такое решение было достаточно неэффективным, зато имело значительное преимущество: получение послойной структуры контуров фигур и их потомков позволило легко узнавать инвертированный текст и распознавать его, как если бы он был написан стандартно – черным по белому. Tesseract был первым OCR движком, позволившим распознавать такие тексты так просто. На этом этапе контуры фигур организуются в массивы двоичных данных, в дальнейшем – blobs [2].

Они в свою очередь организуются в текстовые линии и блоки для дальнейшего анализа на равную или пропорциональную высоту текста. Линии текста разбиваются на слова в зависимости от типов пробельных символов в линии. Текст фиксированной высоты разбивается на символьные ячейки сразу же, текст пропорциональной высоты разбивается на слова с помощью пробелов разной четкости.

Следующий шаг распознавания проходит в два прохода: на первом этапе производится попытка распознать каждое слово по очереди. Каждое слово передается классификатору в качестве обучающих данных. После этого происходит повышение точности распознаваемых символов далее на изображении[3].

Так как первая попытка распознавания могла оказаться неудачной и классификатор не смог качественно обучиться до конца файла, необходим второй проход по файлу. На этом этапе слова, которые не были распознаны корректно в первый раз, могут быть распознаны повторно с большей точностью.

Наконец, на последнем шаге алгоритма распознаются нечеткие символы и проверяются альтернативные гипотезы о расположении небольших текстовых фрагментов по оси х[1].

1. Распознавание линий и слов

Алгоритм нахождения линий – это одна из немногих частей Tesseract, которые были опубликованы. Этот алгоритм реализован таким образом, что позволяет распознавать страницы, содержащие перекос. При этом не возникает необходимости в введении системы, выравнивающей изображение, а также не происходит потери качества изображения вследствие ее работы. Ключевыми этапами работы являются фильтрация бинарных массивов и выстраивание линий.

Предположим, что уже был произведен анализ расположения элементов на листе, и были получены фрагменты текста более или менее универсального размера. После этого к данным применятся фильтр по высоте в результате чего удаляются символы, имеющие надстрочные и подстрочные элементы, а также буквицы. Высчитывается приближенное значение средней высоты буквы, после чего находятся элементы меньше, чем это высота. Не трудно догадаться, что такими элементами будут являться знаки пунктуации, транскрипции или шум[5].

Отсортированные таким образом двоичные массивы более вероятно подойдут под модель не перекрывающихся, параллельных, но косых линий. Сортировка и обработка массивов по координате х делает возможным сборку блоков в уникальные текстовые линии с отслеживанием величины наклона на странице, что приводит к значительному снижению вероятности приведения блока к неподходящей линии из-за скоса. После того, как отфильтрованные массивы собираются в линии, применяется метод наименьших квадратов для оценки базовой линии строки, после чего свои места занимают отфильтрованные до этого двоичные массивы.

Последним шагом процесса создания линий является слияние массивов, которые перекрываются горизонтально хотя бы на половину, расстановка символов ударений на правильные буквы, если это необходимо, а также распознавание частей поврежденных символов[1].

1. Подбор базисной линии

После того, как были определены текстовые линии, с помощью квадратичной кривой более точно определяются базисные линии. Этот подход также был впервые применен в Tesseract и позволил ему обрабатывать страницы с неровными базисными линиями, что является распространенной проблемой при сканировании книг и не только[4].

Базисные линии определяются путем разбиения двоичных массивов на группы с предсказуемым смещением от начальной базисной прямой. После чего применяется квадратичный сплайн для подбора самого отклоняющегося региона с помощь метода наименьших квадратов. Такой подход имеет как преимущества, так и недостатки. К положительным сторонам можно отнести то, что вычисления достаточно постоянны, а к недостаткам – вычисления прерываются, когда требуется применение нескольких сегментов сплайнов. Более традиционный для решения подобных задач кубический сплайн не имеет этого недостатка[3].

В тексте выделяются 4 основные линии: базисная, линия подстрочного элемента, линия высоты строчной буквы и линия надстрочного элемента. Все эти линии можно назвать условно «параллельными», так как расстояние между линиями по оси у сохраняется.

Tesseract проверяет линии текста для того, чтобы определить, фиксированной ли величины текст. Там, где такой фрагмент найти возможно, Tesseract разбивает сова на символы, используя информацию о высоте и скрывает эти слова от дальнейшего разбивания и распознавания на следующем этапе работы[2].

Распознавание текстов не фиксированной величины, а также текстов, содержащих пропорциональные пробелы, является нетривиальной задачей. Пример приведен на рисунке 1. Промежуток между разрядами десятков и единиц в надписи «11.9%» такого же размера, как и обычный пробел и даже больше, чем отступ между словами erated и junk. Горизонтальный промежуток между ограничивающими рамками of и financial и вовсе отсутствует. Tesseract решает большинство этих проблем, измеряя промежутки в ограниченном вертикальном диапазоне между базисной линией и линией высоты строчной буквы. Пробелы, близкие к пороговому значению на этом этапе помечаются как «нечеткие», финальное решение принимается после этапа распознавания текста[1].

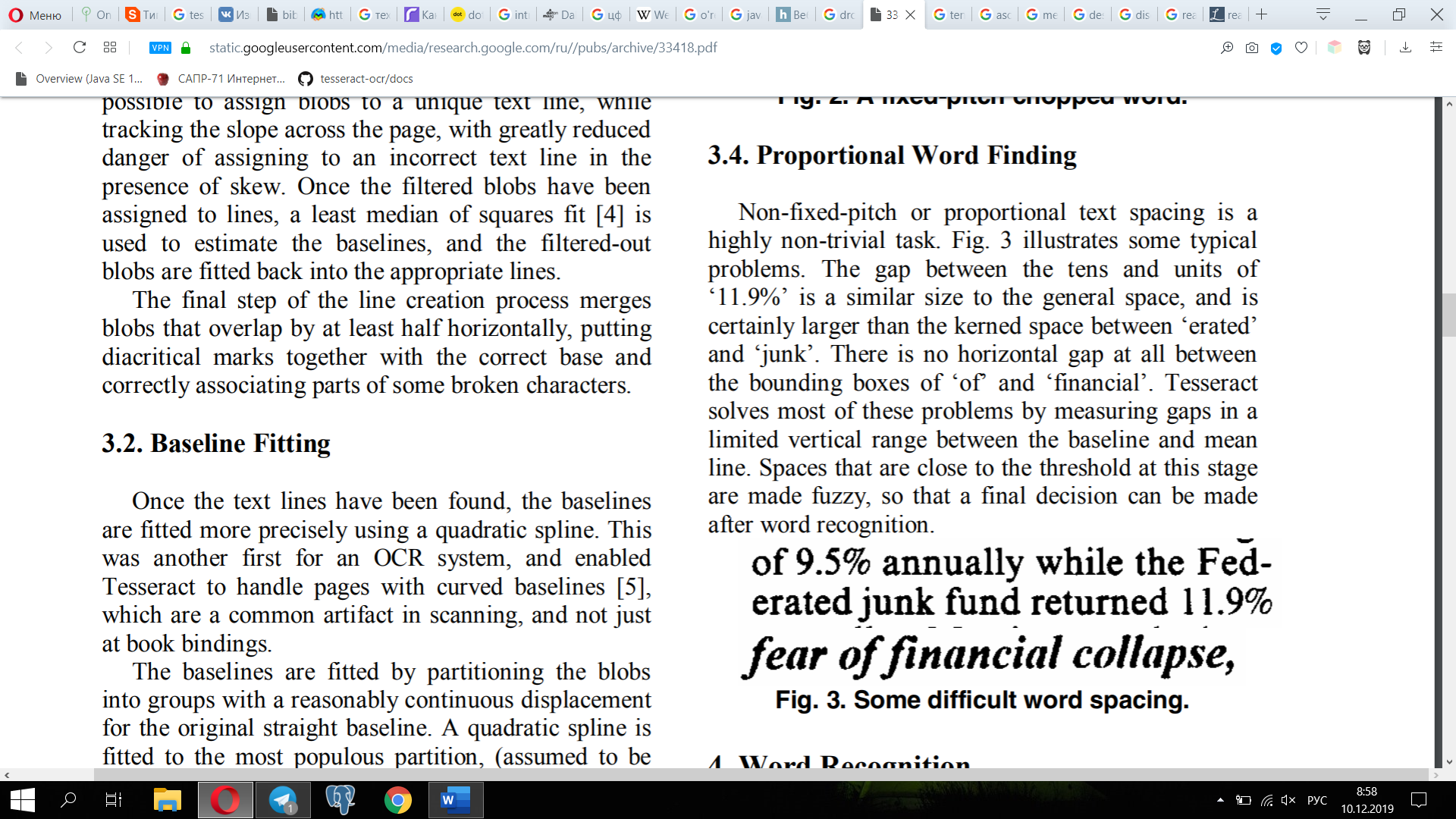


Рисунок 1 – Нетривиальная задача распознавания

Частью процесса распознавания для каждого движка системы распознавания текста является определение, каким образом слово можно разбить на буквы. Начальная сегментация была получена из первого этапа создания линий. Остальные этапы распознавания применяются только к словам разной величины.

На следующем этапе Tesseract пытается улучшить результаты, разбивая двоичные массивы, распознать промежутки между которыми не удалось. После этого он передает их классификатору для распознавания опорных точек буквы с помощью полигональной аппроксимации. Может потребоваться до трех пар точек, чтобы отделить слившиеся буквы с использованием таблицы ASCII[2].

Если механизм назначения основных точек оказался бесполезен и при этом слово все еще недостаточно хорошо распознано, оно передается ассоциатору. Он осуществляет поиск по сегментационному графу всех возможных комбинаций максимально разбитых по опорным точкам бинарных массивов. Это происходит даже без построения настоящего сегментационного графа, но вместо этого создается и поддерживается hash-таблица посещенных состояний. Этот поиск происходит за счет проталкивания кандидата нового состояния из приоритетной очереди и оценивания их через классификацию комбинаций фрагментов.

Можно поспорить, что подход с предварительным разбиением и последующим ассоциированием максимально неэффективен, более всего предрасположен к упущению важных фрагментов, и что все это может привести к неработоспособности приложения. Несмотря на это алгоритм упрощает структуры данных, которые потребуются для обеспечения полного графа сегментации[3].

Когда в 1989 году был реализован первый сегментационный поиск, точность Tesseract в распознавании поврежденных символов опережала коммерческие аналоги того времени. Значительной частью успеха стало то, что классификатор мог легко распознавать битые символы[2].

1. Статическая классификация символов

Первые версии Tesseract использовали топологические фрагменты научных работ, посвященных распознаванию. Благодаря этому система получила довольно хорошую независимость от стиля шрифта и его размера. Несмотря на это, система не могла показать высокую эффективность при работе с изображениями из реальной жизни. Также этот подход не работает с разбитыми символами, то есть когда разные части буквы отделяются друг от друга расстоянием. В таких случаях система считала это двумя разными символами[2].

Прорывной идеей в сложившейся ситуации было то, что черты неизвестных символов не должны быть общими с характеристиками тренировочных данных. На протяжении обучения сегменты полигональной аппроксимации используются для получения характеристик, но в распознавании характеристических точек маленьких фрагментов текста фиксированной длины (нормализованных юнитов) извлекаются из контуров и сопоставляются методом многие-к-одному с кластерным прототипом характеристических точек тренировочных данных[1].

Один прототип объединяет две совершенно не сопоставленные части. Главной проблемой остается вычислительная мощность расчета дистанции между неизвестными символами и прототипами. Характеристические точки, извлеченные из нераспознанных символов, к тому же, хранят три измерения (координаты и угол), а прототип содержит обычно 50-100 характеристических точек в символе и хранит 4 измерения (координаты, угол и длину), с 10-20 характеристиками.

Классификация происходит в два прохода. На первом шаге происходит сокращение классов символов, с которыми неизвестные символы могут совпасть. Для каждой характеристической точки из уточняющей таблицы, содержащей три измерения, составляется битовый вектор классов, с которыми символ может совпасть, и все такие вектора суммируются по всем параметрам. Классы с самым высоким показателем (после коррекции ожидаемого количества точек) переходят в список для последующей обработки[3].

Каждая точка неизвестного символа ищет битовый вектор прототипов данного класса, наиболее с ней совпадающих, после чего вычисляется их истинное подобие. Каждый класс символа прототипа представлен в виде канонической нормальной формы, где каждое произведение представляет конфигурацию, поэтому процесс расчета расстояния оставляет записи о доказательстве полного подобия каждой точки в каждой конфигурации для каждого прототипа. Лучшая подсчитанная дистанция, которая вычисляется суммированием расстояния для каждой точки и будет являться лучшей из всех хранимых конфигураций класса.

Так как классификатор способен легко распознавать поврежденные символы, на них он не обучается. На самом деле, классификатор обучается на всего лишь 20 образцах 94 букв 8 шрифтов одного размера, но с 4 атрибутами (нормальный, жирный, курсив, жирный курсив), итого 60160 тренировочных образцов[2].

1. Лингвистический анализ

Tesseract практически не содержит инструментов для лингвистического анализа. Хотя модуль распознавания слов подразумевает новую сегментацию, лингвистический модуль выбирает лучшую строку со словом в каждой из следующих категорий: самое частое слово, самое частое слово из словаря, самое частое числительное, самое частое слово в верхнем регистре, самое частое слово в нижнем регистре (опционально начинающееся с большой буквы), самое частое слово, выбранное классификатором. Финальным решением по данной сегментации будет являться слово с наименьшим значением расстояния, когда каждая из категорий определяется перемножением на различный коэффициент.

Слова из разных сегментаций могут иметь различное количество символов. Напрямую их сравнить нельзя, даже если классификатор указывает на такую возможность. Эта проблема решена в Tesseract путем генерирования двух чисел на каждую классификацию символа. Первое число, называемое показателем уверенности, представляет величину, противоположную нормализованному расстоянию от прототипа. Это позволяет рассуждать, что чем больше значение этого показателя, тем лучше. Второй параметр – рейтинг – представляет перемножение нормализованной дистанции на общую длину контура символа. Рейтинги можно складывать между собой в пределах одного слова, ведь общая длина контуров слова всегда остается неизменной[3].

OCR-движки могут получить преимущество от использования адаптивного классификатора. Статический классификатор хорош в распознавании любых типов шрифтов, не так хорошо справляется с различением разных символов или букв и небуквенных знаков.

Более чувствительный к шрифту адаптивный классификатор обучается на результатах статического классификатора и широко используется для получения более точных данных в документах с фиксированным набором шрифтов. Tesseract не использует шаблонный классификатор, однако использует многие его особенности и по исполнению приближается к статическому. Единственным значительным отличием между статическим и адаптивным классификатором, кроме обучающих данных, является то, что адаптивный классификатор использует изотопическую нормализацию, хотя статический нормализует символы по центроиду.

Такая нормализация делает проще работу со строками в верхнем или нижнем регистрах и улучшает проверку шума.

1. Дальнейшее развитие

С 2005 года поддержку и развитие Tesseract производит компания Google. Актуальной на данный момент является 4.1.1 версия, вышедшая 12 ноября 2019 года в качестве бета-версии.

Начиная с версии 4.0 в Tesseract используется нейронная сеть LSTM, ее обучение на данный момент возможно исключительно в UNIX системах. Распознавать текст стало возможным на более чем 100 языках с высокой точностью, однако взамен возросла и требуемая мощность вычислительной системы. Обработка сложных языков, однако, может потребовать меньше времени, чем при использовании базового Tesseract[6].

Нейронные сети требуют значительно больше данных для обучения и значительно медленнее функционируют. Для языков латинской группы существующая модель была обучена на 400000 строках текста 4500 видов шрифтов. Для других языков доступно не так много шрифтов, однако предобучение происходило на сходном количестве текстовых линий. Вместо того, чтобы тратить на обучение несколько минут или часов, Tesseract требует для обучения несколько дней или даже неделю. Даже с подготовленными данными могут возникать проблемы, поэтому в некоторых случаях сеть придется переобучить[6].

Есть несколько способов это сделать:

* Если задача немного отличается от тренировочных данных, например, используется необычный шрифт, то требуется лишь дообучить сеть на своих данных. Такой подход сработает даже с небольшим массивом обучающих данных.
* Обрезать один или несколько верхних слоев сети и переобучить ее верхние слои на новых данных. Если не сработал первый способ, этот скорее всего решит проблемы. Также это можно применять на совершенно новых языках и типах документов.
* Переобучение с ноля. Это очень трудоемкая задача, для которой требуются очень репрезентативная и большая выборка данных.

Все эти шаги кажутся разными, однако практически одинаковы в реализации. В актуальной версии присутствует и описанный выше старый OCR-движок, и нейронная сеть[2].

ЗАКЛЮЧЕНИЕ

Точное распознавание [латинских символов](https://ru.wikipedia.org/wiki/%D0%9B%D0%B0%D1%82%D0%B8%D0%BD%D1%81%D0%BA%D0%B8%D0%B9_%D0%B0%D0%BB%D1%84%D0%B0%D0%B2%D0%B8%D1%82) 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.

Точность работы методов может быть измерена несколькими способами и поэтому может сильно варьироваться. К примеру, если встречается специализированное слово, не используемое для соответствующего программного обеспечения, при поиске несуществующих слов, ошибка может увеличиться.

Ещё одной широко исследуемой задачей является [распознавание рукописного текста](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D0%BF%D0%BE%D0%B7%D0%BD%D0%B0%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D1%80%D1%83%D0%BA%D0%BE%D0%BF%D0%B8%D1%81%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B2%D0%B2%D0%BE%D0%B4%D0%B0). В данное время достигнутая точность даже ниже, чем для рукописного «печатного» текста. Более высокие показатели могут быть достигнуты только с использованием контекстной и грамматической информации. Например, в ходе распознания искать целые слова в словаре легче, чем пытаться выявить отдельные знаки из текста. Знание грамматики языка может также помочь определить, является ли слово глаголом или существительным. Формы отдельных рукописных символов иногда могут не содержать достаточно информации, чтобы точно (более 98 %) распознать весь рукописный текст.

Для решения более сложных задач в области распознавания используются, как правило, интеллектуальные системы распознавания, такие, как [искусственные нейронные сети](https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Patel, Chirag & Patel, Atul & Patel, Dharmendra. (2012). Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. International Journal of Computer Applications. 55. 50-56. 10.5120/8794-2784.
2. Smith [R.](https://dl.acm.org/author_page.cfm?id=81440609634&coll=DL&dl=ACM&trk=0) – An Overview of the Tesseract OCR Engine/ Google Inc., Published in: ICDAR '07 Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02, Pages 629-633/September 23 - 26, 2007: IEEE Computer Society Washington, DC, USA ©2007  
   [table of contents](https://dl.acm.org/citation.cfm?id=1304596&picked=prox) ISBN:0-7695-2822-8
3. Tejaswi, Chaitanya & Goradiya, Bhargav & Patel, Ripal. (2018). A Novel Approach of Tesseract-OCR Usage for Newspaper Article Images.
4. Селянкин, В.В. Решение задач компьютерного зрения : учебное пособие / В.В. Селянкин ; Министерство образования и науки РФ, Южный федеральный университет, Инженерно-технологическая академия. – Таганрог : Издательство Южного федерального университета, 2016. – 93 с. : схем., табл. – Режим доступа: по подписке. – URL: <http://biblioclub.ru/index.php?page=book&id=493304> – Библиогр. в кн. – ISBN 978-5-9275-2090-9. – Текст : электронный.
5. Томилов, А.А. Автоматизированный поиск автомобилей на основе обработки результатов видео наблюдения: выпускная квалификационная работа (магистерская диссертация) : студенческая научная работа / А.А. Томилов ; Омский государственный технический университет, Кафедра автоматизированных систем обработки информации и управления. – Омск : б.и, 2019. – 111 с. : ил., табл., схем. –– URL: <http://biblioclub.ru/index.php?page=book&id=562500>.–DOI 10.23681/562500.
6. Официальная документация Tesseract: https://github.com/tesseract-ocr/tesseract/wiki/