УДК 004.9

ПРИМЕНЕНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ В ЗАДАЧАХ РАСПОЗНАВАНИЯ ПЕЧАТНЫХ ТЕКСТОВ 19 ВЕКА

Алейников Марк Владимирович[1], Ершов Николай Михайлович[2]

[1]Инженер по тестированию и качеству;
ООО «Джон Вайли и Сыновья Рус»;
141080, Московская обл., г. Королев, ул. Дзержинского, 29;
e-mail: nevidimka991@gmail.com.

[2]Старший научный сотрудник;
МГУ им. М. В. Ломоносова, факультет вычислительной математики и кибернетики;
119991, Москва, ГСП-1, Ленинские горы, д. 1, стр. 52, факультет ВМК;
e-mail: ershovnm@gmail.com.

Работа посвящена изучению возможностей использования методов машинного обучения в задаче распознавания русских печатных документов 19 века. Приводятся результаты анализа существующих методов и средств для распознавания печатных текстов, в том числе проприетарных, на примере анализа некоторых русских документов 19 века. В работе предлагается подход к распознаванию текстов с использованием программного комплекса Tesseract, на основе которого разработаны две версии программной системы, работающей с оцифрованными изображениями текстовых документов. Приводятся результаты тестирования разработанной программной системы, показывающие перспективность предложенного подхода. Работа выполнена при финансовой поддержке РФФИ (грант № 20-07-01053 А).

Ключевые слова: оптическое распознавание текста, рекуррентные нейронные сети.

Для цитирования:
Алейников М. В., Ершов Н. М. Применение методов машинного обучения в задачах распознавания печатных текстов 19 века // Системный анализ в науке и образовании: сетевое научное издание. 2021. № 1. С. 12–22. URL : http://sanse.ru/download/422.

APPLICATION OF MACHINE LEARNING METHODS IN THE RECOGNITION OF PRINTED TEXTS OF THE 19TH CENTURY

Aleynikov Mark[1], Ershov Nikolay[2]

[1]Testing and quality engineer;
John Wiley and Sons RUS Co.Ltd;
29 Dzerzhinskogo Str., Korolyov, Moscow region, 141070, Russia;
e-mail: nevidimka991@gmail.com.

[2]Senior research associate;
Lomonosov Moscow State University,
Faculty of Computational Mathematics and Cybernetics;
1-52 Leninskiye Gory, GSP-1, Moscow, 119991, Russia;
e-mail: ershovnm@gmail.com.

The paper deals with the study of the possibilities of using machine learning methods in the problem of recognizing Russian printed documents of the 19th century. The results of the analysis of existing methods and tools for recognizing printed texts, including proprietary ones, are presented on the example of the analysis of some Russian documents of the 19th century. The paper proposes an approach to text recognition using the Tesseract software package, on the base of which two versions of a software system were developed and it works with digitized images of text documents. The results of testing the developed software system are presented, showing the prospects of the proposed approach. The work was carried out with the financial support of the Russian Foundation for Basic Research (grant No. 20-07-01053 A).

Keywords: optical character recognition, recurrent neural networks.

For citation:
Aleynikov M., Ershov N. Application of machine learning methods in the recognition of printed texts of the 19th century. System Analysis in Science and Education, 2021;(1):12–22(In Russ). Available from: http://sanse.ru/download/422.

Введение

При разработке программ для распознавания изображений, голоса, видео, и особенно, текста, особое внимание уделяется изучению новых технологий, что может помочь получить наиболее точный результат. Среди таких технологий активное развитие получили искусственные нейронные сети, которые характеризуются возможностью обучаться по аналогии человеческого мозга. Эта особенность является основной чертой искусственных нейронных сетей и позволяет наиболее эффективно применять их в распознавании. Однако, не все программы задействуют нейронные сети при распознавании текста. Существуют разные подходы к созданию системы распознавания текста, поэтому для начала следует разобраться как работает оптическое распознавание символов.

Оптическое распознавание символов (*OCR*) – комплексная и сложная технология, целью которой является преобразование изображения, содержащего текст, в текстовые данные, удобные для редактирования. *OCR* позволяет обрабатывать оцифрованные книги, снимки экрана, фотографии с текстом, и получать конечный результат в таких форматах как *TXT, DOC* или *PDF*. Оптическое распознавание символов широко используется в самых разных областях. Самые современные системы распознавания могут обрабатывать практически любые типы изображений разной степени сложности. Рассмотрим этапы, которые являются типичными для системы оптического распознавания символов:

- Загрузка изображения из файла. Система распознавания должна учитывать разные форматы изображений, такие как: *TIFF, BMP, JPEG, PNG*. Также, стоит отметить, что в некоторых *OCR* реализована поддержка формата PDF, так как многие документы хранятся именно в этом формате.

- Обнаружение основных черт изображения, например разрешение и инверсия. Очень часто, перед тем как распознавать текст с изображения, изображение необходимо изменить, в том числе изменяя его фоновый цвет и переворачивая, изменяя то, под каким углом расположен текст. Также изображение может быть искажено и зашумлено, поэтому применяются специальные алгоритмы для устранения шума и повышения общей читаемости текста для системы распознавания.

- Ещё одной интересной чертой многих алгоритмов распознавания является возможность обработки только черно-белых изображений. Это значит, что для цветных изображений необходимо провести процесс бинаризации. Бинаризацией называется процесс преобразования изображения в черно-белое. При этом бинаризация должна быть корректной, потому что неправильно произведённая бинаризация может привести к проблемам в распознавании.

- Поиск и удаление линий. Такие действия позволяют улучшить анализ изображения, а также помогает в обнаружении таблиц.

- Поиск текстовых строк и слов, а также анализ сочетания текстовых символов. Поиск символов может осложняться разным размером шрифта, расстоянием между словами. Сочетание текстовых символов также может являться проблемным из-за наслоения символов и неполной пропечатки текста.

- Распознавание символов. Это основной этап в системе *OCR*. Изображение каждого символа должно быть преобразовано в соответствующий текстовый символ. Проблема этого шага в том, что иногда система выдаёт несколько текстовых символов в случае неопределённости при распознавании.

- Поддержка словаря. Этот этап может улучшить качество распознавания. Некоторые символы, такие как «1» и «*I*», «*C*» и «*G*», могут выглядеть очень похожими, и словарь может помочь принять правильное решение.

- Сохранение результатов в выбранном формате вывода, например, с возможностью поиска *PDF, DOC, RTF* или T*XT*.

Стоит отметить, что это не полный список всех возможных этапов, и многие другие второстепенные алгоритмы также могут быть реализованы для достижения хорошего распознавания для

самых разных типов изображений, а подходы могут различаться в разных системах распознавания. Каждый этап в работе системы *OCR* очень важен, а существующие доступные решения в этой области могут предложить распознавание вплоть до самого широкого диапазона изображений, поэтому на рынке представлено всего несколько хороших универсальных систем оптического распознавания текста. Однако, если известны некоторые особенности входных изображений, задача становится намного проще, а наилучшего качества распознавания можно достичь, если проектировать систему специально под этот тип изображений.

Программные средства для распознавания текстов

Среди программных продуктов, предназначенных для распознавания текстовых документов, стоит особенно отметить следующие: *Tesseract, Tensorflow, ABBYY FineReader, Google Vision API*, а также различные онлайн-сервисы, которые чаще всего построены на одной из перечисленных технологий. Остановимся на каждом из указанных вариантов подробнее с целью выявить слабые и сильные стороны каждого варианта.

Tesseract – свободная компьютерная программа для распознавания текстов, разрабатывавшаяся *Hewlett-Packard* с середины 1980-х по середину 1990-х, а затем 10 лет «пролежавшая на полке». В августе 2006 г. Google купил её и открыл исходные тексты под лицензией *Apache* 2.0 для продолжения разработки. В настоящий момент программа уже работает с *UTF*-8, поддержка языков (включая русский с версии 3.0) осуществляется с помощью дополнительных модулей [1]. *Tesseract* используется только в задачах распознавания текста и демонстрирует достаточно хорошую производительность и результат в такого рода задачах. *Tesseract*, вплоть до версии 2 включительно, мог принимать только изображения *TIFF* простого одностолбцового текста в качестве входных данных. Обычно задача распознавания с помощью движка *Tesseract* достигается в два этапа, а именно: обучение и тестирование. В качестве задачи распознавания он принимает двоичное изображение в качестве входных данных и выполняет анализ макета страницы, чтобы найти строки, слова и, в конечном итоге, символы. На обоих этапах обучения и тестирования подзадачи необходимые для построения системы *OCR* для конкретного языка, выглядят следующим образом: подготовка изображения обучающих данных, подготовка файла блока, обучение файла блока, вычисление файла набора символов, подготовка шрифта файл свойств, кластеризация, подготовка словаря, переименование файлов, объединение всех сгенерированных файлов для получения пакета, и использование пакета для тестирования.

TensorFlow – открытая программная библиотека для машинного обучения, разработанная компанией *Google* для решения задач построения и тренировки нейронной сети с целью автоматического нахождения и классификации образов, достигая качества человеческого восприятия [2]. Применяется как для исследований, так и для разработки собственных продуктов *Google*. Основной *API* для работы с библиотекой реализован для *Python*, также существуют реализации для *C++, Haskell, Java, Go* и *Swift*. *TensorFlow* заметно упрощает встраивание в приложения самообучаемых элементов и функций искусственного интеллекта, предназначенных для распознавания речи и текста, использования компьютерного зрения и многого другого. Отличительная особенность *Tensorflow* заключается в поддержке широкого спектра устройств. Помимо этого, *Tensorflow* имеет лучшую визуализацию вычислительных графов по сравнению с другими библиотеками, постоянную и активную поддержку обновлениями, в том числе и глобальными с новыми функциями.

Стоит отметить, что и *Tesseract*, и *Tensorflow* являются программным обеспечением с открытым исходным кодом. Указанные ранее *ABBYY Finereader* и *Google Vision API* также показывают очень хороший результат в распознавании текста, однако, подробно останавливаться на их характеристиках и особенностях не имеет смысла из-за их распространения на платной основе. Таким образом, среди всех кандидатов для построения собственной системы распознавания русских документов 19 века остаются два варианта – *Tensorflow* и *Tesseract*. И между ними существует принципиальная разница, так как *Tensorflow* – это инструментарий, предоставляющий широкие возможности по созданию нейронных сетей совершенно различного назначения, а *Tesseract* – это готовая программа, функционирующая на основе нейронной сети, для распознавания именно текста вне зависимости от языка. *Tesseract* можно обучить, доработать, а также использовать в связке с другими инструментами, что делает эту библиотеку наиболее подходящим кандидатом для создания системы распознавания документов 19 века.

Архитектура Tesseract

Рассмотрим подробнее архитектуру *Tesseract*. Стоит сказать, что изначально *Tesseract* задумывался так, чтобы входными данными были бинарные изображения, где бинарное изображение – такое изображение, где каждый пиксель может представлять только один из двух цветов. Обработка изображения идёт по шагам и некоторые из шагов были необычны для своего времени из-за требований к вычислительной мощности, однако, оправдали себя в будущем.

Первым таким шагом являются определение макета страницы и анализ связанных компонентов текста – контуров и очертаний текста. Контуры и очертания преобразуются в специальные *BLOB*-объекты. Объекты анализируются, в них выделяются строки текста. Строки делятся на слова, в зависимости от расстояния между ними.

Следующим шагом является анализ того, какой шрифт был использован в конкретной области текста. *Tesseract* разделяет шрифты на два типа: фиксированный и пропорциональный. В случае фиксированного шрифта каждый из текстовых символов имеет фиксированную ширину, а это позволяет *Tesseract* сразу же рассматривать строку посимвольно. В случае пропорционального шрифта все сложнее – происходит выделение чётких и нечётких промежутков между текстовыми символами в зависимости от межстрочного интервала. Для нечётких промежутков решение принимается в самом конце процесса распознавания.

После окончания подготовительных шагов происходит само распознавание текста в виде двух этапов. На первом этапе делается попытка распознать каждое слово по очереди. Каждое удовлетворительно успешно распознанное слово передаётся адаптивному классификатору в качестве обучающих данных. Это позволяет классификатору более точно распознать текст при продвижении по странице. Так как классификатор, возможно, слишком поздно усвоил что-то полезное для корректного распознавания слов ранее на первом этапе, выполняется второй проход по тому же тексту, в котором слова, которые были недостаточно хорошо распознаны, распознаются снова.

Финальный этап решает проблему нечётких промежутков между символами и проверяет на соответствие некоторых нераспознанных текстовых символов малым заглавным буквам.

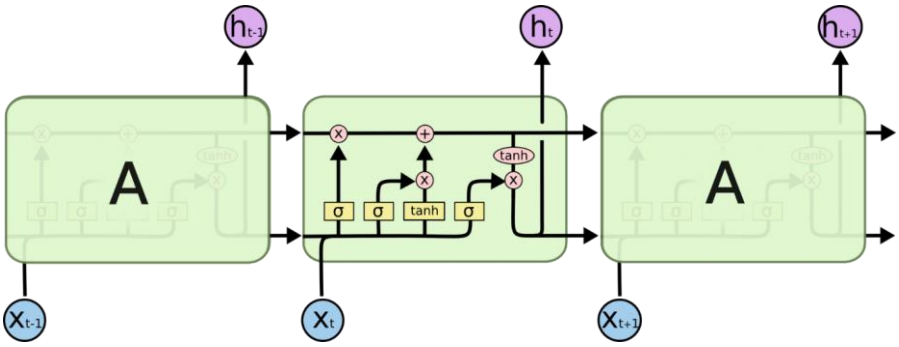


Рис. 1. Модель LSTM сети

Описанные шаги алгоритма относятся к *Tesseract* до версии 4. Дело в том, что начиная с версии 4 разработчики внедрили в *Tesseract* нейронные сети типа *LSTM* (*Long short-term memory*, [3]) для использования их в анализе макета страницы, определении границ текстовых символов и строк. Рассмотрим подробнее что такое искусственные нейронные сети типа *LSTM* (см. рис. 1). Рекуррентные нейронные сети, подвидом которых является *LSTM*, это разновидность архитектуры нейронных сетей, задействующих обратную связь. В отличие от обычных нейронных сетей прямого распространения, где информация передается последовательно от слоя к слою, в рекуррентных нейронных сетях каждый слой нейронов, помимо входных данных, получает некоторую информацию о предыдущем состоянии сети, что делает возможным анализ последовательностей данных. Таким образом, можно выделить главную особенность *LSTM* сетей – способность запоминать и работать с информацией, которая была обработана ранее, некоторое количество циклов назад. Такая архитектура делает *LSTM* сети очень хорошим выбором в распознавании текста.

Тестирование существующих решений

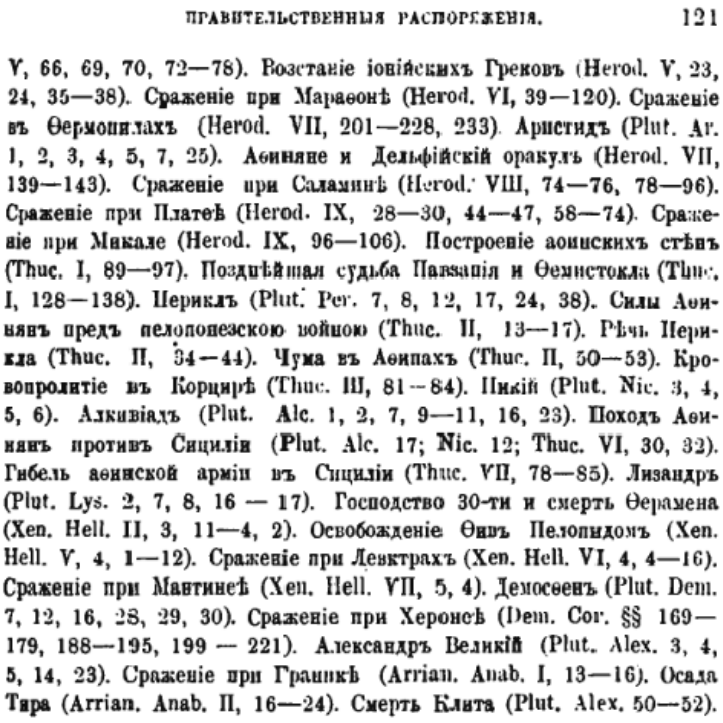
Первым этапом проведённого исследования было тестирование существующих средств для распознавания текстов журнала Министерства народного просвещения, датированных 19 веком. Выпуски этого журнала свободно доступны в оцифрованном виде в высоком разрешении [4]. Все документы написаны на русском языке, однако, в связи с датой написания, в документах используется русская дореформенная орфография, содержащая старые символы, неиспользуемые в современном русском языке. Эта особенность является основным препятствием для всех готовых систем распознавания, не предоставляющих гибкие инструменты для обучения. Тем не менее, прежде чем приступать к работе над своим решением, необходимо протестировать насколько велика ошибка при распознавании таких документов в нескольких распространённых сервисах.

Были рассмотрены следующие популярные сервисы для распознавания текста:

- ABBYY FineReader Online [5].
- Google Vision API [6].
- OnlineOCR.net [7].
- OCR.space [8].

Такой выбор программ обусловлен следующими факторами. *ABBYY FineReader Online* – веб-версия популярного приложения для распознавания текста *ABBYY FineReader*. *Google Vision API* – демонстрационная версия одной из технологий *Google*, созданной для различных целей в области распознавания, в том числе и распознавания текста. *OnlineOCR.net* и *OCR.space* были выбраны в качестве примера демонстрации возможностей бесплатных онлайн сервисов для распознавания текста.

В качестве тестируемых документов выбраны страницы 123-125 Части CXCIII «Журнала Министерства народного просвещения» 1877 года. Из-за использования литерного способа печати текст читается с трудом, но при этом в самих документах отсутствуют какие-либо явные следы повреждений (рис. 2).



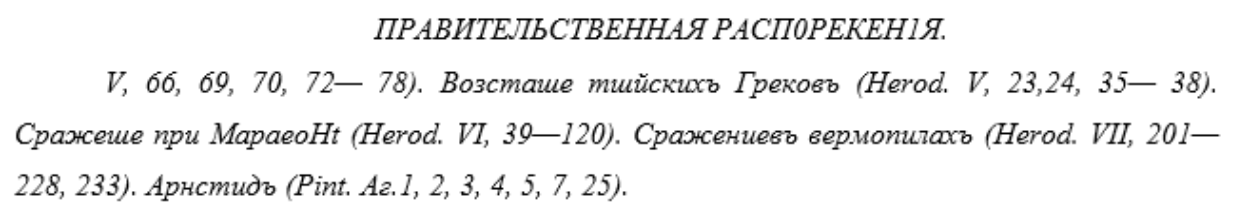
ПРАВИТЕЛЬСТВЕННЫЯ РАСПОРЯЖЕНІЯ. 121

V, 66, 69, 70, 72—78). Возстаніе іонійскихъ Грековъ (Herod. V, 23, 24, 35—38). Сраженіе при Марафонѣ (Herod. VI, 39—120). Сраженіе въ Ѳермопилахъ (Herod. VII, 201—228, 233). Аристидъ (Plut. Ar. 1, 2, 3, 4, 5, 7, 25). Аѳиняне и Дельфійскій оракулъ (Herod. VII, 139—143). Сраженіе при Саламинѣ (Herod. VIII, 74—76, 78—96). Сраженіе при Платеѣ (Herod. IX, 28—30, 44—47, 58—74). Сраженіе при Микале (Herod. IX, 96—106). Построеніе аѳинскихъ стѣнъ (Thuc. I, 89—97). Позднѣйшая судьба Павсанія и Ѳемистокла (Thuc. I, 128—138). Перикл (Plut. Per. 7, 8, 12, 17, 24, 38). Силы Аѳинянъ предъ пелопонезскою войною (Thuc. II, 13—17). Рѣчь Перикла (Thuc. II, 34—44). Чума въ Аѳинахъ (Thuc. II, 50—53). Кровопролитіе въ Корцирѣ (Thuc. III, 81—84). Никій (Plut. Nic. 3, 4, 5, 6). Алкивіадъ (Plut. Alc. 1, 2, 7, 9—11, 16, 23). Походъ Аѳинянъ противъ Сициліи (Plut. Alc. 17; Nic. 12; Thuc. VI, 30, 32). Гибель аѳинской арміи въ Сициліи (Thuc. VII, 78—85). Лизандръ (Plut. Lys. 2, 7, 8, 16 — 17). Господство 30-ти и смерть Ѳерамена (Xen. Hell. II, 3, 11—4, 2). Освобожденіе Ѳивъ Пелопидомъ (Xen. Hell. V, 4, 1—12). Сраженіе при Левктрахъ (Xen. Hell. VI, 4, 4—16). Сраженіе при Мантинеѣ (Xen. Hell. VII, 5, 4). Демосѳенъ (Plut. Dem. 7, 12, 16, 28, 29, 30). Сраженіе при Херонеѣ (Dem. Cor. §§ 169—179, 188—195, 199 — 221). Александръ Великій (Plut. Alex. 3, 4, 5, 14, 23). Сраженіе при Граникѣ (Arrian. Anab. I, 13—16). Осада Тира (Arrian. Anab. II, 16—24). Смерть Клита (Plut. Alex. 50—52).

Рис. 2. Пример распознаваемого текста из журнала Министерства народного просвещения

Рассмотрим подробнее *ABBYY FineReader Online*. *ABBYY FineReader Online* является платным сервисом с бесплатным ознакомительным сервисом, позволяющим распознать 10 страниц, после чего бесплатные возможности закончатся. Использование *ABBYY FineReader Online* в данной работе

подразумевает использование бесплатного ознакомительного сервиса. Рассмотрим, как данный сервис справился с распознаванием тестовых документов на примере заголовка и первых предложений указанного выше примера (рис. 3).

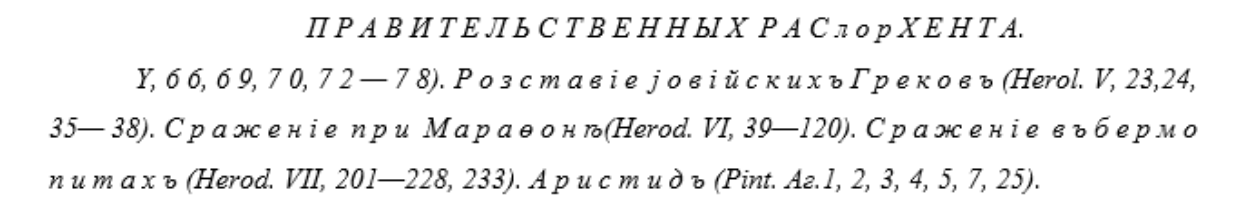

ПРАВИТЕЛЬСТВЕННАЯ РАСП0РЕКЕН1Я.

V, бб, 69, 70, 72— 78). Возсташе тшйскихъ Грековъ (Herod. V, 23,24, 35— 38). Сражеше при МараеоНt (Herod. VI, 39—120). Сраженіевъ вермопилахъ (Herod. VII, 201—228, 233). Арнстидъ (Pint. Аг.1, 2, 3, 4, 5, 7, 25).

Рис. 3. Пример работы системы ABBYY FineReader Online

На основе проведённых экспериментов можно сделать вывод, что *ABBYY FineReader Online* не всегда верно понимает старые русские буквы, которые использовались до реформы русского языка 1918 года, при этом часто корректно распознает остальные символы и цифры, а также слова на иностранном языке. Несмотря на то, что сервис обладает удобным и понятным интерфейсом, использование данного сервиса в работе с быстрым распознаванием столь старых документов будет являться некорректным из-за его очевидных ошибок.

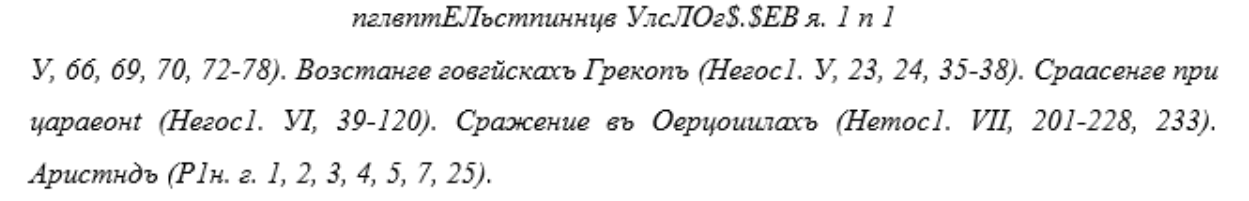
Следующим был рассмотрен сервис *Google Vision API*. Этот сервис является примером демонстрации технологии *Cloud Vision API*, созданной с целью использования разработчиками для определения содержания изображения, используя при этом лишь онлайн-сервис. *Google Vision API* может быстро классифицировать изображения по тысячам категорий, обнаруживать отдельные объекты и лица на изображениях и распознавать напечатанные слова, содержащиеся в изображениях. Использование *Google Vision API* является платным, но предполагает ознакомление с возможностями сервиса. Использование *Google Vision API* в данной работе подразумевает использование бесплатного ознакомительного сервиса. Пример распознавания текста, показанного на рис. 2, приведён на рис. 4.


ПРАВИТЕЛЬСТВЕННЫХ РАСлорХЕНТА.

У, б б, б 9, 7 0, 7 2 — 7 8). Розставіе јовійскихъ Грековъ (Herol. V, 23,24, 35— 38). Сраженіе при Мараѳонъ(Herod. VI, 39—120). Сраженіе въбермо питахъ (Herod. VII, 201—228, 233). Аристидъ (Pint. Аг.1, 2, 3, 4, 5, 7, 25).

Рис. 4. Пример работы системы Google Vision API

Можно заметить, что *Google Vision API* не смог распознать вторую часть заголовка, также возникли существенные проблемы с распознаванием и обычных слов. При этом все цифры были распознаны корректно и большая часть букв тоже. Можно сделать вывод, что хотя *Google Vision API* и «понимает» старые русские буквы и даже использует для их отображения специальный шрифт *RU-PETR*1708, использовать этот сервис для массового распознавания такого рода документов является явно нецелесообразным.


пглвптЕЛьстпиннцв УлсЛОг$.$ЕВ я. 1 п 1

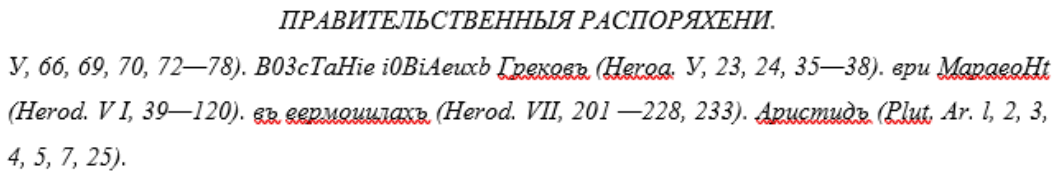
У, бб, 69, 70, 72-78). Возстанге говгйскахъ Грекопъ (Негос1. У, 23, 24, 35-38). Сраасенге при цараеонт (Негос1. VI, 39-120). Сражение въ Оерцоишлахъ (Нетос1. VII, 201-228, 233). Аристндъ (Р1н. г. 1, 2, 3, 4, 5, 7, 25).

Рис. 5. Пример работы системы OnlineOCR.net

Следующая система *OnlineOCR.net* – бесплатный веб-сервис для распознавания печатного текста, который позволяет преобразовывать отсканированные *PDF* документы (в том числе многостраничные файлы), факсы, фотографии или цифровой камеры отснятые изображения в редактируемые электронные документы форматов *PDF*, *Word*, *Excel*, *RTF*, *HTML* и *TXT*. Пример работы этого сервиса приведен на рис. 5. Как можно заметить, сервис *OnlineOCR.net* показал очень плохой результат в

распознавании этого печатного текста. Заголовок распознан полностью некорректно, большинство слов также были распознано некорректно, а английский текст принудительно переведён в русский. Кроме того, сервис *OnlineOCR.net* также не «понимает» старые русские буквы, которые вышли из употребления.

Последний из рассматриваемых сервисов – *OCR.space*. Это бесплатный сервис, созданный на основе бесплатного *API* с платными дополнительными функциями. Создатели уверяют, что используют самые современные технологии для распознавания текста, а качество распознавания сопоставимо с коммерческим программным обеспечением (например, компании *ABBYY*). Ограничение сервиса – изображения или *PDF*-документы не должны превышать 5 *МБ*, что уже является серьёзным препятствием к использованию, так как документы Министерства народного просвещения, выгруженные в формате PDF, занимают объем порядка 100 *МБ*. Пример работы этого сервиса показан на рис. 6.


ПРАВИТЕЛЬСТВЕННЫЯ РАСПОРЯХЕНИ.

У, 66, 69, 70, 72—78). B03cTaHie i0BiAeuxb Грековъ (Herod. У, 23, 24, 35—38). ври МараеоНt (Herod. V I, 39—120). въ вермоилахъ (Herod. VII, 201 —228, 233). Аристидъ (Plut. Ar. l, 2, 3, 4, 5, 7, 25).

Рис. 6. Пример работы системы OCR.space

Видно, что распознавание старых документов в данном сервисе работает некорректно. Частично верно был распознан заголовок, но далее можно заметить, что сервис не понимает старые русские буквы и пытается заменить их цифрами, что, безусловно, является неправильным. Числа и английские слова распознаются верно, тем не менее перечисленные недостатки не могут позволить использовать *OCR.space* в качестве полноценного сервиса для распознавания старых документов.

Результаты проведённых экспериментов говорят о том, что в настоящий момент сервисы для распознавания текста плохо справляются с определением текста в печатных русских документах 19 века. Это говорит о необходимости разработки специального сервиса, который бы распознавал такого рода печатные тексты максимально корректно и предоставлял бы возможности поиска и выделения ключевых слов в распознанном тексте.

Подготовка обучающей выборки и процесс обучения Tesseract

Прежде чем говорить о подготовке обучающей выборки для обучения *Tesserac*t, нужно понять, что для этого требуется. Дело в том, что процесс обучения в *Tesseract* достаточно непростой для понимания. Создание обучающей выборки и использование её в *Tesseract* позволяет создать фактически новый собственный язык на основе данных обучающей выборки. Для этого *Tesseract* создаёт свой алфавит, куда записывает все встречающиеся символы из обучающей выборки. Конечным результатом обучения является специальный *box*-файл, который *Tesseract* может читать и использовать для создания отдельного языкового пакета.

Рис. 7. Пример создания обучающей выборки в Tesseract

При стандартном способе обучения, который предлагают разработчики, генерация *box*-файлов связана с обработкой всех текстовых данных обучающей выборки вручную. *Box*-файл представляет собой текстовый файл, содержащий все символы, находящиеся на изображении и координаты рамок, которые определяют границы текстовых символов. Очевидно, что обучающая выборка может иметь очень большое количество изображений и создание *box*-файла для каждого из них займёт очень много времени. Поэтому было разработано несколько инструментов, самый удобный из которых

называется *tesstrain* [9]. *Tesstrain* позволяет создавать файлы формата *traineddata* при помощи одной команды *make*. Для этого нужно создать выборку из изображений и текстовых файлов в определённом формате. Изображения формата *tiff* должны содержать предложение, или несколько слов, размещённых на одной строке. Текстовые файлы содержат расшифровку изображения. Пример обучающего примера, состоящий из изображения и текстовой расшифровки, показан на рис. 7.

Обучение *Tesseract* при использовании инструмента *tesstrain* представляет собой достаточно простую задачу. Однако стоит учесть, что для правильного обучения *Tesseract* нужна обучающая выборка большого размера. Также само обучение занимает большое количество времени.

Была создана обучающая выборка, состоящая из 1000 отобранных вручную различных изображений текста из документов журнала Министерства народного просвещения. При помощи *tesstrain* все созданные изображения были использованы для создания файла в формате *traineddata*.

Результаты обучения Tesseract

Работа системы *Tesseract*, обученной на 1000 подготовленных примерах, была протестирована на нескольких тестовых фрагментах, один из которых показан на рис. 8.

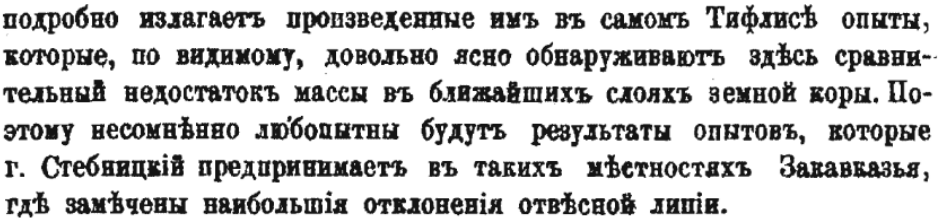

подробно излагаетъ произведенные имъ въ самомъ Тифлисѣ опыты, которые, по видимому, довольно ясно обнаруживаютъ здѣсь сравнительный недостатокъ массы въ ближайшихъ слояхъ земной коры. Поэтому несомнѣнно любопытны будутъ результаты опытовъ, которые г. Себницкій предпринимаетъ въ такихъ мѣстностяхъ Закавказья, гдѣ замѣчены наибольшія отклоненія отвѣсной линіи.

Рис. 8. Тестовый фрагмент текста

Результат распознавания этого фрагмента текста системой *Tesseract* показан на рис. 9.

```
подробно излагаетъ произведенные имъ въ самомъ Тифлисѣ опыты,
которые, по видимому, довольно ясно обнаруживаютъ здѣсь сравни-
тельный недостатокъ массы въ ближайшихъ слояхъ земной коры. По-
этому несомнѣнно любопытны будутъ результаты опытовъ, которые
г. Себницкій предпринимаетъ въ такихъ мѣстностяхъ Закавказья,
гдѣ замѣчены нанбольшія отклоненія отвѣсной липіи.
```

Рис. 9. Результат работы системы Tesseract

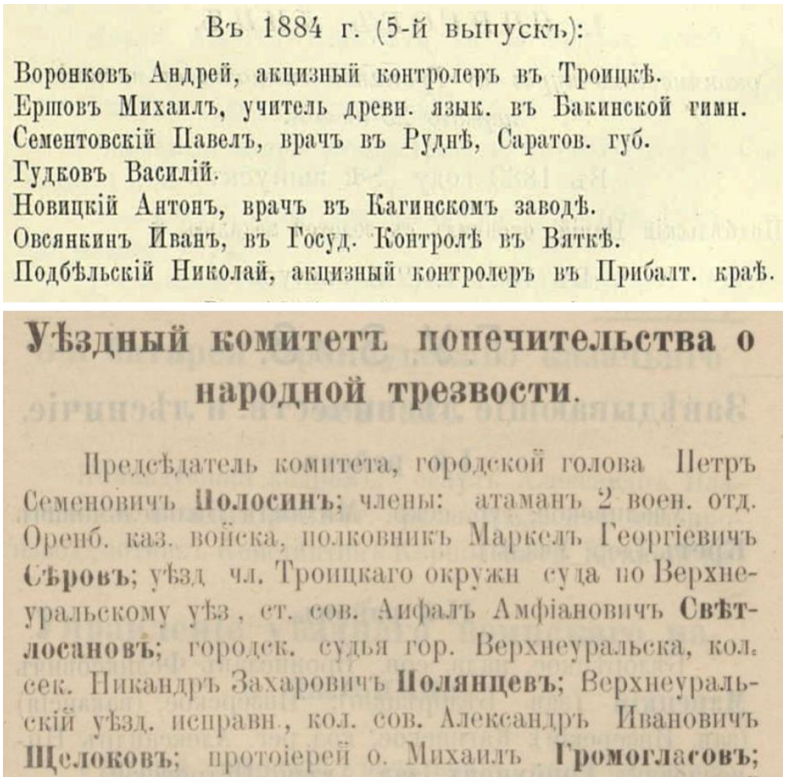
В таблице 1 приведены результаты тестирования системы *Tesseract* на трех текстовых фрагментах и сравнения с системой *ABBYY FineReader Online*.

Табл. 1. Результаты тестирования системы Tesseract

Фрагмент	ABBYY FineReader Online	Tesseract
1	97,6 %	95,2 %
2	89,3 %	85,1 %
3	60.0 %	75,0 %

Несмотря на то, что второй и третий фрагменты были взяты из других источников (т. е. это был не журнал Министерства народного просвещения, см. рис. 10), обученная система *Tesseract* показала результат, не уступающий лучшему платному аналогу. Полученные результаты говорят о возможности использовать *Tesseract* в том числе, и на других фрагментах текста, которые не относятся к журналу Министерства народного просвещения, на текстах которого происходило обучение. Этот важный результат показывает потенциал использования разработанной системы *Tesseract* и её гибкость,

ведь, если расширить обучающую выборку, можно добиться ещё более впечатляющих результатов в распознавании.


Въ 1884 г. (5-й выпускъ):
Воронковъ Андрей, акцизный контролеръ въ Троицкѣ.
Ершовъ Михаилъ, учитель древн. язык. въ Бакинской гимн.
Сементовскій Павелъ, врачъ въ Руднѣ, Саратов. губ.
Гудковъ Василій.
Новицкій Антонъ, врачъ въ Кагинскомъ заводѣ.
Овсянкинъ Иванъ, въ Госуд. Контролѣ въ Вяткѣ.
Подбѣльскій Николай, акцизный контролеръ въ Прибалт. краѣ.

Уѣздный комитетъ попечительства о народной трезвости.

Предсѣдатель комитета, городской голова Петръ Семеновичъ **Полосинъ**; члены: атаманъ 2 воен. отд. Оренб. каз. войска, полковникъ Маркелъ Георгіевичъ **Сѣровъ**; уѣзд. чл. Троицкаго окружн. суда по Верхнеуральскому уѣз., ст. сов. Аѳалъ Амфіановичъ **Свѣтлосановъ**; городск. судья гор. Верхнеуральска, кол. сек. Никандръ Захаровичъ **Полянцевъ**; Верхнеуральскій уѣзд. исправн., кол. сов. Александръ Ивановичъ **Щелоковъ**; протоіерей о. Михаилъ **Громогласовъ**;

Рис. 10. Два дополнительных тестовых фрагмента текста

Программная реализация

Было выполнено две реализации системы работы с оцифрованными текстовыми документами на основе технологии *Tesseract*. Первая версия – десктопная, предназначена для установки на пользовательский компьютер. Графическая оболочка разработана на языке программирования C# совместно с *Microsoft Visual Studio* на платформе *.NET Framework*. Программа позволяет:

- принимать в качестве входных данных документ формата *PDF*;
- выводить содержимое документа постранично с возможностью быстрого выбора нужной страницы;
- выводить распознанный текст страницы рядом с оригинальным изображением с возможностью редактирования распознанного текста.

Интерфейс разработанной программы показан на рис. 11.

Веб-версия программы построена на порте *Tesseract* на язык *Javascript* под названием *Tesseract.js* [10]. Принципиальное отличие от настольной версии программы заключается в использовании клиент-серверной архитектуры. На сервере располагаются основные скрипты, в том числе и обученные файлы в формате *traineddata*, которые применяются точно также, как и при обычном использовании *Tesseract*. Клиент при этом обращается к серверу, где происходит основная обработка документа. Однако, само распознавание происходит уже на стороне клиента. Интерфейс клиентской части веб-приложения показан на рис. 12.

Заключение

В результате выполненной работы получены следующие результаты.

- Проведён обзор и анализ существующих подходов к распознаванию оцифрованных текстовых документов в русской дореформенной орфографии.

- Построена система тренировочных примеров для обучения системы *Tesseract* на основе текстов журнала Министерства народного просвещения 19 века.

- Проведены обучение системы *Tesseract* и её тестирование на документах из различных источников, показавшее перспективность использования данного подхода.

- Реализованы программная и интерфейсная часть системы распознавания, построенной на основе технологии *Tesseract*. Приложение позволяет работать пользователю как с отдельными изображениями, так и с целыми документами в формате *PDF*. Система реализована в двух вариантах – как дескопное приложение и как веб-приложение.

Работа выполнена при финансовой поддержке РФФИ (грант № 20-07-01053 А).

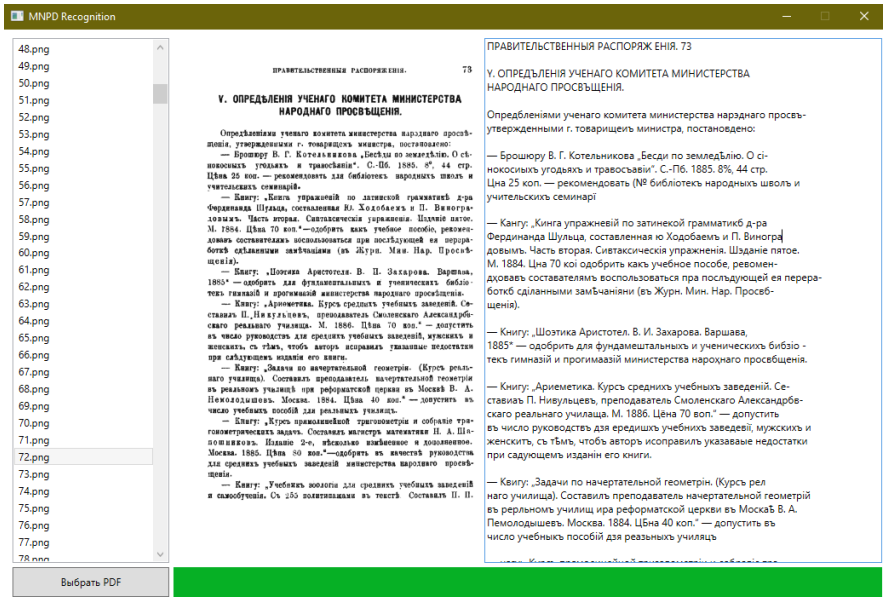


Рис. 11. Интерфейс десктопной версии программы

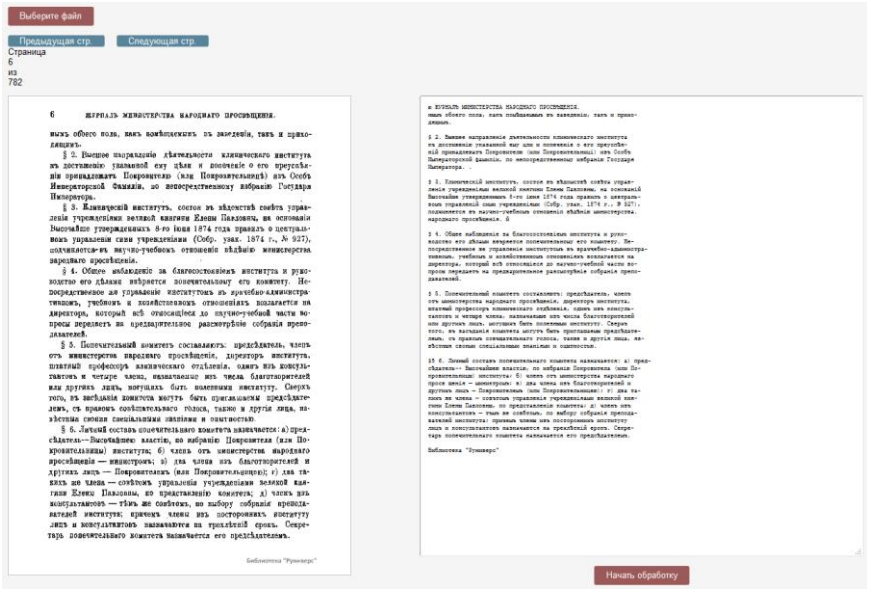


Рис. 12. Интерфейс клиентской части веб-версии программы распознавания

Список литературы

1. Tesseract // GitHub. URL : https://github.com/tesseract-ocr/tesseract.
2. Tensorflow // GitHub. URL : https://github.com/tensorflow/tensorflow.
3. Olah C. Understanding LSTM Networks // colah's blog. 2015. URL : http://colah.github.io/posts/2015-08-Understanding-LSTMs/.
4. Журнал Министерства народного просвещения / Тип. Императорской Академии Наук. СПб., 1834-1917. // Runivers.ru – Россия в подлиннике. URL : https://www.runivers.ru/lib/book7643/.
5. ABBYY FineReader Online / ABBYY. URL : https://finereaderonline.com/ru-ru/Tasks/Create.
6. Google Vision API. UR L: https://cloud.google.com/vision/.
7. Free Online OCR Service. URL : https://www.onlineocr.net/ru/.
8. OCR.space / a9t9 software GmbH. URL : https://ocr.space/.
9. Tesseract User Manual. URL : https://tesseract-ocr.github.io/tessdoc/Home.html.
10. Tesseract.js : Pure Javascript OCR for 100 Languages! URL : https://tesseract.projectnaptha.com.