Почему мы пока не можем компьютерно распознать рукописный архив Р. Г. Назирова?

А. А. Кондратьева
Национальный исследовательский университет «Высшая школа экономики»

1 Введение

Архив Р. Г. Назирова насчитывает сотни дел и десятки тысяч листов. Несмотря на длительную работу над ним, архив не только полностью не оцифрован, но даже не описан. Журнал «Назировский архив» существует уже 4 года, и в нём пока удалось напечатать малую часть имеющихся в распоряжении публикаторов документов. При таких объёмах рукописного наследия его обнародование может растянуться на десятилетия.

Самым трудоёмким этапом при публикации рукописи является её набор. Также сложная текстологическая работа по сравнению вариантов при наличии электронной копии может быть автоматизирована благодаря специализированным программам. Но набор манускрипта по-прежнему остаётся главным препятствием на пути ускорения процесса публикации.

В условиях глобальной информатизации и упрощения многих сфер деятельности с помощью компьютерных инструментов логично было бы подыскать возможность для автоматического распознавания рукописей Р. Г. Назирова, их дигитализации и введения в научный оборот. С первого взгляда, эта задача не выглядит невозможной, тем более, что почерк учёного может быть описан как чёткий и крупный, помарок и исправлений в текстах, которые в первую очередь нуждаются в публикации, обычно мало.

Однако на этом пути мы пока что сталкиваемся с трудно преодолимыми сложностями.

Оптическое распознавание символов (optical character recognition, OCR) — конвертация изображений, содержащих текст, в текстовые данные — является одним из наиболее популярных подразделов теории распознавания образов.

Попытки создать систему, которая бы качественно распознавала текст на изображении, начались еще в конце 1920-х годов[1] и продолжаются до сих пор. На данный момент существует достаточное количество технологий для распознавания текста, часть из которых представлена как коммерческие продукты (Abbyy FineReader[2]), часть — находится в открытом доступе (Tesseract-OCR[3], OCR Cuneiform[4]). Стоит отметить, что когда мы говорим «оптическое распознавание символов», мы подразумеваем оффлайн-распознавание, которое направлено на извлечение текста из отсканированного изображения. Онлайн-распознавание, ориентирующееся, в первую очередь,

[1] https://history-computer.com/ModernComputer/Basis/OCR.html
[2] https://www.abbyy.com/ru-ru/news/2017/01/abbyy-finereader-14
[3] https://github.com/tesseract-ocr/tesseract
[4] https://ru.wikipedia.org/wiki/CuneiForm

на движения во время написания текста, — это принципиально иная технология: примером программного обеспечения, предназначенного для онлайн-распознавания символов, является, например, Lipi Toolkit[5].

Возможность качественно распознать текст на изображениях во многом зависит от типа текста. Традиционно выделяют распознавание печатного, т. н. «рукописного печатного» и рукописного курсивного текста: отличие «рукописного печатного» от рукописного курсивного текста заключается в том, что в первом случае подразумеваются печатные буквы, написанные от руки. OCR-системы направлены, в первую очередь, на распознавание печатного текста. Так, точность распознавания латинских печатных символов при условии сравнительно высокого качества изображений, представленных в качестве образцов, может достигать 99 %[1]. Качество распознавания «рукописного печатного» текста на данный момент тоже находится на относительно высоком уровне. А вот распознавание рукописного курсивного текста все еще остается задачей, требующей детального исследования. Системы, которые хорошо справляются с этой задачей, существуют, однако их нет в свободном доступе (в качестве примера можно привести уже упомянутую Abbyy FineReader).

Однако многие открытые OCR предполагают возможность обучения их новым языкам и шрифтам, и если перед нами стоит задача оцифровать архив рукописей, принадлежащих одному автору, мы можем попробовать обучить систему распознавать его почерк и оценить, полученные результаты. Этому и будет посвящен дальнейший текст.

2 Обзор существующих подходов к распознаванию рукописного текста

На данный момент предпринято уже достаточно много исследований, посвященных попыткам добиться хороших результатов в распознавании рукописного текста с помощью OCR-систем, находящихся в открытом доступе, в частности, с помощью Tesseract-OCR, подробнее о которой речь пойдёт ниже. Так, в одной статье[2] описывается использование Tesseract-OCR для распознавания латинских символов в рукописных аннотациях. Авторам удалось добиться точности в 78.39 %. Ещё одна работа[3] посвящена попыткам обучить Tesseract распознаванию программного кода, написанного от руки: в некоторых случаях результаты достигали 93 – 95 %. Этот достаточно успешный результат обеспечивается формализованностью такого специфического жанра текста, как программный код. Последний состоит,

[5]http://lipitk.sourceforge.net/lipi-toolkit.htm Подробнее о противопоставлении онлайн- и оффлайн-распознавания текста см. в работе Dalbir, S. K. Singh. Review of Online & Offline Character Recognition // International Journal Of Engineering And Computer Science, ISSN: 2319 – 7242, Volume 4, Issue 5, May 2015, P.11729 – 11732.

[1]R. Holley. How Good Can It Get? Analysing and Improving OCR Accuracy in Large Scale Historic Newspaper Digitisation Programs // D-Lib Magazine, Volume 15, Number 3/4, 2009. Web: http://www.dlib.org/dlib/march09/holley/03holley.html

[2]S. Rakshit, S. Basu. Development of a multi-user handwriting recognition system using Tesseract open source OCR engine // Proc. International Conference on C3IT, 2009. P. 240 – 247.

[3]B. M. Gonzalez. Iris: A Solution for Executing Handwritten Code // Master's thesis, University of Agder, 2012. Web: https://brage.bibsys.no/xmlui/bitstream/handle/11250/137557/masteroppgave.pdf

главным образом, из фиксированного набора процедур, записывающихся при помощи повторяющегося набора символов.

Такие результаты не могут не обнадеживать, однако стоит отметить, что в большинстве подобных работ, в том числе и в упомянутых нами, алгоритм обучается на искусственно созданной выборке: датасеты состоят из двух частей, одна из которых — набор отдельных символов, используемых в распознаваемом языке, другая — набор отдельных предложений, чаще всего — панграмм наподобие «The quick brown fox jumps over the lazy dog».

Принципиальное отличие работы с архивом Р. Г. Назирова заключается в том, что мы составляем «естественную» тренировочную выборку, материалом для которой служат отсканированные листы рукописей. С этой точки зрения, логично ожидать худших результатов по сравнению с тем, чего уже удалось достичь на другом материале.

3 Распознавание рукописного текста с помощью Tesseract-OCR

3.1 Обзор Tesseract-OCR

Из доступных на сегодняшний день инструментов распознавания текста наиболее перспективной выглядит система Tesseract-OCR. Это свободно распространяемая программа для распознавания текстов, написанная на C++, использующая технологию оптического распознавания символов. С середины 1980-х до середины 1990-х проект курировала компания Hewlett-Packard, но позже он был заброшен и лишь в 2006 году выкуплен Google для дальнейшего развития. Система нацелена прежде всего на распознавание печатного текста.

3.1.1 Схема работы Tesseract-OCR

Принципы работы Tesseract-OCR подробно разбираются в соответствующей работе[1].

Вкратце процесс можно описать следующим образом: программа определяет макет страницы (layout) и группирует очертания текста в объекты (blobs). Далее в объектах выделяются строки, а строки, в свою очередь, разбиваются на слова в зависимости от длины промежутков между ними.

После этого анализируется, какой тип шрифта был использован для той или иной области текста: моноширинный или пропорциональный (fixed pitch text, proportional text). В первом случае каждый символ имеет фиксированную ширину, а значит, Tesseract способен сразу разбить строку на символы. Во втором случае программа выделяет определенные и неопределенные промежутки (definite spaces, fuzzy spaces) между символами: решение по поводу неопределенных промежутков принимается в самом конце.

Распознавание проходит в два этапа. Сначала алгоритм старается распознать каждое слово по очереди и передает каждую успешную попытку

[1] R. Smith. An overview of the Tesseract-OCR engine. // Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on. IEEE, 2007. Vol. 2. P. 629 – 633.

классификатору в качестве обучающей выборки. Классификатор использует эти данные при дальнейшем продвижении по странице и на втором этапе распознавания, когда алгоритм повторно работает с теми словами, которые не получилось распознать в первый раз.

В самом конце Tesseract принимает решение по поводу неопределенных промежутков и проверяет, не являются ли некоторые нераспознанные строчные буквы малыми заглавными (small-cap letters).

3.1.2 Нововведения в Tesseract 4.0

Принципиальным отличием Tesseract 4.0 от предыдущих версий является использование LSTM-сетей для определения макета страницы (layout analysis), а также для выделения границ строк и отдельных слов параллельно со стандартным алгоритмом.

Рекуррентные нейронные сети (РНС), подвидом которых является LSTM (long short-term memory), — тип нейронных сетей, задействующих обратную связь. В отличие от нейросетей прямого распространения, в которых 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.[1]

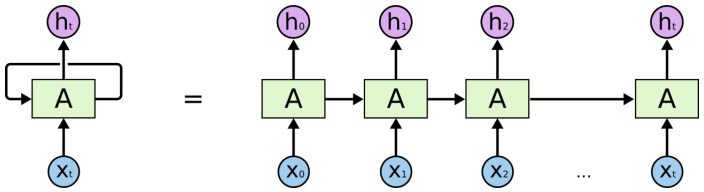

Рис. 1: Схема работы одного слоя нейронов LSTM-сети

Схема работы взята из записи «Understanding LSTM Networks»[2].

Главной же особенностью и преимуществом LSTM-сетей является умение работать с долгосрочными зависимостями: проще говоря, они, в отличие от обычных РНС, умеют работать с информацией, обработанной много циклов назад. Именно поэтому использование LSTM сейчас является одним из главных трендов как в оффлайн-, так и в онлайн-распознавании рукописного текста[3].

Внедрение LSTM-сетей в Tesseract привело к значительному улучшению результатов распознавания. В таблице 1 представлено позаимствованное у разработчиков сравнение результатов распознавания текста на русском языке для стандартной версии OCR (base Tesseract, тестировалась версия

[1] C. Olah. Understanding LSTM Networks // Web: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[3] см. A. Graves, J. Schmidhuber. Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks // Advances in Neural Information Processing Systems 21, NIPS, 2008, A. Graves, S. Fernandez, M. Liwicki, H. Bunke, J. Schmidhuber. Unconstrained online handwriting recognition with recurrent neural networks. // J. Platt, D. Koller, Y. Singer, and S. Roweis (eds.) Advances in Neural Information Processing Systems 20. MIT Press, Cambridge, MA, 2008.

3.04), версии, задействующей LSTM-модели, и версии, использующей LSTM и словарные данные. Одна колонка представляет процент неверно распознанных символов, вторая — неверно распознанных слов. Значения во второй колонке больше, так как вероятность ошибки в слове выше и складывается из вероятности ошибки в каждом из символов, из которых это слово состоит.

Таблица 1: Сравнение результатов распознавания текста на русском языке для разных версий Tesseract-OCR

Версия OCR	Процент ошибки при распознавании отдельного символа	Процент ошибки при распознавании слова
Tesseract	19.06 %	30.83 %
Tesseract + LSTM	4 %	15.34 %
Tesseract + LSTM + словарные данные	4.10 %	14.04 %

Данные для таблицы взяты из репозитория системы Tesseract[1]

Как мы видим, падение процента ошибки довольно существенное. Однако эти успешные результаты достигнуты, опять-таки, на материале печатного текста.

3.2 Подготовка обучающей выборки

Несмотря на то, что разработчики Tesseract-OCR давно предоставили предобученные модели не только для английского, но и для ряда других языков, в частности, для русского, их использование для распознавания рукописей Р. Г. Назирова невозможно, т. к. эти модели предназначены для распознавания печатного текста. Эксперименты с последней версией модели для русского языка на отсканированных изображениях из нашего рукописного архива показали не слишком высокие результаты (см. раздел 3.5). Поэтому приоритетной задачей являлось создание своих моделей, натренированных на распознавание почерка Р. Г. Назирова, и самым важным этапом в решении этой задачи стала подготовка обучающей выборки.

Часть рукописного архива Р. Г. Назирова, в частности, монография «Становление мифов и их историческая жизнь», оцифрована, что несколько облегчило задачу обучения классификатора. Именно эти данные мы использовали в качестве обучающей выборки.

Подготовка была разбита на следующие шаги:

1. Отбор отсканированных изображений из уже оцифрованной части архива.

2. Подготовка файлов в формате plain-text, содержащих тексты, соответствующие каждому из отобранных изображений.

3. Обработка изображений и создание т. н. «боксов». Под термином «бокс» в этой работе мы подразумеваем текстовый файл в формате box, со-

[1]https://github.com/tesseract-ocr/docs/blob/master/das_tutorial2016/7Building%20a%20Multi-Lingual%20OCR%20Engine.pdf

держащий все символы, распознанные на изображении, и координаты рамок, определяющих границы символов. Этот файл генерируется Tesseract с помощью команды batch.nochop makebox.

Обработка изображения включала в себя следующие этапы:

- бинаризация и настройка порога яркости
- увеличение резкости изображения
- удаление шумов
- конвертация в формат TIFF и переименование файла согласно необходимому формату lang.font.exp*.tif, где lang — это название языка, font — используемый шрифт, * — порядковый номер листа (примером корректного имени файла является, например, rus.Arial.exp1.tif).

Редактирование изображение осуществлялось с помощью набора консольных утилит Image Magick[1], однако сделать это можно и другими способами: например, вручную с использованием любого растрового редактора изображений (Adobe Photoshop, Gimp). Весь процесс обработки изображений и создания боксов был нами автоматизирован[2].

4. Ручное редактирование боксов с помощью JTessBoxEditor[3]. Этот этап являлся самым трудоемким и требовал большой аккуратности, так как именно от качества боксов напрямую зависит последующее качество работы обученного классификатора. В целом, Tesseract достаточно неплохо справляется с задачей выделения границ символов, но объем работы, которую нужно выполнять вручную, все еще достаточно внушителен.

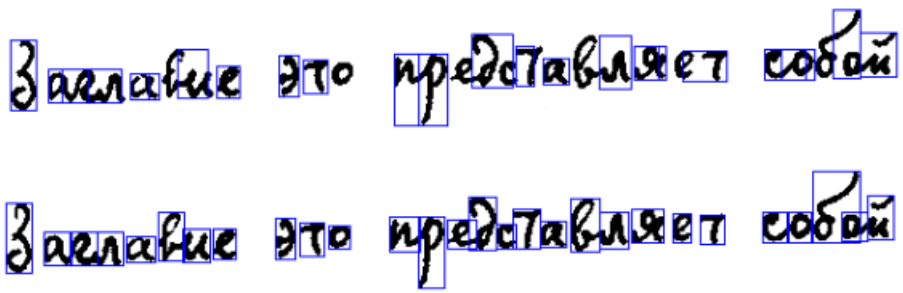

Рис. 2: Сравнение результатов выделения границ символов до ручной редактуры и после

5. Редактирование боксов с помощью скрипта *real_ letters.py*[4]. Этот скрипт заменяет символы, распознанные Tesseract, на соответствующие символы из текстового файла, содержащего текст с нужного изображения.

[1] http://www.imagemagick.org/script/index.php
[2] см. скрипт *make_ box.py* в репозитории https://github.com/deltamachine/rgn_recognition
[3] http://vietocr.sourceforge.net/training.html
[4] см. репозиторий https://github.com/deltamachine/rgn_recognition

3.2.1 Отличия в Tesseract 4.0

Подготовка обучающей выборки для Tesseract 4.0. практически полностью совпадает с подготовкой выборки для Tesseract 3.05, однако к шагам, описанным выше, добавляется ещё один важный этап: в боксах, помимо строк с координатами распознанных символов, должны присутствовать строки с координатами знаков пробела и табуляции, разграничивающих, соответственно, слова и строки. Так как Tesseract 4.0. направлен на анализ последовательностей символов, важно научить его выделять слова и строки: именно для этой цели мы изменяем боксы.

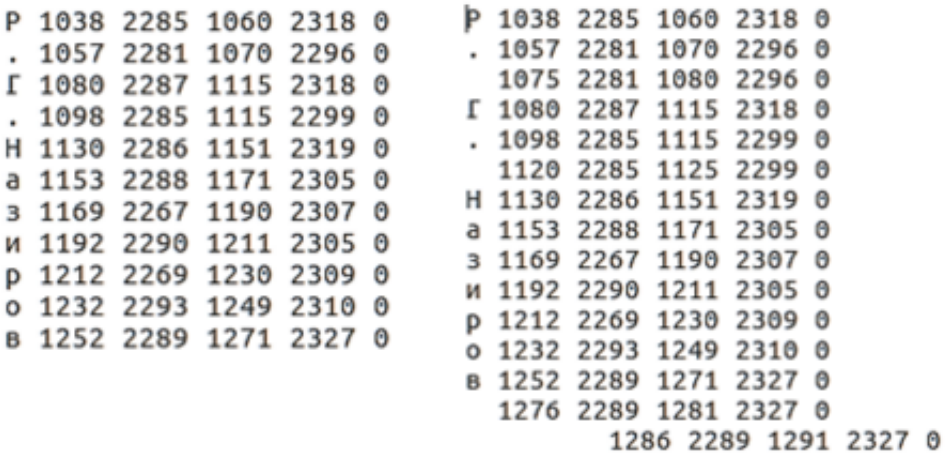

Рис. 3: Сравнение боксов, используемых в Tesseract 3.05 и Tesseract 4.0

Редактирование боксов производилось с помощью скрипта *insert_ tabs_ and_ spaces.py*[1].

3.3 Обучение Tesseract 3.05

Процесс обучения модели для распознавания текста на том или ином языке (в нашем случае — для распознавания текста, написанного конкретным почерком) в Tesseract 3.05 состоит из нескольких этапов.

Для начала необходимо запустить Tesseract в режиме обучения для каждой пары .box — .tif с помощью команды box.train. Результатом этого станут файлы в формате .tr, содержащие признаки каждого символа, имеющегося на тренировочных изображениях.

Далее генерируется файл unicharset, содержащий информацию о каждом символе, который Tesseract научился распознавать. Кроме того, необходимо вручную создать font_properties: это файл в формате plain-text, содержащий описание используемых шрифтов.

Следующий этап — кластеризация выделенных признаков и создание прототипов. Символы группируются на основании схожести признаков, после чего для каждой группы выделяется «наиболее типичный» ее предста-

[1] см. репозиторий https://github.com/deltamachine/rgn_recognition

витель — прототип. В результате этого шага создаются файлы shapetable и normproto.

После этого к данным добавляются словари: несмотря на то, что это не является обязательным шагом, словарные данные помогают улучшить качество распознавания. Информация об использованных словарях представлена в таблице 2:

Таблица 2: Словарные данные, использованные для обучения Tesseract 3.05

Словарный файл	Формат	Содержание	Способ создания
word_list	plain-text	Словарь распознаваемого языка (в нашем случае — список всех словоупотреблений в монографии «Становление мифов и их историческая жизнь»	Был создан из файла, содержащего полный текст монографии, с помощью скрипта create_wordlist.py
bigram_list	plain-text	Список всех биграмм, встречающихся в этой же монографии.	Был создан из файла, содержащего полный текст монографии, с помощью программы AntConc[1].
punc-dawg	DAWG	Файл, содержащий описание всех возможных вариантов расстановки знаков препинания.	Был взят из готовой модели, обученной на русском языке.
number-dawg	DAWG	Файл, содержащий токены, в состав которых входят цифры.	Был взят из готовой модели, обученной на русском языке.

Файлы в формате plain-text конвертируются в формат DAWG (Directed Acyclic Word Graph).

Далее (этот шаг, опять же, необязателен) вручную создается файл unicharambigs, содержащий описание неопределенностей, которые могут возникнуть при распознавании символов.

В конце происходит объединение всех подготовленных файлов в одну модель: на выходе мы получаем один файл rgn.traineddata, финальную предобученную модель, готовую к использованию для распознавания текста.

Весь процесс обучения Tesseract 3.05 также был нами автоматизирован: см. скрипт *train_tesseract3.py*[2]. Кроме того нами была составлена инструкция с подробным описанием каждого этапа тренировки[3].

3.4 Обучение Tesseract 4.0

На данный момент в открытом доступе находится только альфа-версия Tesseract 4.0. Официальный релиз планируется на вторую половину 2017 го-

[2]https://github.com/deltamachine/rgn_recognition

[3]см. https://github.com/deltamachine/rgn_recognition/wiki/tesseract_tutorials

да[4]. В связи с этим настоящий раздел статьи является, скорее, экспериментальной попыткой оценить, на что потенциально будет способен Tesseract 4.0.

В Tesseract 4.0 существует три способа обучения модели:

1. Создание новой предобученной модели с нуля (training from scratch). Для того, чтобы получить хорошие результаты при использовании такого метода, необходимо сравнительно большое количество данных для обучения. Новую модель имеет смысл создавать, если перед исследователем стоит задача распознавания языка, для которого разработчики Tesseract еще не предоставили готового модуля.

2. Тонкая настройка (fine tuning). Этот метод подразумевает обучение готовой модели специфичному шрифту. По утверждению разработчиков, при использовании тонкой настройки не обязательно иметь большую тренировочную выборку для того, чтобы получить сравнительно неплохие результаты.

3. Замена верхних слоёв (replacing top layers). Суть этого способа заключается в замене верхних слоев существующей LSTM-модели на новые и обучении модели на новых данных. Этот способ используется в том случае, когда в новых данных встречаются символы, которых нет в исходной модели. Примером использования этого метода может являться добавление отсутствующей буквы Æ в данные, используемые предобученной моделью для норвежского языка[1].

Мы приняли решение воспользоваться третьим методом, так как в наших данных присутствуют как кириллические, так и латинские символы.

Главной проблемой, с которой мы столкнулись при использовании с Tesseract 4.0, стало то, что эта версия программы пока не предназначена для работы с пользовательскими .box — .tif парами. На данный момент процесс подготовки данных для обучения LSTM-модели выглядит следующим образом: боксы и изображения генерируются Tesseract автоматически при помощи инструмента tesstrain.sh, который получает на вход тренировочный текст, подготовленный пользователем и список предустановленных шрифтов. Далее на основе этих данных создаются .lstmf-файлы, на которых уже и происходит обучение.

Поэтому для «искусственной» генерации нужных файлов из имеющихся .box — .tif пар мы воспользовались модифицированным одним из разработчиков скриптом boxtrain.sh.

На этом этапе перед нами возникла ещё проблема: нам не удалось использовать все имеющиеся данные, т. к. для нескольких пар создание .lstmf-файлов завершилось ошибкой. Выяснить причину этой ошибки пока не удалось: как уже упоминалось ранее, работа над Tesseract 4.0 еще не завершена. Поэтому на этом этапе было использовано лишь 6 листов с данными, в отличие от Tesseract 3.05, натренированного на 10 листах.

Следующим этапом стало непосредственно обучение самой LSTM-модели. За основу была взята предобученная LSTM-сеть из последней версии мо-

[4]https://github.com/tesseract-ocr/tesseract/wiki/ReleaseNotes#in-development

[1]https://github.com/tesseract-ocr/tesseract/wiki/TrainingTesseract-4.00—Replacing-Top-Layer-Example

дуля для русского языка и файл unicharset, сгенерированный в процессе работы с Tesseract 3.05 (см. раздел 4.3).

Само обучение запускается с помощью скрипта, принимающего на вход путь к модели, которую нужно изменить (например, rus.lstm), путь к файлу unicharset, путь к папке, в которой предполагается сохранять новые модели и некоторые дополнительные параметры для обучения. Параметры, использованные нами, представлены в таблице 3:

Таблица 3: Параметры, использованные при обучении Tesseract 4.0

Параметр	Описание	Значение
append_index	Количество слоёв, которые нужно заменить в исходной модели	5
net_spec	Топология новых слоёв	[Lfx256 O1c105]
learning_rate	Коэффициент скорости обучения	10e-5
max_iterations	Число итераций, после которого обучение автоматически прекращается	20000

После того, как обучение было закончено, старая LSTM-модель в предобученном модуле для русского языка была заменена на новую, а также были добавлены новые словарные данные word-dawg и bigram-dawg — те же, что и для Tesseract 3.05 (см. раздел 3.3).

Как и в случае с Tesseract 3.05, нами был составлен подробный тьюториал для Tesseract 4.0[1].

3.5 Результаты

После обучения обе новые модели, подготовленные нами, были протестированы на фрагментах рукописей Р. Г. Назирова.

Результаты оценивались следующим образом: верно выделенным сегментом мы считали сегмент, в котором число распознанных символов совпадало с реальным числом и хотя бы несколько символов было классифицированно верно. К верно распознанным сегментам мы относили как абсолютно точно распознанные фрагменты, так и случаи, когда одна или несколько букв в слове были ошибочно определены как заглавные (например, «Мало» вместо «мало»). При подсчете процента верно распознанных символов строчная буква, ошибочно распознанная как заглавная (и наоборот), оценивалась как наполовину верно распознанная и прибавляла 0.5 к общей сумме верных ответов, данных классификатором.

Сначала мы попробовали распознать изображение с отдельной строкой и сравнить результаты, которых удалось добиться при использовании новых моделей, с результатами, которые показывают готовые модули, предоставленные разработчиками Tesseract.

Следующим экспериментом стала попытка распознать лист рукописи целиком. На 3.5 и в таблицах 6 и 7 представлены, соответственно, фрагмент этого листа и результаты его распознавания.

Можно сделать следующие выводы:

[1] https://github.com/deltamachine/rgn_recognition/wiki/tesseract_tutorials

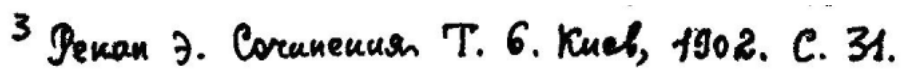

Рис. 4: Фрагмент рукописи Р. Г. Назирова

Таблица 4: Результат распознавания, показанный протестированными моделями

Модель	**Результат распознавания**
rus 3.05	3 З:. . . З. шиизма. . . 'Т. С. Ю“ 1902. С. 34.
rgn 3.05	3 Ренаи эк Омсннгнияп !та 67 Киоив, 1902о Ск 31к
rus 4.0	3 Фекои Э. бучанеки&„ Т. 6. Кие4, 4902. С. 34.
rgn 4.0	3 Фекои Э. бучанеки&„ Т. 6. Кие4, 4902. С. 34.

- Модели Tesseract 4.0 значительно лучше справляются с задачей выделения сегментов текста и разбиения их на символы, кроме того, у них заметно больше полностью правильно распознанных сегментов.
- Качество распознавания моделей Tesseract 4.0 и новой модели Tesseract 3.05 более или менее сопоставимо, однако при этом стоит учитывать тот факт, что Tesseract 4.0 обучался на меньшем количестве данных.
- Tesseract 4.0 лучше справляется с задачей отнесения того или иного символа к конкретной строке: модели Tesseract 3.05 игнорируют символ в том случае, если один из его элементов заходит на другую строку («р» в «перестали», «у» в «умножении»), а модели Tesseract 4.0 успешно выделяют и распознают его.
- Классификаторы допускают вполне логичные и ожидаемые ошибки, путая «Р» и «Ф», «к» и «н», «н» и «и», строчные и заглавные буквы.

4 Заключение

Распознавание русского рукописного текста является задачей, которая на данный момент с трудом поддается решению.

Вероятно, нам удалось бы добиться лучших результатов, если бы мы обладали большим количеством данных, но подготовка достаточно крупной тренировочной выборки силами одного человека практически невозможна, даже несмотря на то, что мы частично автоматизировали процесс. Возможным выходом в данной ситуации может стать привлечение к проекту большой группы добровольцев .

Также надежду на возможное достижение высокого процента распознавания почерка Р. Г. Назирова в ближайшем будущем дают результаты, показанные экспериментальными моделями Tesseract 4.0. Несомненно, имеет смысл вернуться к работе с этой версией после официального релиза и попробовать улучшить имеющиеся результаты.

Однако из сказанного должно быть очевидно, что на текущий момент автоматическое распознавание рукописей из архива Р. Г. Назирова с должным качеством невозможно.

Таблица 5: Точность распознавания фрагмента для протестированных моделей

Модель	% верно выделенных сегментов	% верно распознанных сегментов	% верно распознанных символов
rus 3.05	80 %	40 %	40 %
rgn 3.05	80 %	10 %	56.8 %
rus 4.0	100 %	50 %	62.2 %
rgn 4.0	100 %	50 %	62.2 %

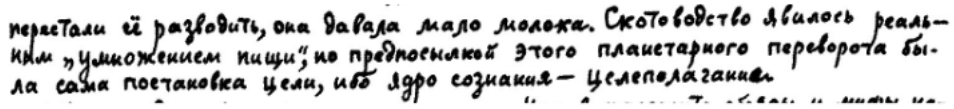

Рис. 5: Фрагмент листа рукописи

Таблица 6: Результат распознавания, показанный протестированными моделями

Модель	**Результат распознавания**
rus 3.05	ис цТМц & ауди“, за: %.?ауш мало ЖМЖ». СкоТв‘о9с1го „Мама „„„„ ц ш „ що ищем пищи“, ио р&мшфта‘ этом макс-ирис три./, . 31. м св « мимо!“ ции, и о 4370 служил — цитатами
rgn 3.05	ne итаяи Ооево auooaaeb, она ОаэваЛа жаипо Мояомкаво Скотоводтво ои«вияоо eaouoo и м мно ением пищиЗв но Водновылкоав этою ПЛаиеторпого Пф;овох, а !во яа са а moевтановка цеаи, и о Луро и«экиакввяоа— Цеми«оояаёанаи
rus 4.0	уерaеТали +4 Ока даЁада Мало молока. СкеТо (обстЁо А{ало©Ь реаль- „> умиожением паши") ие р&мшфта‘ Этого плаиетариего три./, а Д- Ла соби поечакобеа Цели, ибо Аоро созиаки& — аки.
rgn 4.0	уерaеТали +4 бка даЁада ЛМмадло меолока. СкеТо (обстЁо А{ало©Ь реаль- „> умиожением паци’) ие 9к333793 Этого плаиетариего ‚Счтх» а Ла соба поечакобеа Цели, Аоро созиаки& — 2аки@.

Таблица 7: Точность распознавания фрагмента для протестированных моделей

Модель	% верно выделенных сегментов	% верно распознанных сегментов	% верно распознанных символов
rus 3.05	28.6 %	7.1 %	20.2 %
rgn 3.05	32.1 %	7.1 %	53.6 %
rus 4.0	71.4 %	28.6 %	51.9 %
rgn 4.0	42.9 %	14.3 %	46.4 %