Санкт-Петербургский государственный университет
Факультет прикладной математики — процессов управления

Авдеева Анастасия Сергеевна

Выпускная квалификационная работа

Разработка методов распознавания математического текста

Направление 010400
Прикладная математика, фундаментальная информатика и программирование

Научный руководитель,
старший преподаватель Стученков А. Б.

Санкт-Петербург
2017

Содержание

Введение 2
Глава 1. Предварительная обработка изображения документа . . 5
1.1. Удаление шума 5
1.1.1. Обзор методов, выбор оптимального алгоритма 6
1.2. Сегментация и структурный анализ изображения документа 8
1.2.1. Обзор методов 9
1.2.2. Docstrum 11
Глава 2. Формирование структуры строк документа и выражений 17
2.1. Уточнение границ 17
2.2. Сегментация выражений 18
2.2.1. Обзор методов 19
2.2.2. Предложенный метод 20
Глава 3. Распознавание символов и текста 25
3.1. Нейронная сеть 25
3.2. Касающиеся символы 26
3.3. Обработка полученного результата 27
Глава 4. Формирование результата 29
4.1. Формирование макета страницы 29
4.2. Обработка выражений 29
Глава 5. Тестирование, оценка результатов, интерфейс 30
5.1. Тестовые данные 30
5.2. Оценка результата 30
5.3. Интерфейс и реализация 31
Заключение 33
Список литературы 34

Введение

В настоящее время задача распознавания образов актуальна во многих сферах, например — распознавание автомобильных номеров, биометрических данных, текстовой информации — банковские чеки, почтовые адреса, различные бланки и документы. Распознавание документов является непростой задачей, так как помимо задачи корректного распознавания символов и формирования слов возникает задача правильной сегментации, определение шаблона документа — выделение колонок, извлечение изображений, таблиц, графиков и другой нетекстовой информации.

Так же в настоящее время широко распространенно использование электронной информации. Существует множество электронных библиотек, но не все книги и статьи доступны в таком варианте. Множество научных изданий содержит сложные математические формулы и выражения, но большинство существующих систем распознавания не способны представить корректный результат, так как для распознавания двумерной структуры формул необходимы несколько иные методы. Решение задачи распознавания математического текста могло бы дать возможность для перевода книг и статей в цифровой вид, а так же возможность откорректировать информацию перед сохранением.

Обзор существующих решений

На сегодняшний день существует много вариантов программного обеспечения для распознавания печатного текста. Коммерческие программы — Abbyy FineReader [1], OmniPage [2], Acrobat Capture, OCR, font, and page recognition [3]. Эти программы имеют большую функциональность, высокую точность распознавания, поддерживают большое число языков,

например, Abby FineReader обеспечивает поддержку 192 языков. Из свободно распространяемых программ можно отметить Tesseract [4], который имеет открытый исходный код и обеспечивает поддержку 100 языков.

Но все описанные решения либо совсем не обеспечивают распознавание математических символов, либо, как Abby FineReader, поддерживают простые строчные формулы, допуская только верхние и нижние индексы. InftyReader [5] — коммерческое программное обеспечение для распознавания математических формул и текстов, которое предлагает выходной текст в 3 форматах — LaTeX, MathML и XHTML.

Постановка задачи

Цель данной работы – разработать систему, способную распознать математический текст на английском языке и представить результат в формате LaTeX.

Для успешного решения задачи распознавания документа необходимо решить несколько подзадач:

1. Анализ и подготовка изображения страницы: удаление шума, определение угла наклона содержимого страницы, сегментация — разбиение на текстовые и нетекстовые блоки.
2. Формирование структуры документа.
3. Оптическое распознавание извлеченных символов.
4. Формирование и вывод результата.

Данная работа построена следующим образом. Примерно, каждая глава описывает одну из стадий (изображено на рис. 1) обработки документа, проблематику задач данного этапа, методы их решения, реализацию или адаптацию подходящего алгоритма. В Главе 1 описываются методы предварительной обработки документа — удаление шума, определение угла иска-

жения, сегментация на блоки, в Главе 2 описывается структурный анализ, в Главе 3 дается архитектура нейронной сети, описывается распознавание символов, Глава 4 описывает модуль, генерирующий выходной результат. Глава 5 представляет некоторые результаты и разработанный интерфейс.

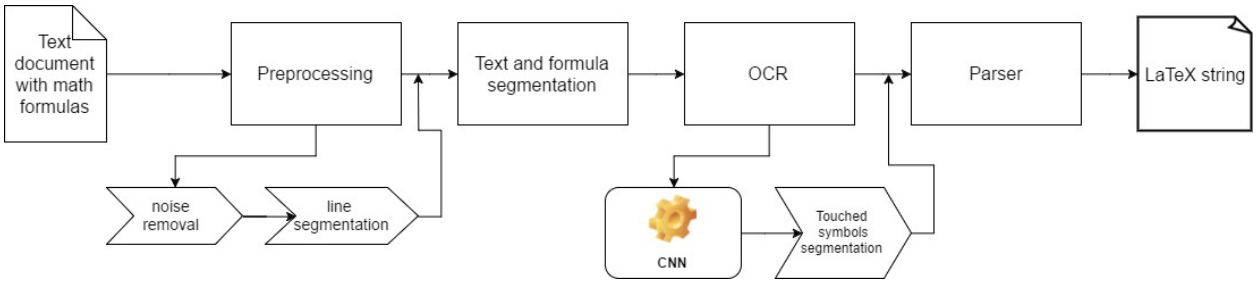

Рис. 1. Блок-схема

Глава 1. Предварительная обработка изображения документа

Предварительная обработка — необходимый этап в распознавании изображения документа, так как, в то время как удаление шума может лишь повысить результаты распознавания, то не решив задачу сегментации, невозможно предоставить корректный результат.

1.1. Удаление шума

Одна из задач обработки изображения документа — удаление шума, который возникает из-за качества бумаги, процесса сканирования или воздействия еще каких-либо факторов. Цель этого шага — улучшение качества изображения для повышения точности сегментации и распознавания символов.

Решить данную задачу в общем случае достаточно трудно, так как шум может быть разных видов: например, при сканировании некачественных или старых документов или при сканировании с низким разрешением может возникнуть шум типа «соль и перец», при сканировании страниц книг могут возникать крупные черные области по бокам изображения. Так же сильный шум «соль и перец» достаточно трудно удалить полностью, не повредив текст. Некоторый обзор видов шума, возникающих в текстовых документах, и алгоритмов удаления даётся в [6].

В данной работе основное внимание уделено алгоритмам и методам борьбы с шумом типа «соль и перец», так как это наиболее часто встречающийся вид шума в отсканированных печатных документах.

1.1.1. Обзор методов, выбор оптимального алгоритма

Стандартным алгоритмом борьбы с данным типом шума является медианный фильтр, адаптивный медианный фильтр и различные модификации [7], а так же модификации для избавления от шума высокой плотности, например, предложенные в [8] и [9]. Недостаток стандартного медианного фильтра применимо к задаче обработке текстового документа состоит в том, что для устранения крупного шума необходимо взять больший размер окна фильтра, а это свою очередь может вызвать размытие и эрозию текста, что отрицательным образом скажется на результатах распознавания.

В [10] предложен альтернативный метод, использующий coordinate logic filter. Фильтр представлен в виде логических операций над результатами эрозии и дилатации, которые реализованы применением логического умножения на некоторую маску двоичного представления всех значений пикселей изображения.

Другой метод — алгоритм kFill, описанный в [11], предложен как метод специализированный для обработки бинарных текстовых изображений. Суть метода в передвижении окна размера k вдоль изображения и заполнении «ядра» — внутренних $(k-2)^2$ ячеек — противоположным цветом, в зависимости от выполнения условия, о котором будет сказано далее. Также необходимо сказать, что «ядро» может быть заполнено, как белым, так и черным цветом. Для каждого положения окна рассчитываются 3 величины для $4(n-k)$ внешних ячеек: c — число компонент связанности черного (белого) цвета, n — число пикселей черного (белого), r — число угловых пикселей черного (белого) цвета. Заполнение происходит при выполнении 1.1:

$$c = 1 \wedge (n > 3k - 4 \vee (n = 3k - 4 \wedge r = 2)) \tag{1.1}$$

Алгоритм является итеративным и выполняется, пока не произойдет итерация, где условие не выполнится ни разу. Также в [12] рассмотрено некоторое улучшение данного метода, с целью сделать возможным уничтожение одиночного шума при больших размерах фильтра k и достигнуть результата за одну итерацию.

В ходе исследования было реализовано два алгоритма — адаптивный медианный фильтр и улучшенный алгоритм kFill. В итоге тестирования было отмечено, что kFill лучше справляется с шумом, но так же сильнее размывает символы и может вызывать их разъединение, что критически сказывается на результатах распознавания. Поэтому был использован адаптивный медианный фильтр, который хуже борется с шумом, однако лучше сохраняет границы символов.

1.2. Сегментация и структурный анализ изображения документа

Перед непосредственно распознаванием символов необходимо решить ряд задач для получения структуры документа. Изображение документа состоит из компонент связанности — символов, которые образуют слова, текстовые линии и блоки. Задача сегментации и структурного анализа — найти и извлечь нетекстовую информацию, разбить текстовую на блоки, если они присутствуют, выделить текстовые линии. Так же сюда можно отнести задачу определения угла искажения документа.

На сегодняшний день существует несколько алгоритмов, решающих данные задачи. Перед их рассмотрением необходимо ввести несколько понятий.

«Манхэттенский» layout — такой, что текст можно разбить на прямоугольные блоки.

«Неманхэттенский» layout — соответственно такой, что разделение на прямоугольные блоки невозможно.

Также стоит выделить задачу определения угла наклона, так как некоторые алгоритмы сегментации работают только с неискривленными изображениями. Алгоритмы для решения этой задачи обычно используют профили проекции [13], преобразование Хафа [14] или основаны на корреляции между строками [15]. После определения угла наклона изображение поворачивается на нужный угол и решается задача сегментации. Однако ниже будут рассмотрены алгоритмы, инвариантные относительно угла наклона и позволяющие вычислить его одновременно с сегментацией.

1.2.1. Обзор методов

Некоторый обзор предложен в [16] — рассмотрены 6 основных методов сегментации текста, а так же проводится их анализ и сравнение.

По принципу работы алгоритма все методы можно разделить на три группы — top-down, bottom-up и гибридные методы. Top-down подход — это подход постепенного разбиения страницы на меньшие блоки, начиная со всей страницы целиком; bottom-up — обратная к нему идея, сегменты формируются начиная с неразделимых структур — компонент связанности или даже пикселей; соответственно гибридный подход – комбинация данных методов. Результат работы алгоритмов представляет сегментацию на блоки - колонки текста, крупные заголовки и т. д. Ниже дается небольшой обзор существующих методов.

Методы, работающие с «манхэттенским» layout:

1. X-Y cut [17] — алгоритм, строящий дерево документа, корнем которого является сам документ, на основе рекурсивного деления блоков с некоторым пороговым значением. На каждом шаге для каждого блока считаются вертикальный и горизонтальный профили проекции и в местах, где значения меньшего некоторого порогового значения, разбиваются на подблоки. Соответственно процесс происходит рекурсивно, пока каждый подблок не окажется неразделимым.
2. Smearing [18] — алгоритм, работающий на бинарных изображениях в RLE-представлении. RLE-представление — представление последовательности в виде пар, содержащих значение элемента и число его повторений подряд в этой последовательности. Пусть белый пиксель — 0, черный — 1. Тогда подпоследовательности нулей меняются на единицы, если длина подпоследовательности больше некоторого порога. Данная операция применяется горизонтально и вертикально вдоль изображе-

ния, к двум полученным изображениям применяется логическое умножение и затем, к результирующему изображению, ещё раз горизонтально применяется описанная операция.

3. В [19] описан алгоритм сегментации с помощь максимальных белых прямоугольников. На изображении отыскиваются максимальные белые прямоугольники, задается некоторый функционал качества, с помощью которого они упорядочиваются. До удовлетворения заданного критерия, прямоугольники достаются из списка и области вырезаются из изображения. В итоге оставшиеся области представляют сегментацию изображения.
4. [20] предлагает сегментацию при помощи фильтра Габора.

Более интересными представляются следующие два алгоритма, предложенные как способные сегментировать и «неманхэтеннский» layout.

1. В [21] описывается алгоритм основанный на построении диаграммы Вороного. Диаграмма Вороного – такое разбиение, при котором каждой области принадлежат точки, ближайшие к одному из элементов области. Суть алгоритма в том, что на основе компонент связанности, найденных на изображении строится диаграмма Вороного. Далее, ребра между двумя компонентами связанности удаляются, если расстояние между центрами ячеек удовлетворяет некоторым заданным соотношения и порогам.
2. Docstrum [22] подробнее изложен ниже с описанием некоторой адаптации под данную задачу.

Также в [23] рассматривается метод, основанный на интеграции двух вышеупомянутых алгоритмах.

1.2.2. Docstrum

Данный метод впервые был предложен в [22], он основан на кластеризации элементов на k-ближайших соседей и анализе расстояний между ними.

Основные шаги алгоритма:

1. Предварительная обработка — очищение изображения от мелкого шума и извлечение крупных, явно нетекстовых, компонент.
2. Выделение компонент связанности и вычисление для каждой координат ограничивающего прямоугольника и его центра.
3. Для каждой компоненты связанности находятся k-ближайших соседей. В качестве расстояния рассматривается евклидово расстояние между центрами ограничивающих прямоугольников. Так же есть альтернативный вариант, где рассматривается расстояние между центроидами компонент связанности.
4. Для каждой пары соседей рассчитывается также угол между центрами (или центроидами).
5. Строится гистограмма с некоторым шагом, которая показывает распределения значений углов между компонентами. Пик гистограммы будет соответствовать среднему углу ориентации линий текста. Эта оценка будет уточняться в следующих шагах.
6. Для оценки расстояний строится две гистограммы — одна для расстояний между элементами угол которых лежит в некоторой окрестности угла, найденного на прошлом шаге, другая — в окрестности угла, перпендикулярного к нему.
7. Используя как пороговое значение пик первой гистограммы прошлого шага, проводится транзитивное замыкание соседних компонент, объединяя компоненты связанности в «слова» в текстовых линиях.

8. Методом наименьших квадратов определяются базовые линии (рис. 2) компонент, найденных на предыдущем шаге, и по ним строится уже более точная оценка угла наклона.

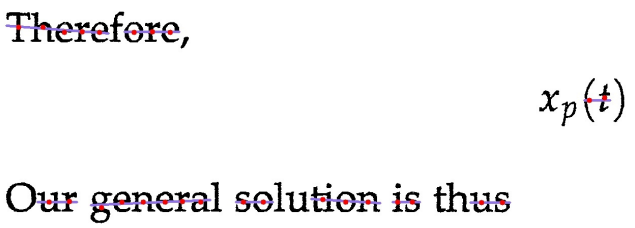

Рис. 2. Линии, найденные регрессией

9. Для каждой пары найденных линий вычисляется 4 значения — разница угла наклона, перпендикулярное расстояние, параллельное расстояние и длина перекрытия. Линии приблизительно параллельные, близкие в перпендикулярной дистанции и близкие в параллельной или имеющие положительную длину перекрытия считаются принадлежащими одному блоку. Длина перекрытия между отрезками i и j рассчитывается так: концы отрезка i проецируются на линию, которой принадлежит отрезок j и, если, спроецированный отрезок пересекает отрезок j, рассматривается их общая часть, иначе — отрезок, лежащий между ними. Если есть пересечение отрезков — длина считается положительной, иначе отрицательной. Точные формулы данных расчетов приводятся в [22]. Параллельное расстояние составляет длина полученного отрезка. Перпендикулярное расстояние рассматривается как расстояние между серединой найденного отрезка и прямой, которой принадлежит отрезок i.

Алгоритм был с использован, с некоторыми модификациями и дополнениями, с целью выделить текстовые блоки, линии, группы слов и математические выражения, особенность которых в том, что они могут быть представлены несколькими линиями. Docstrum был выбран как оптималь-

ное решение поставленной на данном этапе задачи в силу следующих причин. Статистика, найденная в ходе его работы, может быть использована для выделения линий и выражений, метод позволяет вычислить угол искажения изображения — отпадает необходимость совершать дополнительные вычислительные затраты. Кроме этого, в ходе работы был реализован поиск угла искажения с помощью трансформации Хафа и данный алгоритм показал более точные результаты.

В оригинальной статье предложены параметры, корректно работающие на тестовой выборке. Некоторые из них необходимо изменить в силу некоторых особенностей рассматриваемого типа документов. В качестве k-ближайших соседей предлагается взять $k = 5$ – в теории это означает, что, скорее всего, будут рассмотрены соседи снизу, сверху, слева и справа и один дополнительно. Но в математических документах крупные формулы часто размещаются посередине строки и имеют достаточно большой отступ справа и слева - для многих компонент среди ближайших соседей может не оказаться ни одного снизу или сверху, поэтому было взято $k = 7$. Разница проиллюстрирована на рис. 3 и рис. 4. Зеленым цветом выделен главный элемент, синим – его соседи.

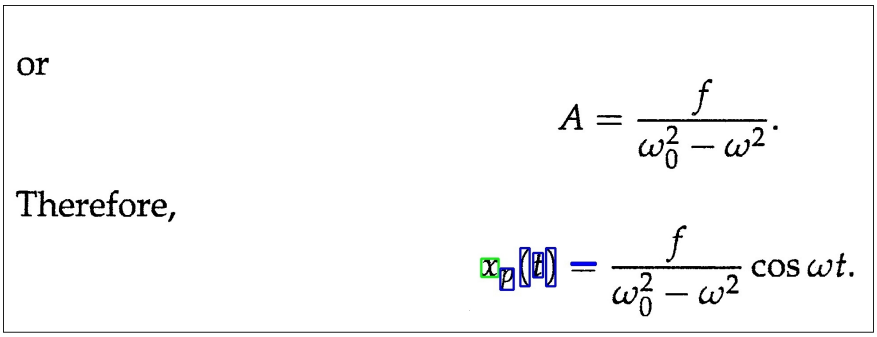

Рис. 3. 5 ближайших соседей

Оценки для определения принадлежности блокам были взяты из оригинальной статьи.

Для определения принадлежности двух фрагментов одной линии или

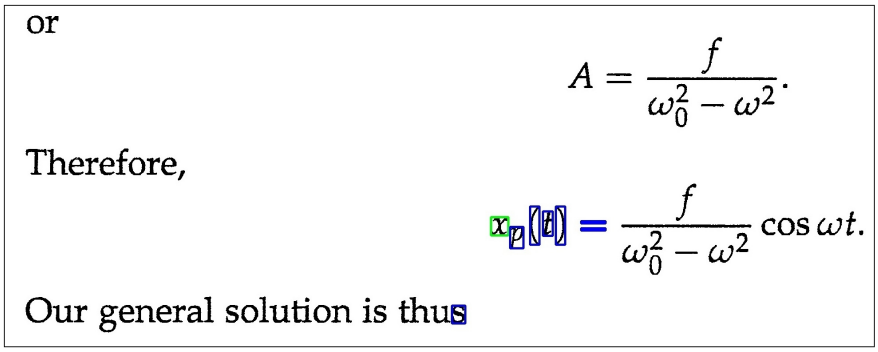

Рис. 4. 7 ближайших соседей

выражению были предложены следующие критерии. Будем считать, что далее речь идет о рассмотрении пар фрагментов, уже определенных принадлежащими одному блоку.

Обозначим h — среднюю величину символа в документе, рассчитанную следующим образом: из длин всех найденных компонент связанности составляется интервальный вариационный ряд, строится гистограмма частот и за h берется её пик. Если рассматриваемая пара фрагментов имеет положительное расстояние перекрытия и перпендикулярная дистанция меньше либо равна αh, то данные фрагменты считаются принадлежащими одному выражению. Если рассматриваемая пара сегментов имеет отрицательное расстояние перекрытия и перпендикулярная дистанция меньше либо равна βh, то данные фрагменты считаются принадлежащими одной линии.

Также стоит отметить, что точки над буквами «i», «j» или в качестве символа производной могут вызвать нежелательное объединение линий, так располагаются между двух линий и дистанция между символом и обеими может быть меньше пороговой. В целях предотвращения этой ситуации рассчитывается число $h_{min} = min(\frac{h}{2}, 1.5h_1)$, где h_1 - соответствует наибольшей частоте из гистограммы, меньшей $\frac{h}{2}$. Фрагменты, высота и ширина, которых меньше или равна h_{min} соединяются с ближайшим к ним, если находится такой, удовлетворяющий критериям.

Коэффициенты α, β были подобраны тестовым путем и составили 1.7 и 0.7 соответственно. Некоторые результаты представлены ниже. Рис. 5 представляет найденные блоки, на рис. 6 показаны найденные границы выражений. Видно, что линии не нарушены, но некоторые выражения оказались разделенными на несколько частей, об их объедении будет сказано ниже, но важно отметить, что такие элементы как дроби оказались объединены в одно выражение, что и было одной из основных целей данного этапа.

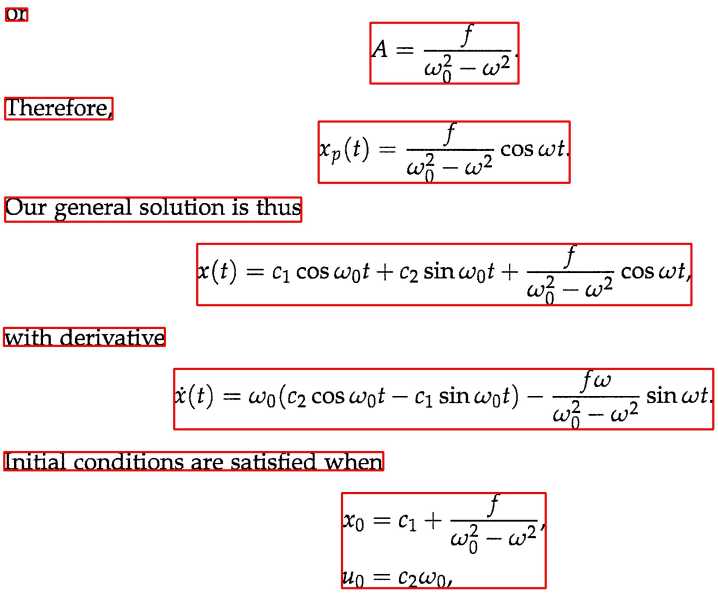

Рис. 5. Найденные границы блоков

Рис. 7 демонстрирует, что использование вышеприведенных критериев в общем случае не объединяет обычные текстовые линии.

Therefore, the solution to the ode that satisfies the initial conditions is

$$x(t) = \left(x_0 - \frac{f}{\omega_0^2 - \omega^2}\right)\cos\omega_0 t + \frac{u_0}{\omega_0}\sin\omega_0 t + \frac{f}{\omega_0^2 - \omega^2}\cos\omega t$$
$$= x_0\cos\omega_0 t + \frac{u_0}{\omega_0}\sin\omega_0 t + \frac{f(\cos\omega t - \cos\omega_0 t)}{\omega_0^2 - \omega^2}.$$

where we have grouped together terms proportional to the forcing amplitude f.

Resonance occurs in the limit $\omega \to \omega_0$; that is, the frequency of the inhomogeneous term (the external force) matches the frequency of the homogeneous solution (the free oscillation). By L'Hospital's rule, the limit of the term proportional to f is found by differentiating with respect to ω:

$$\lim_{\omega\to\omega_0}\frac{f(\cos\omega t - \cos\omega_0 t)}{\omega_0^2 - \omega^2} = \lim_{\omega\to\omega_0}\frac{-ft\sin\omega t}{-2\omega} = \frac{ft\sin\omega_0 t}{2\omega_0}. \tag{3.23}$$

Рис. 6. Найденные границы выражений

where we have grouped together terms proportional to the forcing amplitude f.

Resonance occurs in the limit $\omega \to \omega_0$; that is, the frequency of the inhomogeneous term (the external force) matches the frequency of the homogeneous solution (the free oscillation). By L'Hospital's rule, the limit of the term proportional to f is found by differentiating with respect to ω:

$$\lim_{\omega\to\omega_0}\frac{f(\cos\omega t - \cos\omega_0 t)}{\omega_0^2 - \omega^2} = \lim_{\omega\to\omega_0}\frac{-ft\sin\omega t}{-2\omega} = \frac{ft\sin\omega_0 t}{2\omega_0}. \tag{3.23}$$

At resonance, the term proportional to the amplitude f of the inhomogeneous term increases linearly with t, resulting in larger-and-larger amplitudes of oscillation for $x(t)$. In general, if the inhomogeneous term in the differential equation is a solution of the corresponding homogeneous differential equation, then the correct ansatz for the particular solution is a constant times the inhomogeneous term times t.

To illustrate this same example further, we return to the original ode, now assumed to be exactly at resonance,

$$\ddot{x} + \omega_0^2 x = f\cos\omega_0 t,$$

Рис. 7. Найденные линии

Глава 2. Формирование структуры строк документа и выражений

Данная глава описывает формирование структуры блоков, полученных на предыдущем шаге и построение дерева для каждого выражения. Здесь и далее будем считать, что получено представление страницы в виде блоков, состоящих из массива линий, каждая из которых в свою очередь состоит из выражений. Выражением будем называть набор символов, определенных предыдущей частью алгоритма, как наиболее близких, обычно это слово или несколько слов, или часть математической формулы.

2.1. Уточнение границ

Несмотря на то, что метод, описанный в предыдущей главе работает довольно неплохо, в некоторых случаях, в основном из-за верхних или нижних индексов, некоторые текстовые линии могут быть объединены алгоритмом в одну, пример такой ситуации показан на рис. 8. Для борьбы с этим была предложена следующая эвристика. Для каждой линии блока рассматривается профиль её горизонтальной проекции — на рис. 9 показан график для двух последних линий рис. 8. Далее находятся всё точки, равные нулю, идущие подряд объединяются в отрезок, который представляется центральной точкой; затем для всех таких точек, исключая крайние, если они совпадают с координатами ограничивающего прямоугольника линии, рассматривается длина отрезков сверху и снизу, если для для каждого из них длина больше h — средней высоты символа в тексте, точка считается разделяющей. После отыскания всех таких точек для линии, она и её вложенных сегменты разбиваются линиями, горизонтально проходящими через найденные точки, на пары новых.

of the eigenvalues of the Jacobian from negative real part to positive real part can be seen if we transform these equations to cartesian coordinates. We have using $r^2 = x^2 + y^2$,

Рис. 8. Ошибочно объединенные текстовые линии

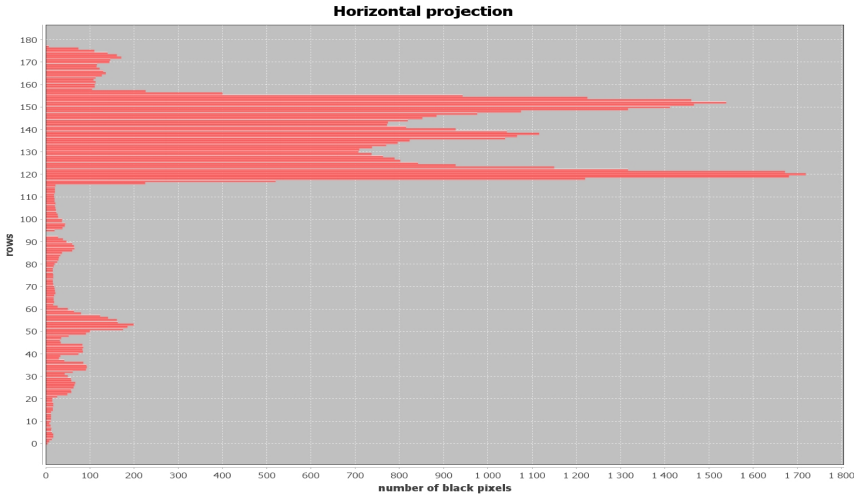

Рис. 9. Горизонтальный профиль проекции строк

Результат работы для линий рис. 8 показан на рис. 10

fixed point is stable for $\mu < 0$ and the latter is stable for $\mu > 0$. The transition of the eigenvalues of the Jacobian from negative real part to positive real part can be seen if we transform these equations to cartesian coordinates. We have using $r^2 = x^2 + y^2$,

Рис. 10. Результат после уточнения границ

2.2. Сегментация выражений

В данном подпараграфе описывается модуль, реализованный для распознавания структуры выражений, а так же приводится некоторый обзор решений, используемых для распознавания двумерной структуры математических выражений.

2.2.1. Обзор методов

Достаточно большой обзор дается в [24]. Многие работы посвящены распознаванию рукописных выражений — эта задача сложнее, так как возникает больше сложностей с сегментацией символов и распознаванием, но проблемы анализа структуры остаются такими же. В основном для решения задачи сегментации двумерной структуры используется bottom-up подход — выделяются компоненты связанности и задача ставится в определении пространственных отношений между ними. Обычно для хранения используется какая-нибудь пригодная для этих целей структура — графы или деревья. Например, в [25] выражение представляется в виде графа, вершинами которого являются компоненты связанности, а ребрами отношения между ними; граф перестраивается по заданным правилам, в результате получая структуру выражения. В [26] предлагается метод, использующий дерево — на первой итерации оно строится на основе пространственных взаимоотношений между символами, затем определяются семантические отношения между символами и дерево при необходимости перестраивается.

Концептуально другая группа методов, разработанных для распознавания математических выражений, — методы, основанные на контекстно-свободных грамматиках и нечетких множествах. Эта идея интересна использованием одновременно и информации о пространственном положении соседних символов и четких правил их семантических взаимоотношений. Однако, возникают некоторые проблемы с заданием грамматики и правил, так как математическая нотация, во-первых, достаточно обширна, во-вторых, двумерна. [27] описывает применение 2D-грамматик, строя двумерную сеть, однако требуется, чтобы каждая ячейка сети была заполнена, а к математическим выражениям это неприменимо, поэтому ис-

пользуются слегка модифицированные 2D-грамматики, не требующие подобного расположения символов. Также используется вероятностный подход — стохастические грамматики, [28] описывает метод, использующий контекстно-стохастические грамматики, основанный на CYK-алгоритме. [29] представляет подход, основанный на контекстно-стохастических грамматиках и нечетких множествах. Эта работа интересна тем, что подробно описывает введенный для описания выражений формализм и грамматику, однако авторы указывают, что некоторая часть математической нотации не поддерживается их системой.

2.2.2. Предложенный метод

Предложенный метод — некоторая комбинация подходов описанных выше, в целом идея состоит в том, чтобы использовать как пространственную информацию о символе и его соседях, так и семантические отношения между ними, но при этом не ограничивать их строгими правилами.

Сначала стоит ввести некоторые обозначения и пояснения. В программной реализации выражение представляется в виде дерева, корнем которого является само выражение целиком. Каждый узел дерева представлен классом `Node`, который содержит информацию о данной части выражения — координаты ограничивающего прямоугольника, тип выражения и составляющие его элементы . Элементы могут быть двух типов — линейные, если данный элемент разделим в вертикальной проекции и зависимые (рис. 11) в ином случае, так же в случае корня — вложенные.

Введем следующие типы узлов:

1. Одиночный символ — содержит одну компоненту связанности (`SINGLE`).
2. Сегмент, содержащий несколько компонент связанности, но не имеющий точек разрыва в горизонтальной проекции (`MULTI`).

3. Сегмент, содержащий несколько компонент связанности, имеющий одну или более точек разрыва — несколько линий (`MULTILINE`).
4. Сегмент, содержащий несколько линий, одна из который предположительно является знаком деления (`FRACTION`).
5. Сегмент, содержащий две линии, одна из которых является акцентом (`ACCENTED`).

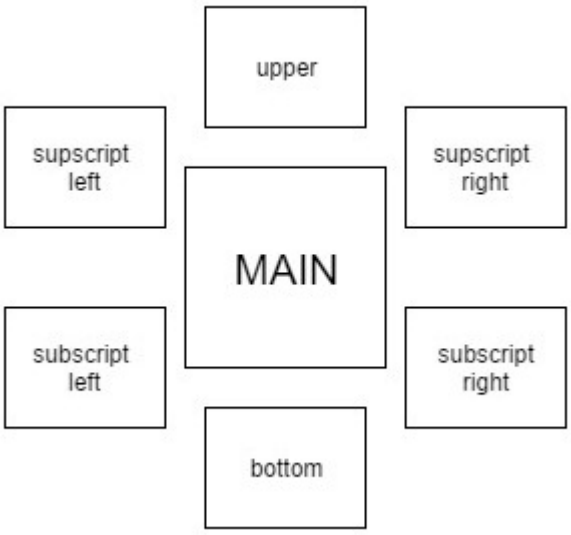

Рис. 11. Схема сегмента

Таблица 1 также представляет отчасти условное разбиение поддерживаемых символов на классы в зависимости от их написания.

Таблица 1. Типы символов

Ascender	A-Z, f, k, h, l, t, 0-9, $\sum$, $\int$, (,), {, }, [,], |, $\cup$, $\cap$, ∂, λ, δ
Descender	g, j, p, q, y, γ, β
Normal	a-z (except: asc., desc.), +, -, $\leftarrow$, $\rightarrow$, $>$, $<$, π, ∞, ω, $\neq$, $\in$, σ, ε, α

Алгоритм является рекурсивным. Парсер принимает на вход изображение и определяет число компонент связанности (это делается с использова-

нием функции библиотеки OpenCV), если найдена одна — выход из функции, выражение считается сегментированным, иначе — находятся точки разрыва в вертикальной проекции, для каждого фрагмента определяется тип, границы между ними уточняются, полученные сегменты сохраняются как линейные элементы текущего выражения или, если их количество равняется одному, разделяются на вложенные. Далее для каждого из полученных линейных/вложенных вызывается парсер, пока каждый элемент не будет представлен одной компонентой связанности.

Тип фрагмента очевидно определяется из числа и расположения компонент связанности его составляющих. Определение фрагмента как `SINGLE` и `MULTI` понятно из их определения, для `MULTILINE` и `FRACTION` точками разрыва в горизонтальной проекции находятся все составляющие их линии. Если их две и одна из них является акцентом — тип фрагмента отмечается как `ACCENTED`, иначе линии анализируются, если линия содержит одну компоненту связанности, символ распознается и, если он является горизонтальной чертой, а линия имеет соседние сверху и снизу — он помечается как знак дроби и запоминается его ширина. После анализа всех линий, если найден хотя бы один знак дроби — тип фрагмента изменяется на `FRACTION`, находится основной знак дроби — если он не один, берется наибольший по ширине, если не один с равной шириной — берется $\lceil \frac{n}{2} \rceil$ по счету, где n — число найденных знаков, фрагмент разбивается на верхнюю часть и нижнюю относительно определенного знака.

Затем все полученные фрагменты обрабатываются в зависимости от типа:

1. `SINGLE` — определяется тип символа.
2. `MULTI` — обычно этот тип представляет либо корень с подкоренным выражением или символы, написанные курсивом, и поэтому нераздели-

мые точками в вертикальной проекции. За стартовый символ считается крайний левый, все найденные ограничивающие прямоугольники сортируются по x-координате левой вершины прямоугольника в порядке возрастания. Будем рассматривать это как очередь и постепенно убирать из неё элементы. Изначально стартовый символ считается текущим. Например, обозначим текущий символ за n. Тогда, если n — знак корня, все символы, границы, которых входят в ограничивающий прямоугольник n, отмечаются как вложенные и убираются из очереди, и если очередь не пуста — берем следующий символ. Иначе — берем следующий символ m из очереди, если m является индексом n — координаты прямоугольника m расширяют соответствующие координаты вложенной части n, иначе — m считается текущим, n добавляется в массив линейных частей, составляющих рассматриваемый сегмент. В конце последний текущий сегмент, добавляется в массив линейных.

3. `MULTILINE` — если тип уже отмечен как `ACCENTED` — фрагмент считается обработанным. Иначе, если одна из линий, проанализированных на прошлом шаге, является либо «$\sum$», либо «$\int$» — эта линия отмечается, как главный элемент фрагмента, а остальные относительно её прямоугольника делятся на нижнюю и верхнюю части. В ином случае осуществляется ещё один проход по массиву найденных линий, и линии, представленные символом акцента, или высота, которых меньше порогового значения объединяются с ближайшей. Далее ограничивающий прямоугольник делится пополам и линии, оказавшиеся в верхней части соответственно становятся верхней частью фрагмента, нижние - нижней, а если линия пересекает середину, то она относится к главной.
4. `FRACTION` — было указано выше, на данном этапе это вся обработка.

Уточнение границ — функция, анализирующая должны ли два фраг-

мента быть связаны, она рассматривает два соседних сегмента и при необходимости соединяет их. Например, для двух фрагментов типа SINGLE определяется принадлежат ли они одной базовой линии или нет, в случае необходимости определяются верхние и нижние индексы. Здесь стоит отметить, что после анализа предыдущих работ было замечено, что во многих случаях ошибки связаны с тем, что иногда индексы либо слишком крупные, либо символы принадлежат разным типам (Таблица 1) и унифицировать критерий, определяющий является ли для данного символа соседний индексом, трудно. Для улучшения результатов для разных пар типов определяется их взаимное расположение по несколько разным критериям, подобранным экспериментальным путем. Для остальных классов рассматриваются пространственные отношения между их частями, например: для SINGLE и MULTI рассматривается отношение SINGLE к первому из линейных элементов MULTI, для SINGLE и MULTILINE всё зависит от типа второго фрагмента — в случае ACCENTED они рассматриваются как два SINGLE, если MULTILINE имеет главную компоненту — рассматривается отношение с ней, в ином случае рассматривается является ли SINGLE доминантным над ними или, наоборот, принадлежит одной из линий.

Затем для всех вложенных частей линейных компонент данного фрагмента снова вызывается парсер. Таким образом в итоге каждое выражение представлено в виде дерева.

Глава 3. Распознавание символов и текста

Глава описывает нейронную сеть, реализованную для распознавания символов, и некоторую коррекцию данных полученных на предыдущих шагах.

3.1. Нейронная сеть

Для распознавания символов была реализована сверточная нейронная сеть [30], архитектура представлена в Таблицах 2 и 3.

Таблица 2. Архитектура сети, часть 1

Слой	Число ядер(для слоя свертки)	Размер
Сверточный	6	5
Субдескритизации		2
Сверточный	19	5
Субдескритизации		2
Сверточный	200	5

Таблица 3. Архитектура сети, часть 2

Слой	Число нейронов	Функция активации
Полносвязный	200	ReLU
Полносвязный	150	Tanh
Полносвязный	104	Softmax

В качестве метода оптимизации был использован `Adam` [31], основанный на вычислении моментов, для регуляризации и борьбы с переобучени-

ем использовалась l_2-регуляризация весов. Сеть была обучена на данных, описанных в Главе 5, для тренировки выделено 90% датасета, для тестирования — 10%, результат составил 98% на тестовой выборке.

3.2. Касающиеся символы

Одна из проблем распознавания текста — касающиеся символы, они неразделимы в вертикальной проекции и очевидно, что система, обученная только на отдельных символах, никогда не даст правильный результат. В случае изображения рукописного текста или печатного с очень низким качества — это проблема весьма весомая и для её решения применяются различные подходы, строящие деревья возможных вариантов сегментации, выбирая оптимальный вариант.

Однако, так как на данном этапе разрабатываемая система предполагает изображение текста в достаточно хорошем качестве, был рассмотрен более простой случай, который тем не менее часто встречается в печатных документах даже хорошего качества и сильно портит конечный результат. Это соединения букв типа «li», «lm», «in».

В [32] предложен метод, который, применяя к столбцам бинаризованного изображения логическую операцию «И», ищет в полученном векторе точки минимума. Они считаются точками-кандидатами разбиения изображения, строится дерево возможных вариантов и выбирается наиболее вероятный вариант. Однако применять к каждой сегментированной части изображения алгоритм, во-первых, весьма неоптимально, во-вторых, может вызвать новые ошибки, разделив правильно распознанный символ, не состоящий из касающихся. В оригинальной статье кандидаты в касающиеся символы выбираются на основе сопоставления профилей сторон, что в данном случае также достаточно затратно в вычислительном плане,

учитывая большое количество классов и символов в распознаваемом документе. В связи с этим была предложена следующая эвристика. Для всех символов из коллекции, на которой обучалась нейронная сеть, была рассчитано среднее отношение высоты к ширине, далее оно вычисляется для распознанного символа и, если результат отличается от эталонного более чем на 50%, к символу применяется описанный алгоритм. К построенному алгоритмом дереву применяется аналогичная логика — оценивается разница между эталоном и полученным результатом для каждого узла дерева и выбирается поддерево, имеющее наименьшее отклонение.

Также был рассмотрен метод с использованием Hu-moments [33], однако он давал много ошибок классифицируя корректный символ, как касающийся.

3.3. Обработка полученного результата

Полученные выражения обрабатываются в зависимости от их типа.

Для предположительно математических выражений — представлены деревом, которое имеет глубину 2 и более, необходимо решить две основные задачи: объединить символы из нескольких компонент связанности («=», «$\leqslant$», «$\geqslant$» и т. д.), найти слова, являющиеся названиями функций («lim», «sin» и т. д.). Для решения первой задачи рассматриваются узлы дерева, верхняя, нижняя и центральная части которых представлены одной компонентой связанности или отсутствуют. Для решения второй рассматриваются линейные части узлов, которые не имеют дочерних частей.

Текстовую часть необходимо разбить на слова и inline-формулы, дерево которых имеет глубину, равную 1. Так как интерес в данной работе в основном представляло выделение формул и их распознавание, то для разделения текста на слова был выбран простой алгоритм, определяющий в

одном ли слове два соседних символа по расстоянию между ними. Для этого необходимо получить некие средние значения пробелов во всем тексте. Если построить гистограмму частот по значениям пробела между парами соседних символов и взять два её пика, то один из них соответствующий меньшему расстоянию будет, вероятнее всего, соответствовать приблизительно межбуквенному расстоянию, а другой — расстоянию между словами. Таким образом такая гистограмма строится по полученным выражениям, учитывая только те, глубина дерева которых равна 1. Соответственно затем выражения разделяются, встречая либо расстояние больше порогового, либо нетекстовый символ. Далее слова ищутся в словаре [34], и если такое слово не найдено, то оно меняется на ближайшее по расстоянию Левенштейна. Буквы, принадлежащие одному слову, соединяются в одно выражение, которое также содержит слово, которое они составляют.

Так же на данном этапе выражения классифицируются как `MATH` или `TEXT` для последующей обработки.

Глава 4. Формирование результата

Данная глава описывает обработку блоков и выражений и формирование на их основе строки LaTeX.

4.1. Формирование макета страницы

На этом этапе блоки анализируются и объединяются, образуя колонки.

Все блоки отсортированы по координатам верхней левой точки их ограничивающего прямоугольника. Блоки по очереди сравниваются и, если, встречаются два, которые перекрываются по оси y, формируются колонки относительно этих блоков. Далее, если сформирована хотя бы одна пара колонок — для каждой, в её рамках, происходит аналогичная процедура, пока вся страница не будет размечена.

4.2. Обработка выражений

В зависимости от типа выражения их значение формирует строку. Для выражений типа `TEXT` — слово, представляющее выражение добавляется в строку, для выражений типа `MATH` алгоритм рекурсивно проходит по всем вложенным частям выражения, формируя выход в соответствии с правилами LaTeX— добавляя верхние/нижние индексы, подстрочные/надстрочные символы, знак дроби, если часть выражения отмечено, как дробь. На рисунке 12 показан промежуточный шаг в виде блок-схемы.

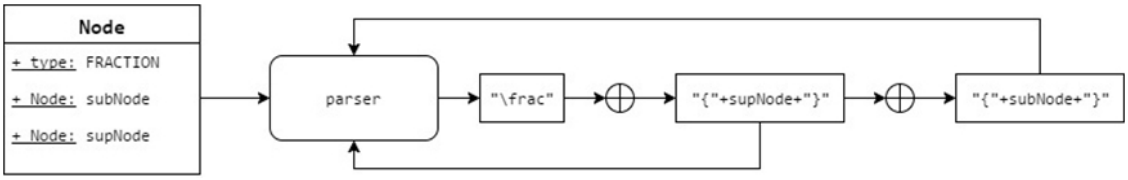

Рис. 12. Промежуточный шаг

Глава 5. Тестирование, оценка результатов, интерфейс

Глава описывает проблемы метрики и оценки полученных результатов и демонстрирует предложенный интерфейс.

5.1. Тестовые данные

В качестве тестовых данных и данных для обучения были использованы Infty-датасеты [35]. Один содержит 500 страниц из математических журналов и информацию для извлечения символов — они были использованы для обучения нейронной сети. Другой содержит примерно 4000 изображений отдельных изображений математических формул — эти данные были использованы для подбора параметров и тестирования модуля, отвечающего за сегментацию выражений, представленного в Главе 2.

5.2. Оценка результата

Оценка результата для данной системы — нетривиальная задача. Оценить даже работу модуля, отвечающего за распознавание выражений достаточно трудно — не существует четко введенной метрики. Методы оценки в предыдущих работах обычно основаны на реализации и применимы только для предложенной работы, поэтому результаты исследований затруднительно сравнивать даже между собой. Соответственно по этой причине нет крупной базы, содержащей допустим изображения выражений и какого-либо экспертного представления результата. Кроме того, даже если говорить о проблеме в ключе данного исследования, представляющего результаты в виде строки LaTeX, сравнение двух строк также не очевидно,

так как одно и тоже выражение может быть записано по-разному.

В [36] рассматриваются проблемы, связанные с оценкой результатов систем распознавания математических выражений, предлагая ввести 3 меры — процент правильной сегментации, процент распознанных символов и процент соответствия полученного выражения входному. Однако предложенный алгоритм оценки и сопоставления также не универсален, так как для этого используется представление в формате MathML.

Ввиду описанных проблем, в рамках данной работы тестирование было организовано следующим образом. Из [35] было взято 150 произвольных изображений и представлено в виде строки LaTeX, далее результат вручную сравнивался со строкой, полученной алгоритмом. Результат представлен в Таблице 4.

Таблица 4. Результаты

Корректно распознанные выражения	70%
Ошибки распознавания	20%
Ошибки распознавания структуры	10%

5.3. Интерфейс и реализация

На данном этапе предложен простой web-интерфейс, позволяющий пользователю загрузить изображение формулы или текста, и получить распознанный результат. Есть окошко предварительного просмотра, которое отображает результат компиляции строки с помощью библиотеки `MathJax`, и поле, где отображается строка в формате LaTeX. Пользователь может сохранить и откорректировать результат, пример представлен на рис. 13.

Программная реализация выполнена на языке `java`, для реализации серверной части использован framework `Spring`, интерфейс - `javascript`, `AngularJs`.

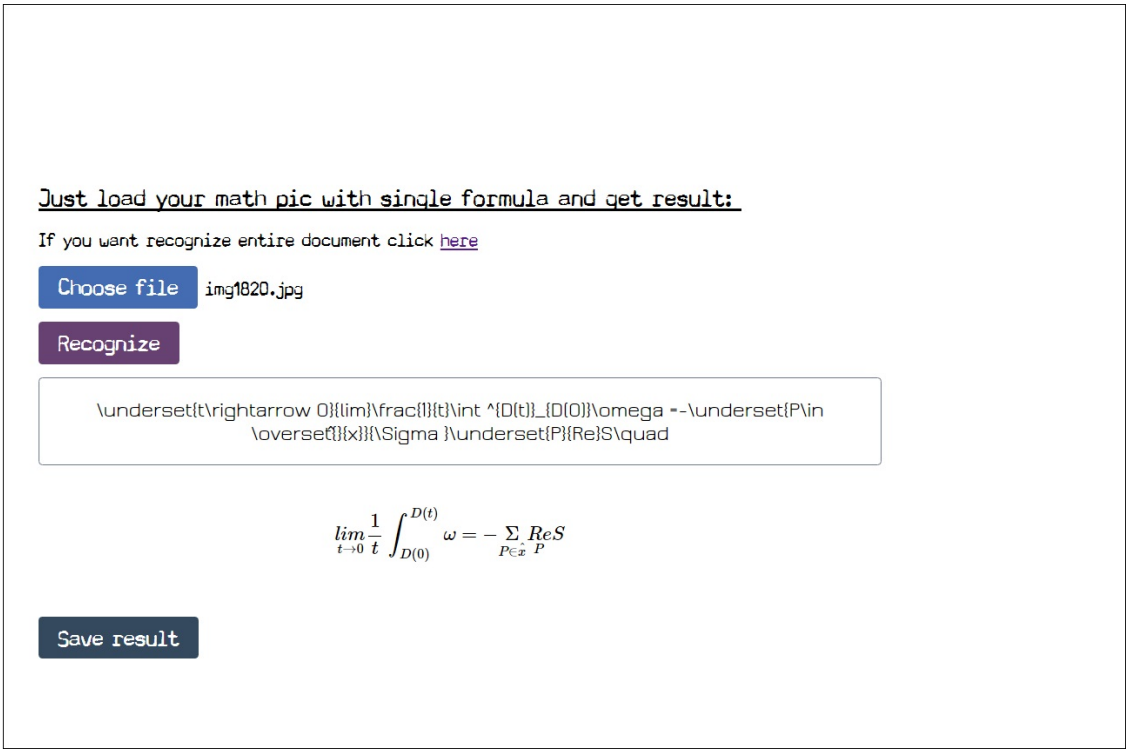

Рис. 13. Пример интерфейса

Заключение

В работе изучена проблема распознавания изображения математического документа. Рассмотрены этапы и проблемы предварительной обработки текстовых документов, выбраны и адаптированы наиболее подходящие методы. Реализована система для распознавания документа, в которую интегрирован разработанный модуль для распознавания математических выражений и восстановления их структуры. Так же предложен web-интерфейс, с помощью которого пользователь может загрузить изображение формулы или текста и получить результат, который можно откорректировать и сохранить.

Конечно, система нуждается в доработке и улучшении. Например, поддержка более сложных математических конструкций — на данном этапе система может распознавать системы, однако матрицы, которые могут иметь не стандартную структуру, содержать специальные символы — диагональные точки, крупные символы, занимающие несколько ячеек, на данный момент не поддерживаются. Кроме того, возможно, добавление большего числа правил и семантики фрагментам может улучшить результаты распознавания структуры.

Также, если говорить в целом о системе обработки изображения документа, то необходим модуль классифицирующий нетекстовые части изображения — картинки, графики, распознающий таблицы. Однако, это отдельная нетривиальная задача, не являющаяся целью данной работы.

Список литературы

1. Abby FineReader. https://www.abbyy.com/en-gb/
2. OmniPage. https://www.nuance.com/
3. Acrobat Capture, OCR, font, and page recognition. http://www.adobe.com/
4. Tesseract. https://github.com/tesseract-ocr/tesseract
5. InftyReader. http://www.inftyreader.org/
6. Farahmand A., Sarrafzadeh A., Shanbehzadeh J. Document image noises and removal methods // Proc. of the International MultiConference of Engineers and Computer Scientists, 2013. Vol 1.
7. Pok G., Jyh-Charn L. Decision based median filter improved by predictions // ICIP 99, 1999. Vol. 2. P. 410–413.
8. Aiswarya K., Jayaraj V., Ebenezer D. A new and efficient algorithm for the removal of high density salt-and-pepper noise in images and videos // Proc. of the 2nd International Conference on Computer Modeling and Simulation, 2010. P. 409–13.
9. Esakkirajan S., Veerakumar T., Subramanyam AN., Chand CHP. Removal of high density salt-andpepper noise through modified decision based unsymmetric trimmed median filter // IEEE Signal Proc. Lett, 2011. Vol. 18. P. 287–90.
10. Mostafavi M., Kazerouni I., Haddadnia J. Noise removal from printed text and handwriting images using coordinate logic filters // International Conference on Computer Applications and Industrial Electronics, 2010. P. 161–164.
11. O'Gorman L. Image and Document Processing Techniques for the Rightpages Electronic Library System // Proc 11 IAPR Int'l conf. Pattern Recognition, 1992. Vol. 2. P. 260–263.

12. Chinnasarn K., Rangsanseri Y., Thitimajshima P. Removing salt-and-pepper noise in text/graphics images // IEEE Asia-Pacific Conference on Circuits and Systems, 1998. P. 459–462.
13. Smith R. A Simple And Efficient Skew Detection Algorithm Via Text Row Accumulation // In Proc. of the 3th International Conference on Document Analysis and Recognition, 1995. P. 1145–1148.
14. Shivakumara P., Hemantha G., Guru D.S., Nagabhushan P. A New Boundary Growing And Hough Transform Based Approach For Accurate Skew Detection In Binary Document Images // Proc. of ICISIP, 2005.
15. Yan H. Skew Correction Of Document Images Using Interline Crosscorrelation // CVGIP Graphic Models Image Processing, 1993. Vol. 55. P. 538–543.
16. Shafait F., Keysers D., Thomas M. Breuel. Performance comparison of six algorithms for page segmentation.
17. Nagy G., Seth S., Viswanathan M. A prototype document image analysis system for technical journals // Computer 7, 1992. P. 10–22.
18. Wong K. Y., Casey R. G., Wahl. F. M. Document analysis system // IBM Journal of Research and Development 26, 1982. P. 647–656.
19. Baird, H.S. Background structure in document images // In: Document Image Analysis, World Scientific, 1994. P. 17–34.
20. K. Jain A., Bhattacharjee S. Text segmentation using Gabor filters for automatic document processing // Machine Vision and Applications, 1992. P. 169–184.
21. Kise K., Sato A., Iwata M. Segmentation of page images using the area Voronoi diagram // CVIU 70, 1998. P. 370–382.
22. O'Gorman L. The Document Spectrum for Page Layout Analysis // IEEE Trans. on Pattern Analysis and Machine Intelligence, 1993. Vol. 15, No 11. P. 1162–1173.

23. Agrawal M., Doermann D. Voronoi++: A Dynamic page segmentation approach based on Voronoi and Docstrum features // 10th International Conference on Document Analysis and Recognition, 2009. P. 1101–1115.
24. Chan K., Yeung D. Mathematical expression recognition: a survey // IJDAR, 2000. Vol. 3. P. 3–15.
25. Lavirotte S., Pottier L.. Mathematical formula recognition using graph grammar // Electronic Imaging, 1998.
26. Zanibbi R., Blostein D. Recognizing mathematical expressions using tree transformation // IEEE Trans. on Patter Analysis and Machine Intelligence, 2002. Vol. 24, No. 11.
27. Tomita M. // Parsing 2-dimensional languages. Current Issues in Parsing Technology, 1991. P. 277–289.
28. Álvaro F., Sánchez J, Benedí J. Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars // Proc. of the International Conferenceon Document Analysis and Recognition, 2011. P. 1225–1229.
29. MacLean S., Labahn G. A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets // IJDAR, 2012.
30. LeCun Y., Bottou L., Bengio Y., Haffner P. GradientBased Learning Applied to Document Recognition // PROC OF THE IEEE. 1998.
31. Kingma D., Ba J. ADAM: A method for stohastic optimisation // ICLR. 2015.
32. Min-Chul J., Yong-Chul S., Sargur N. Machine printed character segmetation method using side profiles.
33. HU M. Visual pattern recognition by moment invariants // IRE Trans. on Information Theory. 1962.
34. English dictionary. https://github.com/dwyl/english-words
35. Nomura A., Uchida S., Suzuki M. // A Ground-Truthed Mathematical

Character and Symbol Image Database.

36. Awal A., Mouchère H., Viard-Gaudin C. The problem of handwritten mathematical expression recognition evaluation // 12th International Conference on Frontiers in Handwriting Recognition, 2010.