Федеральное государственное бюджетное учреждение науки Институт системного программирования им. В.П. Иванникова Российской академии наук

на правах рукописи

Беляева Оксана Владимировна

Автоматическое восстановление структуры текстовых документов

Специальность 2.3.5 —
«Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»

диссертации на соискание учёной степени
кандидата технических наук

Научный руководитель:
кандидат физ.-мат. наук
Турдаков Денис Юрьевич

Москва — 2026

Оглавление

Введение

Актуальность темы. В условиях стремительного роста объема электронных документов, создаваемых в различных сферах деятельности, возникает острая потребность в их автоматической обработке с целью экономии человеческих ресурсов. Большинство документов представлены в неструктурированном виде, что требует применения интеллектуальных методов обработки документов для их структуризации.

Автоматический анализ информации из социальных сетей, интернета, сайтов, структурирование как открытых, так и закрытых баз знаний невозможно выполнить качественно без автоматического извлечения содержимого и структуры электронных текстовых документов в данных источниках.

Под анализом информации в данной работе понимается извлечение фрагментов текстовой и графической информации с последующей её структуризацией для целей хранения, организации поиска, вычисления статистических данных и обобщения результатов. В обработке электронных документов анализ информации сводится к анализу содержимого документов, а он в свою очередь невозможен без первоначального этапа - извлечения содержимого и восстановления структуры из документа.

Под структурой текстового документа понимается иерархическая структура, то есть иерархическое представление совокупности частей документа таким образом, чтобы его части располагались на соответствующих уровнях иерархии и была обеспечена возможность навигации с использованием разбивки документа на главы, секции, разделы и тому подобное.

Наличие информации о содержимом и структуре электронных документов облегчает их цифровую обработку. В первую очередь это требуется для информационно-аналитических систем, обеспечивающих сбор и поиск информации с последующей интеллектуальной обработкой содержимого документов. Знания об иерархической структуре документа способствуют решению таких задач, как обеспечение навигации по документу (восстановление оглавления), суммаризации (составление краткого описания к тексту) и фрагментации документа, проверки правильности составления документа, поиск по заголовкам, поиск связей внутри одного или нескольких документов.

Автоматическая обработка электронных текстовых документов является трудной задачей, поскольку документы могут быть представлены в различных форматах, таких как PDF, DOCX, HTML, изображений, а их структура и виды фрагментов содержимого могут существенно различаться в разных предметных областях и принятых там типах документов (например, технические задания, законы, рекламные брошюры или исследовательские работы). Поэтому для качественного анализа информации необходимо учитывать особенности предметной области документа и его формат, задающий спецификацию хранения текстовой, графической, табличной информации и работы с ней.

К особенностям предметной области документа (свойствам типов документа) относят совокупность правил составления документа: правила составления структуры содержимого документа, правила визуального оформления (форматирования) содержимого, тематика текстового содержимого. Качество разработанных методов в диссертационной работе демонстрируется для трех разных типов документов “Техническое задание”, “Выпускная квалификационная работа” и “Нормативно-правовой акт”, но спектр применимости методов гораздо шире и не ограничивается только данными типами.

Среди широкого набора разнообразных форматов электронных документов можно выделить две основные группы. Документы могут быть представлены форматами, такими как PDF с текстовым слоем, HTML, DOCX, и т.д. Такие форматы являются ***структурированными***, то есть в них содержатся структурные теги, позволяющие выделить в документах заголовки разного уровня, списки, таблицы, данные о форматировании. При этом в каждом формате внутренняя разметка (теги) и их виды определены по-разному.

Кроме того, существуют форматы ***неструктурированных*** данных, например, изображения или PDF-документы, содержащие страницы, являющиеся сканированными копиями напечатанных на бумаге или написанных от руки документов. Такие документы легко воспринимаются человеком, но плохо поддаются автоматическому анализу, поскольку не содержат ни текстовой (копируемый текст), ни структурной (встроенные в формат теги/разметка) информации о содержимом документа.

Область автоматического извлечения содержимого и восстановления структуры документов различных форматов, в частности неструктурированных форматов изображений и PDF остается по сей день вызовом для систем

автоматического интеллектуального анализа текстовых электронных документов.

Объектом исследования являются текстовые электронные документы различных предметных областей в виде структурированных и неструктурированных форматов документов.

Предметом исследования выступают методы автоматического извлечения содержимого и восстановления структуры из исследуемых текстовых документов.

Степень разработанности темы. Исследования в области обработки электронных документов неструктурированных форматов активно ведутся уже более двадцати лет. Важные результаты были получены в работах Mao S., Namboodiri A., где авторы отметили важность проблемы извлечения содержимого и восстановления структуры документов. В последнее время чаще появляются работы для обработки изображений сканированных документов с использованием нейронных сетей для разного рода задач, например для сегментирования страницы документа (Binmakhashen G, Eskenazi S.), табличной обработки (Schreiber S., Zhong X., Gao L.).

Проводятся исследования (Михайлов А.) в области автоматической обработки некорректных PDF-документов, а также исследуются причины, приводящие к нарушению корректности. Некоторые работы авторов, такие как Gonesh C, James H исследуют вопросы около данной темы, например классификация текстов на корректность.

Множество последних исследований в области восстановления иерархической структуры документов формата PDF стало возможным благодаря международному соревнованию FINTOC по обработке финансовых документов, где участники применяют современные методы и подходы на основе машинного обучения.

Целью исследования является разработка методов и расширяемого программного средства для автоматического извлечения содержимого и восстановления структуры из электронных текстовых документов. Разрабатываемые методы и программные средства должны удовлетворять следующим требованиям:

1. Обеспечение автоматического анализа информации для текстовых документов технических заданий, нормативно-правовых актов, выпускных квалификационных работ;
2. Возможность расширения средств извлечения содержимого и восстановления структуры для новых форматов документов и документов новых предметных областей.

Для достижения поставленной цели решаются следующие **задачи**:

1. Разработать метод автоматического извлечения содержимого PDF документов с использованием проверки текстового слоя, обеспечивающий достоверность извлечения и скорость обработки документов;
2. Разработать метод автоматического восстановления структуры из содержимого текстовых документов;
3. Реализовать предложенные методы в виде программного комплекса, обладающего возможностью расширения новыми форматами и типами текстовых документов.

Научная новизна заключается в следующих результатах работы:

1. Метод автоматического извлечения содержимого PDF документов с использованием проверки текстового слоя, обеспечивающий достоверность извлечения и скорость обработки документов на русском и английском языках;
2. Метод автоматического восстановления иерархической структуры из содержимого документов. Метод показывает более высокое качество восстановления структуры, по сравнению с другими методами на наборе данных соревнования FINTOC2022;

Теоретическая и практическая значимость. Теоретическая значимость диссертации заключается в разработке и усовершенствовании методов извлечения содержимого и восстановления структуры документов неструктурированных форматов в автоматическом режиме. В рамках диссертации разработан метод, позволяющий с большей точностью восстанавливать иерархическую структуру, что подтверждено измерениями на наборе данных FINTOC2022. Важными результатами диссертации являются новые методы автоматической обработки документов в формате PDF. Увеличивает теоретическую ценность работы рассмотрение и использование нейросетевых методов в построенной обработке изображений сканированных документов, что позволяет достигать высокого качества извлечения текстовой информации.

В плане практической значимости важным результатом является открытый программный комплекс для автоматического извлечения содержимого и восстановления иерархической структуры текстовых электронных документов различных форматов и предметных областей, который может быть использован в качестве первоначального этапа для систем автоматической интеллектуальной обработки электронных документов. Внедрения и ПО с открытым доступом:

- интеграция в открытую библиотеку LangChain[1];
- внедрение в платформу Талисман, предназначенную для построения интеллектуальных информационно-аналитических систем сбора и обработки данных;
- внедрение в систему анализа выпускных квалифицированных работ.

Результаты диссертации применимы для разработчиков информационно-аналитических систем, предназначенных для структуризации и анализа сырых необработанных данных, в том числе электронных документов.

Методология и методы исследования. В диссертационной работе применялись методы обработки изображений, машинного обучения, теории вероятностей и оптимизации. Основные методы исследования включают анализ существующих решений, разработку и экспериментальное исследование алгоритмов.

Основные положения выносимые на защиту:

1. Метод автоматического извлечения содержимого PDF документов с использованием проверки текстового слоя, обеспечивающий достоверность извлечения и скорость обработки документов;
2. Метод автоматического восстановления иерархической структуры из содержимого текстовых документов;
3. Архитектура и реализация расширяемого программного комплекса в виде открытой библиотеки DEDOC[2] для автоматического извлечения содержимого и восстановления иерархической структуры из электронных текстовых документов структурированных и неструктурированных форматов.

Апробация работы. Результаты работы докладывались на конференциях, форумах:

1. IVMEM2019 Международная конференция ”Иванниковские чтения 2019”, Великий Новгород, 2019, РФ;
2. FNP 2021 The 3rd Financial Narrative Processing Workshop, 2021, Marseille, France; LREC;
3. IVMEM2022 Международная конференция ”Иванниковские чтения 2022”, 2022, Казань, РФ;
4. ISPRAS OPEN 2022 Открытая конференция ИСП РАН им. В.П. Иванникова, Москва, РФ;
5. AINL: Artificial Intelligence and Natural Language Conference, 2023, Ереван, РА;

[1] https://github.com/langchain-ai/langchain/releases/tag/langchain-community%3D%3D0.2.10
[2] https://github.com/ispras/dedoc

6. ISPRAS OPEN 2023 Открытая конференция ИСП РАН им. В.П. Иванникова, 2023, Москва, РФ;
7. IVMEM2024 Международная конференция ”Иванниковские чтения 2024”, 2024, Великий Новгород, РФ;
8. DataFest 2024, в гостях у VK, 2024, Москва, РФ;
9. Гравитация. Международная университетская премия в области искусственного интеллекта и больших данных, 2024, Москва, РФ.

Публикации и личный вклад автора. Автор имеет 10 научных публикаций по теме диссертации. Работы [8, 9, 10] индексируются в Scopus и Web of science. Основные результаты по теме диссертации изложены в 8 печатных изданиях, 5 из которых [5-8, 10] изданы в журналах, рекомендованных ВАК. Остальные 5 работ опубликованы по результатам конференций. В работах [1-8] автором проведено исследование предметной области, выполнен основной объем теоретических и экспериментальных исследований. В работах [2, 3, 6] Беляевой О.В. и Козлову И. принадлежит постановка задачи, разработка подхода и анализ экспериментов. Работы [1, 4, 5, 7, 9, 10] выполнены под непосредственным руководством Беляевой О.В. В работе [5] автором разработан подход и метод исправления ориентации, разработка экспериментов и анализ результатов проводилась совместно с соавторами. Работа [7] выполнена полностью автором, редакторские правки и анализ результатов выполнялись совместно соавторами. По теме диссертации имеется 3 свидетельства о государственной регистрации программы для ЭВМ [11-12].

Предлагаемые в диссертации инструменты, текстовые наборы данных и исследования разработаны и выполнены автором или при его непосредственном участии.

Внедрение результатов. Результаты, полученные в рамках данной работы, внедрены в следующих организациях (Приложение Б):

1. Внедрены в систему “Киберпрофессор” анализа выпускных квалификационных работ студентов (акт о внедрении № 20/01-6 от 06.02.2025);
2. Внедрены в состав платформы Талисман, которая используется в ООО “Интерпроком” (акт о внедрении № 18/25 от 28.01.2025);
3. Внедрены в состав платформы Талисман, которая используется в федеральном государственном автономном образовательном учреждении высшего

образования «Московский государственный институт международных отношений (университет) Министерства иностранных дел Российской Федерации» (акт о внедрении от 7.02.2025);

4. Внедрены в сервис распознавания изображений документов в ЗАО “ЕС Лизинг”, что подтверждает официальное письмо № ЕСЛ-36 от 10.02.2025).

Глава 1. Обзор работ

Первая глава посвящена обзору области автоматического извлечения содержимого и восстановления структуры из различных типов и форматов документов. Особое внимание уделяется существующим методам и системам для автоматического восстановления структуры текстовых документов, а также автоматической обработке изображений сканированных документов. В данном разделе также рассматриваются основные проблемы, возникающие при обработке PDF-документов в контексте автоматического анализа, а также представлено подробное исследование причин, влияющих на извлечение некорректного текста из копируемых PDF-документов.

В общей схеме автоматической обработки документов разных форматов выделяют два основных этапа:

1. Автоматическое извлечение текстового содержимого с форматированием согласно спецификации формата документа. На данном этапе используются решения/библиотеки, учитывающие особенности структуры формата обрабатываемого документа. Например для обработки документа в формате HTML можно использовать библиотеку beautifulsoup. Более трудным процессом в автоматической обработке - является обработка документов слабоструктурированных и неструктурированных форматов (изображений и PDF). Обработке таких документов посвящено немалое количество научных работ, рассмотренных в данной главе в разделе 1.2.
2. Автоматическое восстановление иерархической структуры документа на основе его содержимого с форматированием. На основании полученного содержимого с форматированием на данном этапе восстанавливается иерархическая структура документа, учитывающая особенности его предметной области (тип документа). Здесь учитываются правила, согласно которым данный тип документа создавался. Например, документ содержит оглавление, определенное количество глав, титульный лист и тому подобное. Обзор работ методов восстановления структуры представлен в разделе 1.3.

Общая схема обработки разноформатных документов подробно описана во второй главе работы. В первой главе будут рассмотрены основные направления

автоматической обработки документов с целью восстановления иерархической структуры, Первая глава состоит из четырех основных разделов.

В разделе 1.1 рассмотрена исследуемая в диссертации предметная область документов, состоящая из таких типов документов “Техническое задание” (ТЗ), “Нормативно-правовой акт” (НПА), “Выпускная квалификационная работа” (ВКР). Также выделены особенности исследуемой предметной области изображений сканированных документов.

В разделе 1.2 работы приведен обзор существующих методов и систем извлечения содержимого из текстовых документов различных форматов. Исследуется извлечение содержимого в виде текста, табличной информации, стилевого форматирования текста. В разделе также рассматриваются такие проблемы, как *автоматическая работа с PDF* документами с некорректным текстовым слоем, а также PDF документами не содержащими текстовый слой.

В разделе 1.3 представлен обзор методов автоматического восстановления иерархической структуры из полученного содержимого со стилевым форматированием с предыдущего этапа. В данном разделе рассматривается проблема разнообразия предметных областей (типов) электронных текстовых документов. Здесь рассмотрены различные научные работы, использующие как классические эвристические подходы, так и методы машинного обучения.

В разделе 1.4 описаны существующие системы автоматической обработки разных электронных текстовых документов и их возможности.

1.1 Предметная область

1.1.1 Топология форматов и типов электронных документов

Разнообразие электронных документов заключается в:

1) Разнообразии форматов файлов документов;
2) Разнообразии вида структуры документа (или предметной области документа, или типа документа).

Форматы файлов электронных документов разделяют на:

- **Структурированные форматы,** которые содержат в себе специфичную для каждого формата файла разметку, представляющую собой набор инструкций (или тегов), благодаря которым можно извлечь содержимое документа без использования дополнительных эвристических правил или машинного обучения. Данные теги позволяют структурировать содержимое документа, определять местоположение и тип каждой части в пределах документа. В инструкциях хранится информация о тексте, таблицах, рисунках и иных объектах с дополнительной информацией об их форматировании (т.е. визуальном представлении). За счет структурированного хранения информации, для автоматического извлечения содержимого из документа достаточно обрабатывать структуру формата согласно его спецификации. К структурированным документам относят такие форматы, как DOC/DOCX, HTML, CSV, PPTX и т.д.
- **Неструктурированные форматы** в отличие от структурированных не содержат внутри себя разметку (или содержат неполную разметку), что кардинально затрудняет их автоматическую обработку. В качестве таких документов выступают сканированные документы в форматах изображений и PDF.
- В отдельную категорию к **слабоструктурированному формату** относят формат PDF, полученный путем конвертации из структурированных форматов (например, DOCX). Такие PDF хранят текстовую информацию и разметку, что позволяет быстрее их обрабатывать, в отличие от неструктурированных форматов (например, изображений). Такие PDF могут содержать некорректный текстовый слой, что требует дополнительной проверки при автоматической обработке PDF-документов.

Отсюда неструктурированные и слабоструктурированные форматы документов трудны в автоматическом анализе и требуют разработки отдельных методов.

Несмотря на вариативность форматов документов их структура может быть составлена по разным правилам, регламентированным, в том числе, по ГОСТам. Например, техническое задание по структуре отличается от законов. Иными словами, тип (или предметная область) документа – это набор правил, задающий определенный шаблон структуры документа.

1.1.2 Предметная область изображений сканированных документов

Существует много направлений обработки изображений текстовых документов [14]. Как правило, современные методы основанные на глубоких нейронных сетях требуют больших объемов данных для обучения. Отступление от предметной области, на которой обучалась сеть или были созданы эвристические правила может приводить к серьезному ухудшению качества [7]. Поэтому для такого рода задач необходимо четко обозначить предметную область данных.

В работе рассматриваются распространенные электронные документы, представляющие собой изображения сканированных черно-белых документов. Документы характеризуются манхэттенским шаблоном (текстовые блоки расположены параллельно друг к другу), печатным текстом на русском и английском языках, без артефактов (искажений) на изображении, таких как засветы, затемнения, размытость. Страницы документа могут иметь разную ориентацию 0, 90, 180, 270 градусов. Таблицы в документах имеют явные границы.

Вышеперечисленным критериям предметной области соответствуют следующие примеры типов документов:

- технические задания (ТЗ);
- законы (нормативно-правовые акты (НПА));
- выпускные квалификационные работы и научно исследовательские работы (ВКР);
- отчетная документация.

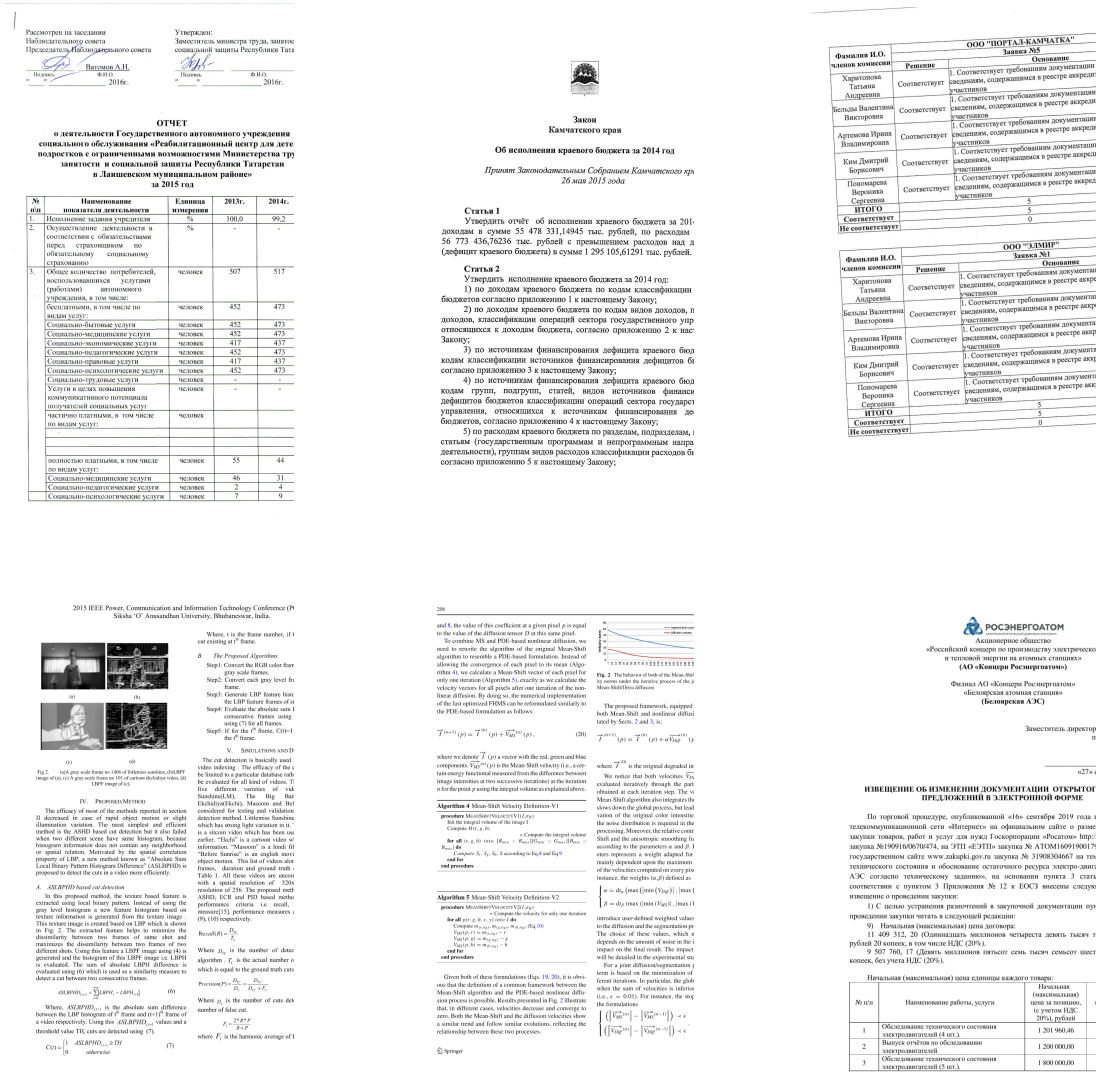


Рисунок 1.1.2.1 – Примеры обрабатываемых изображений сканированных документов.

1.2 Методы извлечения содержимого из форматов PDF и изображений

В данном разделе будут рассмотрены подходы и методы извлечения содержимого из слабоструктурированных (PDF с текстовым слоем) и неструктурированных документов (PDF без текстового слоя и изображения сканированных документов).

Будут рассмотрены существующие методы предобработки сканированных документов и методы извлечения табличной информации из изображений сканированных документов.

В разделе 1.2.7.2 будут рассмотрены причины возникновения некорректного текстового слоя в PDF-документах, который приводит к низкому качеству извлекаемого текста при автоматической обработке.

1.2.1 Обработка изображений сканированных документов

В автоматической обработке сканированных документов есть проблемы, которые могут повлиять на качество извлечения содержимого:

1. *Плохое качество сканирования*: если сканирование документов произведено с низким разрешением, лист положили на сканер с неправильной ориентацией или имеются повреждения самого бумажного листа, то это может затруднить процесс распознавания и извлечения информации.
2. *Визуальное разнообразие страниц документов*: существует широкий спектр документов с различными шаблонами страниц, размером и шрифтами.
3. *Распознавание рукописного текста*: распознавание рукописного текста остается сложной задачей в автоматической обработке документов [10]. Различные стили и почерки могут затруднять процесс распознавания и требовать дополнительной обработки. В данной работе не рассматриваются документы, содержащие рукописный текст.
4. *Обработка сложных структурных элементов (графиков, таблиц)*: документы могут содержать сложные схемы, таблицы, графики и другие элементы. Автоматическая обработка должна быть способна корректно обрабатывать и интерпретировать такие элементы для точного извлечения информации. В работе уделено внимание только обработке табличной информации.

Обработка сканированных документов сводится к работе с пикселями изображения. В силу сложности автоматического анализа изображений документов, исследователи активно прибегают к методам машинного обучения и компьютерного зрения. Как правило, в процессе анализа изображений документов решаются следующие задачи:

- *Предобработка изображений документов* – повышение качества обрабатываемого изображения с целью улучшения результатов его распознавания. Частными случаями предобработки изображений являются локализация документов на изображении, устранение шумов (бинаризация) на изображении, повышение разрешения изображения, выравнивание и исправление ориентации страницы.
- *Анализ макета документа или сегментация документа (DLA - Document Layout Analysis, document segmentation)* – это задача анализа шаблона страницы документа. В рамках страницы документа определяется месторасположение и тип (класс) конкретных его частей. В частности, на изображении документа могут обнаруживаться текстовые блоки, изображения, таблицы, формулы и т.д.
- *Распознавание объектов, найденных в документе.* Например, одной из широко известных задач является задача распознавания таблиц, в которой анализируются ячейки таблицы, их структура и содержимое.
- *Извлечение текстовой информации.* Этот этап, как правило, является заключительным в процессе обработки документов. Здесь применяются методы оптического распознавания символов с изображений (OCR - Optical Character Recognition). В частности, одной из подзадач является задача распознавания рукописных символов (HCR - Handwriting Character Recognition).

Процесс обработки изображения страницы документа представлен на рисунке 1.2.1.1.

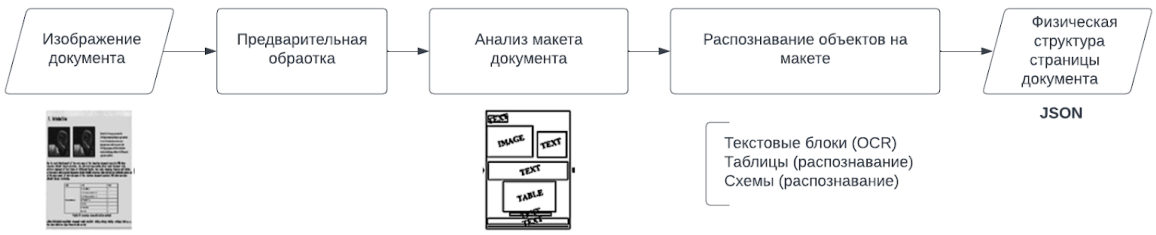


Рисунок 1.2.1.1 – Процесс извлечение содержимого из изображения документа.

1.2.2 Виды страниц документов

Предметная область документов задает не только ограничение на текстовое содержимое документа, но и на его визуальные особенности. Например, предметная область изображений документов технических заданий и дипломов визуально более строги, в отличии от рекламных брошюр, где нет определенных правил составления макета документа.

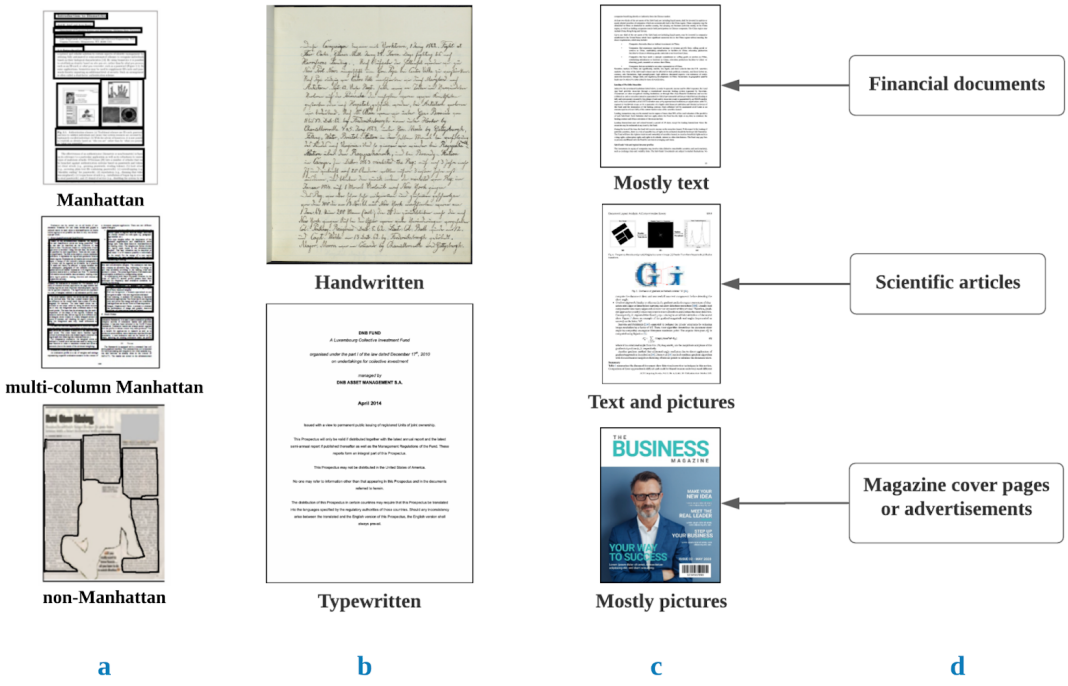


Рисунок 1.2.2.1 – Классификация текстовых документов с точки зрения автоматической обработки. (a) –- типы шаблонов страницы документов, (b) – тип текста документа, (c) – тип содержимого страницы.

На рисунке 1.2.2.1 представлена классификация текстовых документов, которая состоит из следующих классов:

A. Тип макета страницы документа задает правила расположения элементов на странице. Распространенным макетом является тип “Манхеттен”, где блоки (элементы) расположены параллельно относительно друг друга;
B. Тип текстового содержимого: рукописный или печатный;
C. Тип объема текстового содержимого. Здесь различают на сколько много графического контента может присутствовать на странице.

В диссертационной работе делается упор на анализ текстовых документов с печатным текстом, с превалирующим количеством текстовой информации. При этом документы имеют шаблон типа “Манхеттен” и могут содержать несколько колонок текста.

1.2.3 Предварительная обработка сканированных документов

Предварительная обработка сканированных документов имеет свои особенности, которые связаны с тем, что сканированные изображения могут содержать различные дефекты, такие как шумы, искажения, неровности, размытие, искаженную геометрию и т.д. Одной из особенностей предобработки сканированных документов является необходимость учета различных типов дефектов и выбор оптимальных методов для их коррекции. Например, для устранения шумов на изображении может быть применена медианная фильтрация, а для исправления искажений геометрии - аффинные и проективные преобразования.

Еще одной особенностью предварительной обработки является важность сохранения качества изображения. При применении различных методов может происходить потеря мелкоразмерных деталей, например, затирание символов или размытие границ объектов на изображении, что может отрицательно повлиять на дальнейшее распознавание страницы документа.

Тем не менее, сканированные изображения документов, как правило, имеют более высокое качество, чем сфотографированные, по ряду причин: сканеры обычно обладают более высоким разрешением; при сканировании документ размещается плоско, без искажений перспективы; отсутствуют отражения и блики, а лист равномерно освещен. Эти преимущества позволяют сократить список задач предварительной обработки, которые решаются при автоматической обработке изображений документов.

В рамках работы рассматриваются методы предварительной обработки:

- исправление малого угла наклона страницы;
- исправление ориентации изображений документов.

1.2.3.1 Обнаружение и исправление малого угла наклона страницы

Для обработки черно-белых сканированных документов существует ряд зарекомендовавших себя классических методов [14], например, определение угла поворота по вычислению горизонтальных проекций на ось Y.

Для изображения I размером $M \times N$ параллельная горизонтальная проекция определяется следующим образом:

$$\pi_I(i) = \sum_{j=0}^{N} I(i,j)$$

Определение перекоса в этом случае сводится к последовательному повороту исходного изображения на заданный набор углов от ϕ_1 до ϕ_L и выбору лучшей проекции. Очевидно, что для ранжирования полученных проекций должен быть предоставлен некоторый критерий $f(\pi)$. В результате выбирается угол ϕ_k, при котором достигается глобальный экстремум $f(\pi)$. В качестве критерия часто используют сумму квадратов значений элементов проекции, согласно формуле:

$$f(\pi) = \sum_{i=0}^{M} \pi^2(i)$$

1.2.3.2 Обнаружение и исправление ориентации страницы документов

Определение ориентации страницы документа в диссертационной работе будем рассматривать как задачу определения ориентации входной страницы на 4 угла в 0, 90, 180, 270 градусов.

Для решения часто рассматриваются классические подходы [15] определения угла наклона документов такие, как преобразование Хафа (Hough Transform) и использование проекций значений пикселей на оси. Оба метода позволяют определить только две ориентации горизонтальную или вертикальную. С помощью таких методов невозможно отличить 90◦ от 270◦ или 0◦ от 180◦.

Метод с использованием преобразование Хафа позволяет выделить прямые линии на бинаризованном изображении и статистически определить наиболее часто встречающийся наклон. Помимо неточного определения ориентации, у этого метода есть ряд существенных недостатков. Во-первых, этот метод чувствителен к шумам на

изображении и к графикам разных видов, таблицам и иллюстрациям, которые, как описано в разделах выше, присутствуют в используемых данных. Во-вторых, этот метод достаточно медленно работает. Реализация его в библиотеке OpenCV обрабатывает изображение примерно за 0.5 секунд.

Метод с использованием проекций работает следующим образом. Подсчитывается количество чёрных пикселей в столбцах бинаризованного изображения, далее вычисляется сумма квадратов полученных значений. Предполагается, что полученное значение максимально, когда текст на изображении расположен горизонтально (т.е. ориентация документа 0◦ или 180◦). Этот метод работает достаточно быстро, но тоже достаточно чувствителен к наличию иллюстраций, таблиц и графиков на изображениях.

В работе [16] предлагается отслеживать, сколько пересечений имеют буквы, содержащиеся на изображении документа с вертикальными линиями. Предполагается, что для каждого языка среднее количество таких пересечений разное. Эта информация используется для определения ориентации документа. Но, как и в предыдущих методах, этот метод не даёт возможности отличить 0◦ от 180◦ и 90◦ от 270◦.

В работах [17, 18, 19] используются свёрточные сети для определения угла поворота изображений относительно горизонтальной оси. Также, метод с использованием проекций может быть применен для выделения признаков с изображений для дальнейшей классификации с помощью классических алгоритмов машинного обучения. Например, для изображения 1200 × 1200 пикселей вычисляется 2400 признаков, из которых 1200 будут числами, отражающими суммы в соответствующих строках, и ещё 1200 - в соответствующих столбцах.

Стоит сказать про процесс предобработки изображения перед его подачей в классификатор. Согласно предложениям в работе [21], в процесс предварительной обработки изображений входит удаление шума, исправление небольшого угла поворота и бинаризация.

1.2.4 Анализ макета документа

Сегментация страниц — важнейший этап автоматического анализа документов, на котором определяются значимые области документа (Рисунок 1.2.4.1).

Анализ макета страницы документа (Document Layout Analysis — DLA) — это процесс сегментации страницы изображения документа на разные области, например, на текстовую, табличную, заголовок и т.д. Для каждой области требуются специальные обработчики для извлечения информации. Например, обработчики текстовых блоков — это OCR-системы, обработчики табличных блоков — это методы распознавания таблиц.

Как правило, методы сегментации страницы играют существенную роль для распознавания документов со сложным макетом (шаблон страницы не типа “Манхэттен”) [22, 23, 24]. На таких данных, без этапа сегментирования будет неизвестен порядок чтения страницы. Например, содержимое раздела извлекается раньше заголовка раздела, или две колонки в документе склеиваются в один блок текста.

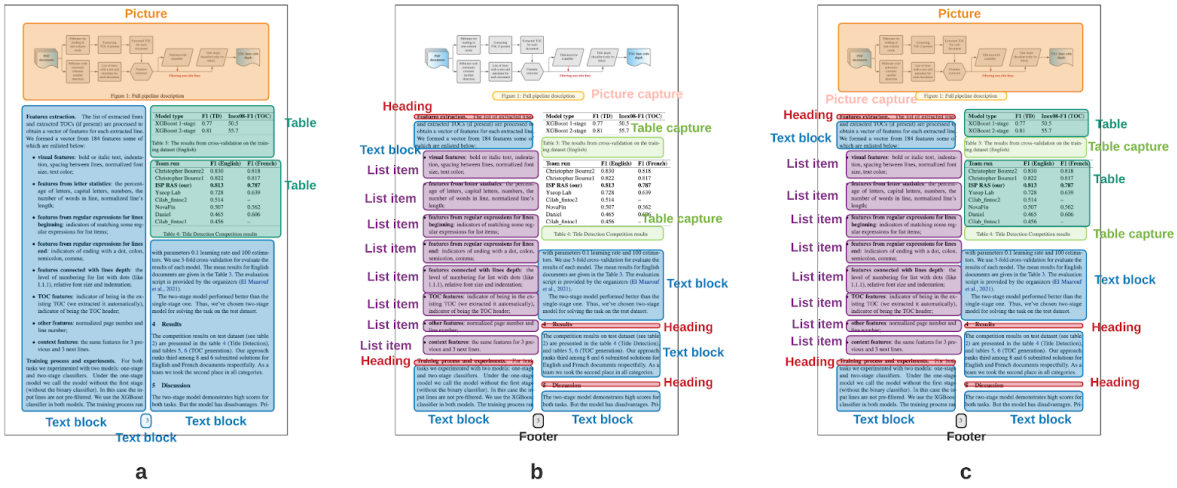


Рисунок 1.2.4.1 – Пример работы этапа анализа страницы текстового документа.

Различные работы [22, 23, 24] могут описывать этот процесс по-разному, но есть некоторые этапы, общие в работах. Выделяют следующие этапы анализа макета страницы документа:

- **Предварительная обработка**. Необязательный этап, который может играть решающую роль во всем процессе понимания документа. Он может включать устранение искажений, выравнивание, шумоподавление, бинаризацию и т. д. Этот этап используется для подготовки изображения для последующего распознавания содержимого.

- **Сегментация страницы**. На этом этапе используется алгоритм сегментации. Методы, используемые на этом этапе, могут возвращать извлеченные области с определенными как с метками, так и без них.
- **Постобработка**. Этап постобработки является не обязательным в большинстве алгоритмов DLA, но в некоторых случаях он может значительно улучшить результаты модели.

Подводя итоги, стоит отметить, что открытые решения с глубоким обучением для сегментации страниц предназначены для обработки страниц с определенным шрифтом, межстрочным интервалом или макетом, а адаптация к новым шрифтам требует сбора нового тренировочного набора данных [7, 8]. Для обработки страниц с большим количеством текста, как в рассматриваемых предметных областях “Технические задания”, “ВКР”, “Законы”, справляется метод сегментации, встроенный в открытую библиотеку Tesseract OCR. Сегментация встроенная в библиотеку позволяет находить текст на разных уровнях (текстовые строки, текстовые блоки) с помощью заданных настроек макета. В диссертационной работе извлечение информации и анализ структуры документов проводится через анализ текстовых строк. Для извлечения исключительно текстовой информации достаточно использовать готовые средства сегментирования страниц [25] документа, доступные в Tesseract OCR. Для Tesseract OCR достаточно задать параметр сегментирования ‘psm’, который позволяет находить текстовые строки как для одноколоночного (‘psm’=4), так и для многоколоночного документа (‘psm’=3).

1.2.5 Извлечение текстовой информации

Извлечение текстовой информации – одна из самых главных задач распознавания документов. Извлечение информации, как правило, является финальным и самым важным этапом обработки документов. Для извлечения содержимого важно иметь четкое изображение документа с различимым для человека шрифтом и текстом. Согласно рассматриваемой предметной области, сканированные документы содержат машинописный текст, характеризуются светлым фоном и темным текстом, имеют высокое разрешение. Для анализа точности методов распознавания символов важно рассмотреть факторы, отрицательно влияющие на точность распознавания документа.

1.2.5.1 Факторы, влияющие на точность распознавания OCR

Процесс OCR зависит от ряда факторов, радикально влияющих на точность процесса распознавания:

- неоднородность фона документа;
- избыточная сегментированность текста;
- переменные размеры и типы шрифта;
- пространство между строками;
- пространство между колонками текста;
- наличие опечаток в тексте;
- плохое качество документа (например, непропечатанные чернила);
- присутствие шума на изображении документа;
- недостаточная освещенность во время экспозиции документа;
- наличие рукописного текста;
- смешанные языки;
- заведомо не установлен язык текста в системе OCR;
- искривленные текстовые линии;
- направление текстовых линий отклонено от горизонтального направления;
- текст обрезан в результате плохих условий съемки;
- изображение искажено в результате движения во время экспозиции или плохой фокусировки камеры.

Накопленный на сегодняшний день опыт показывает, что программное обеспечение OCR выдает высокую точность распознавания символов на изображениях хорошего качества с заведомо известным алфавитом и заданным шрифтом. В данном случае, большинство символов будут распознаваться правильно, что приводит к успешному извлечению слов из изображения.

1.2.5.2 Выбор библиотеки распознавания текстовой информации

Для извлечения текстовой информации используют методы OCR и методы DLA. DLA необходим для правильной компоновки распознанного текста в соответствии с порядком чтения в документе. Например, в структурированных документах, таких как DOCX, HTML и PDF с разметкой прописано в каком порядке

извлекается текстовое содержимое. При анализе изображения документа мы не обладаем такой информацией, поэтому выполняют анализ макета документа. Точность извлеченного текста напрямую зависит от точности распознавания макета страницы.

Существуют несколько популярных OCR библиотек, например [26, 27, 28], которые достигают высокой точности и предназначены для обработки сканированных текстовых документов высокого качества. Одной из наиболее известных библиотек является Tesseract [26]. Tesseract – открытое программное обеспечение для OCR, активно поддерживаемое компанией Google. Tesseract работает на основе сети LSTM, которая позволяет повысить точность распознавания текста.

В Tesseract OCR поддерживается более 100 языков. Также Tesseract OCR предоставляет возможность обучения на новые языки. Анализ макета страницы включен в функциональность Tesseract и задается настройкой “psm”, позволяющей анализировать определенные шаблоны макета страницы (раздел 1.2.5). В Tesseract OCR включена постобработка для исправления ошибок в распознанном тексте.

Tesseract OCR подходит для распознавания текста с предварительно обработанной страницы сканированного документа.

1.2.6 Распознавание табличных данных со сложной структурой

Таблицы — один из распространённых способов представления информации. Они повсеместно используются в документах и веб-пространстве, таких как HTML, PDF, EXCEL, CSV. Таблицы характеризуются большим разнообразием компоновок, стилей и содержания, а также высокой скоростью роста их объёма. Большой объём и структурные особенности таблиц делают их ценным источником для приложений в области науки о данных и бизнес-аналитики. Однако, как правило, они не сопровождаются явной семантикой, необходимой для машинной интерпретации их содержания, как задумано автором. Информация в них часто не является структурированной и стандартизированной. Анализ таких данных требует их предварительного извлечения и трансформации в структурированное представление с заданной формальной моделью.

Анализ табличных данных фокусируется на трех задачах:

1) Обнаружение таблиц (обнаружение координат сегмента на изображении, в пределах которого расположена таблица);
2) Распознавание структуры таблицы (физической структуры) — это процесс определения столбцов, строк и ячеек на изображении, а также выявления признаков объединения ячеек;
3) Интерпретация (распознавание логической структуры) — это восстановление смысла табличной структуры путем определения заголовка таблицы, роли каждой ячейки в таблице и их связей друг с другом.

Некоторые работы [44, 45] под задачей распознавания таблиц рассматривают только первые две описанные выше подзадачи. В диссертации акцент сделан на работах, связанных с первыми двумя подзадачами, поскольку логическая структура таблиц (их интерпретация) тесно связана с конкретной предметной областью таблицы и требует разметки данных. Методы, решающие последнюю задачу, обычно основаны на предопределенных шаблонах [46, 47, 48].

Методы распознавания таблиц можно систематизировать в следующем виде:

1) Методы обнаружения таблиц и распознавания структуры одновременно.
2) Методы обнаружения таблиц. Здесь выделяют методы, основанные на:
    a) моделях обнаружения объектов на изображениях;
    b) других подходах;
3) Методы распознавания структуры таблиц, основанные на:
    a) моделях обнаружения объектов на изображениях;
    b) моделях обнаружения ячеек;
    c) обнаружении ячеек и распознавании текста;
    d) другие методы: заточенные на специфику предметной области, извлечение структурной иерархии.

В данной работе рассматриваются методы по распознаванию физической структуры таблиц без анализа интерпретации таблиц. Задача заключается в определении границ таблиц и ячеек, выявлении объединённых ячеек и распознавании текстового содержимого ячеек.

Современные методы на основе машинного обучения [29] направлены на распознавание таблиц без границ. Такие методы зависят от тренировочных наборов данных, где модели затачиваются на размер шрифта, межстрочный интервал и интервал между ячейками. Такие методы применимы в областях, где табличную информацию нельзя извлечь классическими подходами на основе связанных

компонент или обнаружения вертикальных и горизонтальных линий на таблицах с явными границами. Но несмотря, на активное развитие методов на основе глубокого машинного обучения, область распознавания таблиц без границ остается вызовом для исследователей. Для обнаружения таблиц и столбцов и ячеек здесь часто применяются классические модели обнаружения объектов на изображении FasterRCNN, FasterRCNN, YOLO и другие модели [30, 31, 32, 33, 34, 35].

Различные работы, связанные с распознаванием структуры таблиц, рассматривают эту задачу по-разному. Некоторые методы [36, 32, 37, 38] решают задачу распознавания таблиц как задачу сегментации. Другими словами, они находят ячейки таблицы как физические элементы без анализа их содержимого. Существуют также методы извлечения как физической, так и текстовой информации из ячеек таблицы [39, 40]. Эти методы не нацелены на глубокое понимание содержимого ячейки, но они позволяют интеллектуально анализировать текст ячеек на следующих этапах при автоматической обработке. Существуют методы, решающие более сложные задачи, связанные с табличной структурой, например, преобразование изображений таблиц в структурированные форматы с иерархией ячеек [41] в TEX, или в HTML [42]. Такие методы основаны на архитектурах трансформер и требуют большого тренировочного набора данных.

В рассматриваемой предметной области таблицы представлены с явными границами, что облегчает процесс их обнаружения и распознавания. Существующие классические подходы [43, 44] обнаружения таблиц с границами подразумевает обнаружение вертикальных и горизонтальных линий (границ) таблиц, с последующей работой над ними. Обнаружение границ таблиц проводится с помощью применения различных фильтров к изображению и применения контурного анализа для обнаружения линий на изображениях. При этом открытых решений, позволяющих распознавать физическую структуру таблиц с границами не было найдено. В разделе 2.2.2 будет представлен классический метод обнаружения и распознавания таблиц, используемый в программном комплексе на основе применения контурного анализа к изображению.

Наборы данных и оценка качества. Для оценки решений распознавания таблиц существуют доступные наборы данных. Открытые наборы данных [49, 50, 42], содержат таблицы без явных границ с разметкой, где содержится информация о связях между ячейками, а в некоторых наборах текстовое содержимое ячеек. Например, набор WTW [49] содержит изображения таблиц плохого качества,

полученных путем съемки на смартфон в плохих условиях. ICDAR 2013 Table Competition [50] набор данных, содержащий PDF-документы с текстовым слоем и таблицы без явных границ. PubTabNet [42] содержит PDF-документы медицинских статей, в которых таблицы представлены без границ. Набор данных с соревнования ICDAR 2019 [51] содержит таблицы с рукописным текстом.

В литературе метрика TEDS наиболее показательна и популярна среди исследователей в данной области. Метрика показывает сходство (расстояние редактирования) между двумя деревьями ячеек таблиц. Иными словами, эта метрика учитывает структуру таблицы, представленной в виде дерева ячеек. Оценка TEDS (формула 1.2.6.1) рассчитывается как минимальное количество правок, необходимых для преобразования предсказанного дерева ячеек в дерево ячеек из разметки. При этом в метрике можно учитывать текстовое содержимое ячеек, где стоимость замещения в дереве вычисляется как нормализованное сходство Левенштейна [52] (от 0 до 1) между текстами двух ячеек (узлов двух деревьев). Метрика рассчитывается по формуле 1.2.6.1.

$$TEDS(T_a, T_b) = 1 - \frac{EditDist(T_a, T_b)}{max(|T_a|, |T_b|)} \qquad (1.2.6.1)$$

, где EditDist - это расстояние (число операций редактирований), чтобы из одного дерево получить второе, |T| - число узлов в дереве.

1.2.7 Обработка PDF-документов

Выделяют два вида PDF:

- Некопируемые PDF, которые предназначены для хранения изображений;
- Копируемые PDF, в которых хранится информация о выводе на печать глифов. Такие PDF содержат текстовый слой;

Извлечение текстовой информации из PDF-документов является важным процессом, который может быть выполнен с использованием технологии оптического распознавания символов (OCR) [53] или путем извлечения текстового слоя с помощью PDF-библиотек, например [59, 60, 61]. При этом, процесс OCR вычислительно сложен, затратен по времени и ресурсам и неэффективен для документов с уже существующим правильным текстовым слоем в формате PDF. К

тому же, OCR-системы подвержены распознаванию с ошибками, которые зависят от различных факторов, таких как неправильная идентификация языка, качество изображения, особенности шрифта и появление новых символов (подробная информация представлена в разделе 1.2.5.1). Отсюда, для PDF-документов эффективнее получать текст из текстового слоя PDF, если он присутствует.

Цель обработки — автоматически извлечь корректную текстовую информацию из PDF-файла, независимо от того, хранится ли она в текстовом слое или на изображении отсканированного текста.

При автоматической обработке PDF-документов возникает необходимость проверки правильности извлекаемого текста. По ряду причин извлечённый текст из текстового слоя PDF-документов может быть несвязным или некорректным, например, состоять из бессмысленных символов (кракозябр), несмотря на то, что документ отображается правильно в PDF-редакторе и без искажений в тексте, т.е. глифы в PDF, отображающие символы, правильные. Примеры изображены на Рисунке 1.2.7.1. Такие PDF-документы будут далее в тексте называться некорректными.

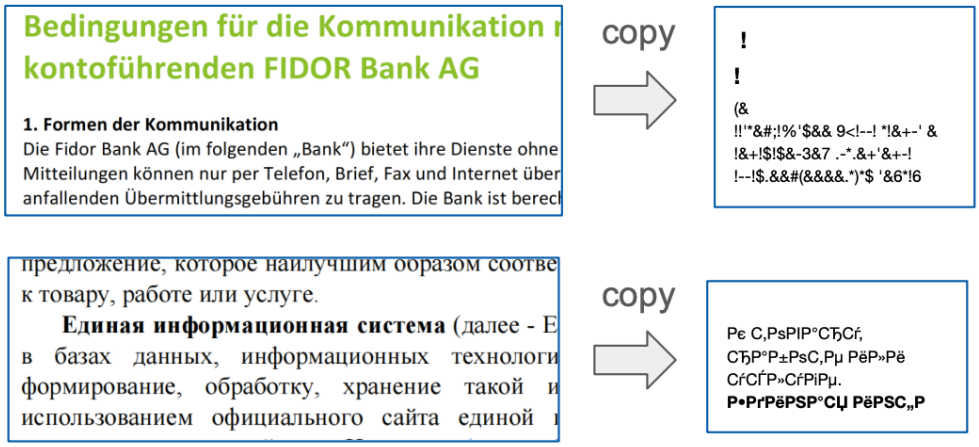


Рисунок 1.2.7.1 — Примеры некорректного текста в текстовом слое PDF-документа.

В ходе обзора было установлено, что в настоящее время не существует комплексного решения, которое бы полностью решало задачу автоматического извлечения текстового содержимого из некорректных PDF. Также была найдена только одна работа [20], исследующая обработку некорректных PDF документов, авторы восстанавливают поврежденный шрифт в PDF документе с использованием сверточной нейронной сети.

Существуют работы, посвященные разработкам около данной темы, например [65], направленные на исправление ошибок после распознавания текста системами

OCR. Однако эти методы не решают всех аспектов обнаружения корректности, поскольку не учитывают ошибки глифов, не исследуют проблемы шрифтов PDF. Такие методы ориентированы на интеллектуальный анализ единичных ошибок в тексте, созданных людьми или методами OCR, что требует использования больших языковых моделей, способных учитывать контекст и грамматику текста, направленных на проверку семантики текста. Такие методы затратны и избыточны для задачи обнаружения некорректного текста как показано на примере (Рисунок 1.2.7.1). Далее описаны группы методов, позволяющих обнаружить некорректность текста.

1.2.7.1 Методы проверки корректности текстов

Проверка правильности слов с использованием N-грамм. В [54] авторы описывают систему проверки правильности слов бенгальского языка (bengali language) с использованием вероятностной модели на основе N-грамм. N-грамма в обработке текстов - это последовательность из N букв. В [57] авторы используют стохастическую модель правильности предложений на ассамском языке (assamese language). Была разработана вероятностная система, которая учитывает оценку вероятности моделей униграмм, биграмм и триграмм. Для обработки нулевых вероятностей биграмм или триграмм используется линейная интерполяционная модель. Первый шаг включает в себя разбиение большого корпуса на униграммы и подсчет их частот в корпусе. То же самое делается для биграмм, триграмм и квадрграмм. Второй шаг — разработка набора программных модулей для вычисления вероятностей биграмм, триграмм и квадрграмм с использованием N и N-1 грамм.

MLE-оценка вероятности N-граммной моделей. В контексте N-грамм моделей оценка вероятности становится решающей. Оценка максимального правдоподобия (MLE) [58] для оценки параметров N-граммных моделей является классическим методом. В случае, когда N-граммы не встречались в обучающих данных и дают нулевые вероятности на тестовой выборке, то используют сглаживание, например сглаживание Лапласа (или аддитивное сглаживание) [55]. В случае с проверкой корректности текста в качестве N-граммы берутся не слова, а символы, как это предложено в работе [54] (т. е. униграмма - это один символ, биграмма - это два символа, триграмма - это три символа.

1.2.7.2 Причины возникновения некорректности текстового слоя в PDF

Для разработки метода определения корректности текста текстового слоя необходимо изучить причины возникновения некорректного текста из PDF. В результате проведенных исследований были выявлены следующие основные причины: отсутствующий или некорректно внедренный шрифт в PDF-файл, некорректная кодировка шрифта, некорректная обработка программой для конвертации или редактирования PDF файлов, неправильный выбор языка OCR для восстановления текстового слоя, установлена безопасность файла от копирования. Стоит отметить, что причины возникновения некорректного текстового слоя не были найдены в научной литературе и были собраны из документации Adobe [63, 66] и форумах [60 - 62, 67 - 69] по проблемам чтения PDF файлов.

Проблема отсутствующих шрифтов. Проблема отсутствия шрифтов в документе PDF или в системе, где открывается документ приводит к тому, что файл открывается с некорректной визуализацией или некоторые символы полностью отсутствуют. Этой проблеме способствуют несколько причин:

- Когда шрифт отсутствует, система, в которой открывается PDF, может заменить его другим стандартным шрифтом, что может сделать текстовый слой некорректным. Когда программа пытается найти отсутствующий шрифт, могут возникнуть несоответствия между разными шрифтами. При открытии файла PDF в другой кодировке, если последовательность байтов соответствует разным символам в исходной кодировке и кодировке, используемой при чтении PDF, текст может отображаться неправильно или стать нечитаемым [9]. Например, при использовании кодировки KOI8-R вместо CP866 фраза «Привет, мир!» будет отображаться, как показано на рисунке 1.2.7.2.
- Работа с нестандартными шрифтами End-User-Defined Characters (EUDC) [62], созданными пользователем, может привести к неправильному текстовому слою. Шрифты EUDC используют область Private Use Area (PUA) [64] в таблице Unicode. В таких случаях, при чтении/копировании PDF-файла EUDC-символы отображаются как «U+E000..U+F8FF» при отсутствии пользовательского шрифта EUDC.
- Существуют старые шрифты, которые больше не включены в стандартный список шрифтов, что приводит к неправильному текстовому слою.

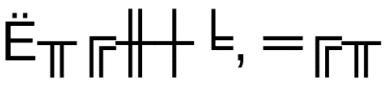


Рисунок 1.2.7.2 — Пример отображения некорректной кодировки в PDF.

Неправильная кодировка шрифта. Неправильная кодировка шрифта является распространенной проблемой в документах PDF, и существует несколько причин, по которым кодировка может измениться:

- Сжатие документа и включение опции «сжатие шрифта» во время сохранения файла может привести к потере шрифта. Когда детали шрифта теряются, это может привести к неправильному отображению в будущем в процессе открытия PDF.
- Проблемы отображения в Unicode-символы. Чтобы уменьшить размер PDF-файлов, особенно при использовании восточноазиатских шрифтов с большими наборами символов, вместо отдельных графических глифов для представления символов используются специальные CID-коды (character id) [63, 66]. CID-коды предоставляют стандартизированный метод представления символов в файлах PDF, особенно для языков с обширными наборами символов. В таких случаях, PDF использует составные шрифты, также известные как CID-шрифты, которые эффективно сопоставляют коды CID с соответствующими глифами. Такой подход помогает оптимизировать размер файла, обеспечивая при этом точное представление текста, особенно в документах со сложными требованиями к символам. Такие шрифты могут включать раздел/файл ToUnicode CMap, который сопоставляет код CID символа с Unicode символом при операциях копирования, вставки и поиска текста [56]. Однако сопоставления ToUnicode могут быть неполными и иногда даже содержать намеренно неверные символы. В первом случае символы будут отображаться как «(cid:N)» [60], а во втором — как явно неверные символы.

Неправильная обработка системами конвертирования/редактирования. Проблемы с PDF могут также возникнуть из-за использования неправильно установленных, устаревших или нелицензированных программ, предназначенных для работы с PDF-файлами. Это может привести к ошибкам при отображении документа в редакторах и в процессе конвертации. Например, при открытии PDF-файла в Adobe Reader может возникнуть несколько ошибок [68]. Чтобы избежать подобных проблем,

важно использовать надежное и надлежащим образом лицензированное программное обеспечение [69].

Защита PDF. Некоторые PDF файлы могут быть защищены пользователями, чтобы предотвратить изменение, печать или открытие файла. В первом случае текстовый слой файла PDF недоступен, или извлекаемый текст может быть намеренно запутан [67] путем поврежденным отображением глиф-символ Unicode, которое хранится в секции ToUnicode CMap. Для автоматизированной обработки одна из самых существенных проблем возникает, когда редакторы заменяют шрифты собственными нестандартными шрифтами, что может изменить кодировку.

Неправильный выбор языка для OCR. Неправильный выбор языка в OCR может привести к неправильному распознаванию текста с изображений. Например, если вы захотите добавить текстовый слой в отсканированный PDF-документ, написанный на русском языке с помощью инструмента оптического распознавания символов, при выбранном английском языке, текст будет распознан неправильно, как показано на рисунке 1.2.7.3. Это зачастую происходит при автоматическом восстановлении текстового слоя PDF документов.

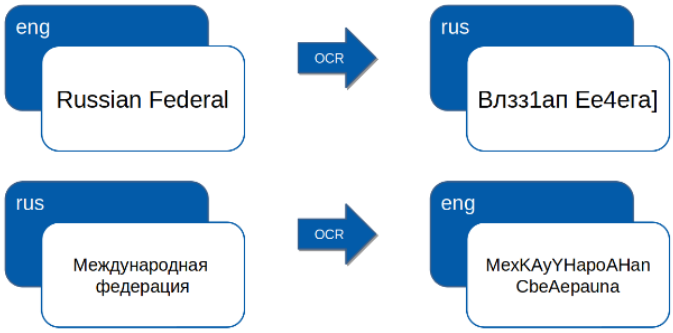


Рисунок 1.2.7.3 — Результат некорректного выбора языка распознавания.

1.2.8 Выводы к разделу 1.2

В главе 1.2 были рассмотрены методы обработки изображений сканированных документов и особенности извлечения текста из PDF-документов, содержащих текстовый слой.

Согласно обзору раздела 1.2.1 обработка изображений сканированных документов с целью извлечения текстового и табличного содержимого требует построения последовательности решений, каждое из которых важно для достижения

эффективной автоматической обработки. Для обработки изображений предлагается следующий набор последовательно применяемых методов:

- Предобработки изображения (раздел 1.2.3), где исправляется малый угол поворота и ориентация страницы сканированного документа.
- Сегментирования страницы (раздел 1.2.4), где устанавливается правильный порядок чтения текста на странице документа. Для рассматриваемой предметной области документов с простыми шаблонами страниц достаточно использования встроенных средств сегментирования страницы в Tesseract OCR.
- Распознавание текста (раздел 1.2.5), где применяются методы OCR. Согласно обзору раздела 1.2.5 открытое решение Tesseract OCR показывает высокие результаты распознавания и имеет ряд преимуществ.
- Распознавание табличной информации (раздел 1.2.6), где производится как обнаружение, так и распознавание структуры таблиц, учитывающее связи между ячейками таблиц. Согласно рассматриваемой предметной области, таблицы в документах имеют явные границы, поэтому для построения решения стоит обратиться к классическому подходу обнаружения горизонтальных и вертикальных линий на изображении.

Исходя из обзора обработки PDF-документов (раздел 1.2.7) и возможных причинах возникновения некорректного текстового слоя в них, было решено разработать метод позволяющий автоматически определять некорректный текстовый слой в PDF-документах при условии его наличия. Такое решение позволит увеличить качество и скорость получения содержимого из PDF-документов в автоматической обработке.

1.3 Восстановление иерархической структуры документа

Восстановление структуры проводится на втором этапе анализа документа после того, как было извлечено его содержимое. Иерархическая структура позволяет правильно разбить текстовый документ на текстовые блоки и присвоить текстовым блокам логические метки (например классы), которые можно использовать в

дальнейшем интеллектуальном анализе документов, производить поиск по документу, выявлять более важные составляющие документа.

Иными словами, под структурой текстового документа понимается (иерархическая структура) иерархическое представление структурных частей документа так, чтобы его разные части располагались на разных уровнях и имели логические метки (классы или структурные элементы). Таким образом, была обеспечена возможность навигации по документу.

Анализ структуры документа — это процесс назначения логических меток физическим областям, выявленным в ходе анализа макета и содержимого документа [1]. Логические метки могут включать заголовки, оглавления, аннотации, параграфы, списки, колонтитулы, подписи, номер страницы и т.д. Структурные элементы для каждой предметной области документа могут меняться. Например, для технического задания структурными элементами будут: заголовок, содержимое, элемент содержимого, списки, список литературы и т.д. Для законов - глава, пункт, подпункт, приложение к закону, обычный список и т.д.

Структура документа описывает, как документ делится на более мелкие части, например, разделы или главы. Различают несколько способов представления структуры документа (Рисунок 1.3.1).

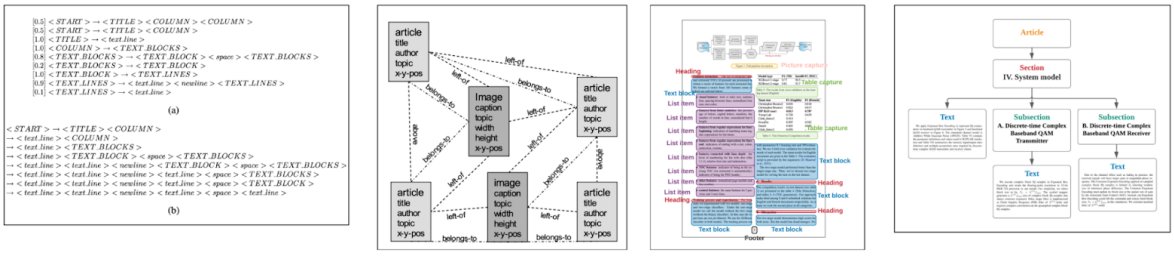


Рисунок 1.3.1 — Типы представления структуры документа: (a) — в виде грамматики, b -— в виде графа, c — последовательности линий (или текстовых блоков) с семантическими метками , d — в виде дерева.

Типы представления документа могут быть представлены в виде графовой структуры [70], формальной грамматики [82], последовательности строк (или текстовых блоков) с семантическими метками [6]. Но наиболее популярным типом представления структуры является древовидная иерархическая структура [87, 90, 93], где присутствуют иерархические связи между компонентами в документе.

В литературе выделяют две основные группы методов для восстановления структуры:

I. Поиск именованных сущностей в документе, например, поиск адреса, организаций, имен. Такая задача выполняется для поиска определенной (ключевой) информации из текста документа. Такая задача не будет рассматриваться в диссертационной работе;
II. Восстановление иерархии документа и определение уровня сегментов путем формирования иерархической структуры (дерева) из фрагментов текстового содержимого с учетом форматирования документа.

В диссертационной работе будет рассматриваться вторая группа методов восстановления структуры.

В последнее время восстановление иерархической структуры документов особенно актуально для решения задачи Retrieval Augmented Generation (RAG) [71]. RAG — это подход, который объединяет модели поиска и генеративные модели для создания текстового ответа на запрос (используются большие языковые модели LLM (Language Large Model)). Данный подход не только контекстуально точен, но и насыщен искомой актуальной информацией для поискового запроса. Прежде чем выполнить поиск по данным, документы в базе знаний должны хранится структурировано и быть разделены на управляемые блоки или сегменты. Этот процесс фрагментации документа гарантирует, что система сможет эффективно искать данные и обеспечит быстрый поиск запрошенного контента. При этом эффективные стратегии разделения документов на иерархические фрагменты могут значительно повысить скорость и точность системы поиска путем использования иерархического индексирования документов [78]. Пример изображен на Рисунке 1.3.2. В настоящее время открытый фреймворк Lang Chain[3] позволяет создавать системы RAG без трудоемкости с использованием готовых элементов, анализировать документы, векторизировать их, хранить векторизованное представление и т.д.

[3] https://www.langchain.com/

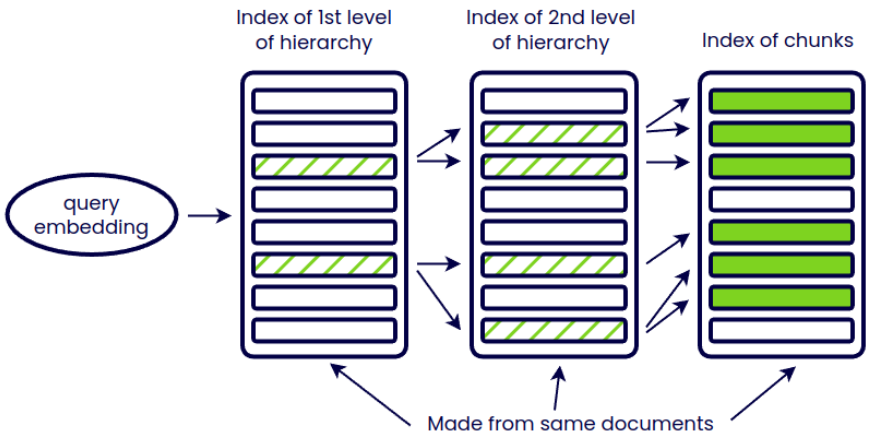


Рисунок 1.3.2 — Схема иерархического индексного поиска. Поиск выполняется пошагово. Поиск начинается с более крупных сегментов документа и переходит к более мелким фрагментам. На каждом уровне иерархии система работает только с сегментами, которые были извлечены как релевантные. Каждый уровень представляет собой произвольную структуру документа [78].

1.3.1 Обзор методов восстановления иерархической структуры документов

Согласно [79], иерархическая структура документа состоит из иерархии сегментов документа, каждый из которых соответствует визуально выделенному семантическому компоненту документа. Древовидная структура является наиболее интуитивно понятной структурой представления документа. Более того, для удобства навигации, в документах может хранится оглавление (TOC — Table Of Content), которое также имеет древовидную структуру. Таким образом, TOC можно рассматривать как представление иерархической структуры документа, а задача восстановления TOC почти аналогична задаче восстановления иерархической структуры, только ограничена построением иерархии из заголовков документа. Восстановление иерархии имеет фундаментальное значение для задач NLP, связанных с анализом структуры документов или тематическим извлечением, таким как извлечение информации или ответы на вопросы.

Как было сказано в начале главы для восстановления структуры применяют два основных этапа:

I. Извлечение содержимого с форматированием из документов. Содержимое получают используя сторонние библиотеки (парсеры формата файла) для соответствующего формата обрабатываемого документа;

II. Восстановление структуры, используя данные, полученные на первом этапе. Здесь анализируется не только текстовая информация, но и информация о визуальных признаках (например тип и размер шрифта).

Существует ряд работ [72, 75, 76, 77] для восстановления иерархической структуры заголовков документа, известную как оглавление (TOC). На тематику восстановления оглавления из финансовых PDF-документов организуют соревнование FinTOC [76], где участники концентрируются на восстановлении оглавления с определением глубины вложенности каждого заголовка. Другая работа [75], схожая с тематикой соревнования, описывает восстановление оглавления, но уже из книг в формате PDF. Восстановление иерархической структуры иногда ассоциируют с восстановлением модели документа DOM (Document Object Model), где узлами дерева являются логические теги, а листьями - неделимые элементы документа (например, текст, ссылки или картинки) (Рисунок 1.3.1.1).

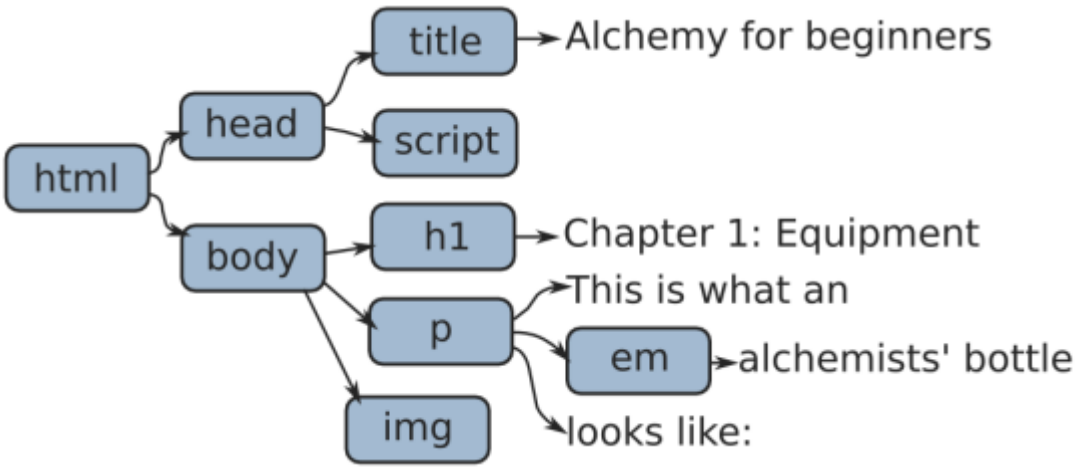


Рисунок 1.3.1.1 — Пример представления документа по модели DOM.

Есть другие работы, посвященные поиску элементов структуры для частных предметных областей. В литературе [72, 73, 74, 77] задача получения структуры документа упоминается как задача поиска на основе шаблонов документов (Template-based information retrieval). Данные работы концентрируются на отдельных ограниченных шаблонах и не решают проблему обработки разнообразия типов документов. Например, [72] извлекает оглавление TOC из научных статей и

инвестиционных документов формата PDF, в то время как работа [74] фокусируются на поиске ключевых полей в научных статьях с определенной конференции.

Существует множество примеров иерархических структур документов разных предметных областей. Научные статьи состоят из разделов, подразделов, подподразделов, книги состоят из глав, документы других предметных областей могут иметь некоторые отличные структурные элементы. Так, в нескольких работах рассматривается иерархическая структура различных типов документов, таких как научные статьи [72, 74, 80, 81, 83], финансовые документы [72, 76, 82], юридические документы [84], книги [75].

1.3.1.1 Соревнования и наборы данных

Соревнования восстановления структуры книг. Ряд соревнований был посвящен восстановлению структуры книг [85]. Целью соревнования – восстановить оглавление из оцифрованных книг. Размер набора более 2000 книг в [85]. Набор данных включает книги с оглавлением и без него. Метрика оценки описана в [86]. В этой статье запись TOC определяется как триплет Title (текст, например, «Глава 1 Введение»), Link – номер страницы, на которой начинается глава, и Depth Level – глубина, на которой глава находится в дереве TOC. Записи TOC считаются равными, если они имеют одинаковые Link и Depth Level, а тексты «достаточно похожи». Достаточное сходство измерялось с помощью модифицированного расстояния Левенштейна.

Соревнования восстановления структуры финансовых документов.

FinTOC 2019. Конкурс FinTOC-2019 [87] был посвящен восстановлению оглавления финансовых документов, а именно брошюр, технических описаний продуктов. Организаторы собрали набор данных из 58 документов из Люксембурга, написанных на английском языке. Все документы были представлены в формате PDF с текстовым слоем. Участникам было предложено два задания:

1. Определение заголовка — документ был разделен на текстовые блоки, участникам нужно было классифицировать каждый блок как «заголовок» или «не-заголовок». В качестве метрики оценки использовалась мера F1.
2. Извлечение TOC — участникам нужно было определить уровень иерархии заголовков. Использовалась мера ICDAR 2013 [85].

Лучшее определение заголовка. Лучшее решение [88] для определения заголовка основывается на рекуррентной модели LSTM с использованием аугментации данных. В работе модель word2vec была обучена на всем наборе данных, а полученные вектора потом классифицировались моделью LSTM. В качестве дополнительных признаков использовалась информация о длине текста, жирном и курсивном шрифте, заглавных буквах и информация о начале текста.

Лучшее извлечение глубины TOC. Лучшее решение [89] для задачи извлечения TOC не использует глубокое обучение. Вместо этого авторы полагаются на классификатор дерева решений и обнаружение страницы каждой записи TOC. Использование страницы TOC приводит к высокой точности, но относительно низким показателям полноты.

FinTOC 2020. FinTOC-2020 [90] является преемником FinTOC-2019 [87]. Организаторы собрали набор данных из 71 французского и 72 английских документов из Люксембурга. Все французские документы имеют общий шаблон, поэтому этот набор данных довольно однороден, но ни один из документов не имеет оглавления. Английские документы не имеют общего шаблона и характеризуются большим разнообразием форматов, с другой стороны, почти все они имеют страницу оглавления. Участникам было предложено извлечь оглавление (TOC) из французских (задание 1) или английских (задание 2) документов. В качестве метрики оценки использовалась адаптированная мера ICDAR 2013. В конкурсе FinTOC есть две задачи и два набора данных, поэтому есть четыре номинации:

(1) Лучшее определение заголовка для французских документов (лучшее решение [91]). Команда сосредоточилась только на задачах определения заголовка и получила лучший результат для французского набора данных и второй лучший результат для английского языка. Лучшее решение было получено с помощью метода максимальной энтропии. Использовались признаки разных типов: n-граммы символов, метаданные текста, такие как размер текста или жирность текста и т. п., название типа шрифта.

(2) Лучшее определение заголовка для английских документов (лучшее решение [92]). Команда из AMEX-AI Labs использовала нейронные сети для обнаружения заголовков и восстановления TOC. Они объединили геометрические и текстовые признаки. LSTM и CharCNN использовались для текстовых признаков, а полносвязная сеть использовалась для геометрических признаков. Модели были предварительно обучены на 6000 архивных документах.

(3) Лучшее извлечение TOC для французских документов (лучшее решение [93]).

(4) Лучшее извлечение TOC для английских документов (лучшее решение [93])

Авторы [93] получили лучший результат в задачах по извлечению TOC как для английского, так и для французского языка. Для извлечения текстовых областей использовался инструмент Tesseract OCR. Расстояние Левенштейна использовалось для сопоставления найденных текстовых областей и заданных меток. Для классификации использовалась лингвистическая и структурная информация: расстояние сверху и снизу (расстояние между текстовыми блоками), отступы, жирность, высота, ширина, информация о регистре текста, информация о том, что текст начинается с нумерации. Использовались три классификатора: SVM, логистическая регрессия и случайный лес, лучшие результаты были получены классификатором случайного леса.

FinTOC-2021. FinTOC-2021 [94] — это соревнование, похожее на FinTOC-2020. Были представлены те же 4 задания: определение заголовка и извлечение оглавления для английских и французских финансовых документов. Как и в 2020 году, большинство английских документов имели оглавление, французские документы почти не имели оглавления. Первые две команды в таблице лидеров — Christopher Bourez [95] и ИСП РАН [3]. Первая команда извлекала текстовые блоки из документов PDF с помощью ABBYY Finereader, вторая извлекала текстовые строки с помощью PdfMiner. Обе команды использовали свойства стиля и текста, а также некоторую статистику по ним для построения векторов признаков, которые передаются в классификатор XGBoost. Команда ИСП РАН предложила двухэтапную классификацию с использованием отдельных классификаторов для каждого этапа.

FinTOC-2022. В FinTOC-2022 [96] - были представлено 6 заданий: определение заголовка и извлечение оглавления из финансовых документов на английских, французских и испанских языках. Информация о наборах данных, о количестве документов с оглавлением и глубиной заголовков представлена на Рисунке 1.3.1.2. Команда ИСП РАН [2] в 4-х задачах по определению заголовков и определения глубины TOC для английских и французских документов заняла первое место. Разработанный метод [2], занявший первое место в соревновании, описан в этой диссертационной работе в разделе 2.3. Для испанских документов первое место заняла работа swapUNIBA [97], которая разработала систему анализа изображений документов, используя такие особенности макета, как заголовок, таблица, список и

текст. Для обнаружения заголовков использовалась модель обнаружения Faster R-CNN, предварительно обученная на открытом наборе данных PubLayNet. Для определения уровня заголовка авторы используют классификатор Random Forest, в качестве признаков используются первые пять и последние два символа заголовка текста, название и размер шрифта, координаты.

		English	**French**	**English**	**French**	**Spanish**
train	Number of documents	49	47	79	81	80
	Mean number of pages	64	30	77	27	119
	Number of TOC	43	4	69	6	74
	Mean number of titles	181	142	225	134	150
	Max title depth	9	6	9	9	7
test	Number of documents	10	10	10	10	10
	Mean number of pages	66	26	102	20	198
	Number of TOC	9	0	8	2	9
		2021		*2022*		

Рисунок 1.3.1.2 — Информация о наборах данных с соревнований FinTOC-2021, 2022.

1.3.1.2 Методы восстановления иерархической структуры

Задача восстановления структуры может быть интерпретирована как классификация. Например, в конкурсе FinTOC одной из задач была задача классификации заголовка/не заголовок. Подход победителей [91, 92, 96, 97] описан в разделе 1.3.1.1.

В работе [81] абзацы документа классифицируются по нескольким предопределенным категориям, таким как заголовок, группа авторов, информация и т.д. Для классификации использовался алгоритм Random Forest. Существует три типа признаков: визуальные (например, размер шрифта), поверхностно-текстовые (например, простые регулярные выражения) и синтаксически-текстовые (анализ частей речи). Для каждого абзаца выполняется бинарная классификация (например, заголовок/не заголовок).

Множество работ описывают восстановление иерархической структуры из документов разных доменов. Для восстановления иерархической структуры используются разные подходы.

В [83] восстановление структуры состоит из двух этапов. Первый этап включает классификацию строк на заголовок раздела и обычный текст. Второй этап включает классификацию разделов на верхний уровень, подраздел и подподраздел.

Авторы используют классические алгоритмы машинного обучения, такие как SVM, деревья решений и рекуррентные нейронные сети на уровне символов. К признакам относятся длина текста, количество именных фраз, размер шрифта, больший межстрочный интервал, полужирный шрифт, курсив, цвет и последовательность цифр в начале строки.

В [72] восстанавливают древовидную структуру из инвестиционных документов и научных статей. Задача решается в два этапа. Первый этап включает обнаружение заголовка. Это классификация текстовых блоков (несколько похожих строк объединяются в одну) на заголовок или нет. Для классификации используется Char-CNN. Первый слой — свертка, после чего к векторам текстовых блоков добавляется набор вручную сгенерированных признаков (для одного и для пары блоков). Второй полносвязный слой выполняет классификацию. Второй этап — упорядочивание заголовков. Векторы для заголовков формируются аналогично первому этапу, но на уровне слов. Для определения уровня используются модель обработки последовательностей BiLSTM и CRF. В результате извлекают иерархическое оглавление документа с ограниченной глубиной.

Некоторые работы [73, 74], как правило, заточены только на эвристические правила и регулярные выражения и не применяют машинное обучение.

1.3.2 Выводы к разделу 1.3

Структуру документа можно представить различными способами. Наиболее популярным способом представления является иерархическая структура.

Рассмотренные методы восстановления структуры из документов различаются как по рассматриваемому формату документа, так и по типу извлекаемой структуры. Методы восстановления структуры предоставляют возможность классификации строк (или параграфов) документа и выстраивания некоторой структуры, однако следует выделить следующие особенности:

- Рассмотренные методы ограничивают восстанавливаемую структуру: число уровней вложенности ограничивается заранее заданным числом. Методы построения дерева произвольной глубины опираются на правила, прописанные вручную, что уменьшает универсальность работы метода. Поэтому построение дерева документа произвольной глубины без

прописывания конкретных правил является улучшением существующих методов.

- Поскольку структура документа не является строго определенным понятием, для решения задачи восстановления иерархической структуры все чаще используются алгоритмы машинного обучения, в том числе нейронные сети. Чаще всего такой задачей является классификация абзацев документа. Для восстановления иерархической структуры некоторые методы предполагают двухэтапную классификацию: классификацию абзацев с заголовком/без заголовка и классификацию на уровне заголовка.
- Среди рассматриваемых типов документов наиболее популярными объектами исследования являются научные статьи и документы, связанные с бизнесом. Так, в последние годы было проведено несколько конкурсов FINTOC [90, 94, 96] по восстановлению структуры из финансовых PDF-документов.

На основании рассмотренных работ в данной области, стоит акцентировать внимание на соревновании FINTOC для восстановления иерархической структуры документа. На основании исследуемых работ победителей за 19 и 20 год, стоит использовать подход комбинирующий в себе и машинное обучение и эвристические правила над содержимым документа, а в качестве метода машинного обучения использовать деревья решений. В разделе 2.3 описан разработанный метод восстановления иерархической структуры, занявший первое место в соревновании FinTOC 2022. Данный метод был адаптирован для восстановления иерархической структуры без ограничения по глубине вложенности из рассматриваемых в диссертационной работе предметных областей: “Технические задания”, “ВКР”, “Нормативно-правовые акты”.

1.4 Обзор систем автоматической обработки документов

В процессе обзора было найдено некоторое количество существующих открытых систем для автоматической обработки текстовых документов разных форматов. Найденные системы рассмотрены по разным критериям, а именно по форматам документов, которые способны обрабатывать автоматически и по

способности восстанавливать иерархическую структуру в документах. Далее рассмотрена каждая их систем.

Система с открытым исходным кодом Textricator[4] извлекает структурированные данные из формата PDF с текстовым слоем. Система не позволяет анализировать неструктурированные форматы документов (изображения сканированных документов). Извлечение структуры документа проводится на основании правил, которые прописывает пользователь. Система не использует методы машинного обучения. Отсюда, система способна обрабатывать только строго шаблонные документы по правилам заданным пользователем. Система не предоставляет возможности расширения.

Библиотека с частью открытого исходного кода Unstructured[5] работает с несколькими структурированными форматами документов DOCX, HTML, XML, PDF. Во время обработки PDF библиотека не анализирует корректность текстового слоя, а просто проверяет его наличие. В случае, если PDF является набором сканированных изображений, то применяется библиотека Tesseract OCR без предварительной обработки страниц документов (поворота, исправления ориентации), в системе нет распознавания таблиц. Библиотека извлекает структуру из содержимого документа с помощью правил заданных пользователем. Библиотека внедрена в библиотеку построения LLM-систем Langchain. Система не предоставляет возможности расширения.

Система docling[6] приобрела высокую популярность, поскольку система позволяет обрабатывать как хорошо структурированные форматы (например DOCX, HTML) так и неструктурированные форматы изображений и PDF и интегрирована в несколько других систем построения LLM систем. Система не восстанавливает иерархическую структуру документа. Система не предоставляет возможности расширения.

Система deepdoctection[7] больше предназначена для автоматической обработки документов неструктурированных форматов изображений и PDF. Система включает несколько моделей глубокого машинного обучения для высокоточной обработки документов разных предметных областей. При этом система не восстанавливает иерархическую структуру содержимого документов. Система не предоставляет

[4] https://textricator.mfj.io
[5] https://github.com/Unstructured-IO/unstructured
[6] https://github.com/DS4SD/docling
[7] https://deepdoctection.readthedocs.io/en/latest/

возможности расширения поддержки новых форматов документов, при этом есть скрипты дообучения для некоторых поддерживаемых моделей глубокого обучения.

Библиотека pandoc представляет из себя библиотеку для конвертации большого перечня структурированных форматов в другие структурированные форматы. Особенностью является разнообразие поддерживаемых структурированных форматов документов Markdown, HTML, LaTeX, DOCX и т.п. Библиотека не предназначена для работы с PDF и изображениями. Библиотека не извлекает структуру. pandoc представляет из себя не систему, а набор библиотек для конвертации структурированных форматов файлов.

Аккумулирующая Java-библиотека Apache Tika[8] содержит в себе набор разных библиотек для извлечения содержимого из различных форматов документов. Способна извлекать содержимое из PDF с текстовым слоем с помощью библиотеки Apache PDFBox. Через библиотеку можно вызывать другие библиотеки, например Tesseract OCR или opencv. Представляет из себя не систему, а набор библиотек для разработки систем на языке Java.

Основным ограничением вышеупомянутых систем является отсутствие расширяемости для новых форматов и типов документов.

Существует ряд проприетарных систем для работы с разными форматами документов. Самой популярной проприетарной системой для работы с электронными документами является Abbyy Finereader. Система обладает интеллектуальными решениями для распознавания содержимого из изображений документов и PDF с последующим структурированием по настройке пользователя.

В Таблице 1.4.1 описано сравнение рассмотренных выше систем автоматической обработки электронных документов с точки зрения поддержки разных форматов, автоматической обработки, восстановления иерархической структуры документов, обработки некорректных PDF-документов и возможностей архитектур систем добавить поддержку новых форматов и предметных областей документов.

[8] https://cwiki.apache.org/confluence/display/TIKA/Parsers

Таблица 1.4.1 — Сравнительная таблица систем/библиотек для автоматической обработке электронных текстовых документов.

Система/библиотека	Открытость	Извлечение содержимого из структурированных форматов	Извлечение содержимого из неструктурированных форматов	Извлечение иерархической структуры	Расширяемость архитектуры	Является самостоятельной системой	Анализ некорректных PDF
Textricator	да	да	нет	на правилах	нет	да	нет
Unstructured	частично	да	да	на правилах	нет	да	нет
pandoc	да	да	нет	нет	нет	нет	нет
Apache Tika	да	да	да	нет	нет	нет	нет
Abbyy Finereader	нет	да	да	да	нет	да	нет
docling	да	да	да	нет	нет	да	нет
deepdoctection	да	нет	да	нет	нет	да	нет
Разработанный прогр.комплекс	да	да	да	да	да	да	да

1.5 Выводы к первой главе

В первой главе была раскрыта актуальность поставленных в диссертационной работе задач. Описан подход к автоматической обработке документов различных форматов. Были описаны особенности исследуемой предметной области документов, таких как технические задания, нормативно-правовые акты, выпускные квалификационные работы.

В главе рассмотрены проблемы автоматической обработки PDF-документов с текстовым слоем и исследованы причины возникновения некорректного текстового слоя в таких документах. Проанализированы существующие работы, касающиеся обнаружения некорректного текста в текстовом слое, что может помочь решить проблему извлечения содержимого из поврежденных PDF-документов в автоматическом режиме.

Также рассмотрены существующие работы по восстановлению иерархической структуры документов. Согласно обзору, выделяются два последовательных этапа: извлечение содержимого документа и восстановление иерархической структуры из полученного содержимого. Существующие научные работы ориентированы на обработку либо хорошо структурированных форматов (HTML, DOCX), либо PDF-документов с корректным текстовым слоем. При этом работы не охватывают обработку сканированных изображений документов, доля которых не уступает структурированным форматам.

Был проведен обзор существующих методов обработки изображений текстовых документов с учетом особенностей рассматриваемой предметной области. Рассмотрены методы обработки сканированных документов, а именно: методы исправления малого угла поворота и ориентации страницы документа, сегментации страницы, распознавания табличной информации и текста.

Согласно таблице 1.4.1, в которой приведено сравнение рассмотренных выше систем/библиотек, среди открытых решений для автоматического извлечения содержимого и восстановления структуры документов выделяются Textricator и Unstructured. Однако обе системы восстанавливают иерархическую структуру только по заданным пользователем правилам, без использования машинного обучения. Ручная настройка на основе шаблонов и правил восстановления ограничена лишь этими правилами, и качество восстановленной структуры зависит от опыта

пользователя. Кроме того, Textricator и Unstructured не ориентированы на работу с неструктурированными форматами документов, такими как изображения сканированных документов. Также во всех рассмотренных системах не поддерживается автоматическая обработка PDF с некорректным текстовым слоем. Стоит отметить, что архитектура существующих систем не обеспечивает расширяемость для поддержки новых форматов и типов документов.

Исходя из вышеизложенного, разработанный программный комплекс для автоматического извлечения содержимого и восстановления структуры из текстовых электронных документов обладает рядом преимуществ: он работает в автоматическом режиме как для структурированных, так и для неструктурированных форматов, поддерживает обработку некорректных PDF-документов и предоставляет возможность добавления поддержки новых форматов и типов документов. Эти преимущества особенно актуальны в условиях разнообразия форматов и типов документов.

Глава 2. Построение автоматической обработки электронных текстовых документов

В главе описано решение задачи автоматической обработки разноформатных документов рассматриваемых предметных областей. Согласно обзору области, исследовательские работы извлекают содержимое и восстанавливают структуру в два этапа. На первом этапе извлекается содержимое из документа, на втором этапе восстанавливается иерархическая структура из полученного содержимого.

В рамках диссертационной работы была построена двухэтапная автоматическая обработка документов как структурированных, так и неструктурированных форматов документов. На Рисунке 2.1 изображена схема предложенной обработки, где блоки 1, 2 и 3 предназначены для извлечения содержимого и формируют первый этап. Блок 4 относится ко второму этапу и отвечает за восстановление иерархической структуры обрабатываемого документа. Для автоматического извлечения содержимого из неструктурированных документов в формате PDF разработан метод, который учитывает некорректность PDF-документов. Метод с использованием бинарной классификации автоматически определяет корректность текстового слоя PDF и далее выбирает обработчик PDF-документа. На схеме (Рисунок 2.1 блок 1) метод стоит в самом начале обработки PDF-документов. Подробное описание метода представлено в разделе 2.1.

Для извлечения содержимого из изображений сканированных документов предложены разработанные методы (Рисунок 2.1 блок 2), описание которых представлено в разделе 2.2. Для извлечения содержимого из структурированных форматов документов используются сторонние библиотеки (Рисунок 2.1 блок 3).

Стоит отметить, что в случае необходимости, обработчики в блоках 2 и 3 в разработанном программном комплексе могут быть заменены другими. Например, если возникнет необходимость поддержки другого типа сканированных документов или будет найдено решение/библиотека, которое будет точнее извлекать текстовое содержимое (например лучше обрабатывать таблицы другого вида). Подробнее о расширяемости программного комплекса описано в главе 3.

На втором этапе извлеченное содержимое (текст с форматированием и табличная информация) передается методу восстановления иерархической структуры

документа (Рисунок 2.1 блок 4). Метод вычисляет уровни текстовых областей согласно предметной области документа и возвращает иерархическое представление содержимого, разбитого на секции. Метод подробнее описан в разделе 2.4 данной главы.

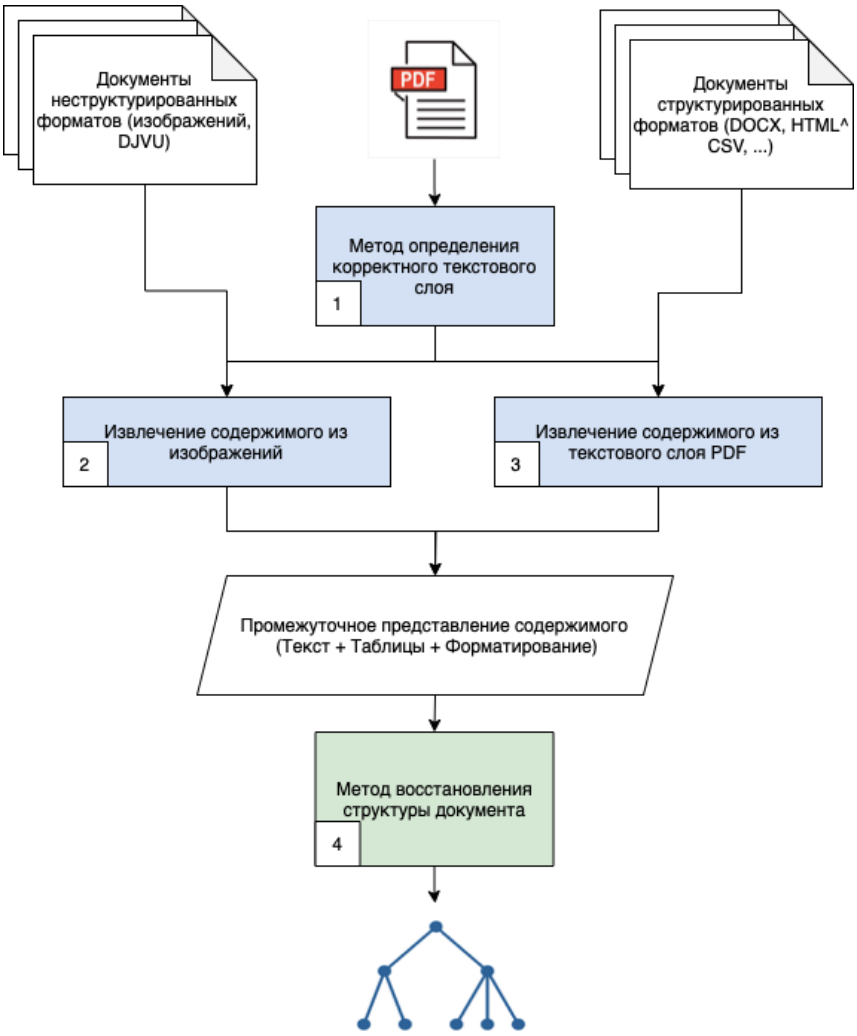


Рисунок 2.1 — Общая схема обработки электронных текстовых документов для извлечения их содержимого и восстановления структуры.

2.1 Метод автоматического определения корректности текстового слоя PDF

Разработанный метод берет на себя функцию автоматического определения корректности входного PDF-файла, и автоматически выбирает один из двух обработчиков для эффективного извлечения текстового содержимого документа (Рисунок 2.1.2) в автоматической обработке. Правило выбора обработчика следующее:

- В случае, если текстовый слой PDF-файла поврежден (некорректен), то эффективнее обработать PDF-файл как набор изображений сканированных документов через обработчик изображений сканированных документов с применением методов OCR. Методы обработки изображений программного комплекса описаны в разделе 2.2.
- В случае, если текстовый слой PDF-файла не пустой и является корректным, то эффективнее извлечь содержимое из текстового слоя формата PDF с помощью обработчика, использующего внешнюю библиотеку обработки PDF.

Таким образом, задача сводится к бинарной классификации (корректен/некорректен) текст текстового слоя PDF-документа.

Эффективность метода. Для получения текста из PDF с использованием методов обработки изображений и OCR требуется значительно больше времени и ресурсов памяти, чем при извлечении текста из текстового слоя PDF, поскольку форматы изображений содержат только визуальную информацию, и в этом случае обрабатываются только значения пикселей. Таким образом, в такой обработке обычно применяются методы машинного обучения. Однако если PDF-документ не содержит текстового слоя или он некорректен, анализ документа как набора изображений всё равно позволяет извлечь текстовое содержимое. В случае, если PDF-документ качественный и содержит корректный текстовый слой, извлечение текста с помощью внешней библиотеки будет быстрее и точнее. Таким образом, эффективность автоматической обработки PDF-документов сводится к автоматическому выбору одного из двух способов обработки.

Особенность предметной области. В обрабатываемых документах первая страница может быть представлена в виде сканированного изображения, в то время как остальные страницы PDF-документа имеют качественный текстовый слой и были получены путем конвертации из хорошо структурированных форматов, таких как офисные форматы DOC/DOCX. Это объясняется необходимостью сохранить подписи и печати, которые обычно размещаются на первой странице (Рисунок 2.1.1).

Техническое задание
на приобретение сертифицированных противообледенительных жидкостей
АО «Международный аэропорт «Казань»

Утверждаю
Директор по наземному
обеспечению перевозок
АО «Международный аэропорт «Казань»
Васина А.А.
2019 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ
на приобретение сертифицированных
противообледенительных жидкостей
АО «Международный аэропорт «Казань»

Разработчик:
Александр Н.Кирюшин
Начальник службы наземного
обеспечения перевозок

Согласовано:
Александр В.Агафонов
Начальник производственно-
диспетчерской службы аэропорта

стр.1 из 3

Рисунок 2.1.1 — Пример первой сканированной странице в документе.

По описанной причине, текст первой страницы анализируется отдельно от остальных страниц PDF документов (Рисунок 2.1.2). Причем, нет необходимости в анализе всех страниц в документе на корректность, достаточно выбрать первых N страниц.

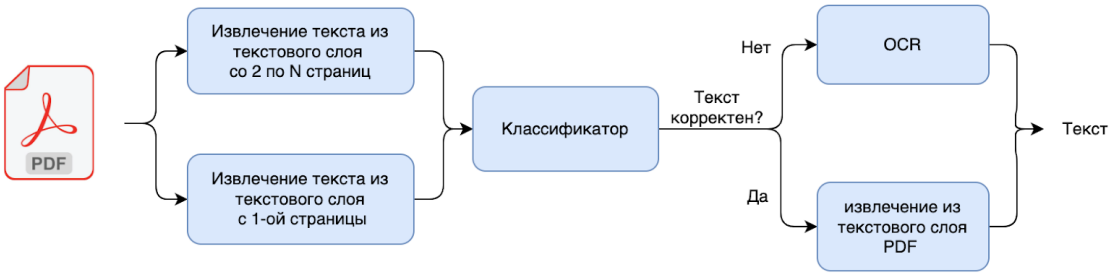


Рисунок 2.1.2 — Метод автоматической обработки текстовых PDF-документов.

2.1.1 Формальная постановка

Формальная постановка задачи выглядит следующим образом. Извлеченный с помощью PDF библиотеки g текст x_N из первых N страниц документа d можно задать как $x_N = g(d)$. Результат бинарного классификатора f тогда задается выражением $f(g(d)) \in [0;1]$, где 0 - корректный, 1 - некорректный текст. Таким образом, описанный метод можно представить следующей формулой:

$$document = \begin{cases} correct & \text{if } f(x_N) = 0 \;\&\; x_N \notin \varnothing \\ incorrect & \text{otherwise} \end{cases}$$

где, $x_N \notin \varnothing$ — это не пустой текст из первых N текстовых страниц документа d.

2.1.2 Текстовые признаки

Обзор работ исследующих корректность текста представлены в разделе 1.2.7.1.

В рамках исследования модели классификации представлено сравнение разных моделей машинного обучения на собранных реальном и синтетическом наборах данных. Сравнение различных моделей классификации проводилось на двух видах входных признаках извлекаемых из текста:

1. Извлечение N-грамм на символах с последующим применением меры TF-IDF;
2. На эвристических правилах, извлекаемых из текста.

Извлечение N-грамм на символах. Согласно работам [54, 57] определение корректности решается путем вычисления N-грамм на текстах. N-грамма — это последовательность из N элементов. В качестве эксперимента берутся униграммы, биграммы и триграммы. Поскольку извлекаемый текст из PDF может быть результатом неправильной кодировки (пример на Рисунке 1.2.7.1) и представлять из себя тарабарщину (gibberish text), где нарушается частота встречи букв в тексте согласно языкам, то целесообразно использовать меру TF-IDF над извлеченными N-граммами. Мера TF-IDF вычисляет меру важности N-граммы для документа в условиях текущего рассматриваемого набора документов. Согласно TF-IDF мере, N-граммам назначается вес, измеряемый не частотой появления в документах, а их релевантностью. Структура формулы TF-IDF (2.1.2.3) основывается на формулах (2.1.2.1) и (2.1.2.2).

TF (*term frequency* — частота слова) — отношение числа вхождений некоторого слова к общему числу слов документа. Таким образом, оценивается важность слова *ti* в пределах отдельного документа.

$$tf(t,d) = \frac{n_t}{\sum_k n_k} \quad (2.1.2.1)$$

IDF (inverse document frequency — обратная частота документа) — инверсия частоты, с которой некоторое слово встречается в документах коллекции. Формула измеряет, насколько слово является уникальным во всей коллекции документов. Слова, которые появляются в большинстве документов, имеют низкое IDF, так как они не вносят большой информационной ценности.

$$idf(t,D) = \log \frac{|D|}{|\{d_i \in D_i | t \in d_i\}|} \quad (2.1.2.2)$$

, где $|D|$ — число документов в коллекции; $|\{d_i \in D_i | t \in d_i\}|$ — число документов из коллекции ***D***, в которых встречается ***t***.

$$tf - idf(t,d,D) = tf(t,d) \times idf(t,D) \quad (2.1.2.3)$$

Извлечение признаков по эвристическим правилам. Также, было опробовано использование признаков, полученных по эвристическим правилам. В качестве эвристических правил были взяты признаки описанные в Таблице 2.1.2.1. Наиболее важные для классификатора XGBoost оказались «Частота изменения регистра», «Медиана кода Юникода символов», «Пропорция букв», «Пропорция дозволенных символов», “Средняя длина слова”. Признаки выделяются для русских и английских текстов. Все используемые признаки представлены в Таблице 2.1.2.1.

Таблица 2.1.2.1 — Описание групп эвристических признаков, выделяемых в тексте.

Группа признаков	**Описание группы**
Пропорция букв	Пропорция русских и английских букв (любых регистров) в тексте
Пропорция цифр	Пропорция цифр в тексте
Пропорция специальных символов	Пропорция специальных символов "<>~!@#$%^&*_+-/\"|?.,:;'`= " в тексте

Пропорция скобок	Пропорция символов "{}[]()" в тексте
Пропорция русских букв	Пропорция букв русского алфавита в верхних и нижних регистров
Пропорция английских букв	Пропорция букв англ. алфавита в верхних и нижних регистров
Пропорция каждого символа	Пропорция каждого отдельного символа в группах: буквы, цифры, спец. символы, скобки
Пропорция дозволенных символов	Пропорция символов из группы: буквы, цифры, спец. символы, скобки
Частота изменения регистра	Число изменения регистра между соседними символами поделенные на длину текста
Частота изменения между дозволенными и остальными символами	Количество случаев, только один из соседних символов входит в группу дозволенных символов, деленное на длину текста
Частота изменения букв	Число случаев, когда только один из соседних символов является буквой, деленное на длину текста
Средняя длина слова	Средняя длина слов в тексте
Медиана длины слов	Медиана длины слов в тексте
Пропорция мусорных символов	Пропорция символов с кодом Юникодом меньше чем 32 или между 160 до 879
Число мусорных символов	Количество символов с кодом Юникодом меньше чем 32 или между 160 до 879
Стандартное отклонение кода Юникода символов	Стандартное отклонение кода Юникода символов в тексте
Среднее значение кода Юникода символов	Среднее значение кода Юникода символов в тексте
Медиана кода Юникода символов	Значение медианы кода Юникода символов в тексте

2.1.3 Наборы данных

Для обучения классификатора f использовался синтетический набор данных, который генерировался двумя способами, на основе текстов на русском и английском языках, полученных из Wikipedia:

1) путем декодирования текста заведомо неправильной кодировкой;
2) путем сохранения текста на изображении и его распознаванием OCR методом с заведомо неправильным языком распознавания;

Таким образом, был сгенерирован сбалансированный синтетический набор данных мощностью в 15 тысяч текстов (Рисунок 2.1.3.1).

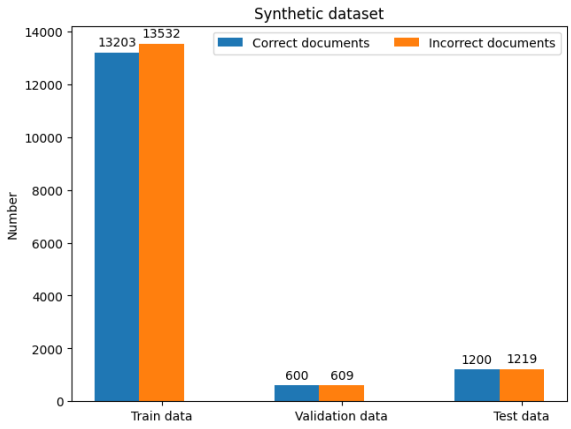


Рисунок 2.1.3.1 — Мощность синтетического сбалансированного набора данных (корректных/некорректных текстов).

Для сравнения моделей классификации был собран несбалансированный тестовый набор реальных данных, состоящий из 257 PDF-документов объемом в среднем 51 страница, из которых 233 документа корректных, а остальные 24 — некорректные PDF. Некорректных PDF-документов меньше, чем корректных, что отражает типичный дисбаланс данных в реальных условиях.

2.1.4 Оценки качества

В задачах бинарной классификации точность часто используется как метрика, однако её эффективность может быть ограничена, особенно при наличии несбалансированных классов. Поэтому важно дополнять точность другими метриками, такими как полнота (recall), точность (precision) и оценка F1. Эти метрики

позволяют получить более полное представление о качестве модели, особенно при работе с несбалансированными наборами данных.

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$F1 = \frac{precision \cdot recall}{precision + recall}$$

В формулах выше TP (True Positive) — это количество правильного текста, идентифицированного как правильное, FP (False Positive) — это количество неправильного текста, идентифицированного как правильное. FN (False Negative) — это количество правильного текста, идентифицированного как неправильное.

Причем на несбалансированной выборке при вычислении итоговых Precision, Recall, F1 учитывается дисбаланс классов, вычисляя среднее значение двоичных показателей, в которых оценка каждого класса взвешивается по его присутствию в истинной выборке данных.

2.1.5 Выбор модели классификации для определения корректности текста

Результаты сравнения моделей классификации на двух видах признаков на синтетическом наборе данных представлено в Таблице 2.1.5.1. Для вероятностных моделей MLE (метод максимального правдоподобия) [58], Logistic Regression (логистическая регрессия) порог классификации подбирался на валидационной выборке синтетических данных.

Время работы, представленное в Таблице 2.1.5.1, включает как вычисление признаков классификации, так и время расчёта модели. Согласно результатам из Таблицы 2.1.5.1, модели демонстрируют схожую высокую точность, но по времени наибольшую скорость обработки показывают Logistic Regression и MLE.

Затем метрики были получены на реальных данных. Результаты представлены в Таблице 2.1.5.2. Метрики Precision, Recall и F1 в Таблице 2.1.5.2 взвешены с учетом дисбаланса реальных данных.

Согласно результатам Таблицы 2.1.5.2 на реальных собранных данных, наиболее эффективными (по качеству, времени) являются методы XGBoost на

эвристических признаках, MLE-оценка на биграммах. При этом, самой точным классификатором является модель RuBert, но ее вычисления требуют значительного времени.

Таблица 2.1.5.1 — Результат работы моделей классификации на тестовом синтетическом наборе данных текстов (на тестовой выборке).

Модель	**Признаки**	**F1-score**	**Time (ms)**
Logistic Regression	TF-IDF	0.990	**1.221**
	Custom features	0.810	1.452
Random Forest	TF-IDF	0.998	9.593
	Custom features	0.998	10.322
XGBoost	TF-IDF	0.998	7.104
	Custom features	0.998	6.276
RuBert	BERT Features	**1.000**	82.213
MLE оценка со сглаживанием Лапласа	Unigram	0.985	14.846
	Bigram	0.991	15.031
	Trigram	0.980	15.200

Таблица 2.1.5.2 — Результат моделей классификации на тестовом наборе реальных текстов.

Models	**Features**	**Precision**	**Recall**	**F1**
Logistic Regression	TF-IDF	0.922	0.868	0.886
	Custom features	0.897	0.739	0.791
Random Forest	TF-IDF	0.923	0.872	0.889
	Custom features	0.957	0.953	0.955
XGBoost	TF-IDF	0.916	0.837	0.863
	Custom features	0.961	0.961	0.961
RuBert	BERT Features	**0.970**	**0.965**	**0.966**
MLE оценка со сглаживанием Лапласа	Unigram	0.949	0.946	0.935
	Bigram	0.952	0.953	0.952
	Trigram	0.955	0.957	0.955

Итоговое решение по выбору модели классификации. В качестве итоговой модели было решено выбрать классификатор XGBoost на эвристических признаках, поскольку результат прогноза дерева решений на специально разработанных признаках более интерпретируем по сравнению с вероятностными моделями. Согласно замерам на реальных данных модель XGBoost по качеству (Таблица 2.1.5.2) и времени вычисления (Таблица 2.1.5.1) выигрывает. Также следует учитывать, что эффективность языковых моделей (MLE оценка для N-граммных моделей), может варьироваться в зависимости от таких факторов, как размер текста и выбранный порог на валидационной выборке, что может повлиять на их точность на текстах, отличных от тренировочных.

Для извлечения текстового содержимого из PDF с корректным текстовым слоем использовалась внешняя библиотека tabby [59]. Для извлечения текста из PDF без/с некорректным текстовым слоем использовался построенное решение для обработки изображений сканированных документов.

2.1.6 Оценка разработанного метода автоматического определения корректности текстового слоя

Метод был успешно интегрирован в разработанный программный комплекс, описание которого представлено в главе 3.

Существует два основных способа извлечения текстовой информации из файла PDF: использование технологии OCR и непосредственное чтение его текстового слоя с помощью обработчиков PDF. Предложенный метод автоматически определяет наличие и корректность текстового слоя и решает как обрабатывать PDF-файл.

Чтобы оценить влияние классификатора на общее качество извлечения текста, был собран двухстраничный набор данных[9], включающий 30 корректных PDF-документов и 30 некорректных. К каждому документу был подготовлен (размечен) вручную текст, который изображен на PDF-документе. Таким образом, можно оценить извлеченный текст с тем, что есть в разметке по метрике Character Accuracy (Формула 2.1.6).

[9] https://at.ispras.ru/owncloud/index.php/s/IvVzvoqZsj5PMpM

Каждый документ был обработан в программном комплексе тремя режимами обработки:

1. Обработка PDF как изображений сканированных документов (т.е. только применение методов обработки изображений с OCR). Обработка описывается блоком 2 на Рисунке 2.1;
2. Обработка только с использованием внешней библиотеки PDF, которая работает с текстовым слоем. Обработка описывается блоком 3 на Рисунке 2.1;
3. Обработка с использованием разработанного метода автоматического определения корректности текстового слоя и выбором одного из двух первых режимов обработки документа. Обработка отражается блоками 1, 2, 3 Рисунка 2.1.

Набор данных реальных PDF-документов описан в разделе 2.1.3.

Таблица 2.1.6.1 - Таблица среднего времени обработки одной страницы PDF в разработанном программном комплексе Dedoc.

Набор данных	Время извлечения содержимого с использованием OCR (обработка режимом 1)	Время извлечения содержимого с использованием разработанного метода (обработка режимом 3)	Время извлечения содержимого с использованием PDF-библиотеки текстового слоя (обработка режимом 2)
Двухстраничный набор данных	3.336 сек/стр.	2.665 сек/стр.	1.260 сек/стр.
Набор данных реальных документов	3.374 сек/стр.	0.616 сек/стр.	0.167 сек/стр.

Таблица 2.1.6.2 - Точность извлеченного текста для трех режимов обработки PDF-документов в разработанном программном комплексе Dedoc на двухстраничном наборе данных.

Character Accuracy (Mean Levenshtein ratio) с использованием OCR (обработка режимом 1)	Character Accuracy (Mean Levenshtein ratio) с использованием разработанного метода (обработка режимом 3)	Character Accuracy (Mean Levenshtein ratio) с использованием PDF-библиотеки (обработка режимом 2)	Точность бинарной классификации (метрика Accuracy)
0.906	**0.939**	0.589	0.9

Результаты замеров представлены в Таблицах 2.1.6.1 и 2.1.6.2. В Таблице 2.1.6.1 представлено время выполнения каждого режима обработки на реальном наборе данных, состоящем в среднем из 51 страницы. В Таблице 2.1.6.2 проведен замер качества извлечения текста на размеченном наборе двухстраничных размеченных PDF-документах. Режим обработки 3 (с разработанным методом) показал:

- сокращение времени обработки в 5.47 раз по сравнению с режимом 1 (с применением OCR);
- увеличение на 3,3% выше качество извлечение текста по сравнению с режимом 1 и на 35% по сравнению с режимом 2 по метрике Character Accuracy (Формула 2.1.6).

Точность извлекаемого текста (Character Accuracy) в программном комплексе замеряется с использованием расстояния Левенштейна для каждой страницы:

$$CharacterAccuracy = \frac{n - E}{n} \quad (2.1.6)$$

, где n - число символов на странице, E - число ошибок распознавания текста на странице (количество операций редактирования удаления/вставки по Левенштейну).

Сравнение с другими открытыми системами. В Таблице 2.1.6.3 приведено сравнение результатов обработки 30 некорректных PDF-документов из размеченного набора двухстраничных документов на некоторых открытых системах, представленных в Таблице 1.4.1. Сравнение проводилось с системами, которые могут обрабатывать как структурированные, так и неструктурированные форматы документов. Согласно Таблице 1.4.1, такими системами являются Unstructured и docling. Как видно из Таблицы 2.1.6.3, рассмотренные открытые системы не обрабатывают некорректные PDF-документы.

Таблица 2.1.6.3 - Точность извлеченного текста разными открытыми системами для некорректных PDF документов с размеченным текстом из набора двухстраничных документов.

Открытые системы для автоматической обработки PDF документов	Среднее значение Character Accuracy (Mean Levenshtein ratio)
Разработанный программный комплекс (dedoc v2.3.2)	**0.914**
docling (version 2.15.1)	0.097
Unstructured (version 0.16.15)	0.08

2.2 Построение методов для автоматического извлечения содержимого из изображений сканированных текстовых документов

Согласно проведенному обзору методов (разделы 1.2.3-1.2.6) обработки изображений сканированных документов в рамках диссертационной работы было решено собрать последовательность методов, позволяющую извлекать текст и таблицы из изображений документов, которые соответствуют рассматриваемой предметной области (раздел 1.1.2).

Обработка состоит из следующих этапов:

Этап 1) Предобработка страницы документа, состоящая из исправления малого угла наклона страницы и исправления ориентации страницы документа, с учетом того, что страница может быть повернута на 0, 90, 180, 270 градусов.

Этап 2) Обнаружение и распознавание табличной информации. В рассматриваемых документах таблицы представлены с границами.

Этап 3) Обнаружение количества колонок и применением технологии оптического распознавания символов.

2.2.1 Методы предобработки страницы документа

2.2.1.1 Метод исправления малого угла наклона страницы

Во время сканирования страницы документа в итоговом изображении документ может быть повернут на небольшой угол. Метод исправления малого угла поворота исправлялся согласно методу [14], описанному в разделе 1.2.3.1.

2.2.1.2 Метод определения ориентации и колоночности страницы

Исправление ориентации изображений текстовых документов можно разделить на две части: определение ориентации изображения текстового документа и поворот изображения этого документа на соответствующий угол, чтобы финальная ориентация была 0° (далее будем называть эту ориентацию вертикальной). В данном случае, предполагается, что изображение изначально может иметь одну из четырех возможных ориентаций: 0°, 90°, 180° или 270°. Пример представлен на Рисунке 2.2.1.1.

Обнаружение количества колонок текста на странице документа тоже можно с помощью задачи бинарной классификации: одна колонка или много колонок.

Рисунок 2.2.1.1 – Пример изображения в четырех возможных ориентациях. Слева направо: 0°, 90°,180°, 270°.

Задачи определения ориентации и количества колонок текста на изображении документа можно объединить, т.е. построить выход модели таким образом, чтобы она за один проход (forward propagation) вычисляла класс для количества колонок и класс для ориентации. Таким образом, построенная модель за один проход решает две задачи.

Листинг 2.2.1.1 – Выходной линейный слой модели классификации с использованием библиотеки pytorch.

```
// init model step
"""
    first 2 classes mean columns number
    last 4 classes mean orientation
"""
num_classes = len([1, 2, 0, 90, 180, 270])
self.net = models.efficientnet_b0(pretrained=model_path is None)
self.net.classifier[1] = nn.Linear(in_features=1280, out_features=num_classes)

...

// predict step

outputs = self.net(tensor_image)
columns_out, orientation_out = outputs[:, :2], outputs[:, 2:]
_, columns_predicted = torch.max(columns_out, 1)
_, orientation_predicted = torch.max(orientation_out, 1)
```

Выход сети представлен в Листинге 2.2.1.1 и состоит:

- Первые два значения выхода сети резервируются для определения количества столбцов текста: одноколоночный или многоколоночный текст;
- Следующие 4 значения выхода сети резервируются для решения задачи обнаружения ориентации страницы документа на классы: 0°, 90°, 180°, 270°.

Модель классификации. В качестве модели классификации была выбрана архитектура EfficientNet B0, которая является одной из эффективных архитектур глубоких нейронных сетей, разработанная для решения задач компьютерного зрения, таких как классификация изображений. Главной идеей этой модели является повышение эффективности архитектуры (точность, время выполнения) путем подбора оптимального масштабирования трех составляющих: количества слоев, размеров слоев и глубины слоев архитектуры нейронной сети. Преимущества такой архитектуры следующие:

- Масштабирование модели может помочь уменьшить склонность к переобучению. Большие модели могут быть более подвержены переобучению из-за большого числа параметров, особенно если у вас ограниченный объем обучающих данных. Меньшие модели, более адаптированные к размеру обучающих данных, могут быть менее склонны к переобучению.
- Масштабирование позволяет уменьшить размер некоторых слоёв без явной потери качества. Меньшие модели имеют меньше параметров, что может ускорить процесс обучения и саму работу модели.

На Рисунке 2.2.1.2 представлено схематичное изображение архитектуры модели EfficientNet B0. Conv — обозначение свёрточного слоя, после которого указан размер ядра свёртки, Pooling — обозначение слоя max pooling, FC — обозначение полносвязного слоя. Основным строительным блоком данной модели является MBConv, структура которого напоминает Residual-блок, используемый в архитектурах ResNet. Сначала количество каналов увеличивается с помощью свертки 1x1, при этом цифра в названии блока указывает на то, во сколько раз увеличивается количество каналов. Затем используется depth-wise свертка (для каждого канала) с ядром, размер которого указан в соответствующей ячейке на схеме. После используется свертка 1x1 для уменьшения количества каналов, чтобы начало и конец блока можно было сложить.

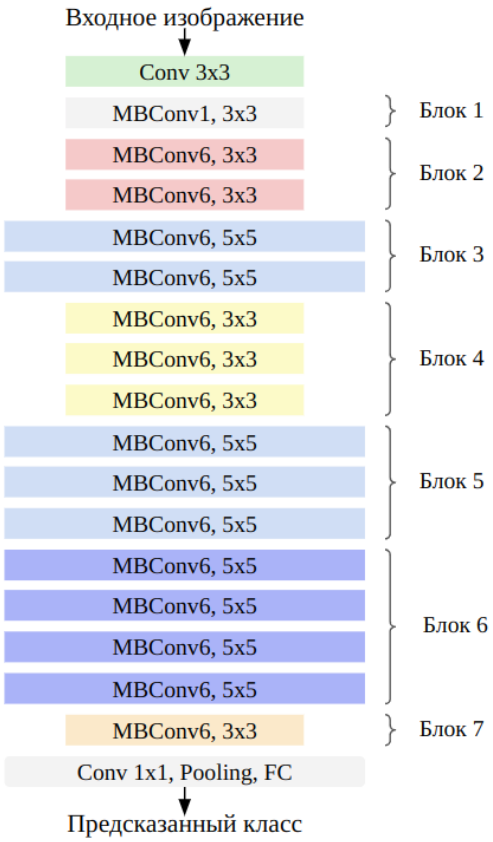


Рисунок 2.2.1.2 – Схематическое представление архитектуры EfficientNet B0.

Набор данных. Для обучения и тестирования данной модели был сформирован набор данных, насчитывающий около 2.5 тысяч вертикальных изображений документов разных типов: статьи, технические задания, положения, законы. Значение метки класса “количество текстовых колонок” на изображении устанавливалось вручную. Набор данных включает:

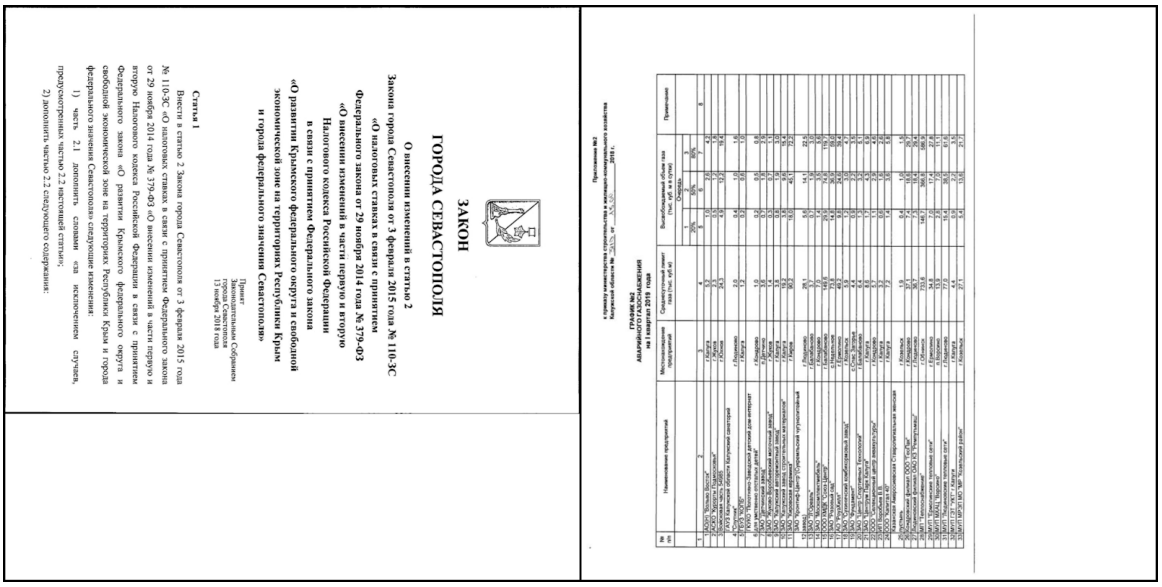
- Научные статьи, в которых могут присутствовать: двухколоночный текст, таблицы, графики и иллюстрации. Документы в большинстве случаев имеют книжную ориентацию. Текст преимущественно на английском языке. Таблицы, иллюстрации и графики могут распространяться как в центре документа, так и в одной из колонок. Текст разбит на абзацы. Кроме самого текста в статьях встречаются формулы и численные расчеты. Статьи также имеют оглавления на первой странице, ориентированные по центру.
- Документы типа технического задания, ведомости, постановления, законов имеют менее строгие, но схожие черты. Эти документы имеют частично альбомную, частично книжную ориентацию и обычно состоят из одной колонки. Обычно на титульном листе этих документов имеется заголовок,

расположенные по центру, и печать. Также разного рода печати и подписи встречаются в других местах этих документов. Кроме текста, в этих документах также встречаются таблицы с разным количеством строк и столбцов. При этом документы, таблицы занимают около трети от общего объема документов подобного типа (пример документов представлены на Рисунке 1.1.2.1).

Описанные выше изображения дальше поворачивались на углы, кратные 90° и сохранялись как отдельные копии. То есть документ, который раньше имел вертикальную ориентацию, в наборе данных был представлен в ориентациях 0°, 90°, 180°, 270°. В качестве функции поворота использовалась функция, основанная на отражении и транспонировании матрицы поворота, реализованная в библиотеке Numpy. Этот поворот изображения был применен таким образом, чтобы все направления ориентации документа встречались в наборе данных одинаковое количество раз. Таким образом из 2,5 тысяч изображений было получено 10 тысяч. Обучающая выборка составила 8 тысяч изображений, тестовая – 2 тысячи. Документы для генерации набора данных доступны по ссылке[10].

Перед подачей в модель изображения проходят предварительную обработку: сначала растягиваются так, чтобы большая сторона имела длину 1200 пикселей, затем справа или снизу (в зависимости от ориентации изображения) добавляются белые пиксели до квадратного размера 1200 × 1200 пикселей. Это необходимо, поскольку классификатор принимает только квадратные изображения, и важно сохранить всю информацию на изображении. Примеры итоговых изображений представлены на Рисунке 2.2.1.3. После этого изображения переводятся в тензор и проходят нормализацию.

[10] https://huggingface.co/datasets/dedoc/orientation_columns_dataset



ЗАКОН

ГОРОДА СЕВАСТОПОЛЯ

О внесении изменений в статью 2

Закона города Севастополя от 3 февраля 2015 года № 110-ЗС

«О налоговых ставках в связи с принятием

Федерального закона от 29 ноября 2014 года № 379-ФЗ

«О внесении изменений в часть первую и вторую

Налогового кодекса Российской Федерации

в связи с принятием Федерального закона

«О развитии Крымского федерального округа и свободной

экономической зоне на территориях Республики Крым

и города федерального значения Севастополя»

Принят
Законодательным Собранием
города Севастополя
13 ноября 2019 года

Статья 1

Внести в статью 2 Закона города Севастополя от 3 февраля 2015 года № 110-ЗС «О налоговых ставках в связи с принятием Федерального закона от 29 ноября 2014 года № 379-ФЗ «О внесении изменений в часть первую и вторую Налогового кодекса Российской Федерации в связи с принятием Федерального закона «О развитии Крымского федерального округа и свободной экономической зоне на территориях Республики Крым и города федерального значения Севастополя» следующие изменения:

1) часть 2.1 дополнить словами «за исключением случаев, предусмотренных частью 2.2 настоящей статьи»;

2) дополнить частью 2.2 следующего содержания:

Рисунок 2.2.1.3 – Пример двух изображений, прошедших предобработку перед подачей в классификатор ориентации.

Обучение и точность классификации. Обучение проводилось с помощью библиотеки Pytorch. В качестве функции потерь использовалась перекрёстная энтропия, в качестве оптимизатора использовался стохастический градиентный спуск. Шаг обучения был равен 0.001, обучение длилось 75 эпох с размером батча 16. Для ускорения вычислений во время обучения была использована видеокарта Tesla A100 с 40 Гб видеопамяти. Само время обучения составило примерно восемь часов. На тестовом наборе данных после обучения модель показала метрику F1-score = 0.996. Модель насчитывает около 5.3 миллионов параметров и обрабатывает изображение одного документа примерно за 20 миллисекунд. Файл с весами обученной модели, которые загружаются при первом запуске классификатора, весит 16.4 МБ.

Точность классификатора на сгенерированном тестовом наборе данных представлена в Таблице 2.2.1.1.

Таблица 2.2.1.1 – Точность модели на сгенерированном тестовом наборе данных классификации ориентации и колоночности текстовых сканированных документов.

Набор данных	**Количество изображений документов**	**mean F1-мера**

Сгенерированный набор данных	3300	0.999

2.2.2 Метод обнаружения и распознавания табличной информации

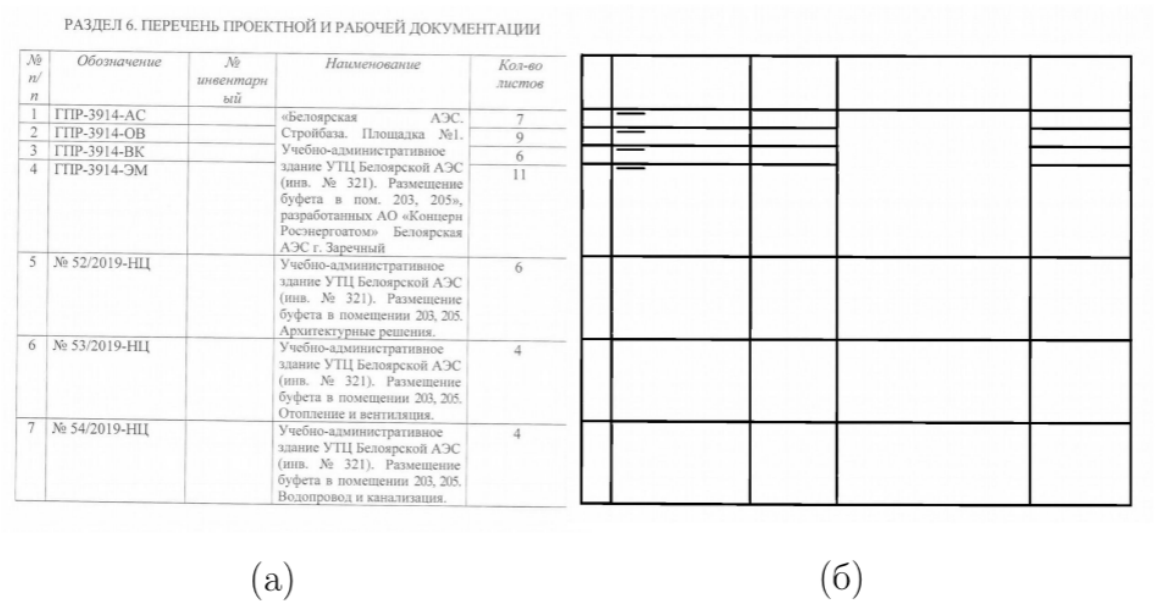
Поскольку отчеты, технические задания и правовые документы содержат в основном таблицы с границами, для точного обнаружения таких таблиц наилучшим образом подходит метод контурного анализа с применением набора эвристических правил для определения границ таблиц и внутренних ячеек. Далее будет представлено описание данного метода. Все стадии процесса детектирования и распознавания таблиц показаны на Рисунке 2.2.2.1.

Рисунок 2.2.2.1 – Процесс распознавания таблиц на изображениях документов.

Этапы метода.

1 Этап. Бинаризация изображения по порогу, порог подбирается экспериментально.

2 Этап. Определение вертикальных и горизонтальных линий на изображении. Изображение подвергается двум последовательным морфологическим операциям сужения и расширения (“erode” и “dilate”), позволяющим выделить элементы описанные заданным ядром и отфильтровать неэквивалентные ядру объекты. В качестве ядер выбираются прямые вертикальные и горизонтальные линии, задающие каркас таблицы. В начале последовательно применяются две морфологические операции “erode” и две операции “dilate” с нормализованным ядром [1, ширина_изображения / 55] для обнаружения горизонтальных линий. Затем аналогичным образом выделяются вертикальные линии с ядром [высота_изображения / 100, 1]. Два изображения с выделенными вертикальными и горизонтальными линиями суммируются в одно итоговое с одинаковыми весами вхождения (0.5 и 0.5).



РАЗДЕЛ 6. ПЕРЕЧЕНЬ ПРОЕКТНОЙ И РАБОЧЕЙ ДОКУМЕНТАЦИИ

№ п/п	Обозначение	№ инвентарный	Наименование	Кол-во листов
1	ГПР-3914-АС		«Белоярская АЭС. Стройбаза. Площадка №1. Учебно-административное здание УТЦ Белоярской АЭС (инв. № 321). Размещение буфета в пом. 203, 205», разработанных АО «Концерн Росэнергоатом» Белоярская АЭС г. Заречный	7
2	ГПР-3914-ОВ			9
3	ГПР-3914-ВК			6
4	ГПР-3914-ЭМ			11
5	№ 52/2019-НЦ		Учебно-административное здание УТЦ Белоярской АЭС (инв. № 321). Размещение буфета в помещении 203, 205. Архитектурные решения.	6
6	№ 53/2019-НЦ		Учебно-административное здание УТЦ Белоярской АЭС (инв. № 321). Размещение буфета в помещении 203, 205. Отопление и вентиляция.	4
7	№ 54/2019-НЦ		Учебно-административное здание УТЦ Белоярской АЭС (инв. № 321). Размещение буфета в помещении 203, 205. Водопровод и канализация.	4

(а) (б)

Рисунок 2.2.2.2 – Пример работы этапа детектирования линий модуля распознавания таблиц. (а) – исходное изображение, (б) – маска с объединенными детектированными горизонтальными и вертикальными линиями.

3 Этап. Определение малого угла наклона таблицы. На данном этапе алгоритма устанавливается априорная информация – все таблицы на изображении могут быть повернуты на малый угол из-за погрешности угла наклона страницы в сканере. Отсюда страница может быть повернута на малый угол поворота (примерно до 5 градусов). Такое искажение часто встречается на сканированных документах, поэтому на данном этапе необходимо обнаружить такое искажение. Для определения угла поворота изображения, вычисляются линии Хафа с помощью функции `HoughLinesP` из библиотеки OpenCV на изображении с детектированными линиями. В алгоритме проводится поиск небольшого угла не превышающий 5 градусов. Угол не может быть точно идентифицирован при анализе исходного изображения с текстовой информацией, которая дает погрешность в 2-3 градуса. По этой причине было принято проводить анализ угла искажения на изображениях с выделенными границами, вычисленных на втором этапе. Таким образом, снижается погрешность искомого угла поворота меньше чем высота строк в тексте. Найденные линии Хафа подвергаются дополнительной фильтрации с целью устранения выбросов, в результате на изображении фильтруются линии с отклонением от горизонта более 5 градусов.

4 Этап. Изображение с выделенными горизонтальными и вертикальными линиями и исходное изображение поворачивается на найденный угол с помощью операции аффинного преобразования.

5 Этап. На данном этапе применяется контурный анализ с помощью функции `findContours` библиотеки OpenCV. Результатом функции является сгруппированные элементы в многоуровневую иерархию (дерево). Каждый элемент иерархии задается кортежем из четырех индексов:

- индекс следующего контура на текущем слое;
- индекс предыдущего контура на текущем слое;
- индекс первого контура на дочернем слое;
- индекс родительского контура.

На основе полученной иерархии контуров, можно выделить контур таблицы и контуры ячеек (расположение таблицы и ячеек).

2.2.2.1 Постобработка результатов контурного анализа

Полученное дерево контуров в предыдущем разделе можно представить в виде древовидной структуры, представленной на Рисунке 2.2.2.3.

Обход дерева ограничивается длиной 2. Контуры второго уровня (ячейки) сортируются по координатам x, y верхнего левого угла контура и записываются в матричное представление таблицы. При обнаружении таблиц использовались следующие эвристики:

- Таблица должна состоять минимум из двух столбцов;
- Высота ячейки >= 10;
- Ширина ячейки >= 20;

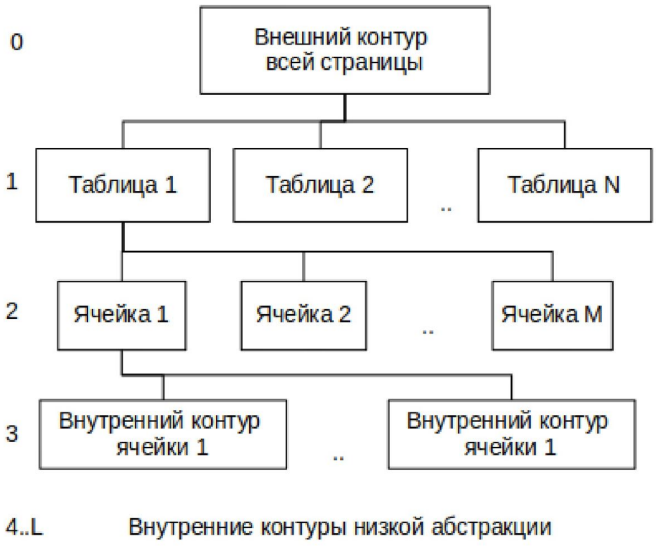


Рисунок 2.2.2.3 – Иерархия, полученная в результате контурного анализа изображения сканированного документа.

2.2.2.2 Вычисление содержимого ячеек

На следующем этапе каждое изображение ячейки контура подается на вход Tesseract OCR для распознавания содержимого ячейки в текстовом виде. Признак пустых ячеек качественнее находить не через Tesseract, а путем выделения границ на изображении с использованием фильтра Канни (Canny 1986). Если процент черных пикселей на отфильтрованном изображении меньше порогового значения, то ячейка помечается как пустая, иначе ее изображение подается на вход Tesseract OCR. Пороговое значение в методе подбиралось эмпирическим путем.

2.2.2.3 Разделение совмещенных ячеек

Модуль работает с таблицами, в которых могут присутствовать объединенные ячейки как по вертикали, так и по горизонтали. На основе этого, координаты ячеек в методе сравниваются между собой и крупные ячейки дробятся на мелкие с сохранением признаков разделения по вертикали ”v” или по горизонтали ”h”.

Признак дробления вертикально объединенной текущей ячейки ‘v’ (cell_current ячейки) устанавливается при соблюдении следующих условий:

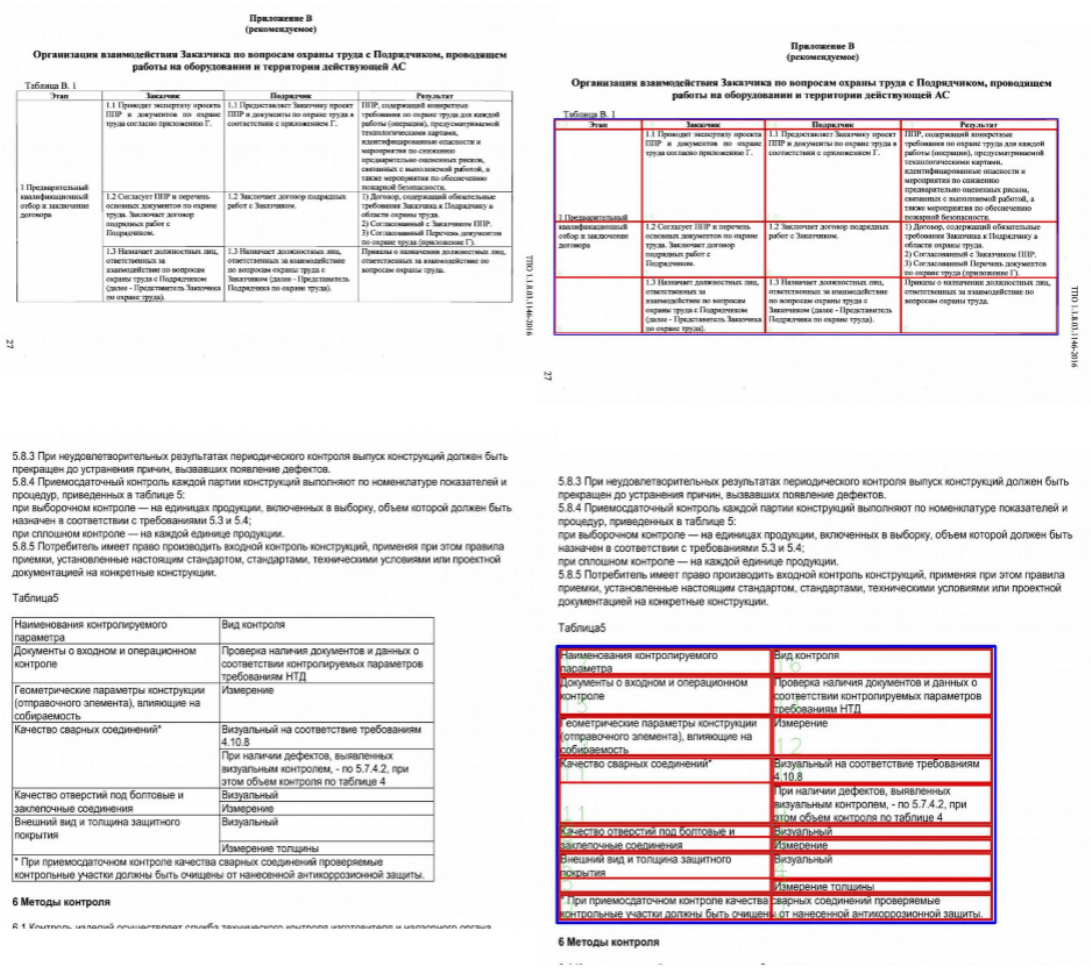
- текущая ячейка cell_current.y == cell_small.y в диапазоне +-5 пикселей,

- высота cell_current.height >= cell_small.height + min_height_of_cell.

Устанавливается признак дробления горизонтально объединенной текущей ячейки ‘h’ при соблюдении следующих условиях:

- текущая ячейка cell_current.x == cell_small.x в диапазоне +-5 пикселей;
- ширина cell_current.width >= cell_small.width + min_width_of_cell.

При разделении крупной ячейки ее значение (текстовая информация) передается новым разделенным.



Приложение В
(рекомендуемое)

Организация взаимодействия Заказчика по вопросам охраны труда с Подрядчиком, проводящем работы на оборудовании и территории действующей АС

Таблица В. 1

Этап	Заказчик	Подрядчик	Результат
1 Предварительный квалификационный отбор и заключение договора	1.1 Проводит экспертизу проекта ППР и документов по охране труда согласно приложению Г.	1.1 Предоставляет Заказчику проект ППР и документы по охране труда в соответствии с приложением Г.	ППР, содержащий конкретные требования по охране труда для каждой работы (операции), предусматриваемой технологическими картами, идентифицированные опасности и мероприятия по снижению предварительно оцененных рисков, связанных с выполняемой работой, а также мероприятия по обеспечению пожарной безопасности.
	1.2 Согласует ППР и перечень основных документов по охране труда. Заключает договор подрядных работ с Подрядчиком.	1.2 Заключает договор подрядных работ с Заказчиком.	1) Договор, содержащий обязательные требования Заказчика к Подрядчику в области охраны труда. 2) Согласованный с Заказчиком ППР. 3) Согласованный Перечень документов по охране труда (приложение Г).
	1.3 Назначает должностных лиц, ответственных за взаимодействие по вопросам охраны труда с Подрядчиком (далее - Представитель Заказчика по охране труда).	1.3 Назначает должностных лиц, ответственных за взаимодействие по вопросам охраны труда с Заказчиком (далее - Представитель Подрядчика по охране труда).	Приказы о назначении должностных лиц, ответственных за взаимодействие по вопросам охраны труда.

27

ТПО 1.1.03.1146-2016

5.8.3 При неудовлетворительных результатах периодического контроля выпуск конструкций должен быть прекращен до устранения причин, вызвавших появление дефектов.
5.8.4 Приемосдаточный контроль каждой партии конструкций выполняют по номенклатуре показателей и процедур, приведенных в таблице 5:
при выборочном контроле — на единицах продукции, включенных в выборку, объем которой должен быть назначен в соответствии с требованиями 5.3 и 5.4;
при сплошном контроле — на каждой единице продукции.
5.8.5 Потребитель имеет право производить входной контроль конструкций, применяя при этом правила приемки, установленные настоящим стандартом, стандартами, техническими условиями или проектной документацией на конкретные конструкции.

Таблица5

Наименования контролируемого параметра	Вид контроля
Документы о входном и операционном контроле	Проверка наличия документов и данных о соответствии контролируемых параметров требованиям НТД
Геометрические параметры конструкции (отправочного элемента), влияющие на собираемость	Измерение
Качество сварных соединений*	Визуальный на соответствие требованиям 4.10.8
	При наличии дефектов, выявленных визуальным контролем, - по 5.7.4.2, при этом объем контроля по таблице 4
Качество отверстий под болтовые и заклепочные соединения	Визуальный
	Измерение
Внешний вид и толщина защитного покрытия	Визуальный
	Измерение толщины

* При приемосдаточном контроле качества сварных соединений проверяемые контрольные участки должны быть очищены от наносимой антикоррозионной защиты.

6 Методы контроля

Рисунок 2.2.2.4 – Результаты распознавания каркаса таблицы на изображении сканированных документов.

2.2.2.4 Выделение атрибутных ячеек

Атрибутные ячейки в таблице задают ее заголовок. Вычисление признака, что ячейка является атрибутной (частью заголовка) или простой, можно путем получения логической структуры таблицы из анализа признаков объединенных ячеек. Логическая структура таблицы описывает иерархическое отношение ячеек друг с другом.

Физическая структура таблицы описывает физическое расположение ячеек (их координат в пространстве изображения). Табличное представление имеет много интерпретаций. В методе рассматривается наиболее распространенный случай таблиц, где атрибутные (заголовочные) ячейки располагаются сверху таблицы, а первый столбец носит более информативный характер, по сравнению с другими.

В данном разделе будет представлено описание правил, согласно которым искались атрибутные (заголовочные) ячейки в таблицах. Будет рассмотрено 4 типа анализа заголовка в таблице.

- В простой таблице первая строка является атрибутной, поскольку отсутствуют какие-либо объединения в таблице согласно Рисунку 2.2.2.5 (а).
- Анализ заголовка для вертикально разделенных ячеек. В данном случае рассматривается первый столбец. За заголовок принимается столько строк таблицы сколько первых ячеек с признаком объединения по вертикали ‘v’ в первом столбце см. Рисунок 2.2.2.5 (б).
- Анализ заголовка для горизонтально объединенных ячеек. Здесь производится перебор строк от начала таблицы до момента, когда на текущей строке таблице будут отсутствовать признаки разделения по горизонтали ‘v’. В результате, заголовок таблицы будет содержать верхние строки, в которых присутствуют разделенные по горизонтали ячейки, включая одну строку ниже, см. Рисунок 2.2.2.5 (в).
- Существуют заголовки с объединенными ячейками по горизонтали и вертикали. В данном случае, поиск заголовка производится по 3-му способу. Пример шаблона представлен на Рисунке 2.2.2.5 (г).

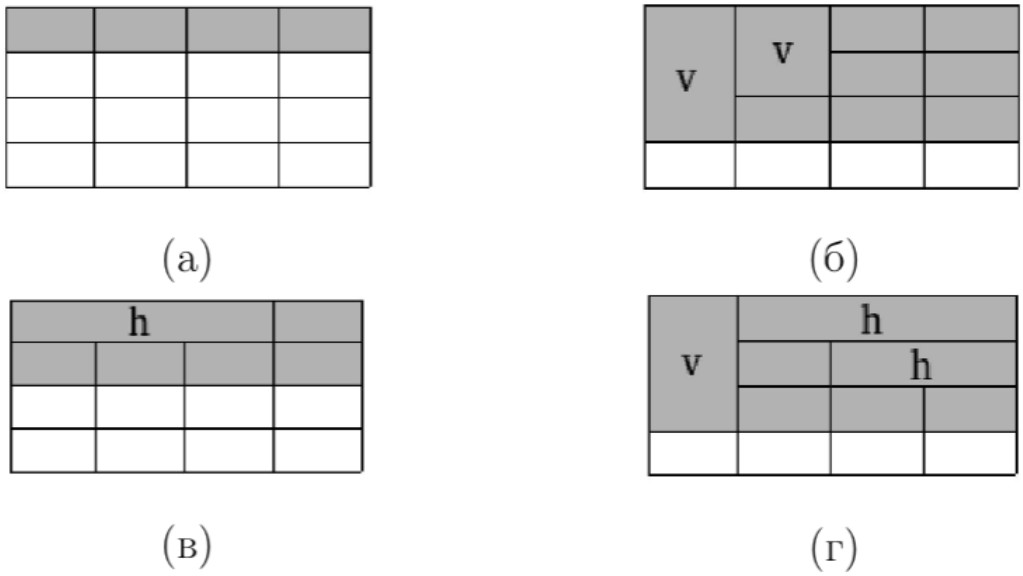


Рисунок 2.2.2.5 – Виды обрабатываемых шаблонов заголовков таблиц.

2.2.2.5 Анализ многостраничных таблиц

Поиск многостраничных таблиц производится после обнаружения одностраничных таблиц из всего документа. На данном этапе проводится обход всех одностраничных таблиц с проверкой эвристических правил, согласно которым можно утверждать, мы имеем дело с одностраничной таблицей или многостраничной.

Ниже представлены признаки присущие одностраничным таблицам-кандидатам table1 и table2, которые можно объединить в одну многостраничную таблицу:

- Таблицы-кандидаты table1 и table2 должны располагаться на соседних страницах документа;
- Первая часть многостраничной таблицы table1 должна быть последней таблицей на странице;
- Последующая часть многостраничной таблицы table2 должна быть первой на странице;
- Количество столбцов таблиц-кандидатов должно совпадать;
- Ширина таблиц-кандидатов должна совпадать с погрешностью 3% от ширины первой таблицы-кандидата;

- Ширины всех столбцов таблиц-кандидатов должны совпадать с погрешностью в 10% от ширины столбца первой таблицы-кандидата;
- Таблиц-кандидатов для одной многостраничной таблицы не может превышать количество страниц в анализируемом документе;
- В случае совпадения заголовков у таблиц-кандидатов, сохраняется заголовок только у первой таблицы-кандидата, а у остальных устраняется. Сравнение заголовков основан на сравнении содержимого их ячеек. Сравнение содержимого производится нестрого, через вычисление расстояний Левенштейна.

Погрешности в правилах выбирались эмпирическим путем. В случае невыполнения хотя бы одного условия таблицы-кандидаты нельзя считать как одной многостраничной таблицей.

2.2.2.6 Оценка качества распознавания таблиц

Для оценки метода использовалась метрика TEDS, описанная в разделе 1.2.6 формулой 1.2.6.1, которая оценивает как распознанный в ячейках текст, так и на сколько правильно была распознана структура таблицы в виде иерархического дерева ячеек. Качество измерялось на собранном и размеченном наборе таблиц из документов “Технические задание”, “ВКР” и “Нормативно-правовые акты”. Набор данных доступен по ссылке[11]. Также метод измерялся на таблицах сгенерированных с помощью работы Zhong [42]. Сгенерированные данные доступны по ссылке[12].

Таблица 2.2.2.1 – Точность извлечения таблиц с явными границами.

Набор данных	Кол-во таблиц	Метрика TEDS
Сгенерированные таблицы 1-ой категории Zhong et al. (2020) [42]	500	0.90
Черно-белые сканированные документы	31	0.97

[11] https://at.ispras.ru/owncloud/index.php/s/nYgwbhVk5SpvD3z/download
[12] https://at.ispras.ru/owncloud/index.php/s/gItWxupnF2pve6B/download

SEC	Test pale-	MTN	Sample ASSETS,
and	Defense	k working	water
plant	Unfavorable	8,961	.1
	Engineering b		137.9
Volcanogenic	Soda NA	3,735	44
COMBAT	Calcite	94	1.1
ND CHnv	Hydril p.	1992	0.110
atm overpack	mm Ford	3	1
in Rancho		64	71
ACTIVITY MANAGEMENT	Operating	375	1919.9
and	HER p.	5	0.003
changes Diffusion	pH	23	1.3
	Hematite	0.02	11
Profit	Edge before	45.06	3.3
earnings	welded	258	0.44
Cells Item	arroyo Net	5	2,446

per	x Jr.	and Rail	to Value	RIB	three Outline	all. Record	properties, Gaylussite
use	adverse sale	Stratigraphic	Smec- costs	New	CRASHES	DECEMBER	other
Capitalization	2,727.8	11,606	gal/min		5	for Unexercised	nuclear Iron
and Instant	335.5	212	40	Operations of	11	USED	is Bikini
ended Increase	97	1.1	performance;	tunnels	.20	AND Organic	Plan Years
Cumulative	2,357	203	$3995	Phenocrysts, m	62,	Unnamed change	
Disney,	45	150	(a)	Comfort	10	PRNTAR	TAUDRY
Ratios	1,856.1			Purpose meters	4	stock of	REVISION
miles	4.03	0.3	5812e02	depth share	0.0	is	Main Elements
U.S	10	5	(870)	acquisitions from	63,278	VAN Borehole	and Atmosphere
Calico Other	450	34,162	(in)	BE	40	pumice, WorkmansidP	ignites
Total coefficient	1142.9	43	<10	PINOCCHIO km	0.21	Nonhydrocarbons	Normal,
case MONTH	0.000	12	c241	capability tuff,	0.001	from SYSTEM	Lyons, Income
Fire SE	0.010	19	7000e01	x	4.	nuclear	TBD Cash
operator of	60.3	129	1±1	Information Sales	1002	Mil	to

Рисунок 2.2.2.6 – Примеры сгенерированных таблиц по методу [42].

2.2.3 Извлечение текстовой информации

2.2.3.1 Описание метода распознавания Tesseract OCR

Tesseract OCR работает пошагово согласно блок-схеме, показанной на Рисунке 2.2.3.1 [100]. Первым шагом выполняется адаптивная бинаризация, которая преобразует изображение в бинарное. Следующим шагом выполняется анализ связанных компонентов [99], который используется для получения контуров символов. Затем контуры символов объединяются в объекты-символы. Большие объекты-символы организуются в текстовые строки, а строки анализируются на предмет некоторой текстовой области (блока текста) [99]. Далее текст разбивается на

слова с использованием определенных и нечетких пробелов [99]. В конечном итоге двухпроходное распознавание слов осуществляется с использованием методов кластеризации и классификации. Для окончательного решения о распознанном слове Tesseract OCR обращается как к языковому словарю, так и к пользовательскому словарю. В качестве вывода предоставляется слово с наименьшим расстоянием. Более подробную информацию о каждом этапе можно найти в статье авторов [99].

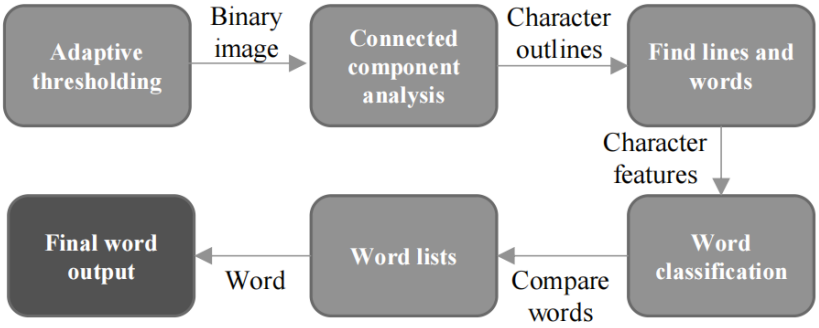


Рисунок 2.2.3.1 – Архитектура движка Tesseract OCR [100].

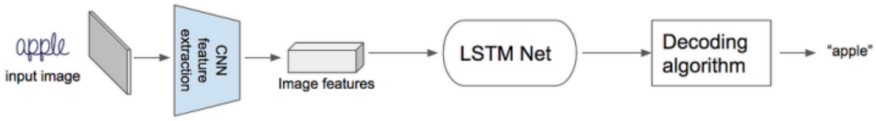


Рисунок 2.2.3.2 – Общая архитектура сети распознавания текста из изображения на основе LSTM.

На странице системы Tesseract OCR [102] разработчики указывают, что в качестве модели распознавания символов в Tesseract используется модель LSTM (Long short-term memory). Эта модель искусственного интеллекта берёт начало в реализации LSTM на базе Python из проекта OCRopus [27], но была полностью переработана для Tesseract на C++ [103].

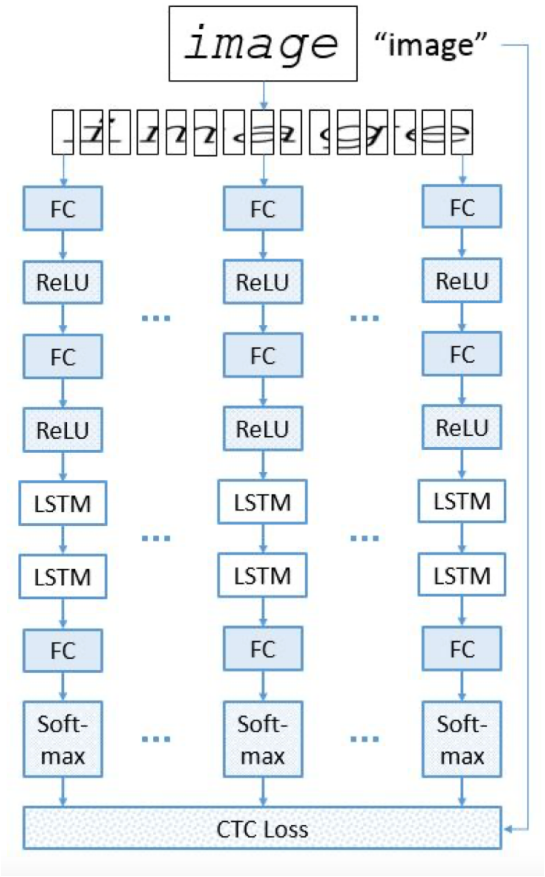


Рисунок 2.2.3.3 – Обзор всей архитектуры. Слой извлечения признаков принимает часть входного изображения в качестве данных, а LSTM-слои функционируют как слои распознавания [101].

В задачах на основе анализа последовательностей, хорошо зарекомендовали себя рекуррентные нейронные сети. Особенность использования рекуррентных сетей состоит в способности запоминать длинные последовательности входных данных. Использование данных сетей хорошо применяются для задачи распознавания текста на изображениях.

Далее рассмотрим пример архитектуры LSTM, как разновидность рекуррентной сети в задаче оптического распознавания текста:

1) Первым этапом архитектуры является извлечение высокоуровневых признаков изображения (Рисунок 2.2.3.2). Данные признаки можно получить с помощью сверточных нейронных сетей или полносвязных слоев (Fully Connected (FC)). Например, в работе [101] сеть принимает кадры изображения, полученные с помощью скользящего окна шириной 2 пикселя, преобразованные в вектор 60x1, в качестве этапа извлечения признаков используется 2 полносвязных слоев по 60 единиц каждый (Рисунок 2.2.3.2).

2) Вторым компонентом является сеть с долговременной и кратковременной памятью LSTM, являющаяся разновидностью рекуррентной нейронной сети. Блок LSTM состоит из набора пороговых функций, которые позволяют запоминать представления как на длинные, так и на короткие промежутки времени. Особенностью поведения блоков LSTM является то, что они устойчивы к проблеме с долгосрочной памятью. Т.е. в блоках нейроны не имеют возможности надолго сохранить информацию, которая была обработана много итераций назад. Подробное описание блока LSTM приведено в [105]. Блок LSTM содержит 3 или 4 вентиля (логические функции), которые выдают значения от 0 до 1. Умножение на это значение позволяет частично ограничить входную информацию в LSTM-блок или наоборот сдержать выход. «Входной вентиль» контролирует меру вхождения нового значения в память, а «вентиль забывания» контролирует меру сохранения значения в памяти. «Выходной вентиль» контролирует меру того, в какой степени значение, находящееся в памяти, используется при расчёте выходной функции активации для блока. Ниже (Рисунок 2.2.3.4) представлен LSTM-блок с тремя вентилями с описанием вентилей:

$$
\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

, где

x_t - входной вектор,

h_t - выходной вектор,

c_t - вектор состояний,

W, U и b - матрицы параметров и вектор байеса (сдвига),

f_t - вектор вентиля забывания, вес запоминания старой информации,

i_t - вектор входного вентиля, вес получения новой информации,

o_t - вектор выходного вентиля, вес выходной информации из LSTM-блока.

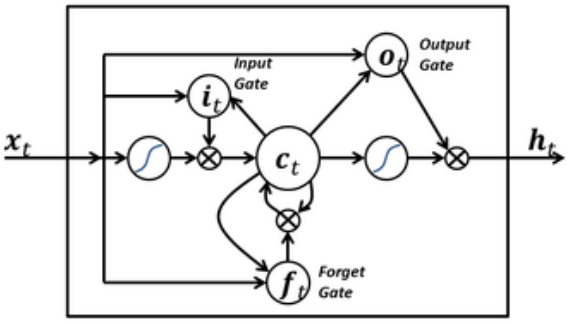


Рисунок 2.2.3.4 – Простой LSTM-блок с тремя вентилями: входным, выходным и забывания [106].

3) Важным и последним компонентом представленной нейронной сети является коннекционистская временная классификация (CTC-Connectionist Temporal Classification), представленной Грейвсом [104]. Данная классификация позволяет избавиться от аннотирования каждой позиции буквы, для данной функции классификации достаточно иметь только изображение слова и метку – текстовое представление слова, без позиций каждой буквы в слове.

2.2.3.2 Использование Tesseract OCR

Ниже описаны настраиваемые параметры Tesseract OCR в разработанном программном комплексе:

- В Tesseract OCR необходимо правильно указать язык распознавания. Согласно нашей предметной области пользователь может выбрать язык русский, английский, либо мультиязычный русский+английский. Начиная с версии Tesseract OCR 5.0.0 хорошо распознает и мультиязычный язык. По умолчанию установлен язык русский+английский. Данная настройка задается пользователем с помощью api-параметра ‘language’.
- В Tesseract OCR минимально анализируется макет изображения документа в процессе распознавания текста, поэтому для достижения наилучших результатов работы Tesseract OCR, стоит знать макет входного текстового документа. В Tesseract есть параметр psm, настраиваемый для алгоритма связных компонент. Описание параметра psm представлено в Листинге 2.1.

В программном комплексе используется 3 режима сегментирования страницы Tesseract OCR (Листинг 2.2.3.1) для разных случаев:

- для многоколоночных документов со сложным макетом страницы подходит режим анализа макета страницы psm=3 (режим по умолчанию);
- для простых страниц с одной колонкой текста подходит режим анализа макета документа psm=4;
- для распознавания текста из ячеек таблиц подходит режим анализа макета страницы psm=6, поскольку мы считаем, что внутри ячейки таблицы нет сложной структуры и текст представлен простым текстовым блоком.

- Tesseract OCR возвращает уверенность распознанного текста, поэтому можно дополнительно фильтровать текст с низкой уверенностью от -1.0 до 100.0. Значение -1.0 возвращается в случае, если нет уверенности распознавания и от 0.0 до 100.0 иначе. В библиотеке установлены следующие пороги уверенности:
  a) для таблиц порог уверенности установлен в 1.0;
  b) для распознавания документов порог установлен в 40.0.

Листинг 2.2.3.1 – Инструкции режимов сегментирования страниц Tesseract OCR.

```
Page segmentation modes:
  0    Orientation and script detection (OSD) only.
  1    Automatic page segmentation with OSD.
  2    Automatic page segmentation, but no OSD, or OCR. (not implemented)
  3    Fully automatic page segmentation, but no OSD. (Default)
  4    Assume a single column of text of variable sizes.
  5    Assume a single uniform block of vertically aligned text.
  6    Assume a single uniform block of text.
  7    Treat the image as a single text line.
  8    Treat the image as a single word.
  9    Treat the image as a single word in a circle.
  10   Treat the image as a single character.
  11   Sparse text. Find as much text as possible in no particular order.
  12   Sparse text with OSD.
  13   Raw line. Treat the image as a single text line,
       bypassing hacks that are Tesseract-specific.
```

2.2.4 Оценка качества методов обработки изображений

Для оценки качества распознавания была создана разметка к 83 изображениям документов. Точность извлекаемого текста (character accuracy) замеряется с помощью расстояния Левенштейна для каждой страницы [98]:

$$CharacterAccuracy = \frac{n - E}{n}$$

, где n – число символов на странице, E – число ошибок, сделанных на странице (количество операций редактирования удаления/вставки по методу вычисления расстояния Левенштейна).

Таблица 2.2.4.1 – Точность извлекаемого текста с помощью последовательности методов обработки сканированных документов.

Набор данных	**Количество изображений документов**	**Character Accuracy**[13]
Черно-белые сканированные изображения документов	83	97.541

2.3 Метод восстановления иерархической структуры

Для второго этапа двухэтапной обработки текстовых документов был разработан метод автоматического извлечения логической структуры из содержимого с форматированием, полученного на предыдущем этапе.

2.3.1 Структура документа

Структура документа – это иерархическое представление структурных частей документа так, чтобы имелась приоритетность частей документа и возможность навигации по нему.

В разработанном методе выделяются два типа структурных элементов:

- *На уровне заголовков.* Здесь восстанавливается иерархия заголовков в документе. Примером иерархии заголовков документа является “Оглавление”.
- *На уровне текстовых блоков.* Здесь восстанавливается иерархия текстового содержимого внутри секций документа. Сюда относятся разделение на

[13] S. V. Rice. *Measuring the Accuracy of Page-Reading Systems*. Ph.D. dissertation,41 University of Nevada, Las Vegas, 1996

параграфы и списки. Приоритеты таких структурных элементов ниже заголовочных.

Содержимое многих документов можно представить в виде дерева. Одним из представлений древовидной структуры является оглавление документа. В документах оглавление задает удобный способ поиска информации в документе, перечисление заголовков и их иерархии. Оглавление позволяет читателю быстрее понять содержимое и смысл документа. Представление документа в виде дерева так же как и оглавление важно для анализа документа. Иерархическое представление содержимого документа предоставляет:

- Понимание приоритетов содержимого документа, а именно понимание того, какая часть документа содержит наиболее важную информацию.
- Возможность удобного поиска информации в документе. Таким образом, нет необходимости считывать весь документ полностью, достаточно провести анализ заголовков.
- Возможность структурирования документа и его хранение в наиболее удобном виде для пользователя, отображать документ структурировано.
- Возможность более точно сравнивать документы между собой. Таким образом, можно определить полноту документа, определить соответствует ли рассматриваемый документ заданным критериям и стандартам.
- Автоматический анализ содержимого документов. Например, с помощью методов анализа текста извлекать необходимые факты из заголовков документа или других значимых его частей.

Древовидная структура документа достаточно универсальна. Независимо от того, какую предметную область документа мы рассматриваем – техническое задание или нормативно-правовой акт, содержимое можно представить в виде иерархии. Например, в нормативно-правовом акте следующие значимые части составляют иерархию: Глава → подглава → пункт → подпункт и так далее.

Стоит отметить структурирование документов, представляющих собой формы (например, счета на оплату, квитанции и так далее). Даже в таких документах можно выделить иерархию в виде заголовка документа и его сегментов, например, заголовок – “Номер счета на оплату”, его дочерними узлами тогда будут – “информация о продавце”, “информация о покупателе”, “информация о товаре” и так далее. Таким образом, даже из документа, представляющего форму или бланк, можно выделить иерархию и проанализировать содержимое автоматически.

Таким образом, каждый узел дерева будет представлять собой значимый структурный элемент содержимого документа. Причем значимость структурного элемента тем выше, чем она ближе к корню дерева. По данному правилу, корневой узел дерева представляет собой заголовок всего документа, его подузлы представляют собой главные разделы документа и так далее. Листья дерева представляют собой структурные элементы, которые невозможно дальше разбить на меньшие составляющие. К листьям относятся параграф или простой элемент списка (пример на Рисунке 2.3.1.1).

Рисунок 2.3.1.1 – Представление документа в виде дерева на примере жилищного кодекса.

Причем в каждой предметной области есть свое правила построения иерархии дерева документа. В технических заданиях – это разделы, подразделы, списки, в законах это - глава → подглава → пункт → подпункт. Каждый структурный элемент имеет свою иерархию. При наличии знаний об иерархии мы можем построить иерархическую структуру, а именно дерево документа.

2.3.2 Формализация структуры документа согласно его предметной области

В соответствии с поставленной задачей, необходимо формализовать структурные элементы иерархической структуры, извлекаемых из Технических заданий, ВКР и Нормативно-правовых актов.

На основе изученных предметных областей документов можно выделить следующие структурные элементы:

- Титульный лист находится в начале документа и содержит название, подписи и даты. Вместо титульного листа в документах может присутствовать только название на первой странице.
- Содержание, которое может располагаться после титульного листа. Содержание не всегда может присутствовать в документах. Содержание встречается в документах типа “Техническое задание” и “ВКР”. В документах типа нормативно-правовой акт, содержания нет.
- Базовые структурные элементы (разделы, подразделы и т.д.), которые визуально выделяются и делят документ на крупные составные части. Структурные элементы составляют содержимое документа и вкладываются друг в друга.
- Элементы списков, которые не отличаются визуально от основного текста, но структурируют текст с помощью нумерации.
- Текстовые строки (или параграфы).

Рисунок 2.3.2.1 - Пример структуры технического задания.

В результате изучения структуры документов было решено представлять в виде дерева произвольной глубины, в котором узлы упорядочены, имеют один из пяти описанных выше типов и представляются следующим образом:

- Корень дерева содержит название документа, наиболее выделяющееся визуально. Остальные части названия являются дочерними узлами корня и располагаются в порядке чтения.
- Заголовок содержания (если содержание присутствует) является дочерним узлом названия-корня. Содержимое содержания является деревом с корнем в заголовке содержания. Иерархическая структура содержания соответствует структуре этого дерева.
- Структурные элементы (главы/подглавы/секции) являются дочерними узлами названия-корня (в порядке чтения). Если один из структурных элементов непосредственно вложен в другой (например, подраздел вложен в раздел), то он является дочерним узлом структурного элемента, в который он вкладывается.
- Элементы списков могут быть вложенными в структурные элементы, текстовые параграфы (если перед перечислением находится его описание с двоеточием в конце) или другие элементы списков. В соответствии с этим определяется узел-предок элемента списка. Все элементы одного списка являются дочерними узлами одного предка.
- Простые текстовые параграфы могут вкладываться в структурные элементы или элементы списков. Таким определяется узел-предок для текстового параграфа.

На Рисунке 2.3.2.1 изображен пример структуры технического задания, которое можно извлечь из документа.

Таким образом, необходимо автоматизировать восстановление описанной выше древовидной структуры. Структура не является строго определённой, поэтому лучше использовать методы машинного обучения.

2.3.3 Описание метода восстановления иерархической структуры

В силу недостаточной точности следования шаблонам составления документов из конкретной предметной области, для решения задачи восстановления структуры

было решено в основу взять классификацию текстовых строк документа. Для классификации строк используются методы машинного обучения. Это позволит обойтись без написания жестких правил для восстановления структуры, а также промоделировать интеллектуальную деятельность человека в процессе разделения структурированного документа на части. На Рисунке 2.3.3.1 продемонстрирована схема этапов восстановления структуры из текстового документа.

Рисунок 2.3.3.1 – Обработка текстовых строк документа относительно его формата и предметной области для извлечения иерархической структуры.

Метод восстановления иерархической структуры состоит из следующих шагов:

- Извлечение строк ‘Оглавления’. Данный шаг применяется только для документов с наличием оглавления, представленного в начале документа. К таким документам относятся ВКР (Диплом / магистерская диссертация), технические задания, документы набора данных соревнования FINTOC2022.
- Получение машиночитаемых численных свойств текста (матрицы признаков) на основе текста и его визуальных свойств в виде матрицы признаков. Описание входных признаков приведено в разделе 2.3.5.
- Обучение ММО (модели машинного обучения) (классификатора строк) на полученной матрице признаков, используемой в качестве входных данных модели. Классификатор строк предсказывает класс строки документа согласно

конкретному типу структуры (согласно предметной области, раздел 2.3.2). Структура документа определяет список типов (классов) строк, на которые необходимо классифицировать строки документа. Каждый тип/класс строки определяется своим приоритетом в предметной области. Перечень классов и их приоритет задается заранее разработчиком. Например приоритет класса “Глава” нужно задать выше приоритета класса “Подглава” или класса “Элемент списка” для документа типа “НПА”;

- Постобработка результатов классификатора. На данном этапе уточняются классы некоторых структурных элементов. Это позволяет в случае изменения приоритетов структурных элементов, не размечать заново весь набор данных, а изменить код постобработки. Описание постобработки описано в разделе 2.3.6.
- Построение иерархического дерева документа. Поскольку нам известны на данном этапе классы текстовых строк с их предустановленными приоритетами, то можно определить положение (глубину в дереве документа) одной строки относительно других строк в иерархической структуре документа.

Класс каждой текстовой строки (или параграфа) вычисляется по ряду признаков. В методе выделяется три группы признаков:

- *Не-относительные признаки форматирования.* В эту группу относятся признаки форматирования, не вычисляемые относительно других строк с форматированием: полужирный текст, курсив, зачеркнутый текст, подчеркнутый текст, цвет текста;
- *Относительные признаки форматирования.* Значения признаков вычисляются для всех строк документа, далее вычисляется медиана значений, и значением признака принимается величина строки относительно медианы. К таким относительным признакам относятся: отступ строки слева (расстояние), межстрочное расстояние, размер шрифта;
- *Текстовые признаки строки.* К данной группе относятся признаки на основе регулярных выражений, а именно булевые признаки срабатывания регулярных выражений.

Некоторые из строк можно определить автоматически с помощью правил (Например, определить строку, содержащую номер страницы), либо с помощью мета-информации, предоставляемой форматом (сноски, но не всегда). Модели классификации подаются на вход только те строки, тип которых нельзя определить

простым и очевидным способом (Например заголовочные строки отличаются в некоторых документах просто жирностью, в других заглавными буквами, в других тип шрифта или его цвет заголовочных строк может быть отличен от обычного текста). Отсюда восстановление структуры только на основе правил требует досконального анализа предметной области документа и самостоятельного выявления правил, по которым документы в данной предметной области строятся. Это требует хороших навыков и трудозатрат разработчиков таких систем, и исключает возможности автоматизации процесса.

Выбор модели. В качестве модели классификации строк выбрана модель градиентного бустинга над решающими деревьями XGBoost. Данная модель является классической для выявления зависимостей в разнокотегариальных признаках. Причем, согласно обзору области многие работы [81, 83, 89, 93, 95] показывают хорошие результаты с использованием решающих деревьев. Модели глубокого обучения, т. е. нейронные сети различных архитектур, могут проигрывать модели градиентного бустинга по времени работы и по качеству результатов в силу разнородности признаков, подаваемых на вход модели.

Градиентный бустинг – это техника машинного обучения для задач классификации и регрессии, которая строит модель предсказания в форме ансамбля слабых предсказывающих моделей, обычно деревьев решений.

Дерево решений – это бинарное дерево, в котором:

- каждой внутренней вершине v приписан предикат $B_v : X \to 0,1$, где X – пространство признаков объектов выборки;
- каждой листовой вершине v приписан прогноз $c_v \in Y$, где Y — область значений целевой переменной (в случае классификации, листу может быть также приписан вектор вероятностей классов).

В ходе предсказания осуществляется проход по дереву к некоторому листу. Для каждого объекта выборки x движение начинается из корня. В очередной внутренней вершине v проход продолжится вправо, если $B_v(x) = 1$, и влево, если $B_v(x) = 0$,. Проход продолжается до момента, пока не будет достигнут некоторый лист, и ответом алгоритма на объекте x считается прогноз c_v, приписанный этому листу. Предикат B_v может иметь, вообще говоря, произвольную структуру, но, как правило, на практике используют сравнение с порогом $t \in \mathbb{R}$ по j-му признаку: $B_v(x, j, t) = [x_j \leq t]$. При проходе через узел дерева с данным предикатом объекты будут отправлены в правое

поддерево, если значение j-го признака у них меньше либо равно t и в левое – если больше.

Бустинг над решающими деревьями – это один из видов композиции слабых моделей, при котором предсказания начальных моделей композиции уточняются предсказаниями последующих. При обучении такого ансамбля моделей на каждой итерации вычисляются отклонения предсказаний уже обученного ансамбля на обучающей выборке. Следующая модель, которая будет добавлена в композицию будет предсказывать эти отклонения. Таким образом, при добавлении предсказания нового дерева к предсказаниям обученного ансамбля можно уменьшить среднее отклонение модели от верного ответа.

Обучающий набор данных. Для документов каждого из представленных типов был подготовлен обучающий набор данных. Типы строк определялись с помощью ручной разметки несколькими экспертами. Набор документов типа “НПА” был составлен из документов, загруженных с официального сайта pravo.gov.ru, набор документов типа “Техническое задание” – с официального сайта zakupki.gov.ru, набор документов типа “ВКР” был получен от высшего учебного заведения РАНХиГС. В результате разметки были сформированы наборы данных, основная информация о которых представлена в Таблице 2.3.6.1. В технических заданиях и выпускных работах также присутствуют списки на уровне текстовых блоков. Собранные наборы содержат размеченные документы в форматах DOCX, PDF, TXT. Разметка данных заключалась в задании класса для текстовых строк документов и выполнялась с помощью внешней системы разметки.

Формат PDF в отличии от других структурированных форматов DOCX, HTML не содержит разметку по параграфам (текстовым блокам). Для того, чтобы текст внутри каждой секции был разбит на параграфы, для обработки PDF-документов был разработан бинарный классификатор текстовых строк (строка 4 в Таблице 2.3.6.1) для определения является ли текстовая строка началом нового параграфа в документе или нет. В качестве классификационной модели используется используется тот же классификатор строк (XGboost).

2.3.4 Матрица признаков для классификации строк

Для документов разных предметных областей можно формировать конкретный набор числовых признаков, который поможет осуществить классификацию наилучшим образом. Тем не менее существует некоторый общий набор признаков, на основе которого можно делать конкретизацию и добавление новых признаков:

- *Индикаторы соответствия регулярным выражением для начала строки*. Регулярные выражения описывают следующие шаблоны строк: нумерованные и маркированные списки различных типов; строки, начинающиеся с предопределенных ключевых слов.
- *Индикаторы соответствия регулярным выражением для конца строки*. Регулярные выражения описывают шаблоны строк, заканчивающихся на различные знаки препинания (точку, запятую, двоеточие, точку с запятой), буквы или цифры.
- *Визуальные признаки для строки.* К таким признакам относятся переведенные в числовые значения метаданные текста строки: индикатор жирности, курсива и подчеркивания текста и процент таких символов в строке; отступ от левого края страницы и расстояние до предыдущей строки; наиболее часто встречающийся размер шрифта в строке. В некоторых случаях может также использоваться цвет текста - численное значение каждой цветовой компоненты (красный, зеленый, синий) и его дисперсия.
- *Статистика по буквам, словам и строкам.* К данной группе признаков относится следующая информация о строке: процент букв, заглавных букв, цифр и скобок в строке; число слов в строке; номер страницы, номер строки в документе и длина строки в символах.
- *Индикатор того, что в документе есть оглавление.* Оглавление документа ищется с помощью набора правил и поиска соответствия регулярным выражениям. Оглавление может повысить качество нахождения заголовков в документе.
- *Индикатор того, что строка является частью оглавления* и индикатор того, что в оглавлении есть ссылка на данную строку, то есть данная строка фигурирует в оглавлении.

- *Признаки глубины вложенности строк*: уровень вложенности списков с точкой (например, 1.1.1), относительный размер шрифта и отступ от левого края.
- *Контекстные признаки*: вышеперечисленные признаки для 3 предыдущих и 3 последующих строк. Количество соседних строк можно менять, было выбрано число 3, так как оно позволило получить наилучшие результаты. Информация о других строках позволяет модели классифицировать текущую строку, опираясь на дополнительную информацию о её соседях.

Некоторые признаки, значение которых должно быть относительным для каждого документа (например, номер строки в документе, размер шрифта, размер отступа слева), проводится нормализация значений по следующей формуле:

$$xnorm_i = \frac{x_i - mean(x)}{max(x) - min(x)} \text{ если } max(x) \neq min(x), \text{ иначе } xnorm_i = 0,$$

где x – вектор признаков, x_i – i-я компонента вектора признаков, $xnorm$ – вектор признаков с нормализованными значениями.

Таким образом, для документа, состоящего из n строк, в результате извлечения признаков получается матрица с n строками и m столбцами, где m – число признаков, полученных для каждой строки. Совокупность таких матриц вместе с метками классов для строк может использоваться для обучения модели классификации строк документа.

2.3.5 Постобработка результатов классификации

Поскольку строки имеют разный приоритет в документе и единичная ошибка в классификации одной строки в заголовке может быть более заметна, чем правильная классификация 100 других менее значимых строк, например, если речь идет об определении заголовков или строк титульного листа. Такие ошибки классификации связаны в первую очередь с тем, что значимых строк в документе значительно меньше, чем обычных текстовых строк и не всегда примеров значимых строк достаточно для того, чтобы модель научилась их выделять. В этом случае применение постобработки к результатам классификации строк может существенно улучшить результаты с точки зрения их восприятия человеком.

Как было сказано ранее, постобработка результатов классификации может существенно улучшить результаты и сделать их более правдоподобными для человека. Помимо исправления результатов модели, на данном шаге можно скорректировать уровень вложенности строки, или ее “важность”, так как это определяется на основе класса, присвоенного строке моделью классификации, соответствующей определенному типу документа.

Рассматриваемые в данной работе типы документов имеют некоторое сходство в своей структуре. Это сходство заключается в глобальных составных частях и порядке их следования:

- У всех таких документов имеется титульный лист, расположенный в начале документа (до начала структурных элементов).
- После строк, относящихся к титульному листу, идет тело документа, первой строкой которого является строка - заголовок (структурный элемент). В программном комплексе этот класс обозначен как named_item.
- Если в документе есть содержание, то оно располагается в начале документа после титульного листа и имеет заголовок “Содержание” или “Оглавление”. Будем считать, что содержание включается в тело документа.

Помимо базовой структуры, каждый из документов имеет ряд отличительных особенностей, характерных для конкретного типа документов. Например, для законов слова “Глава” и “Статья” являются ключевыми и заголовок с названием “Статья” ниже по значимости, чем заголовок с названием “Глава”. На этапе постобработки производится уточнение класса заголовочных структурных элементов (named_item) и назначение им приоритетов. Это удобно при поддержке разработанного классификатора, поскольку со временем может поменяться приоритетность структурных элементах для типа документа и будет достаточно изменить их приоритет в коде постобработки, нежели заново переразмечать тренировочный набор данных. Например приоритет заголовочных строк с точками 1.1.1 выше приоритета строк с римскими цифрами IV вначале строки. Демонстрационный пример показан на Рисунке 2.3.5.1. Приоритеты заголовочных структурных элементов могут поменяться со временем в предметной области документа. В любом случае, написание процесса постобработки - это этап не является обязательным, в случае, если разработчикам программного комплекса захочется добавить поддержку нового типа структуры.

В дипломах и магистерских диссертациях обязательно должен присутствовать заголовок с названием “Введение” и заголовок с названием “Заключение”. Нахождение таких особенностей и их использование в процессе постобработки позволяет восстанавливать структуру с большей точностью.

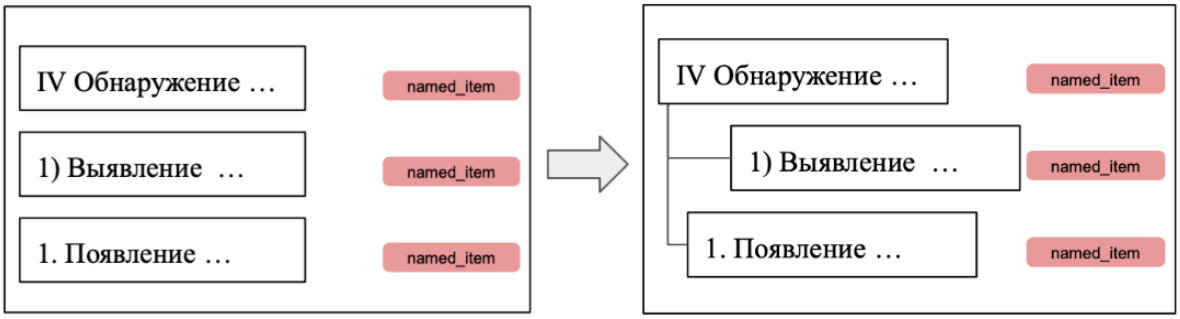


Рисунок 2.3.5.1 – Уточнение класса заголовочного структурного элемента named_item.

Таким образом, на этапе постобработки уточняются классы структурных элементов и выделяются структурные элементы, которые не требуют применения методов машинного обучения (например строки-заголовки “Заключение” и “Введение”).

2.3.6 Оценка качества восстановления структуры

Точность обученных классификаторов строк с постобработкой результатов метода представлена в Таблице 2.3.6.2. Качество классификации строк измерялось с использованием кросс-валидации на 10 итерациях на размеченных данных. Количественная информация о количестве классов в размеченных наборах представлена в Таблице 2.3.6.1.

Таблица 2.3.6.1 – Описание используемых наборов данных документов.

Тип документа	**Количество документов**	**Количество строк**	**Распределение строк по типам (при обучении)**
НПА	349	20471	титульный лист: 2058 заголовок: 4104 приложение: 696 сноска: 266 автор: 261 текст: 13086

Техническое задание	51	8556	титульный лист: 208 заголовок: 892 пункт: 2891 содержание: 677 текст: 3888
ВКР (Диплом / магистерская диссертация)	50	25545	титульный лист: 1159 заголовок: 1074 текст: 23312
Начало параграфа/продолжение параграфа	50	7502	начало параграфа: 2710 продолжение параграфа: 4472

Таблица 2.3.6.2 – Точность классификации строк по типу (предметным областям) документов.

Тип документа	**Точность классификации (accuracy)**	**F1-мера (macro)**
Закон РФ	0.91057	0.81411
Техническое задание	0.88010	0.80836
Диплом / магистерская диссертация	0.95152	0.94894
Начало параграфа/продолжение параграфа	0.91326	0.91491

2.3.6.1 Оценка метода на наборе данных соревнования FINTOC

Точность разработанного метода восстановления иерархической структуры была также оценена (Таблица 2.3.6.3) на наборе данных международного соревнования FINTOC2022 (Рисунок 2.3.6.1), содержащего финансовые документы на английском, испанском и французском языках. Документы набора имеют формат PDF с качественным текстовым слоем. В соревновании решается две задачи:

- Задача 1: обнаружение заголовков в документе;
- Задача 2: определение уровня вложенности обнаруженных заголовков.

Для решения каждой задачи используется двухэтапный подход извлечения содержимого и восстановления структуры, а именно извлечение содержимого с форматированием, извлечение признаков каждой текстовой строки, обнаружение оглавления документа, получение матрицы признаков, предсказание классификации, постобработка.

Для решения первой задачи (Задача 1) классификатор строк вычисляет 2 класса (заголовок/не заголовок), а для второй задачи (Задача 2) классификатором предсказывается уровень вложенности заголовка от 0 до 9.

Поскольку разметка набора FINTOC содержит только структуру из заголовков (нет разметки структуры в рамках одной секции, например списки), то для решения второй задачи участвовали не все текстовые строки, а только заголовочные. Чтобы измерить качество метода восстановления структуры на данном наборе, метод применялся два раза:

- классификатор строк применяется ко всем текстовым строкам для обнаружения заголовков (решается задача бинарной классификации заголовок/незаголовок).
- для найденных заголовков классификатор вычисляет класс для каждого заголовка от 0 до 9 для вычисления глубины заголовка.

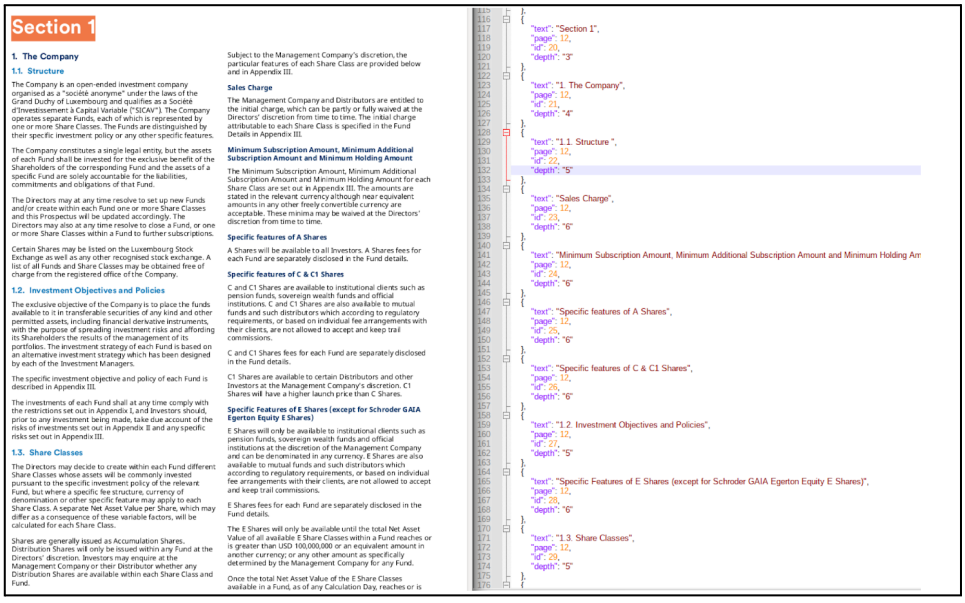


Рисунок 2.3.6.1 – Пример документа FINTOC2022 с примером разметки заголовков документа.

Итоговое сравнение на двух задачах (Задачи “1” и “2”) разработанного метода с другими решениями на наборе данных FINTOC2022 на английском языке

представлено в Таблице 2.3.6.3. Разработанный метод занял первое место на данном наборе [2].

Таблица 2.3.6.3 – Сравнение с другими методами на наборе данных соревнования FINTOC2022 English.

	F1-мера обнаружения заголовка (Задача 1)	**Inex-F1-мера определения уровня заголовка (Задача 2)**	**Level Accuracy точность определения уровня заголовка (Задача 2)**	**Гармоническое среднее F1-меры и точности Level Accuracy определения уровня заголовка (Задача 2)**
Christopher Bourez (2021) [95]	0.830	53.6	30.6	38.95
CILAB [96]	0.738	56.5	27.5	36.99
swapUNIBA [97]	0.793	63.6	42.9	51.23
Разработанный метод	**0.900**	**68.8**	**58.4**	**63.17**

В задачах 1 и 2 правильно обнаруженными заголовками является те, которые совпали по расстоянию Левенштейна превышающим 0.85. Правильно обнаруженным заголовком является:

- для задачи 1: текста заголовков (обнаруженного и в разметке) совпали по расстоянию Левенштейна превышающим 0.85 и имеющими совпадающие номера страниц;
- для задачи 2: текста заголовков (обнаруженного и в разметке) совпали по расстоянию Левенштейна превышающим 0.85 и имеющими совпадающие номера страниц и уровень заголовка (от 0 до 9).

Precision (Точность) – это отношение правильно предсказанных заголовков к общему числу найденных заголовков в документе. *Recall (Полнота)* – это отношение корректно предсказанных заголовков к числу заголовков в разметке. *F1-мера* – гармоническое среднее между Precision и Recall.

Level Accuracy (Точность определения уровня заголовков) второй задачи определяется по формуле:

$$LevelAccuracy = \sum \frac{E'_{ok}}{E_{ok}}$$

, где E_{ok} - это количество предсказанных заголовков с совпадающей страницей из разметки, E'_{ok} - это количество предсказанных заголовков с совпадающей страницей и уровнем вложенности совпадающей с разметкой.

Гармоническое среднее (harmonic mean) определение уровня заголовка вычисляется между F1-мерой и точностью Level Accuracy определения уровня заголовка.

2.4 Выводы ко второй главе

В главе 2 были описаны разработанные методы, позволяющие автоматически извлечь содержимое из неструктурированных и структурированных форматов документов с последующим восстановления иерархической структуры документов. Был описан разработанный метод определения корректности текстового слоя PDF-документов для повышения качества извлечения текста из PDF. Предложенный метод в разработанном программном комплексе показал на 3.3% качество выше и в 5.5 раз быстрее обработку на размеченном наборе данных, содержащих некорректные PDF-документы. Также было проведено сравнение результатов автоматического извлечения текстового содержимого из PDF с некорректным текстовым слоем с другими открытыми системы. Результаты показали, что другие системы, способные обрабатывать структурированные и неструктурированные форматы не проводят автоматический анализ таких документов на корректность и показывают качество извлеченного текста неудовлетворительное.

Также были описаны разработанные методы для извлечения текста и таблиц из изображений сканированных документов для рассматриваемых предметных областей,

которые показывают высокое качество извлечения текста и таблиц с границами. Методы также были включены в разработанный программный комплекс.

Для восстановления иерархической структуры из полученного содержимого документов был разработан метод на основе классификации текстовых строк, полученных на предыдущем этапе. Благодаря методу становится возможным разбить документ на иерархические секции и сформировать иерархическую структуру документа. Разработанный метод был оценен на существующем наборе данных соревнования FINTOC (Таблица 2.3.6.3), где метод показал наилучший результат восстановления структуры по сравнению с другими работами соревнования и занял первое место [2].

Глава 3. Программный комплекс

3.1 Архитектура программного комплекса

Для реализации двухэтапной обработки документов, описанной в главе 2, была спроектирована архитектура, общая схема которой представлена на Рисунке 3.1.1.

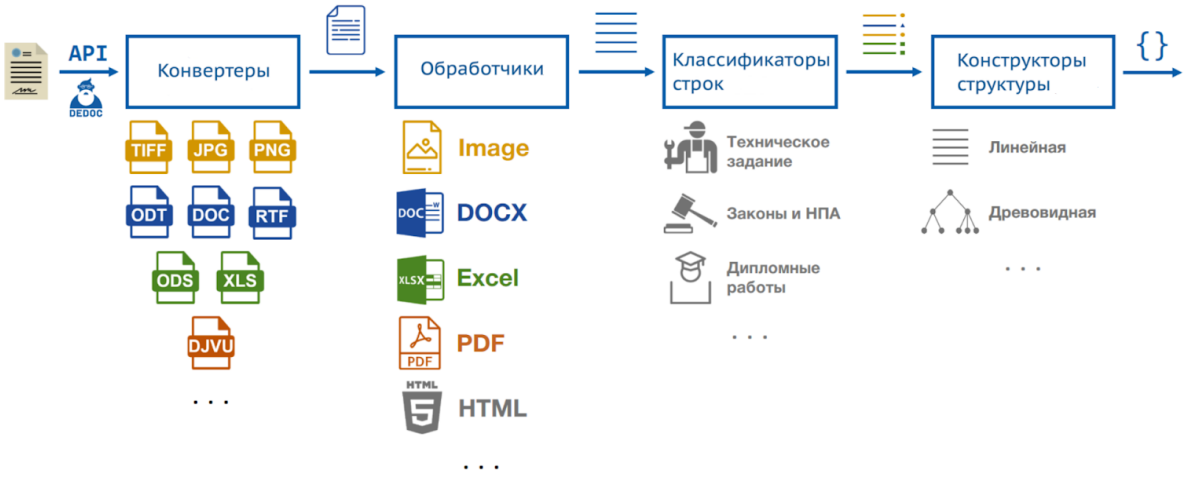


Рисунок 3.1.1 – Общая схема архитектуры программного комплекса.

Согласно этой архитектуре, каждый документ в процессе обработки проходит следующие стадии:

- *Конвертирование* (при необходимости) полученного документа в один из форматов, поддерживаемых библиотекой. Некоторые форматы обрабатываемых файлов схожи друг с другом (например форматы редактора Word odt, docx, doc или растровые изображения png, jpeg, jpg и т.п.) и обработка документов близких форматов производится схожим образом. Таким образом, мы избегаем проблемы создания большого количества обработчиков при поддержка большого перечня форматов документов;
- *Обработка для извлечения содержимого и метаданных (форматирования)* из документа в промежуточное представление. На данном этапе выбирается необходимый обработчик, который извлекает содержимое (текст, таблицы) с форматированием из конкретного формата документа в промежуточное

представление. На данном этапе применяются сторонние библиотеки на усмотрение разработчика;

- *Восстановление* из промежуточного представления документа иерархической *структуры* в зависимости от предметной области документа: определение типа (класса) каждого элемента (строки) и его значимости (уровня вложенности) внутри документа. Тип (предметная область) документа задается пользователем системы при обработке документа в качестве входного параметра;
- *Преобразование* промежуточного представления структурированного документа *в выходной формат* согласно древовидному представлению. Тип выходного формата выбирается пользователем системы, например HTML, json или обычный текст без структуры и дополнительной информации.

Архитектура библиотеки спроектирована таким образом, чтобы можно было ее расширить, а именно:

- Добавить обработку новых форматов документов, путем добавления нового преобразователя формата (обработчика конвертации) или нового обработчика извлечения содержимого (текста и метаданных);
- Добавлять обработку документов, относящихся к новой предметной области (поддержка нового типа структуры документов), путем добавления нового обработчика восстановления структуры (классификатора строк).

Подробнее о расширении описано в методике добавления поддержки нового формата или типа документа в разделе 3.3.

3.2 Основные программные модули

Библиотека состоит из следующих программных модулей, представленных на UML диаграмме компонентов на Рисуноке 3.2.1:

- **API модуль**, позволяющий пользоваться библиотекой посредством POST- запросов.

- Модуль **manager**, отвечающий за управление очередью обработки файлов. В данном модуле происходит объединение разных стадий обработки документа в единый пайплайн обработки. Стадии обработки представлены в разделе “Архитектура библиотеки”.
- Модуль **Converter Composition**, в котором происходит конвертация документов к одному из поддерживаемых форматов в системе. Проверка возможности проведения конвертирования определяется автоматически. В библиотеке есть возможность конвертировать документ в один из следующих форматов: docx, excel, txt, pdf, png, pptx. Конвертирование из одних форматов в другие производится с использованием внешних библиотек. AbstractConverter - это базовый класс для конвертирования формата документа, является родителем нескольких дочерних классов, например PNGConverter для конвертирования формата изображений в формат PNG.
- Модуль **Reader Composition**, который содержит основные обработчики документов, извлекающие из документов текст и его визуальную информацию (метаданные). На данный момент модуль позволяет извлекать текст и метаданные в едином внутреннем представлении из документов следующих форматов: docx, excel, pptx, txt, json, csv, pdf с текстовым слоем, а также изображения документов (или pdf без текстового слоя). Извлечение содержимого из структурированных форматов производится с использованием внешних библиотек. AbstractReader - это базовый класс для извлечения содержимого, является родителем нескольких дочерних классов, например HTMLReader, извлекающий содержимое из формата HTML с использованием библиотеки beautifulsoup.
- Модуль **Metadata Extractor Composition** позволяет извлекать метаданные документа, такие как имя файла, время создания, последнего чтения и изменения, mime-тип файла, размер файла. Для разных форматов документов можно извлекать разные метаданные, поэтому базовый набор метаданных может пополняться специфическими метаданными, зависящими от формата документа. Извлечение метаданных производиться с использованием внешних библиотек.
- Модуль **Structure Extractor Composition**, который отвечает за обогащение выделенного с помощью модуля ‘readers’ промежуточного представления. К информации о строке добавляется ее тип (класс) и уровень вложенности в

соответствии с выбранной предметной областью документа. На данный момент доступны следующие типы документов: НПА, технические задания, ВКР (дипломы и магистерские диссертации) и тип по умолчанию “другое”.

- Модуль **Structure Constructor Composition**, который компонует промежуточное представление документа в выходную структуру определённого вида. На данный момент реализовано построение структуры документа двух типов: линейная и древовидная структуры.
- Модуль **Attachments Extractor**, который позволяет получать вложения из документов, например, изображения из документов в формате docx или сжатые файлы из архива.

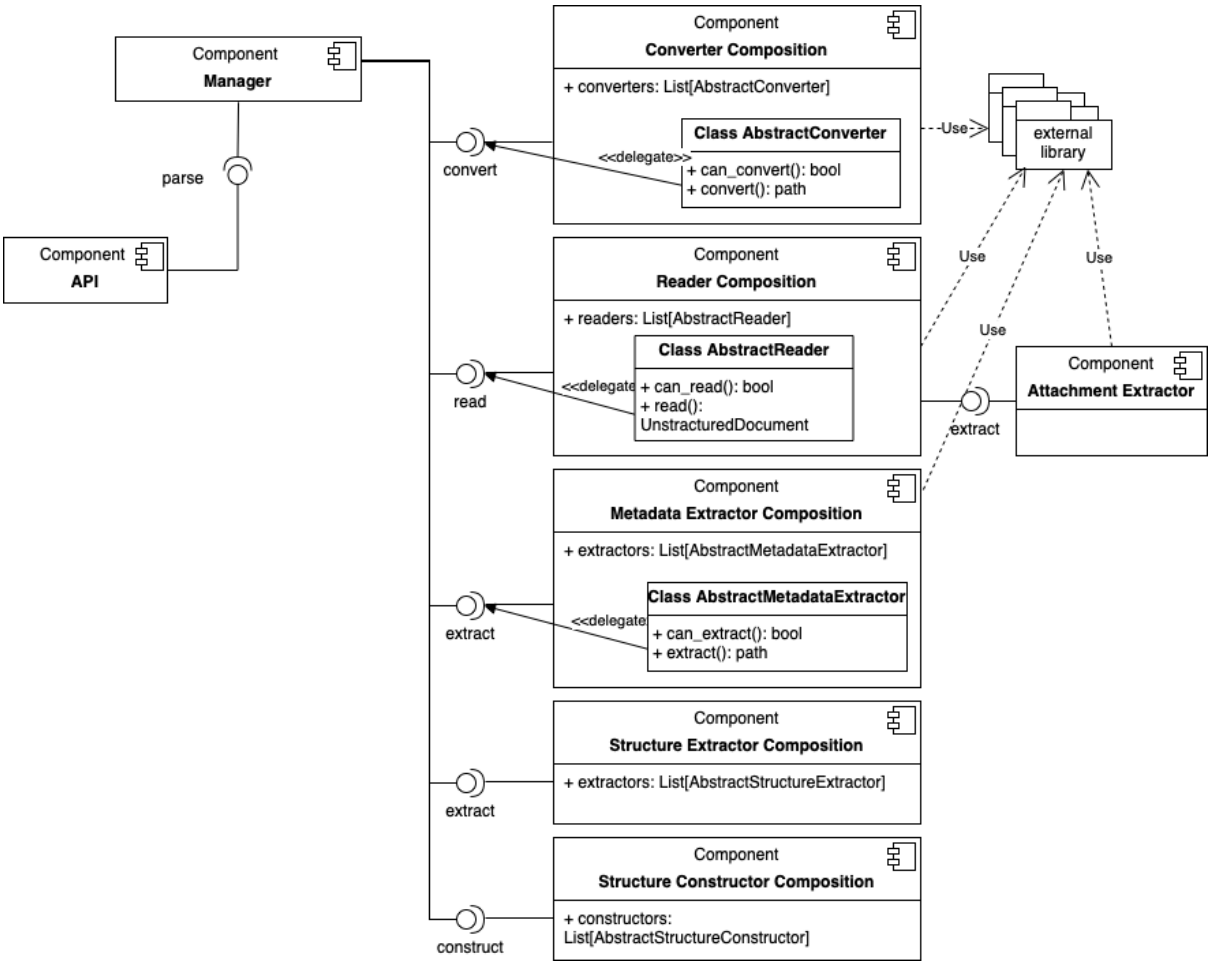


Рисунок 3.2.1 – UML-диаграмма компонентов программного комплекса.

Кроме того, были реализованы вспомогательные модули, содержащие основные структуры данных и дополнительные функции, необходимые для работы программного комплекса.

Из описания основных программных модулей следует, что каждая из частей может изменяться и дополняться независимо от других. Например, можно добавить обработку нового формата документов, добавив дополнительный обработчик конвертации или обработчик извлечения содержимого в модуль Reader Composition. Аналогично, можно добавить новый обработчик извлечения структуры из документов для новой предметной области на базе уже имеющегося общего для всех форматов документов промежуточного представления.

3.3 Методика расширения программного комплекса

Методика расширения программного комплекса предназначена для разработчиков и состоит из двух частей:

1) Добавление поддержки обработки нового формата документа;
2) Добавление поддержки обработки нового типа документа (новой структуры документа).

3.3.1 Добавление поддержки нового формата документа

Добавить поддержку обработки нового формата документа можно двумя способами:

1) Добавить новый класс Converter для конвертирования нового формата документа в тот, для которого есть обработчик (Reader) в системе. Например для обработки формата DJVU, достаточно реализовать класс, преобразующий формат DJVU в формат PNG.
2) Добавить новый класс Reader, позволяющий извлечь содержимое из нового формата документа.

Добавление класса Converter. Для добавления нового класса конвертера достаточно выполнить следующие действия:

1) Реализовать класс NewFormatConverter (Листинг 3.1), который является потомком базового класса AbstractConverter, расположенному по пути dedoc / converters / concrete_converters / abstract_converter.py.

Листинг 3.1 – Пример реализации нового класса-конвертера NewFormatConverter.

```
from dedoc.converters.concrete_converters.abstract_converter import AbstractConverter

class NewFormatConverter(AbstractConverter):
    def __init__(self, config: Optional[dict] = None) -> None:
        super().__init__(config=config)

    def can_convert(self,
                    file_path: Optional[str] = None,
                    extension: Optional[str] = None,
                    mime: Optional[str] = None,
                    parameters: Optional[dict] = None) -> bool:
        pass  # some code here

    def convert(self, file_path: str, parameters: Optional[dict] = None) -> str:
        pass  # some code here
```

2) Реализовать код функций can_convert() и convert(). Функция can_convert() должна проверять может класс обработать файл по пути file_path, если может, то вернуть True, если нет, то вернуть False. Функция convert() выполняет конвертацию файла, сохраняет его и возвращает путь к новому файлу.
3) Добавить реализованный класс NewFormatConverter в конфигурационный файл системы dedoc / manager_config.py. Ниже описано как изменить/добавить свой обработчик в конфигурационный файл.

Пример реализованного класса DjvuConverter расположен в приложении А в листинге А.1.

Добавление класса Reader. Для добавления обработчика для извлечения содержимого документа, нужно:

1) Реализовать класс NewFormatReader (Листинг 3.2), как потомка от базового класса BaseReader, расположенному по пути dedoc / readers / base_reader.py.

Листинг 3.2 – Пример реализации нового класса-обработчика NewFormatReader.

```
from dedoc.readers.base_reader import BaseReader

class NewFormatReader(BaseReader):

    def can_read(self, file_path: Optional[str] = None, mime: Optional[str] = None, extension: Optional[str] = None,
parameters: Optional[dict] = None) -> bool:
        pass  # some code here

    def read(self, file_path: str, parameters: Optional[dict] = None) -> UnstructuredDocument:
        pass  # some code here
```

2) Реализовать код функций can_read() и read(). Функция can_read() должна проверять может ли класс обработать файл, расположенному по пути file_path

и возвращать True, если может, и False иначе. Функция read() выполняет чтение файла расположенного по пути file_path, извлекает содержимое документа и заполняет структуру UnstructuredDocument.

3) Добавить реализованный класс NewFormatReader в конфигурационный файл системы dedoc / manager_config.py. Ниже описано как изменить/добавить свой обработчик в конфигурационный файл.

Пример реализованного класса PdfReader расположен в приложении А в листинге А.2.

Изменение/добавление конфигурации системы. Все используемые в процессе анализа обработчики должны быть прописаны в конфигурационном файле системы dedoc / manager_config.py. Можно реализовать свой конфигурационный файл с теми обработчиками, которые нужны при обработке и реализованными собственными обработчиками. Пример конфигурационного файла представлен в Листинге 3.3.

Листинг 3.3 – Пример конфигурационного файла dedoc_manager.py системы.

```
import os

from djvu_converter import DjvuConverter
from pdf_reader import PdfReader

from dedoc import DedocManager
from dedoc.attachments_handler import AttachmentsHandler
from dedoc.converters import ConverterComposition
from dedoc.metadata_extractors import BaseMetadataExtractor, DocxMetadataExtractor,
MetadataExtractorComposition
from dedoc.readers import ReaderComposition
from dedoc.structure_constructors import LinearConstructor, StructureConstructorComposition, TreeConstructor
from dedoc.structure_extractors import DefaultStructureExtractor, StructureExtractorComposition

manager_config = dict(
   converter=ConverterComposition(converters=[DjvuConverter()]),
   reader=ReaderComposition(readers=[PdfReader()]),
   structure_extractor=StructureExtractorComposition(extractors={DefaultStructureExtractor.document_type:
DefaultStructureExtractor()}, default_key="other"),
   structure_constructor=StructureConstructorComposition(
      constructors={"linear": LinearConstructor(), "tree": TreeConstructor()},
      default_constructor=LinearConstructor()
   ),
   document_metadata_extractor=MetadataExtractorComposition(extractors=[DocxMetadataExtractor(),
BaseMetadataExtractor()]),
   attachments_handler=AttachmentsHandler(),
)
```

3.3.2 Добавление поддержки нового типа документа

Метод восстановления иерархической структуры документа из содержимого использует метод машинного обучения градиентного бустинга над деревьями решений для классификации текстовых строк. По этой причине для добавления поддержки нового типа структуры документа нужно разметить документы и обучить классификатор. Для нового документа нужно будет выполнить несколько шагов:

1) подготовить набор данных;
2) прописать какие признаки будут извлекаться из текстовых строк;
3) обучить классификатор;
4) добавить обученный классификатор в программный комплекс;
5) изменить конфигурационный файл для поддержки нового классификатора.

Создание набора данных. На данном этапе нужно запустить программный комплекс (dedoc) в режиме разметки новых документов. Этот режим запускается командой:

```
docker-compose -f labeling/docker-compose.yml up --build
```

Подробнее о подготовке программного комплекса к созданию заданий по разметке описано в документации[14] В этом режиме система поднимается по адресу `localhost:1232`. На главной странице показаны инструкции как загрузить новые документы для создания разметки (Рисунок 3.3.2.1). В этом режиме, система извлекает содержимое из документов в виде текстовых строк с форматированием и подготавливает документы для разметки текстовых строк. На данном этапе, лучше загружать те форматы документов, которые будут использоваться на этапе прогноза (после обучения классификатора). На последнем этапе будут созданы задания для разметки (tasks) в виде zip-архивов для разметки во внешней системе разметки[15]. Инструкции по использованию внешней системы разметки предоставлена в ее репозитории в README.md. Далее пользователь размечает текстовые строки согласно подготовленным заданиям разметки (Рисунок 3.3.2.2).

[14]https://dedoc.readthedocs.io/en/latest/tutorials/add_new_structure_type/dataset_creation.html#add-tasker

[15] https://github.com/dronperminov/ImageClassifier

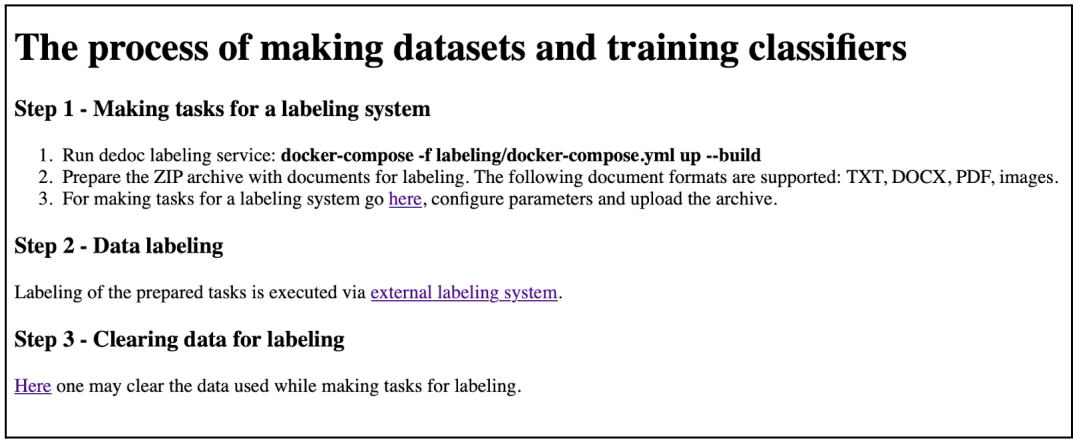


Рисунок 3.3.2.1 – Интерфейс системы в режиме разметки новой структуры документов.

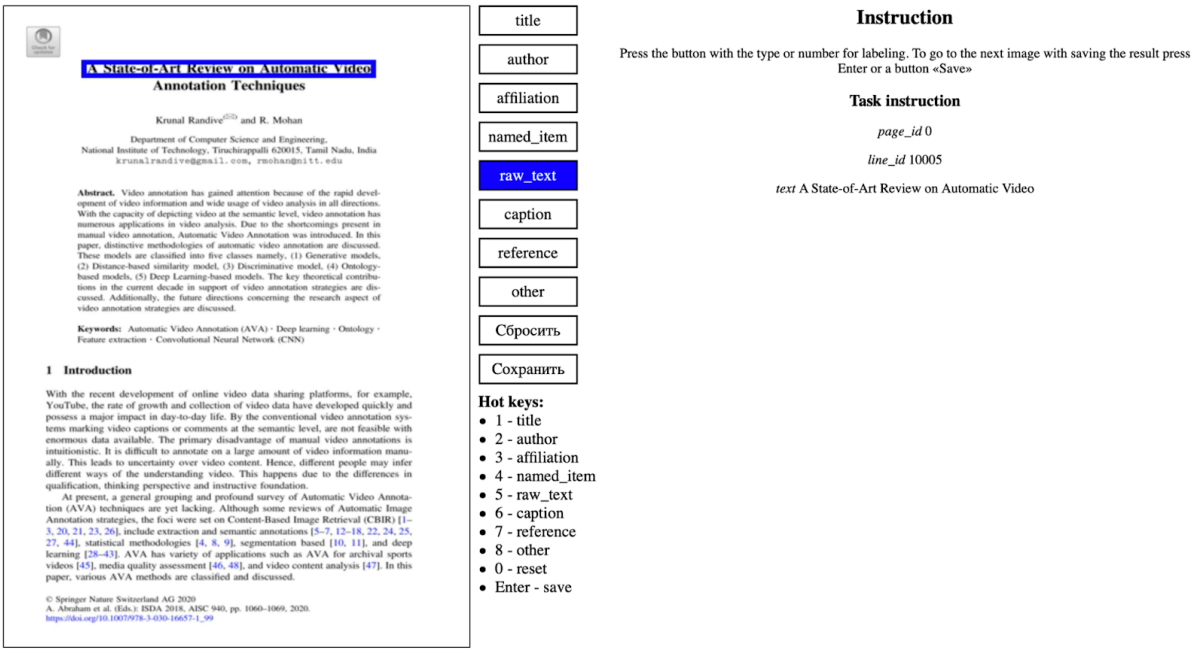


Рисунок 3.3.2.2 – Интерфейс внешней системы разметки, после загрузки в нее подготовленных заданий для разметки (tasks).

Реализация извлечения признаков из строк. На данном этапе разработчику нужно задать какие признаки нужно извлекать из строк. Для этого этапа используется класс, являющийся потомком класса AbstractFeatureExtractor. Для класса требуется реализовать метод transform(), в которой реализуется логика извлечения признаков из строк с форматированием. Здесь стоит использовать уже реализованные методы для извлечения признаков из базового класса AbstractFeatureExtractor, например get_color(), get_bold_percent() и так далее. Более подробная информация расположена

на странице[16] документации. Пример реализации кода такого класса извлечения признаков из научных статей на английском языке приведен в приложении А в листинге А.3.

Обучение классификатора. Для обучения классификатора нужно использовать существующий класс XGBoostLineClassifierTrainer, в который нужно передать параметры для обучения нового классификатора. В Листинге 3.4 приведена инициализация параметров для класса.

Обучение классификатора выполняется с помощью вызова метода fit() у инициализированного класса XGBoostLineClassifierTrainer.

Листинг 3.4 – Пример проинициализированных параметров для обучения нового классификатора строк для восстановления структуры из научных статей на английском языке.

```
classifier_name = "article_classifier"
data_url="url with training dataset",

# configure path for saving a trained classifier
classifier_directory_path = os.path.join(os.path.expanduser("~"), ".cache", "dedoc", "resources", "line_type_classifiers")
os.makedirs(classifier_directory_path, exist_ok=True)
classifier_path = os.path.join(classifier_directory_path, f"{classifier_name}.zip")

# configure paths for saving scores and features importances (this is not obligatory)
resources_path = os.path.abspath(os.path.join(os.path.dirname(__file__), "..", "..", "resources"))
assert os.path.isdir(resources_path)
path_scores = os.path.join(resources_path, "benchmarks", f"{classifier_name}_scores.json")
path_feature_importances = os.path.join(resources_path, "feature_importances",
f"{classifier_name}_feature_importances.xlsx")

# features extractor for the classifier
feature_extractor = ArticleFeatureExtractor()

# parameters of the XGBClassifier
(https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBClassifier)
classifier_parameters = dict(learning_rate=0.5, n_estimators=600, booster="gbtree", tree_method="hist", max_depth=3,
colsample_bynode=0.8)

# dedoc configuration (just in case)
config = get_config()
```

Добавление обученного классификатора в систему. К данному этапу был подготовлен набор данных, размечен и обучен классификатор строк для нового типа документов. Теперь нужно реализовать класс (Листинг 3.5) на основе базового класса dedoc.structure_extractors.line_type_classifiers.abstract_pickled_classifier.AbstractPickled

[16] https://dedoc.readthedocs.io/en/latest/tutorials/add_new_structure_type/classes.html#features-extraction

LineTypeClassifier, который будет использовать обученный классификатор и вычислять классы для строк входного документа в функции predict().

Листинг 3.5 – Пример класса ArticleLineTypeClassifier для вычисления прогнозов нового обученного классификатора.

```
class ArticleLineTypeClassifer(AbstarctPickledLineTypeClassifier):
    def __init__(self, path: str, *, config: Optional[dict] = None) -> None:
        super().__init__(config=config)
        self.classifier, feature_extractor_parameters = self.load("article", path)
        self.feature_extractor = ArticleFeatureExtractor()

    def predict(self, lines: List[LineWithMeta] -> List[str]:

        features = self.feature_extractor.transform([lines])
        labels_probability = self.classifier.predict_proba(features)
        // …
```

Добавление класса извлечения структуры. Класс извлечения структуры устанавливает приоритет классам, которые возвращаются классификатором. Т.е. этот этап также можно отнести к этапу постобработки результатов классификации. Здесь достаточно реализовать класс, как наследника класса dedoc.structure_extractors.abstract_structure_extractor.AbstractStructureExtractor. В этом классе нужно реализовать код функции extract(). На данном этапе для разных классов строк, назначается соответствующий уровень. В листинге 3.6 приведен пример назначения приоритетов, согласно классам.

Листинг 3.6 – Пример назначения приоритетов строк, согласно классам строк.

```
def extract(self, document: UnstructuredDocument, parameters: Optional[dict] = None) -> UnstructuredDocument:
        predictions = self.classifier.predict(document.lines)
        assert len(predictions) == len(document.lines)

        for line, line_type in zip(document.lines, predictions):

            if line_type == "title":
                # for root, level_1=0, level_2=0, can_be_multiline=True
                line.metadata.hierarchy_level = HierarchyLevel.create_root()
                continue
            if line_type == "named_item":
                # for named_item, level_1=1,2, can_be_multiline=True
                line.metadata.hierarchy_level = self.__handle_named_item(line=line, prediction=line_type, level_1=1)
                continue

            if line_type in ("author", "affiliation", "reference"):
                line.metadata.hierarchy_level = HierarchyLevel(level_1=3, level_2=1, can_be_multiline=False,
line_type=line_type)
                continue

            # for caption and raw_text, level_1=None, level_2=None, can_be_multiline=True
            line.metadata.hierarchy_level = HierarchyLevel.create_raw_text()
            line.metadata.hierarchy_level.line_type = line_type
```

Изменение конфигурационного файла системы. Как сказано в разделе 3.3.1 чтобы новые написанные обработчики заработали нужно прописать их в конфигурационном файле системы, например в dedoc / manager_config.py. В случае

добавления поддержки типа структуры документа (например, обработки научных статей на английском языке) достаточно добавить вызов созданных классов как в листинге 3.7.

Листинг 3.7 – Пример составления конфигурации системы с поддержкой нового типа документа научной статьи на английском языке в формате PDF.

```
from article_structure_extractor import ArticleStructureExtractor

from dedoc import DedocManager
from dedoc.attachments_handler import AttachmentsHandler
from dedoc.converters import ConverterComposition
from dedoc.metadata_extractors import MetadataExtractorComposition, PdfMetadataExtractor
from dedoc.readers import PdfAutoReader, ReaderComposition
from dedoc.structure_constructors import StructureConstructorComposition, TreeConstructor
from dedoc.structure_extractors import StructureExtractorComposition

manager_config = dict(
    converter=ConverterComposition(converters=[]),
    reader=ReaderComposition(readers=[PdfAutoReader()]),
    structure_extractor=StructureExtractorComposition(extractors={ArticleStructureExtractor.document_type:
ArticleStructureExtractor()},
                                                      default_key=ArticleStructureExtractor.document_type),
    structure_constructor=StructureConstructorComposition(constructors={"tree": TreeConstructor()},
default_constructor=TreeConstructor()),
    document_metadata_extractor=MetadataExtractorComposition(extractors=[PdfMetadataExtractor()]),
attachments_handler=AttachmentsHandler(),)

manager = DedocManager(manager_config=manager_config)
```

Запуск обработки документов с добавленным новым типом структуры будет выглядеть так:

result = manager.parse(file_path, parameters = {"document_type": "article"})

3.4 API-интерфейс

Библиотека может быть использована для запуска web-сервера с определенным набором API-запросов. По умолчанию, сервер поднимается на порту 1231 и предоставляет следующий интерфейс:

- На главной странице расположена основная информация о проекте и ссылка на страницу загрузки;
- На странице загрузки /upload можно загрузить файл на обработку, предварительно установив настройки обработки;
- На странице /docs расположена онлайн-документация проекта, полученная автоматически на основе исходного кода проекта.

Результат обработки можно получить в разном формате вывода в зависимости от установленных настроек: в формате html, json, “дерево”.

Обрабатывать документы можно также с помощью post-запросов на /upload с приложенным файлом для обработки и набором параметров (листинг 3.4.1).

Листинг 3.4.1 – Пример запроса на языке Python с помощью библиотеки *requests*.

```
data = {
            "pdf_with_text_layer": "auto_tabby",
            "document_type": "diploma",
            "language": "rus",
            "need_pdf_table_analysis": "true",
            "need_header_footer_analysis": "false",
            "is_one_column_document": "true",
            "return_format": 'html'
}
with open(filename, 'rb') as file:
    files = {'file': (filename, file)}
   r = requests.post("http://localhost:1231/upload", files=files, data=data)
   result = r.content.decode('utf-8')
```

Сервис по обработке документов принимает на вход следующие параметры:

- **language** – язык, на котором написан текст на сканированных документов. Эта настройка необходима для корректного распознавания текста документов, на данный момент доступны русский и английский языки. Доступные опции: "rus+eng", "rus", "eng", опция по умолчанию "rus+eng".
- **with_attachments** – опция для извлечения вложений из документов. По умолчанию установлена в “false”, то есть вложения из документа извлекаться не будут, даже если они есть. Доступные опции: “false”, “true”.
- **insert_table** – опция, управляющая вставкой извлеченных из документа таблиц в итоговое выходное дерево документа. По умолчанию таблицы возвращаются отдельно от основного документа, т.е. опция выставлена в “false”. Доступные опции: “false”, “true”.
- **return_format** – выходной формат, сервис может вернуть результат в виде html-страницы, json-документа или дерева специального вида. Доступные опции: “json”, “pretty_json”, “html”, “tree”, опция по умолчанию “json”.

- **structure_type** – тип выходной структуры документа. Структура может быть линейной – список строк документа, либо в виде дерева – иерархическая структура вложенных друг в друга узлов. Доступные опции: “linear”, “tree”, опция по умолчанию "tree".
- **delimiter** – сепаратор, использующийся для разделения ячеек в csv и tsv файлах. По умолчанию используется “,”.
- **encoding** – кодировка документов (в основном используется для файлов в формате txt). По умолчанию значение не задается, кодировка определяется автоматически.
- **document_type** – тип документа, соответствующий определенной предметной области. На данный момент доступны обработка НПА, технических заданий, текстов курсовых работ и ВКР, слайдов презентаций, а также тип по умолчанию, позволяющий получить представление в виде дерева для любого документа. Доступные опции: "", "law", "tz", "diploma", "article", "slide", опция по умолчанию "".
- **pdf_with_text_layer** – опция, на основе значения которой выбирается способ обработки pdf документов: документы могут иметь текстовый слой или быть сканированными изображениями текста. Опция позволяет явно задать тип документа, однако имеется возможность автоматически определить наличие текстового слоя. Доступные опции: "true", "false", "auto", "auto_tabby", "tabby", опция по умолчанию "auto_tabby".
- **pages** – диапазон страниц pdf-документа, которые необходимо прочитать, настраивается как “начальная страница:конечная страница”, обе страницы включаются в диапазон. Если с какой-то стороны нет ограничения, поле можно оставить пустым. Значение по умолчанию “:”, т.е. документ читается полностью.
- **orient_analysis_cells** – опция, используемая при распознавании изображений таблиц, позволяет настроить распознавание повернутых ячеек в заголовках таблиц. Доступные опции: “false”, “true”, опция по умолчанию “false”.
- **orient_cell_angle** – опция, используемая при распознавании изображений таблиц, позволяет настроить угол поворота ячеек в заголовках таблиц. Доступные опции: “90”, “270”, опция по умолчанию “90”.
- **is_one_column_document** – опция, используемая при обработке pdf-документов без текстового слоя. Значение “true” означает то, что нужно

обработать документ как одноколоночный, “false” – многоколоночный. Также есть возможность автоматического определения многоколоночности. Доступные опции: “false”, “true”, “auto”, опция по умолчанию “auto”.

- **document_orientation** – параметр, используемый для настройки автоматического исправления ориентации повернутых изображений документов. Значение “no_change” означает то, что ориентацию документа определять не нужно, оставить его как есть, значение “auto” предоставляет возможность автоматического определения ориентации каждой страницы документа (определяется поворот на 0, 90, 180, 270 градусов). Доступные опции: “no_change”, “auto”, опция по умолчанию “auto”.
- **html_fields** – список путей в json-документе до значений, которые необходимо обработать как html-документы, значение по умолчанию "".
- **need_header_footer_analysis** – используется при обработке pdf-документов, если значение установлено в “true”, верхние и нижние колонтитулы убираются из выходного результата. Доступные опции: “false”, “true”, опция по умолчанию “false”.
- **need_pdf_table_analysis** – опция, управляющая анализом таблиц в pdf-документах без текстового слоя. По умолчанию анализ таблиц выполняется. Доступные опции: “false”, “true”, опция по умолчанию “true”.
- **handle_invisible_table** – опция, управляющая обработкой таблиц без видимых границ в html-документах. По умолчанию такие таблицы рассматриваются как обычный текст. Доступные опции: “false”, “true”, опция по умолчанию “false”.
- **return_base64** – опция, позволяющая возвращать вложенные изображения в формате base64. Доступные опции: “false”, “true”, опция по умолчанию “false”.
- **need_content_analysis** – опция, отвечающая за обработку вложений в рамках обработки основного документа. Если установлен в “true”, содержимое вложений будет возвращено вместе с содержимым основного документа. Доступные опции: “false”, “true”, опция по умолчанию “false”.
- **recursion_deep_attachments** – глубина обработки вложенных файлов, значение по умолчанию 10.
- **need_binarization** – опция, позволяющая обрабатывать сканированные документы с неоднородным фоном. Доступные опции: “false”, “true”, опция по умолчанию “false”.

Вышеописанные параметры позволяют настроить работу сервиса для конкретных нужд пользователя.

3.5 Внутреннее и выходное представления документа

В результате обработки документа программным комплексом, все обрабатываемые документы приводятся к внутреннему представлению документа, который описывается классом ParsedDocument, состоящего из следующих полей:

- **content** (DocumentContent) – содержит иерархическую структуру документа с текстовым содержимым и форматированием. Состоит из:
    - **structure (**TreeNode) - содержит иерархическую структуру, где узел структуры описывается классом TreeNode, которая состоит из полей:
        - **node_id** (строка) – уникальный идентификатор узла секции в иерархической структуре.
        - **text** (строка) – текст узла секции в иерархической структуре.
        - **annotations** (List[Annotation]) – список форматирования текста узла секции. Здесь Annotation состоит из полей start (индекс символа начала форматирования), end (индекс символа конца форматирования), name - имя форматирования, например полужирный текст ‘bold’, размер текста ‘size’, выравнивание текста ‘alignment’, ссылка на таблицу ‘table’), value - значение форматирования (например, для true/false или вещественное значение для обозначения размера шрифта).
        - **metadata (**словарь) – содержит служебную информацию секции, например номер страницы.
        - **subparagraphs** (List[TreeNode]) – содержит список дочерних узлов секций в иерархической структуре.
        - **parent** – (TreeNode) - ссылка на родительский узел.
    - **tables**: List[Table] – список таблиц с текстом ячеек и информацией о физической структуре (объединении ячеек). Класс Table состоит из:

        - **metadata** (TableMetadata) – служебная информация о таблице (уникальный идентификатор таблицы, название таблицы, угол поворота таблицы).
        - **cells** (List[List[CellWithMeta]]) – класс CellWithMeta содержит о каждой ячейке таблицы, а именно информацию о тексте и форматировании (lines: List[LineWithMeta]) в ячейке, информацию о физической структуре (colspan - объединение по горизонтали, rowspan – объединение по вертикали, invisible - ячейка должна отображаться или нет)
- **attachments** (List[ParsedDocument]) – список вложенных документов.
- **metadata** (DocumentMetadata) – содержит служебную информацию о документе (тип и размер файла, даты создания/редактирования, имя автора).
- **warnings** (List[строка]) – список предупреждений, возникших в результате анализа документа.

В структуре на выходе каждая строка является узлом со своим уникальным node_id, который состоит из пары чисел, разделенных точкой, и формируется следующим образом:

- Первое число означает уровень вложенности строки в документе по отношению к корню. Уровень вложенности корня равен 0, далее увеличивается.
- Второе число означает номер строки внутри списка строк с одинаковым уровнем вложенности. Например, первые два элемента списка будут иметь одинаковое значение первого числа (например, 2) в node_id, но разные значения вторых чисел: 2.0 и 2.1.

Класс ParsedDocument схож с объектной моделью документа известной как DOM (Document Object Model). Модель DOM представляет собой объектную модель документа, которую браузеры создают в памяти во время обработки html-кода страницы. DOM модель представляет документ в виде иерархического дерева тегов (Рисунок 1.3.1.1). Каждая ветвь дерева заканчивается узлом, а каждый узел содержит совокупность дочерних объектов. Модель документа имеет метаданные документа в корне дерева. Модель DOM легла в основу разработки класса внутреннего представления документа ParsedDocument.

Обработанные системой документы приводятся к единому выходному формату. В системе поддерживается несколько форматов выходных представлений документа на выбор:

1. Сериализованный json-формат класса ParsedDocument.
2. HTML-формат.
3. Сплошной текст TXT, без выделенной структуры в документе (может быть использован, в случае потребности в получении только содержимого документа).

HTML формат и TXT-формат формируются на основе внутреннего представления документа ParsedDocument.

3.6 Установка программного комплекса

Программный комплекс может быть использован разными способами: как docker-контейнер и как Python пакет, устанавливаемый командой pip. Далее каждый из этих методов описан более подробно.

Сборка docker-образа. Для облегчения установки зависимостей и запуска системы, существует возможность собрать проект с помощью специального инструмента контейнеризации для сборки и запуска приложений – docker. Сервер может быть запущен с помощью системы docker-compose, в которой docker-образ приложения собирается автоматически.

Преимуществом использования контейнеризации является простота установки и развертывания программный комплекс в виде сервиса в специальный docker-образ. Контейнеризация предоставляет возможность автоматической установки требуемого окружения со всеми зависимостями, необходимыми для запуска библиотеки. Все зависимости хранятся в docker-образе. Это позволяет поддерживать актуальное состояние окружения библиотеки на любом устройстве и в любой операционной системе, в которой можно установить пакет docker. Таким образом, библиотека dedoc может быть запущена в качестве самостоятельного сервиса как в операционной системе Windows, так и в операционных системах на базе UNIX.

Docker-образ удобен в использовании со сторонними приложениями, которые могут его использовать как отдельный компонент наряду с другими компонентами. В данном случае, обращение к поднятому сервису обеспечивается через API-интерфейс.

Кроме того, использование docker-compose удобно в процессе тестирования приложения, а именно – для автоматического тестирования API-запросов, а также при непрерывной интеграции (CI – Continuous Integration).

Для автоматизации сборки и запуска проекта через docker были написаны вспомогательные файлы:

- Dockerfile, описывающий последовательность команд установки необходимых зависимостей и запуска приложения;
- Файл docker-compose.yml, предоставляющий возможность запуска нескольких контейнеров и используемый при тестировании приложения.

Сборка приложения основана на использовании “базового” docker-образа с ОС Ubuntu и предустановленными пакетами для работы с языком программирования Python. В этот образ также входит собранная из исходных файлов программа Tesseract, позволяющая производить оптический анализ символов (OCR), а также библиотеки по компьютерному зрению Pytorch и Torchvision. Данный образ помогает ускорить процесс сборки образа приложения, он находится в открытом реестре образов docker-hub.

Таким образом, для запуска проекта достаточно установить на компьютере программу docker. При удовлетворении этого условия, сборку и запуск приложения можно осуществить с помощью следующей команды:

```
docker-compose up --build
```

Сборка и публикация в виде библиотеки. Программный комплекс разработан преимущественно на языке программирования Python, соответственно для его автоматической сборки и распространения должен быть выбран стандартный инструмент. На языке Python существует несколько популярных инструментов для сборки и управления библиотеками. В силу своей популярности и разнообразия возможностей, для автоматической сборки библиотеки был выбран инструмент setuptools. Наиболее современным методом применения setuptools является использование вспомогательного файла pyproject.toml.

Для осуществления автоматической сборки библиотеки, исходный код проекта должен соответствовать определенной структуре. Структура проекта для сборки библиотеки с помощью setuptools может варьироваться, однако пример типичной структуры проекта выглядит следующим образом:

- Все необходимые для работы файлы должны лежать в корневой папке проекта, которая может иметь название, идентичное названию итоговой библиотеки.
- Внутри корневой папки должна находиться папка с исходным кодом проекта. Название этой папки должно совпадать с названием создаваемой библиотеки. Внутри этой папки располагаются модули с расширением .py, а также файл __init__.py, который делает папку с кодом пакетом Python.
- Внутри корневой папки также может находиться папка tests, содержащая модульные тесты для разрабатываемой библиотеки. Отделение кода тестов от исходного кода библиотеки является хорошей практикой, так как будущему пользователю этот код не нужен.
- Для использования setuptools в процессе сборки, в корневой папке должен находиться файл pyproject.toml, содержащий метаданные проекта и настройки сборки.
- В корневую папку по общепринятым нормам могут быть добавлены и другие файлы, например, LICENSE.txt, содержащий текст лицензии библиотеки, README.md, содержащий описание библиотеки, инструкции по установке и использованию, requirements.txt, содержащий зависимости проекта.

Листинг 3.6.1 – Пример частичного pyproject.toml файла для сборки Python-библиотеки с использованием setuptools.

```
# Средства, необходимые для автоматической сборки
[build-system]
requires = ["setuptools"]
build-backend = "setuptools.build_meta"

# Основная метаинформация по проекту
[project]
name = "dedoc"
authors = [
    {name = "Dedoc team", email = "dedoc@ispras.ru"}
]
readme="README.md"
license = {file = "LICENSE.txt"}
dynamic = ["dependencies", "version"]
requires-python = ">=3.6"

# Поля, настраиваемые динамически
[tool.setuptools.dynamic]
version = {file = "VERSION"}  # Версия библиотеки
dependencies = {file = ["requirements.txt"]}  # Зависимости проекта

# Файлы, включаемые в пакет
[tool.setuptools.packages.find]
```

```
where = ["."]
include = ["dedoc*"]

# Зависимости проекта, устанавливаемые по требованию
[project.optional-dependencies]
torch = ["torch~=1.11.0", "torchvision~=0.12.0"]

# Точки входа для библиотеки
[project.scripts]
dedoc = "dedoc.main:main"

# Ссылки, относящиеся к проекту
[project.urls]
homepage = "https://github.com/ispras/dedoc"
repository = "https://github.com/ispras/dedoc.git"
documentation = "https://dedoc.readthedocs.io"
```

Для создания установочных пакетов Python с использованием setuptools и инструмента сборки, указанного в pyproject.toml, используется следующая команда: `python -m build -w` , где флаг -w или --wheel указывает на необходимость создания бинарного пакета в формате Wheel (.whl). Wheel — это бинарный формат пакета, который позволяет более эффективно распространять и устанавливать пакеты Python. После успешного выполнения команды python -m build -w, будут созданы установочные пакеты в соответствии с настройками и зависимостями проекта. Созданные пакеты помещаются в директорию dist, находящуюся в корневом каталоге проекта. Таким образом, описанная команда позволяет автоматизировать процесс сборки и создания пакетов, что облегчает распространение и установку библиотеки на других системах.

После сборки библиотеки, пакет в формате Wheel публикуется в общедоступное хранилище подобных пакетов, например, на PyPI[17]. PyPI (Python Package Index) – это репозиторий пакетов для языка программирования Python. Он является официальным и наиболее широко используемым репозиторием пакетов Python. PyPI предоставляет информацию о пакетах, такую как имя, версия, описание, автор и зависимости.

Для распространения пакетов можно использовать twine — это инструмент командной строки, который облегчает публикацию пакетов Python на PyPI. Он позволяет упаковать пакет в установочный пакет формата Wheel или в архив формата Source Distribution (sdist) и загрузить его на сервер PyPI. Для публикации библиотеки можно использовать команду twine upload с указанием путей к пакетам в виде

[17] https://pypi.org/

шаблонов, например, dist/* для загрузки всех пакетов из директории dist. Twine передаст учетные данные для аутентификации на сервере PyPI и отправит пакеты на сервер.

Таким образом, осуществлена сборка библиотеки с использованием инструмента setuptools, а также ее публикация на PyPI с помощью twine. Это позволило облегчить пользователям процесс установки разработанного программного комплекса. Библиотека располагается на сервере пакетов PyPI по ссылке https://pypi.org/project/dedoc.

Для того чтобы установить библиотеку в текущее окружение, достаточно выполнить команду:

```
pip install dedoc
```

Зачастую одной этой команды недостаточно в силу того, что библиотека обладает рядом зависимостей для запуска отдельных модулей, например, tesseract, soffice, ddjvu. Без установки этих зависимостей функции библиотеки могут быть использованы, но не в полном объеме. Подробная инструкция по установке зависимостей доступна на сайте с документацией проекта (https://dedoc.readthedocs.io).

Минимальные технические требования для работы с программным комплексом:

- интерпретатор Python 3.8, Python3.9 или Python3.10;
- ОС Ubuntu 22.04.5 LTS (Jammy Jellyfish);
- Оперативная память от 32 Гб;
- 128 Гб диск.

Конфигурация использованного тестового стенда в экспериментах:

- Ubuntu 22.04.5 LTS (Jammy Jellyfish);
- 16Gb RAM;
- Процессор Intel Core i5-12400 CPU 2.5 ГГц;
- 512 SSD;
- dedoc==2.6 (при тестировании исходников использован тег v2.6 https://github.com/ispras/dedoc/tree/v2.6)

3.7 Документация

Целевой аудиторией документации разработанного программного комплекса являются разработчики ПО. Документация предназначена для сокращения времени изучения предназначения, функций и кода системы.

Для документирования программного обеспечения на языке программирования Python существует несколько популярных инструментов. Популярные инструменты документирования:

1. **Docstrings в стандарте PEP 257**. Встроенный в Python стандарт PEP 257 определяет правила для написания docstrings — строк документации, которые встраиваются непосредственно в исходный код Python. Docstrings могут быть использованы для описания модулей, классов, методов и функций. Их можно извлечь автоматически и использовать для генерации документации в различных форматах.
2. **Sphinx** — это мощный инструмент для генерации документации на основе docstrings. Он позволяет создавать красиво оформленные HTML, PDF, ePub и другие форматы документации. Sphinx поддерживает множество функций, таких как генерация автоматического содержания, индексирование, перекрестные ссылки и многое другое. Он часто используется для документирования проектов на Python, включая библиотеки и фреймворки.
3. **MkDocs** — это простой инструмент для создания статической документации на основе Markdown. Он предоставляет простой способ организации и представления документации в виде красивого сайта. MkDocs поддерживает автоматическое создание навигационной панели, поиск, настраиваемые темы оформления и множество плагинов для расширения функциональности.
4. **Read the Docs** (https://readthedocs.org/) — это платформа хостинга документации, которая интегрируется с репозиториями на GitHub, Bitbucket и других популярных платформах. Она автоматически строит документацию на основе исходного кода и обновляет ее при каждом коммите. Read the Docs позволяет удобно хранить и публиковать документацию, а также предоставляет функции поиска, версионирования и обратной связи для пользователей.

При написании документации к программному комплексу был использован широко известный инструмент Sphinx. Он обладает определенными особенностями, например, использует разметку ReStructuredText (reST) для написания документации, позволяет генерировать документацию в различных форматах, таких как HTML, PDF, ePub и др., автоматически генерирует содержание на основе структуры документации и добавленных заголовков, предоставляет возможность перевода документации на разные языки, интегрируется с популярными системами контроля версий, такими как Git, Mercurial и Subversion.

3.8 Выводы к третьей главе

В третьей главе было представлено описание архитектуры разработанного программного комплекса для автоматического извлечения содержимого и восстановления структуры. Программный комплекс содержит разработанные в диссертационной работе методы, описанные во второй главе. Также были описаны основные компоненты архитектуры и их основные функции в системе.

В разделе 3.3 представлена методика расширения программного комплекса для добавления поддержки новых форматов и типов документов. Методика была продемонстрирована на примере добавления поддержки извлечения структуры из PDF-документов научных статей на английском языке.

В разделе 3.4 приведено описание программного комплекса в виде API-сервиса с API-интерфейсом, которым удобно пользоваться разработчикам.

В разделе 3.5 подробно описано внутреннее и выходное представления документа в программном комплексе, а также доступные пользователю выходные форматы обработанных документов.

Кроме того, в главе содержится информация о запуске, установке и сборке программного комплекса, как в виде Python-библиотеки, так и в виде изолированного API-сервиса. Также приведена информация об автоматически генерируемой документации.

Заключение

В диссертационной работе были достигнуты следующие результаты:

1. Разработан обладающий научной новизной метод автоматического извлечения содержимого PDF-документов с использованием проверки текстового слоя, обеспечивающий достоверность извлечения и скорость обработки документов. Благодаря методу точность извлечения текста повышается на 3.3% (Character Accuracy), а время обработки сокращается более чем в 5 раз по сравнению с обработкой изображений сканированных документов;
2. Разработан обладающий научной новизной метод восстановления иерархической структуры из содержимого документов. Метод демонстрирует высокое качество на размеченных документов трех типов и показывает лучшие результаты на наборе данных международного соревнования FINTOC;
3. Разработана расширяемая архитектура программного комплекса, которая позволяет добавлять поддержку обработки новых форматов и типов структур документов. Архитектура позволяет обрабатывать документы в автоматическом режиме и приводить обрабатываемые документы к единому унифицированному виду;
4. На основе разработанной архитектуры и методов реализован программный комплекс в виде открытой библиотеки/системы, позволяющий автоматически обрабатывать документы разных форматов и типов структур с целью извлечения их содержимого и восстановления структуры в едином унифицированном виде. Внедрения программного комплекса подтвердили актуальность и практическую значимость диссертации.

Благодарности

Хочу выразить искреннюю благодарность моему мужу Дмитрию и моей маме Елене за неоценимую поддержку и понимание на всех этапах работы над диссертацией. Особую признательность выражаю Турдакову Денису за профессиональное научное руководство. Отдельную благодарность хочу выразить коллегам и студентам из моей команды за содействие в проведении исследований и достижении высоких результатов.

Список сокращений и условных обозначений

OCR	Optical Character Recognition, оптическое распознавание символов с изображений
DLA	Document Layout Analysis, анализ шаблона страницы документа
LLM	Language Large Model, большая языковая модель
Бинаризация изображения	Процесс преобразования из изображения (или градациях серого) в черно-белое
Предметная область (тип) документа	Содержимое документа, составленное по определенным правилам предметной области, которые описывают форматирование, структуру и семантику документа
Формат документа	Формат файла документа, содержащий определенную структуру (теги) согласно спецификации формата
MMO	Модель машинного обучения
F1	F-мера, A-score
RAG	Retrieval Augmented Generation, поисковая модель, дополненная генераций
Character Accuracy	Метрика оценки сходства между двумя текстами (истинным и распознанным) с вычислением расстояния Левенштейна
NLP	Natural Language Processing, обработка текстов на естественном языке
ICDAR	The International Conference on Document Analysis and Recognition
FINTOC	Financial Document Structure Extraction Shared Task, соревнование в рамках конференции FNP (Financial Narrative Processing)

Список литературы

1. Belyaeva O. Dedoc: A Universal System for Extracting Content and Logical Structure From Textual Documents / Belyaeva O., Bogatenkova A., Turdakov D. // 2023 Ivannikov Ispras Open Conference (ISPRAS). — IEEE, 2023. — P. 20-25.
2. Anastasiia Bogatenkova. ISPRAS@FinTOC-2022 Shared Task: Two-stage TOC Generation Model / Anastasiia Bogatenkova, Oksana Vladimirovna Belyaeva, Andrew Igorevich Perminov, Ilya Sergeevich Kozlov. // In Proceedings of the 4th Financial Narrative Processing Workshop @LREC2022, Marseille, France. European Language Resources Association. — 2022. — P. 89–94.
3. Kozlov I. Ispras@ fintoc-2021 shared task: Two-stage toc generation model / Kozlov I. Belyaeva. O., [et al.] // Proceedings of the 3rd Financial Narrative Processing Workshop. — 2021. — P. 81-85.
4. Belyaeva O. V. Automatic verification of the text layer correctness in PDF documents / Belyaeva O. V., Golodkov A., Bukhatov B. // 2024 Ivannikov Memorial Workshop (IVMEM). — IEEE, — 2024. — P. 1-7.
5. Golodkov A.O. Real Application of CNN Interpretation Methods: Document Image Classification Model Errors' Detection and Validation / Golodkov A.O., Belyaeva O.V., Perminov A.I. // Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS). — 2023. — Vol 35. — P. 7-18. — (ВАК).
6. Bogatenkova A. O. A.I. Logical structure extraction from scanned documents / Bogatenkova A. O. Kozlov I. S., Belyaeva O. V., Perminov // Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS). — 2020. — Vol 32. — P. 175-188. — (ВАК).
7. Belyaeva O.V. Synthetic data usage for document segmentation models fine-tuning / Belyaeva O.V., Perminov A.I., Kozlov I.S. // Proceedings of the Institute for System Programming of the RAS (Proceedings of ISP RAS). — 2020. — Vol 32. — P. 189-202. — (ВАК).
8. Perminov A.I. Loss functions for train document image segmentation models / Perminov A.I., Turdakov D.Yu., Belyaeva O.V. // Programming and Computer Software. – 2023. — Vol 49. — P. 574-589. — (ВАК, WoS).
9. M. S. Akopyan. Text Recognition on Images from Social Media / M. S. Akopyan, O. V. Belyaeva, T. P. Plechov and D. Y. Turdakov // *2019 Ivannikov Memorial Workshop (IVMEM),* Velikiy Novgorod, Russia. — 2019. — P. 3-6. — (WoS).

10. A. O. Bogatenkova. Generation of Images with Handwritten Text in Russian / A. O. Bogatenkova, O. V. Belyaeva, A. I. Perminov // Programming and Computer Software. — 2024. — Vol 50. — P. 483-492. — (ВАК, Scopus).
11. Puredoc: сервис обработки изображений документов / Беляева О.В., Богатенкова А.О., Перминов А.И., Голодков А.О., Шевцов Н.С., Рахматуллаев Т.А., Михайлов А.А., Зыкин Я.И.; ФГБУН Институт системного программирования РАН. — No 2023688256; заявл. 15.12.2023 (Рос. Федерация).
12. Docreader / Козлов И.С., Беляева О.В., Богатенкова А.О., Перминов А.И.; ФГБУН Институт системного программирования РАН. — No 2020666950; заявл. 21.12.2020 (Рос. Федерация).
13. Dedoc / Козлов И.С., Беляева О.В., Богатенкова А.О., Перминов А.И.; ФГБУН Институт системного программирования РАН. — No 2020667079; заявл. 21.12.2020 (Рос. Федерация).
14. Arlazarov V.V. Document image analysis and recognition: a survey / Arlazarov V.V., [et al.] // Компьютерная оптика. — 2022. — Vol. 46, no 4. — P. 567-589.
15. Jonathan J. Hull. Document image skew detection: Survey and annotated bibliography / Jonathan J. Hull // Document Analysis Systems II. World Scientific. — 1998. — P. 40-64.
16. Shijian Lu. Automatic document orientation detection and categorization through document vectorization / Shijian Lu, Chew Lim Tan // Proceedings of the 14th ACM international conference on multimedia. — 2006. — P. 113-116.
17. Shivam Aggarwal. Text Document Orientation Detection Using Convolutional Neural Networks / Shivam Aggarwal, Safal Singh Gaur // Intelligent Learning for Computer Vision: Proceedings of Congress on Intelligent Systems 2020. Springer. – 2021. — P. 153-164.
18. Shaheera Saba Mohd Naseem Akhter. Improving Skew Detection and Correction in Different Document Images Using a Deep Learning Approach / Shaheera Saba Mohd Naseem Akhter, Priti P Rege // 2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT). IEEE. — 2020. — P. 1-6.
19. E.I. Andreeva. Document recognition method based on convolutional neural network invariant to 180 degree rotation angle. / E.I. Andreeva, [et al.] //

Информационные технологии и вычислительные системы. — 2019. — Vol. 4. — P. 87-93.

20. Загородников М. В. Восстановление текстового слоя PDF документов со сложным фоном / Загородников М. В., Михайлов А. А. // Труды Института системного программирования РАН. — 2024. — Т. 36, №. 3. — С. 189-202.
21. Wojciech Bieniecki. Image preprocessing for improving ocr accuracy / Wojciech Bieniecki, Szymon Grabowski, Wojciech Rozenberg // 2007 international conference on perspective technologies and methods in MEMS design. IEEE. —2007. — P. 75-80.
22. Binmakhashen G. M. Document layout analysis: a comprehensive survey / Binmakhashen G. M., Mahmoud S. A. // ACM Computing Surveys (CSUR). — 2019. — Vol. 52, no 6. — P. 1-36.
23. Eskenazi S. A comprehensive survey of mostly textual document segmentation algorithms since 2008 / Eskenazi S., Gomez-Krämer P., Ogier J. M. // Pattern recognition. — 2017. — Vol. 64. — P. 1-14.
24. Mao S. Document structure analysis algorithms: a literature survey / Mao S., Rosenfeld A., Kanungo T. // Document recognition and retrieval X. — 2003. — Vol. 5010. — P. 197-207.
25. linkcode Pytesseract Page Segmentation Models (PSMs) [Электронный ресурс]. URL: https://www.kaggle.com/code/dhorvay/pytesseract-page-segmentation-modes-psms (дата обращения: 06.02.2025).
26. Smith R. An Overview of the Tesseract OCR Engine / Smith R. // In proceedings of Document analysis and Recognition. ICDAR 2007. IEEE Ninth International Conference. — 2007. —DOI: 10.1109/ICDAR.2007.4376991.
27. Breuel T.M. The OCRopus open source OCR system / Breuel T.M. // Proceedings of IS&T/SPIE 20-th Annual Symposium. — 2008.
28. ABBYY FineReader Engine. [Электронный ресурс]. URL: https://www.abbyy.com. (дата обращения: 06.02.2025)
29. Kasem M. Deep learning for table detection and structure recognition: A survey / Kasem M. [et al.] // ACM Computing Surveys. – 2022.
30. Arif S. Table detection in document images using foreground and background features / Arif S., Shafait F. // 2018 Digital Image Computing: Techniques and Applications (DICTA). — IEEE, 2018. — P. 1-8.

31. Luo S. Deep structured feature networks for table detection and tabular data extraction from scanned financial document images / Luo S. [et al.] // arXiv preprint arXiv:2102.10287. — 2021.
32. Schreiber S. Deepdesrt: Deep learning for detection and structure recognition of tables in document images / Schreiber S. [et al.] // 2017 14th IAPR international conference on document analysis and recognition (ICDAR). — IEEE, 2017. — Vol. 1. — P. 1162-1167.
33. Sun N. Faster R-CNN based table detection combining corner locating / Sun N., Zhu Y., Hu X. // 2019 international conference on document analysis and recognition (ICDAR). — IEEE, 2019. — P. 1314-1319.
34. Zheng X. Global table extractor (gte): A framework for joint table identification and cell structure recognition using visual context / Zheng X. [et al.] // Proceedings of the IEEE/CVF winter conference on applications of computer vision. — 2021. — P. 697-706.
35. Gilani A. Table detection using deep learning / Gilani A. [et al.] // 2017 14th IAPR international conference on document analysis and recognition (ICDAR). – IEEE, 2017. — Vol. 1. — P. 771-776.
36. Siddiqui S. A. Deeptabstr: Deep learning based table structure recognition / Siddiqui S. A. [et al.] // 2019 international conference on document analysis and recognition (ICDAR). — IEEE, 2019. — P. 1403-1409.
37. Siddiqui S. A. Rethinking semantic segmentation for table structure recognition in documents / Siddiqui S. A. [et al.] // 2019 international conference on document analysis and recognition (ICDAR). — IEEE, 2019. — P. 1397-1402.
38. Khan S. A. Table structure extraction with bi-directional gated recurrent unit networks / Khan S. A. [et al.] // 2019 International Conference on Document Analysis and Recognition (ICDAR). — IEEE, 2019. — P. 1366-1371.
39. Schreiber S. Deepdesrt: Deep learning for detection and structure recognition of tables in document images / Schreiber S. [et al.] // 2017 14th IAPR international conference on document analysis and recognition (ICDAR). — IEEE, 2017. — Vol. 1. – P. 1162-1167.
40. Rashid S. F. Table recognition in heterogeneous documents using machine learning / Rashid S. F. [et al.] // 2017 14th IAPR International conference on document analysis and recognition (ICDAR). — IEEE, 2017. — Vol. 1. — P. 777-782.

41. Deng Y. Challenges in end-to-end neural scientific table recognition / Deng Y., Rosenberg D., Mann G. // 2019 International Conference on Document Analysis and Recognition (ICDAR). — IEEE, 2019. — P. 894-901.
42. Zhong X. Image-based table recognition: data, model, and evaluation / Zhong X., ShafieiBavani E., Jimeno Yepes A. // European conference on computer vision. Cham : Springer International Publishing. — 2020. — P. 564-580.
43. Kasar T. Learning to detect tables in scanned document images using line information / Kasar T. [et al.] // 2013 12th International Conference on Document Analysis and Recognition. — IEEE, 2013. — P. 1185-1189.
44. Zanibbi R. A survey of table recognition: Models, observations, transformations, and inferences / Zanibbi R., Blostein D., Cordy J. R. // Document Analysis and Recognition. — 2004. — Vol. 7. — P. 1-16.
45. Embley D. W. Table-processing paradigms: a research survey / Embley D. W. [et al.] // International Journal of Document Analysis and Recognition (IJDAR). — 2006. — Vol. 8. — P. 66-86.
46. Milosevic N. A framework for information extraction from tables in biomedical literature / Milosevic N. [et al.] // International Journal on Document Analysis and Recognition (IJDAR). — 2019. — Vol. 22. — P. 55-78.
47. Tijerino Y. A. Towards ontology generation from tables / Tijerino Y. A. [et al.] // World Wide Web. — 2005. — Vol. 8. — P. 261-285.
48. Hurst M. F. The interpretation of tables in texts: дис. — 2000. — P. 300.
49. Yu Y. Structextv2: Masked visual-textual prediction for document image pretraining / Yu Y. [et al.] // International Conference on Learning Representations. — 2023.
50. Göbel M. ICDAR 2013 table competition / Göbel M. [et al.] // 2013 12th international conference on document analysis and recognition. — IEEE, 2013. — P. 1449-1453.
51. Gao L. ICDAR 2019 competition on table detection and recognition (cTDaR) / Gao L. [et al.] // 2019 International Conference on Document Analysis and Recognition (ICDAR). — IEEE, 2019. — P. 1510-1515.
52. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals / V. I. Levenshtein. // in Soviet physics doklady. — Vol. 10, no. 8. — 1966. — P. 707-710.
53. Stephen V. Rice. Optical Character Recognition: An Illustrated Guide to the Frontier / Stephen V. Rice, George Nagy, Thomas A. Nartker // Proceedings of SPIE - The

International Society for Optical Engineering. Gonesh Chandra Saha, Bappa Sarkar, Md Habibur Rahman — Vol. 3967. — 1999. — P. 58-69.

54. Gonesh Chandra Saha. Checking the Correctness of Bangla Words using N-Gram / Gonesh Chandra Saha, Bappa Sarkar, Md Habibur Rahman // International Journal of Computer Applications. — Vol. 89, issue 11. — 2014. — P. 2-4.
55. “3.6 Smoothing, Interpolation, and Backoff” / D. Jurafsky, J. H. Martin // Speech and Language Processing. — 3rd ed. — London: Pearson, 2025. — P. 50–52.
56. Article “What are CID or composite fonts?”. [Электронный ресурс]. URL: https://enfocus.my.site.com/customers/s/article/What-are-CID-or-composite-fonts?language=en_US. (дата обращения: 07.02.2025).
57. M.P.Bhuyan. Natural Language Processing based Stochastic Model for the Correctness of Assamese Sentences / M.P.Bhuyan, S.K.Sarma, M. Rahman // 2020 5th International Conference on Communication and Electronics Systems (ICCES). — 2020. — P. 2-4.
58. Chapter 3. N-gram Language Models / D. Jurafsky, J. H. Martin // Speech and Language Processing. — 3rd ed. — London: Pearson, 2025. — P. 37–60.
59. Shigarov A. TabbyPDF: Web-based system for PDF table extraction / Shigarov A. [et al.] // Information and Software Technologies: 24th International Conference, ICIST 2018, Vilnius, Lithuania, October 4–6, 2018, Proceedings 24. — Springer International Publishing, 2018. — P. 257-269.
60. Pdfminer: Why are there (cid:x) values in the textual output?. [Электронный ресурс]. URL: https://pdfminersix.readthedocs.io/en/latest/faq.html#why-are-there-cid-x-values-in-the-textual-output. (дата обращения: 07.02.2025).
61. Apache PDFBox, Frequently Asked Questions (FAQ). [Электронный ресурс]. URL: https://pdfbox.apache.org/2.0/faq.html#text-extraction. (дата обращения: 07.02.2025).
62. Microsoft. End-User-Defined and Private Use Area Characters. [Электронный ресурс]. URL: https://learn.microsoft.com/en-us/windows/win32/intl/end-user-defined-characters. (дата обращения: 07.02.2025).
63. Документация Adobe PDF – Portable document format – Part 1: PDF 1.7. Adobe Systems Incorporated. [Электронный ресурс]. URL:

https://opensource.adobe.com/dc-acrobat-sdk-docs/standards/pdfstandards/pdf/PDF32000_2008.pdf. (дата обращения: 07.02.2025).

64. Unicode. Private Use Characters (Private Use Area). [Электронный ресурс]. URL: https://www.unicode.org/faq/private_use.html. (дата обращения: 07.02.2025).
65. Singh S.. Systematic review of spell-checkers for highly inflectional languages / Singh S., Singh S // Artificial Intelligence Review. — 2020. — Vol. 53, np. 6. — P. 4051-4092.
66. Документация Adobe. Adobe CMap and CIDFont files Specification. [Электронный ресурс]. URL: https://adobe-type-tools.github.io/font-tech-notes/pdfs/5014.CIDFont_Spec.pdf. (дата обращения: 07.02.2025).
67. DocParser. What to do when a PDF document is converted to garbled characters and symbols? [Электронный ресурс]. URL: https://help.docparser.com/hc/en-us/articles/16254860582676-What-to-do-when-a-PDF-document-is-converted-to-garbled-characters-and-symbols. (дата обращения: 07.02.2025).
68. Examples of errors when opening PDF file. [Электронный ресурс]. URL: https://repairit.wondershare.com/file-repair/pdf-not-opening.html. (дата обращения: 07.02.2025).
69. Example of PDF text layer corruption after using PDF editor. [Электронный ресурс]. URL: https://superuser.com/questions/285684/pdf-has-an-extra-blank-in-all-words-after-running-through-ghostscript. (дата обращения: 07.02.2025).
70. Gerhard Paaß. Machine learning for document structure recognition / Gerhard Paaß, Iuliu Konya // In Modeling, Learning, and Processing of Text Technological Data Structures. — Springer, 2011. — P. 221–247.
71. Lewis P. Retrieval-augmented generation for knowledge-intensive nlp tasks / Lewis P. [et al.] // Advances in Neural Information Processing Systems. — 2020. — Vol. 33. — P. 9459-9474.
72. Bentabet N. I. Table-of-contents generation on contemporary documents / Bentabet N. I., Juge R., Ferradans S. // 2019 International Conference on Document Analysis and Recognition (ICDAR), Sydney, NSW, Australia. — 2019. — P. 100-107.

73. Constantin A. PDFX: fully-automated PDF-to-XML conversion of scientific literature / Constantin A., Pettifer S., Voronkov A. // Proceedings of the 2013 ACM symposium on Document engineering. — 2013. — P. 177-180.
74. Ahmad R. Information extraction from PDF sources based on rule-based system using integrated formats / Ahmad R., Afzal M. T., Qadir M. A. // Semantic Web Challenges: Third SemWebEval Challenge at ESWC 2016, Heraklion, Crete, Greece, May 29-June 2, 2016, Revised Selected Papers 3. — Springer International Publishing, 2016. — P. 293-308.
75. Doucet A. Enhancing table of contents extraction by system aggregation / Doucet A. [et al.] // 2017 14th IAPR international conference on document analysis and recognition (ICDAR). — IEEE, 2017. — Vol. 1. — P. 242-247.
76. Kang J. Advancements in Financial Document Structure Extraction: Insights from Five Years of FinTOC (2019-2023) / Kang J. [et al.] // 2023 IEEE International Conference on Big Data (BigData). — IEEE, 2023. — P. 2839-2844.
77. Zaman G. Information extraction from semi and unstructured data sources: A systematic literature review / Zaman G. [et al.] // ICIC Express Letters. — 2020. — Vol. 14, no. 6. — P. 593-603.
78. RAG Strategies - Hierarchical Index Retrieval. [Электронный ресурс]. URL: https://pixion.co/blog/rag-strategies-hierarchical-index-retrieval. (дата обращения: 07.02.2025).
79. Kristen Summers. Automatic discovery of logical document structure. Technical Report. Cornell University: дис. — 1998. — P. 119.
80. Semere Kiros Bitew. Logical structure extraction of electronic documents using contextual information. Master's thesis. University of Twente: дис. — 2018. — P. 65.
81. Yi He. Extracting document structure of a text with visual and textual cues. Master's thesis. University of Twente: дис. — 2017. — P. 66.
82. Anoop M Namboodiri. Document structure and layout analysis / Anoop M Namboodiri, Anil K Jain // In Digital Document Processing. — Springer, 2007. — 29–48.
83. Muhammad Mahbubur Rahman. 2017. Understanding the logical and semantic structure of large documents. / Muhammad Mahbubur Rahman, Tim Finin // arXiv preprint arXiv:1709.00770 — 2017.

84. Hirokazu Igari. Document structure analysis with syntactic model and parsers: Application to legal judgments / Hirokazu Igari, Akira Shimazu, and Koichiro Ochimizu // In JSAI International Symposium on Artificial Intelligence. — Springer, 2011. — P. 126–140.
85. Antoine Doucet. Icdar 2013 competition on book structure extraction / Antoine Doucet, Gabriella Kazai, Sebastian Colutto, and Günter Mühlberger // In 2013 12th International Conference on Document Analysis and Recognition. — IEEE, 2013. — P. 1438–1443.
86. Antoine Doucet. Setting up a competition framework for the evaluation of structure extraction from ocr-ed books / Antoine Doucet, Gabriella Kazai, Bodin Dresevic, Aleksandar Uzelac, Bogdan Radakovic, and Nikola Todic // International Journal on Document Analysis and Recognition (IJDAR). — 2011. — Vol 14, no. 1 — P. 45–52.
87. Rémi Juge. The fintoc-2019 shared task: Financial document structure extraction / Rémi Juge, Imane Bentabet, and Sira Ferradans // In Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019) — 2019. — P. 51–57.
88. Ke TianFinance document extraction using data augmentation and attention / Ke Tian, Zi Jun Pen // In Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019). — 2019. — P. 1–4.
89. Emmanuel Giguet. Daniel@ fintoc-2019 shared task: toc extraction and title detection / Emmanuel Giguet, Gaël Lejeune // In Proceedings of the Second Financial Narrative Processing Workshop (FNP 2019). — 2019. — P. 63–68.
90. Najah-Imane Bentabet. The Financial Document Structure Extraction Shared task (FinToc 2020) / Najah-Imane Bentabet, Rémi Juge, Ismail El Maarouf, Virginie Mouilleron, Dialekti Valsamou-Stanislawski, Mahmoud El-Haj // In Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation. — 2020. — P. 13–22.
91. Tomáš Hercig. 2020. UWB@FinTOC-2020 Shared Task: Financial Document Title Detection / Tomáš Hercig, Pavel Kral // In Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation. COLING, Barcelona, Spain. — 2020. — P. 158–162.
92. Dhruv Premi. AMEX-AI-LABS: Investigating Transfer Learning for Title Detection in Table of Contents Generation. / Dhruv Premi, Amogh Badugu, and Himanshu Sharad Bhatt // In Proceedings of the 1st Joint Workshop on Financial Narrative

Processing and MultiLing Financial Summarisation. COLING, Barcelona, Spain. — 2020. — P. 153–157.

93. Dijana Kosmajac. 2020. DNLP@FinTOC'20: Table of Contents Detection in Financial Documents / Dijana Kosmajac, Stacey Taylor, Mozhgan Saeidi // In Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation. COLING, Barcelona, Spain. — 2020. — P. 169–173.
94. Ismail El Maarouf. The Financial Document Structure Extraction Shared Task (FinTOC2021) / Ismail El Maarouf, Juyeon Kang, Abderrahim Ait Azzi, Sandra Bellato, Mei Gan, and Mahmoud El-Haj // In Proceedings of the 3rd Financial Narrative Processing Workshop. — 2021. — P. 111–119.
95. Christopher Bourez. FinTOC 2021-Document Structure Understanding Christopher Bourez // In Proceedings of the 3rd Financial Narrative Processing Workshop. — 2021. — P. 89–93.
96. Kang J. The financial document structure extraction shared task (FinTOC 2022) / Kang J. [et al.] // Proceedings of the 4th Financial Narrative Processing Workshop@ LREC2022. — 2022. — P. 83-88.
97. Cassotti P. swapuniba@ fintoc2022: Fine-tuning pre-trained document image analysis model for title detection on the financial domain / Cassotti P. [et al.] // Proceedings of the 4th Financial Narrative Processing Workshop@ LREC2022. — 2022. — P. 95-99.
98. S. V. Rice. Measuring the Accuracy of Page-Reading Systems : дис. — 1996. — P. 81.
99. Ray Smith. An overview of the Tesseract OCR engine / Ray Smith // In: Document Analysis and Recognition, ICDAR (2007). — 2007.
100. Gjoreski M. Optical character recognition applied on receipts printed in Macedonian Language / Gjoreski M. [et al.] // International Conference on Informatics and Information Technologies At: Bitola, Macedonia — 2014.
101. Sabir E. Implicit language model in lstm for ocr / Sabir E., Rawls S., Natarajan P. // 2017 14th IAPR international conference on document analysis and recognition (ICDAR). — IEEE, 2017. — Vol. 7. — P. 27-31.
102. Overview of the new neural network system in Tesseract 4.00: [Электронный ресурс]. URL:

https://tesseract-ocr.github.io/tessdoc/tess4/NeuralNetsInTesseract4.00.html (дата обращения: 06.02.2025).

103. CLSTM is an implementation of the LSTM recurrent neural network model in C++: [Электронный ресурс]. URL: https://github.com/tmbdev/clstm (дата обращения: 06.02.2025)

104. Graves A. et al. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks / Graves A. [et al.] // Proceedings of the 23rd international conference on Machine learning. — 2006. — P. 69-376.

105. Долгая краткосрочная память: [Электронный ресурс]. URL: https://neerc.ifmo.ru/wiki/index.php?title=Долгая_краткосрочная_память (дата обращения:06.02.2025)

106. Greff K. LSTM: A search space odyssey / Greff K. [et al.] // IEEE transactions on neural networks and learning systems. — 2016. — Vol. 28, no 10. — P. 2222-2232.

Приложение А. Примеры расширения программного комплекса

Листинг А.1 – Пример добавления нового класса DJVUConverter в систему.

```
import os
from typing import Optional

from dedoc.converters.concrete_converters.abstract_converter import AbstractConverter
from dedoc.utils.utils import get_mime_extension, splitext_


class DjvuConverter(AbstractConverter):

    def __init__(self, config: Optional[dict] = None) -> None:
        super().__init__(config=config)

    def can_convert(self,
                    file_path: Optional[str] = None,
                    extension: Optional[str] = None,
                    mime: Optional[str] = None,
                    parameters: Optional[dict] = None) -> bool:
        _, extension = get_mime_extension(file_path=file_path, mime=mime, extension=extension)
        return extension == ".djvu"

    def convert(self, file_path: str, parameters: Optional[dict] = None) -> str:
        file_dir, file_name = os.path.split(file_path)
        name_wo_ext, _ = splitext_(file_name)
        converted_file_path = os.path.join(file_dir, f"{name_wo_ext}.pdf")
        command = ["ddjvu", "--format=pdf", file_path, converted_file_path]
        self._run_subprocess(command=command, filename=file_name, expected_path=converted_file_path)

        return converted_file_path
```

Листинг А.2 – Пример добавления нового класса PdfReader в систему.

```
from typing import List, Optional

import tabula
from PyPDF2 import PdfFileReader
from pdf_attachment_extractor import PdfAttachmentsExtractor

from dedoc.data_structures import CellWithMeta, LineMetadata
from dedoc.data_structures.line_with_meta import LineWithMeta
from dedoc.data_structures.table import Table
from dedoc.data_structures.table_metadata import TableMetadata
from dedoc.data_structures.unstructured_document import UnstructuredDocument
from dedoc.extensions import recognized_extensions, recognized_mimes
from dedoc.readers.base_reader import BaseReader
from dedoc.utils.utils import get_mime_extension


class PdfReader(BaseReader):

    def __init__(self, config: Optional[dict] = None) -> None:
        super().__init__(config=config)
        self.attachment_extractor = PdfAttachmentsExtractor(config=self.config)

    def can_read(self, file_path: Optional[str] = None, mime: Optional[str] = None, extension: Optional[str] = None,
parameters: Optional[dict] = None) -> bool:
```

```
        mime, extension = get_mime_extension(file_path=file_path, mime=mime, extension=extension)
        return extension in recognized_extensions.pdf_like_format or mime in recognized_mimes.pdf_like_format

    def read(self, file_path: str, parameters: Optional[dict] = None) -> UnstructuredDocument:
        parameters = {} if parameters is None else parameters
        lines = self.__process_lines(file_path)
        tables = self.__process_tables(file_path)
        attachments = self.attachment_extractor.extract(file_path=file_path, parameters=parameters)
        return UnstructuredDocument(lines=lines, tables=tables, attachments=attachments)

    def __process_tables(self, path: str) -> List[Table]:
        dfs = tabula.read_pdf(path, stream=True, pages="all")
        tables = []
        for df in dfs:
            metadata = TableMetadata(page_id=None)
            cells = [[CellWithMeta(lines=[LineWithMeta(line=text_cell)]) for text_cell in row]for row in df.values.tolist()]
            tables.append(Table(cells=cells, metadata=metadata))
        return tables

    def __process_lines(self, path: str) -> List[LineWithMeta]:
        with open(path, "rb") as file:
            lines_with_meta = []
            pdf = PdfFileReader(file)
            num_pages = pdf.getNumPages()
            for page_id in range(num_pages):
                page = pdf.getPage(page_id)
                text = page.extractText()
                lines = text.split("\n")
                for line_id, line in enumerate(lines):
                    metadata = LineMetadata(page_id=page_id, line_id=line_id)
                    lines_with_meta.append(LineWithMeta(line=line, metadata=metadata, annotations=[]))
        return lines_with_meta
```

Листинг А.3 – Пример добавления класса для извлечения признаков из текстовых строк типа документов “научные статьи”.

```
def __init__(self) -> None:
        self.named_item_keywords = ("abstract", "introduction", "relatedwork", "conclusion", "references", "appendix",
"acknowledgements")
        self.caption_keywords = ("figure", "table", "listing", "algorithm")

        self.start_regexps = [
            regexps_item,  # list like 1.
            regexps_digits_with_dots,  # lists like 1.1.1. or 1.1.1
            re.compile(r"^\s*\d+\s"),  # digits and space after them
        ]


def transform(self, documents: List[List[LineWithMeta]], y: Optional[List[str]] = None) -> pd.DataFrame:
        # merge matrices for all documents into one
        result_matrix = pd.concat([self.__process_document(document) for document in documents], ignore_index=True)
        # sort columns names for reproducibility on different systems
        features = sorted(result_matrix.columns)
        return result_matrix[features].astype(float)


def _one_line_features(self, line: LineWithMeta, total_lines: int) -> Iterator[Tuple[str, int]]:
        # visual features
        yield "indentation", self._get_indentation(line)
        yield "spacing", self._get_spacing(line)
        yield "font_size", self._get_size(line)
```

```
        yield "bold", self._get_bold(line)
        bold_percent = self._get_bold_percent(line)
        yield "bold_percent", bold_percent
        yield "fully_bold", int(bold_percent == 1.)
        # textual features
        text = line.line.lower()
        text_wo_spaces = "".join(text.strip().split())
        yield "is_named_item", int(text_wo_spaces in self.named_item_keywords)
        yield "is_caption", len([word for word in self.caption_keywords if word in text_wo_spaces])
        yield "digits_number", sum(c.isdigit() for c in text_wo_spaces)
        yield "at_number", text_wo_spaces.count("@")
        yield "is_lower", int(line.line.strip().islower())
        yield "is_upper", int(line.line.strip().isupper())
        yield from self._start_regexp(line.line, self.start_regexps)
        prefix = get_prefix([DottedPrefix], line)
        yield ("dotted_depth", len(prefix.numbers)) if prefix.name == DottedPrefix.name else ("dotted_depth", 0)
        # statistical features
        yield "text_length", len(text.strip())
        yield "words_number", len(text.strip().split())
        yield "line_id", normalization_by_min_max(line.metadata.line_id, min_v=0, max_v=total_lines)

    def __process_document(self, lines: List[LineWithMeta]) -> pd.DataFrame:
        # features for numbered items
        _, list_features_df = self.list_feature_extractor.one_document(lines)
        list_features_df["list_item"] = self._list_features(lines)

        # other features
        features_dict = defaultdict(list)
        for line in lines:
            for feature_name, feature in self._one_line_features(line, len(lines)):
                features_dict[feature_name].append(feature)
        features_df = pd.DataFrame(features_dict)

        # features normalization
        features_df["indentation"] = self._normalize_features(features_df.indentation)
        features_df["font_size"] = self._normalize_features(features_df.font_size)

        # add features of 3 previous and 3 next neighbor lines
        features_df = self.prev_next_line_features(features_df, 3, 3)

        # merge all features in one matrix
        result_matrix = pd.concat([features_df, list_features_df], axis=1)
        return result_matrix
```

Приложение Б. Акты о внедрении результатов диссертационного исследования

ООО «Интерпроком»
Телефон: (495)781-92-64, Факс: (495) 781-92-64
www.interprocom.ru partner@interprocom.ru
Почтовый адрес: 117105, г. Москва, ул. Нагатинская, дом 1, стр. 5
Юр. адрес: 117218 Москва, ул. Б. Черемушкинская, д. 34, оф. 219
ИНН 7727693181, КПП 772701001, ОГРН 1097746368741, ОКПО 62128012

исх. № 18/25 от 28.01.2025

Для представления в диссертационный совет

АКТ
о внедрении результатов кандидатской диссертационной работы
Оксаны Владимировны Беляевой

Результаты диссертационного исследования Оксаны Владимировны Беляевой на тему «Автоматическое восстановление структуры текстовых документов» использованы ООО “Интерпроком” в подсистеме обработки технологических карт, разрабатываемой на базе платформы ИСП РАН “Талисман”. Разработанный О.В.Беляевой программный комплекс “dedoc” используется в качестве программного компонента в цепочке извлечения структуры информации для последующей алгоритмической обработки и сохранения в реляционной базе данных системы управления производственными активами компании ООО “Интерпроком”.

Разработанный компонент предназначен для автоматического извлечения содержимого и восстановления структуры документов, в частности, технологических карт, представленных в различных форматах, в том числе в виде сканированных документов. Целью подсистемы, разрабатываемой на базе платформы “Талисман”, является исключение ручного труда при обработке технологических документов за счёт извлечения из них набора ключевых сущностей, их структуры и связей для дальнейшей алгоритмической обработки и внесения в реляционную базу данных.

Генеральный директор
ООО “Интепроком”
П.Ю. Кузнецов

“28” января 2025 года

АКТ

о внедрении результатов кандидатской диссертационной работы

Беляевой Оксаны Владимировны

Результаты диссертационного исследования Беляевой Оксаны Владимировны на тему «Автоматическое восстановление структуры текстовых документов» внедрены в Институте международных исследований (ИМИ) федерального государственного автономного образовательного учреждения высшего образования «Московский государственный институт международных отношений (университет) Министерства иностранных дел Российской Федерации» (МГИМО МИД России). Разработанные результаты Беляевой О.В. используются в качестве программного компонента в Системе интеллектуального анализа данных в области международных отношений на базе платформы Талисман (Талисман.МГИМО) в университете МГИМО МИД России. Оператором Системы выступает Лаборатория интеллектуального анализа данных в области международных отношений ИМИ. С помощью Системы подготовлено 6 научных статей, более 100 аналитических материалов в интересах МИД России.

Созданный компонент решает задачу автоматической обработки электронных документов различных форматов, получаемых из разных источников средств массовой информации. Цель системы Талисман.МГИМО — создание в МГИМО МИД России информационно-аналитического хаба для изучения международных отношений, зарубежной общественно-политической и деловой информации. В рамках системы решаются задачи автоматического интеллектуального анализа содержимого документов, включая выявление именованных сущностей, установление связей и мониторинг запрашиваемой информации. Методы, разработанные в диссертационной работе, обеспечивают автоматическое извлечение содержимого электронных документов для последующего интеллектуального анализа в системе (Приложение 1).

Директор Института международных исследований
МГИМО МИД России
“07” февраля 2025 года

М.А. Сучков

Приложение 1

Результаты диссертационного исследования Беляевой О.В., внедрённые в Систее интеллектуального анализа данных МГИМО МИД России

Под руководством диссертанта разработан расширяемый программный комплекс «dedoc», предназначенный для:

- Автоматического извлечения содержимого (текстовой и табличной информации с форматированием) из документов различных форматов. Программный комплекс обрабатывает как хорошо структурированные форматы, такие как DOC/DOCX/HTML/CSV и т.д., так и плохо структурированные форматы, такие как изображения и PDF;
- Автоматического восстановления иерархической структуры из содержимого текстовых документов разного типа;
- Приведения поддерживаемых документов к единому унифицированному виду.

Программный комплекс содержит методы, разработанные Беляевой О.В. в диссертационной работе:

- Метод автоматического извлечения содержимого PDF-документов с использованием проверки текстового слоя, обеспечивающий достоверность извлечения и скорость обработки документов;
- Методы обработки изображений сканированных документов для извлечения текстовой и табличной информации;
- Метод восстановления иерархической структуры из содержимого документов. Метод показывает высокое качество на размеченных документов трех типов и лучшие результаты на наборе данных международного соревнования FINTOC;
- Расширяемая архитектура программного комплекса для добавления поддержки новых форматов и типов документов.

исх. № 20/01-6 от 06.02.2025

Российская академия народного хозяйства и государственной службы при Президенте Российской Федерации (РАНХИГС)
Юридический адрес: 119571, Москва, проспект Вернадского, 82
Почтовый адрес: 119571, Москва, проспект Вернадского, 82

АКТ

о внедрении результатов кандидатской диссертационной работы
Беляевой Оксаны Владимировны

Результаты диссертационного исследования Беляевой Оксаны Владимировны на тему «Автоматическое восстановление структуры текстовых документов» внедрены в систему «Киберпрофессор», используемую в РАНХИГС. В частности, разработанный Беляевой О.В. программный комплекс “dedoc” используется в качестве программного модуля в системе «Киберпрофессор» для автоматической обработки текстовых документов. Программный модуль обеспечивает извлечение текстовой и табличной информации и восстановление структуры выпускных квалификационных работ студентов. Благодаря разработанным методам диссертационной работы обеспечивается автоматическая обработка документов формата PDF в системе «Киберпрофессор». Используемая система «Киберпрофессор» в РАНХИГС предназначена для обеспечения процесса мониторинга написания выпускных квалификационных работ, помощи в формулировании тем, проверки на правильности написания текста, помощи в подборе списка литературы для исследований.

Заведующий Лабораторией интеллектуального анализа данных в области государственного управления ИГСУ РАНХиГС — Панкратов И.Ю.

“06” февраля 2025 года

Закрытое
акционерное общество
ЕС−лизинг

117587, г. Москва,
вн.тер.г. муниципальный округ Чертаново Северное,
ш. Варшавское, д. 125, стр. 1, помещ. 1Н
Тел. (495) 319-58-09 Факс (495) 319-69-90
E-mail: contact@ec-leasing.ru
www.ec-leasing.ru
ОКПО 29484562, ОГРН 1027739072096
ИНН/КПП 7726018586/772601001

Сертифицировано
ГОСТ Р ИСО 9001-2015

от 11.02.2025 № ЕС1-36
На № 127-2025 от 10.02.2025

Директору
Федерального государственного
бюджетного учреждения науки
Институт системного
программирования
им. В.П. Иванникова Российской
академии наук (ИСП РАН)
Аветисяну А.И.

Уважаемый Арутюн Ишханович!

В ответ на Ваше письмо от 10.02.2025 № 127-2025 сообщаем, что в рамках выполнения работ по Договору № 17-03/ИИ от 17 марта 2022 года сотрудниками Института системного программирования им. В.П. Иванникова Российской академии наук (ИСП РАН) был разработан «Сервис обработки изображений документов (Версия 2)» (далее – Сервис), в состав которого были включены результаты диссертационного исследования сотрудника ИСП РАН Беляевой Оксаны Владимировны на тему «Автоматическое восстановление структуры текстовых документов».

В частности, разработанное Беляевой О.В. программное обеспечение “dedoc” используется в качестве программного модуля в Сервисе для автоматической обработки текстовых документов в форматах изображений и PDF. Модуль обеспечивает распознавание текстовой и табличной информации на изображениях документов и восстановление иерархической структуры. Благодаря методам, разработанным в рамках диссертационного исследования, обеспечивается автоматическая обработка изображений сканированных документов.

С учетом вышеизложенного, ЗАО «ЕС-лизинг» сообщает, что «Сервис обработки изображений документов (Версия 2)» был введен в опытную эксплуатацию на основании Акта от 28 июня 2024 года (прилагается).

Приложение: Копия Акта ввода в опытную эксплуатацию на 2 л. в 1 экз.

Генеральный директор
д.т.н., профессор А.В. Шмид

УТВЕРЖДАЮ
Генеральный директор
ЗАО «ЕС-лизинг»

_____ А.В.Шмид

«___» июня 2024 г.

АКТ
ввода в опытную эксплуатацию «Сервиса обработки изображений документов (Версия 2)»

«18» июня 2024 г.

Комиссия в составе:

Председатель	ИТ-директор	Чугунов В.Р.
члены комиссии	Руководитель отдела	Лычагин К.А.
	Руководитель проектов	Архипова Е.А.

Комиссия проверила работоспособность «Сервиса обработки изображений документов (Версия 2)»

1. **Комиссии предъявлены:**

- Программное обеспечение: полная версия «Сервиса обработки изображений документов (Версия 2)», включающая:
  - модуль сервис локализации документов на изображении;
  - сервис классификации изображений текстовых документов;
  - модуль сервис извлечения именованных сущностей (ключевых полей) с использованием заданного пользователем геометрического шаблона документа;
  - использование верифицированных версий библиотек машинного обучения;
- Документация на разработанное ПО в составе:
  - Руководство пользователя сервиса обработки изображений документов № 00140-120 РП1;
  - Руководство программиста сервиса обработки изображений документов № 00140-120 РП2;
  - Руководство программиста сервиса классификации текстовых документов по текстовому содержимому № 00140-96 РП3;
- Программа и методика тестирования № 00140-120 ПМ;

- Протокол тестирования от 16 июня 2024 г.;
- Техническое задание по договору № 17-03/ИИ от 17 марта 2022 года в редакции Дополнительного соглашения № 1 от 3 июля 2023 года.

Комиссии продемонстрирована работоспособность «Сервиса обработки изображений документов (Версия 2)».

2. Комиссия установила:

«Сервис обработки изображений документов (Версия 2)» полностью соответствует требованиям Технического задания

3. Вывод

«Сервис обработки изображений документов (Версия 2) полностью соответствует требованиям Технического задания и введен в опытную эксплуатацию.

Председатель комиссии В.Р.Чугунов

Члены комиссии К.А.Лычагин

Е.А.Архипова

Приложение В. Свидетельства о государственной регистрации программ и ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2020666950

«Docreader»

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт системного программирования им. В.П. Иванникова Российской академии наук (RU)***

Авторы: ***Козлов Илья Сергеевич (RU), Беляева Оксана Владимировна (RU), Перминов Андрей Игоревич (RU), Богатенкова Анастасия Олеговна (RU)***

Заявка № **2020666254**
Дата поступления **10 декабря 2020 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***18 декабря 2020 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2020667079

«Dedoc»

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт системного программирования им. В.П. Иванникова Российской академии наук (RU)***

Авторы: ***Козлов Илья Сергеевич (RU), Беляева Оксана Владимировна (RU), Перминов Андрей Игоревич (RU), Богатенкова Анастасия Олеговна (RU)***

Заявка № **2020666199**

Дата поступления **10 декабря 2020 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***21 декабря 2020 г.***

Руководитель Федеральной службы по интеллектуальной собственности

Г.П. Ивлиев

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023688585

«Puredoc: сервис обработки изображений документов»

Правообладатель: ***Федеральное государственное бюджетное учреждение науки Институт системного программирования им. В.П. Иванникова Российской академии наук (RU)***

Авторы: ***Беляева Оксана Владимировна (RU), Богатенкова Анастасия Олеговна (RU), Перминов Андрей Игоревич (RU), Голодков Александр Олегович (RU), Шевцов Никита Сергеевич (RU), Рахматуллаев Темурбек Анасбекович (UZ), Михайлов Андрей Анатольевич (RU), Зыкин Ярослав Ильич (RU)***

Заявка № **2023688256**

Дата поступления **15 декабря 2023 г.**

Дата государственной регистрации в Реестре программ для ЭВМ ***22 декабря 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов