СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во время обучения студентам не раз приходится писать отчеты к различным видам работ (курсовые, лабораторные, научно-исследовательские работы и т.п.), при этом все эти работы должны быть своевременно проверены и оценены, а также, возможно, отправлены на доработку. Однако, количество студентов намного превышает количество нормоконтроллеров, которые оценивают работы, чтобы ускорить процесс оценивания возможно использование автоматических систем проверки, которые могут генерировать отчет, содержащий результаты проверки работы на наличие наиболее распространенных видов ошибок.

Целью данной научно-исследовательской работы является создание прототипа системы автоматической проверки работ студентов.

Для достижения поставленной цели требуется решить следующие задачи:

- проанализировать существующие виды PDF-документов и связанных с ними ограничений;
- классифицировать типовые требования и ошибки при оформлении отчётов: текста, рисунков, графиков, схем алгоритмов, таблиц, списка источников и т.д.;
- проанализировать существующие решения и разработать алгоритм выделения составных частей (элементов) отчёта, представленного в формате PDF, в соответствии с ГОСТ 7.32 (фрагменты текста, рисунки, графики, схемы алгоритмов, источники, таблицы и пр.) для дальнейшего анализа с использованием средств компьютерного зрения и автоматического анализа текста;
- проанализировать существующие решения и разработать алгоритм проверки рисунков на соответствие ГОСТ 7.32 и дополнительным требованиям;
- проанализировать существующие решения и разработать алгоритм классификации рисунков по содержанию: графики, схемы алгоритмов, UML-диаграммы, IDEF0, BPMN2.0 и прочие изображения;

– проанализировать существующие решения и разработать алгоритм проверки схемы алгоритма на соответствие ГОСТ 7.32 и дополнительным требованиям;

– проанализировать существующие решения и разработать алгоритм проверки текста и списка используемых источников на соответствие ГОСТ 7.32 и дополнительным требованиям;

– реализовать предложенные алгоритмы в едином ПО для целевой ОС «Astra Linux» и «ROSA Linux».

1 Аналитический раздел

1.1 Структура PDF файла

1.1.1 Представление PDF файла

PDF документ имеет иерархическую структуру (дерево), корнем которого является словарь Catalog. Визуализацию данного дерева можно рассмотреть на рисунке 1.1.

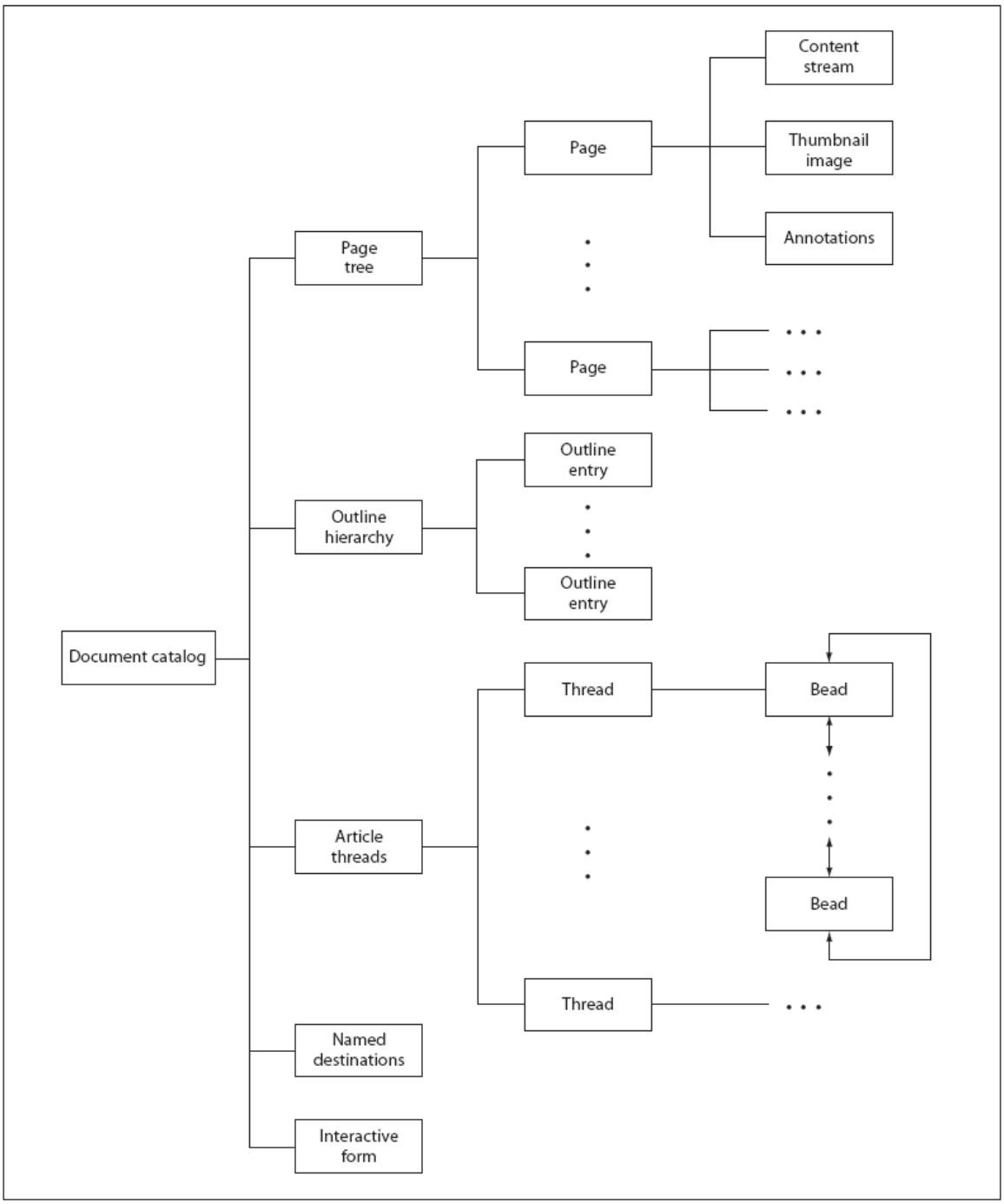


Рисунок 1.1 – Пример дерева PDF документа

Каталог содержит ссылки на вершины описания страниц. Поддеревья страниц отсортированны, что позволяет быстро находить необходимую страницу. Словарь каждой страницы хранит ссылку на словарь ресурсов, который хранит требуемые шрифты, изображения т. д. [pdf_object_def].

1.1.2 Разделы PDF файла

Структура PDF файла включает 4 раздела:

1) заголовок;

2) тело;

3) таблица перекрестных ссылок;

4) хвост [pdf_object_def].

Рассмотрим каждый раздел по отдельности.

Заголовок

Заголовком называется первая строка файла. Она содержит информацию о версии PDF [pdf_object_def]. Пример заголовка выглядит как %PDF-1.5.

Тело

Все содержимое документа находится в теле файла. Информация, которая отображается пользователю представлена восемью типами данных:

1) булевы значения. Принимают значения true или false);

2) числа. Включают два типа данных — integer (целочисленный) и real (вещественный). Дробная часть в вещественных числах отделяется точкой;

3) имена. Представляют собой последовательность ASCII символов. Они начинаются со слеша, который не входит в имя. Вместо непосредственно символов могут включать их шестнадцатеричные коды, начинающиеся с символа #;

4) строки. Ограничены длиной в 65535 байтов. Записываются в круглых либо треугольных скобках. Могут быть представлены как ASCII символами, так и шестнадцатеричными или восьмеричными кодами.

5) массивы. Могут содержать любые PDF-объекты. Элементы разделяются пробелом и заключаются в квадратные скобки;

6) словари. Представляют коллекцию пар ключ-значение. Ключом должно быть имя, а значением может быть любой объект. Запись словаря начинается с символов «, а заканчиваются — »;

7) потоки. Потоки содержат неограниченные последовательности байтов. В них содержится основное содержимое документов. Поток начинается с ключевого слова stream и заканчивается словом endstream. Перед началом потока записывается словарь с мета-информацией. Он включает данные о количестве байтов, фильтре применимом их к обработке и так далее;

8) null-объекты. Представляются ключевым словом null [pdf_object_def].

PDF объектом является любой вышеперечисленный тип, содержащий информацию [pdf_object_def]. Пример «хвоста» PDF файла приведен в листинге А.1.

Хвост

Данный раздел начинается с ключевого слова trailer и содержит:

1) словарь;

2) смещение относительно таблицы перекрестных ссылок (англ. cross-reference table);

3) маркер конца файла %%EOF.

В словарь данного раздела входят:

1) Данные о количестве объектов (ключевое слово Size);

2) ссылки на каталог документа (ключевое слово Root);

3) информационный словарь (ключевое слово Info);

4) идентификатор файла (ключевое слово ID) [pdf_object_def].

Пример «хвоста» PDF файла приведен в листинге А.2.

Таблица перекрестных ссылок

Cross-reference table позволяет получать произвольный доступ к любому объекту в файле. Данная таблица состоит из секций. Каждая секция соответствует новой версии документа, данная таблица начинается с ключевого слова xref, так что иногда ее называют xref таблицей [pdf_structure_trans].

Листинг 1.1 – Пример таблицы перекрестных ссылок

```
xref
0 44
0000000000 65535 f
0000000361 00000 n
0000000257 00000 n
0000000015 00000 n
```

Любой PDF-объект может быть помечен уникальным идентификатором и использоваться как ссылка. Такие объекты называются косвенными. Они начинаются с идентификатора, номера поколения и ключевого слова obj. Заканчивается косвенный объект словом endobj. На эти объекты можно ссылаться в таблице cross-reference table и любом другом объекте (для этого используется символ R) [pdf_structure_trans].

1.2 Виды PDF форматов

В данной части работы будут проанализированы существующие виды PDF документов. Существует несколько различных видов PDF документов, каждый из которых имеет свои особенности и ограничения:

1) PDF/A;

2) PDF/X;

3) PDF/E;

4) PDF/UA.

Рассмотрим каждый из них по отдельности.

1.2.1 PDF/A

Данный формат, предназначенный для долгосрочного хранения документов. Он обеспечивает сохранность и неприкосновенность содержимого даже через длительные периоды времени. Однако, PDF/A ограничен в функциональности и не поддерживает некоторые расширенные возможности форматов PDF [pdf_levels_std]. Данный формат также разделяется на несколько подклассов: PDF/A-1, PDF/A-2, PDF/A-3, PDF/A-4.

Также вводится новое понятие уровня соответствия, оно накладывает дополнительные требования на классы PDF/A, для предоставления дополнительных возможностей. Расссмотрим уровни соответствия.

1) Уровень b (Basic). Цель: обеспечение надёжного воспроизведения внешнего вида документа. Распространяется на файлы формата: PDF/A-1b, PDF/A-2b, PDF/A-3b;

2) уровень a (Accessible). Цель: обеспечение возможности поиска и преобразования содержимого документа. Включает все требования уровня b и дополнительно требует, чтобы была включена структура документа. Также вводит требования:

    1) Содержимое должно быть помечено деревом иерархической структуры, что означает, что такие элементы, как порядок чтения, рисунки и таблицы, явно идентифицируются с помощью метаданных.
    2) Должен быть указан естественный язык документа.
    3) Изображения и символы должны иметь альтернативный описательный текст. Файл должен включать сопоставление символов с Unicode.

    Распространяется на файлы формата: PDF/A-1a, PDF/A-2a, PDF/A-3a;

3) уровень u (Unicode). Распространяется на файлы формата: PDF/A-2u, PDF/A-3u. Требуется сопоставление символов с Unicode. Изменения: отбрасываются требования уровня a, включая встроенную логическую структуру (т. е. теги и дерево структур);

4) уровень f (Format). Распространяется на файлы формата: PDF/A-4f. Изменения: позволяет встраивать типы файлов любого другого формата;

5) уровень e (Engineering). Распространяется на файлы формата: PDF/A-4e. Изменения: поддержка аннотаций типов RichMedia и 3D [pdf_levels_std].

PDF/A-1

PDF/A-1 - самый распространенный формат оригинального PDF/A на сегодняшний день. Он основан на PDF 1.4 и является наиболее ограниченным, так как не поддерживает JPEG 2000, вложения, слои и прозрачность. Часть 1 стандарта была опубликована 28 сентября 2005 года и определяет два уровня соответствия для файлов PDF: PDF/A-1b и PDF/A-1b [pdf_a_2].

PDF/A-2

PDF/A-2 предоставляет собой ряд новых функций:

1) сжатие JPEG2000, что особенно полезно для отсканированных документов, таких как карты, книги, а также документов с цветным содержимым, таких как чеки или паспорта;

2) вложенные файлы PDF/A через коллекции: Acrobat позволяет пользователям создавать коллекции (иногда также называемые "портфелями"), где несколько документов PDF/A объединяются в один "контейнерный"документ PDF;

3) необязательное содержимое (слои): Необязательное содержимое, иногда также называемое слоями, полезно для приложений картографии или инженерных чертежей, где отдельные слои могут быть показаны или скрыты в соответствии с требованиями просмотра;

4) новый уровень соответствия PDF/A-2u - "u"для Unicode. Он упрощает поиск и копирование текста Unicode для цифровых PDF-документов и PDF-документов, которые были отсканированы с последующим оптическим распознаванием символов (OCR);

5) метаданные на уровне объекта XMP: PDF/A-2 определяет требования к настраиваемым метаданным XMP;

6) цифровые подписи: В то время как PDF/A-1 уже позволяет использовать цифровые подписи, PDF/A-2 определяет правила, которые должны быть применены для гарантии взаимодействия [pdf_a_2].

PDF/A-3

PDF/A-3 полностью аналогичен PDF/A-2, однако поддерживает добавление любых файлов, а не только PDF типа A. Однако не гарантирует валидность их прочтения в будущем [pdf_a_2].

Также стоит отметить, что файлы данного вида возможно использовать в электронном документообороте [nalogi].

PDF/A-4

Основное отличие данного вида, является замена уровней соответствия b и u с целью упростить стандарт. PDF/A-4 требует отображения в Юникоде для всех шрифтов в любое время [pdf_a_4].

1.2.2 PDF/X

Формат, разработанный специально для обмена и печати документов в издательской отрасли. Он обеспечивает точность цветов и расположения элементов страницы, что особенно важно при печати. Однако, PDF/X имеет ограниченные возможности вставки мультимедийных элементов и интерактивности [abdobe_PDF].

1.2.3 PDF/E

Формат, предназначенный для обмена и хранения документов в инженерной отрасли. Он поддерживает вставку трехмерных моделей, векторных изображений и других инженерных элементов. Однако, PDF/E может быть ограничен в возможности обработки сложных макетов и мультимедийных элементов [abdobe_PDF].

1.2.4 PDF/UA

Формат, предназначенный для создания доступных документов для пользователей с ограниченными возможностями. Он обеспечивает структурированное представление контента и поддержку технологий чтения вслух и управления навигацией. Однако, PDF/UA может иметь ограничения в отображении сложных макетов и интерактивных элементов [abdobe_PDF].

1.3 Основные ошибки в отчетах

В данном разделе будут рассмотрены наиболее часто встречающиеся ошибки, которые совершают студенты при написании различных отчетов.

1.3.1 Общие ошибки

Выход за границы листа является одной из самых распространенных ошибок. В ГОСТ 7.32 указаны следующие размеры полей: левое — 30 мм, правое — 15 мм, верхнее и нижнее — 20 мм [GOST732].

Каждый объект (например: таблица, рисунок, схема алгоритма, формула) должен быть подписан и пронумерован, однако более подробно подписи к каждому из них будут рассмотрены в следующих разделах.

Если таблица или схема не влезает на одну страницу, то она разбивается на несколько частей, каждая из них должна быть подписана.

1.3.2 Ошибки в тексте

В предыдущем разделе уже были рассмотрены поля документа, однако во время оформления текста отчетов могут возникнуть и другие ошибки.

Слова в тексте должны быть согласованы в роде, числе и падеже.

Страницы отчета должны быть пронумерованы, однако, номер на титульном листе не ставится (но он является первой страницей, это означает, что следующая страница должна иметь номер 2).

Ненумерованный заголовок (введение, список литературы, оглавление и т.п.) должен быть выравнен по центру, при этом он состоит только из прописных букв.

Абзацный отступ должен быть одинаковым по всему тексту отчета и равен 1,25 см [GOST732].

Возможна потеря научного стиля и переход к публицистике, что является ошибкой. Также текст работы должен быть написан на государственном языке.

1.3.3 Ошибки в рисунках

Каждый рисунок должен быть подписан, при этом подпись должна располагаться строго по центру, внизу рисунка.

Все рисунки должны быть выполнены в высоком качестве, если обратное

не требуется в самой работе.

Если рисунок не вмещается в ширину страницы, то допускается повернуть его таким образом, чтобы верх рисунка был ближе к левой части страницы.

Ошибки в графиках

Для каждого графика должна существовать легенда, для оформления которой есть два варианта:

1) в одном из углов графика находится область, в которой указаны все обозначения;

2) в подписи к графику описано каждое обозначение.

Должны быть подписаны единицы измерения каждой из осей (даже в том случае, если на гистограмме оси подписываются словами, например, если измерение идет в штуках или на оси обозначены времена года).

Отчеты могут быть напечатаны в черно-белом варианте, поэтому на графиках должны быть маркеры, которые позволят отличить графики друг от друга даже не в цветом варианте.

При большом количестве графиков на одном рисунке возможна ситуация, при которой невозможно отличить один график от другого, что является ошибкой.

Ошибки в схемах алгоритмов

Если схема не влезает на одну страницу, то она разбивается на несколько частей, каждая из них должна быть подписана. Для разделения схемы алгоритма на части используется специальный символ-соединитель, который отображает выход в часть схемы и вход из другой части этой схемы, соответствующие символы-соединители должны содержать одно и то же уникальное обозначение.

Довольно часто вместо символа начала или конца алгоритма используют овал, однако в этом случае должен быть использован прямоугольник с закругленными углами.

Также при использовании символа процесса (прямоугольник) используют прямоугольник с закругленными углами.

При соединении символов схемы алгоритмов не нужны стрелки, если они соединяют символы в направлении слево-направо или сверху-вниз, в остальных случаях символы должны соединяться линиями со стрелкой на конце.

При использовании символа процесса—решение как минимум одна из соединительных линий должна быть подписана, однако возможен также вариант, когда подписаны обе линии.

Пояснительный текст не должен пересекаться с символами, использующимися для составления схем.

1.3.4 Ошибки в таблицах

Каждая таблица должна быть подписана. Наименование следует помещать над таблицей слева, без абзацного отступа в следующем формате: Таблица Номер таблицы - Наименование таблицы. Наименование таблицы приводят с прописной буквы без точки в конце[GOST732].

Таблицу с большим количеством строк допускается переносить на другую страницу. При переносе части таблицы на другую страницу слово «Таблица», ее номер и наименование указывают один раз слева над первой частью таблицы, а над другими частями также слева пишут слова «Продолжение таблицы» и указывают номер таблицы[GOST732].

1.3.5 Ошибки в формулах

Каждая формула должна быть пронумерована вне зависимости от того, есть ли на нее ссылка в тексте или нет. Нумерация может осуществляться в двух вариантах:

1) сквозная нумерация (номер формулы не зависит от раздела, в котором она находится);

2) нумерация, зависящая от раздела (в том случае номер формулы начинается с номера раздела).

После каждой формулы должен находиться знак препинания (точка, запятая и т.п.).

Если в формуле содержится система уравнений, то после каждого из них (за исключением последнего) ставится запятая, а после последнего — точка,

либо запятая.

Номер формулы должен быть выравнен по правому краю страницы и находиться по центру формулы.

Если формула вставляется в начале страницы, то часто перед ней может присутствовать отступ, которого быть не должно.

1.3.6 Ошибки в списках

Ненумерованные списки должны начинаться с удлиненного тире.

В нумерованных списках после номера пункта обязательно должна стоять скобка.

В конце каждого пункта списка должен быть знак препинания, от которого зависит первая буква первого слова следующего пункта:

– если пункт заканчивается на точку, то первое слово следующего пункта должно начинаться на прописную букву;

– если пункт заканчивается запятой или точкой с запятой, то следующий первое слово следующего слово должно начинаться со строчной буквы.

1.3.7 Ошибки в списке литературы

Часто при описании одного из источников не указывается одна из составных частей (автор, издательство и т.п.).

Также нередко встречаются ссылки на так называемые «препринтовские» издательства (сама статья еще не вышла).

Введение

Рисунок 1.2 – Пример ошибочного оформления ненумерованного заголовка

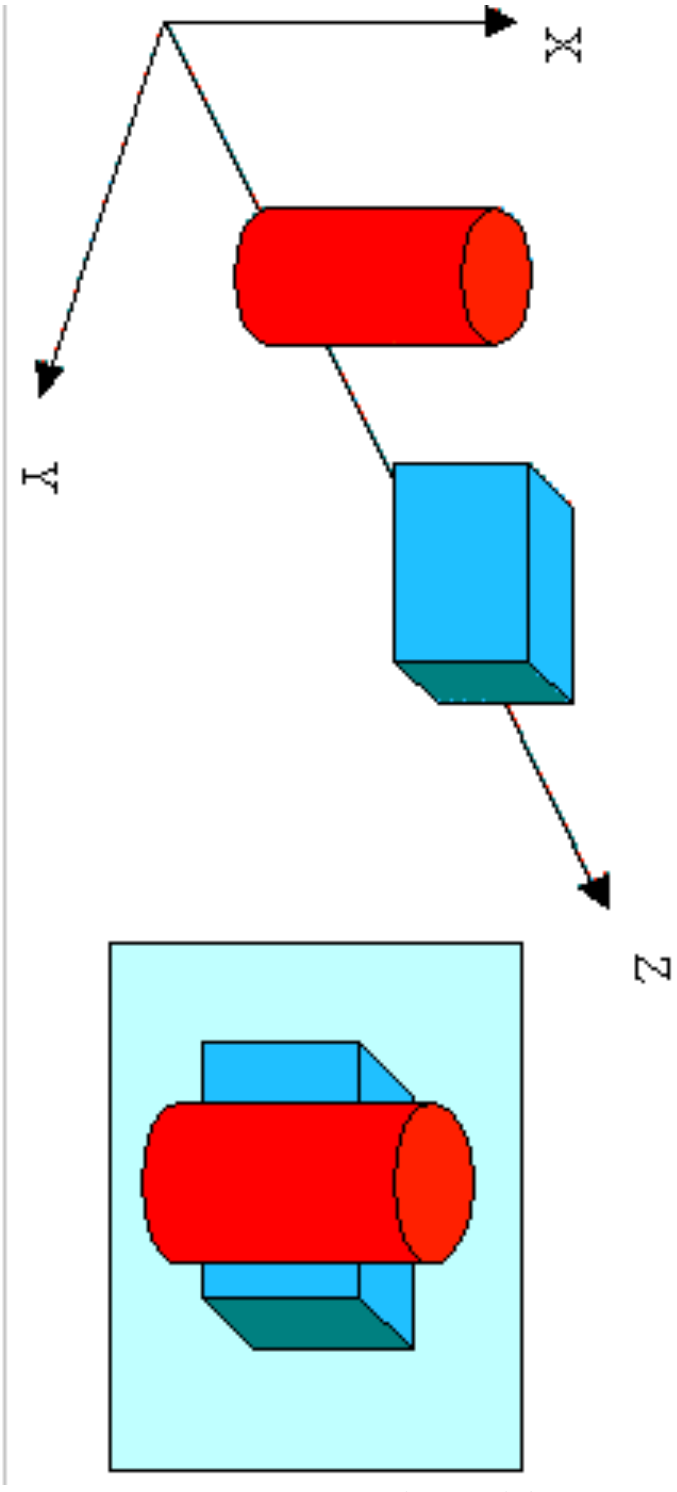


Рисунок 1 - Пример работы Z-буфера

Рисунок 1.3 – Пример ошибочного оформления рисунка — некорректный поворот

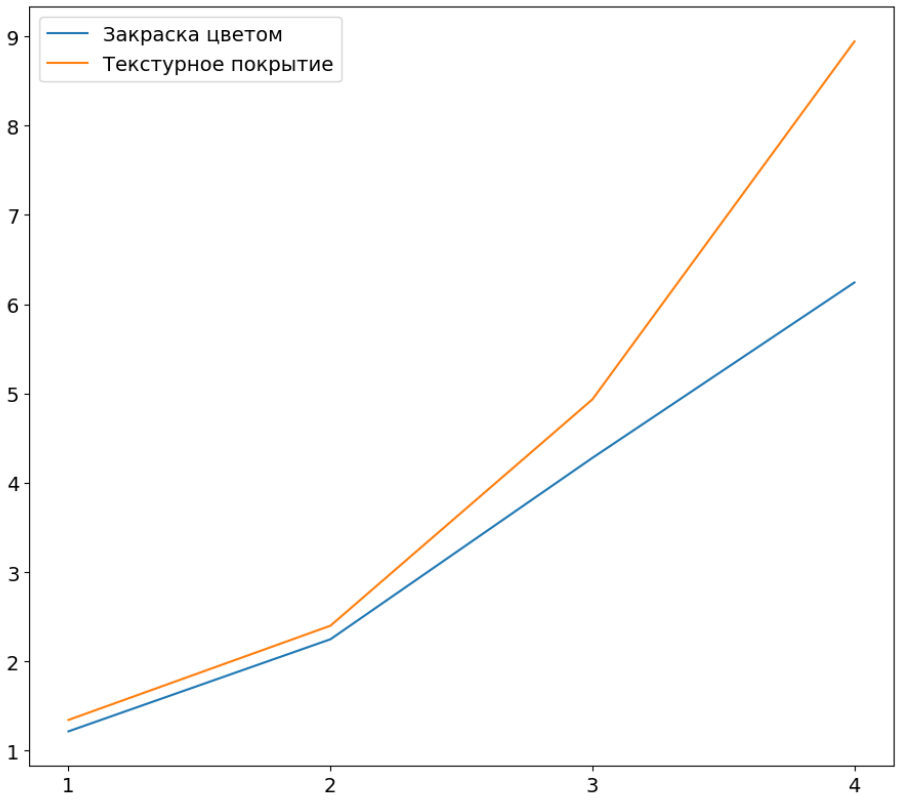


Рисунок 1.4 – Пример ошибочного оформления графика — отсутствуют единицы измерения

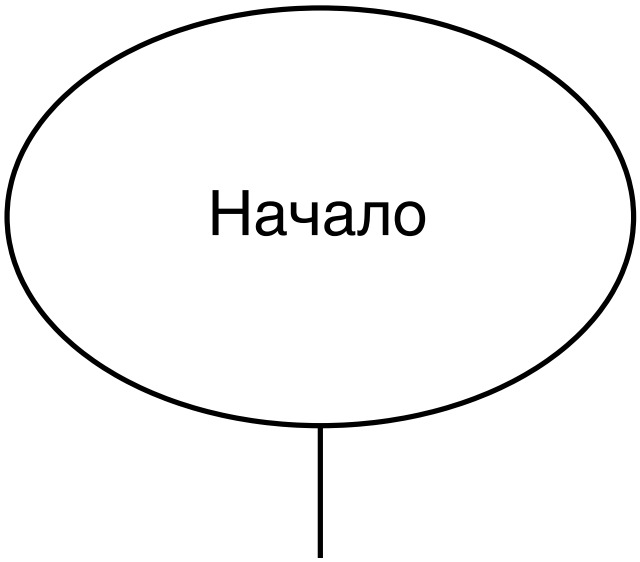


Рисунок 1.5 – Пример ошибочного оформления схемы — некорректный символ начала

Рисунок 1.6 – Пример ошибочного оформления схемы — некорректный символ процесса

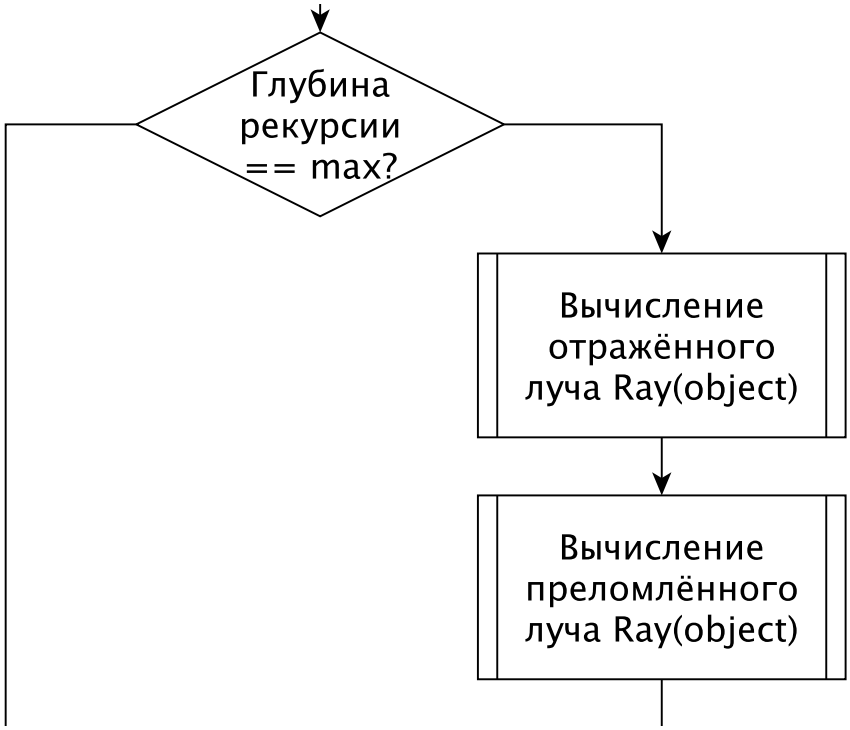


Рисунок 1.7 – Пример ошибочного оформления схемы — не подписана ни одна из веток символа процесса—решение

$$D(i,j) = \begin{cases} 0 & \text{если } i = 0, j = 0 \\ j & \text{если } i = 0, j > 0 \\ i & \text{если } j = 0, i > 0 \\ min(min(D(i,j-1)+1 & \\ \quad D(i-1,j)+1) & \\ \quad D(i-1,j-1) + m(S_1[i], S_2[j]) & \\ \quad \begin{bmatrix} D(i-2,j-2)+1 & \text{если } i > 1, j > 1, \\ & S_1[i-1] == S_2[j-2], \\ & S_1[i-2] == S_2[j-1] \end{bmatrix} & \text{иначе} \end{cases} \quad (1)$$

Рисунок 1.8 – Пример ошибочного оформления системы уравнений — отсутствуют знаки препинания после уравнений

- One
- Two
- Three

Рисунок 1.9 – Пример ошибочного оформления ненумерованного списка — некорректный символ перед элементами списка

— Первый,

— Второй,

— Третий.

Рисунок 1.10 – Пример ошибочного оформления ненумерованного списка — некорректный регистр буквы следующего пункта после запятой в предыдущем

1. первый,
2. второй,
3. третий.

Рисунок 1.11 – Пример ошибочного оформления нумерованного списка — некорректный символ после номера элемента списка

1.4 Библиотеки по работе с PDF-файлами

В данной части работы будут сравниваться существующие Python-библиотеки для извлечения данных из PDF-файлов, охватывая из возмож-

ности с точки зрения извлечения текста, изображений и таблиц, скорости выполнения и обширности функциональности.

1.4.1 PyPDF2

PyPDF2 - это библиотека на чистом Python, которая позволяет читать PDF-файлы и манипулировать ими. Хотя она в основном ориентирована на извлечение текста, она также предоставляет ограниченную поддержку для извлечения изображений. Однако извлечение таблиц не является встроенной функцией. PyPDF2 получил широкое распространение благодаря небольшой, но достаточной, функциональности и обширной документации.

1.4.2 pdfminer.six

pdfminer.six - это поддерживаемая сообществом библиотека Python, основанная на оригинальном проекте PDFMiner. Она предлагает расширенные возможности для извлечения текста из PDF-файлов, включая возможность извлекать информацию о макете текста. Однако она не обеспечивает прямой поддержки извлечения изображений или таблиц. pdfminer.six известен своей точностью при извлечении текста, так как была специально разработана для его извлечения из PDF-файлов.

1.4.3 PyMuPDF

PyMuPDF - это привязка Python для библиотеки MuPDF, которая известна своими высокопроизводительными возможностями рендеринга и синтаксического анализа. PyMuPDF предлагает обширные возможности для извлечения как текста, так и изображений из PDF-файлов. Хотя он не обеспечивает встроенного извлечения таблиц, он обеспечивает прочную основу для реализации пользовательских алгоритмов извлечения таблиц. PyMuPDF полностью документирован и предоставляет богатый набор функциональных возможностей.

2 Конструкторский раздел

2.1 Описание системы автоматической проверки отчета

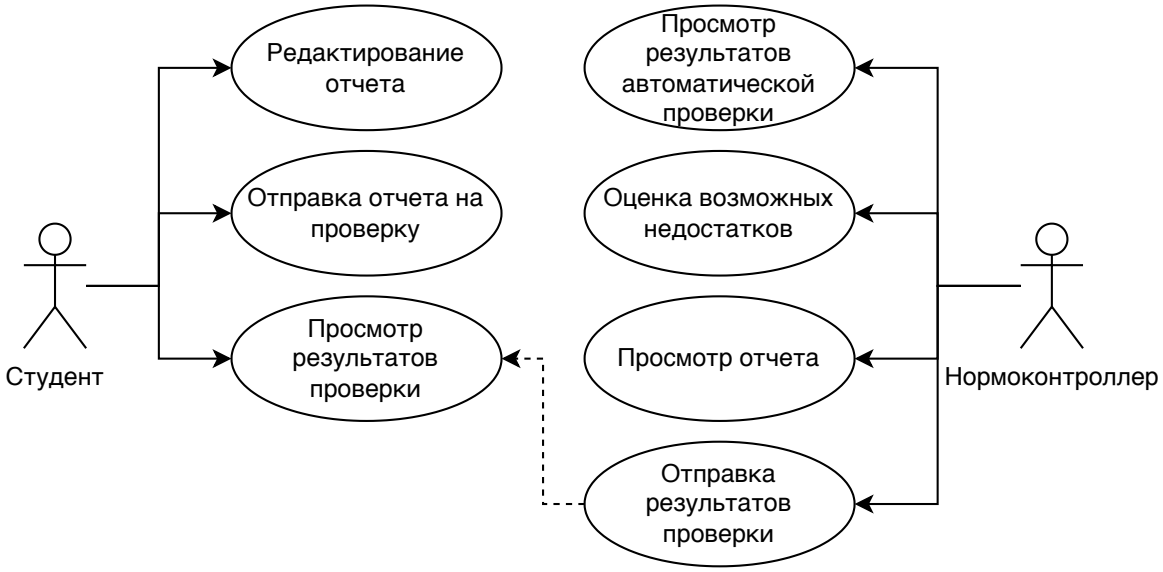


Рисунок 2.1 – Диаграмма вариантов автоматической проверки отчета

Рисунок 2.2 – Диаграмма последовательности действий

С помощью использования алгоритма автоматической проверки отчета

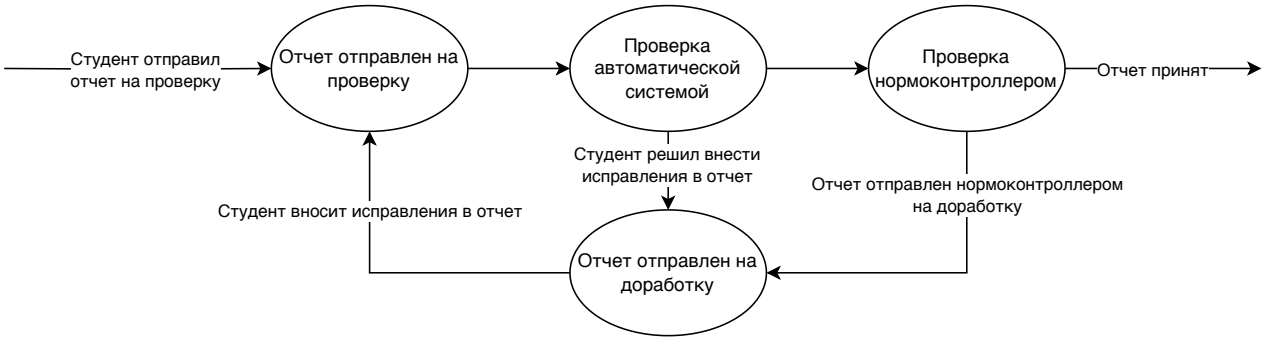


Рисунок 2.3 – Диаграмма состояний проверки отчета

возможно существенно сократить временные ресурсы, выделяемые нормоконтроллером на проверку огромного количества отчетов, однако, полностью отказаться от финального контроля результатов человеком невозможно, таким образом существует две роли при проверки отчета на соответствие ГОСТ, а именно: студент и нормоконтроллер.

Студент отправляет отчет на проверку, а затем получает результат со списком ошибок (если имеются). Нормоконтроллер же анализирует отчет, составленный автоматической системой проверки, и при необходимости может внести необходимые правки.

3 Технологический раздел

3.1 Средства реализации

3.1.1 Используемые библиотеки

OpenCV

OpenCV (Библиотека компьютерного зрения с открытым исходным кодом) - это библиотека программного обеспечения для компьютерного зрения и машинного обучения с открытым исходным кодом.

Библиотека содержит более 2500 оптимизированных алгоритмов, которые включают в себя полный набор как классических, так и самых современных алгоритмов компьютерного зрения и машинного обучения. Эти алгоритмы могут быть использованы для обнаружения и распознавания лиц, идентификации объектов, классификации действий человека в видео, отслеживания движений камеры, отслеживания движущихся объектов, поиска похожих изображений из база данных изображений, распознавание пейзажа и т. д. [about_openCV].

Данная библиотека реализуют следующий функционал:

1) поиск по шаблону (англ. template matching), данная функция позволяет находить на изображении большего размера шаблон меньшего размера и выделять его, данная функция упростит поиск геометрических примитивов на изображении [pattern_matching];

2) классификация изображений из модуля глубоких нейронных сетей (англ. dense neural networks module) позволит разбивать изображения на необходимые подклассы [DL_openCV].

3) Благодаря оптическому распознаванию текста (англ. optical character recognition) возможно определение местоположения и получение информации о содержании текста [OCR_openCV].

3.1.2 YOLO

Популярная модель обнаружения объектов и сегментации изображений YOLO (You Only Look Once) была разработана Джозефом Редмоном и

Али Фархади из Вашингтонского университета. YOLOv8, использующаяся в данной работе является эволюцией серии моделей YOLO [YOLOv8].

1) Модель YOLOv2, выпущенная в 2016 г., была усовершенствована за счет использования пакетной нормализации, якорных блоков и размерных кластеров.

2) YOLOv3, выпущенная в 2018 году, позволила еще больше повысить производительность модели за счет использования более эффективной опорной сети, множества якорей и объединения пространственных пирамид.

3) YOLOv4, выпущенная в 2020 году, обзавелась такими инновациями, как увеличение данных Mosaic, новая головка обнаружения без якорей и новая функция потерь.

4) YOLOv5 позволила еще больше повысить производительность модели, в этой версии быди добавлены такие новые возможности, как оптимизация гиперпараметров, интегрированное отслеживание экспериментов.

5) YOLOv6 была открыта компанией Meituan в 2022 году и используется во многих автономных роботах-доставщиках компании.

6) В YOLOv7 добавлены дополнительные задачи, такие как оценка позы по набору данных COCO keypoints.

7) YOLOv8 - это последняя версия YOLO от Ultralytics. Являясь передовой, современной (SOTA) моделью, YOLOv8 опирается на успех предыдущих версий, представляя новые возможности и улучшения для повышения производительности, гибкости и эффективности. YOLOv8 поддерживает полный спектр задач искусственного интеллекта, включая обнаружение, сегментацию, оценку положения, отслеживание и классификацию. Такая универсальность позволяет пользователям использовать возможности YOLOv8 в различных приложениях и областях.

4 Исследовательский раздел

В данном разделе будут рассмотрены методы классификации и проверки выбранных объектов документа на валидность.

4.1 Анализ изображений

Для анализа изображений (таблиц, схем, списка информационных ресурсов) необходимо получить их представление из отчета. Так как изображения могут быть представлены в векторном формате, то необходимо решать задачу детекции изображений.

4.1.1 Использование соответствия по шаблону

Предположение: наличие отличительных объектов на картинке, не встречающийся в других (например, оси для графиков) позволит классифицировать объект. Для поиска объектов используется поиск по шаблону [pattern_matching].

Однако объект, представленный на изображении может находиться в любом положении и под любым наклоном, таким образом для поиска соответствия необходимо рассматривать все возможные повороты шаблона. Например при необходимости поиска изображения стрелки в изображении 4.1 с использованием шаблона 4.2. При использовании метода CV-TM-CCOEFF-NORMED — корреляция Пирсона, результаты представлены на изображении 4.3, перед использованием шаблона изображение было переведено в вид оттенков серого(один канал). Было найдено только одно изображение стрелки, без учета ее поворота, что не позволяет использовать данный метод при всевозможных ее поворотах.

4.1.2 Использование YOLOv8 для детекциии изображений

Для детекции изображений была использована модель YOLOv8 [YOLOv8]. Для разметки изображений был использован labelimg [labelimg]. Данные для разметки были взяты из отчетов студентов по предмету Анализ Алгоритмов. Было выделено 5 классов изображений:

1) формулы(имеют метку eq);

2) схема(имеют метку scheme);

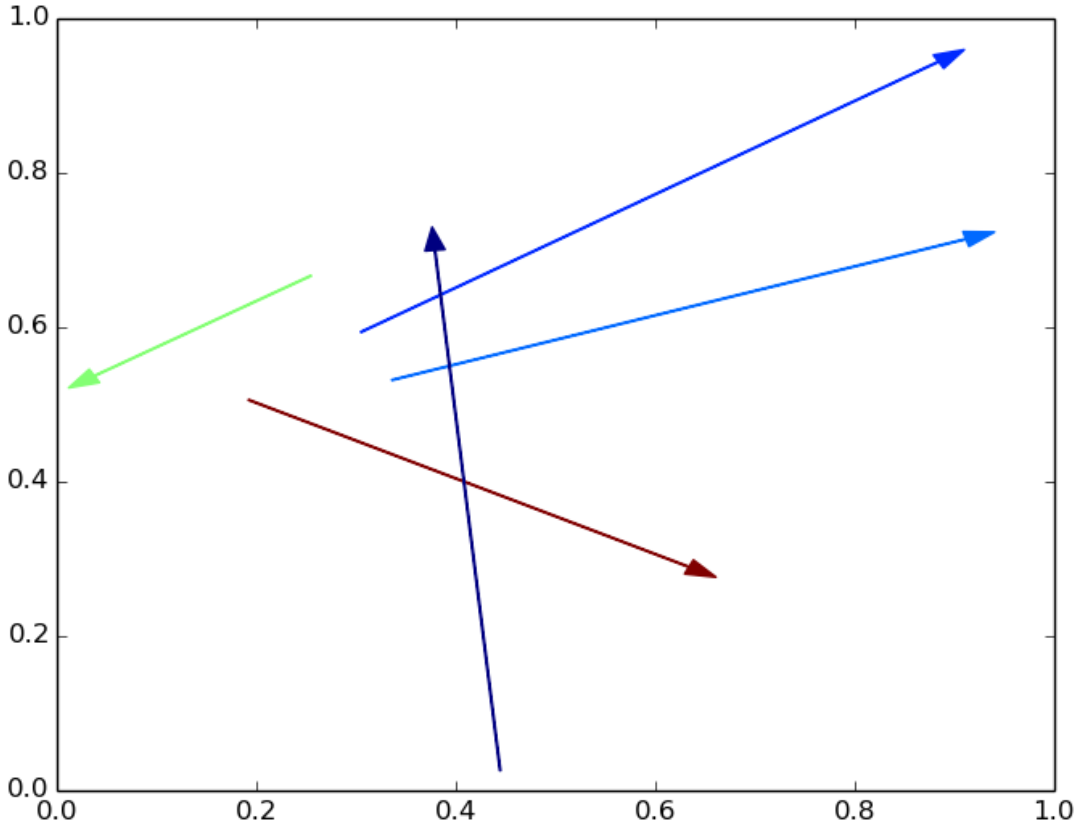


Рисунок 4.1 – Пример изображения для поиска шаблона

3) таблица(имеют метку table);

4) график(имеют метку graph);

5) список информационных ресурсов(имеют метку lit);

После чего на 294 изображениях была обучена модель YOLOv8, с гиперпараметрами обучения $iou = 0.5$,$conf = 0.001$ на 10 эпохах. Результаты полученных изображений приведены на рисунке 4.4.

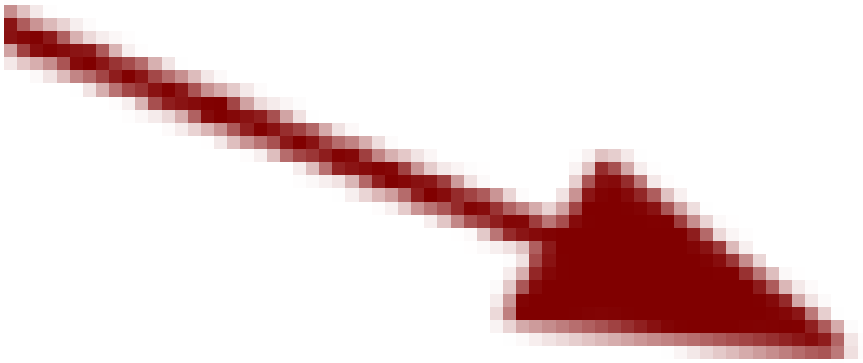


Рисунок 4.2 – Шаблон изображения для поиска

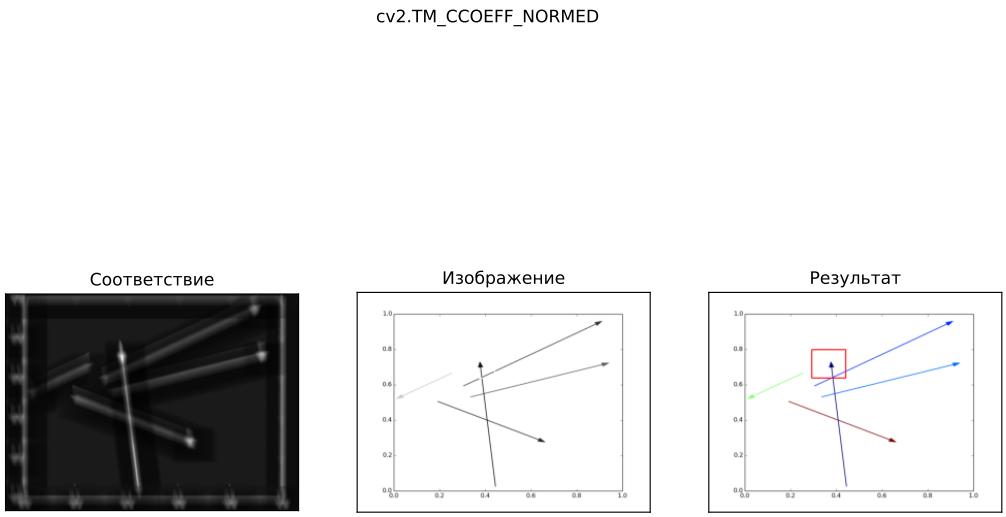


Рисунок 4.3 – Результаты применения шаблона

Actual image

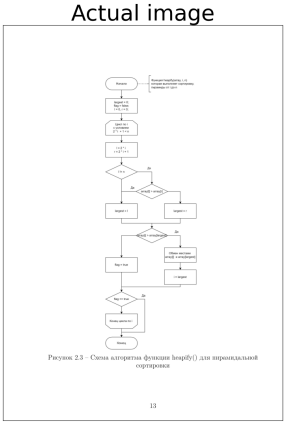


Рисунок 2.3 – Схема алгоритма функции heapify() для пирамидальной сортировки

13

Image with predictions

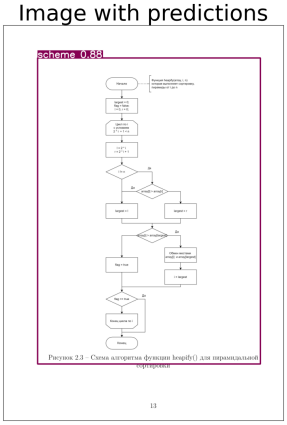
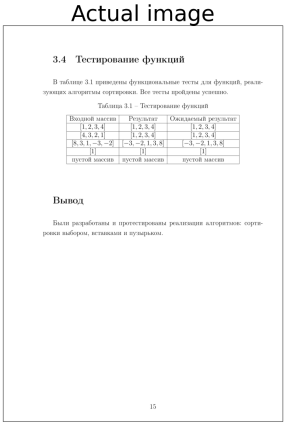


Рисунок 2.3 – Схема алгоритма функции heapify() для пирамидальной сортировки

13

Actual image



3.4 Тестирование функций

В таблице 3.1 приведены функциональные тесты для функций, реализующих алгоритмы сортировки. Все тесты пройдены успешно.

Таблица 3.1 – Тестирование функций

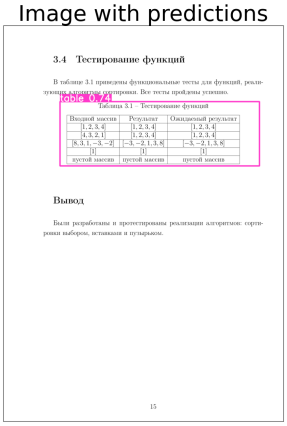
Входной массив	Результат	Ожидаемый результат
[1, 2, 3, 4]	[1, 2, 3, 4]	[1, 2, 3, 4]
[4, 3, 2, 1]	[1, 2, 3, 4]	[1, 2, 3, 4]
[8, 3, 1, −5, −2]	[−5, −2, 1, 3, 8]	[−5, −2, 1, 3, 8]
[1]	[1]	[1]
пустой массив	пустой массив	пустой массив

Вывод

Были разработаны и протестированы реализации алгоритмов: сортировки выбором, вставками и пузырьком.

15

Image with predictions



3.4 Тестирование функций

В таблице 3.1 приведены функциональные тесты для функций, реализующих алгоритмы сортировки. Все тесты пройдены успешно.

Таблица 3.1 – Тестирование функций

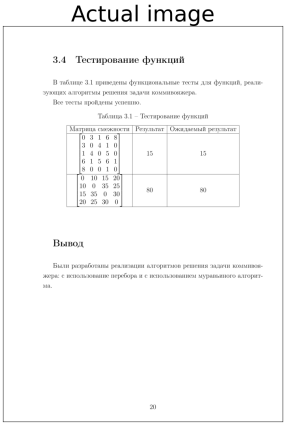
Входной массив	Результат	Ожидаемый результат
[1, 2, 3, 4]	[1, 2, 3, 4]	[1, 2, 3, 4]
[4, 3, 2, 1]	[1, 2, 3, 4]	[1, 2, 3, 4]
[8, 3, 1, −5, −2]	[−5, −2, 1, 3, 8]	[−5, −2, 1, 3, 8]
[1]	[1]	[1]
пустой массив	пустой массив	пустой массив

Вывод

Были разработаны и протестированы реализации алгоритмов: сортировки выбором, вставками и пузырьком.

15

Actual image



3.4 Тестирование функций

В таблице 3.1 приведены функциональные тесты для функций, реализующих алгоритмы решения задачи комбинаторика.
Все тесты пройдены успешно.

Таблица 3.1 – Тестирование функций

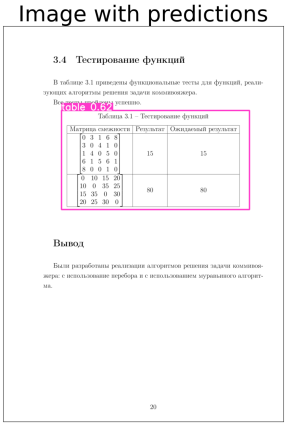
Матрица смежности	Результат	Ожидаемый результат
[0 3 1 6 8; 3 0 4 1 0; 1 4 0 5 0; 6 1 5 6 1; 8 0 0 1 0]	15	15
[0 10 15 20; 10 0 35 25; 15 35 0 30; 20 25 30 0]	80	80

Вывод

Были разработаны реализации алгоритмов решения задачи комбинаторика: с использование перебора и с использованием муравьиного алгоритма.

20

Image with predictions



3.4 Тестирование функций

В таблице 3.1 приведены функциональные тесты для функций, реализующих алгоритмы решения задачи комбинаторика.
Все тесты пройдены успешно.

Таблица 3.1 – Тестирование функций

Матрица смежности	Результат	Ожидаемый результат
[0 3 1 6 8; 3 0 4 1 0; 1 4 0 5 0; 6 1 5 6 1; 8 0 0 1 0]	15	15
[0 10 15 20; 10 0 35 25; 15 35 0 30; 20 25 30 0]	80	80

Вывод

Были разработаны реализации алгоритмов решения задачи комбинаторика: с использование перебора и с использованием муравьиного алгоритма.

20

Actual image

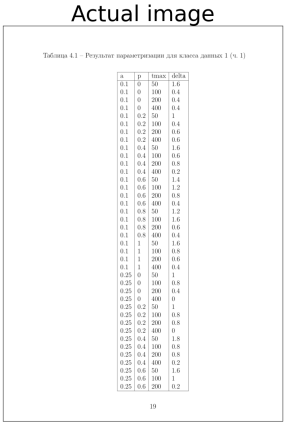


Таблица 4.1 – Результат параметризации для класса данных 1 (ч. 1)

a	p	tmax	delta
0.1	0	50	1.6
0.1	0	100	0.4
0.1	0	200	0.4
0.1	0	400	0.4
0.1	0.2	50	1
0.1	0.2	100	0.4
0.1	0.2	200	0.6
0.1	0.2	400	0.6
0.1	0.4	50	1.6
0.1	0.4	100	0.6
0.1	0.4	200	0.8
0.1	0.4	400	0.2
0.1	0.6	50	1.4
0.1	0.6	100	1.2
0.1	0.6	200	0.8
0.1	0.6	400	0.4
0.1	0.8	50	1.2
0.1	0.8	100	1.6
0.1	0.8	200	0.6
0.1	0.8	400	0.4
0.1	1	50	1.6
0.1	1	100	0.8
0.1	1	200	0.6
0.1	1	400	0.4
0.25	0	50	1
0.25	0	100	0.8
0.25	0	200	0.4
0.25	0	400	0
0.25	0.2	50	1
0.25	0.2	100	0.8
0.25	0.2	200	0.8
0.25	0.2	400	0
0.25	0.4	50	1.8
0.25	0.4	100	0.8
0.25	0.4	200	0.8
0.25	0.4	400	0.2
0.25	0.6	50	1.6
0.25	0.6	100	1
0.25	0.6	200	0.2

19

Image with predictions

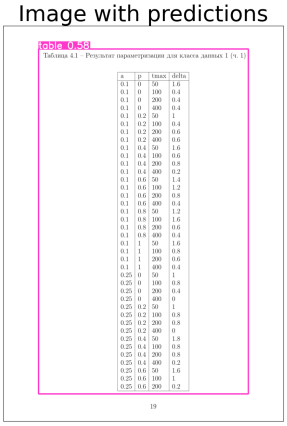


Таблица 4.1 – Результат параметризации для класса данных 1 (ч. 1)

a	p	tmax	delta
0.1	0	50	1.6
0.1	0	100	0.4
0.1	0	200	0.4
0.1	0	400	0.4
0.1	0.2	50	1
0.1	0.2	100	0.4
0.1	0.2	200	0.6
0.1	0.2	400	0.6
0.1	0.4	50	1.6
0.1	0.4	100	0.6
0.1	0.4	200	0.8
0.1	0.4	400	0.2
0.1	0.6	50	1.4
0.1	0.6	100	1.2
0.1	0.6	200	0.8
0.1	0.6	400	0.4
0.1	0.8	50	1.2
0.1	0.8	100	1.6
0.1	0.8	200	0.6
0.1	0.8	400	0.4
0.1	1	50	1.6
0.1	1	100	0.8
0.1	1	200	0.6
0.1	1	400	0.4
0.25	0	50	1
0.25	0	100	0.8
0.25	0	200	0.4
0.25	0	400	0
0.25	0.2	50	1
0.25	0.2	100	0.8
0.25	0.2	200	0.8
0.25	0.2	400	0
0.25	0.4	50	1.8
0.25	0.4	100	0.8
0.25	0.4	200	0.8
0.25	0.4	400	0.2
0.25	0.6	50	1.6
0.25	0.6	100	1
0.25	0.6	200	0.2

19

Рисунок 4.4 – Результаты использования модели

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были выполнены следующие задачи:

- проанализированы существующие виды PDF-документов и связанные с ними ограничения;
- классифицированы типовые требования и ошибки при оформлении отчётов: текста, рисунков, графиков, схем алгоритмов, таблиц и списка источников;
- проанализированы существующие решения выделения составных частей (элементов) отчёта, представленного в формате PDF, в соответствии с ГОСТ 7.32 (фрагменты текста, рисунки, графики, схемы алгоритмов, источники, таблицы и пр.) для дальнейшего анализа с использованием средств компьютерного зрения и автоматического анализа текста;
- реализовано программное обеспечение, позволяющее выделить составные части (элементы) отчета, представленного в формате PDF для дальнейшего анализа на соответствие ГОСТ.

В ходе обучение модели YOLOv8 было использовано 297 изображений (267 изображений для обучения и 30 для проверки корректности работы). Для оценки качества работы модели были выбраны следующие метрики:

- first;
- second;
- third.

ПРИЛОЖЕНИЕ А

Листинг А.1 – Пример части тела PDF файла

```
/Type /Page
/Contents 169 0 R
/Resources 167 0 R
/MediaBox [0 0 595.276 841.89]
/Parent 93 0 R
/Annots [ 166 0 R ]
>>
endobj
166 0 obj
<<
/Type /Annot
/Subtype /Link
/Border[0 0 0]/H/I/C[0 1 0]
/Rect [119.772 380.481 128.456 396.796]
/A << /S /GoTo /D (cite.0@pdf_levels_std) >>
>>
endobj
170 0 obj
<<
/D [168 0 R /XYZ 84.039 825.051 null]
>>
endobj
25 0 obj
<<
```

Листинг А.2 – Пример «хвоста» PDF файла

```
trailer
<<
/Size 44
/Root 42 0 R
/Info 43 0 R
/ID [<7298F57CACD45F4041F17029C0BBF710>
   <7298F57CACD45F4041F17029C0BBF710>] >>
startxref
11085
\%\%EOF
```