Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

Метод выделения составных частей научного текста на основе анализа распределения пикселей в сканирующей строке

Студент	ИУ7-84Б		К. А. Рунов
	(группа)	(подпись)	(инициалы, фамилия)
Руководитель ВКР			Ю. В. Строганов
		(подпись)	(инициалы, фамилия)
Консультант			
		(подпись)	(инициалы, фамилия)
Консультант			
		(подпись)	(инициалы, фамилия)
Нормоконтролер			А. С. Кострицкий
		(подпись)	(инициалы, фамилия)

2025 год

РЕФЕРАТ

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

1 Аналитический раздел

В данном разделе будет проведен анализ предметной области анализа структуры документов, будет приведена формализация задачи предметной области, будут описаны существующие методы и алгоритмы, будет проведена классификация существующих методов, будет обоснована потребность в разработке нового метода, будет сформулирована цель данной работы и формализована постановка задачи.

1.1 Анализ предметной области

В данном подразделе будет проведен анализ предметной области анализа структуры документов, будут рассмотрены существующие типы макетов документов, будут описаны структура научно-технического текста и его составные части.

1.1.1 Анализ структуры документов

Анализ структуры документов (Document layout analysis, DLA) — процесс сегментирования входного изображения документа на однородные компоненты, такие как блоки текста, рисунки, таблицы, графики и т.д., и их классификации [1].

В общем случае анализ структуры документа делится на два взаимосвязанных процесса: физический и логический анализ. Целью физического анализа является выявление структуры документа и определение границ его однородных областей. Целью логического анализа является разметка обнаруженных областей. Выявленные области классифицируются как элементы документа — рисунки, заголовки, абзацы, логотипы, подписи и другие. [2]

Процесс анализа структуры документов состоит из двух основных этапов — этапа предварительной обработки и этапа анализа макета документа [2, 3].

На рисунке ниже приведена схема процесса анализа структуры документов.

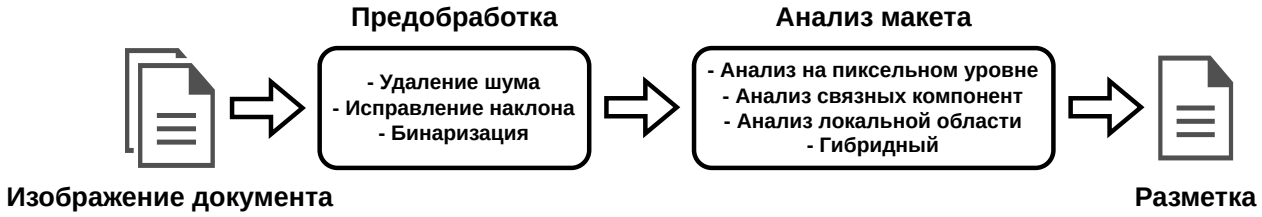


Рисунок 1 – Схема процесса анализа структуры документов [3]

Этап предварительной обработки

Этап анализа макета документа в любом методе анализа структуры документов (далее DLA) часто основывается на определённых предположениях о входных изображениях, таких как отсутствие шума, бинаризация, отсутствие наклона текста или все перечисленные факторы [2, 3].

Цель этапа предварительной обработки — преобразовать входное изображение в соответствии с требованиями этапа анализа макета документа конкретного метода [2, 3].

В общем случае на этом этапе используются одна или несколько процедур предварительной обработки, таких как бинаризация, выравнивание и улучшение изображения [2, 4].

Этап анализа макета документа

Анализ макета документа включает в себя определение границ и типов составляющих областей входного изображения документа. Процесс определения границ областей документа называется сегментацией областей документа, а классификация найденных областей по их типу — классификацией областей документа. [3]

Существуют три типа стратегий анализа макета документа: снизу вверх (bottom-up), сверху вниз (top-down) и гибридная (hybrid).

По стратегии снизу вверх (bottom-up) параметры анализа часто вычисляются на основе исходных данных. Анализ макета документа начинается с небольших элементов, таких как пиксели или связанные компоненты. Затем однородные элементы объединяются, создавая более крупные области. Про-

цесс продолжается, пока не будут достигнуты заранее определённые условия остановки.

По стратегии сверху вниз (top-down) анализ макета документа начинается с крупных областей, например, на уровне всего документа. Затем эта большая область разбивается на более мелкие, такие как колонки текста, на основе определённых правил однородности. Анализ сверху вниз прекращается, когда дальнейшее разбиение областей становится невозможным или достигаются условия остановки.

Гибридная стратегия (hybrid) представляет собой комбинацию обеих стратегий (снизу вверх и сверху вниз). [2]

После сегментации областей происходит их классификация с помощью различных алгоритмов, в результате чего формируется логическая структура документа.

По завершении данного этапа извлеченные геометрическая и логическая структуры сохраняются для последующей реконструкции. Для этого, как правило, используется иерархическая древовидная структура данных. [3]

1.1.2 Типы макетов документов

Макеты документов могут иметь различные структуры. Печатные документы можно разделить на шесть типов [5]: прямоугольные, Манхэттенские, не-Манхэттенские, многоколоночные Манхэттенские, с горизонтальным наложением и с диагональным наложением.

Рисунок 2 – Макеты документов: (a) Стандартный (прямоугольный), (b) Манхэттенский, (c) Не-Манхэттенский, (d) Многоколоночный Манхэттенский, (e) Произвольный (сложный), (f) С горизонтальным и диагональным наложением. [2]

На рисунке выше показаны примеры описанных типов макетов документов:

— Стандартный макет характеризуется большими прямоугольными текстовыми блоками, расположенными в одной или нескольких колонках, при этом каждая колонка содержит по одному абзацу.

— Если документ содержит несколько абзацев в колонках, его можно отнести к Манхэттенскому макету. Примеры таких документов — научно-технические статьи, журналы и другие.

— Не-Манхэттенские макеты включают зоны непрямоугольной формы.

— Макеты с наложением содержат элементы, такие как текст, который перекрывает другие элементы документа. Наложение может возникать, например, из-за просвечивания (см. Рисунок 2(f)).

— Документы с произвольными (или сложными) макетами могут включать рукописный и/или печатный текст, содержащий различные стили, типы и размеры шрифтов.

Таким образом, документы, содержащие научно-технические тексты, обычно используют Манхэттенский макет.

1.1.3 Структура научно-технического текста

Научно-технический текст обычно [6, 7, 8] следует четко определенному шаблону и имеет следующую структуру:

1) Название;

2) Информация об авторах;

3) Аннотация и ключевые слова;

4) Введение;

5) Основная часть (кроме текста содержащая в том числе таблицы, рисунки, графики, листинги);

6) Заключение;

7) Ссылки на литературу.

Содержимое научного текста часто не ограничивается текстом, а содержит также следующие составные части:

1) таблицы,

2) листинги,

3) схемы алгоритмов,

4) рисунки,

5) графики.

Зная структуру научного текста и его основные части можно перейти к формализации задачи выделения составных частей научного текста.

1.2 Формализация предметной области

Пусть D — документ, представленный в виде набора изображений, содержащих текст, листинги, таблицы, рисунки и прочие структурные элементы.

Документ

$$D = \{P_1, P_2, \ldots, P_n\}$$

состоит из страниц $P_1, P_2, \ldots, P_n$, а каждая страница P_i в свою очередь содержит множество объектов $O_{i,1}, O_{i,2}, \ldots, O_{i,m}$.

Объект $O_{i,j}$ — кортеж $(x_{i,j}, y_{i,j}, w_{i,j}, h_{i,j})$, где $(x_{i,j}, y_{i,j})$ — координаты верхнего левого угла, $w_{i,j}$ — ширина, $h_{i,j}$ — высота объекта.

Требуется построить отображение

$$F : D \to \{(O_{i,j}, C_{i,j})\},$$

где каждому объекту $O_{i,j}$ ставится в соответствие класс

$$C_{i,j} = C_{i,j}(O_{i,j}),$$

область допустимых значений которого определяется исходя из требований к разметке.

Например, в случае задачи выделения составных частей научного текста, $C_{i,j} \subseteq \{$Фон, Текст, Таблица, Листинг, Схема алгоритма, Рисунок, График, Неопределенность$\}$; Объект классифицируется как «Неопределенность» в случае, когда не удалось распределить его ни в один из предыдущих классов.

Поставленную задачу можно решить, разбив на две подзадачи и решив каждую подзадачу соответственно: первая подзадача — нахождение объектов на страницах и выявление их геометрических свойств, вторая подзадача — классификация найденных объектов (определение $C_{i,j}$ для каждого объекта $O_{i,j}$).

Решением первой подзадачи является построение отображений

$$P_i \to \{O_{i,j}\},$$

решением второй подзадачи является построение отображений

$$O_{i,j} \to C_{i,j}.$$

Далее будут рассмотрены существующие методы, позволяющие решить поставленную задачу.

1.3 Описание существующих методов

В данном подразделе будет описана суть методов на основе анализа связных компонент, методов на основе анализа проекционного профиля, алгоритма размазывания по длине серии, методов на основе машинного обучения и гибридных методов на основе анализа проекционного профиля и анализа связных компонент.

1.3.1 Анализ связных компонент

Методы на основе связных компонент (Connected Component Analysis, CCA) анализируют и объединяют связные компоненты для формирования однородных областей.

Определение начальных компонент, которые впоследствии объединяются, происходит, как правило, следующим образом. Изображение проходит стадию бинаризации (преобразование к черно-белому формату и назначение каждому пикселю интенсивности 0 или 1), после чего смежные пиксели объединяются на основе 4- или 8-связности. При 4-связности два пикселя считаются связными, если они расположены друг за другом по горизонтали или вертикали. При 8-связности два пикселя считаются связными, если они являются 4-связными, либо расположены друг за другом по диагонали.

После определения начальных компонент, компоненты объединяются в однородные области путем применения специальных алгоритмов. В качестве таких алгоритмов могут выступать, например, преобразование Хафа (Hough transform) или алгоритм K-ближайших соседей (K-nearest neighbor, KNN) [3].

Далее происходит классификация однородных областей. Для классфикации области могут использоваться эвристические алгоритмы (классификация на основе ширины штриха, размера или формы компонента) и алгоритмы на основе машинного обучения.

Методы на основе связных компонент могут работать в условиях скошенного текста при условии, что межстрочный интервал меньше пробела между абзацами [3].

1.3.2 Анализ проекционного профиля

Суть методов на основе анализа проекционного профиля (Projection Profile Analysis, PPA) заключается в следующем. Пиксели документа проецируются на вертикальную и горизонтальную ось, после чего строятся гистограммы распределения пикселей. Далее анализируются пики и впадины на гистограммах. Впадины на вертикальном профиле указывают на пробелы между колонками текста. Впадины на горизонтальном профиле указывают на пробелы между строками или блоками текста.

На основе проведенного анализа можно разделить документ на логические компоненты — текстовые блоки, заголовки, таблицы, изображения.

Данные методы работают только с Манхэттенскими макетами документов и чувствительны ко скошенности текста в документе [3].

1.3.3 Алгоритм размазывания по длине серии

Методы на основе RLSA преобразуют бинарное изображение документа путем «размазывания» черных пикселей горизонтально и/или вертикально для формирования однородных областей.

На рисунке ниже можно видеть пример работы RLSA.

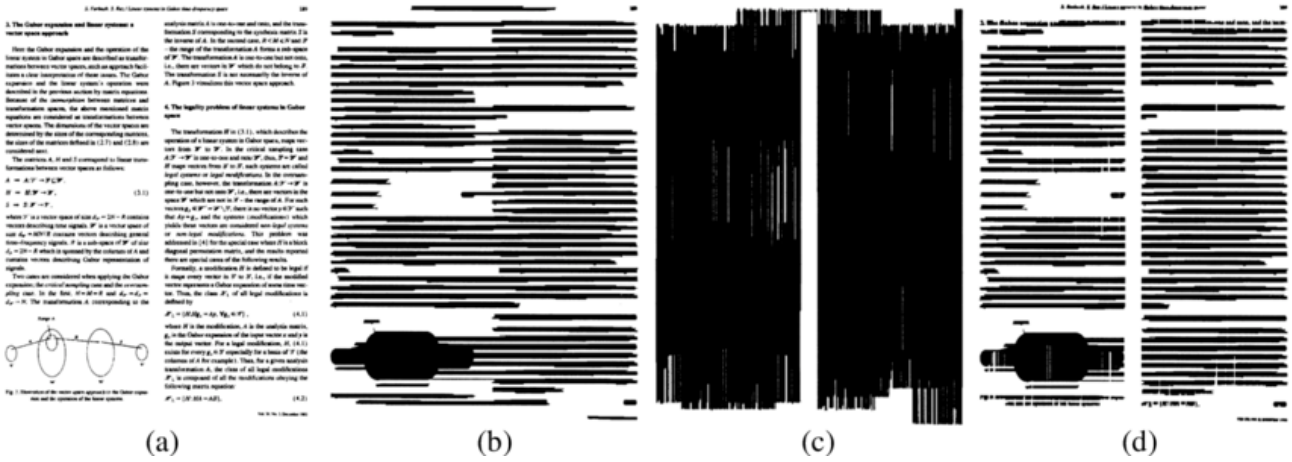


Рисунок 3 – Пример работы алгоритма RLSA. К начальному изображению документа (a) применяется горизонтальный (b) и вертикальный (c) RLSA, после чего в результате применения операции И к изображениям (b) и (c) формируется (d)

Для классификации полученных областей также используются эвристические алгоритмы и алгоритмы на основе машинного обучения.

Данные методы, как и методы на основе анализа проекционного профиля, работают преимущественно с Манхэттенскими макетами документов и чувствительны к скошенному тексту [3].

1.3.4 Методы на основе машинного обучения

Методы, не основанные на глубоком обучении, используют простые архитектуры нейросетей для обучения. Анализ с использованием нейросети происходит на трех уровнях: уровне пикселей, уровне блоков текста и уровне страниц.

Методы на основе машинного обучения в области анализа макетов документов страдают от несбалансированности данных и отсутствия контекстной информации. Если модель обучалась на документах, в наборе данных для обучения количество текстовой и фоновой информации сильно превосходит

количество информации о рисунках и графиках. В связи с этим обученная модель может склоняться в сторону текстовых или фоновых пикселей. [2]

Обучение моделей лишь на основе информации о пикселях чревато потерей контекстной информации. Поэтому при обучении на уровнях блоков текста и страниц прибегают к использованию методов извлечения признаков для создания более надежных моделей. [2]

Методы на основе машинного обучения работают с любыми макетами документов и не чувствительны к скошенному тексту.

1.3.5 Гибридные методы на основе PPA и CCA

Гибридные методы на основе анализа проекционного профиля и связных компонент используют работают следующим образом. Начальные компоненты определяются применением метода из PPA, после чего происходит их уточнение применением методов из CCA.

Такой комбинированный подход позволяет лучше сегментировать текст, чем каждый метод по отдельности.

1.4 Классификация существующих методов

Для сравнения рассмотренных методов можно выделить следующие критерии:

— Стратегия анализа макета документа;

— Скорость работы — требования метода к вычислительным ресурсам;

— Гибкость — способность метода адаптироваться к различным типам макетов документов;

— Устойчивость — способность метода адаптироваться к шумам и искажениям текста.

— Специальное требование — позволяет сегментировать не только текст, но и такие составные части научного текста, как таблицы, листинги, схемы, рисунки, графики и прочее.

Ниже приведена таблица со сравнительным анализом рассмотренных методов.

Таблица 1 – Классификация методов DLA

Метод	**Стратегия**	**Скорость**	**Гибкость**	**Ус-ть**	**СпецТреб**
CCA	Снизу вверх	2	2	3	Нет
PPA	Сверху вниз	2	3	3	Нет
RLSA	Сверху вниз	1	3	3	Нет
ML	Снизу вверх	3	1	1	Да
PPA + CCA	Гибридный	2	3	2	Нет

Как можно видеть по сравнительной таблице, ни один из рассмотренных методов не позволяет «быстро», на основе лишь различных эвристик, произвести разметку документа, сегментирующую не только текст, но и другие составные части научного текста.

1.5 Формализованная постановка задачи

Целью данной работы является разработка метода, позволяющего выделять составные части научного текста на основе простых правил, без использования нейросетей, а также разработка алгоритма, реализующего данный метод.

Формализованная в виде IDEF0 диаграммы постановка задачи представлена на рисунке 4 ниже.

Рисунок 4 – Постановка задачи

Вывод

В данном разделе был проведен анализ предметной области анализа структуры документов, была приведена формализация задачи предметной области, были описаны существующие методы и алгоритмы, была проведена классификация существующих методов, была обоснована потребность в разработке нового метода, была сформулирована цель данной работы и формализована постановка задачи.

2 Конструкторский раздел

В данном разделе будут приведены требования и ограничения разрабатываемого метода, будут описаны основные этапы разрабатываемого метода, сценарии тестирования, классы эквивалентности тестов, а также структура разрабатываемого ПО.

2.1 Требования и ограничения метода

Метод выделения составных частей научного текста на основе анализа распределения пикселей в сканирующей строке должен:

1) Работать с одноколоночными Манхэттенскими макетами документов;
2) Выделять текстовые блоки;
3) Выделять таблицы;
4) Выделять листинги;
5) Выделять схемы алгоритмов;
6) Выделять рисунки;
7) Выделять графики;
8) Работать на основе простых правил и эвристик, без использования нейросетей.

2.2 Описание разрабатываемого метода

Поставленная задача решается в четыре этапа:

1) Преобразование PDF документа в изображения;
2) Первичная разметка страниц;
3) Создание уточненной разметки на основе первичной;
4) Объединение уточненной разметку в более крупные блоки.

Разметка, ее уточнение и объединение происходят на основе определенных правил, которые будут описаны в данном разделе далее.

Основные этапы разрабатываемого метода представлены на IDEF0 диаграмме первого уровня (см. Рисунок 5).

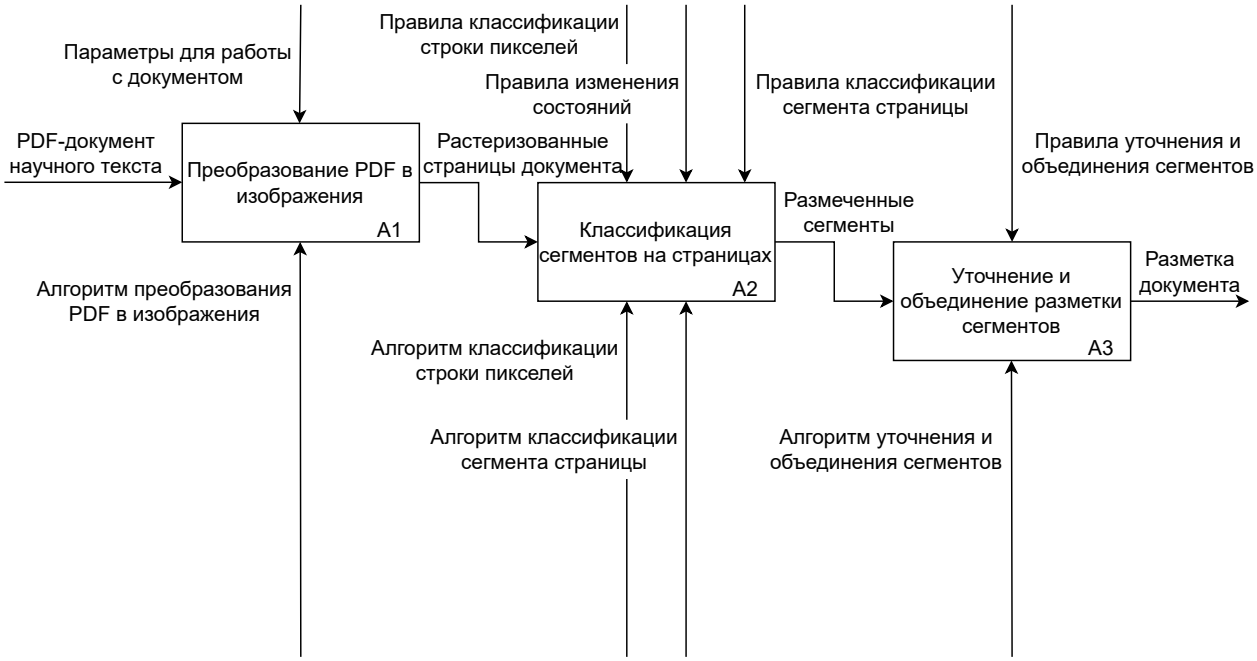


Рисунок 5 – IDEF0-диаграмма метода выделения составных частей научного текста на основе анализа распределения пикселей в сканирующей строке

2.2.1 Первичная разметка

Исходные данные — изображение страницы документа. Получаемый результат — разметка страницы и информация о каждом сегменте на ней.

Разметка страницы — массив кортежей типа

$$(y\_start, y\_end, C),$$

где y_start — y-координата начала сегмента в пространстве изображения, y_end — y-координата конца сегмента в пространстве изображения, C — класс сегмента, где $C \subseteq$ {Фон, Немного текста, Много текста, Цвет, Черная линия средней длины, Длинная черная линия, Не определено}.

Такое множество классов было выделено по двум причинам. Во-первых, на основе распределения пикселей в строке ее уже можно причислить к одному из перечисленных классов. Во-вторых, на основе принадлежности строки

к одному из классов можно делать предположения насчет класса сегмента, которому строка принадлежит.

Так, если строка классифицируется как «Много текста» можно предположить, что сегмент вероятно принадлежит классу «Текст».

Если строка классифицируется как «Цвет», можно предположить, что сегмент вероятно принадлежит классу «Рисунок» или «График».

Если строка классифицируется как «Черная линия средней длины», можно предположить, что сегмент вероятно принадлежит классу «Схема алгоритма».

Если строка классифицируется как «Длинная черная линия», можно предположить, что сегмент вероятно принадлежит классу «Таблица» или «Листинг».

Информация о сегменте содержит следующие данные:

1) start — ордината начала сегмента;

2) end — ордината конца сегмента;

3) count_long_black_line — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Длинная черная линия»;

4) count_single_long_black_line — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Длинная черная линия», считая несколько подряд идущих «Длинных черных линий» за одну;

5) count_medium_black_line — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Черная линия средней длины»;

6) count_single_medium_black_line — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Черная линия средней длины», считая несколько подряд идущих «Черных линий средней длины» за одну;

7) count_total_medium_black_line — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Черная линия

средней длины», с учетом всех «Черных линий черной длины» если таких было зафиксировано несколько внутри одной сканирующей строки;

8) count_many_text — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Много текста»;

9) count_few_text — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Немного текста»;

10) count_color — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Цвет»;

11) count_undefined — количество раз, когда при разметке сегмента встретилась строка, идентифицированная, как «Не определено»;

12) count_white_px — количество белых пикселей в сегменте;

13) count_color_px — количество цветных пикселей в сегменте;

14) count_gray_px — количество черных пикселей в сегменте (сумма трех данных счетчиков дает общее количество пикселей в сегменте);

15) heatmap_black — массив, i-й элемент которого отражает количество черных пикселей в i-й колонке пикселей сегмента;

16) heatmap_color — массив, i-й элемент которого отражает количество цветных пикселей в i-й колонке пикселей сегмента.

Данная информация будет использоваться для уточнения разметки в следующем этапе.

Первичная разметка создается в результате классификации строк на основе распределения пикселей в них и изменения состояний конечного автомата первичной разметки.

Диаграмма изменения состояний конечного автомата изображена на рисунке 6. Классификация любого сегмента начинается из состояния «Фон», и заканчивается в состоянии «Фон», причем в результате классификации сегменту присваивается класс в соответствии с предпоследним состоянием, в котором находился конечный автомат (до последнего «Фона»).

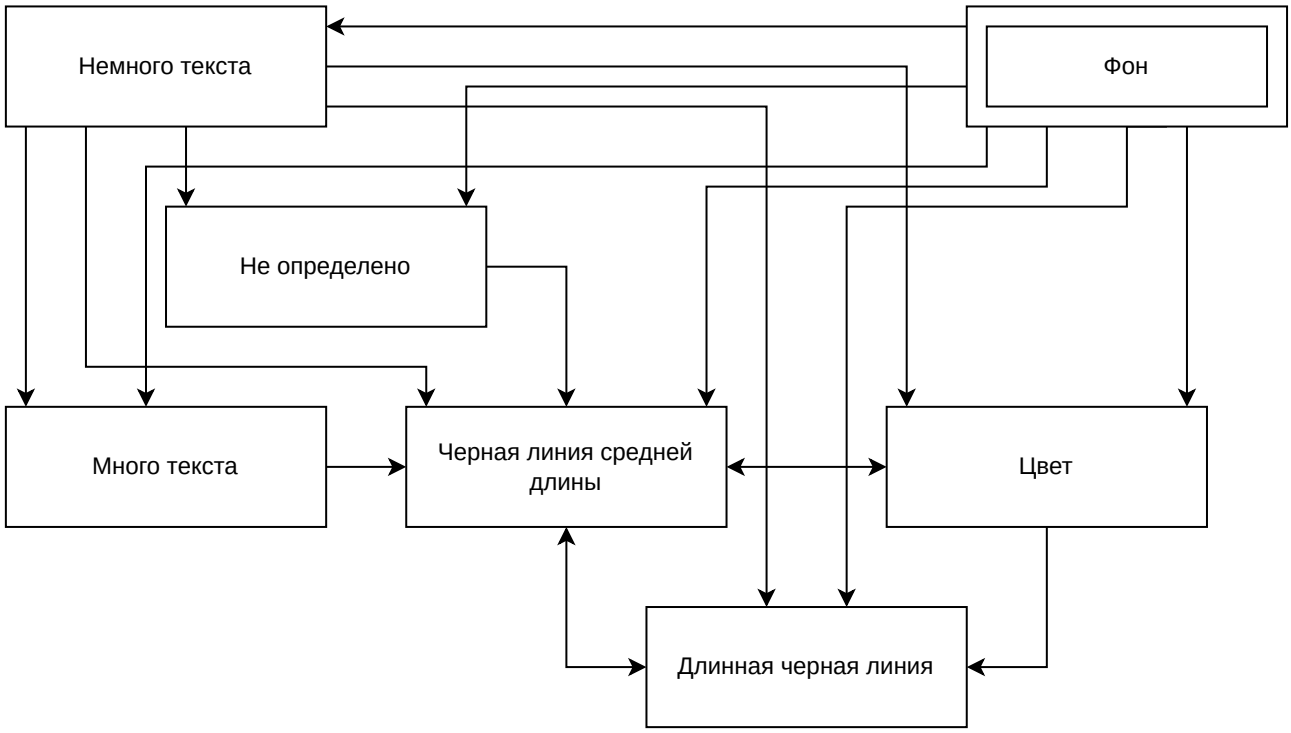


Рисунок 6 – Конечный автомат, все состояния

Из состояния «Фон» можно попасть в любое состояние. Из состояния «Немного текста» можно попасть в любое состояние, кроме «Фона».

На рисунке 7 изображена редуцированная диаграмма конечного автомата, опускающая состояния «Фон» и «Немного текста».

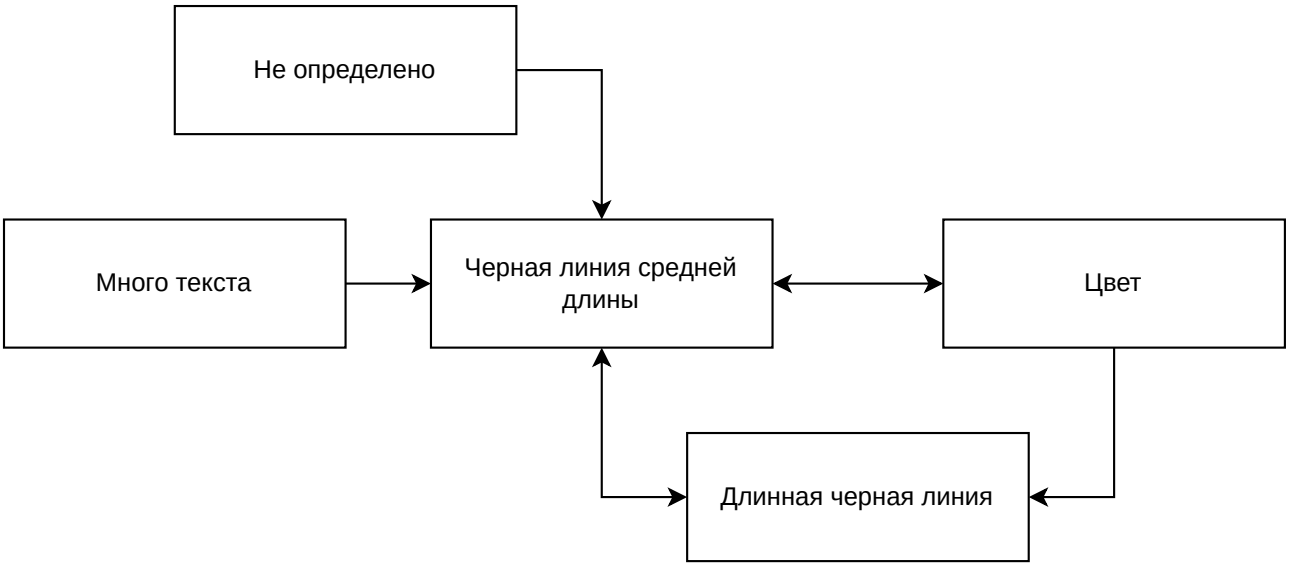


Рисунок 7 – Конечный автомат, состояния кроме «Немного текста» и «Фон»

На рисунке 8 ниже изображена схема алгоритма классификации конкретной сканирующей строки.

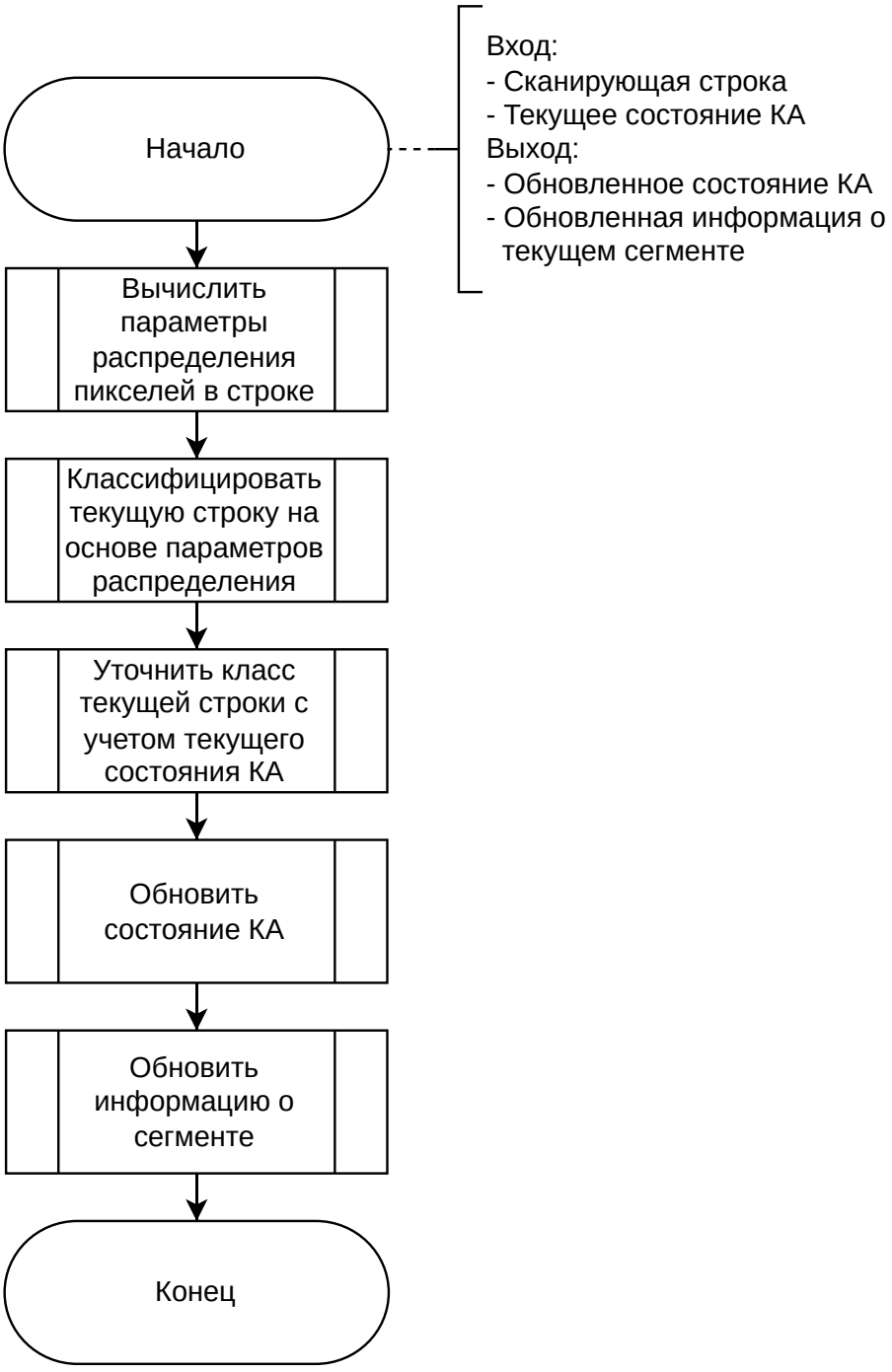


Рисунок 8 – Разметка сканирующей строки

Классификация строки

Классификация строки производится на основе следующих параметров распределения пикселей в ней:

1) count_white — количество белых пикселей в строке;
2) count_color — количество цветных пикселей в строке;
3) count_gray — количество серых (почти черных) пикселей в строке;
4) comp_lengths — массив длин участков подряд идущих не белых пикселей;
5) gap_lengths — массив длин промежутков (белых пикселей) между участками подряд идущих не белых пикселей;
6) gray_comp_lengths — массив длин участков подряд идущих серых пикселей;
7) color_comp_lengths — массив длин участков подряд идущих цветных пикселей;
8) first_nonwhite_index — индекс первого не белого пикселя в строке.

Классификация как «Фон»

Строка классифицируется как «Фон», если выполнено условие «first_nonwhite_index не установлен» — в строке не нашлось не белых пикселей.

Классификация как «Длинная черная линия»

Строка классифицируется как «Длинная черная линия», если: строка содержит единственную серую компоненту И эта компонента достаточно длинная.

Содержание единственной серой компоненты определяется на основе одновременного выполнения следующих условий:

— Длина comp_lengths равна 1;

— Длина gap_lengths равна 0;

— Длина color_comp_lengths равна 0.

Длина серой компоненты для классификации строки как «Длинная черная линия» считается достаточно большой, если count_gray больше некоторого параметра, который является произведением длины строки на некоторую наперед заданную константу, принимающую значения от 0 до 1, например 1/2.

Классификация как «Черная линия средней длины»

Строка классифицируется как «Черная линия средней длины», если строка содержит компоненту, длина которой больше некоторого параметра, который является произведением длины строки на некоторую наперед заданную константу, принимающую значения от 0 до 1, например, 1/16.

Классификация как «Много текста»

Строка классифицируется как «Много текста», если она либо содержит очень много черных компонент, либо содержит много черных компонент и не содержит цвета.

Считается, что строка содержит очень много черных компонент, если длина comp_lengths превышает некоторую наперед заданную константу, например 100.

Считается, что строка содержит много черных компонент, если длина comp_lengths превышает некоторую наперед заданную константу, например 80.

Таким образом, если строка содержит очень много черных компонент, она будет классифицирована как «Много текста» вне зависимости от наличия в ней цвета.

Классификация как «Цвет»

Строка классифицируется как «Цвет», если count_color больше нуля.

Классификация как «Немного текста»

Строка классифицируется как «Немного текста», если содержит немного компонент И компоненты преимущественно небольшого размера И промежутки между компонентами преимущественно небольшого размера И отсутствуют большие промежутки.

Считается, что компоненты и/или промежутки между компонентами преимущественно небольшого размера, если среднее арифметическое comp_lengths и/или gap_lengths меньше некоторой наперед заданной константы, например, 20 пикселей.

Считается, что большие промежутки отсутствуют, если в массиве gap_lengths отсутствуют элементы с индексом стандартного отклонения больше шести.

Индекс стандартного отклонения Z для элемента массива x вычисляется по формуле:

$$Z(x) = \frac{x - \mu}{\sigma},$$

где μ — среднее значение элементов массива, σ — стандартное отклонение элементов массива.

Классификация как «Не определено»

В остальных случаях строка классифицируется как «Не определено».

Изменение состояний КА

Далее будут представлены таблицы переходов состояний конечного автомата, в которых некоторые правила были подобраны исходя из логических соображений, а некоторые — эмпирическим путем.

Таблица 2 – Таблица переходов состояний из начального состояния «Фон»

Состояние текущей строки	**Конечное состояние**
Не определено	Не определено
Много текста	Много текста
Немного текста	Немного текста
Длинная черная линия	Длинная черная линия
Черная линия средней длины	Черная линия средней длины
Цвет	Цвет
Фон	Фон

Таблица 3 – Таблица переходов состояний из начального состояния «Не определено»

Состояние текущей строки	**Конечное состояние**
Не определено	Не определено
Много текста	Много текста
Немного текста	Не определено
Длинная черная линия	Не определено
Черная линия средней длины	Черная линия средней длины
Цвет	Цвет
Фон	Фон

Таблица 4 – Таблица переходов состояний из начального состояния «Много текста»

Состояние текущей строки	**Конечное состояние**
Не определено	Много текста
Много текста	Много текста
Немного текста	Много текста
Длинная черная линия	Много текста
Черная линия средней длины	Черная линия средней длины
Цвет	Много текста
Фон	Фон

Таблица 5 – Таблица переходов состояний из начального состояния «Немного текста»

Состояние текущей строки	**Конечное состояние**
Не определено	Не определено
Много текста	Много текста
Немного текста	Немного текста
Длинная черная линия	Длинная черная линия
Черная линия средней длины	Черная линия средней длины
Цвет	Цвет
Фон	Фон

Таблица 6 – Таблица переходов состояний из начального состояния «Длинная черная линия»

Состояние текущей строки	**Конечное состояние**
Не определено	Длинная черная линия
Много текста	Длинная черная линия
Немного текста	Длинная черная линия
Длинная черная линия	Длинная черная линия
Черная линия средней длины	Черная линия средней длины
Цвет	Длинная черная линия
Фон	Фон

Таблица 7 – Таблица переходов состояний из начального состояния «Черная линия средней длины»

Состояние текущей строки	**Конечное состояние**
Не определено	Черная линия средней длины
Много текста	Черная линия средней длины
Немного текста	Черная линия средней длины
Длинная черная линия	Длинная черная линия
Черная линия средней длины	Черная линия средней длины
Цвет	Цвет
Фон	Фон

Таблица 8 – Таблица переходов состояний из начального состояния «Цвет»

Состояние текущей строки	**Конечное состояние**
Не определено	Цвет
Много текста	Цвет
Немного текста	Цвет
Длинная черная линия	Длинная черная линия
Черная линия средней длины	Черная линия средней длины
Цвет	Цвет
Фон	Фон

Пример первичной разметки

Пусть при сканировании документа встретилось данное предложение. Разметка будет происходить следующим образом:

1) Начальное состояние конечного автомата — «Фон».

2) Встретилась строка с не белым пикселем — вершина буквы «П». В ней содержится одна сплошная черная компонента, но она недостаточно длинная для классификации, как «Черная линия средней длины». Также не хватает информации для причисления ее к какому-либо другому классу, поэтому строка классифицируется как «Не определено».

3) Конечный автомат меняет состояние: из «Фона» по «Не определено» переходит в «Не определено».

4) Обновляется информация о сегменте.

5) Сканируется следующая строка, в которой встречается два очень маленьких компонента с небольшим расстоянием между ними — вертикальные линии буквы «П». Этой информации оказывается достаточно для классификации строки как «Немного текста».

6) Конечный автомат не меняет состояние: из «Не определено» по «Немного текста» остается в «Не определено».

7) Обновляется информация о сегменте.

8) Подобные действия продолжаются, пока в сканирующей строке не встретятся вершины других букв.

9) Сканируется строка, в которой встречаются вершины других букв — много маленьких компонентов на протяжении почти всей длины строки. Этой информации достаточно для классификации строки как «Много текста».

10) Конечный автомат меняет состояние: из «Не определено» по «Много текста» переходит в «Много текста».

11) Обновляется информация о сегменте.

12) На протяжении следующих итераций строки продолжат классифицироваться как «Много текста» до тех пор, пока не начнут встречаться «хвосты» букв «р» и «у».

13) При сканировании «хвостов» строки будут классифицироваться как «Немного текста», но это не повлияет на состояние конечного автомата «Много текста».

14) Обновляется информация о сегменте.

15) После «хвостов» наконец встретится «Фон», конечный автомат перейдет из состояния «Много текста» в состояние «Фон», и сегмент будет классифицирован в соответствии с предпоследним состоянием конечного автомата — «Много текста».

2.2.2 Уточненная разметка

Исходные данные — разметка страницы и информация о каждом сегменте на ней. Получаемый результат — уточненная разметка страницы.

Уточненная разметка страницы — массив кортежей типа

$$(y\_start, y\_end, C),$$

где y_start — y-координата начала сегмента в пространстве изображения, y_end — y-координата конца сегмента в пространстве изображения, C —

класс сегмента, где $C \subseteq$ {Фон, Текст, Таблица, Листинг, Схема алгоритма, Рисунок, График, Не определено}, причем координаты начала и конца сегментов совпадают с соответствующими координатами сегментов первичной разметки, уточняется только класс на основе информации о сегментах.

Далее будут перечислены правила, на основе которых сегмент меняет класс из первичного, соответствующего одному из состояний конечного автомата, в уточненный из множества {Фон, Текст, Таблица, Листинг, Схема алгоритма, Рисунок, График, Не определено}.

Из «Не определено» в...

«Текст»

Правило: Высота сегмента небольшая ИЛИ Много строк в сегменте было классифицировано как «Немного текста».

Высота сегмента считается небольшой, если

$$height < C1,$$

где $height = end - start$, а C1 — наперед заданная константа, соответствующая максимальному количеству пикселей, при котором сегмент считается небольшим.

Считается, что много строк в сегменте было классифицировано как «Немного текста», если

$$\frac{count\_few\_text}{height} > C2,$$

где C2 — наперед заданная константа, соответствующая минимальному отношению количества строк в сегменте, классифицированных как «Немного текста», к высоте сегмента, при котором считается, что сегмент содержит много строк, классифицированных как «Немного текста».

Обоснование: Если сегмент находится в состоянии «Не определено», значит на протяжении его разметки не встретилось ни одной строки, позволяющей классифицировать его, как «Много текста», «Черная линия средней длины» или «Цвет». Следовательно, данный сегмент вероятно относится либо к классу «Текст», либо к классу «Не определено». Если высота сегмента

небольшая, вероятно он все же относится к тексту. Такое возможно, например, в случае использования буквы «й» — сегмент, содержащий ее дугу будет классифицирован, как «Не определено».

«Таблица»

Сегмент может получить класс «Таблица» только из состояний «Длинная черная линия» или «Много текста».

«Листинг»

Правило: Сегмент содержит ровно две вертикальные черные линии высотой с весь сегмент (вычисляется на основе heatmap_black).

«Схема алгоритма»

Сегмент может получить класс «Схема алгоритма» только из состояний «Черная линия средней длины» или «Длинная черная линия».

«Рисунок»

Правило: Сегмент большой.

Сегмент считается большим, если

$$height > C3,$$

где С3 — наперед заданная константа, соответствующая минимальной высоте «Не определенного сегмента», чтобы классифицировать его, как «Рисунок».

Обоснование: Если на протяжении длительного отрезка документа ни разу не встретилось линии, классифицируемой, как «Много текста», «Черная линия средней длины» или «Цвет», то, вероятно, перед нами ни «Текст», ни «Схема алгоритма», ни «График», а «Рисунок».

«График»

Правило: Сегмент содержит одну высокую вертикальную линию.

Считается, что сегмент содержит высокую вертикальную линию, если в массиве heatmap_black содержится единственный элемент, значение которого превышает

$$height * C4,$$

где С4 — наперед заданная константа такая, что $height * C4$ не выше оси ординат графика.

Такая корректирующая константа нужна, потому что часто высота сегмента графика превышает высоту его оси абсцисс, так как на оси ординат

есть «засечки», которые увеличивают размер сегмента, но не увеличивают высоту вертикальной оси.

«Не определено»

Сегмент получает класс «Не определено», если ранее не удалось причислить его ни к одному другому классу.

Порядок проверки принадлежности к уточненному классу:

1) Текст,
2) Листинг,
3) Рисунок,
4) График,
5) Не определено.

Из «Немного текста» в...

«Текст»

Правило: Всегда.

Обоснование: В соответствии с конечным автоматом, сегмент получает класс «Немного текста» только в том случае, если на протяжении его разметки встречались только строки, классифицируемые, как «Немного текста». Следовательно, вероятно, данный сегмент относится к «Тексту».

Из «Много текста» в...

«Таблица»

Правило: Сегмент достаточно большой И содержит больше двух вертикальных линий И минимальное расстояние между двумя вертикальными линиями не слишком маленькое.

«Листинг»

Правило: Сегмент достаточно большой И содержит ровно две вертикальные линии И не содержит черных линий среднего размера И (содержит хотя бы одну строку, классифицированную как «Много текста» ИЛИ не содержит строк, классифицированных как «Цвет»)

Обоснование: В листингах кода как правило не встречается черных линий среднего размера, но при этом бывают схемы алгоритмов, например IDEF0 диаграммы, которые, как и листинг, ограничены двумя вертикальными черными линиями.

Если сегмент содержал хотя бы одну строку, классифицированную как «Много текста» (что часто бывает в случае с листингами), то, вероятно, данный сегмент относится к классу «Листинг» несмотря на наличие цвета, которым, бывает, выделяют ключевые слова, относящиеся к конкретному языку программирования.

Если же таких строк нет, при этом есть строки, классифицированные, как «Цвет», с большой вероятность утверждать, к какому классу относится сегмент, становится затруднительно.

«Текст»

Сегмент получает класс «Текст», если ранее не удалось причислить его ни к одному другому классу.

Порядок проверки принадлежности к уточненному классу:

1) Таблица,
2) Листинг,
3) Текст.

Из «Цвет» в...

«График»

Правило: Сегмент имеет единственную высокую вертикальную линию И отношение цветных пикселей в сегменте к белым мало.

Считается, что отношение цветных пикселей в сегменте к белым мало, если

$$\frac{count\_color\_px}{count\_white\_px} < C5,$$

где C5 — наперед заданная константа, соответствующая минимальному отношению цветных пикселей в сегменте к белым, чтобы данное отношение считалось малым.

Обоснование: Как правило, графики в документах имеют ось ординат, а сами изображены цветом на белом фоне, следовательно, белые пиксели в

сегменте с графиком преобладают над цветными.

«Не определено»

Правило: Сегмент небольшой.

Считается, что сегмент небольшой, если его высота меньше некоторой наперед заданной константы C6, соответствующей максимальной высоте сегмента, при которой сегмент можно считать небольшим.

«Рисунок»

Сегмент получает класс «Рисунок», если ранее не удалось причислить его ни к одному другому классу.

Порядок проверки принадлежности к уточненному классу:

1) График,
2) Не определено,
3) Рисунок.

Из «Черная линия средней длины» в...

«Текст»

Правило: В сегменте встречалось много строк, идентифицируемых, как «Много текста» ИЛИ (сегмент небольшой И (в сегменте встречалось много строк, идентифицируемых, как «Немного текста» ИЛИ «Не определено»)).

Обоснование: Такое правило позволяет идентифицировать как «Текст» сегменты, в которых подчеркивание текста сливается с самим текстом, например, на титульном листе в разделе с фамилиями студента и преподавателей.

«Рисунок»

Правило: Сегмент достаточно большой И (имеет цвет ИЛИ имеет много строк, идентифицируемых как «Черная линия средней длины»).

Обоснование: Рисунки в документах, как правило, занимают значительную часть страницы, поэтому небольшой сегмент, идентифицированный изначально, как «Черная линия средней длины», вероятно не является «Рисунком» или «Графиком».

Рисунки часто имеют цвет, а если они не цветные, и на них встретилась «Черная линия средней длины», то вероятно строк, идентифицированных таким образом, на них будет достаточно много. Если таких строк мало, то скорее сегмент относится к классу «Схема алгоритма» нежели «Рисунок».

«График»

Правило: Сегмент имеет цвет И имеет не много строк, идентифицированных как «Черная линия средней длины» И количество вертикальных черных линий не меньше двух И количество белых пикселей преобладает над остальными.

Обоснование: Данное правило учитывает ситуации, когда прикрепленные в документе графики имеют небольшой масштаб (иначе они были бы изначально классифицированы как «Длинная черная линия»). А когда прикрепленные в документе графики имеют небольшой масштаб, они часто ограничены черным прямоугольником, откуда и появляется условие «количество вертикальных черных линий не меньше двух».

«Не определено»

Правило 1: Сегмент имеет единственную «Черную линию средней длины» ИЛИ (сегмент не очень высокий И имеет не много «Черных линий средней длины»).

Правило 2: Сегмент небольшой (меньше наперед заданной константы, например, 20 пикселей).

Обоснование: Правило 1 позволяет классифицировать как «Не определено» формулы и уравнения, содержащие квадратные корни, дроби и знаки «$\sum$», ведь «Черная линия средней длины» часто встречается в единственном экземпляре именно в формулах и уравнениях.

Можно было бы классифицировать такие сегменты, как «Формула», а не «Не определено», однако разрабатываемый метод не позволяет отличать более простые формулы от текста, поэтому было принято решение не вводить для данной ситуации отдельный класс.

Если сегмент небольшой, то вероятно он не принадлежит к какому-либо другому классу, кроме как «Не определено».

«Схема алгоритма»

Также сегмент получает класс «Схема алгоритма», если ранее не удалось причислить его ни к одному другому классу.

Порядок проверки принадлежности к уточненному классу:

1) График,

2) Рисунок,

3) Схема алгоритма,

4) Текст,

5) Не определено,

6) Схема алгоритма.

Из «Длинная черная линия» в...

«Таблица»

Правило: Сегмент достаточно большой И содержит больше двух вертикальных линий И минимальное расстояние между двумя вертикальными линиями не слишком маленькое.

«Листинг»

Правило: Сегмент содержит ровно две вертикальные линии И не содержит черных линий среднего размера И (содержит хотя бы одну строку, классифицированную как «Много текста» ИЛИ не содержит строк, классифицированных как «Цвет»).

«Схема алгоритма»

Правило: Сегмент не имеет цвета и содержит не меньше двух черных линий среднего размера.

Обоснование: В схеме алгоритма часто встречается не менее двух черных линий среднего размера — начало блока и конец блока (например на IDEF0 диаграмме).

«График»

Правило: Сегмент имеет цвет И содержит немного длинных черных линий И количество вертикальных черных линий не меньше двух И количество белых пикселей преобладает над остальными.

Считается, что сегмент содержит несколько черных линий, если

$$\frac{count\_long\_black\_line}{height} < C7,$$

где C7 — наперед заданная константа, соответствующая максимальному значению отношения количества строк, идентифицированных как «Длинная черная линия» к высоте сегмента, при котором можно считать, что сегмент содержит немного длинных черных линий.

«Не определено»

Правило: Сегмент небольшой.

«Рисунок»

Сегмент получает класс «Рисунок», если ранее не удалось причислить его ни к одному другому классу.

Порядок проверки принадлежности к уточненному классу:

1) Не определено,
2) График,
3) Таблица,
4) Листинг,
5) Схема алгоритма,
6) Рисунок.

2.2.3 Объединенная разметка

Исходные данные — уточненная разметка страницы. Получаемый результат — объединенная разметка страницы.

Объединенная разметка страницы формируется из уточненной разметки в несколько этапов:

1) Слияние небольших фоновых сегментов с ближайшим наибольшим сегментом;
2) Объединение сегментов после слияния;
3) Слияние фоновых сегментов с соседними, если соседние сегменты имеют один и тот же класс;
4) Объединение сегментов после слияния;
5) Смена класса небольших фоновых сегментов на «Не определено»;
6) Объединение сегментов после слияния;

7) Слияние небольших «Не определенных» сегментов с наибольшими соседними сегментами;

8) Объединение сегментов после слияния.

На рисунке 9 представлен алгоритм объединения сегментов после слияния.

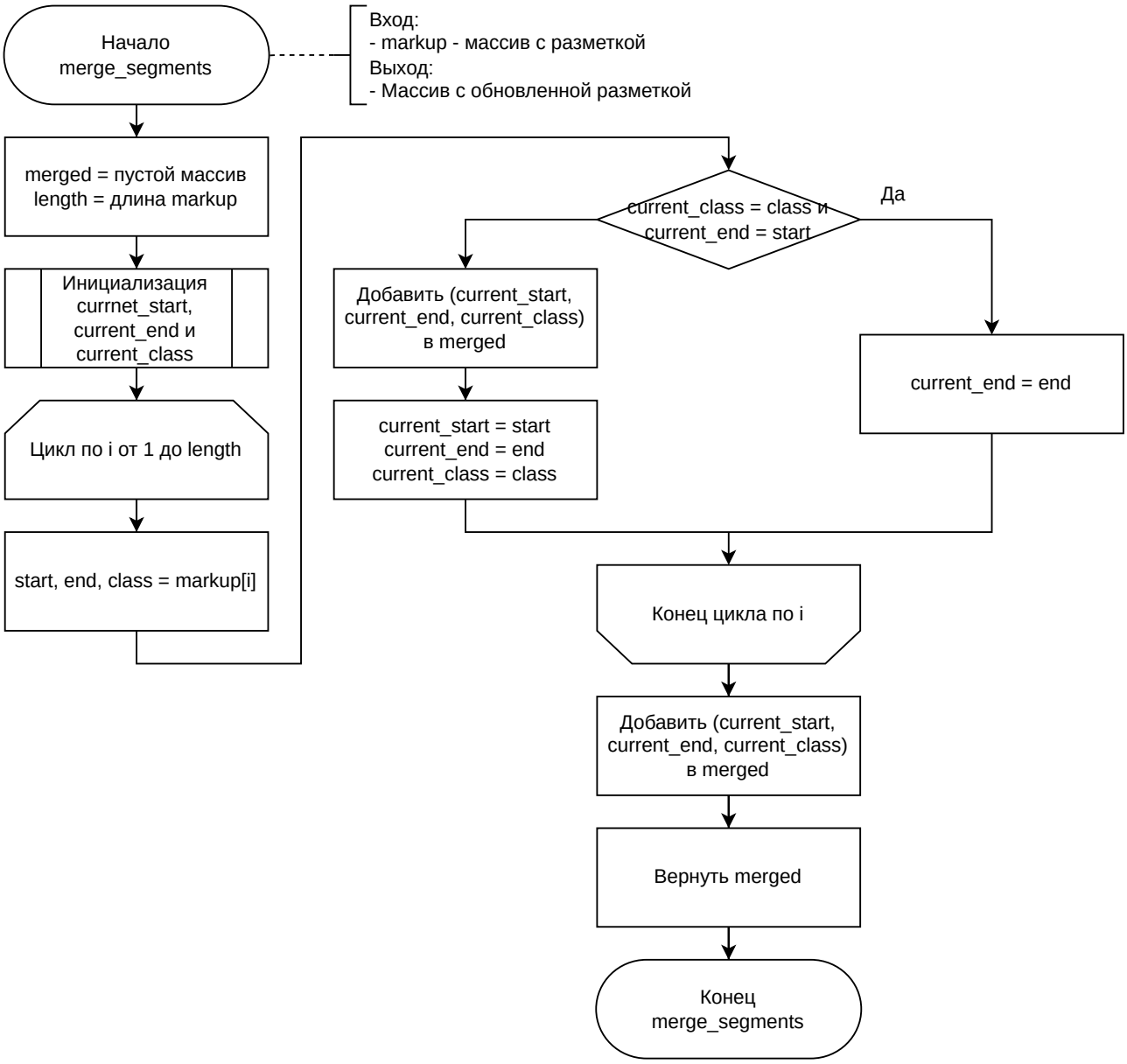


Рисунок 9 – Объединение сегментов после слияния

2.3 Тестирование и классы эквивалентности

Тестировать разрабатываемый метод имеет смысл на каждом этапе:

1) На этапе создания первичной разметки;

2) На этапе создания уточненной разметки;

3) На этапе создания объединенной разметки;

для лучшей локализации возможных ошибок.

Тестировать создаваемые разметки можно сравнивая с «эталонными» разметками — проходиться по документу построчно и сравнивать класс текущей строки с классом соответствующей строки в эталонной разметке.

Пример такого алгоритма тестирования приведен на рисунке ниже.

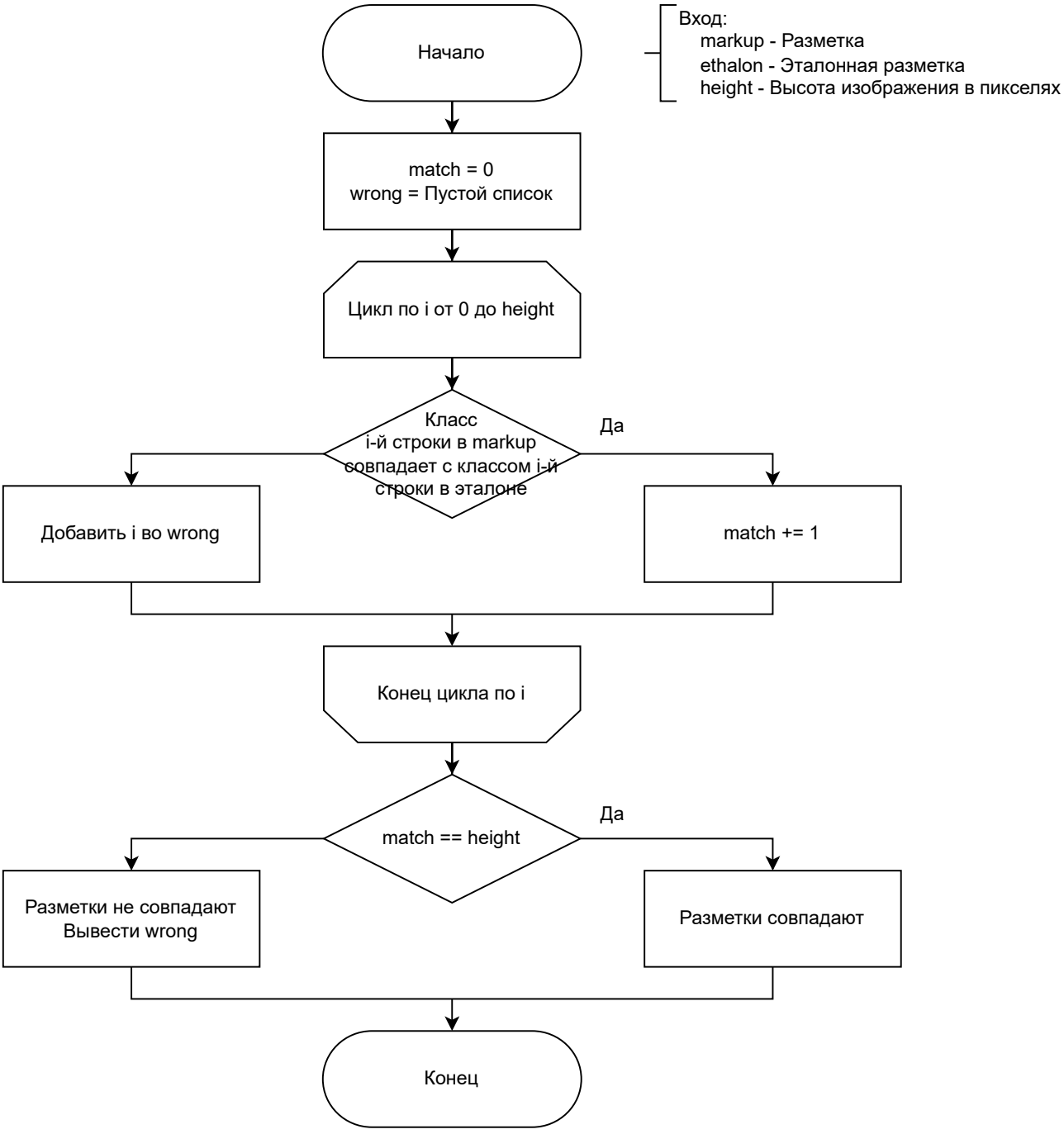


Рисунок 10 – Алгоритм сравнения разметки с эталонной

2.3.1 Тестирование первичной разметки

Классы эквивалентности при тестировании первичной разметки естественным образом соответствуют ее классам:

— Фон;

— Много текста;

— Немного текста;

— Длинная черная линия;

— Черная линия средней длины;

— Цвет;

— Не определено.

Для тестирования разметки Фона можно предложить следующие тестовые сценарии:

— Белая строка высотой в один пиксель;

— Больше одной фоновой строки подряд;

— Пустая страница.

Для тестирования вывода состояния «Много текста» можно предложить следующие тестовые сценарии:

— Строка текста на русском языке;

— Строка текста на английском языке;

— Строка текста на китайском языке;

— Строка текста длиной во всю ширину документа;

— Строка текста длиной больше половины документа.

Для тестирования вывода состояния «Немного текста» можно предложить следующие тестовые сценарии:

— Текст «Немного текста»;

— Текст «аа»;

— Строка текста длиной меньше половины документа.

Для тестирования вывода состояния «Длинная черная строка» можно предложить следующие тестовые сценарии:

— Длинная черная строка высотой в один пиксель;

— Длинная черная строка высотой в несколько пикселей;

— Листинг кода;

— Таблица;

— Черно-белое изображение в рамке.

Для тестирования вывода состояния «Черная строка средней длины» можно предложить следующие тестовые сценарии:

— Небольшая черная линия;

— Подчеркнутый текст;

— Блок схемы алгоритма;

— Схема алгоритма.

Для тестирования вывода состояния «Цвет» можно предложить следующие тестовые сценарии:

— Цветное изображение без рамки;

— Цветной текст;

— Цветной текст внутри не цветного.

Для тестирования вывода состояния «Не определено» можно предложить следующие тестовые сценарии:

— Граф сверточной нейронной сети без прямых линий;

— Черно-белое изображение без прямых черных линий.

2.3.2 Тестирование уточненной разметки

При тестировании уточненной разметки должна проверяться корректность вывода уточненного состояния для первичной разметки на основе описанных в данном разделе ранее правил.

Таким образом, классы эквивалентности при тестировании уточненной разметки следующие:

— «Не определено» → «Текст»,

— «Не определено» → «Таблица»,

— «Не определено» → «Листинг»,

— «Не определено» → «Схема алгоритма»,

— «Не определено» → «Рисунок»,

— «Не определено» → «График»,

— «Не определено» → «Не определено»,

— «Немного текста» → «Текст»,

— «Много текста» → «Таблица»,

— «Много текста» → «Листинг»,

— «Много текста» → «Текст»,

— «Цвет» → «График»,

— «Цвет» → «Не определено»,

— «Цвет» → «Рисунок»,

— «Черная линия средней длины» → «Текст»,

— «Черная линия средней длины» → «Рисунок»,

— «Черная линия средней длины» → «График»,

— «Черная линия средней длины» → «Не определено»,

— «Черная линия средней длины» → «Схема алгоритма»,

— «Длинная черная линия» → «Таблица»,

— «Длинная черная линия» → «Листинг»,

— «Длинная черная линия» → «Схема алгоритма»,

— «Длинная черная линия» → «График»,

— «Длинная черная линия» → «Не определено»,

— «Длинная черная линия» → «Рисунок».

Для каждого класса эквивалентности следует подобрать тестовые сценарии, покрывающие все правила, которые могут быть применены в конкретном классе эквивалентности для получения соответствующей уточненной разметки.

2.3.3 Тестирование объединенной разметки

При тестировании объединенной разметки должна проверяться корректность каждого этапа создания объединенной разметки.

Классы эквивалентности в таком случае соответствуют этапам создания объединенной разметки:

— Слияние небольших фоновых сегментов с ближайшим наибольшим сегментом;

— Слияние фоновых сегментов с соседними, если соседние сегменты имеют один и тот же класс;

— Смена класса небольших фоновых сегментов на «Не определено»;

— Слияние небольших «Не определенных» сегментов с наибольшими соседними сегментами;

— Объединение сегментов после слияния.

2.4 Структура разрабатываемого ПО

Структура разрабатываемого ПО представлена на рисунке 11 ниже.

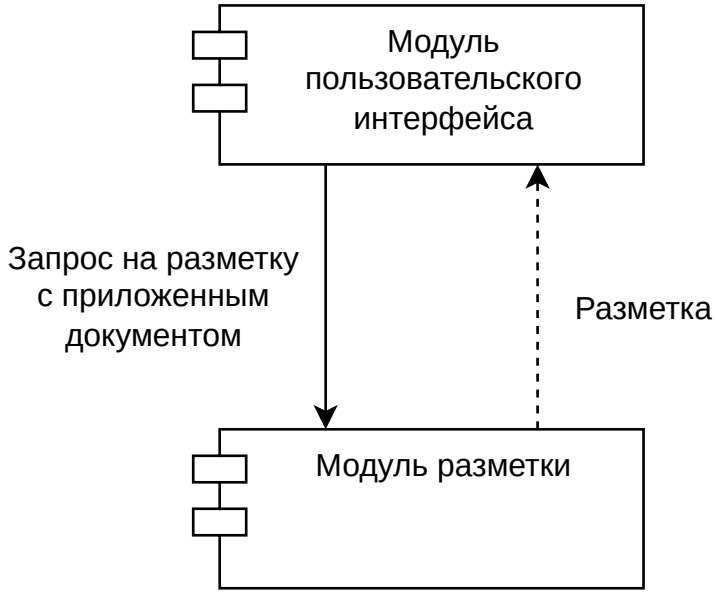


Рисунок 11 – Структура разрабатываемого ПО

Вывод

В данном разделе были приведены требования и ограничения разрабатываемого метода, были описаны основные этапы разрабатываемого метода, сценарии тестирования, классы эквивалентности тестов, а также структура разрабатываемого ПО.

3 Технологический раздел

3.1 Выбор средств реализации

Для реализации метода выделения составных частей научного текста на основе анализа распределения пикселей в сканирующей строке был выбран Python [9] по следующим причинам:

— язык программирования позволяет быстро создавать рабочие прототипы в связи с наличием большого количества библиотек;

— имеются библиотеки для работы с PDF документами и изображениями;

— есть опыт работы с данным языком программирования.

Для управления зависимостями Python-проекта и организации изолированных виртуальных окружений, обеспечивая воспроизводимость и независимость среды выполнения, был выбран uv [10].

Для работы с массивами данных (в том числе сканирующей строкой пикселей) была выбрана библиотека NumPy [11].

Для работы с изображениями и PDF были выбраны библиотеки PIL [12] и PyMuPDF [13] соответственно, так как в процессе работы программы используются как PDF документы, так и изображения, а PyMuPDF позволяет преобразовывать результат рендеринга PDF в формат, поддерживаемый PIL, который, в свою очередь, может быть преобразован в NumPy массив и использован для дальнейшей обработки.

Для создания веб-интерфейса была выбрана библиотека Gradio [14], так как данная библиотека позволяет быстро создать рабочий прототип графического интерфейса к программе.

В качестве среды разработки был выбран Neovim [15] по следующим причинам:

— данный текстовый редактор позволяет редактировать файлы с исходным кодом программы;

— быстрая и удобная навигация по файлам проекта;

— есть опыт использования данного текстового редактора.

3.2 Реализация программного обеспечения

3.3 Результаты тестирования

После разработки программного обеспечения было проведено тестирование модуля разметки. Тестирование проводилось в соответствии с описанными в конструкторском разделе классами эквивалентности. При тестировании уточненной разметки классы эквивалентности были объединены в более крупные — на основе получаемого класса разметки на выходе. Результаты тестирования представлены в листинге 1 ниже.

```
1 ========= short test summary info ==================
2 PASSED test.py::TestPrimaryMarkup::test_background
3 PASSED test.py::TestPrimaryMarkup::test_few_text
4 PASSED test.py::TestPrimaryMarkup::test_undefined
5 PASSED test.py::TestPrimaryMarkup::test_many_text
6 PASSED test.py::TestPrimaryMarkup::test_long_black_line
7 PASSED test.py::TestPrimaryMarkup::test_medium_black_line
8 PASSED test.py::TestPrimaryMarkup::test_color
9 PASSED test.py::TestSpecifiedMarkup::test_to_text
10 PASSED test.py::TestSpecifiedMarkup::test_to_table
11 PASSED test.py::TestSpecifiedMarkup::test_to_code
12 PASSED test.py::TestSpecifiedMarkup::test_to_diagram
13 PASSED test.py::TestSpecifiedMarkup::test_to_figure
14 PASSED test.py::TestSpecifiedMarkup::test_to_plot
15 PASSED test.py::TestSpecifiedMarkup::test_to_undefined
16 PASSED test.py::TestMergedMarkup::test_merge_little_bg
17 PASSED test.py::TestMergedMarkup::test_merge_bg_interpolate
18 PASSED test.py::TestMergedMarkup::test_swap_bg_to_undefined
19 PASSED test.py::TestMergedMarkup::test_merge_undefined
20 PASSED test.py::TestMergedMarkup::test_merge_segments
```

Листинг 1 – Результаты тестирования

3.4 Пользовательский интерфейс

На рисунках 12 – 14 ниже представлен порядок взаимодействия пользователя с графическим веб-интерфейсом.

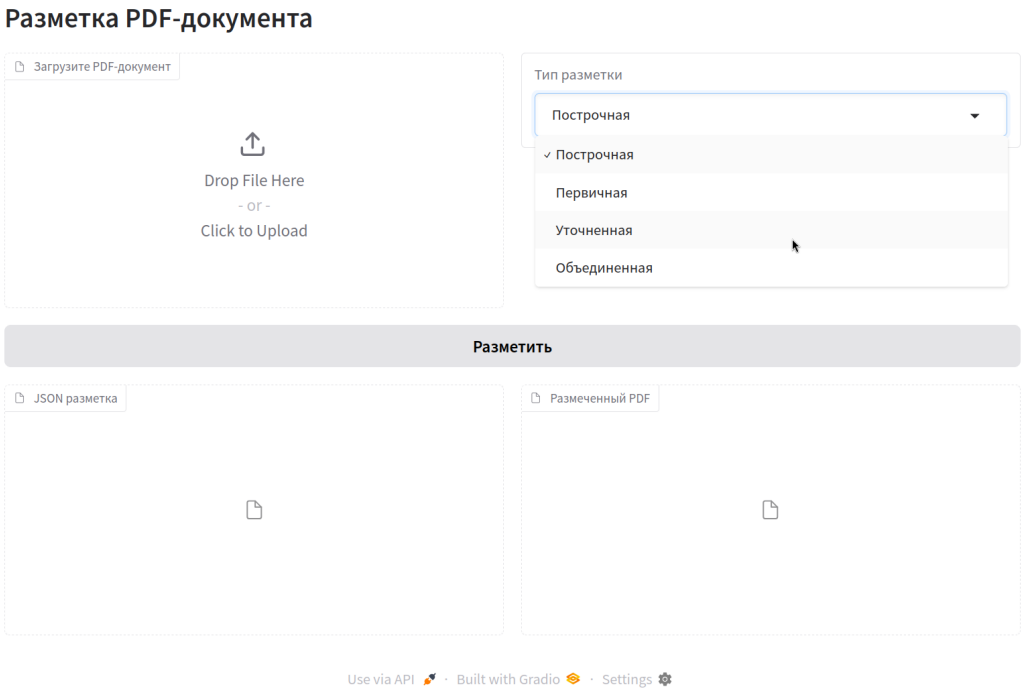


Рисунок 12 – Выбор типа разметки

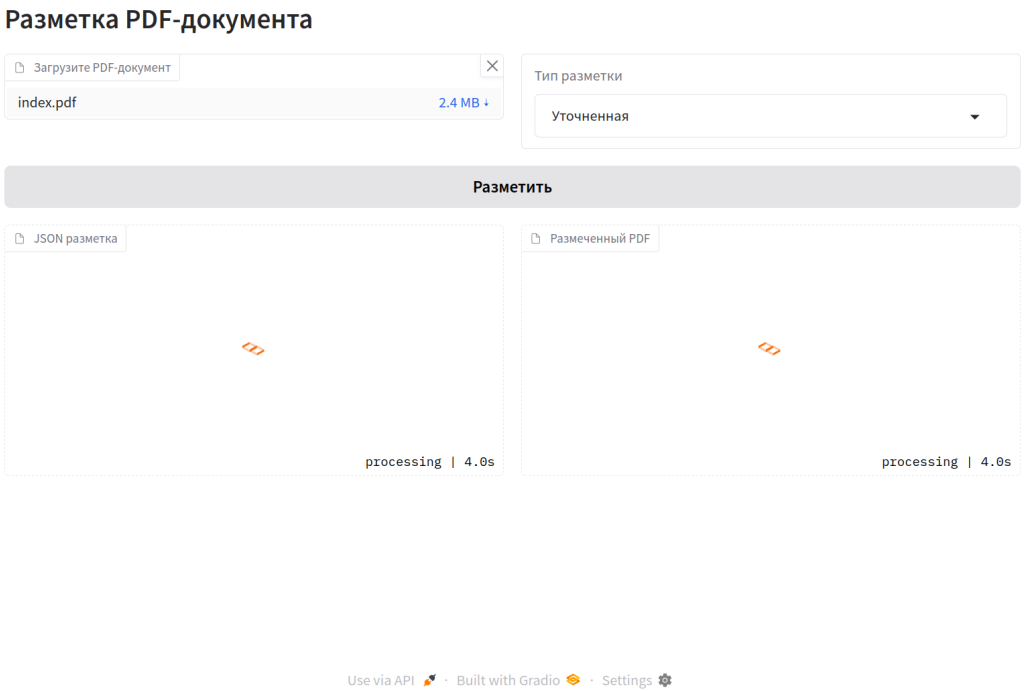


Рисунок 13 – Разметка в процессе

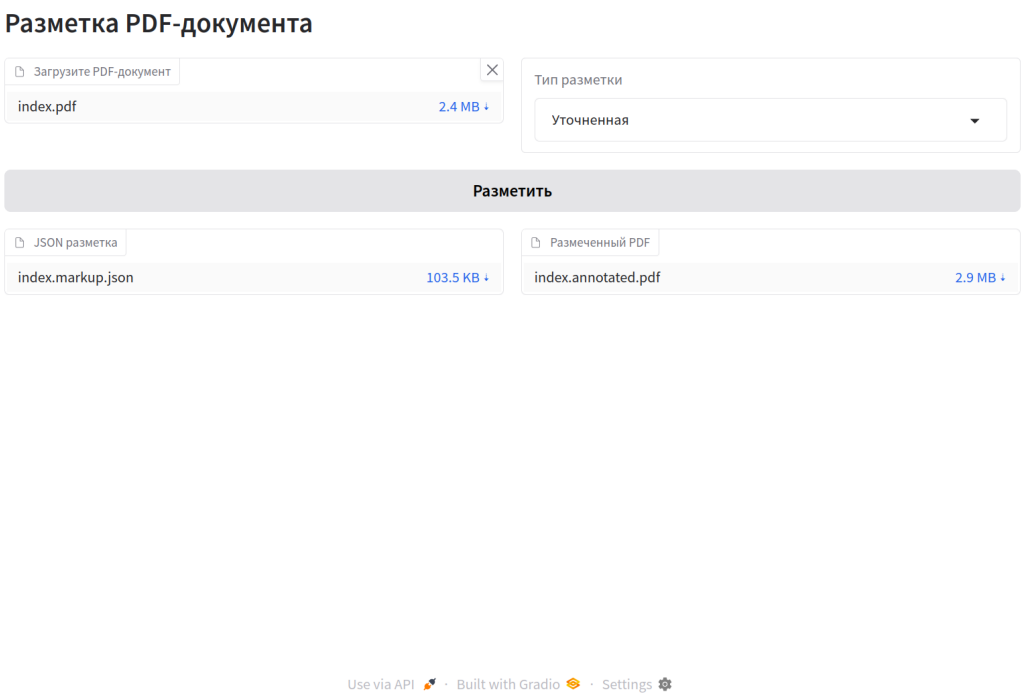


Рисунок 14 – Разметка завершена — пользователь может скачать разметку в формате JSON и/или PDF документ для ее визуализации

3.5 Демонстрация работы программы

На рисунках 15 – 18 ниже приведена демонстрация работы программы. Представлены результаты построчной (строке пикселей назначется класс на основе распределения пикселей в ней, без использования конечного автомата), первичной, уточненной и объединенной разметок.

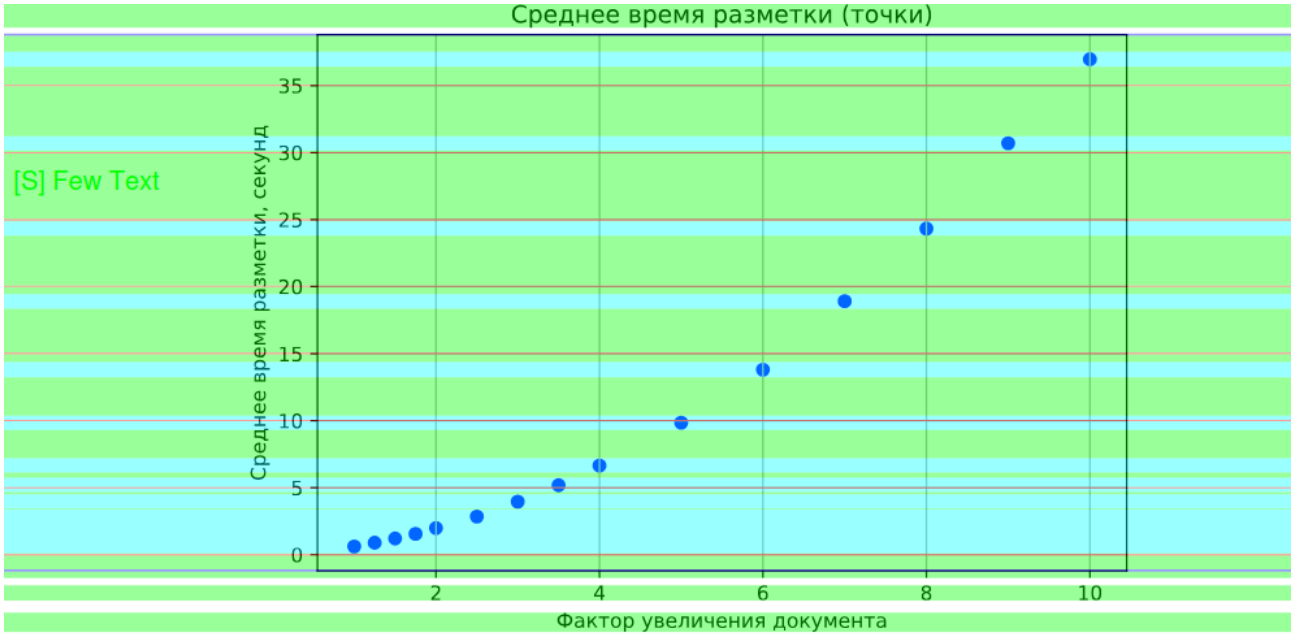


Рисунок 5 – Зависимость времени работы алгоритма от коэффициента увеличения документа при его преобразовании в изображение, исходные данные

Для определения зависимости времени разметки от фактора увеличения документа проведем степенную аппроксимацию на полученных данных.

[S] Few Text

```
1 import numpy as np
2 x = np.array([ ... ])
3 y = np.array([ ... ])
4 log_x = np.log(x)
5 log_y = np.log(y)
6 b, log_c = np.polyfit(log_x, log_y, 1)
7 c = np.exp(log_c)
8 print(f'y = {c:.3f} * x^{b:.3f}') # y = 0.583 * x^1.782
```

Листинг 3 – Степенная аппроксимация

Рисунок 15 – Пример построчной разметки

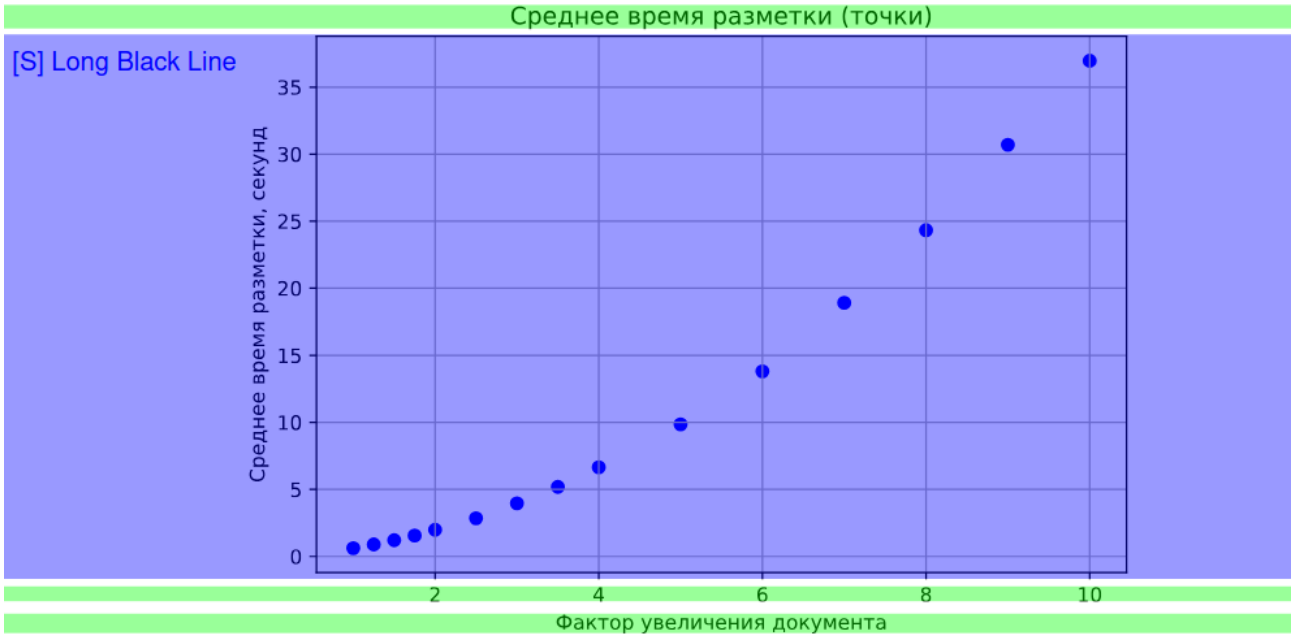


Рисунок 5 – Зависимость времени работы алгоритма от коэффициента увеличения документа при его преобразовании в изображение, исходные данные

Для определения зависимости времени разметки от фактора увеличения документа проведем степенную аппроксимацию на полученных данных.

```
import numpy as np
x = np.array([ ... ])
y = np.array([ ... ])
log_x = np.log(x)
log_y = np.log(y)
b, log_c = np.polyfit(log_x, log_y, 1)
c = np.exp(log_c)
print(f'y = {c:.3f} * x^{b:.3f}') # y = 0.583 * x^1.782
```

Листинг 3 – Степенная аппроксимация

Рисунок 16 – Пример первичной разметки

Plot

Рисунок 5 – Зависимость времени работы алгоритма от коэффициента увеличения документа при его преобразовании в изображение, исходные данные

Для определения зависимости времени разметки от фактора увеличения документа проведем степенную аппроксимацию на полученных данных.

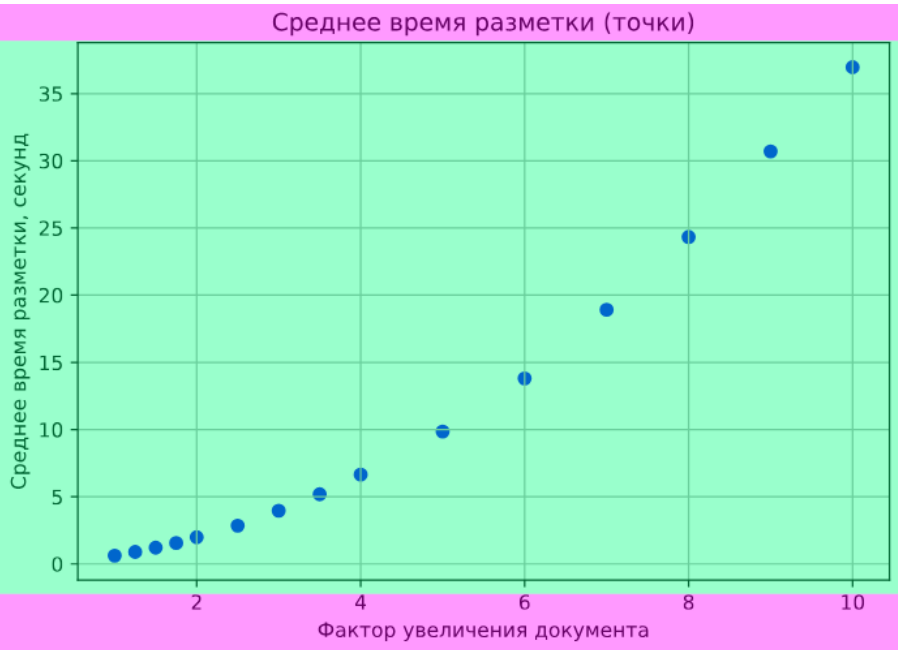
Code

```
import numpy as np
x = np.array([ ... ])
y = np.array([ ... ])
log_x = np.log(x)
log_y = np.log(y)
b, log_c = np.polyfit(log_x, log_y, 1)
c = np.exp(log_c)
print(f'y = {c:.3f} * x^{b:.3f}') # y = 0.583 * x^1.782
```

Листинг 3 – Степенная аппроксимация

Рисунок 17 – Пример уточненной разметки

Plot



Text

Рисунок 5 – Зависимость времени работы алгоритма от коэффициента увеличения документа при его преобразовании в изображение, исходные данные

Для определения зависимости времени разметки от фактора увеличения документа проведем степенную аппроксимацию на полученных данных.

Code

```
import numpy as np
x = np.array([ ... ])
y = np.array([ ... ])
log_x = np.log(x)
log_y = np.log(y)
b, log_c = np.polyfit(log_x, log_y, 1)
c = np.exp(log_c)
print(f'y = {c:.3f} * x^{b:.3f}') # y = 0.583 * x^1.782
```

Text

Листинг 3 – Степенная аппроксимация

Рисунок 18 – Пример объединенной разметки

В листингах 2 и 3 ниже представлен результат работы программы при запуске из терминала.

Для запуска программы требуется установить uv командой:

```
uv run main.py ~/index.pdf x.json 3
Pages processed: 31/31 [00:06<00:00,  5.11it/s]
```

Листинг 2 – Запуск и результат работы разметки из терминала

```
uv run apply.py ~/index.pdf x.json output.pdf
Annotating pages: 31/31 [00:00<00:00, 383.41it/s]
Сохранено в: output.pdf
```

Листинг 3 – Запуск и результат применения разметки к PDF из терминала

3.6 Руководство пользователя

Для запуска программы требуется установить uv командой:

```
curl -LsSf https://astral.sh/uv/install.sh | sh
```

Листинг 4 – Установка uv

После установки uv, следует установить зависимости приложения. Для этого, находясь в директории с программой, следует прописать команду:

```
uv sync
```

Листинг 5 – Установка зависимостей

Графический интерфейс запускается локально командой:

```
uv run webgui.py
```

Листинг 6 – Запуск графического веб-интерфейса

Также можно создать разметку без использования графического интерфейса. Для этого используется скрипт main.py:

```
Использование: main.py [-w КОЛИЧЕСТВО_ПРОЦЕССОВ]
                       [-p СТРАНИЦЫ]
                       pdf_path json_path {0,1,2,3}

Сегментировать PDF страницы и экспортировать разметку в JSON.

Позиционные аргументы:
```

```
  pdf_path              Путь ко входному PDF файлу
  json_path             Путь к выходному JSON файлу
  {0,1,2,3}             Тип разметки: 0 - построчная разметка,
                                      1 - первичная разметка,
                                      2 - уточненная разметка,
                                      3 - объединенная разметка

Опции:
  -w, --workers КОЛИЧЕСТВО_ПРОЦЕССОВ
                        Количество рабочих процессов
                        (по умолчанию: 8)
  -p, --pages СТРАНИЦЫ  Диапазоны страниц для обработки,
                        например, '1-3,5,7-9'
```

Листинг 7 – Запуск скрипта для создания разметки

Для применения разметки к документу без графического интерфейса используется скрипт apply.py:

```
Использование: apply.py [-h] pdf_path json_path output_path

Разметить PDF цветными сегментами из JSON файла разметки.

Позиционные аргументы:
  pdf_path     Путь ко входному PDF файлу
  json_path    Путь к JSON файлу с разметкой
  output_path  Путь для сохранения размеченного PDF
```

Листинг 8 – Запуск скрипта для применения разметки к PDF документу

Вывод

4 Исследовательский раздел

4.1 Описание исследования

Технические характеристики

4.2 Результаты исследования

Вывод

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Bhowmik et al. Text and non-text separation in offline document images: a survey // International Journal on Document Analysis and Recognition (IJDAR). 2018. Т. 21.

2. Binmakhashen G.M., Mahmoud S.A. Document Layout Analysis: A Comprehensive Survey // ACM Comput. Surv. 2019. Т. 52, № 6.

3. Bhowmik S. Document Layout Analysis. — Springer Singapore, 2023 — 86 с.

4. Kasturi R., O'Gorman L., Govindaraju V. Document image analysis: A primer // Sadhana — Academy Proceedings in Engineering Sciences. 2002. Т. 27. С. 3–22.

5. Kise K. Page Segmentation Techniques in Document Analysis // Doermann D., Tombre K. Handbook of Document Image Processing and Recognition. — Springer London, 2014. С. 135–175.

6. Бутенко Ю.И. Модель текста научно-технической статьи для разметки в корпусе научно-технических текстов // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2022. Т. 20, № 1. С. 5–13.

7. Романов Д.А. Кратко о структуре экспериментальной научной статьи на английском языке // Вестник Казанского технологического университета. 2014. Т. 17, № 6. С. 325–327.

8. Раицкая Л.К. Структура научной статьи по политологии и международным отношениям в контексте качества научной информации // Полис. Политические исследования. 2019. № 1. С. 167–181.

9. Python — A programming language that lets you work quickly and integrate systems more effectively [Электронный ресурс]. – URL: https://www.python.org/ (дата обращения: 26.05.2025).

10. uv — An extremely fast Python package and project manager, written in Rust [Электронный ресурс]. – URL: https://docs.astral.sh/uv/ (дата обращения: 26.05.2025).

11. NumPy — The fundamental package for scientific computing with Python [Электронный ресурс]. – URL: https://numpy.org/ (дата обращения: 26.05.2025).

12. Pillow — The Python Imaging Library adds image processing capabilities to your Python interpreter [Электронный ресурс]. – URL: https://pypi.org/project/pillow/ (дата обращения: 26.05.2025).

13. PyMyPDF — A high-performance Python library for data extraction, analysis, conversion and manipulation of PDF (and other) documents [Электронный ресурс].

14. Gradio — Build and share delightful machine learning apps [Электронный ресурс]. – URL: https://www.gradio.app/ (дата обращения: 26.05.2025).

15. Neovim — Hyperextensible Vim-based text editor [Электронный ресурс]. – URL: https://neovim.io (дата обращения: 26.05.2025).

ПРИЛОЖЕНИЕ А