

Московский физико-технический институт (государственный университет)
Факультет Управления и Прикладной Математики
Кафедра интеллектуальных систем

На правах рукописи

Чувилин Кирилл Владимирович

**Автоматический синтез правил коррекции
текстовых документов формата \LaTeX**

05.13.17 – Теоретические основы информатики

ДИССЕРТАЦИЯ

на соискание ученой степени

кандидата технических наук

Научный руководитель

д. ф.-м. н.

Воронцов Константин Вячеславович

Москва – 2013 г. (редакция 2014 г.)

Данный документ представляет собой отредактированную версию текста диссертации, представленного на защиту, которая успешно состоялась 12-го декабря 2013 г. в диссертационном совете Д 002.017.02 при Федеральном государственном бюджетном учреждении науки Вычислительный центр им. А. А. Дородницына Российской академии наук. В этой редакции я, по возможности, учёл замечания, указанные в отзывах ведущей организации — Федерального государственного бюджетного учреждения науки Институт системного анализа Российской академии наук — и официальных оппонентов: Ульянова Михаила Васильевича и Гурова Сергея Исаевича. Также я исправил некоторые другие опечатки и неточности, не замеченные ранее, а рисунки заменил на более качественные. Это было сделано в надежде на то, что полученные мной результаты найдут применение в работах других исследователей.

Кирилл Чувиллин,

14.11.2014

`kirill.chuvilin@gmail.com`

Оглавление

Введение	6
Глава 1. Постановка задачи. Обзор существующих методов.	
Предлагаемый подход	14
1.1. Постановка задачи синтеза правил коррекции типографических ошибок	18
1.2. Обзор существующих методов	19
1.2.1. Автоматизация обработки текстов	19
1.2.2. Интеллектуальная коррекция ошибок в поисковых запросах	22
1.2.3. Автоматический перевод, основанный на анализе параллельных тестов	23
1.2.4. Обработка исходного кода программ	24
1.3. Предлагаемый подход	26
1.3.1. Анализ структуры документов формата \LaTeX	26
1.3.2. Выделение различий, вносимых корректорами в документы	27
1.3.3. Синтез правил коррекции	27
1.4. Выводы главы 1	28
Глава 2. Структура документов формата \LaTeX	30
2.1. Система компьютерной верстки \LaTeX	30
2.1.1. Кратко о \TeX е	30
2.1.2. Кратко о \LaTeX е	31
2.2. Структура разметки документа формата \LaTeX	32
2.2.1. Состояния	33

2.2.2.	Типы элементов	34
2.2.3.	Типы лексем	36
2.2.4.	Действия и операнды	38
2.3.	Синтаксическое дерево документа \LaTeX	39
2.3.1.	Элементы синтаксического дерева	39
2.4.	Выводы главы 2	43
Глава 3.	Алгоритмы сравнения документов формата \LaTeX	45
3.1.	Выделение различий между конечными последовательностями	46
3.1.1.	Построение отображения	47
3.1.2.	Применение для документов формата \LaTeX	50
3.2.	Построение различий между деревьями	51
3.2.1.	Алгоритм Zhang–Shasha	53
3.2.2.	Применение для синтаксических деревьев \LaTeX	58
3.3.	Гибридный алгоритм	61
3.3.1.	Разбиение текста	61
3.3.2.	Отображение фрагментов текста	62
3.3.3.	Отображение символов	62
3.3.4.	Отображение токенов	62
3.3.5.	Структура алгоритма	63
3.3.6.	Эксперимент	65
3.4.	Выводы главы 3	66
Глава 4.	Правила коррекции	68
4.1.	Правила коррекции с простой структурой	68
4.1.1.	Синтез правил с линейным шаблоном	69
4.1.2.	Поиск соответствий правилам	72
4.1.3.	Предварительная оценка правила	73

4.1.4.	Выбор оптимальных шаблонов	73
4.1.5.	Редукция набора правил	73
4.1.6.	Операции поднятия и опускания	74
4.1.7.	Оценки качества набора правил	78
4.1.8.	Эксперимент	80
4.2.	Групповые правила	82
4.2.1.	Построение групповых правил	85
4.2.2.	Применение групповых правил	85
4.2.3.	Эксперимент	85
4.3.	Правила с древовидными шаблонами	86
4.3.1.	Построение древовидных правил	86
4.3.2.	Применение древовидных правил	87
4.3.3.	Эксперимент	89
4.4.	Выводы главы 4	90
Заключение		92
Список литературы		94
Приложение А. Символы \LaTeX		101
Приложение Б. Команды \LaTeX		105
Приложение В. Окружения \LaTeX		121
Приложение Г. Примеры синтезированных правил коррекции		123

Введение

Актуальность темы. В связи с ростом числа электронных научных изданий постоянно увеличивается число издательств, редакционно-издательских отделов вузов и научных учреждений, индивидуальных авторов, использующих систему компьютерной верстки \LaTeX , которая является стандартом де-факто для научного общения и публикаций. Постоянно растёт доля электронных изданий, к которым предъявляются повышенные требования оперативности публикаций.

При этом, уровень подготовки пользователей в области компьютерной верстки, знания типографических правил и традиций остается невысоким. К таким правилам относятся оформление заголовков, списков, таблиц, библиографии, формул, чисел, и многое другое. Ошибки, связанные с несоблюдением этих правил, называются типографическими. При текущем уровне технологий исправление таких ошибок производится корректорами вручную, что требует значительных затрат времени. Большинство ошибок являются типовыми, и это создает предпосылки для автоматизации процесса корректуры.

Автоматизация стадии корректуры при подготовке научных изданий позволила бы существенно сократить затраты и сроки и повысить качество верстки. В данной работе эта задача ставится как задача автоматической обработки текста и решается методами машинного обучения. Такой подход к проблеме автоматизации корректуры до сих пор не применялся.

Степень разработанности темы. Существуют инструменты для облегчения процесса ручной корректуры [7], но, тем не менее, обработка одной страницы занимает до двух часов времени. Вообще говоря, идея автоматизации коррекции текстов не нова [8], и на данный момент существуют качественные инструменты для автоматического поиска и исправления орфографических ошибок [9], использующие словари и морфологический анализ

словоформ текста. Кроме того, схожие проблемы возникают для интеллектуальной коррекции ошибок в запросах поиска [10] с помощью лексических и статистических признаков и для автоматизации перевода текстов на естественных языках статистическими методами [11]. Но подобные подходы не применимы для исправления типографических ошибок, рассматриваемых в данной работе, которые связаны не только с текстовым содержанием документа, но и разметкой форматирования, и зачастую для описания ошибки не достаточно локальной информации в тексте, но также требуется знание контекста, дополнительной информации о позиции в структуре документа.

С другой стороны, существует область исследований, посвящённая улучшению характеристик исходного кода программ (вероятности возникновения ошибок в отдельных модулях, степени связности модулей и др.). Известны методы [12, 13], позволяющие оценивать характеристики, основываясь на анализе истории изменений репозитория, и использовать их для поиска ошибок в коде. Они позволяют создавать рекомендательные системы [14] для улучшения качества кода программы при редактировании. Документы в формате \LaTeX можно рассматривать как исходный код, который используется компилятором \TeX , но в издательской практике не распространено использование репозитория, пригодных для последующего анализа, нет единых стандартов, и, кроме того, текстовое содержимое документов не может быть подвержено подобной обработке.

Таким образом, возникает необходимость нового исследования, направленного непосредственно на автоматизацию процесса исправления типографических ошибок.

Цели и задачи исследования. Объектом исследования являются хорошо структурированные текстовые документы, которые могут быть описаны с помощью синтаксического дерева. Предмет исследования — алгоритмы автоматического синтеза правил коррекции структурированных текстовых документов

по выборке пар «черновик–чистовик».

Целью диссертационного исследования является разработка методов, алгоритмов и технологий для создания автоматизированной системы, позволяющей многократно повысить эффективность труда корректоров при работе с текстовыми документами формата \LaTeX .

Для достижения цели исследования в диссертации решаются следующие задачи.

1. Разработка эффективных алгоритмов для представления и сравнения файлов в формате \LaTeX как древовидных структур данных.
2. Формализация описания правил коррекции документов формата \LaTeX и разработка эффективных алгоритмов поиска мест ошибок в документах и синтеза правил для их исправления. Множество проблем вызваны тем, что при ручной обработке документов корректоры придерживаются недостаточно формализованных рекомендаций. И составление вручную достаточно полного описания набора правил для автоматического использования трудно реализуемо. Некоторые из используемых рекомендаций довольно сложны и сильно зависят от контекста, что требует сложных моделей для описания правил коррекции.
3. Задача автоматического синтеза правил коррекции текстовых документов формата \LaTeX заключается в построении совокупности формальных инструкций, которые могут быть использованы в алгоритмах локализации ошибок (определение фрагментов текста, содержащих ошибки) и исправления ошибки (построение ранжированного списка вариантов замены фрагмента текста, содержащего ошибку).
4. Разработка методики оценивания синтезированных правил коррекции для последующего ранжирования. Это необходимо при выборе наибо-

лее подходящих вариантов найденной ошибки для предоставления их пользователю.

5. Экспериментальное исследование полноты и точности разработанных алгоритмов сравнения документов и построения правил коррекции с использованием корпуса реальных статей.

Научная новизна. В работе впервые предложен подход к синтезу правил коррекции структурированных (обладающими синтаксическим деревом) текстовых документов по обучающей выборке, составленной из пар документов «черновик–чистовик». Задача автоматизации корректуры структурированных текстовых документов никогда ранее не ставилась как задача синтеза правил коррекции методами машинного обучения.

В работе предложен новый гибридный алгоритм для выявления различий между структурированными (обладающими синтаксическим деревом) текстовыми документами, который корректно учитывает логическую структуру текстов, но при этом, как минимум, в три раза быстрее алгоритма, основанного на сравнении только синтаксических деревьев.

Теоретическая и практическая значимость. Теоретическая ценность работы заключается в том, что предложены подход для синтеза правил автоматической коррекции структурированных текстов по обучающей выборке, составленной из пар документов «черновик–чистовик», и методика оценки качества таких правил. Кроме того, разработан быстрый и менее требовательный к памяти, по сравнению с аналогами, алгоритм сравнения синтаксических деревьев документов в формате \LaTeX .

Практическая ценность результатов диссертации заключается в том, что разработанные методы, алгоритмы и технологии позволяют реализовать систему автоматизации корректуры, в несколько раз сокращающую трудозатраты при коррекции текстовых документов формата \LaTeX . При этом автоматизи-

руются процессы поиска различий между структурированными документами, поиска возможных типографических ошибок, синтеза правил коррекции, формирования наборов вариантов исправления.

Предлагаемый подход. В данной работе предлагается формально описывать правила автоматической коррекции. Для этого каждый документ в формате \LaTeX отождествляется с синтаксическим деревом, для которого и формулируются правила [4].

Обучающая выборка составляется из пар документов: черновик (документ, не прошедший обработку профессиональным корректором) и чистовик (документ, содержащий корректорские правки). Для сравнения синтаксических деревьев используется гибридный алгоритм, который учитывает и текстовую природу документов \LaTeX , и их древовидную структуру [5]. В результате работы алгоритма строится отображение вершин синтаксического дерева черновика в вершины дерева чистовика.

Построенное отображение используется для синтеза правил, из которых каждое характеризуется шаблоном (линейным или древовидным), применяющимся к вершинам синтаксического дерева. На основе предварительных оценок точности строятся групповые правила [6].

Для оптимизации построенного набора правил коррекции и последующего их ранжирования строятся оценки качества на основе статистики применимости правил к документам обучающей выборки [2].

Результаты, выносимые на защиту.

1. Алгоритм сравнения структурированных текстов, использующий их представление в виде синтаксических деревьев (на примере текстов формата \LaTeX).
2. Алгоритмы построения линейных, древовидных и групповых правил коррекции документов по обучающей выборке пар документов «чер-

новик–чистовик», позволившие достичь точности 76% и полноты 69% на коллекции из 85 пар документов.

3. Программа для построения набора правил коррекции документов и эмпирического оценивания полноты и точности построенного набора.

Достоверность результатов. Обоснованность и достоверность результатов и выводов подтверждена:

- сравнением реализованных алгоритмов и подходов с аналогами;
- опытом практического применения результатов исследования на реальных коллекциях текстовых документов;
- обсуждением результатов исследования на российских и международных научных конференциях;
- публикациями результатов исследования в рецензируемых научных изданиях, в том числе рекомендованных ВАК РФ.

Апробация результатов исследования. Основные результаты диссертационного исследования докладывались на следующих конференциях:

- 54-я научная конференция Московского физико-технического института (Долгопрудный, 2011 г.),
- Международная научная конференция студентов, аспирантов и молодых учёных «Ломоносов-2012» (Москва, 2012 г.),
- Вторая научная конференция молодых ученых «Теория и практика системного анализа» ТПСА-2012 (Рыбинск, 2012 г.),
- Девятая международная конференция «Интеллектуализация обработки информации» ИОИ-2012 (Черногория, Будва, 2012 г.),

- 55-я научная конференция Московского физико-технического института (Долгопрудный, 2012 г.),
- 16-я всероссийская конференция с международным участием «Математические методы распознавания образов — 2013» ММРО-16 (Казань, 2013 г.).

В рамках работы над диссертацией был реализован прототип системы полуавтоматической коррекции типографических ошибок. Проект «Самообучающаяся система для автоматизации коррекции документов в формате \LaTeX » прошел отборочные этапы программы «Участник молодежного научно-инновационного конкурса» («У.М.Н.И.К.») и вошел в число победителей конкурса в 2012 году¹.

Основные результаты работы опубликованы в [1–3, 6], в том числе в изданиях [4, 5], входящих в список ВАК.

Обоснование специальности. Данная работа по своей тематике и направленности полученных результатов соответствует следующим пунктам паспорта специальности 05.13.17 — «Теоретические основы информатики»:

1. Исследование, в том числе с помощью средств вычислительной техники, информационных процессов, информационных потребностей коллективных и индивидуальных пользователей.
2. Исследование информационных структур, разработка и анализ моделей информационных процессов и структур.
3. Исследование методов и разработка средств кодирования информации в виде данных. Принципы создания языков описания данных, языков манипулирования данными, языков запросов. Разработка и исследование моделей данных и новых принципов их проектирования.

¹ http://miptic.ru/UMNIK/a_5lekjo.html

5. Разработка и исследование моделей и алгоритмов анализа данных, обнаружения закономерностей в данных и их извлечениях разработка и исследование методов и алгоритмов анализа текста, устной речи и изображений.
7. Разработка методов распознавания образов, фильтрации, распознавания и синтеза изображений, решающих правил. Моделирование формирования эмпирического знания.

Структура и объем диссертации. Диссертация состоит из введения, 4 глав основного содержания, заключения, библиографии и 4 приложений. Работа содержит 124 страницы основного текста, включая 25 иллюстраций. Перечень библиографических источников включает 74 наименования.

Глава 1

Постановка задачи. Обзор существующих методов. Предлагаемый подход

Многие научные издательства и конференции работают с издательской системой \LaTeX [15]. И в каждом издательстве есть определённые традиции и требования к оформлению публикуемого материала [16–19]. К ним относятся оформление заголовков, списков, таблиц, библиографии, формул, чисел и многое другое. Ошибки, связанные с несоблюдением этих правил, называются типографическими. Как правило, рукописи в формате \LaTeX , присылаемые авторами для публикации, содержат большое число таких ошибок.

Пример 1. Ниже приведены собранные вместе фрагменты исходных текстов статей сборника конференции ИОИ-8 [20], содержащие типовые типографические ошибки (места ошибок выделены красным цветом).

```
\begin{document}
\organization{Новосибирск, Институт математики
им. С.Л.Соболева СО РАН, Новосибирский государственный
университет}
\thanks{Работа выполнена при финансовой поддержки РФФИ, проекты
\#08-07-00129-а, \#10-07-00135-а, \#10-07-00478-а.}
\author[]{Устинин~М.\,Н., Панкратова~Н.\,М., Панкратов~А.\,Н.}
В_настоящей работе представлены результаты исследования,
полученные в_работах [1]-[2].
{\bf Задача  $\$m\$$ ПВО}: {\it Задана последовательность векторов
в_ $\$_R^k$  и натуральные числа  $\$m$  и  $\$l$ , удовлетворяющие условию
 $\$lm < n$ .\$ Требуется выделить в_ $\$_V$  подмножество векторов
```

$U = \{\vec{v}_{a_1}, \dots, \vec{v}_{a_m}\}$, обладающее максимальной нормой суммы, при соблюдении ограничений на номера соседних векторов выделенного подмножества:

$$a_{i+1} - a_i \geq 1 \quad \text{для } i = 1, 2, \dots, m-1$$

где L --- параметр алгоритма.

Специфика задачи позволяет сделать это за то же время, т.е.

$$O(k^2 n^{2k}).$$

Если Γ --- гладкая кривая на плоскости \mathbb{R}^2 ,

то кривизну можно рассматривать как результат действия

$$\text{на } \Gamma \text{ оператора } \text{Cur}_R \{ \cdot \}; \quad \text{Cur}_R \{ \mathbf{g} \} =$$

$$\text{Cur}_R [\Gamma] (\mathbf{g}).$$

$$\mathbf{g}_k \psi \mathbf{g}_{k+1} \quad \text{для всех } k=0, \dots, n-1,$$

где ψ --- некоторое отношение связности.

B --- промежуточный ключ второго пользователя.

Для анализируемой алгоритмической схемы в работе строится ее

абстрактная модель в виде Сети Петри (СП) --- \cite{Kot}.

Модель схемы представляет собой тройку (S, W, M_0) ,

где S - сеть, $W: F \rightarrow N$ - функция кратности дуг.

`\begin{thebibliography}{1}`

`\bibitem{BGGP}`

`\BibAuthor{Бабурин_А._Е., Гимади_Э._Х., Глебов_Н._И.,`

`Пяткин_А._В.}`

`\BibTitle{Задача отыскания подмножества векторов с максимальным суммарным весом} //`

Дискрет. анализ и исслед. операций, Серия 2, 2007, Т. 14, N 1.

С. 22-32.

`\bibitem{Luk}`

`\BibAuthor{Лукьянова_Е._А.}`

```

\BibTitle{Метод верификации свойств реактивной системы на_модели}
//Таврический вестник информатики и
математики. ---2006. ---№2. ---С.60-68.
\end{thebibliography}
\end{document}

```

В следующих сроках представлены те же примеры фрагментов текста, но с исправленными типографическими ошибками согласно правилам конференции (места ошибок выделены синим цветом).

```

\begin{document}
\organization{Новосибирск, Институт математики
им. С.\,Л-Соболева СО РАН, Новосибирский государственный
университет}
\thanks{Работа выполнена при финансовой поддержки РФФИ, проекты
\No\,08-07-00129-а, \No\,10-07-00135-а, \No\,10-07-00478-а.}
\author{Устинин~М.\,Н., Панкратова~Н.\,М., Панкратов~А.\,Н.}
В-настоящей работе представлены результаты исследования,
полученные в-работах [1]--[2].
\textbf{Задача  $\mathbb{P}BO$ }: \emph{Задана последовательность векторов
в- $\mathbb{R}^k$  и натуральные числа  $m$  и  $l$ , удовлетворяющие условию
 $l < n$ . Требуется выделить в- $V$  подмножество векторов
 $U = \{\vec{v}_{a_1}, \dots, \vec{v}_{a_m}\}$ , обладающее
максимальной нормой суммы, при-соблюдении ограничений на-номера
соседних векторов выделенного подмножества:
 $a_{i+1} - a_i \geq l$  \mbox{ для }  $i = 1, 2, \dots, m-1$ .}
 $(k-1)/(8L^2)$ , где  $L$  --- параметр алгоритма.
Специфика задачи позволяет сделать это за-то-же время, т.\,е.
за-время  $O(k^2 n^{2k})$ .
Если  $\Gamma$  --- гладкая кривая на-плоскости  $\mathbb{R}^2$ ,

```


то-кривизну можно рассматривать как-результат действия на- Γ оператора $Cur_R \colon k(\mathbf{g}) = Cur_R [\Gamma](\mathbf{g})$.

$\mathbf{g}_k \psi_{k+1}$

для-всех $k=0, \dots, n-1$,

где- ψ --- некоторое отношение связности.

ψ --- промежуточный ключ второго пользователя.

Для анализируемой алгоритмической схемы в-работе строится ее абстрактная модель в-виде Сети Петри (СП) [Kot].

Модель схемы представляет собой тройку (S, W, M_0) ,

где S --- сеть, $W \colon F \rightarrow N$ --- функция кратности дуг.

`\begin{thebibliography}{1}`

`\bibitem{BGGP}`

`\BibAuthor{Бабурин~А.\,Е., Гимади~Э.\,Х., Глебов~Н.\,И.,
Пяткин~А.\,В.}`

`\BibTitle{Задача отыскания подмножества векторов с~максимальным
суммарным весом}~//`

Дискрет. анализ и исслед. операций, Серия 2, 2007, Т.\,14, \No\,1.
С.\,22-32.

`\bibitem{Luk}`

`\BibAuthor{Лукьянова~Е.\,А.}`

`\BibTitle{Метод верификации свойств реактивной системы
на~модели}~//`

Таврический вестник информатики и
математики. --- 2006. --- \No\,2. --- С.\,60--68.

`\end{thebibliography}`

`\end{document}`

При текущем уровне технологий исправление типографических ошибок производится корректорами вручную. Это порождает сложности, которые можно разделить на две категории. Во-первых, проблемы, связанные со временем: на обработку одной страницы текста может уходить до двух часов. Во-вторых, проблемы, связанные с качеством: для профессиональной подготовки издания требуется обладать довольно большим корректорским опытом. Кроме того, работа рутинная, и трудно замечать все недостатки.

Важно отметить, что очень много типовых ошибок. Это создает предпосылки для автоматизации процесса исправления типографических ошибок с помощью формально описанных правил коррекции, что позволило бы существенно сократить время ручной работы.

1.1. Постановка задачи синтеза правил коррекции типографических ошибок

Под правилом коррекции подразумевается формально описанная инструкция, которая может быть использована алгоритмом для:

- локализации ошибки в документе формата \LaTeX (определение фрагмента исходного текста, содержащего ошибку),
- предложения варианта исправления (построения текста для замены фрагмента с ошибкой).

Рассматривается постановка задачи автоматического синтеза правил коррекции текстовых документов формата \LaTeX как задачи обучения по прецедентам [21–23].

Пусть X — множество пар документов: черновик (документ, не прошедший обработку профессиональным корректором) и чистовик (документ, содер-

жащий корректорские правки). R — множество правил коррекции документов. И пусть дана обучающая выборка $X^m = \{x_1, \dots, x_m\}$ из m пар документов.

Требуется построить набор правил $R^n(X^m) = \{r_1, \dots, r_n\} \subset R$ коррекции документов, который бы обладал наилучшими оценками полноты и точности [24]. Методика построения оценок качества описана в разделе 4.1.7. Для этого используются наборы документов, обработанных корректорами. В разных издательствах допустимы разные требования к оформлению документов. Поэтому искомые наборы правил могут различаться в зависимости от используемых данных.

Такая постановка задачи позволяет использовать её решение в практических задачах для автоматизации процесса корректуры научных текстов, написанных в формате \LaTeX .

1.2. Обзор существующих методов

Стоит отметить, что исследований непосредственно рассматриваемой задачи обнаружено не было. Тем не менее, существуют близкие с разных точек зрения области (анализ и автоматическая обработка текстов, системы автоматического перевода, основанные на анализе параллельных тестов, аудит программного кода, массовая обработка документов), для которых известны готовые решения.

1.2.1. Автоматизация обработки текстов

Регулярные выражения. Одним из наиболее известных и гибких подходов к контекстному анализу текстов является применение регулярных выражений [25, 26]. Это система обработки текста, которая использует специальную нотацию для обозначения искомых образцов (шаблонов). Регулярные выражения реализуются множеством утилит (`sed`, `grep` и т. п.) и популярны для приме-

нения в текстовых редакторах при поиске и изменении текста по указанному шаблону. Многие языки программирования имеют встроенный в синтаксис механизм обработки регулярных выражений.

При составлении шаблонов используется специальный синтаксис, поддерживающий, обычно, следующие операции:

- перечисление: вертикальная черта разделяет допустимые варианты;
- группировка: круглые скобки используются для определения области действия и приоритета операторов;
- квантификация: квантификатор после символа или группы определяет, сколько раз предшествующее выражение может встречаться (заданное число раз, интервал значений, верхние и нижние ограничения, 0 или 1 раз, любое число раз, хотя бы 1 раз).

Но регулярные выражения обладают рядом недостатков, которые являются критичными для рассматриваемой задачи: нет возможности анализировать структуры скобок произвольной глубины вложенности, нет возможности группировать правила, основываясь на свойствах команд \LaTeX , нет способа задать область действия.

Визуальная коррекция. Существуют инструменты для облегчения процесса ручной корректуры документов (в том числе, для создания которых используется \LaTeX). Например \Amaya+PEN [7] — система для редактирования и коррекции электронных документов с помощью графического интерфейса: пользователь использует указатель мыши или стилус, с помощью которого корректор определяют то, как нужно изменить текст (перечеркивание слов для их удаления, подчеркивание для курсива и т. п.). Нет обучения жестам, но есть набор распознаваемых действий, который не сложно выучить. Таким образом,

естественные жесты позволяют удобным образом вносить правки в документы.

Эта система основана на распознавании символов текста [27] и изучении эргономики жестов редактора для структурированных документов [28, 29], распознавании жестов в контексте корректуры и моделировании команд правки. Интерфейс управления встраивается в Амауа [30], интерактивный редактор Web-страниц.

Но Амауа+PEN не предусматривает никакой автоматизации процесса корректуры, и обработка одной страницы все ещё занимает много времени.

Автоматизация поиска ошибок в текстах. Вообще говоря, идея автоматизации коррекции текстов не нова [8, 31, 32], и на данный момент существуют качественные инструменты для автоматического поиска и исправления орфографических ошибок, использующие словари и морфологический анализ словоформ текста.

Примером такого инструмента является Lightproof [9]. Это средство проверки правописания. Оно поддерживает работу с различными средствами редактирования текста, например, OpenOffice.org и LibreOffice.

Lightproof обладает следующими особенностями:

- инструмент корректуры не зависит от языка,
- нативная поддержка OpenOffice.org,
- язык высокого уровня для определения правил, основанный на регулярных выражениях,
- дополнительные условия в правилах с помощью языка Python,
- интеграция с морфологическим анализатором Hunspell [33],

- написан на Python с помощью быстрого модуля регулярных выражений CPython,
- отложенная загрузка языковых модулей,
- шаблон для новых языков,
- дополнительные секции кода в файлах правил для добавления новых функций Python.

Но типографические ошибки нельзя формализовать с помощью словарей, и, кроме того, нет единого набора требований, применимого для всех издательств. А для Lightproof можно назначить только фиксированную, не развивающуюся, систему правил.

Ещё один продукт, позволяющий автоматизировать процесс поиска ошибок и различий в текстовых документах — это Docu-Proof компании Global Vision [34], который обладает широким набором функций, среди которых сравнение текстов и документов в форматах MS Word, PDF, XML и поиск ошибок в этих документах. Но эта система не предоставляет возможности внесения правок в исходные тексты документов формата L^AT_EX и, опять же, обладает фиксированным набором правил.

1.2.2. Интеллектуальная коррекция ошибок в поисковых запросах

Кроме того, схожая проблема возникает для интеллектуальной коррекции ошибок в запросах поиска [35–37]. В том числе, с помощью лексических и статистических признаков. Примером исследования в этой области является работа [10], в которой решалась проблема повышения эффективности автоматического исправления ошибок в поисковых запросах. В качестве целевого класса были выбраны словарные ошибки (пропуск/вставка/замена/перестановка букв в словах), составляющие две трети всех ошибок в запросах. Было показано, что

значительная часть ошибок являются тривиальными (исправление очевидно и однозначно), не зависят от словарного окружения и могут быть исправлены в автоматическом режиме.

Для принятия решения о возможности автоматического исправления использовался бинарный классификатор, разделяющий исправления на надежные, пригодные для автозамен, и ненадежные, пригодные только для подсказок. Поскольку тривиальные ошибки не зависят от контекста, для определения надежности исправлений достаточно признаков словарного (бесконтекстного) уровня, что значительно упростило задачу подбора признаков. В работе были использованы наиболее распространённые лексические и статистические признаки, применяемые при решении поисковых и лингвистических задач. Построенный на их базе классификатор показал приемлемое качество и возможность регулирования баланса полнота/точность. С помощью предложенного в работе метода можно с высокой точностью автоматически исправлять больше половины словарных опечаток, т. е. почти треть всех ошибок в поисковых запросах.

Но подобные подходы не применимы для исправления типографических ошибок, рассматриваемых в данной работе, которые связаны не только с текстовым содержанием документа, но и разметкой форматирования, и зачастую для описания ошибки не достаточно локальной информации в тексте, но также требуется знание контекста, дополнительной информации о позиции в логической структуре документа.

1.2.3. Автоматический перевод, основанный на анализе параллельных тестов

Задача автоматического изменения текстов возникает и при генерации переводов с одного естественного языка на другой. Рынок коммерческих про-

ектов этого профиля сложился в 1980-е, и в настоящее время множество компаний занимаются коммерческой разработкой систем машинного перевода [38–40]. Выделяют два основных типа таких систем: основанные на правилах (rule-based machine translation, RBMT) и статистические [11].

Системы статистического перевода используют методы машинного обучения и анализ массивов текстов, представленных одновременно на языке оригинала и языке перевода. Такие массивы данных называются параллельными корпусами текстов. На этапе обучения извлекаются статистические данные о переводе отдельных слов и фраз с исходного языка на язык перевода. В процессе перевода вычисляется наиболее вероятный перевод для каждого предложения.

Подход к задаче, исследуемой в данной диссертации, схож с методами систем статистического перевода: исследуются корпуса параллельных текстов, образованные документами формата \LaTeX до внесения корректорских правок и после. Но существенными отличительными особенностями рассматриваемой проблемы являются: сильная схожесть сравниваемых текстов, исходное отсутствие сопоставления элементов текстов, большое число типовых изменений, необходимость сохранения корректности синтаксиса \LaTeX , зависимость типа исправления от контекста.

1.2.4. Обработка исходного кода программ

С другой стороны, существует обширная область исследований, посвящённая улучшению характеристик исходного кода программ (вероятности возникновения ошибок в отдельных модулях, степени связности модулей и др.).

Методы классификации изменений исходного кода разделяются на следующие группы [41, 42]:

- неформальные (эвристические) методы, такие как метод поиска харак-

терных слов в комментариях к изменениям [43, 44] и метод поиска и классификации рефакторингов на основе значений определённых метрик [45];

- методы анализа синтаксиса изменений, такие как метод сравнения синтаксических деревьев версий кода [46], метод анализа синтаксической разницы версий кода с помощью встраиваемых в код тегов [47] и метод, основанный на реализации системы контроля версий, которая хранит абстрактные синтаксические деревья кода, полученные на основе данных из среды разработки [48];
- методы, основанные на data mining [49–52], такие как метод классификации изменений по признаку возможного наличия в них ошибки [53].

Известны методы [12, 13], позволяющие оценивать характеристики, основываясь на анализе истории изменений репозитория, и использовать их для поиска ошибок в коде. Они позволяют создавать рекомендательные системы [14] для улучшения качества кода программы при редактировании. Документы в формате \LaTeX можно рассматривать как исходный код, который используется компилятором \TeX , но в издательской практике не распространено использование репозитория, пригодных для последующего анализа, нет единых стандартов, и, кроме того, текстовое содержимое документов не может быть подвержено подобной обработке.

SSW Code Auditor — это система для аудита программного кода. Подобные системы решают одну из возникающих в ходе данной работы задач — стоят логические структуры, соответствующие текстовым описаниям, и анализируют их. Но цель такого анализа — выявить недостатки и уязвимости в результатах работы программистов, основываясь на закреплённой заранее известной системе правил. Такой формализованной системы правил нет для задачи, рассматриваемой в данной диссертации.

1.3. Предлагаемый подход

В данной работе предлагается разделить исследуемую задачу на три последовательных шага:

1. Для обработки документов формата \LaTeX проанализировать их структуру и построить конструкцию, которую было бы удобно обрабатывать с целью поиска изменений и выделения закономерностей. Такой конструкцией является синтаксическое дерево.
2. Рассмотреть множество документов, обработанных корректорами. Выделить различия, которые были внесены, в виде множества отображений вершин синтаксических деревьев.
3. Для всех изменённых, удалённых и добавленных вершин найти закономерности, основываясь на контексте — вершинах синтаксического дерева, находящихся рядом с изменённой вершиной.

1.3.1. Анализ структуры документов формата \LaTeX

Файлы формата \LaTeX , используемые при подготовке научных изданий (книг и сборников трудов), как правило, обладают естественной древовидной структурой (синтаксическим деревом, подробное описание в разделе 2.3), исследуя которую, можно получить всю необходимую информацию для описания корректорской правки.

Узлы этой структуры будем называть токенами. Выделяются следующие типы токенов: тело окружения \LaTeX , команда \LaTeX , окружение \LaTeX , метка, линейный размер, число, разделитель абзацев, путь к файлу, пробел, символ, параметры таблицы, слово, не распознаваемая последовательность символов (например, для окружения `verbatim`).

Синтаксическое дерево взаимно однозначно (с точностью компилятора \TeX) определяет документ \LaTeX . Правила коррекции удобно формулировать именно для деревьев.

Подробное формальное описание структуры документов формата \LaTeX находится в главе 1.

1.3.2. Выделение различий, вносимых корректорами в документы

Рассматривается пара документов «черновик–чистовик».

Документы можно сравнить как тексты — конечные последовательности символов [54]. Но этот подход не позволяет учитывать логическую структуру документов и разметку форматирования. С другой стороны, для каждого документа можно построить синтаксическое дерево, а затем найти отображение вершин деревьев [55]. Но на практике оказывается, что такой подход не эффективен по времени и требует чрезмерно много ресурсов памяти. Поэтому предлагается гибридный алгоритм сравнения документов, который учитывает логическую структуру документов, но при этом для сравнения используется, в том числе, сравнение текстовых представлений файлов формата \LaTeX .

Как результат сравнения получается отображение вершин синтаксического дерева черновика в синтаксическое дерево чистовика, по которому можно определить какие вершины были удалены, какие добавлены, какие изменены.

Алгоритмы, используемые для выделения различий документов, описаны в главе 2.

1.3.3. Синтез правил коррекции

Рассматривается выборка пар документов «черновик–чистовик».

Для каждой пары выделяются различия в виде отображения вершин синтаксических деревьев. Исследуются позиции изменённых, удалённых и до-

бавленных вершин. На основе анализа их контекста выделяются шаблоны, описывающие ближайшие токены.

Каждое правило коррекции характеризуется шаблоном, локализатором (токеном, к потомкам которого применяется шаблон) и действием (операцией, которая производится с токенами). Правила обладающие близкими позициями применимости исследуются на возможность группировки.

На основе статистики применимости правил к документам строятся оценки точности отдельных правил коррекции и точности и полноты всего набора построенных правил.

Об алгоритмах и принципах построения правил коррекции и методике построения оценок качества рассказывается в главе 3.

1.4. Выводы главы 1

1. В работе рассматривается задача синтеза формальных правил для автоматизации коррекции документов в формате \LaTeX . Под коррекцией подразумевается поиск мест типографических ошибок в тексте документа и предложение вариантов исправления. Методы, описанные в работе, рассматриваются на примере документов, каждый из которых соответствует статье из сборника трудов конференции.
2. В данной работе впервые ставится задача автоматического исправления типографических ошибок как задача обучения по прецедентам.
3. В литературе описываются исследования, смежные с задачами, поставленными в данной диссертации:
 - обработка текстовых документов (включая регулярные выражения, визуальное редактирование, автоматический поиск орфографических ошибок);

- интеллектуальная коррекция ошибок в запросах поиска;
- автоматический перевод, основанный на анализе параллельных текстов;
- обработка исходного кода программ.

4. Следующие особенности исследуемой задачи не позволяют использовать разработанные ранее методики:

- нет единых четко сформулированных стандартов к оформлению документов;
- требуется использовать гибкий набор правил;
- для определения возможности применимости конкретного правила к выбранной позиции требуется учитывать контекст;
- требуется работа непосредственно с исходным кодом документов формата \LaTeX , а не с результатом компиляции;
- не распространено использование репозиторий, позволяющих собирать статистику;
- нет методик проверки качества кода.

5. Предлагаемый в диссертации подход состоит из следующих последовательных шагов:

- построение синтаксического дерева рассматриваемых документов формата \LaTeX ,
- выделение различий, вносимых корректорами в документы,
- синтез правил коррекции на основе обучающей выборки, составленной из пар документов «черновик–чистовик».

Глава 2

Структура документов формата \LaTeX

Источником данных для данной работы являются документы в формате \LaTeX . В этой главе рассказывается о синтаксисе описания документов и логических блоках, из которых они строятся.

2.1. Система компьютерной верстки \LaTeX

\LaTeX является расширением (набором макросов, классов и стилей) для издательской системы \TeX . Поэтому описание начинается с \TeX а.

2.1.1. Кратко о \TeX е

\TeX представляет собой систему правил разметки текста и одновременно их обработчик — компилятор. Он был разработан американским математиком и программистом Дональдом Кнудом для верстки технических текстов, содержащих формулы [56]. \TeX удобен для работы с математическими текстами, поэтому многие научные издательства и журналы его используют.

Большинство текстовых процессоров и систем компьютерной верстки построены по принципу WYSIWYG (What You See Is What You Get — автор сразу видит результат). При создании документа \TeX , напротив, необходимо в исходном коде файла задать текст и разметку и выбрать шаблон, на основе которых компилятор автоматически формирует документ. \TeX предоставляет не только язык разметки, но и возможность определения дополнительных команд и сложного программирования, но для данной работы важен только язык разметки.

\TeX до сих пор не превзойден в качестве и гибкости верстки абзацев и

математических формул. Кроме того, автор, использующий $\text{T}_\text{E}\text{X}$, может подготовить документ на своем компьютере, переслать его в издательство, и быть уверенным, что там его текст будет правильно обработан и на печати будет отображаться в точности то же, как его компьютере. Благодаря этому свойству переносимости $\text{T}_\text{E}\text{X}$ стал очень популярен как язык международного обмена статьями по математике, физике и другим техническим тематикам.

На практике для подготовки изданий с помощью $\text{T}_\text{E}\text{X}$ используются наборы команд и расширений, которые стандартизируют подход к верстке и упрощают этот процесс. Наиболее распространённые расширения (наборы шаблонов, стилей и т. п.): $\text{L}_\text{A}\text{T}_\text{E}\text{X}$, $\text{AMS-T}_\text{E}\text{X}$, $\text{Plain T}_\text{E}\text{X}$. Важно заметить, что ни один из пакетов расширений для $\text{T}_\text{E}\text{X}$ не добавляет новых возможностей (все, что можно сделать в $\text{L}_\text{A}\text{T}_\text{E}\text{X}$, можно сделать и в $\text{Plain T}_\text{E}\text{X}$), но каждый пакет имеет свое назначение и соответствующие упрощения, которые позволяют избежать сложного дополнительного программирования.

2.1.2. Кратко о $\text{L}_\text{A}\text{T}_\text{E}\text{X}$

$\text{L}_\text{A}\text{T}_\text{E}\text{X}$ является наиболее распространенным расширением $\text{T}_\text{E}\text{X}$ [15]. Он, как издательская система, позволяет достичь книжного качества подготавливаемого материала. Например, указав общий стиль текста, автор может не вникать в детали оформления. С другой стороны, при необходимости, их нетрудно изменить (например, изменить шрифт, которым печатаются заголовки, можно не меняя все заголовки, а заменив настройку в стилевом файле). Кроме того, используя пакеты расширения $\text{L}_\text{A}\text{T}_\text{E}\text{X}$, можно быстро (правкой одной строки в исходном коде документа) переключить оформление так, чтобы большая статья стала отображаться как книга, можно вставлять оглавление одной командой, при этом оглавление будет сформировано автоматически. Также автоматически происходит нумерация разделов, теорем, рисунков, таблиц.

Вообще говоря, пакет \LaTeX позволяет автоматизировать многие задачи, связанные с набором текста и подготовкой статей, в том числе набор текста на нескольких языках, нумерацию разделов и формул, перекрёстные ссылки, верстку иллюстраций и таблиц, формирование библиографии и другое. Кроме базового набора существует множество расширений \LaTeX . Первая версия пакета была выпущена Лесли Лампортом в 1984-м году, актуальная версия, $\LaTeX 2\epsilon$, — в 1994-м.

Общий внешний вид скомпилированного документа в \LaTeX определяется стилевым файлом. Есть несколько стандартных стилевых файлов для статей, книг, писем и т. д. Кроме того, многие издательства и журналы предоставляют свои собственные стилевые файлы, что позволяет быстро оформить публикацию, соответствующую стандартам издания.

Существует несколько дистрибутивов программного обеспечения, которое предоставляет возможности \LaTeX для разных операционных систем. Наиболее распространенные и известные следующие:

- \TeX Live (Linux и Windows) [57];
- $\text{Mi}\mathcal{K}\TeX$ (Windows) [58];
- $\text{Mac}\mathcal{T}\TeX$ (MacOS) [59].

2.2. Структура разметки документа формата \LaTeX

Каждый документ \LaTeX должен начинаться с команды, в которой указывается используемый класс — шаблон оформления:

```
\documentclass[<необязательные параметры>]{<имя класса>}
```

Далее могут идти команды для подключения дополнительных файлов со стилями, выбора настроек и т. п. Но весь текст документа с разметкой формати-

рования заключен в окружение `document`:

```
\begin{document}
    ...
\end{document}
```

В данной диссертации исследуется содержимое именно этого окружения и только его. Далее описывается подход к формализации структуры документа формата \LaTeX , рассматриваемый в этой работе.

В исходном коде \LaTeX предусмотрены комментарии — это части строк, ограниченные символом `%` слева и концом строки справа. В данной работе комментарии не рассматриваются (они убираются в процессе обработки документа).

Считается, что непрерывная последовательность пробелов, символов табуляций и не более одного переноса строки эквивалентна одному пробелу. Если в подобной последовательности есть более одного переноса строки, то считается, что вся последовательность эквивалентна двум переносам строки (пустой строке).

2.2.1. Состояния

Каждая позиция в тексте документа может определяться набором состояний, в работе выделяются следующие:

- математическая формула (в противном случае — обычный текст), применяется при верстке математических символов;
- список, применяется при создании списков элементов;
- изображение, применяется при декларативном описании изображений;
- таблица, применяется при описании таблиц;

- вертикальный режим (в противном случае — горизонтальный), применяется, например, при переходах между абзацами.

Допускается одновременное использование нескольких состояний.

2.2.2. Типы элементов

Весь исходный текст документа формата \LaTeX состоит из элементов трёх типов: символ, команда, окружение.

Символы. Символы являются минимальными элементами конструкции документа формата \LaTeX . Каждый символ описывается шаблоном — фрагментом текста, который соответствует символу в коде документа.

Некоторые символы могут обладать меткой конца, в этом случае считается, что символ ограничивает (или «включает в себя») другой код документа. Примером такого символа может служить пара знаков доллара ($\boxed{\$}$. . $\boxed{\$}$), которая включает в себя описание формулы.

Список символов, применяющихся для анализа используемых в данной работе документов \LaTeX , приведен в приложении А.

Команды. Команды представляют собой ещё одни элементы конструкции документов \LaTeX , которые могут использовать аргументы.

Каждая команда определяется именем и шаблоном параметров. Имя команды состоит из знака \backslash , непрерывной конечной последовательности латинских букв и может заканчиваться символом $*$. Шаблон параметров описывает сигнатуру размещения аргументов команды в исходном коде, которые всегда указываются сразу после имени. Например, команда $\backslash put(\#1)\#2$ состоит из имени $\backslash put$ и шаблона $(\#1)\#2$, соответствующего использованию двух аргументов, первый из которых должен быть указан внутри круглых скобок.

Список команд, применяющихся для анализа используемых в данной работе документов \LaTeX , приведен в приложении Б.

Окружения. Окружения позволяют логически и визуально сгруппировать фрагмент исходного кода документа.

Каждое окружение описывается именем, которое состоит из непрерывной конечной последовательности латинских букв и может заканчиваться символом *. В исходном коде окружения описываются с помощью вспомогательных команд `\begin` (начало окружения) и `\end` (конец окружения). Например, окружение `center` описывается с помощью кода

```
\begin{center}
```

код документа внутри окружения

```
\end{center}
```

что взаимно однозначно соответствует выполнению команд

```
\center
```

код документа внутри окружения

```
\endcenter
```

Список окружений, применяющихся для анализа используемых в данной работе документов \LaTeX , приведен в приложении В.

Символ, команда или окружение может обладать различной функциональностью в зависимости от состояния обработчика \TeX , кроме того, элемент может быть определен не для всех состояний.

2.2.3. Типы лексем

Каждый элемент исходного текста документа формата \LaTeX обладает типом лексемы — логическим и функциональным значением. Один и тот же элемент может обладать разными типами лексем для различных состояний обработчика \TeX . Также типом лексемы могут обладать параметры команд. Выделяются следующие типы:

- `binaryOperator` (бинарный математический оператор) — например, знаки арифметических операций ($\boxed{+}$, $\boxed{-}$ и т. п.);
- `brackets` (скобки) — синтаксическая конструкция для группировки элементов ($\boxed{\{}$ и $\boxed{\}}$);
- `cellbreak` (конец ячейки) — разделитель ячеек в таблице ($\boxed{\&}$);
- `char` (символ) — например, символ пунктуации (запятая, точка, двоеточие, апострофы и т. п.);
- `command` (команда) — указание компилятору \TeX для изменения внутренних переменных или состояния (например, команда `\setlength`);
- `digit` (цифра) — арабские цифры ($\boxed{0}$, $\boxed{1}$, $\boxed{2}$ и т. д.)
- `equation` (формула) — математическое выражение (например, окружение `equation`);
- `floatingBox` (плавающий бокс) — например, окружение `figure`;
- `hskip` (горизонтальный отступ) — например, команда `\thickspace`;
- `image` (изображение) — картинка, вставленная в текст (может быть скриптом или внешним ресурсом; например, команда `\includegraphics`);

- `index` (верхний или нижний индекс) — индексы в формулах ($\hat{\square}$ и \square_{-});
- `item` (элемент списка) — команды `\item`, `\bibitem` и т. п.;
- `label` (метка) — идентификатор для перекрестных ссылок (например, параметры команды `\cite` и `\ref`);
- `length` (линейное измерение) — например, параметры команд `\hskip` и `\hspace`;
- `letter` (буква слова) — кириллические и латинские буквы;
- `linebreak` (обрыв строки) — команда `\linebreak` или символ \square_{\backslash} ;
- `list` (список) — например, окружение `itemize`;
- `par` (новый абзац) — команда `\par` или пропущенная строка;
- `path` (путь к файлу или папке) — например параметры команд `\XYtext` и `\includegraphics`;
- `postOperator` (математический постоператор) — например, знак факториала;
- `preOperator` (математический преоператор) — например, команды `\max` и `\min`;
- `raw` (необрабатываемые данные) — текст, который распознается и обрабатывается как неразделимая последовательность символов (например, параметры команд `\special` и `\selectlanguage`);
- `space` (пробел) — пробел в тексте (например, символы пробела и табуляции);
- `table` (таблица) — например, окружение `align`;

- `tableParams` (параметры таблицы) — описание формата столбцов таблицы;
- `tag` (тэг) — сервисная команда, которая не влияет на текстовое содержание (например, `\tabcolsep`);
- `vskip` (вертикальный отступ) — например, команда `\vfill`;
- `wraper` (обертка) — обычно служит для изменения внешнего вида, получаемого при компиляции (например, команды `\bar` и `\widehat`).

2.2.4. Действия и операнды

Каждый элемент документа \LaTeX может изменять состояние обработчика \TeX . Формально это означает совершение одного из двух действий (начало или конец) над операндом. Рассматриваются следующие операнды:

- группа — конец группы означает возвращение к тому состоянию обработчика \TeX , которое было к моменту начала группы;
- список — начало списка добавляет к состоянию обработчика \TeX режим списка;
- математический режим — начало математического режима переводит состояние обработчика \TeX в режим формулы;
- картинка — начало картинки добавляет к состоянию обработчика \TeX режим декларативного описания изображения;
- таблица — начало картинки добавляет к состоянию обработчика \TeX режим описания таблицы;
- текст — начало текста переводит состояние обработчика \TeX в режим верстки текста (а не формулы);

- вертикальный режим — начало вертикального режима переводит в соответствующее состояние обработчик \TeX .

Одно или несколько действий могут совершаться при обработке шаблона или метки конца символа, имени или параметров команды. Каждое действие совершается в порядке следования соответствующего фрагмента текста в исходном коде документа.

2.3. Синтаксическое дерево документа \LaTeX

Для последующего исследования полезно каждому документу формата \LaTeX взаимно однозначно сопоставить структуру данных, которую было бы удобно обрабатывать. Оказывается, что сформулированный в разделе 2.2 подход к формальному описанию документа позволяет задавать его с помощью синтаксического дерева. А правила коррекции удобно формулировать именно для деревьев.

2.3.1. Элементы синтаксического дерева

Узлы синтаксического дерева будем называть токенами. Каждый токен описывается типом лексемы и состоянием, которым обладает обработчик \TeX непосредственно перед началом анализа фрагмента исходного кода документа, соответствующего токену.

Токены строятся для всех окружений, команд и их аргументов. Кроме того, для более полного логического структурирования некоторые символы документов объединяются в общий токен на основе значений типов лексем.

Ниже описаны используемые в работе типы токенов. В таблице 2.1 приведены примеры фрагментов кода, соответствующих токенам разных типов.

Токен окружения \LaTeX . Каждый токен окружения взаимно однозначно соответствует одному из окружений \LaTeX , используемых в документе. Тип лексемы токена совпадает с типом лексемы соответствующего окружения. Каждый

Таблица 2.1. Примеры токенов каждого типа

Тип токена	Пример кода, соответствующего токenu
Тело окружения \LaTeX	$\backslash\text{begin}\{\text{tabular}\}\{c c\}$ <div style="border: 1px solid black; padding: 2px; display: inline-block;">высота & 1,2 м</div> $\backslash\text{end}\{\text{tabular}\}$
Команда \LaTeX	$\backslash\text{includegraphics}$ <div style="border: 1px solid black; padding: 2px; display: inline-block;">[width=10cm]</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">{../figure.eps}</div>
Окружение \LaTeX	$\backslash\text{begin}\{\text{tabular}\}\{c c\}$ высота & 1,2 м $\backslash\text{end}\{\text{tabular}\}$
Метка	$\backslash\text{ref}\{\text{equation1}\}$
Линейный размер	$\backslash\text{textwidth}=\text{10cm}$
Число	высота $\text{1,2}\backslash\text{, м}$
Разделитель абзацев	Абзац □ Новый абзац
Путь к файлу	$\backslash\text{includegraphics}$ <div style="border: 1px solid black; padding: 2px; display: inline-block;">[width=10cm]</div> <div style="border: 1px solid black; padding: 2px; display: inline-block;">{../figure.eps}</div>
Пробел	высота □ 1,2 \, м
Символ	высота 1,2 $\backslash\text{, м}$
Параметры таблицы	$\backslash\text{begin}\{\text{tabular}\}\{c c\}$ высота & 1,2 м $\backslash\text{end}\{\text{tabular}\}$
Слово	<div style="border: 1px solid black; padding: 2px; display: inline-block;">высота</div> 1,2 \, м
Не распознаваемая последовательность символов	$\backslash\text{verb} $ <div style="border: 1px solid black; padding: 2px; display: inline-block;">сложный код</div> $ $

такой токен содержит ровно трёх потомков: токен команды начала окружения, токен тела окружения и токен команды конца окружения.

Токен тела окружения \LaTeX . Токен тела окружения служит для инкапсуляции всех токенов, соответствующих фрагменту исходного кода, ограниченному окружением \LaTeX . Не обладает типом лексемы.

Токен команды \LaTeX . Каждый токен команды взаимно однозначно соответствует одной из команд \LaTeX , используемых в документе. Тип лексемы токена совпадает с типом лексемы соответствующей команды. Количество потомков токена команды совпадает с количеством её аргументов, а каждый потомок является токеном параметра команды.

Токен параметра команды \LaTeX . Каждый токен параметра команды взаимно однозначно соответствует параметру одной из команд \LaTeX , используемых в документе (определяется самой командой и номером параметра в команде). Тип лексемы токена совпадает с типом лексемы параметра, если он указан. В каждом токене хранится информация о наличии фигурных скобок, ограничивающих исходный код параметра, и пробеле перед началом описания параметра в коде команды. Токен содержит потомков, соответствующих тексту, образующему исходный код параметра в команде.

Токен пробела. Терминальный токен, который соответствует одному из разделителей, аналогичных, с точки зрения компилятора \TeX , пробелу, между двумя другими токенами.

Токен разделителя абзацев. Терминальный токен, который соответствует одному из разделителей, аналогичных, с точки зрения компилятора \TeX , пустой строке, между двумя другими токенами.

Токен слова. Терминальный токен, который соответствует непрерывной последовательности символов с типом лексемы `letter` такой, что нет аналогичной последовательности, её содержащей. Обладает типом лексемы только тогда, когда используется как параметр команды, тип лексемы которого известен. В таком случае тип лексемы токена слова совпадает с типом лексемы параметра.

Токен числа. Терминальный токен, который соответствует непрерывной последовательности символов с типом лексемы `digit` такой, что нет аналогичной последовательности, её содержащей. Обладает типом лексемы только тогда, когда используется как параметр команды, тип лексемы которого известен. В таком случае тип лексемы токена слова совпадает с типом лексемы параметра.

Токен метки. Терминальный токен, который соответствует фрагменту исходного кода документа, используемому как параметр, для которого указан тип лексемы `label`, т. е. соответствует имени метки. Обладает типом лексемы `label`.

Токен пути. Терминальный токен, который соответствует фрагменту исходного кода документа, используемому как параметр, для которого указан тип лексемы `path`, т. е. соответствует адресу пути. Обладает типом лексемы `path`.

Токен линейного размера. Терминальный токен, соответствующий фрагменту исходного кода документа, используемому как параметр, для которого указан тип лексемы `length`, т. е. соответствует значению размера. Обладает типом лексемы `length`.

Токен параметров таблицы. Терминальный токен, который соответствует фрагменту исходного кода документа, используемому как параметр, для которого указан тип лексемы `tableParams`, т. е. соответствует описанию форматирования столбцов таблицы. Обладает типом лексемы `tableParams`.

Токен не распознаваемой последовательности символов. Терминальный токен, который соответствует фрагменту исходного кода документа, используемому как параметр, для которого указан тип лексемы `raw`. Обладает типом лексемы `raw`.

Токен символа \LaTeX . Токен, соответствующий одному из символов \LaTeX , которые не покрываются токенами других типов. Тип лексемы токена совпадает с типом лексемы соответствующего символа. Если символ не имеет метки конца, то токен терминальный. В противном случае токен содержит потомков, которые соответствуют фрагменту исходного кода документа, заключённому между позицией в тексте шаблона токена и позицией метки конца.

2.4. Выводы главы 2

1. В работе рассматривается содержимое исходного кода документа \LaTeX , заключённое в окружение `document`.
2. Поддерживаются изменения состояния обработчика \TeX , возникающие при анализе документа: математическая формула, список, изображение, таблица, вертикальный режим.
3. Считается, что исходный код документа состоит из элементов трёх типов: символы, команды, окружения. Символ описывается шаблоном и не обязательной меткой конца. Команда задается именем и шаблоном

параметров. Окружение определяется именем.

4. Каждый элемент обладает типом лексемы, соответствующим логическому смыслу элемента. Типы лексем одного и того же элемента могут различаться для различных состояний обработчика $\text{T}_{\text{E}}\text{X}$.
5. Состояние обработчика $\text{T}_{\text{E}}\text{X}$ может изменяться при обработке команд и их параметров, шаблонов и меток конца символов.
6. Можно построить синтаксическое дерево, которое соответствует формальному описанию элементов исходного кода документа формата $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.
7. Синтаксическое дерево состоит из токенов, каждый из которых описывается типом лексемы и состоянием обработчика $\text{T}_{\text{E}}\text{X}$. Используются следующие типы токенов: токен окружения $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, токен тела окружения $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, токен команды $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, токен параметра команды $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$, токен пробела, токен разделителя абзацев, токен слова, токен числа, токен метки, токен пути, токен линейного размера, токен параметров таблицы, токен не распознаваемой последовательности символов, токен символа $\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}$.

Глава 3

Алгоритмы сравнения документов формата \LaTeX

В работе [1] для построения различий между синтаксическими деревьями используется алгоритм, основанный на алгоритме Zhang-Shasha [55]. Однако практический опыт позволил выявить следующие недостатки его применения. Во-первых, возникают проблемы, связанные со скоростью работы: сложность алгоритма пропорциональна произведению количеств ключевых корней для чернового и чистового деревьев. В случае технических статей, состоящих из четырёх страниц (распространённый вариант для сборников трудов конференций), минимальное количество ключевых корней составляет около 4500. Это приводит к тому, что сравнение документов занимает до трех минут, и становится невозможным использовать его для редактирования в режиме «онлайн». Во-вторых, существуют проблемы связанные с потреблением памяти. Для работы алгоритма требуется хранить попарные расстояния между всеми поддеревьями чернового и чистового деревьев и соответствующими лесами. Это делает невозможным использование алгоритма для сравнения больших документов, соответствующих, например, главам книг.

С другой стороны, существуют алгоритмы сравнения текстовых файлов, избавленные от подобных недостатков [54]. Но в этом случае возникают проблемы с качеством: полученное различие не учитывает структуру документов, и, в итоге, не соответствует логике корректора и не позволяет выявлять верные закономерности.

В данной главе предлагается гибридный алгоритм сравнения документов формата \LaTeX , использующий достоинства алгоритмов сравнения текстов, представляемых как конечные последовательности символов, и синтаксических деревьев, и позволяющий сравнительно быстро выявлять различия,

учитывающие логическую структуру, даже для больших документов.

3.1. Выделение различий между конечными последовательностями

Мера различия (редактирующее расстояние) между конечными последовательностями элементов, включая последовательности символов, которыми являются тексты, задается расстоянием Левенштейна [60, 61].

Определение 1. Пусть для изменения последовательности элементов решается применять операции трех типов: удаление элемента, вставка элемента, изменение элемента. Тогда расстоянием Левенштейна между двумя последовательностями называется минимальное количество таких операций.

Вычисление расстояния Расстояние Левенштейна выражается следующими рекуррентными соотношениями:

$$\begin{aligned} \delta_L(a_1 \dots a_n a_{n+1}, b_1 \dots b_m b_{m+1}) = \\ = \min \begin{cases} \delta_L(a_1 \dots a_n, b_1 \dots b_m b_{m+1}) + \delta(a_{n+1}, \emptyset) \\ \delta_L(a_1 \dots a_n a_{n+1}, b_1 \dots b_m) + \delta(\emptyset, b_{m+1}) \\ \delta_L(a_1 \dots a_n, b_1 \dots b_m) + \delta(a_{n+1}, b_{m+1}) \end{cases} , \end{aligned}$$

где в классическом случае $\delta(a_{n+1}, \emptyset)$ (цена удаления элемента a_{n+1}), $\delta(\emptyset, b_{m+1})$ (цена вставки элемента b_{m+1}) и $\delta(a_{n+1}, b_{m+1})$ (цена изменения элемента a_{n+1} на b_{m+1} при $a_{n+1} \neq b_{m+1}$) приравниваются к 1. Но, вообще говоря, это могут быть другие неотрицательные числа, описывающие степень различия элементов.

Кроме того, в этой работе мы будем использовать относительное расстояние Левенштейна — классическое расстояние Левенштейна, разделённое

на длину наибольшей последовательности:

$$\delta_{RL}(a_1 \dots a_n, b_1 \dots b_m) = \frac{\delta_L(a_1 \dots a_n, b_1 \dots b_m)}{\max(n, m)},$$

которое, очевидно, может принимать значения от 0 до 1.

3.1.1. Построение отображения

Алгоритмы, которые строят отображение, основанное на расстоянии Левенштейна, используют анализ обратных переходов для рекуррентных формул, описанных выше.

Алгоритм Вагнера–Фишера. Наивный подход заключается в выполнении двух шагов:

1. построение матрицы расстояний между всевозможными парами префиксов методами динамического программирования [62],
2. восстановление редакционного предписания с помощью пошагового анализа возможных обратных переходов.

Так, например, устроены алгоритмы Вагнера–Фишера [63] и Нидлмана–Вунша [64].

Пример 2. Рассмотрим применение вычисления редактирующего расстояния и отображения с помощью алгоритма Вагнера–Фишера в том случае, когда из слова «ЛОГИКА» получается слово «АЛГЕБРА».

На рисунке 3.1 (а) представлена матрица L , построенная динамически по рекуррентным формулам. В каждой её клетке находится значение редактирующего расстояния для соответствующих префиксов, в правой нижней клетке — редактирующее расстояние для сравниваемых слов.

	А	Л	Г	Е	Б	Р	А
Л	1	1	2	3	4	5	6
О	2	2	2	3	4	5	6
Г	3	3	2	3	4	5	6
И	4	4	3	3	4	5	6
К	5	5	4	4	4	5	6
А	5	6	5	5	5	5	5

а

	А	Л	Г	Е	Б	Р	А
Л	1	1	2	3	4	5	6
О	2	2	2	3	4	5	6
Г	3	3	2	3	4	5	6
И	4	4	3	3	4	5	6
К	5	5	4	4	4	5	6
А	5	6	5	5	5	5	5

б

Рис. 3.1. Пример матрицы редактирующих расстояний для префиксов слов (а). Восстановление последовательности операций (б).

На рисунке 3.1 (б) отображена восстановленная последовательность шагов для получения итогового редактирующего расстояния. По направлению ломаной можно определить какая операция происходит на каждом шаге:

- вправо ($L(i, j) = L(i - 1, j) + 1$) — добавлен символ,
- вниз ($L(i, j) = L(i, j - 1) + 1$) — удален символ,
- по диагонали — символ изменен ($L(i, j) = L(i - 1, j - 1) + 1$) или не изменен ($L(i, j) = L(i - 1, j - 1)$).

Это позволяет построить отображение, представленное на рисунке 3.2 (D означает, что символ удален, I — добавлен, C — изменен).

		Л	О	Г	И	К	А	
А	Л			Г	Е	Б	Р	А
I			D		C	C	I	
1	0	1	0	1	1	1	1	0

Рис. 3.2. Отображение, восстановленное по матрице расстояний.

Подобный подход оказывается крайне требователен к памяти — необходимо хранить количество элементов, пропорциональное произведению длин сравниваемых последовательностей. Поэтому на практике он редко используется.

Алгоритм Хиршберга. Наиболее эффективным, с точки зрения количества потребляемой памяти, является алгоритм Хиршберга [54].

Пусть нужно построить отображение последовательности $a_1 \dots a_n$ в последовательность $b_1 \dots b_m$. Идея рекурсивного перехода алгоритма основывается на двух равенствах:

$$\delta_L(a_1 \dots a_n, b_1 \dots b_m) = \delta_L(a_n \dots a_1, b_m \dots b_1),$$

$$\begin{aligned} \delta_L(a_1 \dots a_i a_{i+1} \dots a_n, b_1 \dots b_m) = \\ = \min_{j \in \{1, \dots, m-1\}} (\delta_L(a_1 \dots a_i, b_1 \dots b_j) + \delta_L(a_{i+1} \dots a_n, b_{j+1} \dots b_m)). \end{aligned}$$

Рассмотрим всевозможные случаи.

1. $n = 0$ (первая последовательность пустая). Тогда все элементы второй последовательности считаются добавленными.
2. $m = 0$ (вторая последовательность пустая). Тогда все элементы первой последовательности считаются удалёнными.
3. $n = 1$ (первая последовательность состоит из одного элемента a_1). Если во второй последовательности есть элементы, равные a_1 , то первый из них считается образом a_1 , а остальные — добавленными. Если таких элементов нет, то b_1 считается образом a_1 , а остальные — добавленными.
4. $m = 1$ (вторая последовательность состоит из одного элемента b_1). Если в первой последовательности есть элементы, равные b_1 , то первый из них считается прообразом b_1 , а остальные — удалёнными. Если таких элементов нет, то a_1 считается прообразом b_1 , а остальные — удалёнными.

5. $n, m \geq 2$. Первая последовательность разбивается на две равные, по возможности, части $a_1 \dots a_i$ и $a_n \dots a_{i+1}$, где $i = \lfloor \frac{n}{2} \rfloor$. Вторая последовательность разбивается на две части $b_1 \dots b_{j^*}$ и $b_n \dots b_{j^*+1}$ так, чтобы минимизировать сумму:

$$j^* = \arg \min_{j \in \{1, \dots, m-1\}} (\delta_L(a_1 \dots a_i, b_1 \dots b_j) + \delta_L(a_n \dots a_{i+1}, b_m \dots b_{j+1})).$$

Строятся отображения $a_1 \dots a_i$ в $b_1 \dots b_{j^*}$ и $a_n \dots a_{i+1}$ в $b_m \dots b_{j^*+1}$

Вторые части последовательностей записываются в обратном порядке для возможности повторного использования рассчитанного расстояния Левенштейна их префиксов.

3.1.2. Применение для документов формата \LaTeX

Документы формата \LaTeX обычно рассматриваются как текстовые файлы, поэтому возможно использование алгоритма Хиршберга, который можно применять для сравнения произвольных текстов. Естественно представлять текст как линейную последовательность символов и использовать алгоритм для сравнения таких последовательностей. Но часто оказывается (и это применимо к документам формата \LaTeX), что тексты содержат очень большое количество символов, и последовательности получаются чрезмерно длинными, что приводит к завышенному расходу памяти и низкой эффективности. Поэтому на практике сравниваемые тексты разбиваются на неделимые фрагменты, обычно в местах переноса строк, и строится отображение последовательностей таких фрагментов.

Такой подход хорошо работает, например, для визуального выделения различий исходного кода программ, поскольку разбиение на строки довольно неплохо соответствует списку операций, образующих структуру программы, написанной с хорошо выдержанным стилем. Но для поиска закономерностей,

возникающих при редактировании документов в формате \LaTeX , подобный способ применим плохо, поскольку такие документы обладают менее выраженной древовидной структурой. Поэтому требуется, как минимум, использовать более грамотный способ разделения текста на фрагменты.

Пример 3. Утилиты на основе алгоритма diff [65, 66] и его вариаций являются наиболее распространёнными реализациями методов сравнения текстовых данных. На сайте «Diff Checker» [67] можно проверить этот алгоритм в режиме «онлайн». На рисунке 3.3 представлены некоторые результаты его работы для пар документов формата \LaTeX .

Можно сделать вывод, что, не смотря на то, что алгоритм успешно справляется с выделением изменённых слов в предложениях, синтаксические конструкции \LaTeX обрабатываются не корректно — в данном случае они считаются полностью изменёнными. Это приводит и к ошибкам при сравнении участков текста, не содержащих разметки форматирования.

В данной главе предлагается методика, использующая сравнения текстовых фрагментов и учитывающая древовидную структуру документов в формате \LaTeX .

3.2. Построение различий между деревьями

Рассматриваются деревья, обладающие следующими свойствами: каждая вершина содержит ключ (элемент из заранее определённого набора), выбрана вершина, которая является корнем дерева, вершины, имеющие общего родителя, упорядочены. К дереву разрешается последовательно применять следующие операции: удаление вершины (все её потомки переходят родителю), вставка новой вершины (обратная удалению операция), изменение ключа вершины.

<p>51 Each of the intensities, prepared for the separate traffic direction and for assessed day type is always created as the average value of all particular intensities in same day type, specified in the source file by date. In our model, as already mentioned, an example for the day type 'Friday' is presented.</p> <p>52</p> <p>53 Average value of the intensity represents colored scale. Scale is organized from red through yellow, orange, green and blue to violet (values 0 - 130). Values of intensities are represented as number of heavy trucks passed the gate in each previous 15 minutes.</p> <p>54</p> <pre> 55 \begin{figure}[h] 56 \includegraphics[width=0.47\textwidth]{derbek1.eps} 57 \caption{Scale for the traffic intensity of heavy trucks (over 12 tons)} 58 \label{fig:Derbek1} 59 \end{figure} </pre>	<p>61 The intensities, each for the separate traffic direction and for assessed day type is always created as an average value of all particular intensities in same day type and it is specified in the source file by date. An-example for the day type 'Friday' is presented in our model.</p> <p>62</p> <pre> 62 \REVIEWERNOTE{Please, rewrite and make it simple.} </pre> <p>63</p> <pre> 63 \begin{figure}[h] 64 \includegraphics[width=\linewidth]{derbek1.eps} 65 \vskip[-2ex] 66 \caption{Scale for the traffic intensity of heavy trucks (over 12 tons).} 67 \label{fig:Derbek1} 68 \end{figure} 69 \end{figure} 70 The colored scale shows the traffic intensity, see-fig.\, \ref{fig:Derbek1}. 71 \REVIEWERNOTE{The colored scale shows the traffic intensity, see-\ref{fig:Derbek1}?} 72 Scale is described by red through yellow, orange, green and blue to violet (values 0--130). \mbox{Values} of intensities are represented as number of heavy trucks passed the gate in each previous 15 minutes. To view the colored figures refer to the PDF version of the IIP conference proceedings. 73 \REVIEWERNOTE{Please, insert words like 'To view the colored figures refer to the PDF version of the IIP conference proceedings' since the printed version will be is black and white (grayscale).} </pre>
<p>90 Если $n \leq m \leq 2^{n\beta}$, $\beta < 1/2$, то при $n \rightarrow \infty$ для почти всех матриц S из M_{mn}^k справедливо</p> <pre> 91 \begin{align} 92 & 1) \mbox{ } B(L,0) \approx \sum_{r=1}^{r_2} B_r(L,0) \approx \sum_{r=1}^{r_2} C_n^{r_2} p_{r_2} \notag \\ 93 & 2) \mbox{ } \sum_{r \geq r_2} B_r(L,0) \approx S_{r_2}(L,0) \approx \sum_{r \geq r_2} B_r(L,0) \approx \notag \\ & \sum_{r \geq r_2} B_r(L,0) \approx \notag \\ & \sum_{r \geq r_2} C_n^{r_2} p_{r_2} \approx C_n^{r_2} (1 - \exp(-m^{2-r_2}))^{r_2} \notag \\ 94 & 3) \mbox{ } \text{длины почти всех покрытий из } B(L,0) \text{ принадлежат интервалу } [r_1, r_2]. \notag \\ 95 & \mbox{ } \text{ } \notag \\ 96 \end{align} </pre>	<p>145 Если $n \leq m \leq 2^{n\beta}$, $\beta < 1/2$, то при $n \rightarrow \infty$ для почти всех матриц S из M_{mn}^k справедливо</p> <pre> 146 \begin{enumerate*} 147 \item 148 \mbox{ } B(L,0) \approx \sum_{r=1}^{r_2} B_r(L,0) \approx \sum_{r=1}^{r_2} C_n^{r_2} p_{r_2}; 149 \item 150 \mbox{ } \sum_{r \geq r_2} B_r(L,0) \approx \sum_{r \geq r_2} B_r(L,0) \approx \sum_{r \geq r_2} C_n^{r_2} p_{r_2} \approx C_n^{r_2} (1 - \exp(-m^{2-r_2}))^{r_2}; 151 \item 152 \mbox{ } \text{длины почти всех покрытий из } B(L,0) \text{ принадлежат интервалу } [r_1, r_2]. 153 \end{enumerate*} </pre>
<p>161 Если $n \leq k \leq 2^{n\beta}$, $\beta < 1/2$, то при $n \rightarrow \infty$ для почти всех матриц S из M_{mn}^k справедливо</p> <pre> 162 \begin{align} 163 & 1) \mbox{ } B(L) \approx \sum_{r=1}^{r_2} B_r(L) \approx \sum_{r=1}^{r_2} C_n^{r_2} p_{r_2} \notag \\ 164 & 2) \sum_{r \geq r_2} B_r(L) \approx S_{r_2}(L) \approx \sum_{r \geq r_2} B_r(L) \approx \sum_{r \geq r_2} C_n^{r_2} p_{r_2} \approx C_n^{r_2} (1 - \exp(-m^{2-r_2}))^{r_2} \notag \\ 165 & 3) \mbox{ } \text{длины почти всех покрытий из } B(L) \text{ принадлежат интервалу } [r_1, r_2]. \notag \\ 166 & \mbox{ } \text{ } \notag \\ 167 \end{align} </pre>	<p>235 Если $n \leq k \leq 2^{n\beta}$, $\beta < 1/2$, то при $n \rightarrow \infty$ для почти всех матриц S из M_{mn}^k справедливо</p> <pre> 236 \begin{enumerate*} 237 \item 238 \mbox{ } B(L) \approx \sum_{r=1}^{r_2} B_r(L) \approx \sum_{r=1}^{r_2} C_n^{r_2} p_{r_2}; 239 \item 240 \mbox{ } \sum_{r \geq r_2} B_r(L) \approx S_{r_2}(L) \approx \sum_{r \geq r_2} B_r(L) \approx \sum_{r \geq r_2} C_n^{r_2} p_{r_2} \approx C_n^{r_2} (1 - \exp(-m^{2-r_2}))^{r_2}; 241 \item 242 \mbox{ } \text{длины почти всех покрытий из } B(L) \text{ принадлежат интервалу } [r_1, r_2]. 243 \end{enumerate*} </pre>
<p>74 где n — количество отсчетов фрагмента, S — выборочная оценка дисперсии фрагмента, $\chi_{p,n}^2$ соответствующие квантили распределения хи-квадрат.</p> <p>75</p> <p>76 Для расчета «расстояния» между законами распределения фрагментов будем использовать расстояния Вкатачария cite(bibMOKill):</p> <pre> 77 \{ 78 \rho = 79 \left\{ \frac{1}{2} \frac{D_1 - E_2}{D_1 + D_2} \right\} + 80 \left\{ \frac{1}{2} \ln \frac{D_1 + D_2}{\sqrt{D_1 D_2}} \right\} 81 \} 82 \} 83 \} 84 \} 85 \} 86 \} 87 \} 88 \} 89 \} 90 \} 91 \} 92 \} 93 \} </pre> <p>94 где E и D — выборочные оценки среднего и дисперсии для первого и второго фрагментов.</p> <p>95</p> <p>96 В данной работе предлагается следующий критерий однородности фрагмента:</p> <ul style="list-style-type: none"> 97 \item Доверительные интервалы дисперсий S всех подобластей, полностью лежащих внутри фрагмента, должны пересекаться между собой и пересекать доверительный интервал фрагмента. 98 \item Расстояния ρ между законами распределения всех подобластей, полностью лежащих внутри фрагмента, должны быть меньше порога. 99 \item Дисперсия фрагмента должна быть мала. 	<p>96 где n — количество отсчетов фрагмента, S — выборочная оценка дисперсии, $\chi_{p,n}^2$ соответствующие квантили распределения хи-квадрат.</p> <p>97</p> <p>98 Для расчета «расстояния» между законами распределения яркости на фрагментах будем использовать расстояния Вкатачария cite(bibPonce):</p> <pre> 99 \{ 100 \rho = 101 \left\{ \frac{1}{2} \frac{D_1 - E_2}{D_1 + D_2} \right\} + 102 \left\{ \frac{1}{2} \ln \frac{D_1 + D_2}{\sqrt{D_1 D_2}} \right\} 103 \} 104 \} 105 \} 106 \} 107 \} 108 \} 109 \} 110 \} 111 \} 112 \} 113 \} 114 \} 115 \} 116 \} 117 \} 118 \} </pre> <p>109 где E и D — выборочные оценки среднего и дисперсии для яркости на первом и втором фрагментах.</p> <p>110</p> <p>111 В данной работе предлагается следующий критерий однородности фрагмента:</p> <ul style="list-style-type: none"> 112 \item Доверительные интервалы дисперсий яркости-S для всех подобластей, полностью лежащих внутри фрагмента, должны пересекаться между собой и пересекать доверительный интервал дисперсии яркости для фрагмента; 113 \item расстояния ρ между законами распределения яркости для всех подобластей, полностью лежащих внутри фрагмента, должны быть меньше порога; 114 \item дисперсия яркости на фрагменте должна быть мала.

Рис. 3.3. Примеры работы алгоритма diff для документов в формате L^AT_EX.

Определение 2. Редактирующим расстоянием между двумя деревьями называется минимальное количество операций удаления, вставки и изменения ключа вершин, позволяющих получить из первого дерева второе.

3.2.1. Алгоритм Zhang–Shasha

Этот алгоритм позволяет вычислять редактирующее расстояние между двумя деревьями и, кроме того, определять, какую операцию нужно применить к каждой вершине для реализации такого расстояния [55].

Определение 3. В произвольном дереве каждой вершине можно сопоставить наиболее левую для нее вершину: каждой терминальной вершине — её саму, любой другой — наиболее левую для самого левого её потомка.

Пример 4. На рисунке 3.4 показан пример графа с выделенными наиболее левыми вершинами: все вершины с контуром одинакового цвета имеют общую наиболее левую вершину, которая закрашена в тот же цвет.

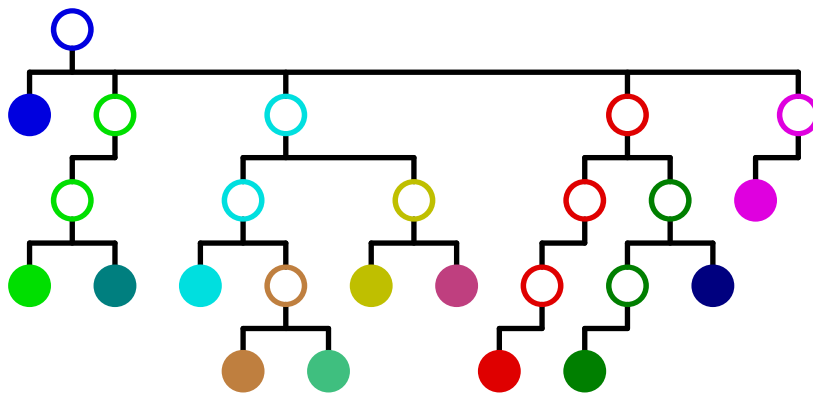


Рис. 3.4. Наиболее левые вершины дерева.

Определение 4. В произвольном дереве вершина, для которой наиболее левая вершина отличается от наиболее левой вершины для её родителя называется ключевым корнем.

Пример 5. На рисунке 3.5 показан пример графа с выделенными ключевыми корнями: все вершины с контуром одинакового цвета имеют общий

ключевой корень, который закрашен в тот же цвет.

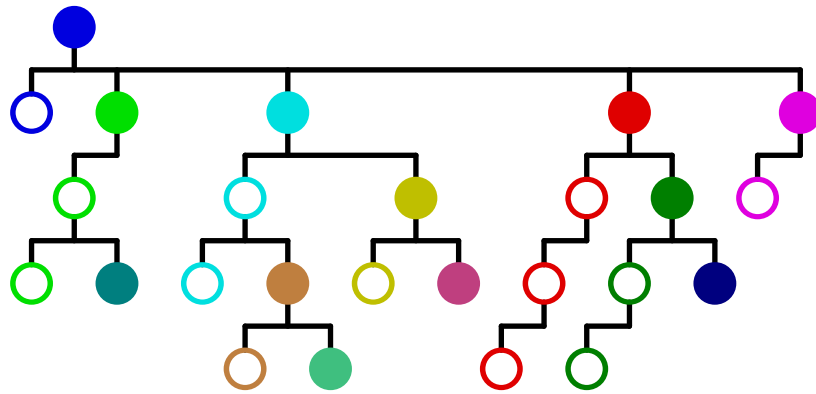


Рис. 3.5. Ключевые корни дерева.

Таким образом, любое дерево разбивается на линейные последовательности вершин, начинающиеся в ключевых корнях и заканчивающиеся в соответствующих наиболее левых вершинах. На рисунках 3.4 и 3.5 каждая такая последовательность образуется всеми вершинами с одинаковым цветом контура.

Определение 5. Пусть у дерева n вершин. Каждой взаимно однозначно сопоставляется номер от 1 до n так, чтобы для любого поддерева выполнялись следующие условия:

- корень поддерева имеет номер больший, чем все остальные вершины,
- для любых двух потомков корня все вершины поддерева, образованного более левым, имеют меньшие номера, чем вершины поддерева, образованного более правым.

Такой порядок нумерации называется обратным.

Пример 6. На рисунке 3.6 показан пример обратной нумерации дерева из шести вершин.

Оказывается, что поддерево, образованное произвольным ключевым корнем, состоит из всех вершин, номера которых не превосходят номера корня,

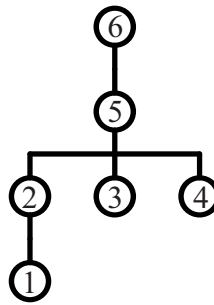


Рис. 3.6. Обратная нумерация вершин дерева.

и только из них. Далее каждая вершина дерева будет обозначаться числом, равным её номеру.

Определение 6. Пусть заданы два дерева, вершины которых упорядочены. Отображением первого дерева во второе называется правило, которое некоторым вершинам первого дерева взаимно однозначно сопоставляет некоторые вершины второго дерева так, чтобы порядок следования вершин сохранился.

Такие отображения принято записывать с помощью набора пар номеров вершин (прообраз, образ). Пусть отображение содержит пары (a, b) и (c, d) . Тогда требуемые условия запишутся следующим образом:

$$a = c \Leftrightarrow b = d, \quad a < c \Leftrightarrow b < d.$$

Каждое отображение может быть описано набором операций:

- если вершина первого дерева не имеет образа, то её нужно удалить;
- если вершина второго дерева не имеет прообраза, то её нужно вставить;
- если вершине первого дерева соответствует вершина второго с другим ключом, то нужно изменить ключ.

Таким образом, отображение, соответствующее минимальному количеству операций, реализует редактирующее расстояние.

Пример 7. На рисунке 3.7 показан пример отображения двух деревьев, соответствующего редактирующему расстоянию. Формально оно записывается следующим образом: $(1, 1)$, $(2, 2)$, $(3, 3)$, $(4, 5)$, $(6, 6)$.

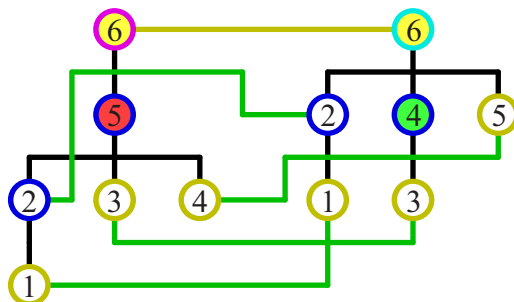


Рис. 3.7. Отображение деревьев.

Вычисление расстояний. В следующих формулах символы \circ и \bullet обозначают деревья с корнями \circ и \bullet соответственно, Δ , \blacktriangle — леса, образованные удалением корней этих деревьев, $\Delta\Delta$ и $\blacktriangle\blacktriangle$ — произвольные леса.

Расстояние между деревьями определяется рекуррентной формулой с помощью расстояния между лесами:

$$\delta(\Delta\Delta, \blacktriangle\blacktriangle) = \min \begin{cases} \delta(\Delta\Delta, \blacktriangle\blacktriangle) + 1; \\ \delta(\Delta\Delta, \blacktriangle\blacktriangle) + 1; \\ \delta(\Delta\Delta, \blacktriangle\blacktriangle) + \delta(\circ, \bullet); \end{cases}$$

где $\delta(\circ, \bullet)$ равно 1, если корни деревьев имеют разный ключ, и равно 0, если одинаковый.

Расстояние между лесами или деревом и лесом, в свою очередь, определяется рекуррентной формулой:

$$\delta(\Delta\Delta\Delta, \blacktriangle\blacktriangle\blacktriangle) = \min \begin{cases} \delta(\Delta\Delta\Delta, \blacktriangle\blacktriangle\blacktriangle) + 1; \\ \delta(\Delta\Delta\Delta, \blacktriangle\blacktriangle\blacktriangle) + 1; \\ \delta(\Delta\Delta, \blacktriangle\blacktriangle) + \delta(\Delta, \blacktriangle). \end{cases}$$

Обозначим: n_1 и n_2 — количества вершин, m_1 и m_2 — количества ключевых корней, $K^1 = \{k_1^1, \dots, k_{m_1}^1\}$ и $K^2 = \{k_1^2, \dots, k_{m_2}^2\}$ — упорядоченные по возрастанию наборы ключевых корней первого и второго деревьев соответственно.

Основной цикл алгоритма запишется следующим образом.

для $i = k_1^1, \dots, k_{m_1}^1$ **выполнять**

для $j = k_1^2, \dots, k_{m_2}^2$ **выполнять**

`treeDist(i, j);`

Здесь функция `treeDist(i, j)` вычисляет расстояние между поддеревьями первого и второго дерева с корнями i и j соответственно.

Построение отображений. Во время вычисления расстояний для ключевых корней заполняются две матрицы:

- матрица $(n_1 \times n_2)$ расстояний между деревьями, где в ячейке (i, j) , образованной пересечением i -й строки и j -го столбца, стоит расстояние между поддеревом первого дерева с корнем i и второго с корнем j ;
- матрица $(n_1 \times n_2)$ расстояний между лесами, где в ячейке (i, j) стоит расстояние между лесами, образованными удалением корней из соответствующих поддеревьев.

	1	2	3	4	5	6
1	0	1	0	1	0	5
2	1	0	1	0	1	4
3	0	1	0	1	0	5
4	0	1	0	1	0	5
5	4	3	4	3	4	2
6	5	4	5	4	5	3

а

	1	2	3	4	5	6
1	0	1	2	3	4	5
2	1	0	1	2	3	4
3	2	1	0	1	2	3
4	3	2	1	2	1	2
5	4	3	2	3	2	2
6	5	4	3	4	3	3

б

Рис. 3.8. Матрицы расстояний между деревьями (а) и лесами (б).

Пример 8. На рисунке 3.8 показаны матрицы расстояний для деревьев из примера 7.

Число в правом нижнем углу таблицы расстояний между деревьями равно редактирующему расстоянию. Для каждой ячейки двух таблиц можно вычислить, из каких других можно перейти в нее, согласно формулам расстояний. Другими словами, определить, какая операция производилась с соответствующей вершиной (не всегда однозначно, в таких случаях можно выбрать любую). Таким образом строится маршрут из правого нижнего угла таблицы расстояний между деревьями в левый верхний. Ячейки этой таблицы, которые попали в маршрут, зададут пары чисел, соответствующих отображению.

Итак, результат работы алгоритма: пары (прообраз и образ) не изменённых вершин, пары (прообраз и образ) изменённых вершин, множество удалённых вершин, множество добавленных вершин.

3.2.2. Применение для синтаксических деревьев \LaTeX

Токену каждого типа синтаксического дерева \LaTeX можно сопоставить ключ по правилу, описанному в таблице 3.1. После определения ключей синтаксические деревья полностью удовлетворяют условиям применимости алгоритма Zhang–Shasha.

Расстояние $\delta(\circ, \bullet) \in [0, 1]$ между парой токенов вычисляется по следующим правилам, которые основаны на практическом опыте:

1. 0, если совпадают;
2. 1, если имеют разный тип (например, слово и команда);
3. для команд: 1, если различаются именем; $\frac{1}{2}$, если различаются только сигнатурой параметров;

Таблица 3.1. Ключи для токенов каждого типа

Тип токена	Ключ	Пример токена	Пример ключа
Тело окружения \LaTeX	Тип окружения	$\begin{tabular}{c c}$ высота & 1,2м $\end{tabular}$	tabular body
Команда \LaTeX	Сигнатура команды	\includegraphics [width=10cm] {../figure.eps}	$\includegraphics[#1]#2$
Окружение \LaTeX	Сигнатура команд начала и конца	$\begin{tabular}{c c}$ высота & 1,2м $\end{tabular}$	\tabular \endtabular
Метка	Имя метки	$\ref{equation1}$	equation1
Линейный размер	Значение размера	$\text{width}=10\text{cm}$	10cm
Число	Значение числа	высота 1,2 \, м	1,2
Разделитель абзацев	Тип токена	Абзац \square Новый абзац	par
Путь к файлу	Значение пути	\includegraphics [width=10cm] {../figure.eps}	../figure.eps
Пробел	Тип токена	высота \square 1,2 \, м	space
Символ	Значение символа	высота 1,2 \, м	\,
Параметры таблицы	Значение параметров	$\begin{tabular}{c c}$ высота & 1,2м $\end{tabular}$	c c
Слово	Значение слова	высота 1,2 \, м	высота
Не распознаваемая последовательность символов	Код последовательности	$\verb сложный код $	сложный код

4. для окружений: 1, если различаются именем; полусумма расстояний между командами начала и конца, если имена совпадают;
5. для слов, имен меток, путей к файлам — относительное расстояние Левенштейна для соответствующих последовательностей символов;
6. для всех остальных: 1.

В своей работе [55] Zhang и Shasha ссылаются на более ранние исследования других авторов [68–74], приводя свой алгоритм как обладающий следующими характеристиками:

- лучшее время и сложность по сравнению с любыми аналогами в литературе,
- хорошее распараллеливание.

Но, тем не менее, следующие особенности алгоритма Zhang–Shasha мешают эффективно применять его для синтаксических деревьев документов \LaTeX :

- две матрицы расстояний для каждой пары вершин — требуемый объем памяти $O(n_1 \times n_2)$: не достаточно объема оперативной памяти персональных компьютеров для сравнения, например, глав книг;
- двойной цикл по ключевым корням — сложность алгоритма $O(m_1 \times m_2)$: синтаксические деревья документов формата \LaTeX имеют тысячи ключевых корней, поэтому скорость алгоритма невысокая.

Текущая глава нацелена на устранение подобных недостатков с помощью уменьшения количества вершин деревьев, которые необходимо сравнивать.

3.3. Гибридный алгоритм

Идея заключается в том, чтобы найти как можно больше совпадений и различий синтаксических деревьев, используя сравнение документов \LaTeX , как текстов, а деревья, составленные из оставшихся токенов, сравнить с помощью алгоритма Zhang–Shasha.

Для работы используются упрощённые текстовые представления документов формата \LaTeX , которые состоят из минимального количества символов, но не изменяют синтаксическое дерево. Это позволяет однозначно определить вид документа и уменьшить количество сравниваемых элементов.

3.3.1. Разбиение текста

В первую очередь построим линейные последовательности фрагментов текста сравниваемых документов.

Каждому токеноу синтаксического дерева соответствует набор последовательных символов в тексте документа. Поэтому можно говорить о границах токена: позициях начала (перед первым из этих символов) и конца (после последнего из символов). Эти позиции удобно использовать в качестве разделителей текста документа на фрагменты, поскольку они отражают логику структуры элементов \LaTeX .

Но если использовать границы всех токенов синтаксического дерева, получается слишком мелкое разбиение. Это приводит к тому, что возникает большое количество фрагментов (требуется много времени и памяти на сравнение их последовательностей), а каждый из фрагментов несет малую смысловую нагрузку.

Поэтому в качестве разделителей выбираются границы только тех токенов, которые имеют потомков. Если два разделителя имеют одинаковые позиции, то они рассматриваются как один.

3.3.2. Отображение фрагментов текста

Отображение фрагментов текста будем находить с помощью алгоритма Хиршберга. Но будем учитывать, что некоторые пары фрагментов могут иметь меньше различий, чем другие: в качестве меры изменения одного фрагмента текста на другой будем использовать относительное расстояние Левенштейна для последовательностей символов, образующих эти фрагменты.

3.3.3. Отображение символов

После построения отображения для каждого фрагмента текста сравниваемых документов возможны следующие случаи.

- Если фрагмент принадлежит первому документу и в качестве образа имеет пустую строку, то будем считать, что все его символы удаляются.
- Если фрагмент текста принадлежит второму документу и в качестве прообраза имеет пустую строку, то будем считать, что все его символы добавляются.

Все остальные фрагменты разбиваются на пары: прообраз и образ. Если прообраз и образ совпадают, то будем считать, что каждый их символ не изменяется. Для не совпадающих образа и прообраза построим отображение символов с помощью алгоритма Хиршберга, рассматривая два этих фрагмента текста как две линейные последовательности символов.

3.3.4. Отображение токенов

Теперь построим отображение синтаксических деревьев, определяя состояние каждого токена с помощью символов, которые ему соответствуют (лежат между его границами).

Определение 7. Если символ соответствует токену, но не соответствует ни одному из его потомков, то он называется персональным символом токена.

Все символы текста разбиваются на классы (некоторые могут быть пустыми), взаимно однозначно соответствующие токенам.

Будем считать, что токен первого дерева удаляется, если удаляются все символы текста первого документа, которые ему соответствуют. Это происходит в том случае, если удаляются все персональные символы и потомки токена.

Будем считать, что токен второго дерева добавляется, если добавляются все символы текста второго документа, которые ему соответствуют. Это происходит в том случае, если добавляются все персональные символы и потомки токена.

Будем считать, что токен первого или второго дерева не изменяется, если не изменяются все символы текста документа, которые ему соответствуют. Это происходит в том случае, если не изменяются все персональные символы и потомки токена, а образы (для первого дерева) или прообразы (для второго дерева) всех потомков имеют общего родителя.

Если из синтаксических деревьев сравниваемых документов убрать все удаляемые, добавляемые и неизменяемые токены, останутся два дерева, состоящие из остальных токенов. Для построения отображения этих токенов используется алгоритм Zhang–Shasha.

3.3.5. Структура алгоритма

Таким образом, предлагаемый алгоритм заключается в последовательном выполнении следующих шагов.

1. Построить упрощённые текстовые представления сравниваемых документов.

2. Разбить тексты на фрагменты, соответствующие границам токенов.
3. Построить отображение линейных последовательностей фрагментов текста.
4. Построить отображение символов текстов сравниваемых документов, основываясь на отображении фрагментов.
5. Выделить удалённые, добавленные и не изменённые токены.
6. Построить отображение деревьев, образованных остальными токенами.

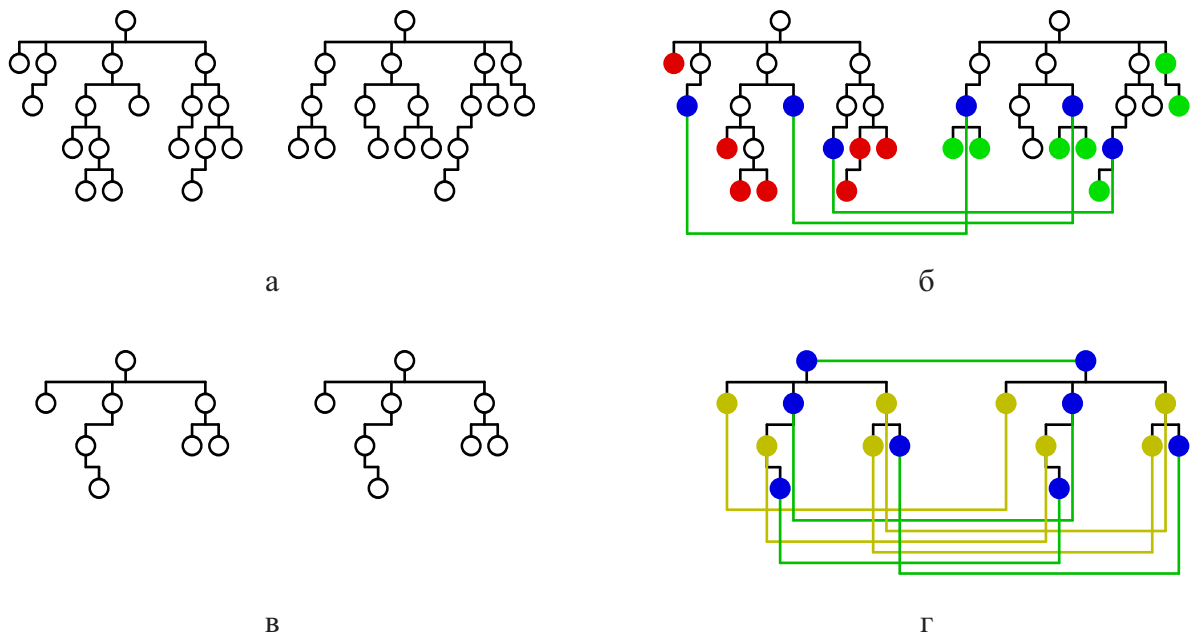


Рис. 3.9. Этапы построения отображения деревьев гибридным алгоритмом: исходные деревья (а), выделены удалённые, добавленные и не изменённые токены (б), оставлены только неизвестные токены (в), построено отображение для оставшихся токенов с помощью алгоритма Zhang–Shasha (г).

Пример 9. Рассмотрим этапы сравнения двух синтаксических деревьев, изображённых на рисунке 3.9 (а). Шагу 4 гибридного алгоритма соответствует состояние на рисунке 3.9 (б). После шага 5 остаются два дерева, изображённые на рисунке 3.9 (в). Результату их сравнения с помощью алгоритма

Zhang–Shasha, получаемому на шаге 6, соответствует отображение на рисунке 3.9 (г).

3.3.6. Эксперимент

Для исследования работы алгоритма использовались 85 пар черновиков и чистовиков, вошедших в сборник трудов восьмой конференции «Интеллектуализация обработки информации» [20]. Статистические характеристики документов приведены в таблице 3.2.

Таблица 3.2. Количественные характеристики статей, используемых в эксперименте.

	Символы	Токены	Ключевые корни
Минимум	3656	1368	1235
Максимум	30736	9939	8912
В среднем	18334,1	5620,46	5163

В таблице 3.3 показаны статистические данные деревьев, которые остаются после выделения удаляемых, добавляемых и не изменяемых токенов во время работы гибридного алгоритма. Можно заметить, что количество токенов и ключевых корней сократилось в разы. Это заметно сказалось на количестве потребляемой памяти. Тем не менее, сравнение документов стало быстрее только приблизительно в три раза, поскольку требуется время на построение отображения символов текстов документов.

Таблица 3.3. Количественные характеристики деревьев, получаемых во время работы гибридного алгоритма.

	Токены	Ключевые корни
Минимум	346	297
Максимум	1689	1325
В среднем	1011,6828	826,08

В таблице 3.4 приведены оценки точности и полноты предлагаемого гибридного алгоритма по отношению к результатам сравнения синтаксических деревьев алгоритмом Zhang–Shasha для всех пар рассматриваемых документов, которые рассчитывались следующим образом. Пусть $M = \{(a_i, b_i)\}$ и $M^* = \{(a_i^*, b_i^*)\}$ — наборы пар токенов, описывающие отображения пары документов, построенные соответственно с помощью гибридного алгоритма и алгоритма Zhang–Shasha. Тогда точность P и полнота R определяются формулами:

$$P = \frac{|M \cap M^*|}{|M|}, \quad R = \frac{|M \cap M^*|}{|M^*|}.$$

Таблица 3.4. Точность и полнота гибридного алгоритма.

	Точность	Полнота
Минимум	0,75	0,78
Максимум	1	1
В среднем	0,91	0,91

3.4. Выводы главы 3

1. Предложен алгоритм, который позволяет сравнительно быстро выявлять различия документов в формате L^AT_EX, не требуя большого объема памяти.
2. Проведен эксперимент для сравнения полученного алгоритма с алгоритмом Zhang–Shasha.
3. Из результатов эксперимента можно видеть, что такой подход позволяет заметно уменьшить требования к памяти и времени работы.
4. Получаемые этим способом отображения не полностью соответствуют результатам алгоритма, сравнивающего только синтаксические деревья.

Но это никак не мешает возможности применения алгоритма на практике.

5. Рассмотренный подход может быть применим в других задачах, связанных с текстовыми документами, обладающими синтаксическим деревом, поскольку используемые алгоритмы не связаны с особенностями формата документов \LaTeX .

Глава 4

Правила коррекции

Результаты сравнения синтаксических деревьев документов в формате L^AT_EX используются для синтеза правил коррекции. Документы из обучающей выборки сравниваются попарно. Наборы правил строятся для каждой пары, потом построенные множества объединяются.

В данной главе:

- рассматриваются основы построения правил на примере правил с линейными шаблонами;
- предлагается методика построения оценок точности и полноты набора построенных правил;
- проводится эксперимент на корпусе статей конференции ИОИ-8 [20];
- рассматриваются два подхода к улучшению качества правил: группировка и использование древовидных шаблонов.

4.1. Правила коррекции с простой структурой

После получения отображения черновых и чистовых деревьев строится начальный набор правил коррекции [1]. Каждое построенное правило характеризуется шаблоном (последовательностью соседних токенов с общим родителем), локализатором (токеном, к потомкам которого применяется шаблон) и действием (операцией, направленной на изменение синтаксического дерева). Таким образом, правило позволяет локализовать ошибку (определить её положение в синтаксическом дереве, а, следовательно, и в исходном тексте

документа; при этом считается, что ошибку содержит фрагмент текста, соответствующий токенам шаблона правила) и предложить вариант исправления (изменение синтаксического дерева, а, следовательно, и исходного теста документа) на основе действия правила.

4.1.1. Синтез правил с линейным шаблоном

Определение 8. Левая шаблонная цепочка радиуса r — это последовательность соседних токенов с общим родителем длиной не больше r . Началом цепочки считается самый правый её токен.

Правая шаблонная цепочка радиуса r — это последовательность соседних токенов с общим родителем длиной не больше r . Началом цепочки считается самый левый её токен.

- Пусть токен x чернового дерева удалён или изменён на токен y . Тогда локализатор — родительский токен x , шаблон составляется из левой и правой шаблонных цепочек, наиболее близких к x , и самого токена x . В таких случаях токен x будем называть целевым токеном правила. Действие правила заключается в удалении целевого токена или изменении его на токен y , в зависимости от типа правила.
- Пусть в чистовое дерево добавлен токен y . Тогда локализатор — прообраз родительского токена y , если он существует; шаблон составляется из левой шаблонной цепочки, начинающейся в прообразе левого соседа y , если он существует, и аналогичной правой. Действие правила заключается в добавлении токена y между левой и правой шаблонными цепочками.

Пример 10 (Правило добавления токена). Часто для улучшения читаемости текста между стоящими рядом формулами вставляется дополнительный

пробел. На рисунке 4.1 показан пример отображения синтаксических деревьев, которое возникает при изменении фрагмента текста Если \$\$\$, \$\$\$, то на Если \$\$\$,\; \$\$\$, то.

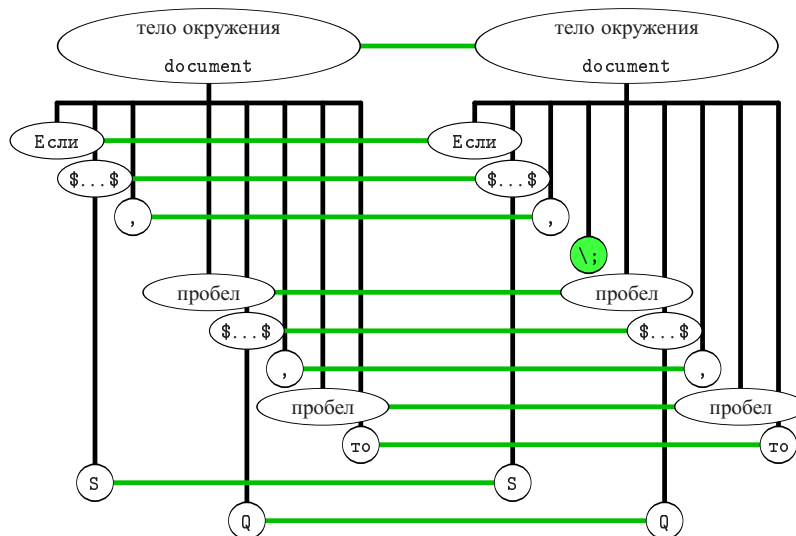


Рис. 4.1. Пример правила добавления токена.

Если \$\$\$, \$\$\$, то → Если \$\$\$,\; \$\$\$, то

Смысл изменения заключается в добавлении символа \; между запятой и пробелом. Но между запятой и пробелом дополнительное пространство нужно добавлять не всегда. Даже если перед запятой формула, или формула после пробела. Правильно условие — это наличие формул и перед запятой, и после пробела.

Поэтому в рассматриваемом случае левая шаблонная цепочка будет состоять из токена запятой и токена формулы, правая — из токена пробела и токена формулы, шаблон — из токена формулы, токена запятой, токена пробела и токена формулы. Действие правила заключается в добавлении токена символа \; между токенами запятой и пробела.

Пример 11 (Правила изменения токена). Одной из самых распространённых типографических ошибок является использование кавычек "... " вместо «...» в русском тексте. Пусть соответствующие фрагменты чернового и чистового синтаксических деревьев выглядят так, как показано на рисунке 4.2,

что соответствует преобразованию фрагмента `''слово` в `<<слово` внутри окружения `document`.

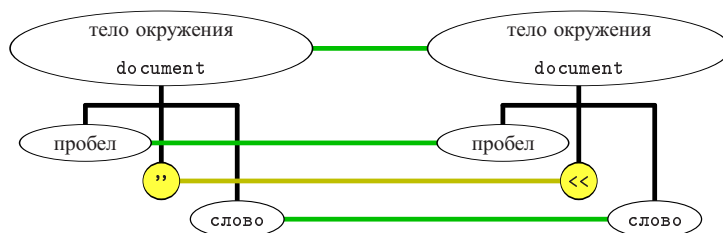


Рис. 4.2. Пример правила изменения токена.

`''слово` → `<<слово`

Тогда локализатором правила является токен тела окружения `document`, левая шаблонная цепочка состоит из токена пробела, правая — из токена слова, шаблон — из токена пробела, токена символа `''` и токена слова. Действие правила заключается в изменении токена, находящегося между токенами пробела и слова на токен символа `<<`.

Аналогичное правило строится для изменения правой кавычки.

С другой стороны, некоторые правила применимы не для всех фрагментов документа. Это подтверждает необходимость учитывать контекст при создании и использовании правил. Например, в списке литературы количество страниц для англоязычных источников обычно принято обозначать с помощью «Рр». Но на практике часто встречается обозначение с помощью одной буквы «Р». Таким образом требуется делать замену «Р» на «Рр», но такая замена в произвольном участке документа, скорее всего, окажется некорректной. Поэтому правило строится только для тела окружения `thebibliography`.

На рисунке 4.3 показано отображение синтаксических деревьев, которое возникает при изменении фрагмента текста `, Р` на `, Рр`.

В этом случае левая шаблонная цепочка состоит из токена запятой и токена пробела, правая — пустая, шаблон — из токена запятой, токена пробела и токена слова «Р». Действие правила заключается в изменении последнего

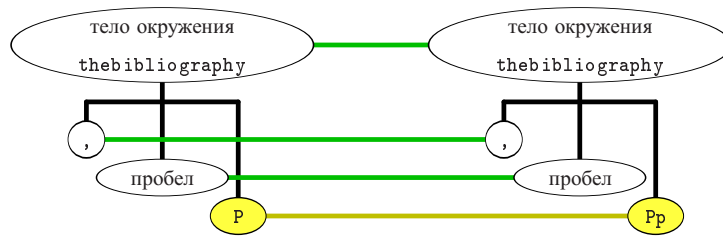


Рис. 4.3. Пример правила изменения токена.

$$\boxed{, P} \rightarrow \boxed{, Pr}$$

токена шаблона на токен слова «Pr».

4.1.2. Поиск соответствий правилам

Считается, что токен l дерева соответствует локализатору правила, если выполняется совпадение типов токенов и типов их лексем.

Среди потомков l ищется непрерывная последовательность, совпадающая с шаблоном по следующим правилам:

- для всех токенов шаблонных цепочек должно выполняться совпадение типов и лексем с соответствующими потомками l ,
- для целевого токена должно выполняться полное совпадение с соответствующим потомком l .

Определение 9. Позиция правила в синтаксическом дереве документа \LaTeX — это совокупность токена, соответствующего локализатору правила, и набора токенов, соответствующих шаблону. Порождающая позиция правила — позиция, которая соответствует элементу отображения синтаксических деревьев, из которого было синтезировано правило. Множество позиций или позиции правила на множестве документов — совокупность всех позиций в синтаксических деревьях этих документов, удовлетворяющих правилу.

4.1.3. Предварительная оценка правила

Для предварительной оценки качества каждого правила вычисляются данные по обучающей выборке [3]. Обозначим: d_t — количество позиций правила на множестве черновики, c_t — количество позиций правила на множестве чистовиков.

Определение 10. Предварительная (на обучающей выборке) точность правила — это отношение количества позиций, которые соответствуют только черновикам (здесь в качестве этого количества выступает разность числа позиций правила на множестве черновики и числа позиций на множестве чистовиков), к общему числу найденных позиций: $\frac{d_t - c_t}{d_t}$.

4.1.4. Выбор оптимальных шаблонов

Набор токенов образующих шаблон правила можно задавать по-разному. Из результатов экспериментов [2] можно сделать вывод, что максимальные шаблоны не всегда дают лучший результат. В данной работе оптимальный шаблон выбирается по следующим критериям:

1. предварительная точность правила не должна быть меньше 0.9,
2. выбирается наименьший размер шаблона, позволяющий построить правило с допустимой точностью,
3. выбирается правило с наибольшей точностью из всех, обладающих шаблонами выбранного размера.

4.1.5. Редукция набора правил

После выявления закономерностей по всем различиям между деревьями обучающей выборки оказывается, что правил избыточно, поскольку многие

дублируются. Для устранения такого эффекта используется процесс редукции, который заключается в удалении некоторых правил так, чтобы общий набор выделенных закономерностей не менялся.

Определение 11. Правило A поглощает правило B , если операции, соответствующие правилам, совпадают, и множество позиций правила B на черновых документах обучающей выборки является подмножеством позиций правила A .

Из построенного набора правил пошагово удаляются те, которые могут быть поглощены другими.

Определение 12. Множество порождающих позиций или порождающие позиции правила A — совокупность порождающей позиции правила A и множества порождающих позиций правил, которые были поглощены правилом A .

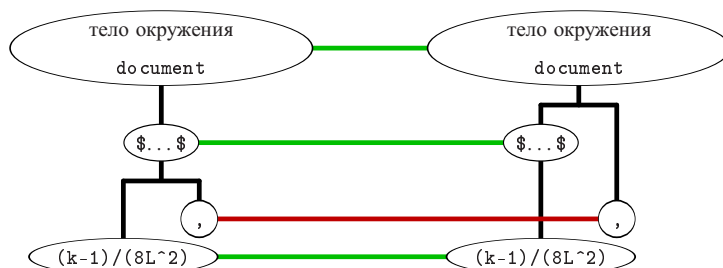


Рис. 4.4. Отображение, не задаваемое тремя видами операций.

$$\boxed{\$(k-1)/(8L^2),\$} \rightarrow \boxed{\$(k-1)/(8L^2)\$,}$$

4.1.6. Операции поднятия и опускания

Пример 12. На рисунке 4.4 показано отображение деревьев, которое должно возникнуть при замене фрагмента $\boxed{\$(k-1)/(8L^2),\$}$ на $\boxed{\$(k-1)/(8L^2)\$,}$, согласно логике корректора. То есть должно произойти перемещение токена, соответствующего запятой, что вызовет нарушение порядка: токен, образованный запятой, является потомком токена формулы, (имеет меньший номер, чем номер токена формулы), а должен стать его правым соседом (иметь номер на 1

больше, чем токен формулы).

Для выделения подобных перемещений набор операций над деревьями был расширен операциями поднятия и опускания. В предположении, что найдено отображение некоторого дерева на другое, введены обозначения: D — множество удалённых вершин, I — множество добавленных вершин, $p(x)$ — родитель вершины x , $f(x)$ — образ вершины x (при этом $\forall x \in D f(x) = \emptyset$), $k(x)$ — ключ вершины x .

Определение 13. Поднятые вершины — это такие вершины x_1, \dots, x_k черного дерева, что для $i = 1, \dots, k$ выполняется:

- $x_i = x_1 + i - 1$ (последовательные),
- $p(x_i) = x_k + 1$ (являются последними потомками общего родителя),

и существуют такие вершины y_1, \dots, y_k чистового дерева, что для $i = 1, \dots, k$ выполняется:

- $y_i = y_1 + i - 1$ (последовательные),
- $k(y_i) = k(x_i)$ (ключи соответствуют удалённым вершинам),
- $p(y_i) = p(y_1)$ (имеют общего родителя),
- $y_1 = f(p(x_1)) + 1$ (следуют за образом родителя x_1, \dots, x_k).

Определение 14. Опущенные вершины — это такие вершины x_1, \dots, x_k черного дерева, что для $i = 1, \dots, k$ выполняется:

- $x_i = x_1 + i - 1$ (последовательные),
- $p(x_i) = p(x_1)$ (имеют общего родителя),

и существуют вершины y_1, \dots, y_k чистового дерева такие, что для $i = 1, \dots, k$ выполняется:

- $y_i = y_1 + i - 1$ (последовательные),
- $k(y_i) = k(x_i)$ (ключи соответствуют удалённым вершинам),
- $p(y_i) = f(x_1 - 1)$ (имеют общего родителя, являющегося образом вершины, предшествующей x_1, \dots, x_k).

Для всех поднятых и опущенных вершин отображение деревьев дополняется парами (x_i, y_i) , $i = 1, \dots, k$.

Справедливо следующее утверждение.

Теорема 1. Пусть есть два дерева T_1 и T_2 , причем T_2 получено из T_1 с помощью операций вставки, удаления, изменения ключа, поднятия и опускания. Если $\varphi_1, \dots, \varphi_n$ — последовательность операций с вершинами, реализующими какое-то отображение T_1 в T_2 , то можно построить набор операций $\varphi'_1, \dots, \varphi'_m$, реализующий выбранное отображение, такой, что $m \leq n$, и все операции поднятия и опускания выполняются после операций вставки, удаления и изменения ключа.

Доказательство. Будем использовать очевидный факт: операции, работающие с разными вершинами, можно менять местами, не изменяя отображения.

Пусть вершина t была удалена. Покажем, что можно исключить другие операции, работающие с t и прообразами t , или заменить их аналогичными, не работающими с t и прообразами t , не изменяя отображения.

Пусть φ_i — удаление t . Тогда, если φ_j работает с t , то $j \leq i$. Пусть φ_j — это последняя из операций, работающих с t , не считая φ_i . Если $i > j + 1$, то переместим операцию удаления t на место $j + 1$, это можно сделать, так как никакая из операций $\varphi_{j+1}, \dots, \varphi_{i-1}$ не работает с t . Теперь удаление t — это операция φ_{j+1} .

Рассмотрим возможные случаи.

- φ_j — вставка t . В этом случае исключение операций φ_j и φ_{j+1} не изменит

отображения.

- φ_j — изменение ключа. В этом случае исключение операции φ_j не изменит отображения.
- φ_j — поднятие вершин x_1, \dots, x_k , причем для некоторого $q \in 1, \dots, k$ $f(x_q) = t$. Если $k = 1$, исключение операции φ_j и замена φ_{j+1} на удаление x_q не изменят отображения. В противном случае, сначала можно удалить вершину x_q , после чего вершины $x_1, \dots, x_{q-1}, x_{q+1}, \dots, x_k$ будут удовлетворять условию для поднятия, то есть операции φ_j и φ_{j+1} можно заменить на операции удаления вершины x_q и поднятия вершин $x_1, \dots, x_{q-1}, x_{q+1}, \dots, x_k$.
- φ_j — опускание вершин x_1, \dots, x_k , причем для некоторого $q \in 1, \dots, k$ $f(x_q) = t$. Если $k = 1$, исключение операции φ_j и замена φ_{j+1} на удаление x_q не изменят отображения. В противном случае, сначала можно удалить вершину x_q , после чего вершины $x_1, \dots, x_{q-1}, x_{q+1}, \dots, x_k$ будут удовлетворять условию для опускания, то есть операции φ_j и φ_{j+1} можно заменить на операции удаления вершины x_q и опускания вершин $x_1, \dots, x_{q-1}, x_{q+1}, \dots, x_k$.

Таким образом можно избавиться от всех операций, работающих с t или образами t .

Аналогично, если вершина t была вставлена, можно исключить другие операции, работающие с t или образами t , или заменить их аналогичными, не работающими с t или образами t , не изменяя отображения.

Таким образом, поскольку каждая из операций удаления и вставки работает только с одной вершиной, можно изменить порядок всех операций так, что удаление и добавление вершин происходит в самом начале.

Остается заметить, что операции поднятия и опускания не зависят от ключа вершин, т. е. их можно менять местами с операциями изменения ключа. \square

Смысл теоремы заключается в том, что для построения отображения деревьев, используя все пять операций, можно сначала воспользоваться алгоритмом для построения редактирующего расстояния, использующего операции вставки, удаления и изменения ключа, затем поднятые и опущенные вершины нужно искать среди удалённых (множество D), а их образы — среди добавленных (множество I).

Единственная цель введения операций поднятия и опускания в этом исследовании заключалась в расширении набора действий, описываемых правилами с линейным шаблоном: становится возможным изменение порядка следования вершин и допускаются операции, изменяющие несколько вершин одновременно. Однако в последующем был разработан более гибкий подход, заключающийся в построении групповых правил. Поэтому далее операции поднятия и опускания вершин не рассматриваются.

4.1.7. Оценки качества набора правил

После того, как построен конечный набор правил можно уточнить построенные ранее оценки. Обозначим: p_t — количество порождающих позиций правила, d_c — количество позиций правила на множестве черновиков контрольной выборки, c_c — количество позиций правила на множестве чистовиков контрольной выборки, p_c — количество позиций правила на множестве черновиков контрольной выборки, которые соответствуют корректным изменениям.

Определение 15. Скорректированная предварительная точность правила — это отношение количества позиций правила, которые действительно соответствуют только черновикам, к общему числу найденных позиций: $\frac{p_t}{d_t}$.

Определение 16. Контрольная точность правила — это отношение количества позиций в контрольной выборке, которые соответствуют только черновикам, к общему числу найденных позиций: $\frac{d_c - c_c}{d_c}$.

Определение 17. Скорректированная контрольная точность правила — это отношение количества позиций в контрольной выборке, которые соответствуют корректным изменениям, к общему числу найденных позиций: $\frac{p_c}{d_c}$.

Теперь можно определить оценки качества для набора правил.

Определение 18. Пусть $P(A_1), \dots, P(A_k)$ — скорректированные предварительные точности правил A_1, \dots, A_k соответственно, а их позиции таковы, что соответствуют изменению одного и того же токена. Весом правила A_i называется $W(A_i) = \frac{P(A_i)}{\sum_{j=1}^k P(A_j)}$.

Определение 19. Пусть $E(A_i)$ — число, равное 0, если правило A_i соответствует верной правке, и 1 в противном случае. Тогда выражение

$$e_x = \sum_{i=1}^k W(A_i) E(A_i) = \frac{\sum_{i=1}^k P(A_i) E(A_i)}{\sum_{i=1}^k P(A_i)}$$

задает среднюю ошибку набора правил на выбранном токене x .

Обозначим: E_t и E_c — суммы средних ошибок набора правил на всех токенах черновых деревьев обучающей и контрольной выборок соответственно, N_t и N_c — количества различных позиций всех правил набора на множествах черновики обучающей и контрольной выборок соответственно, D_t и D_c — суммы редактирующих расстояний для всех пар черновых и чистовых деревьев обучающей и контрольной выборок соответственно.

Поскольку правила синтезируются только при добавлении, удалении или изменении токена, а сумма таких операций для двух деревьев равна редактирующему расстоянию, будут корректны следующие определения [3].

Определение 20. $\frac{N_t - E_t}{N_t}$ — предварительная точность набора правил.

$\frac{N_c - E_c}{N_c}$ — контрольная точность набора правил.

Определение 21. $\frac{N_t - E_t}{D_t}$ — предварительная полнота набора правил.

$\frac{N_c - E_c}{D_c}$ — контрольная полнота набора правил.

4.1.8. Эксперимент

Для оценки качества набора правил был проведен эксперимент, в котором использовалось 85 пар черновых и чистовых статей конференции ИОИ-8 [20]. Моделировалось адаптивное обучение набора правил. Для этого обучающее множество пар документов, используемое для построения правил, постепенно увеличивалось: 2, 3, 4, 6, 9, 13, 19, 28, 42, 63. Обозначим $S_1 \subset \dots \subset S_{10}$ — полученные десять обучающих множеств пар документов, S_{11} — множество всех пар документов. На каждом шаге контрольное множество формировалось из пар документов, которые добавлялись к обучающему множеству на следующем шаге: $S_{i+1} \setminus S_i$, $i = 1, \dots, 10$.

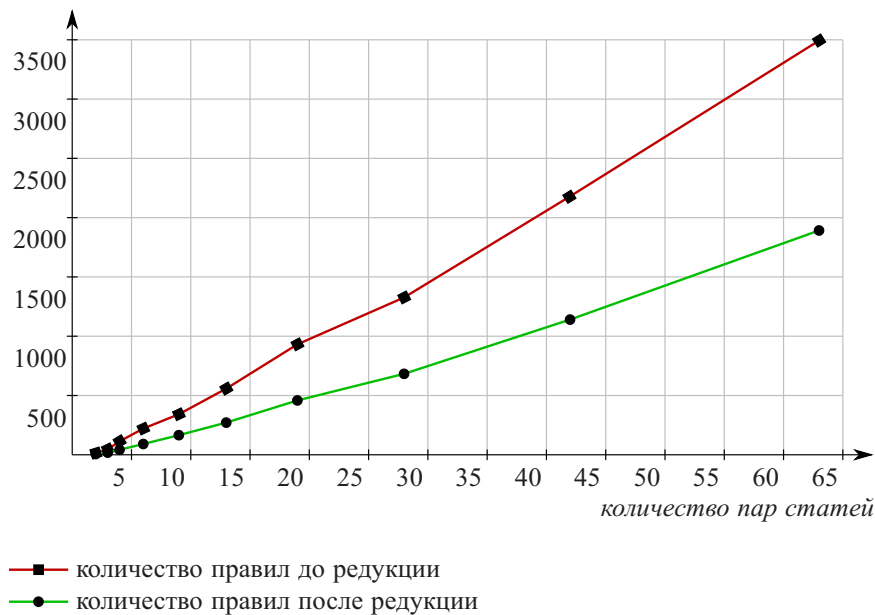


Рис. 4.5. Количества синтезированных правил.

Вычислялись количества синтезированных правил, оценки качества отдельных правил и наборов [3]. Последовательности множеств пар документов строились 50 раз, данные по всем построениям были усреднены. Результаты

проведённых расчетов для наборов правил с простой структурой представлены на рисунках.

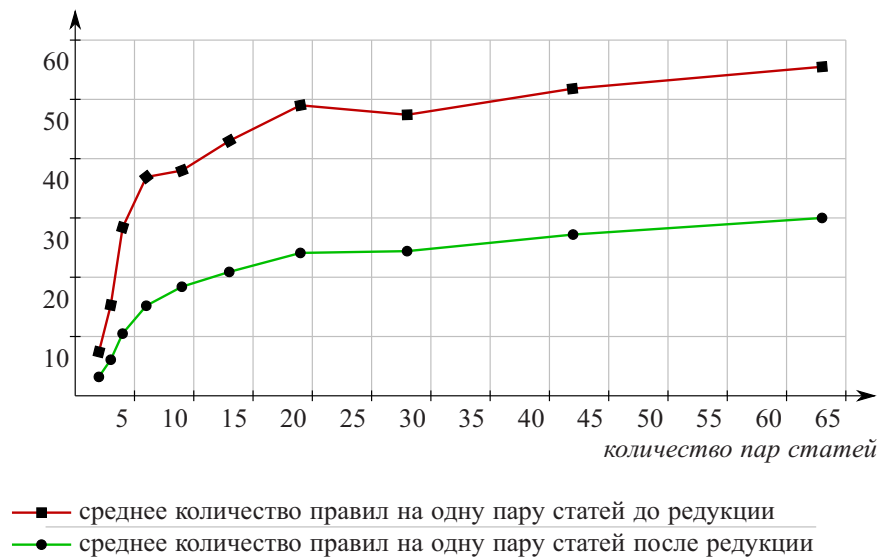


Рис. 4.6. Усреднённые количества синтезированных правил на одну пару «черновик-чистовик».

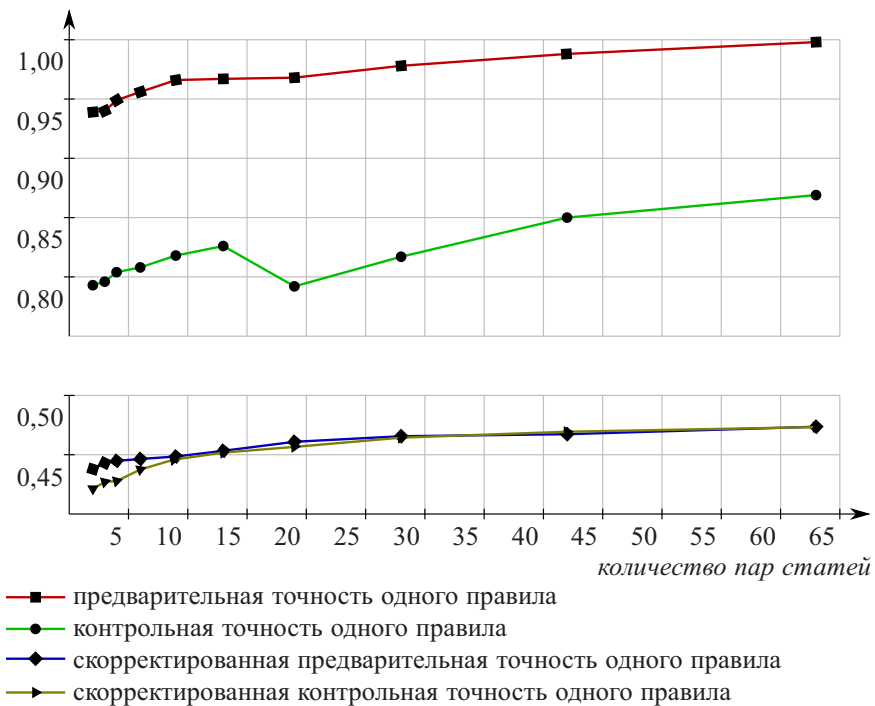


Рис. 4.7. Усреднённые оценки точности одного правила.

Как можно видеть из рисунка 4.5, редукция позволяет значительно сократить количество правил. Это означает, что большинство различий между

черновиками и чистовиками являются типовыми.

Графики на рисунке 4.6 соответствуют возрастающим функциям. Из этого можно сделать вывод, что между черновиками и чистовиками также существует не пренебрежимое число уникальных различий.

На рисунке 4.7 кривые, соответствующие скорректированным оценкам точности правила, расположены довольно близко друг другу. То же можно сказать про кривые на рисунке 4.8, соответствующие оценкам точности и полноты набора правил. Это означает, что синтезированные предложенным способом правила обладают неплохой обобщающей способностью.

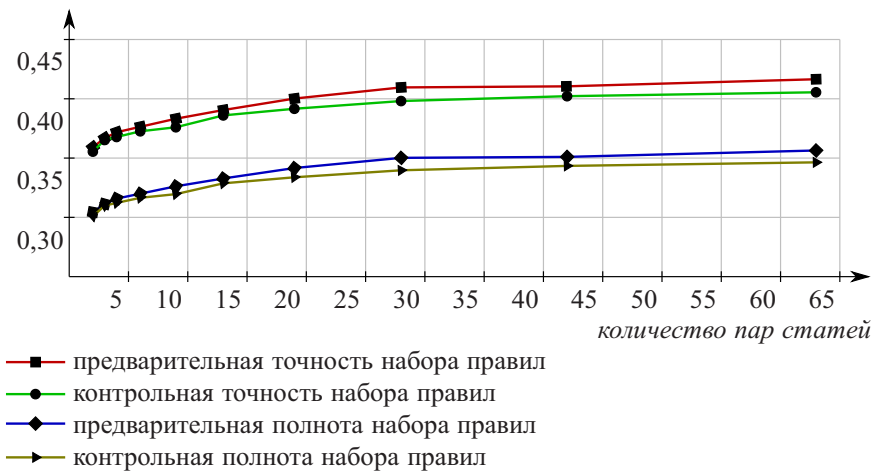


Рис. 4.8. Оценки точности и полноты набора правил с простой структурой.

С другой стороны, и точность, и полнота наборов правил не превосходят 50%. Для точности это означает, что существуют различные правила со схожими шаблонами. Недостаток полноты можно объяснить тем, что рассмотренных типов правил недостаточно для описания действий корректора.

4.2. Групповые правила

На практике встречаются случаи, когда корректор изменяет, удаляет или добавляет более одного токена. Например, перенос одного токена на другую позицию представляет собой совокупность удаления и добавления токена.

Для увеличения спектра обрабатываемых правок корректора мы будем использовать группировку правил.

Пример 13 (Групповые правила). Для разных издательство могут различаться стандарты оформления знаков тире. Это приводит, например, к необходимости изменения $\boxed{\sim\text{---}}$ на $\boxed{\text{'---}}$. На рисунке 4.9 показано отображение синтаксических деревьев, которое возникает при изменении фрагмента текста $\boxed{\text{это}\sim\text{---}\text{определение}}$ на $\boxed{\text{это}\text{'---}\text{определение}}$.

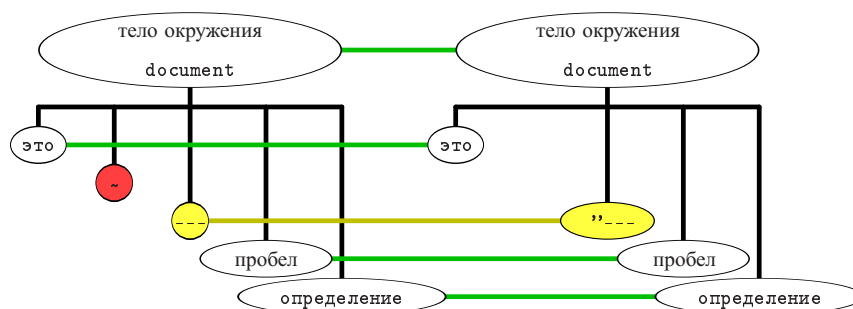


Рис. 4.9. Пример группового правила.

$\boxed{\text{это}\sim\text{---}\text{определение}}$ \rightarrow $\boxed{\text{это}\text{'---}\text{определение}}$

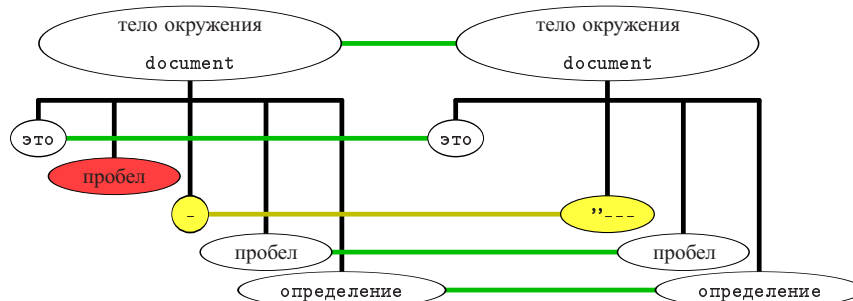


Рис. 4.10. Пример группового правила.

$\boxed{\text{это}\text{---}\text{определение}}$ \rightarrow $\boxed{\text{это}\text{'---}\text{определение}}$

Первый фрагмент состоит из токенов неразрывного пробела ($\boxed{\sim}$) и символа тире ($\boxed{\text{---}}$), второй — из токена символа неотрывного тире ($\boxed{\text{'---}}$). Каждое правило с простой структурой может изменить только один токен, поэтому такая замена ими не реализуется. С другой стороны, совокупность удаления токена неразрывного пробела, стоящего перед токеном тире, и замены токена тире на токен неотрывного тире даёт нужную замену.

Аналогичная проблема возникает при исправлении одной из самых распространённых типографических ошибок — использование дефиса вместо знака тире. Соответствующее отображение синтаксических деревьев показано на рисунке 4.10.

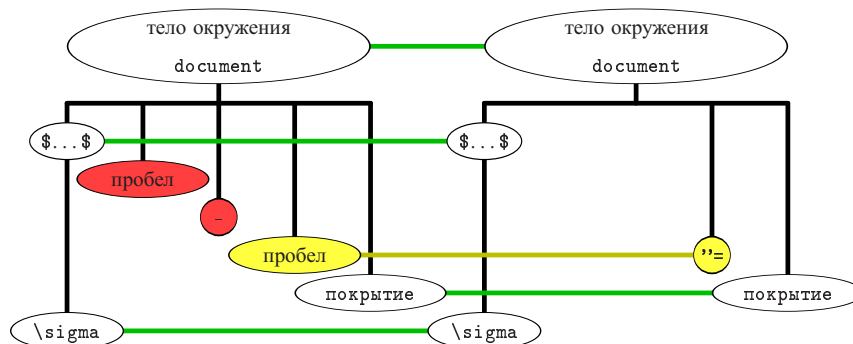


Рис. 4.11. Пример группового правила.

`$$\sigma$ - покрытие` → `$$\sigma$'=покрытие`

Ещё один схожий пример связан с грамотным использованием символа дефиса. На рисунке 4.11 показано отображение синтаксических деревьев, которое возникает при изменении фрагмента текста `$$\sigma$ - покрытие` на `$$\sigma$'=покрытие`.

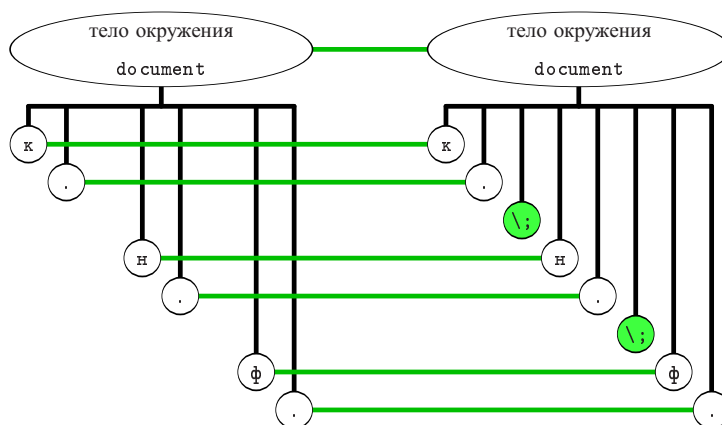


Рис. 4.12. Пример группового правила.

`к.н.ф.` → `к.\;н.\;ф.`

Иногда требуется сделать вставку более, чем одного элемента. Пример отображения синтаксических деревьев такого случая, когда фрагмент `к.н.ф.`

заменяется на $\boxed{\text{к.}\backslash;\text{н.}\backslash;\text{ф.}}$, показан на рисунке 4.12. Здесь добавляются два токена символа $\boxed{\backslash;}$.

4.2.1. Построение групповых правил

Пусть для двух правил существуют позиции такие, что:

- токены, соответствующие локализаторам, совпадают;
- наборы токенов, соответствующих шаблонам, имеют общие элементы.

Тогда построим новое групповое правило, локализатор которого совпадает с локализатором рассматриваемых правил, а шаблон образуется объединением их шаблонов. Построенное правило добавляется в набор, если его предварительная точность выше, чем предварительная точность каждого из рассматриваемых правил.

4.2.2. Применение групповых правил

Считается, что позиция в синтаксическом дереве документа формата L^AT_EX удовлетворяет групповому правилу, если она удовлетворяет каждому из сгруппированных правил. При этом учитывается взаимное расположение их шаблонов: перекрытие должно происходить по тем же токенам, по которым оно происходило в момент построение группового правила.

4.2.3. Эксперимент

На рисунке 4.13 показаны оценки качества набора правил с учетом групповых правил коррекции, построенные в соответствии с экспериментом, описанным в разделе 4.1.8.

Можно видеть, что такой подход позволил получить точность заметно больше половины, но полнота все ещё находится на уровне 50%.

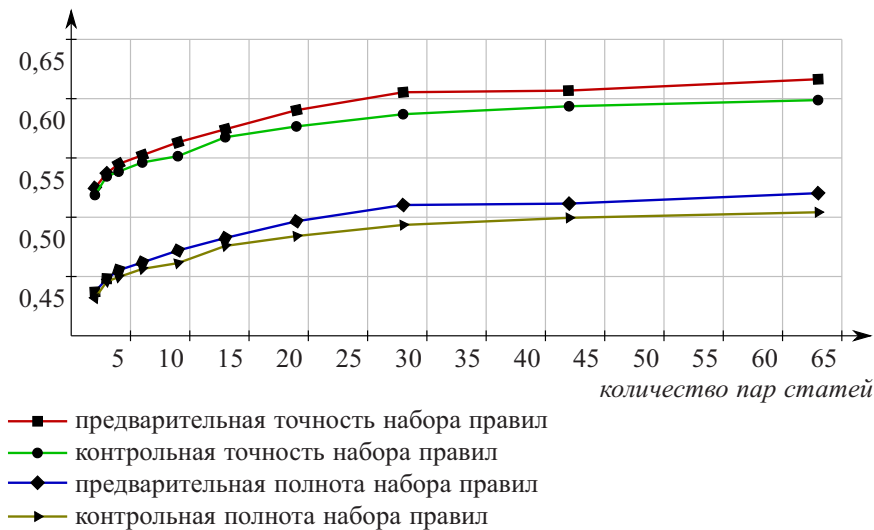


Рис. 4.13. Оценки точности и полноты набора правил с учетом группировки.

4.3. Правила с древовидными шаблонами

Ещё один способ повышения точности и полноты набора правил заключается в усложнении структуры шаблона. Шаблон правила с простой структурой позволяет использовать только соседние токены для определения позиции, что, вообще говоря, не означает использование всего текста, соответствующего этим токенам, поскольку не учитываются структура и содержимое поддеревьев, корни которых образуют шаблон.

Шаблон двевовидного правила будем строить из двух шаблонных деревьев: левого и правого. В этом случае длина шаблона — количество токенов в этих деревьях.

4.3.1. Построение древовидных правил

Синтез таких правил, выбор оптимальных шаблонов и редукция происходят аналогично, синтезу правил с простой структурой.

- Пусть токен x чернового дерева удалён или изменён на токен y . Тогда локализатор — родительский токен x , шаблон составляется из левого и

правого шаблонных деревьев, таких что все потомки корня левого дерева образуют левую шаблонную цепочку, наиболее близкую к x , а потомки корня правого дерева — аналогичную правую цепочку. В таких случаях токен x будем называть целевым токеном правила. Действие правила заключается в удалении целевого токена или изменении его на токен y , в зависимости от типа правила.

- Пусть в чистовое дерево добавлен токен y . Тогда локализатор — прообраз родительского токена y , если он существует; шаблон составляется левого шаблонного дерева, потомки корня которого образуют левую шаблонную цепочку, начинающуюся в прообразе левого соседа y , если он существует, и аналогичного правого дерева. Действие правила заключается в добавлении токена y между левой и правой шаблонными цепочками.

Для выбора оптимального шаблона используются следующие шаги:

1. предварительная точность правила не должна быть меньше 0.9,
2. выбирается наименьший суммарный размер левого и правого шаблонных деревьев, позволяющий построить правило с допустимой точностью,
3. выбирается правило с наибольшей точностью из всех, обладающих шаблонами выбранного размера.

4.3.2. Применение древовидных правил

Поиск мест применимости осуществляется следующим образом. Считается, что токен l дерева соответствует локализатору правила, если выполняется совпадение типов токенов и типов их лексем. Шаблонные деревья и все их поддеревья проверяются на применимость с помощью проверки на каждом уровне, начиная с потомков токена l , условий:

- совпадение самых правых (для левых поддеревьев) или левых (для правых поддеревьев) токенов-потомков с токенами шаблона,
- применимость соответствующего поддерева для каждого токена-потомка.

Пример 14 (Правила с древовидным шаблоном). Ещё одной из самых распространённых типографических ошибок является включение знака препинания в тело формулы. Пусть соответствующие фрагменты чернового и чистового синтаксических деревьев выглядят так, как показано на рисунке 4.14, что соответствует преобразованию фрагмента $\boxed{\$ \ , \$}$ в $\boxed{\$ \$ \ ,}$ внутри окружения document.

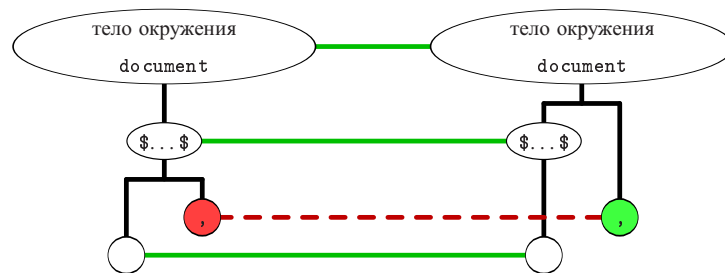


Рис. 4.14. Пример древовидного правила.

$$\boxed{\$ \ , \$} \rightarrow \boxed{\$ \$ \ ,}$$

Такую замену можно описать как совокупность добавления токена запятой после токена формулы, содержащей в конце запятую, и удаления токена запятой из токена формулы. Но для определения наличия запятой внутри формулы требуется обратиться к токенам внутри нее. Таким образом, шаблона правила с простой структурой не будет достаточно.

Поэтому правило добавления запятой описывается древовидным шаблоном, левое поддерево которого состоит из токена формулы, содержащего токен запятой, а правое поддерево пустое. Действие заключается в добавлении токена запятой после левого поддерева.

Из древовидных правил тоже можно получать групповые. На рисунке 4.15 показано отображение синтаксических деревьев, которое возникает при изменении фрагмента текста $\boxed{\text{\textbf{M}}|S|}$ на $\boxed{\text{\Expect}|S|}$. Правило древовидное, поскольку требуется не только обратиться к токenu команды `\textbf`, но и знать её параметр: например, $\boxed{\text{\textbf{D}}}$ нужно заменять на $\boxed{\text{\Var}}$, а не на $\boxed{\text{\Expect}}$. Правило групповое, поскольку нужно изменить не только токен команды `\textbf`, но и удалить токен его параметра.

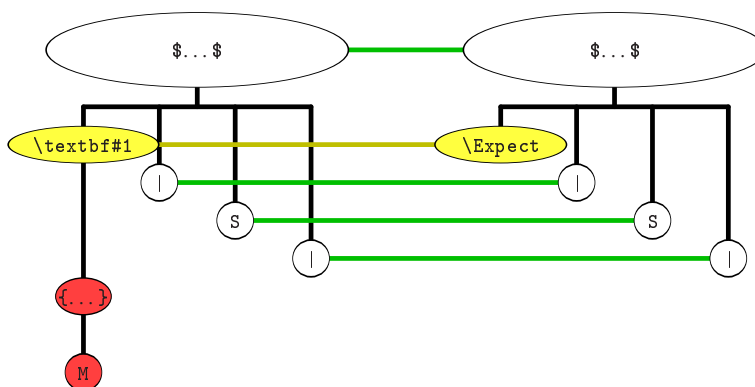


Рис. 4.15. Пример древовидного группового правила.

$$\boxed{\text{\textbf{M}}|S|} \rightarrow \boxed{\text{\Expect}|S|}$$

4.3.3. Эксперимент

На рисунке 4.16 показаны оценки качества набора правил с учетом синтеза древовидных правил, построенные в соответствии с экспериментом, описанным в разделе 4.1.8. Можно видеть, что подобный подход позволил значительно увеличить точность синтезированного набора правил коррекции.

Важно отметить, что итоговая точность превышает 70%, т. е. менее 30% остаются не распознанными, а точность превышает 75%, что позволяет использовать построенный набор правил на практике для полуавтоматического исправления ошибок.

Примеры построенных правил и количества соответствующих позиций приведены в приложении Г. Видно, что правила применимы не только к чер-

новикам, но и к чистовикам. Исследование позиций применимости правил показало, что это вызвано особенностями обучающей выборки — не все типографические ошибки в чистовиках исправлены.

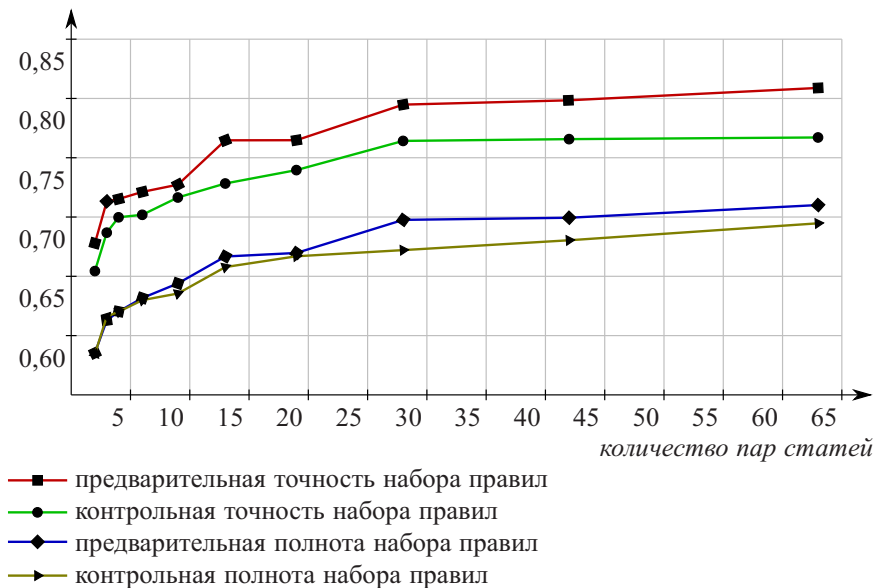


Рис. 4.16. Оценки точности и полноты набора правил с учетом древовидных правил.

4.4. Выводы главы 4

1. Исследованы правила коррекции с простой структурой, описываемые линейным шаблоном. Описана методика построения таких правил на основе обучающей выборки, составленной из пар документов «черновик–чистовик», и последующего их применения. Показано как выбираются оптимальные шаблоны и производится редукция набора построенных правил.
2. Предложена методика для построения оценок точности и полноты набора правил и точности отдельных правил коррекции.
3. Проведен эксперимент на корпусе статей конференции ИОИ-8 [20], в ходе которого исследовалась зависимость точности и полноты набора пра-

вил от количества статей, используемых для обучения.

4. По результатам эксперимента выяснено, что полнота и точность наборов правил с простой структурой не высоки. Это обосновывается небольшим количеством поддерживаемых операций (обрабатываются изменения только одного токена за раз) и чрезмерной простотой используемых шаблонов (требуется иметь возможность исследовать более одного уровня потомков токена-локализатора).
5. Для улучшения качества правил предложены два подхода: построение групповых правил и правил с древовидными шаблонами. Группировка правил позволяет выполнять операции с набором токенов одновременно. Древовидные шаблоны позволяют обращаться к разным слоям синтаксического дерева документа.
6. Для групповых и древовидных правил повторно проведено исследование зависимости полноты набора правил от количества пар статей, используемых для обучения, при тех же условиях, что и для правил с простой структурой.
7. Каждый из подходов позволил заметно улучшить точность и полноту синтезируемого набора правил.
8. Полученные результаты можно считать допустимыми для практического использования.

Заключение

1. В диссертации впервые задача автоматической коррекции текстовых документов формата \LaTeX формулируется как задача обучения по прецедентам, в которой обучающая выборка составляется из пар документов «черновик–чистовик». Такая постановка задачи была вызвана тем, что нет единых требований к оформлению публикуемого материала, и для разных издательств могут требоваться различные алгоритмы коррекции (различные наборы правил). А в силу большого количества и слабой формализации рекомендаций, используемых корректорами при обработке документов, ручное составление набора правил, пригодного для автоматического применения, оказывается чрезмерно трудоемким. Рассматриваемый подход позволяет в автоматическом режиме получить набор правил коррекции документов, используя множество пар документов: до обработки корректором и после.
2. Документы формата \LaTeX можно сравнивать как текстовые файлы или как синтаксические деревья. В первом случае не учитывается логическая структура документа, что приводит к некачественному сравнению и невозможности использования его результатов для поиска правил коррекции. Во втором случае получаются синтаксические деревья с большим количеством вершин и ключевых корней, что приводит к низкой эффективности и высоким требованиям к объему памяти. В диссертации предложен алгоритм сравнения структурированных текстовых документов (на примере файлов формата \LaTeX), использующий их представление в виде синтаксических деревьев. Алгоритм основан на выделении удалённых, добавленных и не изменённых вершин деревьев с помощью сопоставления текстовых представлений документов. Предложенный под-

ход позволяет значительно сократить количество вершин, сравниваемых при построении отображения синтаксических деревьев. Это приводит к уменьшению требований к памяти и времени работы.

3. Построенные различия между синтаксическими деревьями документов формата \LaTeX исследуются для автоматического выделения правил замены, аналогичных используемым корректорами при обработке текстов. В диссертации предлагается алгоритм автоматического построения правил удаления, вставки или изменения отдельных вершин деревьев. Каждое правило описывается линейным шаблоном токенов, позволяет локализовать ошибку и предоставляет вариант её исправления.
4. Показывается, что в некоторых случаях требуются правила, которые изменяют несколько вершин одновременно, и предлагается алгоритм построения групповых правил, обладающих таким функционалом.
5. Показывается, что для описания позиций применимости правил коррекции не достаточно линейной модели шаблона. Предлагается структура правил, обладающих древовидными шаблонами, и алгоритм для построения таких правил.
6. Для определения качества построенного набора правил используются оценки полноты и точности. Оценка точности отдельных правил может быть использована для автоматического выбора наилучшего варианта исправления выбранной ошибки или для построения ранжированного списка вариантов исправления. В диссертации предлагается методика построения оценок полноты и точности синтезированного набора правил и точности отдельных правил. Вычисление оценок производится с использованием корпуса статей конференции ИОИ-8.

Список литературы

1. Чувилин К. В. Синтез правил коррекции документов в формате \LaTeX с помощью сопоставления синтаксических деревьев // Труды 15-й всероссийской конференции «Математические методы распознавания образов». Москва: МАКС Пресс, 2011. С. 597–600.
2. Чувилин К. В. Автоматический синтез правил коррекции документов в формате \LaTeX и их улучшение на основе статистической оценки качества // Труды II Всероссийской научной конференции молодых ученых с международным участием «Теория и практика системного анализа». 2012. С. 17–25.
3. Чувилин К. В. Адаптивное обучение правил коррекции документов в формате \LaTeX // Труды 9-й международной конференции «Интеллектуализация обработки информации». Москва: МАКС Пресс, 2012. С. 652–655.
4. Чувилин К. В. Использование синтаксических деревьев для автоматизации коррекции документов в формате \LaTeX // Компьютерные исследования и моделирование. 2012. Т. 4, № 54. С. 871–883.
5. Чувилин К. В. Гибридный алгоритм сравнения документов в формате \LaTeX // Прикладная информатика. 2013. № 4. С. 56–64.
6. Чувилин К. В. Использование правил со сложной структурой для коррекции документов в формате \LaTeX // Машинное обучение и анализ данных. 2013. Т. 1, № 5. С. 632–640.
7. André J., Richy H. Paper-less editing and proofreading of electronic documents. 1999. URL: <http://www.irisa.fr/imadoc/articles/1999/heidelberg.pdf>.
8. Большаков И. А. Проблемы автоматической коррекции текстов на флективных языках // Итоги науки и техн. Сер. Теор. вероятн. Мат. стат. Теор. кибернет. 1988. Т. 28. С. 111–139.

9. Lightproof grammar checker development framework. URL: <http://extensions.services.openoffice.org/project/lightproof>.
10. Панина М. Ф., Байтин А. В., Галинская И. Е. Автоматическое исправление опечаток в поисковых запросах без учета контекста // Компьютерная лингвистика и интеллектуальные технологии. По материалам ежегодной Международной конференции «Диалог». Т. 1. 2013. С. 556–567.
11. Молчанов А. Статистические и гибридные методы перевода в технологиях компании PROMT // Control Engineering Россия. 2013. № 4. С. 68–72.
12. Williams C., Hollingsworth J. Automatic Mining of Source Code Repositories to Improve Bug Finding Techniques // IEEE Transactions on Software Engineering table of contents archive. 2005. Vol. 31, no. 6. P. 466–480.
13. Князев Е. Г. Методы обнаружения закономерностей эволюции программного кода // Труды XIV Всероссийской научно-методической конференции «Телематика-2007». СПбГУ ИТМО. Т. 2. 2007. С. 435–436.
14. Madou F., Agüero M., Esperón G., López De Luise D. Software for Improving Source Code Quality // World Academy of Science, Engineering and Technology. 2011. Vol. 59. P. 1259–1265.
15. Львовский С. М. Набор и верстка в системе L^AT_EX. М.: МЦНМО, 2006.
16. Журнал «Машинное обучение и анализ данных» — Указания для авторов. URL: <http://jmla.org/papers/index.php/JMLDA/about/submissions#authorGuidelines>.
17. Журнал «Компьютерные исследования и моделирование» — Для авторов. URL: <http://crm.ics.org.ru/journal/page/avtors/>.
18. Конференция «Математические методы распознавания образов» — Правила оформления докладов. URL: <http://mmro.ru/reports.php>.
19. Международная научная конференция студентов, аспирантов и молодых учёных «Ломоносов-2013» — Требования к оформлению тезисов. URL: http://lomonosov-msu.ru/rus/lom_13_rules.html.

20. Труды 8-й международной конференции «Интеллектуализация обработки информации». Москва: МАКС Пресс, 2010.
21. Воронцов К. В. Математические методы обучения по прецедентам (теория обучения машин). URL: <http://www.machinelearning.ru/wiki/images/6/6d/Voron-ML-1.pdf>.
22. Anderson J. R., Michalski R. S., Carbonell R. S., Mitchell T. M. Machine Learning: An Artificial Intelligence Approach. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1983. Vol. 1.
23. Michalski R. S., Carbonell R. S., Mitchell T. M. Machine Learning: An Artificial Intelligence Approach. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1986. Vol. 2.
24. Маннинг К., Рагхаван П., Шютце Х. Введение в информационный поиск. М.: Вильямс, 2011.
25. Фридл Д. Регулярные выражения, 3-е издание. — Пер. с англ. СПб.: Символ-Плюс, 2008.
26. Смит Б. Методы и алгоритмы вычислений на строках (regex) = Computing Patterns in Strings. М.: «Вильямс», 2006.
27. Anquetil E., Couasnon B., Dambreville F. A Symbol Classifier able to Reject Wrong Shapes for Document Recognition Systems // GREC'99, Jaipur (India). 1999.
28. André J. Petite histoire des signes de correction typographique // Cahiers GUTenberg. 1998. — December. no. 31. P. 45–59.
29. Brown H., Harding R., Lay S. et al. Active Alice: Using Real Paper to Interact with Electronic Text // Electronic Publishing, Artistic Imaging, and Digital Typography. 1998. — April. P. 407–419.
30. Amaya Home Page. URL: <http://www.w3.org/Amaya/>.
31. Андреевски А., Дебили Ф., К. Ф. Об одном важном свойстве лексики естественных языков и его использовании при автоматическом исправлении

- опечаток. 1982. Сб. «Прикладные и экспериментальные лингвистические процессоры». ВЦ СО АН СССР.
32. Kukich K. Techniques for Automatically Correcting Words in Text // ACM Computing Surveys. 1992. Vol. 24, no. 4.
 33. Hunspell: open source spell checking, stemming, morphological analysis and generation under GPL, LGPL or MPL licenses. URL: <http://hunspell.sourceforge.net/>.
 34. Docu-Proof Enterprise. URL: http://www.easyfairs.com/uploads/tx_ef/DocuProof_Brochure_LowRes-19865c.pdf.
 35. Baldwin T., Chai J. Y. Autonomous Self-Assessment of Autocorrections: Exploring Text Message Dialogues // Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Montreal, Canada: 2012. P. 710–719.
 36. Байтин А. В. Исправление поисковых запросов в Яндексе. Вероятностная языковая модель // Российские интернет-технологии 2008. 2008.
 37. Whitelaw C., Hutchinson B., Chung G. Y., Ellis G. Using the Web for Language Independent Spellchecking and Autocorrection // EMNLP'09. P. 890–899.
 38. PROMT — переводчики и словари PROMT для перевода текста с английского, русского, немецкого, французского, испанского, португальского и итальянского языков. URL: <http://www.promt.ru/>.
 39. SYSTRAN — Translation Technologies | Online translation, translation software and tools. URL: <http://www.systransoft.com/>.
 40. Linguattec language technologies. URL: <http://www.linguattec.net/>.
 41. Князев Е. Г. Автоматизированная классификация изменений исходного кода на основе кластеризации метрик в процессе разработки программного обеспечения. Диссертация на соискание ученой степени кандидата технических наук. 2009.
 42. Kagdi H., Collard M., Maletic J. Towards a Taxonomy of Approaches for Min-

- ing of Source Code Repositories // Proceedings of the 2005 international workshop on Mining software repositories MSR '05. ACM SIGSOFT Software Engineering Notes. St. Louis, Missouri: 2005. P. 1–5.
43. Hassan A. E., Holt R. C. Source Control Change Messages: How Are They Used And What Do They Mean? 2004. URL: <http://www.ece.uvic.ca/~ahmed/home/pubs/CVSSurvey.pdf>.
 44. Mockus A., Votta L. G. Identifying reasons for software change using historic databases // Proceedings of the International Conference on Software Maintenance (ICSM). San Jose, California: 2000. P. 120–130.
 45. Demeyer S., Ducasse S., Nierstrasz O. Finding refactorings via change metrics // Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA '00). 2000. P. 166–178.
 46. Raghavan S., Rohana R., Podgurski A., Augustine V. Dex: A Semantic-Graph Differencing Tool for Studying Changes in Large Code Bases // Proceedings of 20th IEEE International Conference on Software Maintenance (ICSM'04). Chicago, Illinois: 2004. — September. P. 188–197.
 47. Maletic J. I., Collard M. L. Supporting Source Code Difference Analysis // Proceedings of 20th IEEE International Conference on Software Maintenance (ICSM'04). Chicago, Illinois: 2004. — September. P. 2010–2019.
 48. Robbes R. Mining a Change-Based Software Repository // MSR: International Workshop on Mining Software Repositories. Minneapolis, USA: 2007. P. 120–124.
 49. Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И. Методы и модели анализа данных: OLAP и Data Mining. СПб: БХВ-Петербург, 2004.
 50. Барсегян А. А., Куприянов М. С., Степаненко В. В., Холод И. И. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP. СПб: БХВ-Петербург, 2007.
 51. Чубукова И. А. Data Mining. М.: Лаборатория Базовых Знаний, 2008.

52. Ханк Д. Э., Уичерн Д. У., Райтс А. Д. Бизнес-прогнозирование. 7-е издание. М.: Вильямс, 2003.
53. Kim S., Whitehead E. J., Zhang Y. Classifying Software Changes: Clean or Buggy? URL: <http://www.cs.ucsc.edu/~ejw/papers/cc.pdf>.
54. Hirschberg D. S. A linear space algorithm for computing maximal common subsequences // Communications of the ACM. 1975. — June. Vol. 18, no. 6. P. 871–883.
55. Zhang K., Shasha D. Simple fast algorithms for the editing distance between trees and related problems // SIAM Journal of Computing. 1989. — December. Vol. 18, no. 6. P. 1245–1262.
56. Кнут Д. Все про TEX = The TEXBook. М.: «Вильямс», 2003.
57. T_EX Live — T_EX Users Group. URL: <http://www.tug.org/texlive/>.
58. Home — MiK_TE_X Project Page. URL: <http://miktex.org/>.
59. MacT_EX — T_EX Users Group. URL: <http://www.tug.org/mactex/>.
60. Левенштейн В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Доклады Академий Наук СССР. 1965. С. 845–848.
61. Гасфилд Д. Строки, деревья и последовательности в алгоритмах. Информатика и вычислительная биология. Невский Диалект, БХВ-Петербург, 2003.
62. Беллман Р. Динамическое программирование. М.: Изд-во иностранной литературы, 1960.
63. Wagner R. A., Fischer M. J. The string-to-string correction problem // J. ACM. 1974. Vol. 21, no. 1. P. 168–173.
64. Needleman S. B., Wunsch C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins // Journal of Molecular Biology. 1970. — March. Vol. 48. P. 443–453.
65. Miller W., Myers E. W. A File Comparison Program // Software — Practice and Experience. 1985. no. 15. P. 1025–1040.

66. Ukkonen E. Algorithms for Approximate String Matching // Information and Control. 1985. P. 100–118.
67. Diff Checker—Online diff tool to find the difference between two text files. URL: <http://www.diffchecker.com/diff>.
68. Hoffmann C. M., O’Donnell M. J. Pattern matching in trees // J. Assoc. Comput. Mach. 1982. Vol. 29. P. 68–95.
69. Shellers P. H. The theory and computation of evolutionary distances // J. Algorithms. 1980. P. 359–373.
70. Shapiro B. A. An algorithm for comparing multiple RNA secondary structures // Comput. Appl. Biosci. 1988. P. 387–393.
71. Sussman J. L., Kim S. H. Three dimensional structure of a transfer RNA in two crystal forms // Science. 1976. P. 853.
72. Tai K.-C. The tree-to-tree correction problem // J. Assoc. Comput. Mach. 1979. Vol. 26. P. 422–433.
73. Zhang K. An algorithm for computing similarity of trees, Tech. Report, Mathematics Department, Peking University, Peking, China. 1983.
74. Zhang K. The editing distance between trees: algorithms and applications, Ph.D. thesis, Department of Computer Science, Courant Institute of Mathematical Sciences, New York University, New York. 1989.

Приложение А

Символы L^AT_EX

Шаблон	Метка конца	Лексема	Область применимости
два переноса строки		par	езде
$\&$		cellbreak	езде
табуляция		space	езде
$;\text{---}$		char	езде
$!$		char	езде, кроме математического режима
$!$		postOperator	математический режим
$'$		char	текстовый режим
$'\sim$		char	текстовый режим
$'$		char	езде
$$$$	$$$$	equation	текстовый режим
$\$$	$\$$	equation	текстовый режим
$'$		char	текстовый режим
$'$		char	езде
$''$		char	езде
$''$		char	езде
\ll		char	езде
\gg		char	езде
$''$		char	езде
‘		char	езде
’		char	езде
$($		char	езде
$)$		char	езде
$*$		char	езде, кроме математического режима
$*$		binaryOperator	математический режим

$\boxed{+}$		char	езде, кроме математического режима
$\boxed{+}$		binaryOperator	математический режим
$\boxed{\pm}$		char	езде, кроме математического режима
$\boxed{\pm}$		binaryOperator	математический режим
$\boxed{\cdot}$		char	текстовый режим
$\boxed{\cdot}$		binaryOperator	математический режим
$\boxed{,}$		char	езде
$\boxed{---}$		char	текстовый режим
$\boxed{--}$		char	текстовый режим
$\boxed{-}$		char	текстовый режим
$\boxed{-}$		binaryOperator	математический режим
$\boxed{-}$		char	текстовый режим
$\boxed{-}$		char	текстовый режим
$\boxed{\cdot}$		char	езде
$\boxed{\dots}$		char	езде
$\boxed{/}$		char	текстовый режим
$\boxed{/}$		binaryOperator	математический режим
$\boxed{:}$		char	езде, кроме математического режима
$\boxed{:}$		binaryOperator	математический режим
$\boxed{;}$		char	езде
$\boxed{\ll}$		char	текстовый режим
$\boxed{<}$		char	текстовый режим
$\boxed{<}$		binaryOperator	математический режим
$\boxed{=}$		char	текстовый режим
$\boxed{=}$		binaryOperator	математический режим
$\boxed{\gg}$		char	текстовый режим
$\boxed{>}$		char	текстовый режим
$\boxed{>}$		binaryOperator	математический режим
$\boxed{?!}$		char	езде

<code>?</code>		char	езде
<code>[</code>		char	езде
перенос строки		hskip	езде
<code>\</code>		hskip	езде
<code>\&</code>		char	езде
<code>\%</code>		char	езде
<code>\!</code>		hskip	езде
<code>\''</code>		char	езде
<code>\'</code>		char	езде
<code>\.</code>		hskip	езде
<code>\,</code>		hskip	езде
<code>\(</code>	<code>\)</code>	equation	текстовый режим
<code>\/</code>		char	текстовый режим
<code>\:</code>		hskip	езде
<code>\;</code>		hskip	езде
<code>\-</code>		letter	езде
<code>\[</code>	<code>\]</code>	equation	текстовый режим
<code>\\</code>		linebreak	езде
<code>_</code>		char	езде
<code>\{</code>		char	езде
<code>\}</code>		char	езде
<code>\ </code>		char	езде
<code>]</code>		char	езде
<code>^</code>		index	математический режим
<code>-</code>		index	математический режим
<code>{</code>	<code>}</code>	brackets	езде
<code> </code>		char	езде
<code>~</code>		char	езде
<code>№</code>		char	езде
<code>o</code>		char	езде
<code>A, ..., Z</code>		letter	езде
<code>a, ..., z</code>		letter	езде

А, ..., Я		letter	езде
а, ..., я		letter	езде
©		char	езде

Приложение Б

Команды L^AT_EX

Имя	Шаблон параметров	Лексема	Режим применимости
<code>\AA</code>		char	математический
<code>\abstract</code>	#1	tag	текстовый
<code>\abstractEng</code>	#1	tag	текстовый
<code>\abstractRus</code>	#1	tag	текстовый
<code>\ACCEPTNOTE</code>		tag	любой
<code>\afterlabel</code>	[#1]	tag	текстовый
<code>\endAlgorithm</code>		tag	текстовый
<code>\algorithm</code>	[#1]	tag	текстовый
<code>\endalgorithm</code>		tag	текстовый
<code>\algorithmic</code>	[#1]	tag	текстовый
<code>\algorithmic</code>		tag	текстовый
<code>\endalgorithmic</code>		tag	текстовый
<code>\align</code>		tag	текстовый
<code>\endalign</code>		tag	математический
<code>\align*</code>		tag	текстовый
<code>\endalign*</code>		tag	математический
<code>\aligned</code>		tag	математический
<code>\endaligned</code>		tag	математический
<code>\alpha</code>		letter	математический
<code>\AMENDNOTE</code>		tag	любой
<code>\And</code>		binaryOperator	математический
<code>\approx</code>		binaryOperator	математический
<code>\Apx</code>		char	математический
<code>\ar</code>	[#1]	char	математический
<code>\arcsin</code>		preOperator	математический
<code>\arctan</code>		preOperator	математический

<code>\arg</code>		preOperator	математический
<code>\argmax</code>		preOperator	математический
<code>\argmin</code>		preOperator	математический
<code>\array</code>	#1	tag	математический
<code>\endarray</code>		tag	математический
<code>\ast</code>		char	математический
<code>\author</code>	[#1]#2	tag	текстовый
<code>\author</code>	#1	tag	текстовый
<code>\authorEng</code>	#1	tag	текстовый
<code>\authorRus</code>	#1	tag	текстовый
<code>\backslash</code>		binaryOperator	математический
<code>\balance</code>		tag	любой
<code>\bar</code>	#1	wrapper	математический
<code>\Bbb</code>		tag	математический
<code>\beta</code>		letter	математический
<code>\bf</code>	#1	wrapper	любой
<code>\BibAuthor</code>	#1	wrapper	любой
<code>\bibitem</code>	#1	item	любой
<code>\BibTitle</code>	#1	wrapper	любой
<code>\BibUrl</code>	#1	wrapper	любой
<code>\Big</code>	#1	char	математический
<code>\big</code>	#1	char	математический
<code>\bigcap</code>		preOperator	математический
<code>\bigcup</code>		preOperator	математический
<code>\Biggl</code>	#1	char	математический
<code>\biggl</code>	#1	char	математический
<code>\Biggm</code>	#1	char	математический
<code>\biggm</code>	#1	char	математический
<code>\Biggr</code>	#1	char	математический
<code>\biggr</code>	#1	char	математический
<code>\Bigm</code>	#1	char	математический
<code>\bigm</code>	#1	char	математический

<code>\bigoplus</code>		preOperator	математический
<code>\Bigr</code>	#1	char	математический
<code>\bigr</code>	#1	char	математический
<code>\bigskip</code>		vskip	любой
<code>\bigtriangledown</code>		char	любой
<code>\bigtriangleup</code>		char	любой
<code>\bigvee</code>		preOperator	математический
<code>\binom</code>		binaryOperator	математический
<code>\bmatrix</code>		tag	математический
<code>\endbmatrix</code>		tag	математический
<code>\bmod</code>		char	математический
<code>\boldsymbol</code>	#1	char	математический
<code>\boxed</code>	#1	wrapper	любой
<code>\brop</code>	#1	tag	любой
<code>\bullet</code>		char	любой
<code>\cal</code>	#1	wrapper	математический
<code>\caption</code>	#1	wrapper	любой
<code>\cases</code>		tag	математический
<code>\endcases</code>		tag	математический
<code>\cap</code>		binaryOperator	математический
<code>\cdot</code>		binaryOperator	математический
<code>\cdotp</code>		binaryOperator	математический
<code>\cdots</code>		binaryOperator	математический
<code>\center</code>		tag	текстовый
<code>\endcenter</code>		tag	текстовый
<code>\centering</code>		tag	любой
<code>\chi</code>		letter	математический
<code>\choose</code>		binaryOperator	математический
<code>\circ</code>		char	математический
<code>\cite</code>	#1	char	любой
<code>\citenb</code>	#1	char	любой
<code>\cline</code>	#1	tag	любой

<code>\colon</code>		binaryOperator	математический
<code>\cond</code>		binaryOperator	математический
<code>\const</code>		char	математический
<code>\Corollary</code>		tag	текстовый
<code>\endCorollary</code>		tag	текстовый
<code>\cos</code>		preOperator	математический
<code>\cr</code>		linebreak	любой
<code>\cup</code>		binaryOperator	математический
<code>\ddots</code>		char	любой
<code>\Def</code>		tag	текстовый
<code>\endDef</code>		tag	текстовый
<code>\Definition</code>		tag	текстовый
<code>\endDefinition</code>		tag	текстовый
<code>\Delta</code>		letter	математический
<code>\delta</code>		letter	математический
<code>\det</code>		preOperator	математический
<code>\dfrac</code>	<code>#1#2</code>	wrapper	математический
<code>\diagdown</code>		char	математический
<code>\displaymath</code>		tag	математический
<code>\enddisplaymath</code>		tag	математический
<code>\displaystyle</code>		tag	математический
<code>\div</code>		binaryOperator	математический
<code>\document</code>		tag	текстовый
<code>\enddocument</code>		tag	текстовый
<code>\dot</code>	<code>#1</code>	wrapper	математический
<code>\dotfill</code>		char	любой
<code>\dots</code>		char	любой
<code>\dotsc</code>		char	любой
<code>\ell</code>		char	математический
<code>\em</code>		tag	любой
<code>\email</code>	<code>#1</code>	tag	любой
<code>\emph</code>	<code>#1</code>	wrapper	любой

<code>\emptyset</code>		char	математический
<code>\Eng</code>		tag	любой
<code>\English</code>		tag	любой
<code>\enumerate</code>		tag	текстовый
<code>\endenumerate</code>		tag	текстовый
<code>\enumerate*</code>		tag	текстовый
<code>\endenumerate*</code>		tag	текстовый
<code>\eqnarray</code>		tag	текстовый
<code>\endeqnarray</code>		tag	математический
<code>\eqno</code>	(#1)	char	математический
<code>\equiv</code>		binaryOperator	математический
<code>\eqref</code>	#1	char	любой
<code>\equation</code>		tag	текстовый
<code>\endequation</code>		tag	математический
<code>\equation*</code>		tag	текстовый
<code>\endequation*</code>		tag	математический
<code>\epsfig</code>	#1	image	любой
<code>\epsilon</code>		letter	математический
<code>\eta</code>		letter	математический
<code>\Example</code>		tag	текстовый
<code>\endExample</code>		tag	текстовый
<code>\exists</code>		preOperator	математический
<code>\exp</code>		preOperator	математический
<code>\Expect</code>		preOperator	математический
<code>\figure</code>	#1	tag	текстовый
<code>\figure</code>		tag	текстовый
<code>\endfigure</code>		tag	текстовый
<code>\figure*</code>	#1	tag	текстовый
<code>\figure*</code>		tag	текстовый
<code>\endfigure</code>		tag	текстовый
<code>\flushright</code>		tag	текстовый
<code>\endflushright</code>		tag	текстовый

<code>\footnote</code>	<code>#1</code>	<code>floatingBox</code>	любой
<code>\footnotesize</code>		<code>tag</code>	любой
<code>\FOR</code>	<code>#1#2\ENDFOR</code>	<code>wrapper</code>	текстовый
<code>\forall</code>		<code>preOperator</code>	математический
<code>\frac</code>	<code>#1#2</code>	<code>wrapper</code>	математический
<code>\Gamma</code>		<code>letter</code>	математический
<code>\gamma</code>		<code>letter</code>	математический
<code>\gather</code>		<code>tag</code>	текстовый
<code>\endgather</code>		<code>tag</code>	математический
<code>\gather*</code>		<code>tag</code>	текстовый
<code>\endgather*</code>		<code>tag</code>	математический
<code>\gathered</code>		<code>tag</code>	математический
<code>\endgathered</code>		<code>tag</code>	математический
<code>\ge</code>		<code>binaryOperator</code>	математический
<code>\geq</code>		<code>binaryOperator</code>	математический
<code>\geqslant</code>		<code>binaryOperator</code>	математический
<code>\gg</code>		<code>binaryOperator</code>	математический
<code>\globalpageref</code>	<code>#1</code>	<code>char</code>	любой
<code>\gtrless</code>		<code>binaryOperator</code>	математический
<code>\guillemotleft</code>		<code>char</code>	текстовый
<code>\guillemotright</code>		<code>char</code>	текстовый
<code>\H</code>		<code>letter</code>	текстовый
<code>\hat</code>	<code>#1</code>	<code>wrapper</code>	математический
<code>\hbox</code>	<code>to #1#2</code>	<code>floatingBox</code>	любой
<code>\headline</code>		<code>tag</code>	любой
<code>\hrule</code>		<code>tag</code>	любой
<code>\hfil</code>		<code>hskip</code>	любой
<code>\hfill</code>		<code>hskip</code>	любой
<code>\hphantom</code>	<code>#1</code>	<code>tag</code>	математический
<code>\hskip</code>	<code>#1</code>	<code>hskip</code>	любой
<code>\hspace*</code>	<code>#1</code>	<code>hskip</code>	любой
<code>\hspace</code>	<code>#1</code>	<code>hskip</code>	любой

<code>\hstrut</code>		tag	любой
<code>\Hypothesis</code>		tag	текстовый
<code>\endHypothesis</code>		tag	текстовый
<code>\IF</code>	<code>#1#2\ELSE #3\ENDIF</code>	wrapper	текстовый
<code>\IF</code>	<code>#1#2\ENDIF</code>	wrapper	текстовый
<code>\iff</code>		binaryOperator	математический
<code>\iint</code>		preOperator	математический
<code>\in</code>		binaryOperator	математический
<code>\includegraphics</code>	<code>[#1]#2</code>	image	любой
<code>\includegraphics</code>	<code>#1</code>	image	любой
<code>\inf</code>		preOperator	математический
<code>\infty</code>		char	математический
<code>\int</code>		preOperator	математический
<code>\it</code>		tag	любой
<code>\item</code>		item	любой
<code>\itemize</code>		tag	текстовый
<code>\enditemize</code>		tag	текстовый
<code>\enskip</code>		hskip	текстовый
<code>\jmath</code>		char	математический
<code>\kern</code>		char	математический
<code>\L</code>		char	любой
<code>\label</code>	<code>#1</code>	tag	любой
<code>\Lambda</code>		letter	математический
<code>\lambda</code>		letter	математический
<code>\land</code>		char	математический
<code>\langle</code>		char	математический
<code>\LARGE</code>		tag	любой
<code>\LaTeXe</code>		char	любой
<code>\lbrace</code>		char	математический
<code>\lceil</code>		char	математический
<code>\ldots</code>		char	любой

<code>\le</code>		binaryOperator	математический
<code>\left</code>	#1	char	математический
<code>\leftarrow</code>		binaryOperator	математический
<code>\lefteqn</code>	#1	wrapper	математический
<code>\Leftrightarrow</code>		binaryOperator	математический
<code>\leftrightharrow</code>		binaryOperator	математический
<code>\Lemma</code>		group	текстовый
<code>\endLemma</code>		tag	текстовый
<code>\leq</code>		binaryOperator	математический
<code>\leqslant</code>		binaryOperator	математический
<code>\lessapprox</code>		binaryOperator	математический
<code>\lesssim</code>		binaryOperator	математический
<code>\lfloor</code>		char	математический
<code>\lim</code>		preOperator	математический
<code>\limits</code>		tag	математический
<code>\line</code>	(#1)#2	char	любой
<code>\linebreak</code>		linebreak	любой
<code>\linethickness</code>	#1	tag	любой
<code>\linewidth</code>		length	любой
<code>\ll</code>		binaryOperator	математический
<code>\ln</code>		preOperator	математический
<code>\lnot</code>		binaryOperator	математический
<code>\log</code>		preOperator	математический
<code>\Longlefttrightharrow</code>		binaryOperator	математический
<code>\longrightharrow</code>		binaryOperator	математический
<code>\lor</code>		binaryOperator	математический
<code>\lrcorner</code>		char	математический
<code>\makebox</code>	(#1)[#2]#3	wrapper	любой
<code>\maketitle</code>		tag	любой
<code>\mapsto</code>		binaryOperator	математический
<code>\mathbb</code>	#1	wrapper	математический
<code>\mathbb</code>	#1	wrapper	математический

<code>\mathbb</code>	#1	wrapper	математический
<code>\mathbb</code>	#1	wrapper	математический
<code>\mathbb</code>	#1	wrapper	математический
<code>\mathop</code>	#1	binaryOperator	математический
<code>\mathord</code>	#1	wrapper	математический
<code>\mathrel</code>	#1	wrapper	математический
<code>\mathrm</code>	#1	wrapper	математический
<code>\mathscr</code>		tag	математический
<code>\mathstrut</code>		tag	математический
<code>\matrix</code>		tag	математический
<code>\endmatrix</code>		tag	математический
<code>\max</code>		preOperator	математический
<code>\mbox</code>	#1	wrapper	любой
<code>\medskip</code>		vskip	любой
<code>\mathbb</code>	#1	wrapper	математический
<code>\mid</code>		binaryOperator	математический
<code>\min</code>		preOperator	математический
<code>\mu</code>		letter	математический
<code>\multicolumn</code>	#1#2#3	wrapper	любой
<code>\multirow</code>	#1#2#3	wrapper	любой
<code>\multline</code>		tag	текстовый
<code>\endmultline</code>		tag	математический
<code>\multline*</code>		tag	текстовый
<code>\endmultline*</code>		tag	математический
<code>\mylim</code>	#1	preOperator	математический
<code>\nabla</code>		letter	математический
<code>\ne</code>		binaryOperator	математический
<code>\neg</code>		binaryOperator	математический
<code>\negmedspace</code>		hskip	любой
<code>\neq</code>		binaryOperator	математический
<code>\newblock</code>		tag	текстовый
<code>\newline</code>		linebreak	текстовый

<code>\nleqslant</code>		binaryOperator	математический
<code>\NN</code>		char	математический
<code>\No</code>		char	любой
<code>\nobreakdash</code>		tag	текстовый
<code>\noindent</code>		tag	текстовый
<code>\nolimits</code>		tag	математический
<code>\nonumber</code>		tag	математический
<code>\nopagebreak</code>		tag	текстовый
<code>\Normal</code>		preOperator	математический
<code>\not</code>	#1	wrapper	математический
<code>\notag</code>		tag	любой
<code>\notin</code>		binaryOperator	математический
<code>\NP</code>		char	любой
<code>\nu</code>		letter	математический
<code>\nulldelimiterspace</code>		hskip	любой
<code>\O</code>		char	математический
<code>\oldphi</code>		letter	математический
<code>\Omega</code>		letter	математический
<code>\omega</code>		letter	математический
<code>\oplus</code>		binaryOperator	математический
<code>\organization</code>	#1	tag	любой
<code>\organizationEng</code>	#1	tag	любой
<code>\organizationRus</code>	#1	tag	любой
<code>\oslash</code>		binaryOperator	математический
<code>\otimes</code>		binaryOperator	математический
<code>\over</code>		binaryOperator	математический
<code>\overleftarrow</code>	#1	wrapper	математический
<code>\overline</code>	#1	wrapper	математический
<code>\overrightarrow</code>	#1	wrapper	математический
<code>\overset</code>	#1	wrapper	математический
<code>\P</code>		char	любой
<code>\pagebreak</code>		tag	любой

<code>\par</code>		par	любой
<code>\parbox</code>	<code>[#1]#2#3</code>	floatingBox	любой
<code>\parbox</code>	<code>#1#2</code>	floatingBox	любой
<code>\partial</code>		char	математический
<code>\paragraph</code>	<code>#1</code>	wrapper	любой
<code>\phantom</code>	<code>#1</code>	tag	математический
<code>\Phi</code>		letter	математический
<code>\phi</code>		letter	математический
<code>\Pi</code>		letter	математический
<code>\pi</code>		letter	математический
<code>\picture</code>	<code>(#1)</code>	tag	текстовый
<code>\endpicture</code>		tag	текстовый
<code>\pm</code>		binaryOperator	математический
<code>\prec</code>		binaryOperator	математический
<code>\prime</code>		char	математический
<code>\PRINT</code>		tag	текстовый
<code>\Prob</code>		char	математический
<code>\prod</code>		preOperator	математический
<code>\Proof</code>		tag	текстовый
<code>\endProof</code>		tag	текстовый
<code>\propto</code>		binaryOperator	математический
<code>\Psi</code>		letter	математический
<code>\psi</code>		letter	математический
<code>\put</code>	<code>(#1)#2</code>	wrapper	текстовый
<code>\qquad</code>		char	любой
<code>\quad</code>		char	любой
<code>\raisebox</code>	<code>#1#2</code>	wrapper	любой
<code>\rangle</code>		char	математический
<code>\rank</code>		preOperator	математический
<code>\rbrace</code>		char	математический
<code>\rceil</code>		char	математический
<code>\Re</code>		char	математический

<code>\ref</code>	#1	char	любой
<code>\REJECTNOTE</code>		tag	ТЕКСТОВЫЙ
<code>\Remark</code>		tag	ТЕКСТОВЫЙ
<code>\endRemark</code>		tag	ТЕКСТОВЫЙ
<code>\REPLY</code>	#1	wrapper	любой
<code>\REVIEWERNOTE</code>	#1	wrapper	любой
<code>\rfloor</code>		char	математический
<code>\rho</code>		letter	математический
<code>\right</code>	#1	char	математический
<code>\Rrightarrow</code>		binaryOperator	математический
<code>\rrightarrow</code>		binaryOperator	математический
<code>\rm</code>		tag	любой
<code>\RR</code>		char	математический
<code>\Rus</code>		tag	любой
<code>\section</code>	#1	wrapper	любой
<code>\selectlanguage</code>	#1	tag	любой
<code>\setlength</code>	#1#2	tag	любой
<code>\setminus</code>		binaryOperator	математический
<code>\sharp</code>		char	математический
<code>\Sigma</code>		letter	математический
<code>\sigma</code>		letter	математический
<code>\sim</code>		binaryOperator	математический
<code>\simeq</code>		binaryOperator	математический
<code>\sin</code>		preOperator	математический
<code>\sl</code>		tag	любой
<code>\sloppy</code>		tag	ТЕКСТОВЫЙ
<code>\endsloppy</code>		tag	ТЕКСТОВЫЙ
<code>\small</code>		tag	любой
<code>\smallmatrix</code>		tag	математический
<code>\endsmallmatrix</code>		tag	математический
<code>\smallskip</code>		vskip	любой
<code>\smash</code>	#1	wrapper	математический

<code>\special</code>	<code>#1</code>	tag	любой
<code>\split</code>		tag	математический
<code>\endsplit</code>		tag	математический
<code>\sqrt</code>	<code>#1</code>	wrapper	математический
<code>\sqsubset</code>		binaryOperator	математический
<code>\sqsubseteq</code>		binaryOperator	математический
<code>\star</code>		char	математический
<code>\STATE</code>		char	текстовый
<code>\State</code>		tag	текстовый
<code>\endState</code>		tag	текстовый
<code>\subfigure</code>	<code>#1</code>	wrapper	любой
<code>\subsection</code>	<code>#1</code>	wrapper	любой
<code>\subset</code>		binaryOperator	математический
<code>\subseteq</code>		binaryOperator	математический
<code>\substack</code>	<code>#1</code>	wrapper	математический
<code>\succ</code>		binaryOperator	математический
<code>\sum</code>		preOperator	математический
<code>\sup</code>		preOperator	математический
<code>\supset</code>		binaryOperator	математический
<code>\T</code>		char	любой
<code>\tabcolsep</code>		tag	текстовый
<code>\tabcolsep</code>		length	текстовый
<code>\table</code>	<code>#1</code>	tag	текстовый
<code>\table</code>		tag	текстовый
<code>\endtable</code>		tag	текстовый
<code>\tabular</code>	<code>[#1]#2</code>	tag	любой
<code>\tabular</code>	<code>#1</code>	tag	любой
<code>\endtabular</code>		tag	текстовый
<code>\tabularnewline</code>		tag	текстовый
<code>\tan</code>		preOperator	математический
<code>\tanh</code>		preOperator	математический
<code>\tau</code>		letter	математический

<code>\tbigcup</code>		preOperator	математический
<code>\text</code>	#1	wrapper	любой
<code>\textbf</code>	#1	wrapper	любой
<code>\textcolor</code>	#1#2	wrapper	текстовый
<code>\textit</code>	#1	wrapper	любой
<code>\textnumero</code>		char	текстовый
<code>\textrm</code>	#1	wrapper	любой
<code>\textsc</code>	#1	wrapper	текстовый
<code>\textsf</code>	#1	wrapper	любой
<code>\textstyle</code>	#1	wrapper	математический
<code>\textsuperscript</code>	#1	wrapper	любой
<code>\texttt</code>	#1	wrapper	любой
<code>\tfrac</code>	#1#2	wrapper	математический
<code>\tg</code>		preOperator	математический
<code>\thanks</code>	#1	tag	любой
<code>\thebibliography</code>	#1	tag	текстовый
<code>\endthebibliography</code>		tag	текстовый
<code>\Theorem</code>		tag	текстовый
<code>\endTheorem</code>		tag	текстовый
<code>\Theta</code>		letter	математический
<code>\theta</code>		letter	математический
<code>\thickapprox</code>		binaryOperator	математический
<code>\thickspace</code>		hskip	любой
<code>\tilde</code>	#1	wrapper	математический
<code>\times</code>		binaryOperator	математический
<code>\title</code>	[#1]#2	tag	любой
<code>\title</code>	#1	tag	любой
<code>\titleEng</code>	#1	tag	любой
<code>\titleRus</code>	#1	tag	любой
<code>\to</code>		binaryOperator	математический
<code>\top</code>		char	математический
<code>\tprod</code>		preOperator	математический

<code>\triangle</code>		char	математический
<code>\trivlist</code>		tag	текстовый
<code>\endtrivlist</code>		tag	текстовый
<code>\tsum</code>		preOperator	математический
<code>\u</code>	#1	wrapper	текстовый
<code>\underbrace</code>	#1	wrapper	математический
<code>\underline</code>	#1	wrapper	любой
<code>\underset</code>	#1	wrapper	математический
<code>\unitlength</code>	=#1	tag	текстовый
<code>\url</code>	#1	wrapper	любой
<code>\Var</code>		preOperator	математический
<code>\varepsilon</code>		letter	математический
<code>\varkappa</code>		letter	математический
<code>\varnothing</code>		char	математический
<code>\varphi</code>		letter	математический
<code>\vartriangle</code>		char	математический
<code>\vdots</code>		char	любой
<code>\vec</code>	#1	wrapper	математический
<code>\vee</code>		binaryOperator	математический
<code>\verb</code>	#1	wrapper	любой
<code>\vert</code>		char	математический
<code>\vfill</code>		vskip	любой
<code>\vphantom</code>	#1	tag	математический
<code>\vskip</code>	#1	vskip	любой
<code>\vspace*</code>	#1	vskip	любой
<code>\vspace</code>	#1	vskip	любой
<code>\wedge</code>		binaryOperator	математический
<code>\WHILE</code>	#1#2\ENDWHILE	char	текстовый
<code>\widehat</code>	#1	wrapper	математический
<code>\widetilde</code>	#1	wrapper	математический
<code>\Xi</code>		char	математический
<code>\xi</code>		char	математический

<code>\xrightarrow</code>	#1	wrapper	математический
<code>\XX</code>		char	математический
<code>\xymatrix</code>	#1	table	математический
<code>\XYtext</code>	(#1,#2)#3	tag	любой
<code>\zeta</code>		letter	математический
<code>\ZZ</code>		char	математический

Приложение В

Окружения \LaTeX

Имя	Лексема	Область применимости
Algorithm	floatingBox	текстовый режим
algorithm	floatingBox	текстовый режим
algorithmic	floatingBox	текстовый режим
align	table	текстовый режим
align*	table	текстовый режим
aligned	table	математический режим
array	table	математический режим
bmatrix	table	математический режим
cases	table	математический режим
center	wrapper	текстовый режим
Corollary	wrapper	текстовый режим
Def	wrapper	текстовый режим
Definition	wrapper	текстовый режим
displaymath	equation	текстовый режим
document	wrapper	текстовый режим
enumerate	list	текстовый режим
enumerate*	list	текстовый режим
eqnarray	equation	текстовый режим
equation	equation	текстовый режим
equation*	equation	текстовый режим
Example	wrapper	текстовый режим
figure	floatingBox	текстовый режим
figure*	floatingBox	текстовый режим
flushright	floatingBox	текстовый режим
gather	equation	текстовый режим
gather*	equation	текстовый режим
gathered	equation	математический режим

Hypothesis	wrapper	текстовый режим
itemize	list	текстовый режим
Lemma	wrapper	текстовый режим
matrix	table	математический режим
multline	table	текстовый режим
multline*	table	текстовый режим
picture	floatingBox	текстовый режим
Proof	wrapper	текстовый режим
Remark	wrapper	текстовый режим
sloppy	wrapper	текстовый режим
smallmatrix	table	математический режим
split	wrapper	математический режим
State	wrapper	текстовый режим
table	floatingBox	текстовый режим
tabular	table	езде
thebibliography	list	текстовый режим
Theorem	wrapper	текстовый режим
trivlist	list	текстовый режим

Приложение Г

Примеры синтезированных правил коррекции

Локазизатор	тело окружения document
Шаблон поиска	space символ [] space
Шаблон замены	символ [] space
Соответствия в черновиках: 44 документа, 189 позиций.	
Соответствия в чистовиках: 14 документов, 17 позиций.	

Локазизатор	тело окружения document
Шаблон поиска	символ \$. . . символ [\ ,] \$
Шаблон замены	символ \$. . . \$ символ [\ ,]
Соответствия в черновиках: 18 документов, 122 позиции.	
Соответствия в чистовиках: 13 документов, 11 позиций.	

Локазизатор	содержимое символа { . . . }
Шаблон поиска	команда \em space token-слово
Шаблон замены	space token-слово
Соответствия в черновиках: 5 документов, 45 позиций.	
Соответствия в чистовиках: 3 документа, 4 позиции.	

Локазизатор	тело окружения document
Шаблон поиска	space символ [“] token-слово символ [”] space
Шаблон замены	space символ [<<] token-слово символ [>>] space
Соответствия в черновиках: 12 документов, 37 позиций.	
Соответствия в чистовиках: 2 документа, 3 позиции.	

Локазизатор	тело окружения thebibliography
Шаблон поиска	char space слово [P]
Шаблон замены	char space слово [Pp]
Соответствия в черновиках: 6 документов, 28 позиций.	
Соответствия в чистовиках: 2 документа, 2 позиции.	

Локализатор	телo окружения thebibliography
Шаблон поиска	СИМВОЛ [.] space token-число
Шаблон замены	СИМВОЛ [.] space token-число
Соответствия в черновиках: 11 документов, 23 позиции.	
Соответствия в чистовиках: 1 документ, 1 позиция.	

Локализатор	телo окружения thebibliography
Шаблон поиска	space token-число char СИМВОЛ [---] space слово [P]
Шаблон замены	СИМВОЛ [\\,] token-число char СИМВОЛ [”---] space слово [P]
Соответствия в черновиках: 7 документов, 21 позиция.	
Соответствия в чистовиках: 2 документа, 2 позиции.	

Локализатор	телo окружения thebibliography
Шаблон поиска	space token-число char СИМВОЛ [”---] space слово [C]
Шаблон замены	СИМВОЛ [\\,] token-число char СИМВОЛ [---] space слово [C]
Соответствия в черновиках: 10 документов, 15 позиций.	
Соответствия в чистовиках: 1 документ, 1 позиция.	

Локализатор	параметр команды \thanks#1
Шаблон поиска	команда \No hspace space token-число
Шаблон замены	команда \No hspace token-число
Соответствия в черновиках: 9 документов, 11 позиций.	
Соответствия в чистовиках: 1 документ, 1 позиция.	

Локализатор	телo окружения Corollary
Шаблон поиска	space equation char
Шаблон замены	СИМВОЛ [~] equation char
Соответствия в черновиках: 2 документа, 5 позиций.	
Соответствия в чистовиках: 0 документов, 0 позиций.	