Использование правил со сложной структурой для коррекции документов в формате LaTeX

Чувилин К. В.
kirill.chuvilin@gmail.com
Москва, Московский физико-технический институт (ГУ)

Рассматривается задача автоматического синтеза правил коррекции документов в формате LaTeX. Каждый документ представляется в виде синтаксического дерева. Отображения вершин деревьев черновых документов в вершины деревьев чистовых составляют обучающую выборку, по которой синтезируются правила замены. В первую очередь строятся простые правила, реализующие операции удаления, добавления или изменения одной вершины синтаксического дерева и использующие линейные последовательности вершин для выбора позиции применения. Построенные правила объединяются в группы на основе позиций применимости и оценок качества. Исследуются правила, использующие древовидные структуры вершин для выбора позиции применения. Анализируется изменение качества правил при последовательном наращивании обучающей выборки.

Ключевые слова: *автоматизация, анализ текста, лексема, машинное обучение, метрика, редактирующее расстояние, обучение с подкреплением, синтаксическое дерево, токен, LaTeX.*

The use of rules with complex structure for LaTeX documents correction

Chuvilin K. V.
Moscow Institute of Physics and Technology (SU)

The problem of automatic synthesis of LaTeX documents editing rules is investigated. Each document is represented as a parse tree. Tree nodes mappings of initial documents to edited documents form the training set, which is used to generate the rules. Simple rules that implement removal, insertion, or replacing operations of single node and use linear sequence of vertices to select a position are synthesized primarily. The constructed rules are grouped based on the positions of applicability and quality. The rules that use tree-like structure of nodes to select the position are studied. The changes in the quality of the rules during the sequental increase of training dcocuments set are analized.

Keywords: *automation, editing distance, LaTeX, lexeme, machine learning, metric, parse tree, reinforcement learning, syntax tree, text analysis, token.*

Введение

Поводом для работы послужила подготовка сборников трудов конференций ММРО и ИОИ в 2007–2012 годах. Многие конференции и издательства принимают материалы от авторов в формате LaTeX. В каждом издательстве есть определенные традиции и требования к оформлению публикуемого материала. К ним относятся оформление заголовков, списков, таблиц, библиографии, формул, чисел, и многое другое. Ошибки, связанные с несоблюдением этих правил, называются *типографическими*. Обычно авторские тексты содержат значительное количество (десятки на страницу) таких ошибок, исправление которых производится корректорами вручную. Существуют инструменты для облегчения

процесса ручной корректуры [1], но тем не менее обработка одной страницы занимает до двух часов времени. Поэтому актуальна автоматизация процесса исправления типографических ошибок для сокращения времени и объема ручной работы.

Вообще говоря, идея автоматизации коррекции текстов не нова [2], и на данный момент существуют качественные инструменты для автоматического поиска и исправления орфографических ошибок [3], использующие словари и морфологический анализ словоформ текста. Кроме того, схожая проблема возникает для интеллектуальной коррекции ошибок в запросах поиска [4], с помощью лексических и статистических признаков. 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, посвященная улучшению характеристик исходного кода программ (вероятности возникновения ошибок в отдельных модулях, степени связности модулей и др.). Известны методы [5, 6], позволяющие оценивать характеристики, основываясь на анализе истории изменений репозиториев, и использовать их для поиска ошибок в коде. Они позволяют создавать рекомендательные системы [7] 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 репозиториев, 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. Предлагаемый подход заключается в следующем. Корректор работает с системой, которая использует формально описанные правила коррекции, определяет в исходном тексте возможные места исправлений и предлагает ему вариант замены. Если он согласен с заменой, ему остается только нажать на соответствующую кнопку. Если не согласен, то он делает правку самостоятельно.

Правила можно задавать вручную, непосредственно на основе практического опыта корректоров. Однако ввиду значительного числа и разнообразия ситуаций и вариантов поведения, это приведет скорее к увеличению трудозатрат, особенно на начальном этапе. Поэтому предлагается строить правила, используя корпус уже обработанных пар документов. Документы, не прошедшие корректуру, будем называть *черновиками*, прошедшие — *чистовиками*.

Задача автоматического синтеза правил преобразования документов по обучающей выборке, составленной из пар «черновик–чистовик» рассматривалась в работе [8]. Однако построенные предложенным образом правила обладали рядом недостатков, среди которых: неполное покрытие мест, соответствующих ошибкам, в документах, не входящих в обучающую выборку, и неверные предлагаемые варианты замены.

В данной работе рассматриваются два подхода к повышению качества набора правил с помощью усложнения структуры.

Выделение различий между документами

Документы формата LaTeX в своем текстовом виде представляются как последовательности (допускается вложение) *лексем* следующих типов: синтаксические скобки ({ и }),

пробел, горизонтальный отступ, вертикальный отступ, разделитель абзацев, обрыв строки, символ, цифра, буква слова, команда, тэг, обертка (тэг с замыканием), формула, верхний или нижний индекс, метка, линейное измерение, путь к файлу или папке, список, элемент списка, плавающий бокс, изображение, бинарный математический оператор, математический постоператор, математический преоператор, таблица, конец ячейки таблицы, параметры таблицы, не обрабатываемые данные.

Кроме того, такие файлы обладают естественной древовидной структурой (*синтаксическим деревом*) [9], исследуя которую, можно получить всю необходимую информацию для описания корректорской правки. Узлы этой структуры будем называть *токенами*. При построении синтаксического дерева выделяются следующие виды токенов: символ (буква, цифра, знак математической операции, кавычка, тире и т. п.), команда, параметр команды, окружение, тело окружения, пробел, разделитель абзацев, слово, число, метка, линейный размер, путь к файлу, параметры формата таблицы, не обрабатываемый фрагмент (например, текст внутри окружения `verbatim`). Каждому токену может соответствовать один из типов лексем.

Синтаксическое дерево с точностью компилятора взаимно однозначно определяет документ LaTeX. Правила замены удобно формулировать именно для деревьев.

Будем называть синтаксические деревья чистовиков *чистовыми*, а черновиков — *черновыми*.

Перед выявлением закономерностей в правках корректоров выделяются различия между черновым и чистовым деревьями для каждой пары документов из обучающей выборки. Для этого используется алгоритм Zhang-Shasha [10], который подразумевает, что к синтаксическому дереву разрешается последовательно применять следующие операции: удаление токена (все его потомки переходят родителю), вставка нового токена в произвольное место, изменение токена. Алгоритм Zhang-Shasha позволяет вычислять редактирующее расстояние между двумя деревьями и, кроме того, определять, какую операцию нужно применить к каждой вершине для реализации такого расстояния. В качестве результата его работы получаются пары (прообраз и образ) не измененных токенов, пары (прообраз и образ) измененных токенов, множество удаленных токенов, множество добавленных токенов.

Правила коррекции с простой структурой

После получения отображения черновых и чистовых деревьев строится начальный набор правил коррекции [8]. 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).

Определение 1. *Левая (правая) шаблонная цепочка радиуса r — это последовательность соседних токенов с общим родителем, длиной не больше r. Началом цепочки считается самый правый (левый) ее токен.*

Пусть токен x чернового дерева удален или изменен 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 1 (Правило с простой структурой). Одной из самых распространенных типографических ошибок является использование кавычек "..." вместо «...» в русском тексте. Пусть соответствующие фрагменты чернового и чистового синтаксических деревьев выглядят так, как показано на рис. 1, что соответствует преобразованию фрагмента "**слово** в <<**слово** внутри окружения `document`.

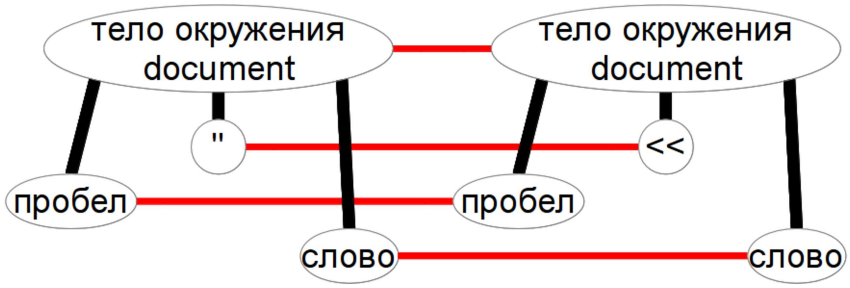

Рис. 1: Соответствующие фрагменты синтаксических деревьев

Тогда локализатором правила является токен тела окружения `document`, левая шаблонная цепочка состоит из токена пробела, правая — из токена слова, шаблон — из токена проблела, 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.

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 2. *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.*

Предварительная оценка правила

Для предварительной оценки качества каждого правила вычисляются данные по обучающей выборке [11]. Обозначим: d_t — количество позиций правила на множестве черновиков, c_t — количество позиций правила на множестве чистовиков.

Определение 3 (Предварительная точность правила). *Предварительная (на обучающей выборке) точность правила — это отношение количества позиций, которые соответствуют только черновикам, к общему числу найденных позиций:* $\frac{d_t - c_t}{d_t}$.

Выбор оптимальных шаблонов

Набор токенов образующих шаблон правила можно задавать по-разному. Из результатов экспериментов [12] 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.

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

Оценки качества правил

Для оценки качества набора правил был проведен эксперимент, в котором использовалось 85 пар черновых и чистовых статей конференции ИОИ-8. 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 $S_1 \subset \ldots \subset S_{10}$ — 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: $S_{i+1} \setminus S_i,\ \ i = 1, \ldots, 10$.

Вычислялись оценки качества для синтезированных правил и наборов правил [11]. Последовательности множеств строились 50 раз, данные по всем построениям были усреднены.

Обозначим: d_t — количество позиций правила на множестве черновиков, c_t — количество позиций правила на множестве чистовиков.

Определение 6. *Предварительная (на обучающей выборке) точность правила — это отношение количества позиций, которые соответствуют только черновикам, к общему числу найденных позиций:* $\frac{d_t - c_t}{d_t}$.

Определение 7. *Пусть $P(A_1), \ldots, P(A_k)$ — предварительные точности правил $A_1, \ldots, A_k$ соответственно, а их позиции правил 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 8. *Пусть $E(A_i)$ — число, равное 0, если правило A_i соответствует верной правке, и 1 в противном случае. Тогда выражение*

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

задает среднюю ошибку набора правил на выбранном токене.

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 черновиков 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 [11].

Определение 9. $\frac{N_t - E_t}{N_t}$ *— предварительная (на обучающей выборке) точность набора правил.* $\frac{N_c - E_c}{N_c}$ *— контрольная (на контрольной выборке) точность набора правил.*

Определение 10. $\frac{N_t - E_t}{D_t}$ *— предварительная (на обучающей выборке) полнота набора правил.* $\frac{N_c - E_c}{D_c}$ *— контрольная (на контрольной выборке) полнота набора правил.*

Результаты проведенных расчетов для наборов правил с простой структурой представлены на рис. 2. Кривые, соответствующие предварительным и контрольным оценкам точности и полноты набора правил, расположены довольно близко друг другу. Это означает, что синтезированные предложенным способом правила обладают неплохой обобщающей способностью.

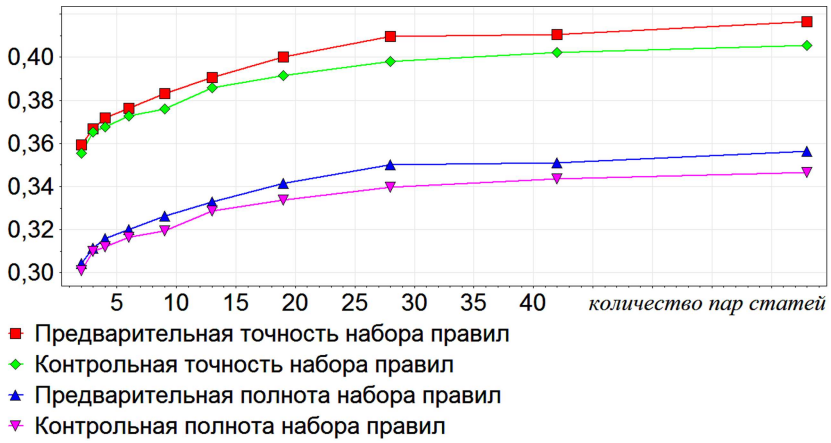

Рис. 2: Оценки точности и полноты набора правил с простой структурой

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.

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.

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.

Пример 2 (Групповое правило). Стандарты оформления знаков тире могут различаться для издательств. Это приводит, например, к необходимости изменения `~---` на `’’---`. Первый фрагмент состоит из токена неразрывного пробела (~) и символа тире (---), второй — из токена символа неотделимого тире (’’---). 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 дает нужную замену.

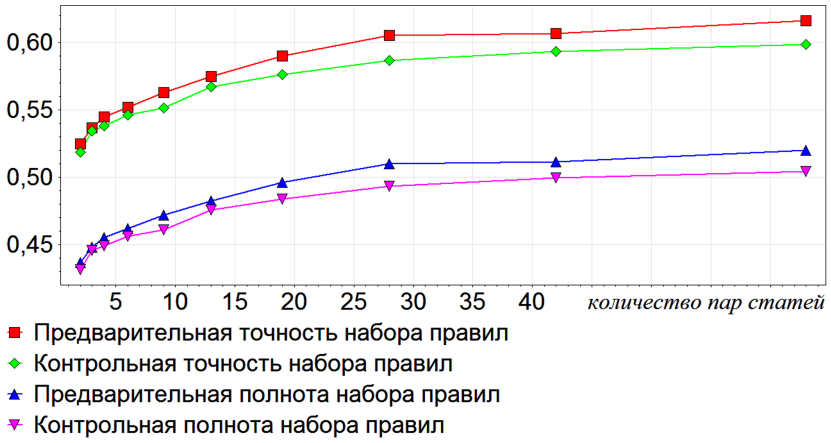

Рис. 3: Оценки точности и полноты набора правил с учетом группировки

На рис. 3 показаны оценки качества набора правил с учетом групповых правил, построенные в соответствии с экспериментом, описанным выше. Можно видеть, что такой подход позволил получить точность заметно больше половины, но полнота все еще находится на уровне 50%.

Правила с древовидными шаблонами

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

Синтез таких правил, выбор оптимальных шаблонов и редукция происходят так же, как и правил с простой структурой, а 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 3 (Правило с древовидным шаблоном). Еще одной из самых распространенных типографических ошибок является включение знака препинания в тело формулы. Пусть соответствующие фрагменты чернового и чистового синтаксических деревьев выглядят так, как показано на рис. 4, что соответствует преобразованию фрагмента `$ ,$` в `$ $,` внутри окружения `document`.

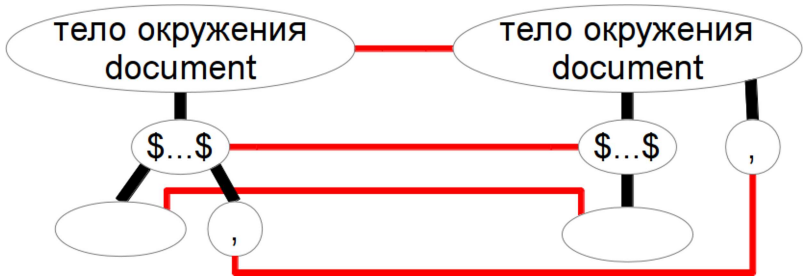

Рис. 4: Соответствующие фрагменты синтаксических деревьев

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, поэтому шаблона правила с простой структурой будет не достаточно.

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

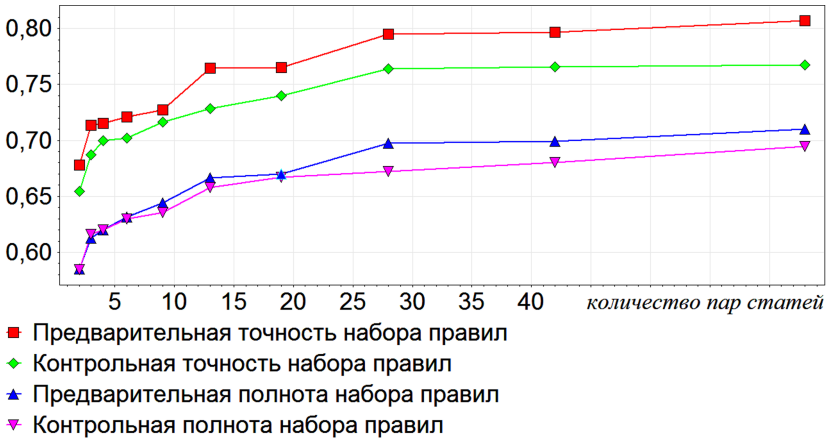

Рис. 5: Оценки точности и полноты набора правил с учетом древовидных правил

На рис. 5 показаны оценки качества набора правил с учетом синтеза древовидных правил, построенные в соответствии с экспериментом, описанным выше. Можно видеть,

что подобный подход позволил значительно увеличить точность синтезированного набора правил.

Заключение

Исследование направлено на улучшение результатов, полученных в работах [8, 9, 11, 12]. Были предложены два новых подхода: группировка правил и древовидные шаблоны. Каждый из подходов позволил заметно улучшить точность и полноту синтезируемого набора правил. Важным является преодоление уровня 50% для точности.

Литература

[1] *André J., Richy H.* Paper-less editing and proofreading of electronic documents. — 1999. — `http://www.irisa.fr/imadoc/articles/1999/heidelberg.pdf`.

[2] *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.

[3] `http://extensions.services.openoffice.org/project/lightproof` — Lightproof grammar checker development framework — 2013.

[4] *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»*. 2013. Вып. 12. Т. 1. С. 556–567.

[5] *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.

[6] *Князев Е. Г.* Методы обнаружения закономерностей эволюции программного кода // *Труды XIV Всероссийской научно-методической конференции «Телематика-2007»*. СПбГУ ИТМО. 2007. Т. 2. С. 435–436.

[7] *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.

[8] *Чувилин К. В.* Синтез правил коррекции документов в формате LaTeX с помощью сопоставления синтаксических деревьев // *Доклады 15-й Всероссийской конференции «Математические методы распознавания образов» ММРО-15*. 2012. С. 597–600.

[9] *Чувилин К. В.* Использование синтаксических деревьев для автоматизации коррекции документов в формате LaTeX // *Компьютерные исследования и моделирование*. 2012. Т. 4, no 4. С. 871–883.

[10] *Zhang K., Shasha D.* Simple fast algorithms for the editing distance between trees and related problems // *SIAM Journal of Computing*. 1989. No. 18. P. 1245–1262.

[11] *Чувилин К. В.* Адаптивное обучение правил коррекции документов в формате LaTeX // *Доклады 9-й Международной конференции «Интеллектуализация обработки информации» ИОИ*. 2012. С. 652–655.

[12] *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» ТПСА*. 2012. С. 17–25.