КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2012 Т. 4 № 4 С. 871–883

Ки&М

МОДЕЛИ В ФИЗИКЕ И ТЕХНОЛОГИИ

УДК: 519.226

Использование синтаксических деревьев для автоматизации коррекции документов в формате LaTeX

К. В. Чувилин

Московский физико-технический институт (ГУ),
141700, Московская область, г. Долгопрудный, Институтский переулок, 9.

E-mail: kirill.chuvilin@gmail.com

Получено 20 июля 2012 г.

Рассматривается задача автоматизации коррекции документов в формате LaTeX. Каждый документ представляется в виде синтаксического дерева. С помощью модифицированного алгоритма Zhang-Shasha строится отображение вершин дерева изначального документа в вершины дерева отредактированного документа, соответствующее минимальному редактирующему расстоянию. Отображения вершины в вершину составляют обучающую выборку, по которой генерируются правила замены для автоматической коррекции. Для каждого правила собирается статистика его применимости к отредактированным документам. На ее основе производится оценка качества правил и их улучшение.

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

The use of syntax trees in order to automate the correction of LaTeX documents

K. V. Chuvilin

Moscow Institute of Physics and Technology (SU), 9 Institutskii per., Dolgoprudny, Moscow Region, 141700, Russia

Abstract. — The problem is to automate the correction of LaTeX documents. Each document is represented as a parse tree. The modified Zhang-Shasha algorithm is used to construct a mapping of tree vertices of the original document to the tree vertices of the edited document, which corresponds to the minimum editing distance. Vertex to vertex maps form the training set, which is used to generate rules for automatic correction. The statistics of the applicability to the edited documents is collected for each rule. It is used for quality assessment and improvement of the rules.

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

Citation: *Computer Research and Modeling*, 2012, vol. 4, no. 4, pp. 871–883 (Russian).

Введение

Многие научные 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 тексты, присылаемые авторами, содержат значительное количество (десятки на страницу) таких ошибок, исправление которых производится корректорами вручную. Обработка одной страницы занимает до двух часов времени.

Предлагаемый подход направлен на значительное сокращение объёма рутинной работы. Корректор работает с системой, которая сама определяет в исходном тексте возможные места исправлений и предлагает ему вариант замены. Если он согласен с заменой, ему остаётся только нажать на соответствующую кнопку. Если не согласен, то он делает правку вручную.

Одним из стандартных способов описания правил изменения текстовых данных являются регулярные выражения [Фридл, 2008]. Однако они позволяют задать лишь вид окружающего текста, тогда как для описания корректорской правки часто требуется знать контекст рассматриваемого фрагмента в логической структуре документа, в том числе текущее состояние синтаксического анализатора LaTeX. Кроме того, регулярные выражения обладают некоторыми ограничениями. Например, невозможно описать структуру из парных скобок произвольной степени вложенности.

Файлы формата LaTeX обладают естественной древовидной структурой (*синтаксическим деревом*), исследуя которую, можно получить всю необходимую информацию для описания корректорской правки. Узлы этой структуры будем называть *токенами*. Корнем является окружение `document`. 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 взаимнооднозначно определяет документ LaTeX. Поэтому правила замены удобно формулировать именно для деревьев.

Правила замены можно задавать вручную, непосредственно на основе практического опыта корректоров. Однако ввиду значительно числа и разнообразия правил, это приведёт 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*. Соответствующие синтаксические деревья — *чистовыми* и *черновыми*.

В данной статье рассматривается задача автоматической генерации правил преобразования документов по обучающей выборке, составленной из пар «черновик-чистовик».

Редактирующее расстояние между деревьями

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

Определение (Редактирующее расстояние). *Редактирующим расстоянием* между двумя деревьями называется минимальное количество операций удаления вершины, вставки вершины и изменения ключа, позволяющих получить из первого дерева второе.

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 [Zhang, Shasha, 1989].

Определение (Наиболее левая вершина). В произвольном дереве каждой вершине можно сопоставить *наиболее левую для нее вершину*: каждой терминальной вершине — ее саму, любой другой — наиболее левую для самого левого ее потомка.

Определение (Ключевой корень). В произвольном дереве вершина, для которой наиболее левая вершина отличается от наиболее левой вершины для ее родителя называется *ключевым корнем*.

Определение (Обратная нумерация вершин). 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*.

Оказывается, что поддерево, образованное произвольным ключевым корнем, состоит из всех вершин, номера которых не превосходят номера корня, и только из них. Далее каждая вершина дерева будет обозначаться числом, равным ее номеру.

Определение (Отображение деревьев). Пусть заданы два дерева. *Отображением* первого дерева во второе называется правило, которое некоторым вершинам первого дерева взаимно однозначно сопоставляет некоторые вершины второго дерева так, чтобы порядок следования вершин сохранялся. 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 такое отображение соответствует набору операций, используемых для построения редактирующего расстояния:

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

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

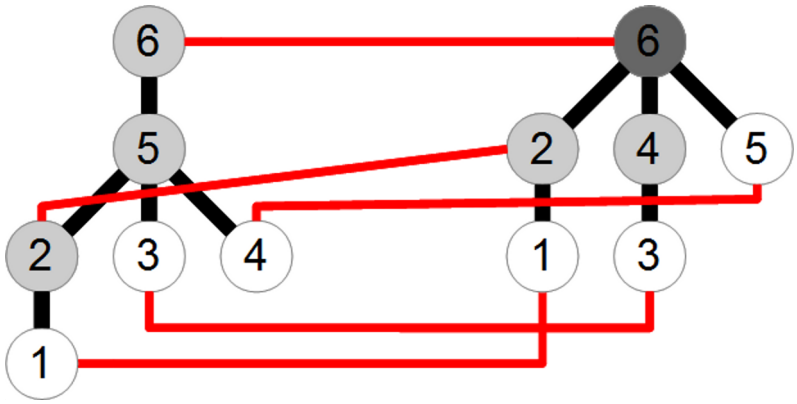

Рис. 1. Обратная нумерация и отображение деревьев

Пример 1. На рис. 1 показан пример обратной нумерации и отображения двух деревьев, соответствующего редактирующему расстоянию. Формально оно записывается следующим образом: $(1,1)$, $(2,2)$, $(3,3)$, $(4,5)$, $(6,6)$.

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

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

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

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

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

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

Если обозначить через $K^1 = \{k_1^1, \ldots, k_{m_1}^1\}$ и $K^2 = \{k_1^2, \ldots, k_{m_2}^2\}$ — упорядоченные по возрастанию наборы ключевых корней первого и второго деревьев соответственно, то основной цикл алгоритма запишется следующим образом.

for $i = k_1^1, \ldots, k_{m_1}^1$ **do**
  for $j = k_1^2, \ldots, k_{m_2}^2$ **do**
    `treeDist(`i, j`);`
  end for
end for

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 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 расстояний между лесами, где в ячейке (i, j) стоит расстояние между лесами, образованными удалением корней из соответствующих поддеревьев.

Пример 2. На рис. 2 показаны таблицы расстояний для деревьев из примера 1.

	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. 2. Таблицы расстояний между деревьями (а) и лесами (б)

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 измененных вершин, пары (прообраз и образ) измененных вершин, множество удаленных вершин, множество добавленных вершин.

Расстояние для синтаксических деревьев LaTeX

Токену каждого типа синтаксического дерева LaTeX можно сопоставить ключ по правилу, описанному в таблице 1.

После определения ключей синтаксические деревья полностью удовлетворяют условиям применимости алгоритма Zhang-Shasha. Но, как оказалось, правки, совершаемые корректорами, не всегда могут быть заданы тремя вышеописанными действиями.

Пример 3. На рис. 3 схематично показано отображение деревьев, которое должно возникнуть при замене фрагмента

```
$(k-1)/(8L^2),$
```

Таблица 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	`\textwidth=10cm`	`10cm`
Число	Значение числа	`высота 1,2\,м`	`1,2`
Разделитель абзацев	Тип токена	Абзац □ Новый абзац	`par`

Путь к файлу	Значение пути	`\includegraphics`
		`[width=10cm]`
		`{../figure.eps}`
			`../figure.eps`

Пробел	Тип токена	`высота□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`

на

```
$(k-1)/(8L^2)$,
```

В этом случае 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).

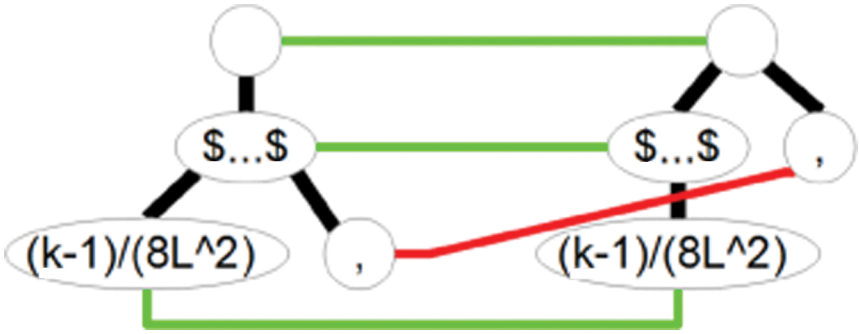

Рис. 3. Отображение, не задаваемое тремя видами операций

Модификация алгоритма Zhang-Shasha

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 удаленных 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 (Поднятые вершины). Вершины $x_1, \ldots, x_k$ чернового дерева такие, что для $i = 1, \ldots, k$ выполняется:

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

При этом существуют вершины $y_1, \ldots, y_k$ чистового дерева такие, что для $i = 1, \ldots, k$ выполняется:

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

Определение (Опущенные вершины). Вершины $x_1, \ldots, x_k$ чернового дерева такие, что для $i = 1, \ldots, k$ выполняется:

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

При этом существуют вершины $y_1, \ldots, y_k$ чистового дерева такие, что для $i = 1, \ldots, k$ выполняется:

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

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

Было доказано следующее утверждение.

Теорема. *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 $\varphi_1, \ldots, \varphi_n$ — последовательность операций с вершинами, реализующими какое-то отображение T_1 в T_2, то можно построить набор операций $\varphi'_1, \ldots, \varphi'_m$, реализующий выбранное отображение, такой, что $m \leqslant n$, 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 $j \leqslant i$. Путь 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 $\varphi_{j+1}, \ldots, \varphi_{i-1}$ 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 $x_1, \ldots, x_k$, причем для некоторого $q \in 1, \ldots, k$ $f(x_q) = t$. 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 $x_1, \ldots, x_{q-1}, x_{q+1}, \ldots, x_k$ будут удовлетворять условию для поднятия, то есть операции φ_j и φ_{j+1} можно заменить на операции удаления x_q и поднятия $x_1, \ldots, x_{q-1}, x_{q+1}, \ldots, x_k$.
- φ_j — опускание вершин $x_1, \ldots, x_k$, причем для некоторого $q \in 1, \ldots, k$ $f(x_q) = t$. 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 $x_1, \ldots, x_{q-1}, x_{q+1}, \ldots, x_k$ будут удовлетворять условию для опускания, то есть операции φ_j и φ_{j+1} можно заменить на операции удаления x_q и опускания $x_1, \ldots, x_{q-1}, x_{q+1}, \ldots, x_k$.

Таким образом можно избавиться от всех операций, работающих с t или прообразами 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, поэтому их можно менять местами с операциями изменения ключа. ■

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 удаленных (множество D), а их образы — среди добавленных (множество I).

Синтез правил коррекции

После получения отображения черновых и чистовых деревьев строится начальный набор правил коррекции [Чувилин, 2011б]. Каждое построенное правило характеризуется *шаблоном* (последовательностью соседних токенов с общим родителем) и типом *локализатора* (токена, к потомкам которого применяется шаблон).

Определение. Токен чернового дерева, к которому применяется одна из пяти вышеописанных операций в процессе перехода к чистовому дереву, называется *измененным*.

Определение. *Левая (правая) шаблонная цепочка радиуса r* — это набор последовательных токенов с общим родителем, длиной не больше r, такая, что если она содержит измененный токен, то только один, причем он самый левый (правый).

Для каждого типа отображения локализаторы и шаблоны строятся следующим образом.

Если токен x чернового дерева удален или у него изменен ключ, то локализатор — это $p(x)$, шаблон составляется из самого токена x, левой шаблонной цепочки радиуса 1, начинающейся в левом соседе x, если он существует, и аналогичной правой. Например, при преобразовании

```
\begin{center}
    казнить, нельзя, помиловать
\end{center}
```

в

```
\begin{center}
    казнить нельзя, помиловать
\end{center}
```

происходит удаление токена, соответствующего запятой. В данном случае локализатор — это тело окружения center (`center body`), левая шаблонная цепочка состоит из токена слова «казнить», правая — из пробела, то есть шаблон состоит из трех токенов, соответствующих слову «казнить», запятой и пробелу.

Если в чистовое дерево добавлен токен y, то локализатор — это прообраз $p(y)$, если он существует, шаблон составляется из левой шаблонной цепочки радиуса 1, начинающейся в прообразе левого соседа y, если он существует, и аналогичной правой. Например, при преобразовании

```
\begin{center}
    казнить нельзя помиловать
\end{center}
```

в

```
\begin{center}
    казнить нельзя, помиловать
\end{center}
```

происходит вставка токена, соответствующего запятой. В данном случае локализатор — это тело окружения center (`center body`), левая шаблонная цепочка состоит из токена слова «нельзя», правая — из пробела, то есть шаблон состоит из двух токенов, соответствующих слову «нельзя» и пробелу.

Если токены $x_1, \ldots, x_k$ чернового дерева были подняты, то локализатор — это $p(x_1)$, шаблон состоит из токенов $x_1, \ldots, x_k$. Например, при преобразовании

```
$a_i = b_i, если i нечетно$
```

в

```
$a_i = b_i$, если i нечетно
```

происходит поднятие токенов. В данном случае локализатор — это парный символ `$...$`, шаблон состоит из семи токенов, соответствующих запятой, пробелу, слову «если», пробелу, слову «i», пробелу и слову «нечетно».

Если токены $x_1, \ldots, x_k$ чернового дерева были опущены, то локализатор — это $p(x_1)$, шаблон состоит из токенов $x_1, \ldots, x_k$ и токена $x_0 = x_1 - 1$ (всегда существует). Например, при преобразовании

```
\begin{document}
    \begin{theorem}
        Эта теорема справедлива.
    \end{theorem}
    \note Почти всегда.
\end{document}
```

в

```
\begin{document}
    \begin{theorem}
        Эта теорема справедлива.
        \note Почти всегда.
    \end{theorem}
\end{document}
```

происходит опускание токенов. В данном случае локализатор — это тело окружения document (`document body`), шаблон состоит из семи токенов, соответствующих команде `\note`, пробелу, слову «Почти», пробелу, слову «всегда», точке и пробелу.

Сразу после того, как для какой-либо операции найден шаблон, эта операция применяется к дереву. Дальнейший поиск происходит в уже измененном дереве.

Использование правил

Для каждого токена синтаксического дерева документа выбираются шаблоны, тип локализаторов которых совпадает с типом рассматриваемого токена. Среди потомков токена ищется цепочка соседних, совпадающая с шаблоном по следующим правилам:

- для всех измененных токенов из шаблона соответствующие потомки должны иметь такие же ключи,
- для всех остальных токенов из шаблона соответствующие потомки должны иметь такие же типы.

Сразу после того, как совпадение найдено, выполняется соответствующая операция. Дальнейший поиск происходит в уже измененном дереве.

Подобный подход позволяет объединять правила, в шаблонах которых совпадают ключи измененных токенов и типы неизмененных токенов. Далее в статье подобные правила считаются одним.

Улучшение правил

После синтеза начального набора правил для каждого производится оценка качества с помощью статистики применимости для чистовиков [Чувилин, 2011a]: исследуется соответствие чистовиков шаблонам правил, каждый случай применимости считается отрицательным прецедентом для правила.

Определение (Порог ошибочности правил). Максимальное количество отрицательных прецедентов, при котором считается, что правило *состоятельно*, то есть не требует улучшения.

Если количество отрицательных прецедентов превышает порог ошибочности, правило заменяется новыми, которые получаются из исходного следующим образом:

- если токен x чернового дерева удален или у него изменен ключ, то радиус шаблонных цепочек увеличивается на 1;
- если в чистовое дерево добавлен токен y, то радиус шаблонных цепочек увеличивается на 1;
- если токены $x_1, \ldots, x_k$ чернового дерева были подняты, то к шаблону добавляется левая шаблонная цепочка радиуса 1, начинающаяся в левом соседе x_1, если он есть; если левая цепочка уже была добавлена, то ее радиус увеличивается на 1.

Для всех новых правил производится оценка качества.

Эксперимент

В качестве данных использовались 85 пар черновиков и чистовиков, вошедших в сборник трудов конференции «Интеллектуализация обработки информации» [ИОИ-2010]. Для каждого количества пар статей, используемых для обучения, 30 раз случайным образом генерировалась выборка. Определялось количество синтезированных правил и количество соответствий правилам в черновиках и чистовиках.

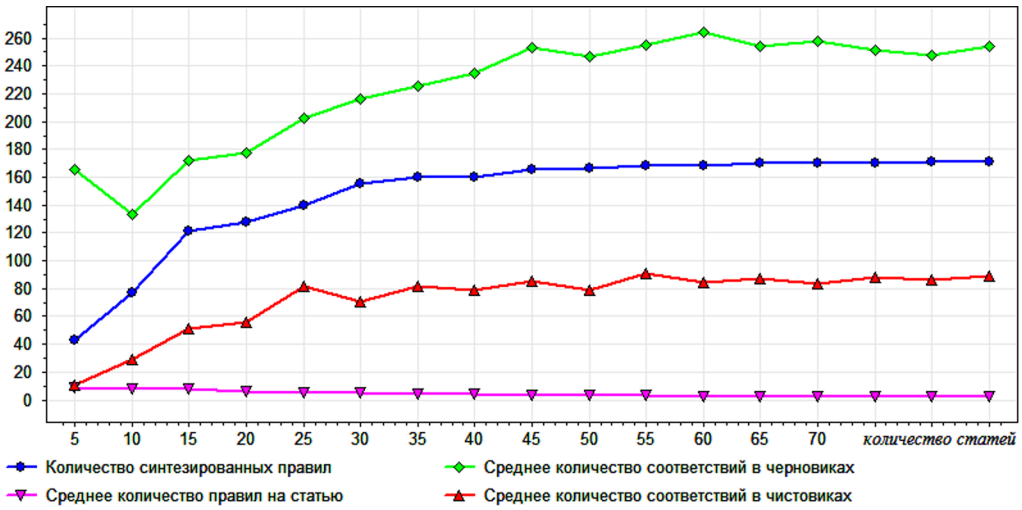

Рис. 4. Синтезированные правила и найденные соответствия

На рис. 4 для каждого количества пар документов, используемых для обучения, отображены усредненные по всем сгенерированным выборкам количества синтезированных правил

и найденных соответствий. Среднее количество правил на статью стремится к нулю, это означает, что почти все типовые правила можно синтезировать, используя не очень большое (около 75) число статей. Среднее количество соответствий в чистовиках соответствует неверным или слишком общим шаблонам. При достаточно большой обучающей выборке оно составляет около 80, что не очень много, поскольку среднее количество всех найденных правил превышает 240 в таких случаях. Тем не менее остается проблема интерпретации большого количества найденных синтезированных правил.

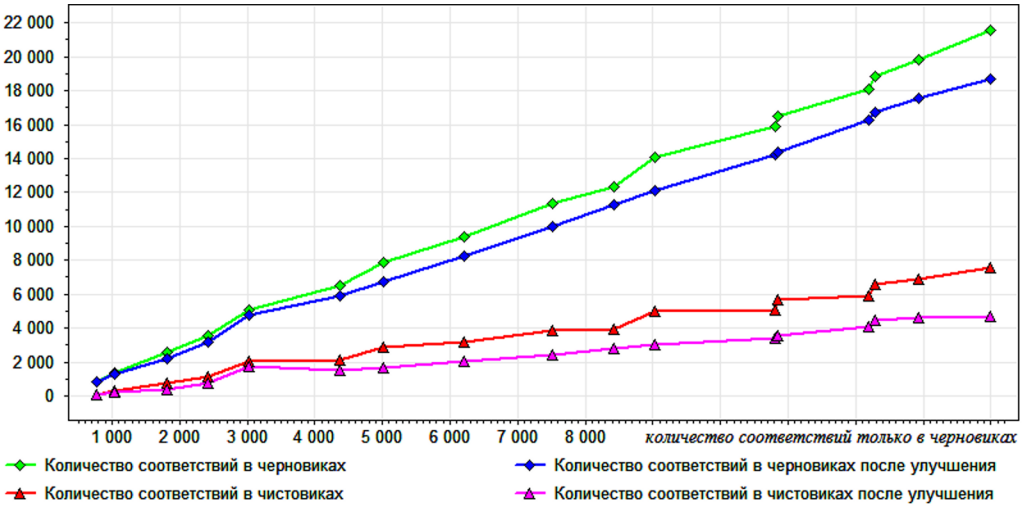

Рис. 5. Количество соответствий до улучшения и после

На рис. 5 показано количество соответствий в статьях исходным правилам и улучшенным в зависимости от количества соответствий только в черновиках, то есть верно определенных мест типографических ошибок. Видно, что количество соответствий для черновиков и чистовиков уменьшилось. Это означает, что шаблоны стали более точными.

Заключение

Предложен подход, который позволяет синтезировать правила коррекции документов в формате LaTeX. Из результатов экспериментов можно видеть, что построенные правила позволяют автоматически обнаруживать типографические ошибки. Но при этом наблюдается большое количество ложных срабатываний. Анализ построенных шаблонов показал, что проблемы обсусловлены следующими факторами: чистовики, полученные после обработки корректорами, могут содержать типографические ошибки, существуют правила, которые не могут быть изменены предложенным способом, требуются правила, модифицирующие несколько токенов. Это задает направление дальнейшим исследованиям.

Рассмотренный подход может быть применим в других задачах, связанных с древовидными структурами данных, поскольку используемые алгоритмы не связаны с особенностями формата документов LaTeX.

Список литературы

Доклады восьмой международной конференции «Интеллектуализация обработки информации». — ИОИ-2010.

Фридл Д. Регулярные выражения / Пер. с англ.— 3-е издание. — СПб.: Символ-Плюс, 2008.

Чувилин К. В. Автоматический синтез и статистическая оценка качества правил коррекции документов в формате LaTeX // Труды 54-й научной конференции МФТИ. Том 2 «Управление и прикладная математика». — 2011. — С. 106–107.

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. — 2011. — С. 597–600.

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