КОМПЬЮТЕРНЫЕ ИССЛЕДОВАНИЯ
И МОДЕЛИРОВАНИЕ 2015 Т. 7 № 2 С. 329–345

Ки&М

МОДЕЛИ В ФИЗИКЕ И ТЕХНОЛОГИИ

УДК: 519.226

Эффективный алгоритм сравнения документов в формате LaTeX

К. В. Чувилин

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

E-mail: kirill.chuvilin@gmail.com

Получено 16 июля 2013 г.,
после доработки 04 февраля 2015 г.

Рассматривается задача построения различий, возникающих при редактировании документов в формате LaTeX. Каждый документ представляется в виде синтаксического дерева, узлы которого называются токенами. Строится минимально возможное текстовое представление документа, не меняющее синтаксическое дерево. Весь текст разбивается на фрагменты, границы которых соответствуют токенам. С помощью алгоритма Хиршберга строится отображение последовательности текстовых фрагментов изначального документа в аналогичную последовательность отредактированного документа, соответствующее минимальному редактирующему расстоянию. Строится отображение символов текстов, соответствующее отображению последовательностей текстовых фрагментов. В синтаксических деревьях выделяются токены такие, что символы соответствующих фрагментов текста при отображении либо все не меняются, либо все удаляются, либо все добавляются. Для деревьев, образованных остальными токенами, строится отображение с помощью алгоритма Zhang–Shasha.

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

An efficient algorithm for LaTeX documents comparing

K. V. Chuvilin

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

Abstract. — The problem is constructing the differences that arise on LaTeX documents editing. Each document is represented as a parse tree whose nodes are called tokens. The smallest possible text representation of the document that does not change the syntax tree is constructed. All of the text is splitted into fragments whose boundaries correspond to tokens. A map of the initial text fragment sequence to the similar sequence of the edited document corresponding to the minimum distance is built with Hirschberg algorithm A map of text characters corresponding to the text fragment sequences map is cunstructed. Tokens, that chars are all deleted, or all inserted, or all not changed, are selected in the parse trees. The map for the trees formed with other tokens is built using Zhang–Shasha algorithm.

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

Citation: *Computer Research and Modeling*, 2015, vol. 7, no. 2, pp. 329–345 (Russian).

Введение

Многие научные конференции и издательства принимают материалы от авторов в формате LaTeX [Львовский, 2006]. В каждом издательстве есть определенные традиции и требования к оформлению публикуемого материала [http://jmlda.org/papers/index.php/JMLDA/about/submissions#authorGuidelines; http://crm.ics.org.ru/journal/page/avtors/; http://mmro.ru/reports.php; http://lomonosov-msu.ru/rus/lom_13_rules.html]. 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, исправление которых производится корректорами вручную. Обработка одной страницы занимает до двух часов времени.

В работах [Чувилин, 2013а; Чувилин, 2012] рассматривается задача автоматической генерации правил преобразования документов как задача обучения по прецедентам [Воронцов, http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций; Anderson et al., 1983; Michalski et al., 1986]. Предлагается разделить исследуемую задачу на три последовательных шага.

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

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

Файлы формата LaTeX, используемые при подготовке научных изданий (книг и сборников трудов), как правило, обладают естественной древовидной структурой (*синтаксическим деревом*), 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, нераспознаваемая последовательность символов (например, для окружения `verbatim`). Синтаксическое дерево взаимно-однозначно (с точностью компилятора TeX) определяет документ LaTeX. Правила коррекции удобно формулировать именно для деревьев.

В работе [Чувилин, 2011] для построения различий между синтаксическими деревьями используется алгоритм, основанный на алгоритме Zhang–Shasha [Zhang, Shasha, 1989]. 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 четырех страниц (распространенный 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 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 [Anderson et al., 1983; Hirschberg, 1975]. 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 статье рассматривается гибридный алгоритм сравнения документов в формате LaTeX, использующий достоинства алгоритмов сравнения неформатированных текстов и синтаксических деревьев и позволяющий сравнительно быстро выявлять различия, учитывающие логическую структуру, даже для больших документов [Чувилин, 2013b].

Выделение различий между конечными последовательностями

Мера различия (*редактирующее расстояние*) между конечными последовательностями элементов, включая последовательности символов, которыми являются тексты, задается расстоянием Левенштейна [Левенштейн, 1965; Гасфилд, 2003].

Определение. Пусть для изменения последовательности элементов разрешается 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:

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

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, разделенное 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.

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

Алгоритм Вагнера–Фишера

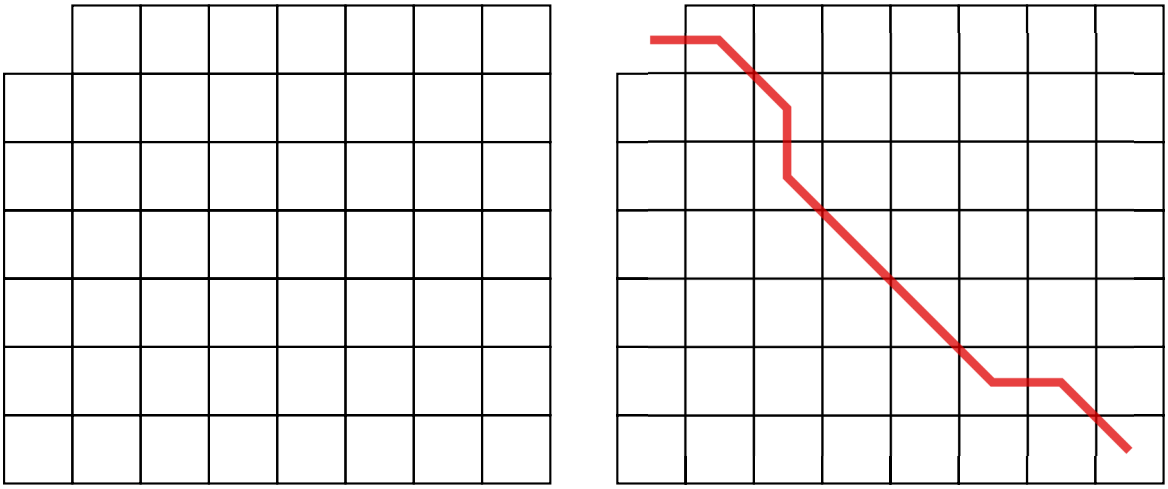
Наивный подход заключается в выполнении двух шагов:

1) построение матрицы расстояний между всевозможными парами префиксов методами динамического программирования [Беллман, 1960];
2) восстановление редакционного предписания с помощью пошагового анализа возможных обратных переходов.

Так, например, устроены алгоритмы Вагнера–Фишера [Wagner, Fischer, 1974] и Нидлмана–Вунша [Needleman, Wunsch, 1970].

Пример 1. Рассмотрим применение вычисление редактирующего расстояния и отображения с помощью алгоритма Вагнера–Фишера в том случае, когда из слова «ЛОГИКА» получается слово «АЛГЕБРА».

На рисунке 1, *а* представлена матрица L, построенная динамически по рекуррентным формулам. В каждой ее клетке находится значение редактирующего расстояния для соответствующих префиксов, в правой нижней клетке — редактирующее расстояние для сравниваемых слов.



(*а*) (*б*)

Рис. 1. Пример матрицы редактирующих расстояний для префиксов слов (*а*). Восстановление последовательности операций (*б*)

На рисунке 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: $(L(i, j) = L(i-1, j-1)+1)$ — символ изменен; $(L(i, j) = L(i-1, j-1))$ — символ не изменен.

Это позволяет построить отображение, представленное на рисунке 2 (D означает, что символ удален, I — добавлен, C — изменен).

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

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

Рис. 2. Отображение, восстановленное по матрице расстояний

Алгоритм Хиршберга

Наиболее эффективным с точки зрения количества потребляемой памяти является алгоритм Хиршберга [Hirschberg, 1975].

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

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

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

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 удаленными.

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 — удаленными. Если таких элементов нет, то a_1 считается прообразом b_1, а остальные — удаленными.

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 $i = [\frac{n}{2}]$. 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.

Применение для документов в формате LaTeX

Документы в формате 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 в формате LaTeX), что тексты содержат очень большое количество символов и последовательности получаются чрезмерное 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 2. Утилиты на основе алгоритма diff [Miller, Myers, 1985; Ukkonen, 1985] и его вариаций являются наиболее распространенными реализациями методов сравнения текстовых данных. На сайте *Diff Checker* [http://www.diffchecker.com/diff] можно проверить этот алгоритм в режиме «онлайн». На рисунке 3 представлены некоторые результаты его работы для пар документов в формате LaTeX.

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

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

Построение различий 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* (обратная удалению операция), *изменение ключа* вершины.

Определение. *Редактирующим расстоянием* между двумя деревьями называется минимальное количество операций удаления, вставки и изменения ключа вершин, 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].

Рис. 3. Примеры работы алгоритма diff для документов в формате LaTeX

Определение. 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.

Пример 3. На рисунке 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.

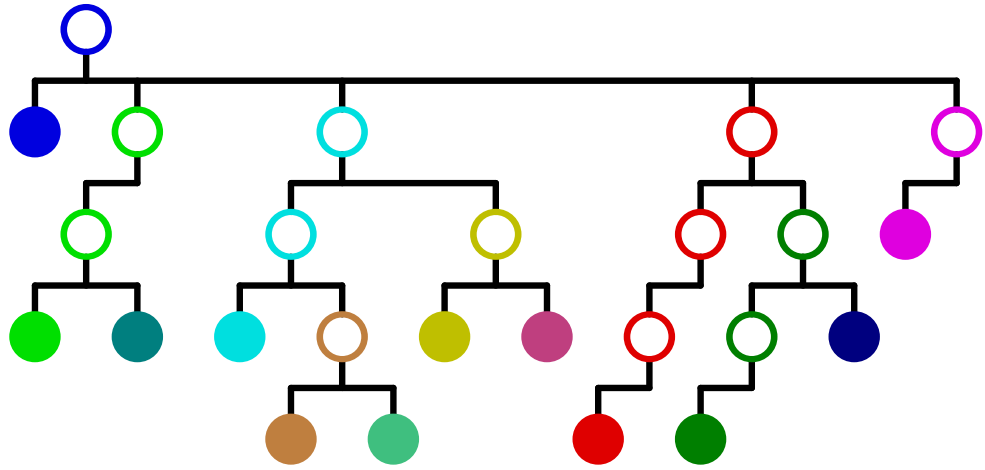


Рис. 4. Наиболее левые вершины дерева

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

Пример 4. На рисунке 5 показан пример графа с выделенными ключевыми корнями: все вершины с контуром одинакового цвета имеют общий ключевой корень, который закрашен в тот же цвет.

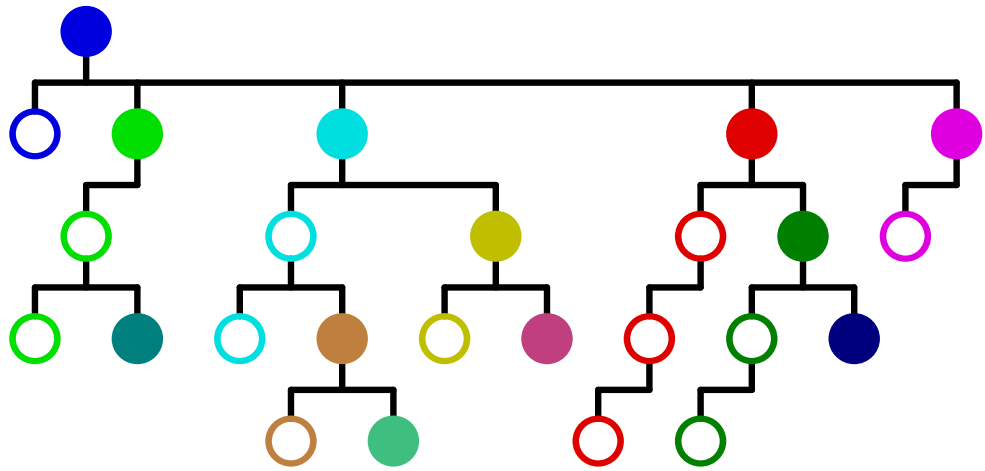


Рис. 5. Ключевые корни дерева

Таким образом, любое дерево разбивается на линейные последовательности вершин, начинающиеся в ключевых корнях и заканчивающиеся в соответствующих наиболее левых вершинах.

На рисунках 4 и 5 каждая такая последовательность образуется всеми вершинами с одинаковым цветом контура.

Определение. 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 5. На рисунке 6 показан пример обратной нумерации дерева из шести вершин.

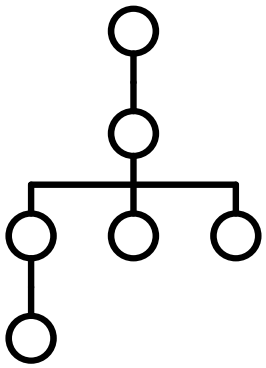


Рис. 6. Обратная нумерация вершин дерева

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. 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 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 отображение может быть описано набором операций:

- если вершина первого дерева не имеет образа, то ее нужно удалить;
- если вершина второго дерева не имеет прообраза, то ее 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.

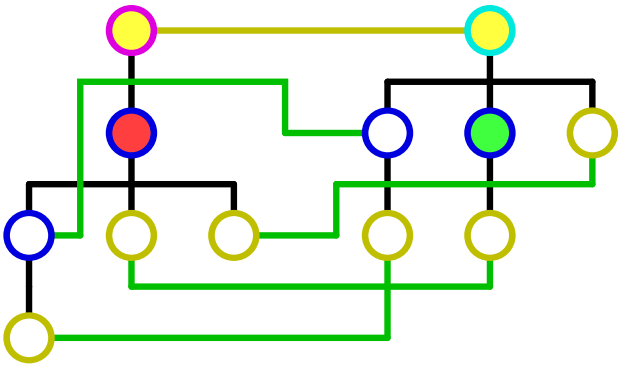


Рис. 7. Отображение деревьев

Пример 6. На рисунке 7 показан пример 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 $\bigcirc\!\!\!\!\bigwedge$ и $\bullet\!\!\!\!\bigwedge$ обозначают деревья с корнями $\circ$ и $\bullet$ соответственно; $\bigwedge\!\!\bigwedge$, $\blacktriangle\!\!\blacktriangle$ — леса, образованные удалением корней этих деревьев; $\triangle\!\!\triangle$ и $\blacktriangle\!\!\blacktriangle$ — произвольные леса.

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

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

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:

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

Обозначим: n_1 и n_2 — количества вершин, m_1 и m_2 — количества ключевых корней, $K^1 = \{k^1_1, \ldots, k^1_{m_1}\}$ и $K^2 = \{k^2_1, \ldots, k^2_{m_2}\}$ — упорядоченные по возрастанию наборы ключевых корней первого и второго деревьев соответственно.

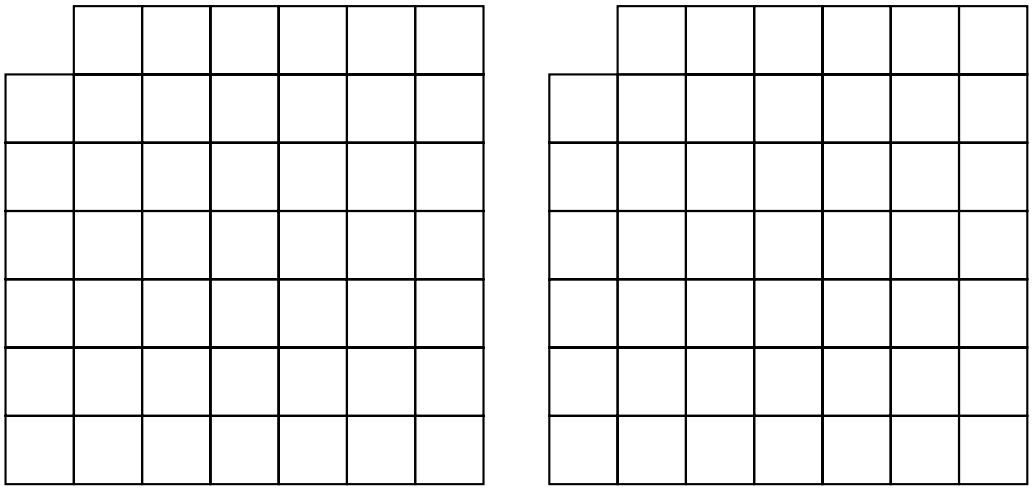
Основной цикл алгоритма запишется следующим образом.

for $i = k^1_1, \ldots, k^1_{m_1}$ **do**
  for $j = k^2_1, \ldots, k^2_{m_2}$ **do**
    `treeDist(`i, j`);`
  end for
end for

Здесь функция `treeDist(`i, j`)` вычисляет расстояние между поддеревьями первого и второго деревьев 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-той 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.



(a) ($б$)

Рис. 8. Матрицы расстояний между деревьями (a) и лесами ($б$)

Пример 7. На рисунке 8 показаны матрицы расстояний для деревьев из примера 6.

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.

Применение 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.

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 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`

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 [Zhang, Shasha, 1989] Zhang и Shasha ссылаются на более ранние исследования других авторов [Hoffmann, O'Donnell, 1982; Shellers, 1980; Shapiro, 1988; Sussman, Kim, 1976; Tai, 1979; Zhang, 1983; Zhang, 1989], 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 $O(n_1 \times n_2)$: недостаточно 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 в формате LaTeX имеют тысячи ключевых корней, поэтому скорость алгоритма невысокая.

Подход, применяемый в данной работе, нацелен на устранение подобных недостатков с помощью уменьшения количества вершин деревьев, которые необходимо сравнивать.

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 *упрощенные 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.

Разбиение текста

В первую очередь построим линейные последовательности фрагментов текста сравниваемых документов.

Каждому токену 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.

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 цены изменения одного фрагмента текста на другой будем использовать относительное расстояние Левенштейна для последовательностей символов, образующих эти фрагменты.

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 несовпадающих образа и прообраза построим отображение символов с помощью алгоритма Хиршберга, рассматривая два этих фрагмента текста как две линейные последовательности символов.

Отображение токенов

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

Определение. 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 пусты), взаимнооднозначно 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.

Структура алгоритма

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

1) построить упрощенные 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 удаленные, добавленные и неизмененные токены;

6) построить отображение деревьев, образованных остальными токенами.

Пример 8. Рассмотрим этапы сравнения двух синтаксических деревьев, изображенных на рисунке 9, *а*. Шагу 4 гибридного алгоритма соответствует состояние на рисунке 9, *б*. После шага 5 остаются два дерева, изображенные на рисунке 9, *в*. Результату их сравнения с помощью алгоритма Zhang–Shasha, получаемому на шаге 6, соответствует отображение на рисунке 9, *г*.

Таблица 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

Эксперимент

Для исследования работы алгоритма использовались 85 пар черновиков и чистовиков, вошедших в сборник трудов восьмой конференции «Интеллектуализация обработки информации» [Труды..., 2010]. Статистические характеристики документов приведены в таблице 2.

В таблице 3 показаны статистические данные деревьев, которые остаются после выделения удаляемых, добавляемых и неизменяемых 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. Количественные характеристики деревьев, получаемых во время работы гибридного алгоритма

	Токены	Ключевые корни
Минимум	346	297
Максимум	1689	1325
В среднем	1011.6828	826.08

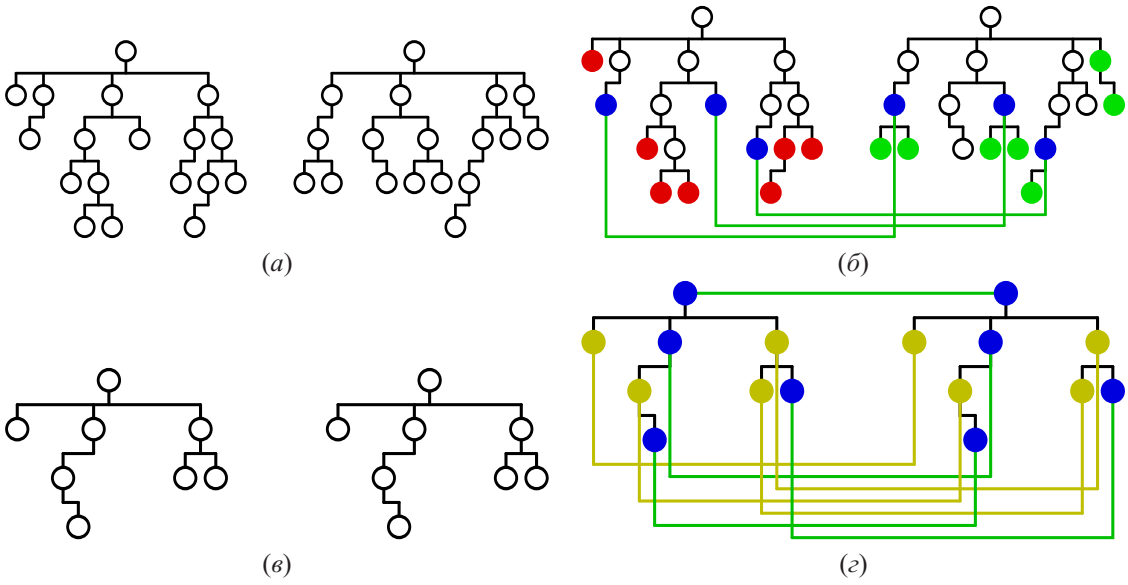


Рис. 9. Этапы построения отображения деревьев гибридным алгоритмом: исходные деревья (*а*); выделены удаленные, добавленные и неизмененные токены (*б*); оставлены только неизвестные токены (*в*); построено отображение для оставшихся токенов с помощью алгоритма Zhang–Shasha (*г*)

В таблице 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 4. Точность и полнота гибридного алгоритма

	Точность	Полнота
Минимум	0.75	0.78
Максимум	1	1
В среднем	0.91	0.91

Заключение

Предложен алгоритм, который позволяет эффективно выявлять различия документов в формате LaTeX. Из результатов экспериментов можно видеть, что такой подход позволяет заметно уменьшить требования к памяти и времени работы. Тем не менее получаемые этим способом отображения не полностью соответствуют результатам алгоритма, сравнивающего только синтаксические деревья. Это задает направление дальнейшим исследованиям.

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

Список литературы

Беллман Р. Динамическое программирование. — М.: Изд-во иностранной литературы, 1960.

Воронцов К. В. Математические методы обучения по прецедентам (теория обучения машин). `http://www.machinelearning.ru/wiki/index.php?title=Машинное_обучение_(курс_лекций,_К.В.Воронцов)`.

Гасфилд Д. Строки, деревья и последовательности в алгоритмах. Информатика и вычислительная биология. — Невский Диалект, БХВ-Петербург, 2003.

Левенштейн В. И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Доклады Академий наук СССР. — 1965. — С. 845–848.

Львовский С. М. Набор и верстка в системе LaTeX. — М.: МЦНМО, 2006.

Труды VIII Международной конференции «Интеллектуализация обработки информации». — Москва: МАКС Пресс, 2010.

Чувилин К. В. Синтез правил коррекции документов в формате LaTeX с помощью сопоставления синтаксических деревьев // Труды XV Всероссийской конференции «Математические методы распознавания образов». — М.: МАКС Пресс, 2011. — С. 597–600.

Чувилин К. В. Использование синтаксических деревьев для автоматизации коррекции документов в формате LaTeX // Компьютерные исследования и моделирование. — 2012. — Т. 4, № 54. — С. 871–883.

Чувилин К. В. Автоматический синтез правил коррекции текстовых документов формата LaTeX: Ph.D. thesis / Диссертационный совет Д 002.017.02 при Федеральном государственном бюджетном учреждении «Вычислительный центр им. А. А. Дородницына Российской академии наук». — 2013a.

Чувилин К. В. Гибридный алгоритм сравнения документов в формате LaTeX // Прикладная информатика. — 2013b. — № 4. — С. 56–64.

Компьютерные исследования и моделирование: Для авторов.
`http://crm.ics.org.ru/journal/page/avtors/`.

Математические методы распознавания образов: Правила оформления докладов.
`http://mmro.ru/reports.php`.

Машинное обучение и анализ данных: Указания для авторов.
`http://jmlda.org/papers/index.php/JMLDA/about/submissions\#authorGuidelines`.

Международная научная конференция студентов, аспирантов и молодых ученых «Ломоносов-2013»: Требования к оформлению тезисов.
`http://lomonosov-msu.ru/rus/lom\_13\_rules.html`.

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.

Diff Checker — Online diff tool to find the difference between two text files.
`http://www.diffchecker.com/diff`.

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.

Hoffmann C. M., O'Donnell M. J. Pattern matching in trees // J. Assoc. Comput. Mach. — 1982. — Vol. 29. — P. 68–95.

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.

Miller W., Myers E. W. A File Comparison Program // Software — Practice and Experience. — 1985. — no. 15. — P. 1025–1040.

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.

Shapiro B. A. An algorithm for comparing multiple RNA secondary structures // Comput. Appl. Biosci. — 1988. — P. 387–393.

Shellers P. H. The theory and computation of evolutionary distances // J. Algorithms. — 1980. — P. 359–373.

Sussman J. L., Kim S. H. Three dimensional structure of a transfer RNA in two crystal forms // Science. — 1976. — P. 853.

Tai K.-C. The tree-to-tree correction problem // J. Assoc. Comput. Mach. — 1979. — Vol. 26. — P. 422–433.

Ukkonen E. Algorithms for Approximate String Matching // Information and Control. — 1985. — P. 100–118.

Wagner R. A., Fischer M. J. The string-to-string correction problem // J. ACM. — 1974. — Vol. 21, no. 1. — P. 168–173.

Zhang K. An algorithm for computing similarity of trees, Tech. Report, Mathematics Department, Peking University, Peking, China. — 1983.

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.

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.