Иерархический алгоритм diff при работе со сложными документами*

Д. В. Луцив
dluciv@math.spbu.ru

Д. В. Кознов
dkoznov@yandex.ru

В. С. Андреев
Andreev@docsvision.com
Digital Design, Санкт-Петербург

Широко известны алгоритмы сравнения двух информационных активов («плоских» файлов XML-файлов, Word-документов, различных моделей и т. д.). Однако бурное развитие информационных приложений для поддержки офисной деятельности ставит новые задачи перед разработчиками программных средств в этой области. В частности, во многих задачах по созданию, редактированию и проверке документов могут использоваться различные вариации алгоритма сравнения двух документов. Однако для того, чтобы пользователи получили полноценные инструменты, существующие алгоритмы сравнения должны быть доработаны в следующих направлениях — иметь возможность рекурсивно, но по-разному на каждом уровне (так называемая *иерархическая неоднородность*), использовать иерархическую визуализацию результатов сравнения, поддерживать сравнение документов «нечёткими» способами. Более того, целесообразно говорить не об отдельном алгоритме, а о платформе, которая поддерживает целое семейство различных базовых алгоритмов, позволяя их реализовывать в виде конечных и удобных целевых сервисов. Данная статья посвящена обсуждению требований к алгоритмам сравнения (diff) в контексте их применения к офисным задачам.

Ключевые слова: diff, merge, офисная документация, сравнение

* Работа выполнена при частичной поддержке компании Digital Design, а также РФФИ (грант № 12-01-00415-а).

файлов, сравнение деревьев, сравнение XML-файлов, нечеткое сравнение документов.

Введение

Алгоритм поиска разницы двух файлов (diff) — в общем случае двух произвольных информационных активов — является широко известным как среди пользователей, так и среди разработчиков ПО. Сравниваются документы (соответствующий алгоритм реализован, например, в MS Office), файлы с текстами программ (например, средства контроля версий CVS и Subversion, утилита UNIX diff) и т. д. Алгоритм diff используется в качестве составной части алгоритмов слияния (merge) двух разошедшихся версий файлов в системах версионного контроля, при синхронизации данных в системах с ограничениями на трафик, при объединении серверных и клиентских данных (например, в случае обрыва соединения) и т. д.

Однако при всей своей известности этот алгоритм имеет ограничения на применение, главное из которых — удобная визуализация результатов. Например, если попробовать воспользоваться функцией diff для сравнения Word-документов, встроенной в MS Office, то если сравниваемые документы достаточно большие (10 страниц и более), а сделанных изменений довольно много, то воспринять такую разницу оказывается очень сложно — пользователь видит огромное количество подсвеченной информации, среди которой и удалённые абзацы, и различия в правилах пунктуации. С другой стороны, очевидно, что в случае с офисными документами очень часто приходится сравнивать разошедшиеся версии, которые имеют одинаковую структуру — разделы, подразделы и т. д., — и эту структуру можно было бы использовать при визуализации результатов сравнения. Также возникает потребность сравнивать не только документы, но и пакеты документов (например, договор может содержать много приложений, которые хранятся в отдельных файлах), т. е. применять алгоритм сравнения рекурсивно, но на разных уровнях (при сравнении папок, файлов, отдельных разделов) алгоритм должен работать по-разному! И, наконец, такой алгоритм должен уметь сравнивать тексты «нечётко», с точностью до пробелов, пустых строк, разных названий сходных документов и т. д. В области документооборота существует большое количество

задач, где такой алгоритм мог быть востребован, и различные целевые пользовательские сервисы можно было бы создавать на его основе.

В данной статье описываются задачи, возникающие при офисной работе с документами, где может быть применён алгоритм diff. Также формулируются требования к иерархическому алгоритму сравнения документов, умеющему: (1) работать рекурсивно, но по-разному на каждом уровне, (2) иерархически визуализировать результаты сравнения, (3) сравнивать тексты «нечёткими» способами.

1. Обзор

Задачи автоматизированного сравнения структурированных и «плоских» текстов актуальны уже на протяжении полувека.

На начальном этапе в 1960–1970 годах исследовались, в основном, теоретические особенности задач сравнения строк и текстов. Так, например, важный результат, касающийся критериев оценки степени схожести текстов (расстояние Левенштейна), был в 1965 году получен Левенштейном в [6], а один из популярных алгоритмов для его вычисления был предложен в 1974 году Вагнером и Фишером [29]. Существенная часть результатов в области «нечёткого» сравнения текстов до сих пор основывается на использовании расстояния Левенштейна. Смежной является задача «нечёткого» поиска подстрок в текстах — см. работы [10, 28].

С середины 70-х годов для работы с текстовыми файлами стали предлагаться алгоритмы *diff*, на базе которых были позже построены многочисленные реализации одноименных утилит. Одно из первых опубликованных описаний алгоритма сравнения файлов и вычисления минимальных различий между ними было сделано в 1976 году [14]. Более поздняя работа [19] содержит описание разных вариантов алгоритма поиска различий и примеры исходных текстов программ, реализующих эти алгоритмы. Алгоритм сравнения текстовых файлов, представленный в работе [20], был реализован в широко известной утилите UNIX diff.

Со временем, в дополнение к diff, появились стандартные утили-

ты *merge*[1] и *patch*[2]. Алгоритм diff можно назвать основным в этой тройке, так как он является необходимой составной частью двух других алгоритмов. Необходимо отметить, что diff/merge утилиты могут быть «двухразмерными», т. е. работать только с двумя файлами (т. е. которые сравниваются/сливаются), а могут — с тремя, добавляя ещё общего предка (если он доступен). В последнем случае качество работы алгоритмов существенно увеличивается, так как становится понятным происхождение различий[3].

Со второй половины 1980-х годов в обиход офисных работников вошли программные пакеты для работы с электронными документами. Соответственно, стали возникать и решаться задачи сравнения структурированных текстов, например, LaTeX-файлов [12]. То есть алгоритмы diff и merge стали ориентироваться на работу с деревьями. Развитие Интернета ещё больше обострило интерес к этим задачам, так как возникли многочисленные практические ситуации, требующие поддержки совместной сетевой работы над документами различного вида. Основным форматом обмена данными по сети стал XML, поэтому появились многочисленные diff/merge алгоритмы для работы с XML-файлами [22, 25, 27]. Эти алгоритмы адаптировались для разных задач, например, применение 3DM-алгоритма [16] слияния XML-файлов для коллективной сетевой работы с картами памяти (mind maps) [2, 4, 5], а также задачу слияния онтологий и баз данных [1, 7, 23]. Отметим, что при решении задачи коллективной работы в сети с одним документом

стали развиваться online-аналоги алгоритма merge [21]. То есть подобные подходы осуществляют немедленную синхронизацию всех копий документа у клиентов, как только один из них изменит хоть что-то. Аналогичный подход был реализован и в средстве Comapping [18]. По сравнению с такими подходами merge является отложенной синхронизацией.

Следует отметить заметный интерес к сравнению документов

[1]Например, эта функция слияния файлов является одной из опций утилиты GNU diff3.

[2]В качестве примера можно привести известную Unix-утилиту, разработанную Larry Wall в 1985 году для объединения различий между версиями двух файлов, найденными с помощью утилиты diff.

[3]В частности, упоминаемая выше diff/merge утилита GNU является «трехразмерной».

по их внешнему виду [8, 9]. Кроме того, достаточно много работ посвящено сравнению текстов с использованием анализа естественного языка [14, 17, 24].

Однако все эти результаты трудно непосредственно применить к задачам сравнения офисных документов. Так, в частности, упомянутые работы по нечёткому сравнению ориентированы на анализ документов, полученных оцифровкой и поиском в Интернете. Для этой области основными критериями являются релевантность поисковых выдач запросам и подтверждение приблизительного соответствия документов друг другу. Для офисных документов вопрос релевантности обычно не возникает, а сравнение требует исчерпывающего представления различий (с последующим анализом человеком), так как речь часто идёт о договорах, актах, технических заданиях, судебных решениях и прочих деловых документах, для которых решающим может оказаться отличие в одном слове. Отметим также, что многочисленные diff/merge алгоритмы обычно оставляют в стороне вопросы визуализации результатов сравнения древовидных структур, и кроме того не учитывают более сложные структуры, чем обычные деревья или файлы определённых форматов (XML, HTML, LaTeX и т. д.).

2. Задачи сравнения офисных документов

В данной статье мы оставим в стороне более широкий класс применимости алгоритмов diff (например, модельно-ориентированную разработку ПО, бизнес-моделирование и т. д.) и сконцентрируемся на работе с офисными документами.

В рамках одной компании или отдельного подразделения за годы работы создаётся огромное количество документов — разные договора, счета, тендерная документация и т. д. При этом постоянно возникает задача создания новых аналогичных документов, что обычно делается на основе уже существующих. Причём очень часто уже существует несколько версий той или иной документации – несколько договоров, несколько заявок на участие в тендере, несколько счетов, руководств пользователя и т. д. Все эти версии были созданы с помощью Copy/Paste — либо на основе первой версии, либо на основе друг друга. Соответственно, возникают многочисленные практические задачи, основанные на сравнении этих

документов — что изменилось, а что осталось прежним, что нужно исправить в новой версии, а что можно оставить в неизменном виде. При этом статус повторно используемой информации может быть очень высоким — например, мы ищем, какие статьи в новом договоре Б были взяты из договора А, и нам это очень важно, так как договор А был тщательно проверен юристами, следовательно, статьи из него, попавшие в договор Б, проверять не нужно. Статус отличий тоже может быть очень высоким — сравнивая два типовых счёта мы хотим видеть, что в них отличается, и захватывают ли эти отличия информацию, которая должна быть разной (дата, основание платежа, сумма и т. д.), а также нет ли лишних отличий — например, название организации написано по-другому (и, значит, не правильно). При этом подобные сервисы могут использоваться как при разработке документов, так и при их проверке различными проверяющими инстанциями при согласовании.

Такие сервисы могут быть интегрированы, с одной стороны, с системами документооборота, бухгалтерского учёта, ERP-системами и пр. — т. е. со средствами, управляющими движением документов. С другой стороны, рассматриваемые сервисы нужны на отдельных рабочих местах, и поэтому они должны быть также интегрированы со средствами редактирования документов — в первую очередь, с продуктами MS Office.

Очевидно, что во всех таких задачах, которых на самом деле значительно больше, ключевую роль играют алгоритмы сравнения документов. При этом важным оказывается устранение излишней точности сравнения (пробелы, пустые строки и пр.), а также удобная визуализация результатов. Родственный сравнению алгоритм слияния (merge) оказывается здесь не востребован, так как не идёт речи о коллективной разработке документов. Также важным оказывается сравнение не только единичных документов, а целых папок, при этом хотелось бы, чтобы сравнение от папок «проникало» в отдельные файлы и в отдельные разделы этих файлов.

3. Требования к иерархическому diff

Остановимся на следующих требованиях к алгоритму сравнения офисных документов — неоднородная иерархичность, иерархическая визуализация, нечёткое сравнение.

3.1. Неоднородная иерархичность

При работе с офисными документами можно выделить три уровня сравнения: сравнение состава двух пакетов документов, сравнение структуры двух файлов, сравнение текстовых фрагментов двух файлов, которые принадлежат одинаковым элементам структуры (проще говоря, одному и тому же разделу). Переход от отдельных файлов к пакетам связан с тем, что именно папка с файлами (и, возможно, с вложенными папками) является семантической единицей нашей предметной области — договор может содержать файл собственно с договором, а также много дополнительных файлов с различными приложениями, тендерная документация также может быть разбита на многочисленные формы-файлы и приложения и т. д. Во многих задачах важно предоставить пользователю удобный инструмент для нахождения разницы в смысле состава файлов. При этом возникает задача идентификации сходных файлов, являющая частным случаем сравниваемых/сливаемых (mapping/-matching) позиций в задачах слияния XML-файлов, онтологий и баз данных. Подробнее этот аспект мы будем обсуждать в разделе 3.3.

После того, как алгоритм нашёл сходства и различия на уровне файлов, он должен начать сравнивать между собой отдельные файлы, которые он определил как сходные. Здесь нам кажется, нужно переходить не к простому сравнению текстов (или XML-представлений файлов), а работать со структурами сравниваемых файлов. Речь идёт об оглавлении файлов (т. е. о делении их на разделы и подразделы), одноимённых колонках в электронных таблицах, а также о выделении в документах общих крупных объектов — больших таблиц в текстовых документах, картинок (в частности, Visio-картинок, вставленных в MS Word файлах, внутри которых можно производить дальнейшее сравнение) и т. д. Выделение разделов, таблиц, картинок и пр. объектов — это опять процедура mapping/matching. На этом уровне также может быть найдена существенная разница между сравниваемыми документами. А для объектов, которые определены как сходные, можно запускать дальнейшее сравнение. Для Word-файлов это будет сравнение текстовых фрагментов.

Итак, в целом у нас получается единый многоуровневый алгоритм сравнения, определяющий все виды различий двух пакетов

документов, при этом на разных уровнях вложенности этот алгоритм работает по-разному. В него может быть интегрирован и алгоритм вычисления разницы двух деревьев (при сравнении структуры заголовков двух файлов) и вычисление разницы «плоского» текста для сравнения тестовых фрагментов.

3.2. Иерархическая визуализация

Представим, что у нас есть задача сравнить два сильно различающихся договора, имеющих, тем не менее, общую структуру. Тогда на первом уровне результат сравнения можно пометить красной иконкой. Далее, открывая один из договоров и просматривая его оглавление первого уровня, мы можем увидеть, что ряд его разделов тоже помечены красными иконками (т. е. они тоже сильно отличаются от аналогичных разделов в другом документе или в другом документе их нет вообще), ряд разделов помечены розовыми иконками (они отличаются незначительно), а остальные не помечены вообще — они идентичные. Открывая, в свою очередь, разделы, помеченные красной и розовой иконкой, мы увидим там разделы второго уровня, также помеченные подобным образом, и т. д. В конце концов, мы дойдём до интересующих нас листовых разделов и уже тут смотрим, собственно, на текстовую разницу. Важно, что при этом объем текста, который открывается нам на экране, не будет очень большим (при условии, конечно, что разделы в документе не очень большие, но и в этом случае мы может иметь иерархическую визуализацию: на первом уровне мы можем просмотреть только существенные различия, не отображая точечные). Кроме того, параллельно мы можем иметь ещё и текстовый лог сравнения — для желающих (опыт показывает, что такие находятся). Такой текстовый лог позволяет быстро взглянуть на результаты сравнения и увидеть много или мало различий, и если их, например, много, то воспользоваться иерархической визуализацией.

3.3. Нечёткое сравнение

При сравнении документов оказывается важным искать не только одинаковые или разные фрагменты текста, но также и похожие фрагменты, различия которых незначительны, поэтому пользова-

телю лучше предоставить «монолитный» результат. Далее, если ему интересно, то он должен иметь возможность получить исчерпывающую информацию обо всех сходствах и различиях в этих фрагментах. Кроме того, при сравнении файлов в двух пакетах важно уметь определять пару родственных файлов — например, в пакетах, содержащих два договора с приложениями, важно идентифицировать файлы с договорами (по одному в каждой папке), а также файлы с приложениями, имея в виду, что как файлы с договорами, так и файлы с приложениями могут по-разному называться в двух пакетах. Можно также уметь находить разные приложения, несмотря на то, что в одном пакете они могут быть собраны в одном файле, а в другом — разбросаны по разным файлам. Внутри, например, файлов с договорами алгоритм сравнения должен уметь игнорировать различные отличия в оформлении и форматировании — различное количество пустых строк между верхней границей файла и заголовком «Договор», разные отступы от правого края и использование различных элементов форматирования заголовков и списков и т. д. При этом использование алгоритмов diff для XML-файлов может быть ограниченно, целесообразно использовать алгоритмы поиска разницы на уровне «плоского» текста.

4. Заключение

Важно, чтобы целевые сервисы для работы с офисными документами, основанные на алгоритмах сравнения, были вариативными, т. е. могли не просто сравнивать два файла, или быть универсальной утилитой, которую можно много где использовать. Пользователю должен быть предоставлен удобный целевой сервис, который чётко выполняет нужную для него задачу, и ему известно, какую именно. Не пользователь должен искать применение таким сервисам. Иначе мы получаем демо-функциональность, наподобие средств версионного контроля и сравнения документов в продуктах MS Office — по мнению авторов эти функции MS Office не нашли активного применения на практике.

Часто бывает, что идеи сервисов, пришедшие из мира разработки ПО (например, версионный контроль документов, слияние и сравнения документов — т. е., в том числе, и привычный программистами diff), как таковые, не воспринимаются пользователя-

ми должным образом. Например, авторам доводилось объяснять необходимость средств версионного контроля типа Subversion бухгалтерам — безуспешно... Такой результат ошеломлял: казалось бы, что может быть проще и естественнее. Но потом мы поняли, что естественно и просто это лишь для нас, *годы* проработавших в среде MS Visual Studio и аналогичных ей, и использовавших MS Visual SourceSafe, Subversion, Git и прочие подобные продукты.

Итак, важно сравнивать не файлы, а, например, договора или счета. И для сравнения договоров это будет один сервис, а для сравнения счетов — другой, а не только разные интерфейсы для одной и той же функции. Важно также и то, что имея больше информации о сравниваемых активах, чем знание, что это просто XML-файлы, можно получить более качественный результат. Таким образом, речь идёт не об одном алгоритме, а о семействе алгоритмов, объединённых общей идеологией, общими требованиями, общими идеями и, возможно, общей платформой разработки (хотя, конечно, такая платформа должна содержать и другую функциональность, отличную от средств разработки сервисов на базе алгоритмов сравнения).

Список литературы

[1] *Гаврилова Т. А.* Логико-лингвистическое управление как введение в управление знаниями. Новости искусственного интеллекта. 2002. № 6. С. 32—49.

[2] *Кознов Д. В.* Методика обучения программной инженерии на основе карт памяти // Системное программирование / Вып. 3, под ред. А. Н. Терехова и Д. Ю. Булычева. СПб.: Изд. СПбГУ, 2008. С. 121–140.

[3] *Кознов. Д. В.* Основы визуального моделирования. Интернет-университет информационных технологий; БИНОМ. Лаборатория Знаний, 2008. 246 с.

[4] *Кознов Д. В., Ларчик Е. В., Плискин М. М., Артамонов Н. И.* О задаче слияния карт памяти (Mind Maps) при коллективной разработке // Программирование. 2011. № 6. С. 1–10.

[5] *Ларчик Е. В., Кознов Д. В., Плискин М. М.* Реализация механизма слияния карт памяти (Mind Maps) в продукте Comapping // Системное программирование. Том 6. № 1. СПб.: Изд-во СПбГУ, 2011. С. 45–64.

[6] *Левенштейн В. И.* Двоичные коды с исправлением выпадений, вставок и замещений символов // Доклады АН СССР. Т. 163. № 4. 1965. С. 845–848.

[7] *Черниговская Т. В., Гаврилова Т. А., Воинов А. В., Стрельников К. Н.* Сенсомоторный и когнитивный латеральный профиль. Физиология человека. 2005. Т. 31. № 2. С. 24–33.

[8] *Ahmadullin I., Allebach J., Damera-Venkata N., Fan J., Lee S., Lin Q., Liu J., O'Brien-Strain E.* Document Visual Similarity Measure for Document Search // Proceedings of The 11th ACM Symposium on Document Engineering (DocEng '11). ACM, New York, NY, USA. 2011. P. 139–142.

[9] *Bitlis B., Feng X., Harris J. L., Pollak I., Bouman C. A., Harper M. P., Allebach J. P.* A Hierarchical Document Description and Comparison Method // IS&T Archiving Conference. 2004.

[10] *Boytsov L.* Indexing Methods for Approximate Dictionary Searching: Comparative Analysis // Journal of Experimental Algorithmics (JEA), Vol. 16, Article 1.1. May 2011. 91 p.

[11] *Busen T.* The Mind Map Book. Penguin Books, 1996. 320 p.

[12] *Chawathe S. S., Garcia-Molina H.* Meaningful Change Detection in Structured Data // Proceedings of The 97 ACM SIGMOD International Conference on Management of Data, May 11–15, 1997, Tucson, AZ ,USA. P. 26–37.

[13] *Hascoët M., Dragicevic P.* Visual Comparison of Document Collections Using Multi-Layered Graphs. Tech Report lirmm-00601851. June 2011. 10 p.
http://hal-lirmm.ccsd.cnrs.fr/docs/00/60/18/51/PDF/vccd_last.pdf

[14] *Hunt J. W., McIl'roy D. M.* An Algorithm for Differential File Comparison. Technical Report, Bell Laboratories 41. June 1976. 9 p.
http://cm.bell-labs.com/cm/cs/cstr/41.pdf

[15] *LaFontaine R.* Merging XML Files: a New Approach Providing Intelligent Merge of XML Data Sets // Proceedings of XML Europe, 2002, Barcelona Spain. 21 p.

[16] *Lindholm T.* A 3-way Merging Algorithm for Synchronizing Ordered Trees — the 3DM Merging and Differencing Tool for XML, Master's Thesis, 2005, Helsinki University of Technology. 205 p.

[17] *Mani I., Bloedorn E.* Summarizing Similarities and Differences Among Related Documents // Information Retrieval 1, 1–2. May 1999. P. 35–67.

[18] *Koznov D., Pliskin M.* Computer-Supported Collaborative Learning with Mind-Maps Communications in Computer and Information Science. 2008. V. 17 CCIS. C. 478–489.

[19] *Miller W., Myers E. W.* A File Comparison Program // Software — Practice and Experience, Vol. 15, N 11. 1985. P. 1025–1040.

[20] *Myers E. W.* An O(ND) Diference Algorithm and its Variations. Algorithmica, Vol. 1, N 2, 1986. P. 251–266.

[21] *Oster P. M. G., Naja-Jazzar H.* Supporting Collborative Writing of XML Documents. INRIA, 2006. 8 p.

[22] *Rönnau S., Philipp G., Borghoff U. M.* Efficient Change Control of XML Documents // Proceedings of the 9th ACM Symposium on Document Engineering (DocEng). ACM, New York, NY, USA. 2009. P. 3–12.

[23] *Shvaiko P., Euzenat J.* A Survey of Schema-Based Matching Approaches. Journal on Data Semantics. Vol. 4. 2005. P. 146–171.

[24] *Suvanaphen E., Roberts J. C.* Textual Difference Visualization of Multiple Search Results Utilizing Detail in Context // Proceedings of Theory and Practice of Computer Graphics. 10 June 2004. P. 2–8.

[25] *Thao C., Munson E. V.* Using Versioned Tree Data Structure, Change Detection and Node Identity for Three-Way XML Merging // Proceedings of The 10th ACM Symposium on Document Engineering (DocEng '10). ACM, New York, NY, USA. 2010. P. 77–86.

[26] *Tichy W. F.* Design, Implementation and Evaluation of a Revision Control System // Proceedings of The 6th International Conference on Software Engineering (ICSE 1982). 1982. P. 58–67.

[27] *Vion-Dury J.-Y.* A Generic Calculus of XML Editing Deltas // Proceedings of The 11th ACM Symposium on Document Engineering (DocEng '11). ACM, New York, NY, USA. 2011. P. 113–120.

[28] *Ukkonen E.* Algorithms for Approximate String Matching // Information and Control, Vol. 64, Issues 1—3. January–March 1985. P. 100–118.

[29] *Wagner R. A., Fischer M. J.* The String-to-String Correction Problem // Journal of the ACM, 21(1). 1974. P. 168—173.