OSTIS-2012
(Open Semantic Technologies for Intelligent Systems)

УДК 004.738.5

ДЕТЕКТИРОВАНИЕ ИЗМЕНЕНИЙ В INTERNET-ДОКУМЕНТАХ

Молчанов Ю. Н., Глоба Л. С., Алексеев Н. А.

Национальный технический университет Украины «Киевский политехнический институт», г. Киев, Украина

molchanov_y@ukr.net

lgloba@its.kpi.ua

n_alexeyev@hotmail.com

В работе рассмотрена проблематика отслеживания изменений информации в сети Internet, предложен алгоритм детектирования изменений в древовидных структурах данных, обеспечивающий корректное отображение изменений при меньшей временной сложности, по сравнению с существующими методами, а также архитектура системы публикования/подписки, реализующая данный метод.

Ключевые слова: детектирование изменений, задача «хорошего соответствия», системы публикования/подписки, XML.

1. Введение

Одной из особенностей среды Internet в настоящее время является то, что любой человек, имеющий доступ к Internet, может выступать в роли как публикующего, так и подписчика информации. Такой широкий круг публикующих информацию приводит к неконтролируемому постоянному обновлению информации в среде Internet, что вызывает проблему отслеживания изменений по интересующим темам. Традиционный подход, основанный на проверке актуальности данных через определенные промежутки времени приводит к усложнению программно-аппаратных комплексов отслеживания изменений, повышенной нагрузке на ресурсы, предоставляющие информацию, и не всегда приводит к своевременному обнаружению изменений. Наибольшее влияние этих недостатков наблюдается в системах, которые критичны ко времени задержки в обновлении информации.

Данная проблема рассматривается в рамках разработки систем публикования/подписки [Eugster, 2003], которые обеспечивают разделение во времени, пространстве и синхронизации для публикующего и подписчика. Данные системы публикования/подписки разделяются на темо-ориентированные системы [Ramasubramanian et al.,2006], событие-ориентированные [Caporuscio, 2002] и типо-ориентированные. Данные системы имеют ряд недостатков: высокая временная сложность решения задачи детектирования изменений в документах, отсутствие поддержки структурированных документов, которые являются стандартом представления данных для среды Internet, некорректное или неполное определение изменений, произошедших в документе.

В данной работе предлагается новый алгоритм детектирования изменений в древовидных документах стандарта XML, который обеспечивает корректное отображение изменений при меньшей временной сложности, по сравнению с существующими методами.

2. Детектирование изменений в XML документах

2.1. Проблема детектирования изменений в XML документах

Определение изменений в документах является основной задачей разрабатываемых систем публикования/подписки. Большинство работ по детектирования изменений в XML документах затрагивают вопросы алгоритмической сложности операции вычисления разницы между документами, как для упорядоченных так и для неупорядоченных деревьев [Khanna et al., 2007], оптимизацией последовательности редактирования (также называемых дельты) - количества элементарных операций редактирования (добавление, удаление, обновление, перестановка узла), модели дельт и использование операции детектирования изменений в архитектурах хранения (обычно в базе данных) [Martinez et al., 2002]. В работе [Ronnau et al., 2008] авторы уделяют большое внимание проблеме алгоритмической и скоростной производительности новых diff алгоритмов, используя подход сверху-

вниз (а также предварительно вычисленные хеш коды). Другим подходом, предложенным сообществом разработчиков баз данных является исследование модели детектирования изменений, основанной на так называемых полных дельтах со строгими свойствами, среди которых обратимость и сокращение композиции дельт [Khandagale et al., 2010]. Тем не менее, механизм обозначения полностью зависит от поддержки постоянных идентификаторов каждого узла дерева. Можно сделать вывод, что большинство посвященных данной проблематике работ сосредоточены на производительности, временной и пространственной сложности операции детектирования изменений, а также оптимальности сгенерированных дельт, однако инструментарию и моделям, позволяющим использовать все это эффективным образом, уделяется слишком мало внимания. С практической точки зрения не много работ рассматривают необходимость и прикладное значение абстрагирования от дельт и связанных моделей операций.

В данной работе представлен метод определения изменений между структурированными документами стандарта XML, который рассматривает проблему детектирования изменений в условиях неопределенной схемы XML-документов. Данный метод не производит поиск последовательности редактирования, так как его основной задачей является корректное отображение произведенных изменений, а не преобразование исходного документа в измененный. Таким образом, основной задачей предложенного алгоритма детектирования изменений является поиск «хорошего соответствия» (англ. good matching problem) между узлами XML документов.

2.2. Определение соответствия между узлами XML документов

Для эффективного детектирования изменений в определенной части веб-страницы, необходимо вначале найти соответствующую часть в новом документе. Так как в новой версии документа изменения могут вноситься в любой его части, то расположение интересующего подписчика текста может измениться. Поэтому вначале необходимо найти такую часть в новом документе, которая бы наиболее соответствовала тексту в старой версии.

В предыдущих работах [Chawathe et al., 1997], [Chawathe et al., 1998] предусматривалось наличие «хорошего соответствия» между узлами в исследуемом документе, поиск которого является сложной задачей для неупорядоченного XML документа. Здесь под понятием «хорошего соответствия» мы понимаем такое соответствие между частями старой и новой версий документов, когда преобразование старой версии документа в новую требует минимального количества элементарных операций – вставки, удаления, перемещения узлов, обновления содержимого узлов. Еще одним направлением для поиска соответствующих частей в старой и новой версии документа является нахождения определенной меры похожести между частями документа, которая будет учитывать содержание, расположение и форматирование данной части документа. В работе [Flesca et al.,2009] был предложен набор параметров для определения меры похожести частей HTML документа. На основе идеи, изложенной в [Flesca et al.,2009], в данной работе были введены следующие параметры для поиска соответствия между частями XML документа: параметр соответствия содержимого, параметр соответствия атрибутов, параметр соответствия положения, а также интегральный критерий соответствия. Данный способ позволяет максимально учесть все возможные характеристики отдельного узла XML документа и найти «хорошее соответствие».

Другим важным вопросом является вопрос о преобразовании XML документа в XML дерево с определенными характеристиками. Такое преобразование впервые было представлено в [Barnard et al., 1995]. В предложенном методе также используется данное преобразование. Полученное XML дерево считается не упорядоченным, у которого отсутствуют метки, которые могли бы идентифицировать однозначно каждый узел. Между старой и новой версией документа возможны как структурные изменения, так и изменения содержимого узлов. Кроме того, были поставлены следующие условиями при решении задачи детектирования изменений в XML документах:

- XML документ представляется в виде упорядоченного дерева, в котором учитывается порядок узлов слева направо, где узлы дерева – тэги XML, метки узлов дерева, обозначенные как $l(a_i)$– названия тэгов XML при условии, что a_i - i-й узел дерева, значение узлов $\mathrm{con}(a_i)$ - текстовое содержимое i -го узла, связи между узлами в иерархии XML. Если тэг a_i включает в себя тэг a_j, то узел a_i является родительским узлом узла a_j;
- поиск соответствия производится только для листьев XML документа, значение которых содержит текстовую информацию, наиболее важную для подписчика;
- при сравнении XML тэгов атрибуты считаются совпадающими только при совпадении их названий и их значений;
- при поиске учитывается очередность дочерних узлов для любого родительского узла XML дерева, а также расположение узла в иерархии XML дерева на основании, введенного в данной работе, индексирования XML тэгов.

Данные условия были сформулированы, исходя из требований пользователя при поиске изменений информации в среде Internet.

Для реализации функции учета порядка узлов слева направо было принято решение о введении индексации в документе XML на основании числовых значений. Данное решение позволит учесть порядок узлов слева направо, при этом

сохранится возможность отслеживания местоположения узла в иерархии дерева XML. Это означает, что для каждого тэга в рассматриваемом документе при обработке будет введен атрибут index, которому будет присваиваться значение по разработанным правилам.

Далее соответствие между узлами XML деревьев будет определяться на основе числовых параметров. Данное решение было принято для уменьшения вычислительной сложности задачи и упрощение его практической реализации.

Определение 1. Пусть версия документа XML будет представлена деревом $T_1 = T_1(a_i,N,R,AT,con(a_i))$. Тогда дерево $T_1(a_i,N,R,AT,con(a_i))$ характеризуется следующими параметрами:

N – количество узлов в дереве T_1, что соответствует количеству тэгов в XML документе;

$R=\{r_i|i=1..m\}$ – совокупность родительских узлов в дереве T_1, где m - количество родительских узлов, r_i - родительский узел i-го узла.

$AT=\{at_i|i=1..N\}$ – совокупность атрибутов узлов в дереве T_1, at_i - атрибут i-го узла.

$con(a_i)$ - содержимое i-го, где a_i - i-й узел дерева T_1, а $A \in \{a_n\}$ - множество всех узлов заданного дерева.

Определение 2. Для заданного узла a_n документа T обозначим $T(a_n)$ - поддерево дерева T с корнем в узле a_n. При этом $T \in (a_n)$.

Определение 3. Узел документа T может быть представлен в виде множества $a_i(\text{con}(a_i),\text{att}(a_i),\text{index}(a_i))$, где параметр $\text{con}(a_i)$ - характеризует содержимое узла a_i,

$\text{att}(a_i)$ - совокупность атрибутов узла a_i,

$\text{index}(a_i)$ - параметр, характеризующий положение узла a_i в структуре дерева T, который определяется при индексировании XML документа

Для узла дерева T, рассматриваемого в данной работе, все родительские узлы будут иметь $con(a_i)=0$, так как полезную для пользователя информацию несут в себе «листья» дерева.

Определение 4. Введем параметр соответствия содержимого узлов a_1 и a_2 следующим образом:

.

$$P_{con}(a_1,a_2) = \frac{|con(a_1) \cap con(a_2)|}{|con(a_1) \cup con(a_2)|}. \quad (1)$$

Параметр $P_{\text{con}}(a_1,a_2)$ показывает отношение количества совпадающих слов в содержимом конечных узлов a_1 и a_2 к общему количеству слов в содержимом узлов a_1 и a_2.

Определение 5. Параметр соответствия атрибутов можно определить между узлами a_1 и a_2 так:

$$P_{att}(a_1,a_2) = \frac{\sum at_i \in \{at(a_1) \cap at(a_2)\}}{\sum at_i \in \{at(a_1) \cup at(a_2)\}} \quad (2)$$

Параметр $P_{\text{att}}(a_1,a_2)$ показывает отношение количества одинаковых атрибутов в двух узлах a_1 и a_2 к максимальному количеству атрибутов, которые содержатся в этих узлах.

Определение 6. Параметр соответствия расположения может быть вычислен по следующей формуле:

$$P_{dist}(a_1,a_2) = \frac{suf(index(a_1),index(a_2))}{\max(index(a_1),index(a_2))}. \quad (3)$$

В заданной формуле *suf* показывает длину общего суффикса между атрибутами узлов a_1 и a_2, которые определяют местоположение узла в иерархии дерева XML – между $index(a_1)$ и $index(a_2)$, а *max* определяет максимальную длину атрибута между $index(a_1)$ и $index(a_2)$.

На основе указанных параметров совпадения значения, атрибутов и положения необходимо вывести интегральный критерий соответствия двух узлов в дереве.

Для учета всех трех параметров необходимо ввести весовые коэффициенты их значений в конечной формуле интегрального критерия соответствия.

Определение 7. Пусть α,β,γ будут весовыми коэффициентами соответственно для $P_{\text{con}}(a_1,a_2)$, $P_{\text{att}}(a_1,a_2)$, $P_{\text{dist}}(a_1,a_2)$.

Тогда $\alpha+\beta+\gamma$ =1, а универсальный критерий соответствия примет вид:

$$CS(a_1,a_2) = -1 + 2 \cdot (\alpha \cdot P_{con}(a_1,a_2) + \beta \cdot P_{att}(a_1,a_2) + \gamma \cdot P_{dist}(a_1,a_2)) \quad (3)$$

Очевидно, что универсальный критерий соответствия может принимать значения [-1,1], где -1 соответствует минимальной схожести, а 1 соответствует максимальному совпадению.

Пусть A_1, A_2 - текстовые узлы старой версии XML документа (дерево T_1), B_1 B_2, B_3, B_4 - текстовые узлы новой версии XML документа (дерево T_2). На основании заданных условий можно сказать, что из старой версии XML документа были удалены две текстовые части, а оставшиеся части – изменены. Необходимо найти какие части были удалены, а какие изменены и как. Для этого нужно найти, какие из A_1, A_2 соответствуют, B_1 B_2, B_3, B_4.

Вначале найти интегральные критерии соответствия для каждой пары узлов. Для узлов A_1 и B_1:

$$CS(A_1,B_1) = -1 + 2 \cdot (\alpha \cdot P_{con}(A_1,B_1) + \beta \cdot P_{att}(A_1,B_1) + \gamma \cdot P_{dist}(A_1,B_1)) \quad (4)$$

.

Аналогично производится расчет для всех пар узлов. Для данного случая матрица критериев интегрального соответствия будет иметь вид, представленный в табл. 1.

Таблица 1 – Матрица интегральных критериев соответствия

	B_1	B_2	B_3	B_4
A_1	$CS(A_1,B_1)$	$CS(A_1,B_2)$	$CS(A_1,B_3)$	$CS(A_2,B_4)$
A_2	$CS(A_2,B_1)$	$CS(A_2,B_2)$	$CS(A_2,B_3)$	$CS(A_2,B_4)$

2.3. Математическая модель задачи поиска хорошего соответствия

Объектом данной задачи является определение «хорошего соответствия» между узлами документов. Целью задачи является исследование максимально возможной величины соответствия между деревьями, которая равна сумме интегральных критериев соответствия каждой из пар соответствующих узлов.

Для формализации данной задачи введем матрицу связности между узлами деревьев T_1, что соответствует старой версии документа и T_2, что соответствует новой версии.

Определение 8. Под параметром x_{ij} будем понимать связность узла A_i и B_j. Если A_i соответствует B_j, то x_{ij}.=1 Если A_i не соответствует B_j, то x_{ij}.=0.

Если для каждой пары узлов в старом и новом дереве назначить данный параметр связности, то данные величины могут быть записаны в виде матрицы, представленной в таблице 1.

Таблица 1 – Матрица связности старой и новой версии XML документа

	B_1	B_2	B_3	B_4
A_1	x_{11}	x_{12}	x_{13}	x_{14}
A_2	x_{21}	x_{22}	x_{23}	x_{24}

Введенный параметр связности будет однозначно определять, соответствует ли данная пара узлов друг другу или нет, то есть – одно из возможных состояний исследуемого объекта.

Найдем область допустимых решений задачи. С учетом условий, которые накладываются задачей поиска хорошего соответствия между узлами на решение и критерия оптимальности для задачи поиска «хорошего соответствия» в виде целевой функции, которая основана на матрице интегральных критериев соответствия, получаем следующую задачу булевого линейного программирования:

$$\begin{aligned} &a_{11}x_{11} + a_{12}x_{12} + a_{13}x_{13} + a_{14}x_{14} = b_1 \\ &a_{25}x_{21} + a_{26}x_{22} + a_{27}x_{23} + a_{28}x_{24} = b_2 \\ &a_{31}x_{11} + a_{35}x_{21} \le b_3 \\ &a_{42}x_{12} + a_{46}x_{22} \le b_4 \\ &a_{53}x_{13} + a_{57}x_{23} \le b_5 \\ &a_{64}x_{14} + a_{68}x_{24} \le b_6 \end{aligned} . \quad (5)$$

Или в сокращенной записи $Ax_{ij} \le B$, где матрица условий имеет значение:

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}. \quad (6)$$

а матрица B имеет вид:

$$B = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}. \quad (7)$$

Решение задачи состоит в нахождение такого решения (5), которое максимизирует линейную функцию:

$$\langle C, X \rangle = c_{11} \cdot x_{11} + c_{12} \cdot x_{12} + \ldots + c_{24} \cdot x_{24}. \quad (8)$$

где $c_{11}=CS(A_1,B_1)$, $c_{12}=CS(A_1,B_2)$, $c_{13}=CS(A_1,B_3)$, $c_{14}=CS(A_1,B_4)$, $c_{21}=CS(A_2,B_1)$, $c_{22}=CS(A_2,B_2)$, $c_{23}=CS(A_2,B_3)$, $c_{24}=CS(A_2,B_4)$.

Если рассматривать общий случай задачи и считать, что n- количество тэгов в исходном дереве A, а m-количество тэгов в обновленном дереве B. Не нарушая общности, будем считать, что $n<m$, тогда задача примет вид:

$$\begin{aligned} &\langle C, X \rangle = c_{11} \cdot x_{11} + c_{12} \cdot x_{12} + \ldots + c_{(n-1)m} \cdot x_{(n-1)m} + \\ &+ c_{nm} \cdot x_{nm} \\ &a_{11}x_{11} + a_{12}x_{12} + \ldots + a_{1m}x_{1m} = b_1 \\ &a_{n1}x_{n1} + a_{n2}x_{n2} + \ldots + a_{nm}x_{nm} = b_n \\ &a_{(n+1)1}x_{11} + a_{35}x_{21} + \ldots + a_{(n+1)(m+1)}x_{n1} \le b_{n+1} \\ &a_{(n+m)1}x_{1m} + \ldots + a_{(n+m)(n+m)}x_{nm} \le b_{(n+m)} \end{aligned} \quad (9)$$

.

Максимизация производится по всем булевым значениям величины x, удовлетворяющим системе (9):

$$\max_{x \in B^n : Ax \le B} \langle C, X \rangle. \quad (10)$$

3. Решение задачи поиска «хорошего соответствия» между узлами XML документов

В общем случае, представленная задача поиска «хорошего соответствия» является задачей булевого линейного программирования. Из этого следует, что решение задачи (10) имеет NP-сложность. Таким образом, правомерным является

использование переборных схем для решения такой задачи.

3.1. Неточные методы решения задачи булевого линейного программирования

Так как в задаче булевого линейного программирования, поставленной в рамках алгоритма детектирования изменений, количество листьев в каждом из деревьев может быть произвольным, то точные методы решения приведут к большой временной сложности в решении алгоритма. Для задач с большой размерностью могут быть использованы неточные методы решения, которые дают приближенное (субоптимальное) решение, но имеют меньшую временную сложность.

Более того, мощное направление в развитии приближенных подходов естественным образом возникло внутри точных методов (в основном методов ветвей и границ). При этом неоднократно отмечалось, что для значительного большинства прикладных задач совершенно достаточно вместо точного получить хорошее приближенное решение [Финкельштейн, 1976].

В данной работе предлагается использовать метод локальной оптимизации, описанный в [25]. Идея локальной оптимизации достаточно проста и используется во многих приближенных алгоритмах. Рассматривается задача дискретного программирования:

$$\begin{matrix} f(X) \to \max \\ X \in D \end{matrix}. \qquad (11)$$

Предполагается, что для каждого X из D определена окрестность $G(X)$, причем $X \in G(X)$. Тогда метод локальной оптимизации состоит в следующем.

Шаг 1. Находим $X^k \in D$. Переходим к шагу 2.

Шаг 2. Если X^k найдено, то перебором точек X^k находим X^{k+1} из условия $f(X^k)=\max\{f(X)|X\in G(X^k)\}$. Если $f(X^k)\geq f(X^{k+1})$, то $f(X^k)$ — локальный оптимум. Если же $f(X^k)<f(X^{k+1})$, то заменяем X^k на X^{k+1} и переходим к шагу 2.

Так как шаг 2 включает полный перебор точек $G(X^k)$, то понятие окрестности следует вводить таким образом, чтобы для каждого X величина $|G(X)|$ была достаточно малой 3.

Если множество D — конечно, то, очевидно, метод локальной оптимизации конечен.

При практической реализации очень важно разумным способом определить понятие окрестности. Если окрестности будут слишком малы, то процесс будет малоэффективен. Если же они окажутся слишком велики, то станет труден (или невозможен) сам акт перебора на шаге 2.

3.2. Применение метода локальной оптимизации для решения задачи поиска «хорошего соответствия»

Рассмотрим метод локальной оптимизации применимо к задаче (9) и запишем матрицу C в следующем виде.

$$C = \begin{pmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \dots & & & \\ c_{k1} & c_{k2} & \dots & c_{km} \\ c_{n1} & c_{n2} & \dots & c_{nm} \end{pmatrix}. \qquad (12)$$

На шаге 1 найдем допустимое решение для задачи БЛП. Для этого необходимо выполнить следующие действия.

Найти максимальное значение среди элементов матрицы C. Пусть максимальным элементом будет c_{k2}, тогда присвоим $x_{k2}=1$. Из этого следует, что в матрице X, $x_{ki}=0|i\in[1,2)\cup(2,n]$, $x_{j2}=0|j\in[1,k)\cup(k,n]$. Так как данный величины в матрице X найдены, то в матрице C необходимо уменьшить область поиска максимальных значений, убрав элементы матрицы C, не находящимися во втором столбце и k -й строчке. Данные действия необходимо повторять, пока не будут найдены все элементы X^k.

На основании указанных действий сформулируем алгоритм для шага 1:

1. Поиск максимального элемента матрицы C - элемент $c_{kl}=\max(C^*|C^*\subset\{c_{11}\dots c_{nm}\}$.

2. Присвоить $x_{kl}=1$. Из этого следует, что $x_{ki}=0|i\in[1,l)\cup(l,n]$, $x_{j2}=0|j\in[1,k)\cup(k,n]$. Если не получен X^k, то ограничить зону поиска максимального значения $C^* \subset \{c_{11}\dots c_{nm}\}\cap\{c_{k1}\dots c_{km}\}\cup\{c_{1l}\dots c_{n1}\}$ и повторить шаг 1.

В конце работы алгоритма будет получено решение X^k. Данное решение на шаге 2 необходимо локально оптимизировать на окрестности $G(X^k)$.

После проверки, осуществляемой на шаге 2, будет получен оптимальный результат.

Сформулируем алгоритм для шага 2:

1. Для всех столбцов $l|l=2\ldots n-m$, получаем значения $X_1^{k+1}...X_{l-1}^{k+1}$ с помощью перестановки строчек, содержащих 1 попарно для столбцов $1\ldots(l-1)$ и l.
2. Находим значение $f(X^{k+1})$ для всех $X_1^{k+1}...X_{l-1}^{k+1}$. Если $f(X^k)\geq f(X^{k+1})$, то $f(X^k)$ – локальный оптимум. Если же $f(X^k)<f(X^{k+1})$, то заменяем X^k на X^{k+1} и переходим к шагу 1.

После работы алгоритма будет получено оптимальное решение задачи БЛП X^k.

3.3. Временная сложность задачи поиска «хорошего соответствия» для узлов XML документов

Оценим число временную сложность решения заданной задачи (10) методом локальной оптимизации.

Ранее было принято без потери общности, что $n<m$. Пусть n – количество строк в матрицах C и X.

В алгоритме первого шага локальной оптимизации необходимо найти максимальное значение в каждой строке матрицы C. При этом на каждом шаге, количество возможных вариантов будет уменьшаться на 1.

Таким образом, сложность задачи первого шага z_1 будет составлять разность двух арифметических прогрессий S_m - суммы всех возможных вариантов перебора от 1 до m и S_{m-n} - суммы неиспользуемых вариантов от 1 до $(m-n)$:

$$R(z_1) = S_m - S_{m-n} = \frac{1+m}{2}\cdot m - \frac{1+(m-n)}{2}\cdot(m-n) = \frac{n^2+2nm+n}{2} \quad (13)$$

Временная сложность второго шага обусловлена количеством перестановок и составляет:

$$R(z_2) = S_{m-n-1} = \frac{1+(m-n-1)}{2}\cdot(m-n-1) = \frac{m^2-2nm-m+n+n^2}{2} \quad (14)$$

.

Таким образом, общая временная сложность решения задачи (10) методом линейной оптимизации составляет:

$$R(z) = R(z_1) + R(z_2) = \frac{n^2+2nm+n+m^2-2nm-m+n+n^2}{2} = \frac{2n^2+2n-m+m^2}{2} \quad (15)$$

$$R(z) \propto (n^2+m^2). \quad (16)$$

Предложенный алгоритм линейной оптимизации относится к неточным методам решения задачи БЛП, но он обеспечит нахождение локального максимума для задачи (10), который в большинстве случае совпадает с глобальным максимумом, обладая при этом более низкой временной сложностью $O(n^2+m^2)$ по сравнению с существующими методами.

3.4. Сравнение эффективности алгоритмов поиска соответствия между частями XML документа

В данной работе представлен новый подход к решению задачи поиска соответствия между XML документами. Было показано, что данный поход позволяет рассматривать эту задачу как задачу булевого линейного программирования. Для решения этой задачи было предложено использовать комбинаторный метод, сложность которого для данной задачи была определена как $O(n^2+m^2)$, где n и m - количество листьев в старом и новом деревьях.

Для показания целесообразности такого подхода необходимо сравнить временную сложность поставленной задачи в данном подходе с временной сложностью задач существующих алгоритмов.

Для этого сравним полученный алгоритм со всеми типами алгоритмов, представленных в обзоре: алгоритмы, основанные на поиске минимальной последовательности редактирования, семантические алгоритмы и алгоритмы, основанные на хеш подписях.

Алгоритм Чанга и Шаша рассматривает два дерева для нахождения минимальной последовательности редактирования. Данный алгоритм применим для XML документов. Минимальная временная сложность данного алгоритма определяется, как $O(p^2)$ [Caporuscio, 2002], где $p=n+m$ - общее количество узлов в двух деревьях. Тогда сложность алгоритма равна: $O((n+m)^2)$.

Семантические алгоритмы рассматривают XML документ как структуру XML тэгов. При сравнении учитывается иерархия синтаксиса языка и его закономерности. С помощью данного алгоритма можно определить только перемещение узлов, и невозможно определить удаление и вставку узла. Поэтому данный алгоритм не всегда находит корректное соответствие между узлами XML документов. Сложность данного подхода [Seung-Jin et al., 2001]: $O(|X|\cdot|Y|\cdot(|X|\log|X|+Y\log|Y|))$, где X и Y - количество ветвей в семантической иерархии каждого из деревьев. Так как количество ветвей иерархии можно принять равной количеством листьев в каждом из деревьев, то сложность данного метода можно представить следующим образом: $O(n\cdot m\cdot(n\log(n)+m\log(m)))$.

В системе детектирования веб-страниц Коури рассматривается схожий данной работой подход. Данный алгоритм рассматривает задачу нахождения соответствия между XML деревьями как задачу венгерского алгоритма. Сложность данной задачи в [Khoury et al., 2007] определена как $O((n+m)^3)$., что гораздо больше, чем сложность предложенного алгоритма.

На Рис. 1. показана сравнительная характеристика рассмотренных методов сравнения XML документов при помощи различных подходов.

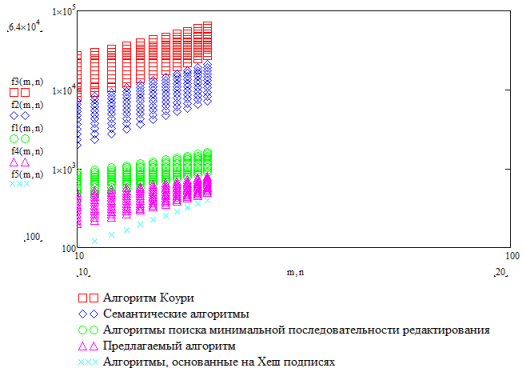

Рисунок 1 Сложность алгоритмов

4. Система публикования/подписки, основанная на методе детектирования изменений в XML документах

4.1. Обобщенная архитектура системы публикования/подписки

Здесь описывается общая архитектура системы публикования/подписки, определяющая основные модули системы и их функции. Система публикования/подписки должна разрабатываться для поддержки многопользовательских синхронных запросов мониторинга. Архитектура наследует клиент-серверную модель – клиентская часть, расположенная на клиентских машинах, обеспечивает интерфейс для общения пользователя с сервером приложения.

Архитектура системы является модульной, таким образом, позволяет изменять функционал или алгоритмы в модулях, не влияя на остальные модули. Модули системы можно условно разделить на 3 группы, что представлено на рис. 2: модули интерфейса, модули алгоритмов и модули хранения.

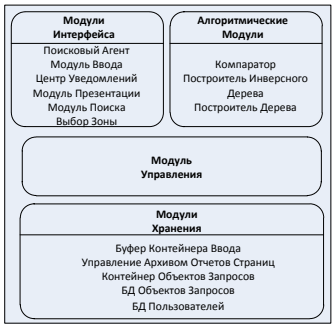

Рисунок 2 Классификация модулей системы

Далее будут представлены основные модули системы публикования/подписки, которые реализуют предложенный метод детектирования изменений в XML документах: построитель XML дерева и компаратор.

4.2. Основные функции построителя XML дерева

Построитель дерева отвечает за преобразование XML страницы в общую древовидную структуру данных. Процесс построения дерева состоит из трех этапов, которые реализуются отдельными подмодулями: фильтром, редактором и конструктором дерева. Первые два подмодуля необходимы для адаптации предложенного метода для детектирования изменений в HTML страницах.

Фильтр выполняет удаление неупорядоченных тэгов, скриптов и комментариев из страницы HTML. Фильтр генерирует новую версию документа в отдельном файле, которая может использоваться для детектирования изменений.

Вторым шагом построения дерева является перегруппировка HTML документа для решения проблемы некорректно закрытых тегов с помощью подмодуля редактирования. Выходными данными данного процесса является визуально аналогичный код, который может быть представлен общим деревом. На следующем этапе проводится индексирование тегов по правилам, используемом в предложенном методе детектирования изменений в XML документах.

На последнем шаге генерируется общее дерево из обработанного HTML документа.

4.3. Основные функции модуля компаратора

Компаратор предусматривает выполнение основных шагов предложенного метода детектирования изменений по следующему алгоритму. На вход компаратора подаются два, дерева, сформированных в построителе дерева. Так как в алгоритме было поставлено условие о выборе и сравнении только наиболее значимых для подписчика узлов, то на первом шаге осуществляется поиск таких узлов.

На втором шаге на вход Вычислителя подаются набор наиболее значимых узлов с их содержимым и атрибутами. В Вычислителе для каждой пары узлов из обоих деревьев происходит вычисление их критериев соответствия и формируется матрица интегральных критериев соответствия. Результат работы вычислителя представляет собой таблицу значений интегральных критериев соответствия всех пар узлов сравниваемых деревьев.

На третьем шаге в Блоке решения задачи булевого линейного программирования (БЛП) происходит формирования задачи и ее ограничений, а также производится ее решения с помощью метода локальной оптимизации. На выходе данного шага формируется информация о том, какие узлы исходного дерева совпадают с узлами измененного дерева.

На четвертом шаге в блоке компаратора происходит сравнение содержимого совпадающих

узлов, определенных на третьем шаге. Полученная разность содержимого и определяет изменения в информации, которая была запрошена абонентом.

В формировщике конечного дерева на основании исходного дерева формируется новое дерево *T'*, в котором будут выделены изменения, определенные между двумя деревьями. Данное дерево будет передано на Построитель Инверсного дерева для дальнейшей обработки и представления информации в удобном виде для конечного пользователя.

Заключение

При сравнении предложенного метода решения задачи поиска соответствия между XML документами с другими алгоритмами было показано, что предложенный алгоритм является эффективным с точки зрения временной сложности. Это достигнуто доказательством возможности рассмотрения задачи поиска соответствия между XML документами как задачи булевого линейного программирования и ее эффективного решения с помощью метода локальной оптимизации.

Библиографический список

[Финкельштейн, 1976] Финкельштейн, Ю.Ю., Приближенные методы и прикладные задачи дискретного программирования / Финкельштейн, Ю. // Москва: Наука. – 1976. – 265 с.

[Barnard et al., 1995] Barnard, D.T. Tree-to-Tree Correction for Document Trees / Barnard, D.T., Larke, G., Duncan, M. // Technical report, Queen's Univ. Computer Science Dept. – 1995. – pp. 23-42.

[Chawathe, et al., 1997] Chawathe, S. Meaningful change detection in structured data / Chawathe, S., Garcia-Molin: Proceedings of the ACM, SIGMOD International Conference on Management of Data, Tuscon, Arizona – 1997 - pp. 26–37.

[Chawathe, et al., 1998] Chawathe, S. Representing and querying changes in semistructured Data /Chawathe, S., Abiteboul,, S, Widom, // Proceedings of the International Conference on Data Engineering, Orlando, Florida. – 1998. – pp. 4–13.

[Coruscio et al., 2002] Coruscio, M. Formal Analysis of Clients Mobility in the Siena Publish/Subscribe Middleware / Coruscio, M., Inverardi, P., Pelliccione, P. // Technical Report, Department of Computer Science, University of Colorado – 2002. - pp. 1-18.

[Eugster et al., 2003] Eugster P. The many faces of Publish/Subscribe / Eugster P., Felber P., Guerraoui R., Kermarrec A.. - ACM Computing Surveys, Vol 35 – 2003.- № 2. – pp. 114-131.

[Flesca et al., 2007] Flesca, E. Efficient and affective Web change Detection / Flesca, E., Masciari, S. // Proceedings of the International Conference on Data Engineering, Tuscon, Arizona – 2007. – pp. 4–13.

[Khandagale et al., 2010] Khandagale, H.. A Novel Approach for Web Page Change Detection System / Khandagale, H. P., Halkarnikar, P. P. // International Journal of Computer Theory and Engineering – 2010.

[Khoury et al., 2007] Khoury, F. An Efficient Web Page Change Detection System Based on an Optimized Hungarian Algorithm / Khoury, F., El-Mawas, R., El-Rawas, O., Mounayar, E., Artail, H. // IEEE Transactions on Knowledge and Data – Vol.19. – 2007. – pp. 48-59.

[Lim et al., 2001] Lim, S. An Efficient Algorithm to Compute Differences between Structured Documents / Lim, S., Yiu-Kai, N. // Information theory, IEEE. – 2001. – pp. 171-180.

[Martinez et al., 2002] Martinez, M. A method for the dynamic generation of virtual versions of evolving documents / Martinez, M., Derniame, J.-C., de la Fuente, P. // Proceedings of the 2002 ACM symposium on Applied computing, New York, NY, USA. – 2002. – pp. 476 - 482.

[Ramasubramanian et al., 2006] Ramasubramanian, V. Corona: A High Performance Publish-Subscribe System for the World Wide Web. / Ramasubramanian, V., Peterson, R., Sirer, E. // Proc. USENIX Symposium on Networked Systems Design and Implementation. – 2006. – pp. 15-28.

[Ronnau et al., 2008] Ronnau, S. Merging changes in xml documents using reliable context fingerprints / Ronnau, S., Pauli, C., Borgho, M. // Proceeding of the eighth ACM symposium on Document engineering, New York, NY, USA. - 2008. – pp.58-62.

[Wang et al., 2003] Wang, Y. A fast change detection algorithm for XML documents / Wang, Y., DeWitt, D., Cai. X-Di, J. // In International Conference on Data Engineering (ICDE'03). Citeseer. – 2003. – pp. 235-258.

CHANGE DETECTION OF INTERNET DOCUMENTS

Molchanov Y.N., Globa L. S., Alexeyev N. A.

National Technical University of Ukraine «Kyiv Polytechnic Institute», Kiev, Ukraine

molchanov_y@ukr.net

lgloba@its.kpi.ua

n_alexeyev@hotmail.com

Information change detection problem in Internet is considered in the work, also the change detection algorithm of data tree structures is proposed, which provides correct detected changes presentation with lower time coplexity in comparison with existing methods.

Introduction

Nowadays, wide range of publishers in Internet environment leads to uncontrolled regular updates of information which causes change detection problem in the interested subjects. Traditional approaches with periodic source check result in complicated hardware and software for changes monitoring, increased information resources load and not always timely and correct change detection.

In this paper new change detection algorithm for structured documents is proposed. The described drawbacks were taken into account during presented change detection algorithm development.

Main Part

The paper is structured as follows. In the first section we describe XML document change detection problem and present boolean linear programming task for search of good matching between tree nodes. In the second section we suggest a solution for boolean linear programming task and good matching problem. Change detection system architecture is presented in the third section.

Conclusion

By comparing existing change detection algorithms for structured documents, high performance and low time complexity of proposed algorithm was shown.