УДК 004.02

Д. Н. Рубцов [1], **В. Б. Барахнин** [1, 2]

[1] Институт вычислительных технологий СО РАН
пр. Акад. Лаврентьева, 6, Новосибирск, 630090, Россия

[2] Новосибирский государственный университет
ул. Пирогова, 2, Новосибирск, 630090, Россия

E-mail: roubtsov@academ.org; bar@ict.nsc.ru

ВЫЯВЛЕНИЕ ДУБЛИКАТОВ В РАЗНОРОДНЫХ БИБЛИОГРАФИЧЕСКИХ ИСТОЧНИКАХ *

При запросах к нескольким разнородным библиографическим источникам возникает проблема выявления повторяющихся записей. В работе проанализированы проблемы, возникающие в процессе установления нечеткого соответствия между двумя записями. Рассмотрены существующие методы и алгоритмы решения задачи исключения дубликатов и, в частности, подходы к определению и вычислению функции похожести строк. С учетом требований конкретной задачи – усовершенствования информационной системы «Научные сотрудники – математики СО РАН» – реализован метод решения, основанный на использовании в качестве функции похожести наибольшей общей подпоследовательности двух строк. Метод был протестирован на трёх базах данных публикаций СО РАН – Базе данных публикаций журнала «Вычислительные технологии», Базе данных публикаций сотрудников Института вычислительных технологий СО РАН и Базе данных публикаций системы «Web-ресурсы математического содержания». По итогам проведённого тестирования метод продемонстрировал высокую эффективность работы и был применён для системы «Научные сотрудники – математики СО РАН» и разрабатываемой в данный момент интегрированной системы удалённого доступа к разнородным ресурсам библиографической тематики.

Ключевые слова: нечеткое соответствие, мера сходства, выявление дубликатов.

Введение

При запросах к нескольким разнородным источникам часто возникает проблема повторяющихся записей, когда два различных источника содержат документы, описывающие один и тот же объект (сущность) реального мира. В информационных системах, работающих с библиографическими описаниями публикаций научной тематики, вероятность возникновения такой ситуации существенно повышается.

Так как библиографические информационные системы, как правило, разрабатываются и поддерживаются независимо, и в каждом конкретном случае разработчики руководствуются своими собственными подходами, то записи, относящиеся к одним и тем же документам, могут быть представлены по-разному. В частности, такие записи могут иметь различную степень полноты или не соответствовать друг другу по причине опечаток создателей записей. В результате этого может возникнуть неоднородность как на уровне модели и схемы данных, так и на уровне самих элементов данных [1].

Для решения задачи интеграции разнородных источников возникает необходимость сопоставления, согласования и объединения различных представлений данных, а также исключения дублирующейся информации.

Процесс выявления и исключения дублирующейся информации может производиться как над двумя источниками одновременно, так и над уже интегрированным набором данных. Можно выделить следующие этапы:

- приведение документов (записей), полученных из разнородных источников, к единой схеме данных;
- выявление (т. е. сопоставление) похожих записей, относящихся к одному и тому же объекту реального окружения;

* Работа выполнена при частичной поддержке РФФИ (проекты № 07-07-00271, 08-07-00229, 09-07-00277), президентской программы «Ведущие научные школы РФ» (грант № НШ-931.2008.9) и интеграционных проектов СО РАН.

- объединение похожих записей в одну, содержащую все соответствующие атрибуты без избыточности;
- удаление избыточных записей, содержащих менее полную информацию.

В данной работе мы рассматриваем алгоритм решения задачи исключения дублирующих записей, получаемых при запросах к разнородным библиографическим базам данных научной тематики. При этом в понятие «библиографическая база данных» мы вкладываем более широкий смысл, чем «база данных, ведущаяся профессиональными библиографами», подразумевая, что речь может идти и о тех или иных библиографических списках, составленных без строгого соблюдения библиотечных стандартов.

В разрабатываемой системе «Научные сотрудники – математики СО РАН», приходится иметь дело с несколькими источниками, содержащими библиографическую информацию о публикациях сотрудников и имеющими различную структуру схемы данных. Публикации интегрируются из трёх баз данных MySQL:

- База данных публикаций журнала «Вычислительные технологии»;
- База данных публикаций сотрудников Института вычислительных технологий СО РАН;
- База данных публикаций системы «Web-ресурсы математического содержания».

Главными атрибутами объекта «научная публикация» являются название, список авторов и другие выходные данные публикации, а также некоторая дополнительная информация (веб-ссылка на полный текст или, по крайней мере, аннотацию к статье и др.). Можно выделить основные проблемы, возникающие на уровне элемента данных:

- орфографические ошибки, транспозиции символов, измененный порядок слов и т. д.;
- несогласованность в написании фамилии автора;
- случай полного совпадения фамилии, имени и отчества двух авторов;
- другие проблемы, связанные с предметной областью (прим. – первая и вторая части одной и той же статьи, опубликованные раздельно).

Заметим, что дополнительная информация о публикациях для некоторых записей из вышеперечисленных источников может, вообще говоря, и отсутствовать. В частности, некоторые записи могут не содержать ссылок как на аннотацию, так и на полный текст статьи. Это сразу же ограничивает нас в применении алгоритмов сравнения документов, работающих с полными текстами. В нашем случае мы можем использовать для установления соответствия между записями только главные атрибуты, являющиеся по своей сути строками, а также некоторые выходные данные публикации. Учитывая вышесказанное, первостепенной задачей, требующей решения для обнаружения дубликатов публикаций, становится выбор функции похожести или метрики для установления нечёткого соответствия двух строк.

Функции похожести строк и алгоритмы их вычисления

Расстояние между двумя объектами может быть вычислено с помощью различных мер близости, которые называют также метриками. Чем меньше это расстояние, тем более похожими считаются объекты сравнения.

Понятие метрики широко используется в различных областях, к примеру – в распознавании образов (букв, речи, изображений, лиц и т. д.).

Одними из наиболее часто встречающихся метрик для подсчёта расстояния в n-мерном пространстве являются меры Хемминга («манхэттенское расстояние»)

$$\sum_i |x_i - y_i|$$

и Евклида

$$\sqrt{\sum_i (x_i - y_i)^2}\ .$$

Для сравнения строк обычно используют метрики, оценивающие минимальное количество действий (операция редактирования), необходимых для преобразования одной строки в другую. К элементарным операциям редактирования относятся операции замены, вставки и удаления символа, из которых последние две иногда объединяют в одну.

Существует множество различных подходов к выбору функции похожести строк. Одной из классических мер является расстояние Левенштейна (также дистанция Левенштейна, функция Левенштейна, алгоритм Левенштейна). Согласно работам [2; 3] функция Левенштейна – это мера разницы двух последовательностей символов (строк) относительно минимального числа элементарных операций редактирования, необходимых для перевода одной строки в другую в случае, когда операции имеют одинаковый вес. Существует также модификация расстояния Левенштейна – расстояние Левенштейна – Дамерау, где в множество элементарных операций включены транспозиции символов. При этом требуется, чтобы к транспонированным символам не применялись другие операции редактирования.

Если придать единичный вес удалению и вставке и удвоенный вес замене, мы получим «расстояние редактирования». Разрешив только операцию замены с единичным весом, мы приходим к расстоянию Хемминга, которое определяется как количество позиций, в которых строки содержат различные символы. Оно пригодно для определения расстояния только в тех случаях, когда сравниваемые строки имеют одинаковую длину. В случае, когда разрешены только операции удаления и вставки с весом, равным единице, мы можем вычислить меру, которую называют наибольшей общей подпоследовательностью двух строк (LCS – Longest Common Subsequence).

Расстояние Джаро – Винклера [1] определяется по формуле:

$$d_j = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right).$$

Здесь m – число совпадающих символов; s_1 и s_2 – длины сравниваемых строк, а t – число перестановок. Два символа считаются совпадающими, когда расстояние между ними не превышает

$$L = \left\lfloor \max\frac{(|s_1|,|s_2|)}{2} \right\rfloor - 1.$$

Каждый символ первой строки сравнивается со всеми совпадающими с ним символами второй строки. Число перестановок определяется как число совпадающих, но идущих в неверном порядке символов, поделённое на два. Существует также модифицированный метод Джаро – Винклера, использующий веса отличные от 1/3.

Распознавание Рэтклиффа – Обершелпа [4] подсчитывает похожесть двух строк как число совпадающих символов, поделённое на общее число символов в обеих строках. Совпадающие символы определяются в виде наибольшей общей последовательности, а также совпадающих символов в остальной части по каждую сторону от наибольшей общей подпоследовательности.

Среди других подходов можно выделить алгоритмы проверки похожести звучания слов с помощью фонетического кодирования (Soundex, Metaphone, NYSIIS и др.) [2]. Обычно такие алгоритмы языкозависимы и плохо работают в случае, когда строки различаются в первом символе или содержат пробелы.

Ряд подходов также основан на сопоставлении лексем (схожесть Джаккарда и др.). В них работа ведётся с векторной моделью документов, а текст представляется в виде набора слов. В некоторых случаях вместо слов в качестве лексем выступают n-граммы (общие подстроки фиксированной длины n). Основным недостатком этих методов, как правило, является не слишком высокая эффективность работы при сравнении коротких строк или при наличии орфографических ошибок в словах [5].

Расстояние Левенштейна может быть вычислено с помощью метода динамического программирования Вагнера – Фишера [6]. Идея метода состоит в том, чтобы последовательно оценивать расстояния между удлиняющимися на каждом шаге префиксами строк до получения окончательного результата. Промежуточные результаты вычисляются итеративно и хранятся в массиве размерности $(m + 1) \times (n + 1)$, что приводит к затратам времени и памяти

[1] См. подробнее: Jaro-Winkler distance – http://en.wikipedia.org/wiki/Jaro-Winkler_distance.
[2] См. подробнее: Soundex – http://en.wikipedia.org/wiki/Soundex.

$O(mn)$, где m и n – длины сравниваемых строк. Для нахождения значения расстояния требуется вычислить mn элементов матрицы динамического программирования.

Согласно Смиту и Ватерману [7] можно дополнить понятие «расстояния редактирования», введя учёт пропуска последовательностей символов. Полученное расстояние обычно называется обобщённым расстоянием редактирования с аффинным штрафом за пропуски и может быть вычислено с помощью метода динамического программирования, использующего три матрицы. Однако сложность алгоритма в этом случае возрастает до $O(m^2n^2)$. Такая же сложность возникает и при вычислении расстояния Левенштейна – Дамерау.

На основе метода динамического программирования было разработано множество алгоритмов, в частности алгоритмы Хиршберга, Ханта и Мак-Илроя, Ханта и Шиманского, Машека и Патерсона, Укконена и Майерса и др. Более подробное описание и исследование этих алгоритмов можно найти в работах [3; 8].

Также для подсчёта длины наибольшей общей подпоследовательности существует ряд алгоритмов, основанных на использовании бит-векторов (AD, CIPR и др.) [9]. Они позволяют получить результат за время $O(mn/\omega)$, где ω – размер используемого алфавита.

Существующие решения

На данный момент существует достаточно большое число публикаций, посвящённых проблеме дублирующихся записей. Как правило, выбор решения в каждом конкретном случае сильно зависит от особенностей предметной области и постановки задачи.

В подавляющей части встретившихся нам работ для сравнения строк используется стандартная метрика Левенштейна, а также не производится никакой предварительной обработки данных.

В работе Н. Л. Цыганова [10] рассматривается методика нечеткого сопоставления записей баз данных персонала. На первом этапе производится предварительная кластеризация данных, после чего применяются алгоритмы нечеткого сопоставления строк (значений) для всех полей записи, по которым осуществляется поиск. Для полей имен используется метод вычисления обобщенного расстояния редактирования с аффинным штрафом за пропуски, а для остальных полей – вычисление схожести, основанной на совпадении лексем. В заключении строится вектор схожести отдельных полей для вычисления результирующей схожести записей.

В работе А. Ю. Гулы, А. П. Игнатенко, А. В. Чадюка [11] рассматривается методика идентификации физических и юридических лиц в хранилищах данных. Плюсом работы является предложенный алгоритм нормализации данных, включающий в себя унификацию как структуры данных (загрузка данных из разных источников в таблицу единого формата), так и самих данных (перевод строк в верхний регистр, удаление непечатных и повторяющихся символов, пробелов в начале и конце строки). Для сравнения записей в случае юридических лиц предлагается использовать алгоритм сопоставления биграмм, а в случае физических лиц – модифицированное расстояние Левенштейна.

Помимо частных решений, на данный момент существует ряд программных пакетов для очистки данных, в которых реализованы средства выявления и удаления дубликатов (DataCleanser DataBlade, ETI Data Cleanse, Integrity, Centrus Merge/Purge и др.). Ими поддерживаются различные подходы к согласованию атрибутов. Некоторые из них также позволяют интегрировать правила согласования, определённые пользователем.

Выбор алгоритма и его реализация

Рассмотрим более подробно процесс выявления и исключения дублирующихся публикаций для web-ресурса «Научные сотрудники – математики СО РАН». На первом этапе происходит интеграция данных из трёх источников, которые были перечислены выше. Информация извлекается с помощью метаданных. Ввиду того, что источники имеют различную структуру схемы данных, в местах, где это необходимо, производится слияние или расщепление соответствующих атрибутов. После этого приведённая к единообразному виду инфор-

мация заносится во временную таблицу, над которой происходит весь дальнейший процесс оперирования данными. Временная таблица содержит следующие поля:

- Authors – Авторы;
- Title – Название статьи (публикации);
- URL – Ссылка (на аннотацию к публикации);
- Description – Дополнительная информация о публикации (источник публикации);
- Year – Год публикации;
- Priority – Приоритет.

Название научной статьи по своей сути уникально, что позволяет добиться достаточно небольшого риска ошибочного определения дубликата. Однако использование только данного атрибута оказалось недостаточным для успешного выявления дубликатов. Как пример можно привести учебные пособия («Математический анализ» и др.), названия которых часто совпадают, при том что в остальных атрибутах могут наблюдаться различия. Поэтому для эффективного определения дубликатов мы использовали совокупное сравнение по нескольким атрибутам, основные из которых список авторов и название статьи.

Исходя из априорной информации о большом количестве абсолютно идентичных записей, на предварительном этапе записи проверяются на полное совпадение по каждому из атрибутов и, при достижении положительного результата, автоматически классифицируются как дубликаты. В противном случае алгоритм переходит к проведению нечёткого сравнения.

Перед непосредственным сравнением двух строк на этапе предобработки данных происходит преобразование букв с акцентами, а также перевод обеих строк в нижний регистр. Это позволяет улучшить качество получаемого результата.

Пороговый показатель сходства для каждого из основных атрибутов подбирался путём тестирования как на реальных базах данных, так и на специально сгенерированной базе данных, содержащей всевозможные ошибки, и составляет 60 % для атрибута ‘Authors’ и 80 % для атрибута ‘Title’. Было установлено, что для используемых баз данных эти показатели являются оптимальными для решения поставленной задачи – выявления всех дублирующихся записей. Во многом этот результат достигается именно за счёт совокупного использования нескольких атрибутов [12]. При варьировании же установленных показателей возможен пропуск некоторых дублирующихся записей, а также возникновение «лишних» пар дубликатов.

Для вычисления длины наибольшей общей подпоследовательности двух строк для сопоставления записей была выбрана одна из модификаций метода динамического программирования, предложенная Хиршбергом. Выбор данного метода был обусловлен достаточной эффективностью и относительной простотой реализации.

Затраты алгоритма относительно памяти и времени вычисления составляют соответственно $O(m+n)$ и $O(mn)$, где m и n – длины сравниваемых строк. Алгоритм реализован с помощью метода динамического программирования, основанного на рекурсии, на каждом шаге определяются длины наибольших общих подпоследовательностей у всё более и более длинных префиксов строк.

Обозначим их как $l(i, j)$, т. е. $l(i, j) = \left|lcs(x(1, i), y(1, j))\right|$.

Здесь функция $lcs(x, y)$ подсчитывает наибольшую общую подпоследовательность строк x и y соответственно. Так как длина наибольшей общей подпоследовательности любой строки и пустой равна нулю, значения границ массива задаются как $l(i, 0) = l(0, j) = 0$. В позиции (i, j), т. е. когда рассматриваются префиксы $x(1, i)$ и $y(1, j)$, если $x_j = y_j$, мы получаем новое значение функции lcs, присоединяя этот символ к текущему значению lcs префиксов $x(1, i-1)$ и $y(1, j-1)$, откуда $l(i, j) = l(i-1, j-1) + 1$. Иначе текущее значение lcs берётся в виде максимума из предыдущих соседних значений: $l(i, j) = \max\{l(i-1, j), l(i, j-1)\}$.

Заметим, что для вычисления строки i требуется только строка $i-1$. Для удобства введем вектор $ll(j) = l(m, j)$. Используется массив h длины $2(n+1)$, в котором нулевая и первая строки выступают в качестве строк $i-1$ и i массива l соответственно.

Граничные условия по j от 0 до n задаются как $h(1, j) = 0$.

Перед вычислением каждой новой «строки i» первая строка сдвигается вверх на место нулевой строки. Для этого используется цикл по i от 1 до m и по j от 0 до n, в котором $h(0, j)$ присваивается значение $h(1, j)$.

По j от 1 до n в позиции (i, j) при $x_j = y_j$ полагаем $h(1, j) = h(0, 1) + 1$. В противном случае полагаем $h(1, j) = \max\{h\,(1, j - 1), h(0, j)\}$.

На последнем этапе по всем j от 0 до n происходит копирование результата $h(1, j)$ в выходной вектор $ll(j)$.

В табл. 1 и 2 отражены результаты сравнения для десяти пар дубликатов по параметрам ‘Authors’ и ‘Title’ соответственно. Помимо разработанного алгоритма (LCS), в результаты теста также включены расчёты для двух стандартных функций PHP – Levenshtein и Similar_text.

Таблица 1

Результаты сравнения записей по параметру ‘Authors’

	1	2	3	4	5	6	7	8	9	10
LCS	99,10	98,19	98,73	80	100	83,3	74,19	80,52	96,55	98,67
Levenshtein(PHP)	98,20	96,36	97,47	–	100	70	–	76,62	93,10	97,33
Similar_text(PHP)	99,10	98,19	98,73	80	100	80	67,74	80,52	96,55	98,67

Таблица 2

Результаты сравнения записей по параметру ‘Title’

	1	2	3	4	5	6	7	8	9	10
LCS	100	100	100	100	99,55	100	100	100	100	100
Levenshtein(PHP)	100	100	100	100	99,10	100	100	100	100	100
Similar_text(PHP)	100	100	100	100	99,55	100	100	100	100	100

Из полученных процентных значений видно, что большинство ошибок приходится на атрибут ‘Authors’, при практически стопроцентном соответствии заголовков статьи. Такая ситуация обусловлена различным представлением списка авторов в различных базах данных и, как следствие, возможными ошибками при интеграции. Кроме того, в некоторых случаях этот список может оказаться неполным.

Для разработанного алгоритма полученные результаты практически совпадают с результатами для функции Similar_text. Однако в столбцах 6 и 7 более высокий результат был получен за счет лучшей обработки нашим алгоритмом ситуаций, когда расположение авторов в списке оказывается различным (случай перестановки слов). Таким образом, разработанный алгоритм продемонстрировал наилучшую эффективность работы.

Записи, для которых показатели сходства по каждому из основных атрибутов превышают пороговое значение, рассматриваются как потенциальные дубликаты, после чего происходит сопоставление по дополнительным атрибутам, таким как год публикации. При отсутствии информации о дополнительных атрибутах (пропущенные значения) записи трактуются как различные.

В случаях, когда приходится иметь дело с разными частями одной и той же статьи или книги, вышеперечисленных методов может оказаться недостаточно для получения ответа на вопрос, являются ли две сравниваемые записи дубликатами или нет. В качестве примера приведем две следующие записи:

1) В. А. Ильин, В. А. Садовничий, Бл. Х. Сендов «Математический анализ. Часть 1», 2006;
2) В. А. Ильин, В. А. Садовничий, Бл. Х. Сендов «Математический анализ. Часть 2», 2006.

В данном примере, при полном совпадении параметров 'Authors' и 'Year', различие заключается только в параметре 'Title', при этом степень сходства очевидно превышает выбранный пороговый показатель, вследствие чего при отсутствии дополнительной проверки записи могут быть ошибочно определены как дубликаты. Для обработки таких уникальных случаев, используется алгоритм поиска всевозможных вхождений вида “(1)”, “[1]”, “Часть 1”, “Часть первая” и других, что позволяет избежать описанной выше ошибки.

Описанная стратегия обеспечивает выявление подавляющего числа дублирующихся записей в рамках решаемой задачи. По завершению процесса выявления дубликатов из результа-

та запроса исключаются дублирующиеся записи, содержащие менее полную информацию. Этот процесс происходит в соответствии с выставленными приоритетами (атрибут ‘Priority’). В нашем случае удалось единственным образом упорядочить источники по полноте, таким образом при выводе предпочтение отдается источникам, содержащим более полную информацию.

Предложенный алгоритм был применен при разработке web-ресурса «Научные сотрудники – математики СО РАН» [3] [15], являющегося частью системы «Web-ресурсы математического содержания». Ресурс отражает информацию о научных сотрудниках – математиках СО РАН, а также ссылки на их научные труды и публикации. ходится иметь дело с разными частями одной и той же статьи или

Заключение

Был проведен анализ проблем и подходов к их решению в задаче исключения дублирующихся записей при одновременном запросе к нескольким библиографическим каталогам. На основе проведенного анализа реализован алгоритм исключения дублирующихся записей.

Также было проведено тестирование алгоритма на реальных базах данных публикаций СО РАН – Базе данных публикаций журнала «Вычислительные технологии», Базе данных публикаций сотрудников Института вычислительных технологий СО РАН и Базе данных публикаций системы «Web-ресурсы математического содержания».

В ходе тестирования были определены оптимальные параметры, необходимые для эффективной работы алгоритма. Алгоритм был применен для web-ресурса «Научные сотрудники – математики СО РАН».

Список литературы

1. *Rahm E., Hai Do H.* Data Cleaning: Problems and Current Approaches // IEEE Data Engineering Bulletin. 2000. № 23 (4). P. 3–13.

2. *Бойцов Л. М.* Классификация и экспериментальное исследование современных алгоритмов нечеткого словарного поиска // Тр. Всеросс. конф. RCDL'2004. [Электронный ресурс]. Режим доступа: http://www.rcdl.ru//papers/2004/paper27.pdf.

3. *Graham A.* «String Search» Technical Report TR-92-gas-01, School of Electronic Engineering Science, University College of North Wales: пер. М. С. Галкиной / Под ред. П. Н. Дубнера [Электронный ресурс]. Режим доступа: http://infoscope.ws/string_search/Stephen-92/index.html.

4. *Ratcliff J., Metzener D.* Pattern Matching: The Gestalt Approach // Dr. Dobb's Journal. July, 1988. P. 46.

5. *Цыганов Н. Л., Циканин М. А.* Исследование методов поиска дубликатов вебдокументов с учетом запроса пользователя // Интернет-математика 2007: Сб. работ участников конкурса. Екатеринбург: Изд-во Урал. ун-та, 2007. С. 211–222.

6. *Wagner R. A., Fisher M. J.* The String to String Correction Problem // Journal of the ACM. 1974. Vol. 21 (1). P. 168–173.

7. *Smith T. F., Waterman M. S.* Identification of Common Molecular Subsequences // Journal of Molecular Biology. 1981. Vol. 147. P. 195–197.

8. *Navarro G. A.* Guided Tour to Approximate String Matching // ACM Computing Surveys. 2001. Vol. 33 (1). P. 31–88.

9. *Hyyro H.* Bit-parallel LCS-length computation revisited // Proc. 15th Australasian Workshop on Combinatorial Algorithms (AWOCA 2004). 2004. [Электронный ресурс]. Режим доступа: http://www.cs.uta.fi/~helmu/pubs/pubs.html.

10. *Цыганов Н. Л.* Методика поиска дублирующихся записей с помощью алгоритма нечеткого сопоставления строк // Научная сессия МИФИ – 2007: Сб. науч. тр. М.: МИФИ, 2007. Т. 2: Технологии разработки программных систем. Информационные технологии. С. 159–160.

[3] См.: Web-ресурс «Научные сотрудники – математики СО РАН» – http://pine.ict.nsc.ru/sbras/math_soran/.

11. *Гула А. Ю., Игнатенко А. П., Чадюк А. В.* Задача идентификации физических и юридических лиц в хранилищах данных // VI Междунар. конф. по программированию УкрПРОГ'2008, 27–29 мая 2008 года, Киев, Украина. [Электронный ресурс]. Режим доступа: http://eprints.isofts.kiev.ua/416/.

12. *Барахнин В. Б., Нехаева В. А., Федотов А. М.* О задании меры сходства для кластеризации текстовых документов // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2008. Т. 6, вып. 1. С. 3–9.

Материал поступил в редколлегию 09.06.2007

D. N. Rubtsov, V. B. Barakhnin

DUPLICATE DETECTION IN HETEROGENOUS BIBLIOGRAPHIC SOURCES

When performing queries to multiple heterogeneous bibliographic sources the problem of repetitive records arises. The problems appearing in the process of detection of fuzzy match between two records are analyzed in this paper. The existing methods and algorithms of duplicate elimination and in particular the approaches to determination and calculation of string similarity function are considered. Taking into account the requirements of the concrete task of modernization of the information system «Mathematicians of SB RAS» the solution method was realized based on the use of longest common subsequence as string similarity function. The proposed method was tested on three SB RAS databases – Database of publications of Journal «Computational Technologies», Database of publications of employees of The Institute of Computational Technologies SB RAS and Database of publications of «Web-resources of the mathematical content». The method showed high efficiency on results of the testing and was applied for the information system «Mathematicians of SB RAS» and the integrated system of remote access to the heterogenous bibliographic resources which is being developed at the present moment.

Keywords: fuzzy match, similarity, duplicate detection.