О возможности борьбы с дубликатами при запросах к разнородным библиографическим источникам

В.Б.Барахнин, Д.Н.Рубцов
Институт вычислительных технологий СО РАН,
Новосибирский государственный университет

Источник проблемы

При запросах к нескольким разнородным источникам зачастую 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.

Этапы процесса выявления и исключения дубликатов

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

Проблемы, возникающие на уровне элементов данных

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

Приведение информации к единой схеме данных

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

Кроме того, данные проходят предобработку: преобразование букв с акцентами, а также перевод строк в нижний регистр.

Установление меры близости двух строк

Известна классическая мера — расстояние Левенштейна, т. е. 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.

Модификацией расстояния Левенштейна является 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”. 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).

На основе метода динамического программирования было разработано множество алгоритмов, в частности алгоритм Хиршберга.

Реализация алгоритма Хиршберга

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 $l(i, j)$, то есть $l(i, j) = |\mathrm{lsc}(x(1, i), y(1, j))|$.

Здесь функция $\mathrm{lsc}(x, y)$ подсчитывает наибольшую общую подпоследовательность строк x и y. 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), то есть когда рассматриваются префиксы $x(1, i)$ и $y(1, j)$, если $x_i = y_j$, мы получаем новое значение функции lsc, присоединяя этот символ к текущему значению lsc префиксов $x(1, i - 1)$ и $y(1, j - 1)$, откуда $l(i, j) = l(i - 1, j - 1) + 1$. Иначе текущее значение lsc берется 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 $x_i = y_j$ 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 $ll(j)$.

Затраты алгоритма относительно памяти и времени вычисления составляют соответственно $O(m + n)$ и $O(mn)$, где m и n — длины сравниваемых строк.

Тестирование алгоритма

Рассматривается интегрированная база данных, составляемая из трех разнородных БД:

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

Пороговый показатель сходства для каждого из основных атрибутов подбирался путём тестирования как на этих 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 повторяющихся записей. Во многом, этот результат достигается именно за счёт совокупного использования нескольких атрибутов. При варьировании же установленных показателей, возможен пропуск некоторых повторяющихся записей, а также возникновение “лишних” пар дубликатов.

Результаты расчетов

В таблицах 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.

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	98.20	96.36	97.47	—	100	70	—	76.62	93.10	97.33
Similar_text	99.10	98.19	98.73	80	100	80	67.74	80.52	96.55	98.67

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	100	100	100	100	99.10	100	100	100	100	100
Similar_text	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 счёт лучшей обработки нашим алгоритмом ситуаций, когда расположение авторов в списке оказывается различным (случай перестановки слов). Таким образом, разработанный алгоритм продемонстрировал наилучшую эффективность работы.

Особенности работы алгоритма

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 приходиться 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 приведём 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.

Заключение

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