Кодекс на РОМИП-2006. Использование B+ Tree для инвертированных файлов

© Губин М.В.

ЗАО «ИК «Кодекс»
max@gubin.spb.ru

Аннотация

Статья содержит описание новой версии программного обеспечения Кодекс, которое использовалось для выполнения экспериментов в рамках РОМИП-2006. Описывается организация индексных структур, алгоритмов их сжатия и определения релевантности документов.

1. Введение

«Кодекс» участвует в РОМИП[1] уже 4-ый раз. Каждое участие в семинаре является отличным стимулом для дальнейшего развития системы и уникальным полигоном для тестирования новых версий.

В этом году подсистема индексирования и поиска была в значительной степени переделана, были реализованы новые индексные структуры, алгоритмы обработки запросов и функция определения веса документов и терминов.

2. Описание системы

2.1 Особенности организации информационно-справочных систем и систем документооборота

Разрабатываемая консорциумом линейка программных продуктов «Кодекс»[2] используется как основа для правовых справочных систем «Кодекс», систем нормативно-технической информации «Техэксперт» и систем документооборота. Подобная область применения накладывает ряд особых требований к подсистеме индексирования и поиска по тексту:

1. Высокие требования к **компактности индекса**. В случае справочной системы пользователь ожидает мгновенной доступности системы после ее развертывания и не готов ожидать переиндексирования нескольких гигабайт входящей в нее информации (что может требовать часы на недостаточно мощных машинах). Поэтому базы поставляются с готовыми индексами и их объем может значительно увеличивать затраты на дистрибутирование. В системах документооборота функция поиска по тексту является не основной, и пользователи не согласны терять на него большие объемы дискового пространства.

2. **Мгновенная доступность** изменений при поиске. В системах документооборота критически важно, чтобы внесенный в систему документ мгновенно был доступен для поиска.

3. Возможность размещения индексов на **медленных устройствах**, таких как CD и DVD накопители или при доступе к файлам по локальной сети. При этом важно минимизировать не только число чтений, но и объем читаемой информации.

2.2 Использование B+дерева в качестве основы инвертированного файла

В системе «Кодекс» для поиска по тексту используется самая распространенная в настоящее время индексная структура, используемая для этих целей – инвертированный файл[3]. В отличие от многих реализаций, она строится «поверх» B+Tree. Данный подход уже описан в литературе[4], но его нельзя назвать широко распространенным.

Использование B+Tree имеет следующие преимущества:

1. Стандартная, хорошо отлаженная структура, которая так же используется в других модулях системы;
2. Вопросы управления памяти и управления совместным доступом хорошо исследованы и эффективно реализуются;
3. Обеспечивается быстрый доступ к конкретному вхождению слова, как при поиске, так и при модификации индекса.

Структура хранения может быть представлена следующим образом – листы вхождения слов разбиты на слоты, структуру которых можно представить на рис.1.

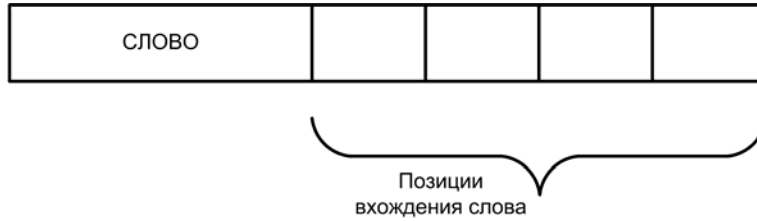

Рисунок 1. Слот дерева

Максимальное количество позиций в одном слоте фиксировано, если документ имеет пост лист длиннее, то он разбивается на несколько слотов. Функция сравнения слотов, упорядочивающая их в дереве, сравнивает сначала слово, затем, если слова совпадают, первую позицию вхождения.

Слоты являются записями, которые помещаются в классическое B+Tree[5]. Из-за сжатия слотов, описанного далее, они могут иметь различную длину на странице дерева. В нашей реализации дерева мы проводим деление страниц не по количеству слотов, а по занимаемому ими объему, поэтому у нас не возникало проблемы с переполнением страниц, которая значительно ухудшала быстродействие в реализации, описанной в [4].

2.3 Сжатие индекса

Требования к компактности индекса приводят к тому, что нами уделяется большое внимание разработке алгоритмов сжатия[6]. В последних версиях мы используем двухуровневый алгоритм сжатия: на уровне слота дерева и на уровне листа дерева.

На уровне слота сжимается слово, оно кодируется в Unicode с использованием алгоритма сжатия BOCU-I[7]. Часть листа вхождения, содержащаяся в слоте, сжимается с помощью дельта кодирования и байтового кодирования, что также является стандартным решением [3]. Известно, что этот алгоритм сжатия не дает хороших результатов в случае коротких листов[6], а в нашем случае все листы получаются достаточно короткими. Для устранения этого недостатка используется второй уровень сжатия, когда все данные листа дерева сжимаются с помощью алгоритма deflate библиотеки gzip[8]. Несмотря на простоту этого подхода, он обеспечивает высокое быстродействие и очень хорошую степень сжатия индекса. Размеры индекса для коллекций РОМИП приведены в части данной статьи, описывающей результаты экспериментов.

2.4 Режимы обновления индекса

В системе реализованы два варианта обновления индекса:

1. **In-place update.** Данный режим используется при изменении отдельных документов в системе документооборота. В этом режиме система сравнивает документ до и после модификации и формирует сортированный список необходимых изменений в индексе (записи об изменении, удалении и добавлений вхождений). Использование специальной нумерации позиций слов [9] уменьшает количество таких изменений. Далее все эти записи «проигрываются», производя соответствующие изменения в B+Tree.

2. **Bulk update.** Данный режим используется при получении пакета обновления в информационно-справочной системе или полном перестроении индекса при восстановлении базы в системе документооборота. При этом используется известный bottom-up алгоритм построения B+Tree из сортированных данных[10]. Для сортировки вхождения слов используется logarithm merge алгоритм, аналогичный используемому при построении индексов в системе Lucene[11].

2.5 Выполениен запроса

Выполнение запроса выполняется в несколько этапов:

1. Разбор запроса. На данном этапе запрос разбирается на отдельные слова и фразы на основании массива шаблонов, которые выделяют даты, номера документов, указания частей документов и т.п. В экспериментах РОМИП шаблоны не использовались, только разбиение запроса на слова.

2. Выборка информации из индекса. На основании отобранных слов формируется массив итераторов по индексу, каждый из которых возвращает вхождения каждого из слов в порядке возрастания позиций. Далее в цикле многопутевого слияния по этим итераторам выбираются позиции и собираются статистики вхождения слов и их сочетаний. Отдельно обрабатываются слова с частым вхождением, что описано в следующем разделе данной статьи.

3. Расчет весов документов на основании прохода по индексам. На данном этапе формируется массив документов-кандидатов на помещение в результат и вычисляются их веса.

4. Учет гипертекстовых связей между документами.

5. Формирование списка документов – результатов поиска.

2.6 Обработка длинных листов вхождений

Одним из недостатков описываемой организации индекса является медленная работа с очень длинными листами вхождений. Однако наличие такого листа говорит о том, что данное слово часто встречается в коллекции и, следовательно, имеет небольшой вес. Поэтому анализ всего листа вхождений данного слова не представляет особой ценности с точки зрения взвешивания документов и необходимо анализировать только вхождения рядом с другими словами запроса. Это можно назвать переходом к «локальным контекстам» [12]. Для определения таких слов на этапе формирования итераторов по позициям для каждого термина оценивается процент индекса, который занимается листом вхождений данного термина. При этом анализируется количество слотов, содержащих данное слово на уровне дерева, где оно встретилось первый раз. Очевидно, что чем на более высоком уровне встретились слоты с данным словом и чем их больше, тем больший процент список вхождений данного слова занимает. На рисунке 2 приведен пример 3-х уровневого дерева с 4 слотами на каждом уровне. На нем видно, что если оказалось, что 2 слота на 2 уровне содержат данное слово, то его лист вхождения составляет от 9 до 12.5%.

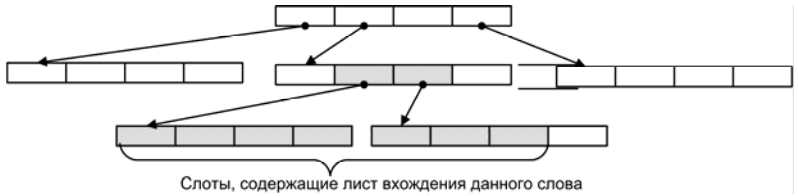

Рисунок 2. "Длинный лист вхождений"

Слова, листы вхождения которых занимают более 1% при минимальной оценке и высоте дерева более 3 уровней, обрабатываются следующим образом: итераторы по вхождениям этих слов не включаются в общий массив итераторов, а при обнаружении вхождения других слов запроса ищутся вхождения данного слова вблизи позиций других слов запроса и учитываются только эти позиции.

2.7 Алгоритм взвешивания документов

Для взвешивания документов использовался вариант TFIDF формулы, известной как BM25[13]. Мы не учитываем частоту слов в запросе, поэтому данную формулу можно представить в виде:

$$S = \sum_{t \in Q} \frac{(K+1)\,f}{K(1-b+bL)+f} * \ln\left(\frac{(r+0.5)(N-n-R+r+0.5)}{(n-r+0.5)(R-r+0.5)}\right) (1),$$

где:

K – настроечный коэффициент;

f – частота слова в документе, более подробно о данной величине далее;

r –число документов, отобранных как релевантные, содержащие термин t, (для часто встречающихся слов в позициях рядом с другими словами запроса);

R – число документов, отобранных в список-результат;

N- общее число документов коллекции;

n – число документов, содержащих термин t (для часто встречающихся слов в позициях рядом с другими словами запроса);

b – настроечный коэффициент;

L – нормализованная длина документа, которая вычислялась по формуле:

$$L = \frac{l}{L_{avg}} \quad (2),$$

где:

l – длина документа в словах;

L_{avg} – средняя длина документа, в отличие от классического варианта реализации BM25, мы используем среднюю длину для документов, которые отобраны как релевантные.

Нужно отметить также следующие особенности реализации данного алгоритма:

1. При вычислении частоты слова в документе различные вхождения учитываются по-разному. Вхождение слова в заголовках учитывается не как 1, а как 1+d, где d – настраиваемый параметр, обычно принимаемый 2;
2. Для учета взаимного появления слов все появления нескольких слов запроса внутри определенного «окна» обрабатываются как «псевдотермин» и учитываются в данной формуле. При этом частота вхождений в документ такого «псевдотермина» вычисляется по формуле, учитывающей расстояние между словами, количество слов запроса, вошедших в «псевдотермин» и совпадение порядка слов в «псевдотермине» и запросе.

После вычисления веса документов по формуле (1) выполняется анализ гипертекстовой структуры. При этом используется алгоритм, аналогичный LocalRank [14].

3. Эксперименты РОМИП-2006

РОМИП является уникальной возможностью провести разностороннее тестирование системы и проверить реализуемые подходы. В 2006 году мы ставили перед собой достаточно простую задачу проверить правильность выбранных решений и провести тестирование их реализации. Собственная разработка подобных тестов, их выполнение и анализ без РОМИП были бы чрезвычайно сложной задачей.

3.1 Характеристики индексов по коллекциям

Новая структура индекса по тексту обеспечила достаточно компактное представление информации. Данные для коллекций legal и web приведены в таблице 1. Данные для коллекции mixed не приводятся, потому что для нее специального индекса не строилось, а использовался совместный поиск по двум индексам.

Таблица 1. Характеристики сжатия

Коллекция	Объем текста (plain text), Mb	Объем индекса Mb	Коэффициент сжатия
Legal	765	220	28%
Web	3100	775	25%

Как видно, характеристики сжатия достаточно хорошие, учитывая простую реализацию, потери из-за блочной структуры индекса и то, что шумовые слова не отбрасывались и морфологический анализ (стемминг) не использовался.

Среднее время выполнения запроса, без времени формирования списка документов, для поиска по mixed коллекции на компьютере Intel Core Duo 1.83 ГГц, 1 Гб ОЗУ HDD FUJITSU MHV2100AH (ноутбук ASUS V6J) составило около 0.2 сек для коллекций legal и web и около 0.3 сек для mixed, что является достаточным для наших задач.

3.2 Качество поиска

Уникальной возможностью РОМИП является возможность объективной проверки качества информационного поиска. В нашей практической работе мы часто используем результаты предыдущих

конференций как тесты и baseline для оценки проводимых изменений и наличия ошибок в программном обеспечении. К сожалению, к моменту выполнения задач РОМИП программное обеспечение находилось в стадии активного тестирования и не все ошибки были устранены, что привело к тому, что сданные результаты оказались средними.

В настоящее время нами были детально проанализированы только результаты поиска по коллекции legal. После получения результатов были проанализированы запросы, у которых получены плохие результаты. Обнаружилось, что это запросы, которые содержат слова, написанные через дефис: счет-фактура, купли-продажи. Анализ показал, что программное обеспечение содержит ошибку, которая приводила к неправильной обработки таких слов. На рисунке 3 приведены 11-точечные графики при строгих требованиях к релевантности для прогона, переданного при выполнении задания РОМИП, и результат, полученный после исправления ПО.

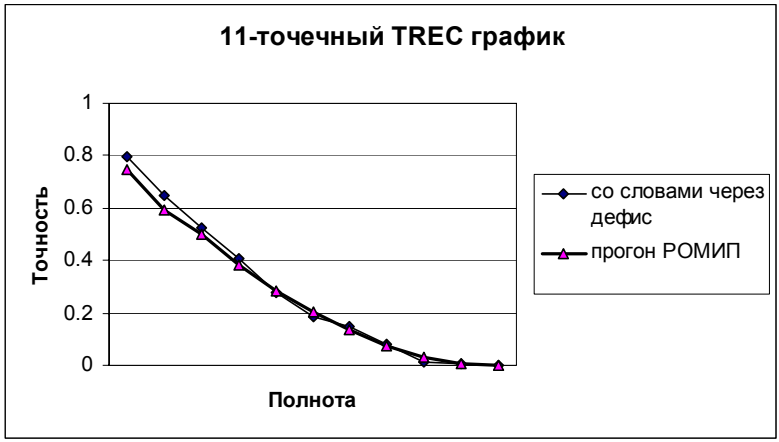

Рисунок 3. Результаты прогонов

4. Выводы

Выполненные нами в этом году эксперименты показали что:

1. Организация инвертированного файла на основе B+Tree является достаточно эффективной.
2. Двухуровневый алгоритм сжатия, когда на первом этапе используется специализированный алгоритм, а на втором

алгоритм общего назначения позволяет достичь достаточно высоких коэффициентов сжатия.

3. Для ускорения обработки запросов, содержащих часто встречающиеся слова, эффективным подходом является переход от анализа всех вхождений к «локальным контекстам», то есть к анализу только фрагментов документов, которые содержат слова запроса внутри фрагмента определенной длины.

Литература

[1] ROMIP Web site, 2005. http://romip.narod.ru

[2] Консорциум «Кодекс» http://www.kodeks.ru; http://www.cntd.ru

[3] *J. Jobel, A. Moffat.* Inverted Files for Text Search Engines, ACM Computing Surveys, Vol. 38, No. 2. (2006)

[4] *S. Melnik, S. Raghavan, B. Yang, and H. Garcia-Molina.* 2001. Building a distributed full-text index for the Web. In Proc.. of the 10th international Conference on World Wide Web (Hong Kong, Hong Kong, May 01 - 05, 2001). WWW '01. ACM Press, New York, NY, 396-406.

[5] *D.E. Knuth.* B-Trees. The Art of Computer Programming, Vol. 3: Sorting and Searching, 2nd ed. Reading, MA: Addison-Wesley, pp. 482-485 and 490-491, 1998.

[6] *М.В. Губин*. Изучение статистики встречаемости терминов и пар терминов в текстах для выбора методов сжатия инвертированного файла**,** Труды RCDL-2002, том 2, стр. 26-38, 2002

[7] *Doug Ewell***.** A Survey of UNICODE Compression, Unicode Technical Note # 14, http://www.unicode.org/notes/tn14/ ,2004

[8] GZip library http://www.gzip.org

[9] *М.В. Губин***.** Электронная библиотека многоверсионных документов*, Труды RCDL-2004, стр. 169-174, 2004*

[10] *Ig-hoon Lee, Junho SHIM and Sang-goo Lee.* Fast Rebuilding B+-Trees for Index Recovery. IEICE Transactions on Information and Systems 2006 E89-D(7):2223-2233.

[11] Lucene, http://lucene.apache.org

[12] *Илья Сегалович, Михаил Маслов***.** Яндекс на РОМИП-2004. Некоторые аспекты полнотекстового поиска и ранжирования Яндекс. Труды РОМИП'2004, 100-110, 2004.

[13] *S.E. Robertson et al***.** Okapi at TREC-3. In Proc. Of the third text retrieval conference (TREC-3)*,* :109−126.

[14] Local Rank http://www.google-directory.co.uk/algorithms-technology/localrank.php

Kodeks at ROMIP 2005. B+Trees based inverted file

M. Gubin
max@gubin.spb.ru

The article contains a report about the new version of Kodeks Information System which was used for ROMIP experiments. Index structures, algorithms of data compression and document ranking are described in detail.