На правах рукописи

Веретенников Александр Борисович

Программный комплекс и эффективные методы организации и индексации больших массивов текстов

05.13.18 – математическое моделирование, численные методы и комплексы программ

Автореферат диссертации на соискание ученой степени кандидата физико-математических наук

Екатеринбург — 2009

Работа выполнена в ГОУ ВПО «Уральский государственный университет им. А.М. Горького» на кафедре вычислительной математики.

Научный руководитель: доктор физико-математических наук,
профессор Владимир Германович Пименов

Официальные оппоненты: доктор физико-математических наук
Попов Владимир Юрьевич

кандидат физико-математических наук
Волканин Леонид Сергеевич

Ведущая организация: ФГОУ ВПО «Санкт-Петербургский государственный университет»,
г. Санкт-Петербург

Защита состоится _____ 2009 г. в ___ ч. ___ м. на заседании диссертационного совета Д 212.286.10 по защите докторских и кандидатских диссертаций при ГОУ ВПО «Уральский государственный университет им. А.М.Горького» по адресу:
620000, г.Екатеринбург, пр. Ленина, 51, комн.248.

С диссертацией можно ознакомиться в научной библиотеке ГОУ ВПО «Уральский государственный университет им. А.М. Горького»

Автореферат разослан _____ 2009 года.

Ученый секретарь
диссертационного совета
доктор физико-математических наук,
профессор В. Г. Пименов

ОБЩАЯ ХАРАКТЕРИСТИКА РАБОТЫ

Актуальность темы.

В последние годы резко повысилась актуальность обработки больших объемов разнородной текстовой информации. В первую очередь, это связано с лавинообразным развитием Интернета, которое привело к появлению громадного количества текстов, представленных самым различным образом: в виде обычных текстов, HTML- и XML- документов, сообщений электронной почты и пр. В частности, юридическая, патентная и новостная информация в Интернете исчисляется уже терабайтами.

Одновременно активно развиваются корпоративные системы, в которых хранятся огромные объемы информации. Постоянно создаются новые системы, идет развитие таких отраслей, как Enterprise Content Management. На сегодняшний день существует общая тенденция к переводу процессов документооборота из бумажного в электронный вид. Поэтому задачи поиска в подобных системах весьма актуальны.

В этой ситуации существенно возрос интерес к классификации больших массивов документов и быстрому поиску в них нужной информации.

Очевидно, что обеспечение быстрого поиска в подобных массивах должно являться комбинацией методов внешнего поиска (в дисковых файлах) и внутреннего поиска (в оперативной памяти). Структуры данных для каждого из этих видов поиска изучены достаточно хорошо, однако, их эффективные комбинации начали рассматриваться в литературе только в последние годы.

Хорошо известно, что для хранения и индексирования данных во внешней памяти обычно используются либо инвертированные файлы, Н. Прайвс (N. S. Prywes) и Г. Грей (H. J. Gray), либо B-деревья и их вариации, предложенные Р. Байером (R. Bayer), Э. Мак-Крейтом (E. M. McCreight) и М. Кауфманом (M. Kaufman) [не опубликовано]. Инвертированные файлы применяются в таких поисковых системах, как Яndex и Google. B-деревья часто применяются при выполнении поисковых запросов в базах данных.

С другой стороны, для быстрого поиска во внутренней памяти чаще всего применяются цифровые деревья Patricia, суффиксные деревья, суффиксные массивы и тернарные деревья поиска. Представляется логичным строить новые комбинированные структуры на основе уже апробированных структур. Первой из известных нам попыток создания та-

кой комбинированной структуры были String B-деревья – комбинация B-дерева и Patricia, предложенные П. Феррагина (P. Ferragina) и Р. Гросси (R. Grossi).

В диссертации рассматриваются задачи поиска в текстах, написанных на естественных языках. Более конкретно, в ней была описана новая структура данных, CLB-дерево, предназначенная для поиска в большом объеме электронных документов, написанных на естественном языке. Новая структура обладает рядом преимуществ по сравнению с другими структурами данных.

Большинство современных поисковых систем выполняют поиск по ключевым словам. Можно выделить несколько основных задач, решаемых этими системами.

Три задачи поиска, в порядке возрастания сложности.

1. Поиск всех документов, содержащих каждое из заданных слов.
2. Точный поиск фразы, нахождение всех документов, включающих в себя подряд все указанные слова (иначе говоря, между искомыми словами в тексте нет других слов).
   a. С учетом порядка искомых слов.
   b. Без учета порядка искомых слов.
3. Поиск с учетом расстояния, задача похожа на задачу пункта 2, только в тексте допускается наличие других слов между искомыми словами. Результатом поиска является набор фраз текста, в которых встречается как искомые слова, так и возможно другие слова. При этом в найденной фразе не должно содержаться фразы меньшего размера, удовлетворяющей условиям поиска.

Разработанный комплекс программ и структура данных CLB-дерево позволяют решать эффективно все эти задачи. Достаточно давно используемые для поиска в текстах инвертированные файлы имеют один существенный недостаток – их сложно обновлять. Остальные упомянутые структуры менее применимы для решения рассматриваемых задач. Вместе с тем, следует сказать, что массивы документов часто изменяются, и задача эффективного обновления индекса весьма актуальна.

Цель работы.

Цель диссертации состоит в создании комплекса программ, предназначенного для эффективного создания индекса и поиска в большом массиве текстовых данных. Основной задачей является разработка та-

кого комплекса, который будет обладать возможностью быстрого добавления новых данных в индекс. Для решения данной задачи требуется разработать новые структуры данных и алгоритмы.

Методы исследования.

В соответствии с концепциями математического моделирования исследование разбито на три этапа: модель – алгоритм – программный комплекс.

На первом этапе построена модель исследуемого объекта – большого набора текстов, с учетом анализа морфологии языка. Построенная модель отражает важнейшие свойства исследуемого объекта, связи между его составляющими частями и позволяет исследовать объект и связанные с ним научные и технические проблемы теоретическими методами. Построенная модель является фундаментом для дальнейшего исследования с использованием современной технологии математического моделирования.

На втором этапе для решения поставленной задачи разработана новая структура данных, CLB-дерево, и широкий набор алгоритмов для работы с ней. Выявленные в результате анализа модели свойства текстов позволили еще в начале разработки определить наиболее важные требования к структуре данных и алгоритмам, а также понять, за счет применения каких подходов можно достичь максимальной эффективности при решении поставленных проблем.

На третьем этапе разработан комплекс проблемно-ориентированных программ, предназначенный для решения задачи. После чего были проведены разнообразные эксперименты с целью подтверждения эффективности созданного комплекса программ, а также разработанных структур данных и алгоритмов.

Исследование основано на современных методах программирования, таких, в частности, как структурное, объектно-ориентированное программирование и активное применение шаблонов и моделей. Систематически применяются понятия и методы теории вычислительной сложности, методов построения и анализа алгоритмов.

В результате организации работы в соответствии с методологией математического моделирования получены гибкие и универсальные инструменты для исследования.

Научная новизна.

Разработана новая структура данных CLB-дерево. CLB-дерево – это структура данных, по скорости поиска не уступающая инвертированным файлам и их аналогам, но в отличие от них в CLB-дерево можно легко и быстро добавлять новые данные. При поиске в текстах с помощью CLB-дерева учитывается морфология языка этих текстов. Эффективность CLB-дерева доказана в ряде теорем. Разработан комплекс программ, на практике показывающий преимущества использования CLB-дерева.

Теоретическая и практическая ценность.

Разработанные в диссертации структуры данных и алгоритмы позволяют эффективно строить быстро обновляемый индекс, предназначенный для поиска в большом массиве текстовых данных. Разработанные методы, структуры данных и алгоритмы реализованы в комплексе программ.

Основные результаты.

1) Разработана структура данных CLB-дерево.
2) Разработаны эффективные алгоритмы создания CLB-дерева на основании большого массива текстовых документов. Основное преимущество CLB-дерева заключается в том, что в CLB-дерево можно легко добавлять новые данные, при этом скорость поиска такая же, как у инвертированных файлов.
3) Получены теоретические оценки затрат ресурсов на добавление данных в CLB-дерево, поиск данных в CLB-дереве и хранение CLB-дерева. Данные оценки позволяют заранее предсказать, сколько времени займут создание индекса, поиск и сколько места во внешней памяти может потребоваться для хранения индекса.
4) Разработан программный комплекс, позволяющий строить индекс на основании большого массива текстовых документов. Разработанный комплекс имеет модульную архитектуру и широкие возможности конфигурации. Реализован интерфейс для использования программного комплекса в других продуктах.
5) Проведены эксперименты подтверждающие эффективность разработанных структур данных и алгоритмов, как в 32-битной среде, так и в 64-битной среде.
6) Проведены сравнительные эксперименты с инвертированными файлами по созданию индекса и поиску. Эксперименты показывают преиму-

щество CLB-дерева при создании индекса, а также то, что скорость поиска в CLB-дереве такая же, как и при использовании инвертированных файлов.

7) Проведены сравнительные эксперименты с рядом широко используемых программных комплексов, предназначенных для решения рассматриваемых задач. Проведенные эксперименты показывают преимущество по скорости создания индекса, основанного на CLB-дереве, по сравнению с аналогами.

Публикации. Основные результаты диссертации опубликованы в работах [1-7]. Результаты, вошедшие в диссертацию получены автором самостоятельно. Работы [1-2] опубликованы в ведущих рецензируемых научных журналах, определенных ВАК. Из совместных работ в диссертацию вошли результаты, полученные лично автором.

Структура и объем диссертации. Диссертация состоит из введения, 4-х глав и списка литературы. Главы разбиты на параграфы, нумерация глав и параграфов в работе сквозная. Нумерация формул и утверждений в работе двойная: первый индекс – номер параграфа, второй индекс – порядковый номер формулы или утверждения внутри параграфа. Общий объем работы составляет 150 страниц, библиография содержит 33 наименования.

Апробация работы. Результаты диссертации докладывались и обсуждались на:

— Международной алгебраической конференции: К 100-летию со дня рождения П. Г. Конторовича и 70-летию Л. Н. Шеврина (Екатеринбург, 2005);

— 37-й Региональной молодежной конференции «Проблемы теоретической и прикладной математики» (Екатеринбург, 2006);

— 39-й Региональной молодежной конференция «Проблемы теоретической и прикладной математики» (Екатеринбург, 2008);

— Электронные библиотеки: перспективные методы и технологии, электронные коллекции. Десятая Всероссийская научная конференция «RCDL'2008» (Дубна: ОИЯИ, 2008);

— Третьей международной научной конференция «Информационно-математические технологии в экономике, технике и образовании» (Екатеринбург, 2008);

— 40-й Всероссийской молодежной конференции «Проблемы теорети-

ческой и прикладной математики» (Екатеринбург, 2009);

— Межвузовской научной конференции по проблемам информатики «СПИСОК 2009». (Екатеринбург 2009);

— Научном семинаре «Системный семинар» в Уральском государственном университете (Екатеринбург, 2005, 2008).

СОДЕРЖАНИЕ ДИССЕРТАЦИИ

Во введении обоснована актуальность темы исследований, сформулирована цель диссертационной работы и пути ее достижения, отмечена новизна и практическое значение работы.

В первой главе построены модели исследуемого объекта, дается обзор существующих структур данных, алгоритмов, применяющихся при работе с внешней памятью, и систем.

Вначале построена базовая модель текстов. Затем путем ее усложнения построена модель, учитывающая морфологию языка. Далее построена модель базы данных поисковой системы, данная модель определяет каким образом должны хранится данные для решения поставленных задач поиска. После чего дана модель программного комплекса поисковой системы. После описания моделей приведены ряд часто используемых структур данных, которые могут быть использованы для построения комплекса программ. Описана структура несколько поисковых систем, применяющихся для поиска в большом наборе текстовых данных.

Формальная модель текста с учетом морфологии

Обрабатываемый текст является последовательностью предложений. Каждое предложение является последовательностью слов.

В том случае, когда слово входит в словарь морфологического анализатора, анализатор возвращает идентификатор слова в словаре и набор идентификаторов базовых форм для данного слова.

В случае, когда слово не входит в словарь морфологического анализатора, требуется сохранять его в отдельном словаре; в этом случае можно считать, что слову соответствует идентификатор в данном словаре, у него одна базовая форма и идентификатор этой базовой формы совпадает с идентификатором слова.

Таким образом, каждому слову можно поставить в соответствие следующую запись

$KNOWN$	1 – слово входит в словарь морфологического анализатора, 0 – не входит
$WORD\_ID$	Идентификатор слова
$BASE\_IDS$	Набор идентификаторов базовых форм слова
$SEQUENCE\_ID$	Идентификатор предложения. Например порядковый номер предложения
$NUMBER$	Порядковый номер слова в тексте
$SEQUENCE\_NUMBER$	Порядковый номер слова в предложении
$DOCUMENT\_ID$	Идентификатор документа

Данную запись далее будем назвать записью о вхождении слова в документе.

Текст после морфологического анализа можно представить как набор подобных записей.

База данных, содержащая подобные записи позволяет выполнять различные запросы.

Например, можно для заданного слова получить набор всех его вхождений в обработанных документах.

Если мы ищем некоторую фразу, то, получив для каждого слова фразы набор его вхождений, можно определить, в каких документах встречается фраза целиком.

Во второй главе дается описание разработанных автором данной работы структур данных, алгоритмов, и ряд теорем, которые доказывают эффективность разработанных алгоритмов.

Дается описание CLB-дерева, приведен основной алгоритм эффективного создания индекса, приведен алгоритм поиска.

CLB-дерево – это структура данных, по скорости поиска не уступающая инвертированным файлам и их аналогам, но, в отличие от них, в CLB-дерево можно легко и быстро добавлять новые данные.

Текущая реализация CLB-дерева использует морфологический анализатор.

В качестве морфологического анализатора может использоваться анализатор, созданный Ю. С. Лукачом. Морфологический анализатор содержит около 3,5 млн. словоформ русского языка, образованных от 205 тысяч базовых форм. Для каждого слова, известного анализатору он возвращает одну или несколько базовых форм. Слова, хранящиеся в словаре морфологического анализатора, составляют более 90% всех слов в типичных документах.

Также может использоваться широко известный морфологический анализатор АОТ (www.aot.ru). По своему объему он сопоставим с анализатором Ю. С. Лукача. Кроме этого, АОТ также поддерживает английский язык и при работе с англоязычными текстами используются словари этого анализатора.

Под известными словами мы будем понимать слова, входящие в словарь морфологического анализатора, а под неизвестными – те слова, которые не входят в этот словарь. Известные слова могут иметь несколько базовых форм. Неизвестные слова всегда имеют одну базовую форму – само слово. Таким образом, каждому слову сопоставляется набор его базовых форм (далее просто форм).

Основные моменты: предотвращение фрагментации данных для последующего быстрого поиска, эффективное использование дискового пространства, сокращение количества обращений к внешней памяти для повышения производительности, методы организации кэша.

Каждой базовой форме слова соответствует некоторая цепочка блоков, которые назовем кластерами. В этой цепочке содержится описание всех вхождений данной базовой формы во всех документах. Например, для каждого вхождения сохраняется номер или идентификатор соответствующего документа и позиция соответствующего слова в этом документе. Каждый кластер имеет фиксированный размер K. Каждая цепочка кластеров описывается небольшой структурой-указателем.

Указатели для базовых форм известных словоформ хранятся в таблице – массиве, находящейся в оперативной памяти. Базовые формы неиз-

вестных словоформ и указатели для них хранятся в B-дереве, оно расположено во внешней памяти и разбито на страницы фиксированного размера H.

При этом, для формы известного слова в памяти сохраняется последний кластер соответствующей цепочки и добавление новой информации о соответствующем слове производится в этот кластер.

Таким образом, у нас есть файл, состоящий из блоков заданного размера. Несколько блоков объединяется в список: в каждом блоке из данного списка сохраняется ссылка на следующий блок.

При создании индекса, когда форма слова встречается первый раз, мы создаем список, состоящий из одного блока, в который записываем информацию о вхождении данной формы слова в тексте. Если это неизвестное слово, мы помещаем его в B-дерево вместе с указателем на созданный список кластеров. Если известное – используем для хранения данного указателя таблицу для известных слов.

Если форма слова встречается повторно, то информация о следующем вхождении добавляется в созданный ранее блок. Если свободное место в блоке заканчивается, мы создаем новый пустой блок (увеличивая файл в размере) и прописываем на него ссылку в текущем блоке, и далее информация добавляется в новый блок. Таким образом, мы достигаем возможности легкого обновления индекса.

Организация данных для эффективного чтения

Зафиксируем некоторое число c и соответствующее ему число $BLOCK\_L = 2^c$. Цель алгоритма заключается в том, чтобы организовать хранение данных таким образом, чтобы список блоков был разбит на группы, и все блоки, входящие в заданную группу были расположены в файле последовательно.

Например, если у нас есть 25 блоков и $BLOCK\_L = 8$, мы разбиваем 25 блоков на группы следующих размеров: 8, 8, 8, 1.

Тем самым весь список блоков разделяется на группы по $BLOCK\_L$ блоков, за исключением последней группы, в которой может быть меньше чем $BLOCK\_L$ блоков, т. к. длина списка может не делится нацело на $BLOCK\_L$.

При проведении экспериментов размеры блоков брались 4 – 16 килобайт. Число $BLOCK\_L$ было выбрано таким образом, чтобы суммарная

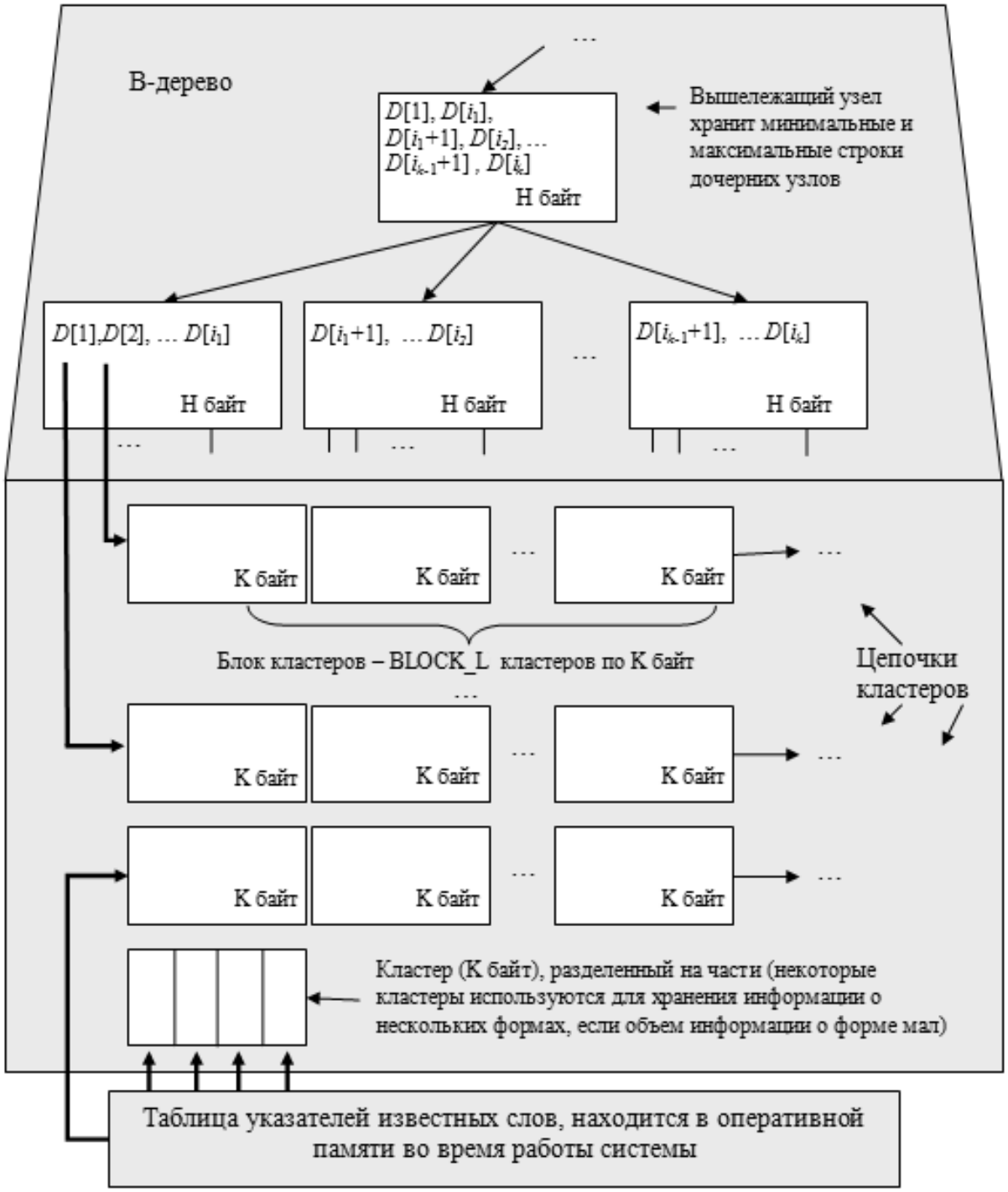

Рис. 1: Структура CLB-дерева. Массив D в дереве составлен из всех неизвестных слов данного текста, и храниться в B-дереве (в верхней части рисунка). Каждой форме каждого слова текста сопоставлена цепочка кластеров, содержащая информацию обо всех вхождениях данной формы в тексте

длина группы последовательно располагающихся блоков была не меньше

8 Мб.

В результате скорость считывания данных из списка блоков практически совпадает со скоростью последовательного чтения данных.

Итак, кластеры объединяются в блоки кластеров. Кластеры в блоке располагаются последовательно. Максимальная длина блока $BLOCK\_L$ является числом степени 2. Последний кластер в блоке хранит в себе указатель на первый кластер следующего блока, если текущий блок не последний. Все блоки, кроме последнего, имеют длину $BLOCK\_L$.

Эффективное заполнение блоков

Для многих базовых форм слов объем информации, который следует хранить в цепочке, мал (меньше половины K). Для этого случая существует отдельный список кластеров. Каждый из таких кластеров делится на определенное количество равных частей, и для конкретной базовой формы слова используется только одна из этих частей. В этом случае мы считаем, что цепочка кластеров состоит из соответствующей части указанного кластера. Остальные части могут использоваться другими базовыми формами слов, для которых объем информации также мал.

Поиск словоформы и извлечение информации о ней

1. Создается пустой список указателей
2. Если словоформа известна, то морфологический анализатор выдает список номеров ее базовых форм. Каждый номер соответствует ячейке таблицы для базовых форм известных словоформ. Из этой ячейки, если она не пуста, извлекается указатель и добавляется в список указателей.
3. Если это неизвестная словоформа, то она ищется в B-дереве и, если она там есть, возвращается ее указатель и добавляется в список указателей.
4. Для каждого указателя из списка определяется цепочка кластеров, и считываются все данные из нее.

Это позволяет реализовать различные алгоритмы поиска. По сути дела, с точки зрения поиска CLB-дерево предлагает те же возможности, которые есть у инвертированных файлов.

Кэширование для слов, входящих в словарь морфологического анализатора

Очевидно, что если мы будем каждый раз при добавлении информации о вхождения слова записывать данные в файл, то мы будем создавать индекс очень долго.

Основная оптимизация осуществляется за счет использования морфологии языка. Слова, входящие в состав данного анализатора составляют 80-90слов в обычных тестах (например, художественная литература, новости). В словарь анализатора Ю. С. Лукача входит 3,5 млн. словоформ русского языка, образованных от 205 тыс. базовых форм.

Словарь анализатора позволяет по словоформе получить набор ее базовых форм. Для большинства словоформ существует только одна базовая форма, однако для многих словоформ существует несколько базовых форм (например, форма «суда» может быть образована как от слова «суд», так и от слова «судно»). В данной работе автор отождествляет термины «слово» и «словоформа». Там, где это необходимо, используется термин «базовая форма слова».

При сохранении данных о вхождении слова в текст слово сначала приводится в своей базовой форме. Если базовых форм несколько, то информация добавляется для всех базовых форм.

Соответственно для каждой базовой формы слова существует список блоков для хранения информации о вхождении данной базовой формы в текстах.

При определенном размере блока мы можем сохранять в оперативной памяти последний блок для каждого из этих списков блоков. Этот блок сохраняется на диске, только когда он становится полностью заполненным.

Данный подход можно применять только для слов, входящих в словарь морфологического анализатора, т. к. мы знаем количество базовых форм и можем заранее создать кэш требуемого размера.

Слов, не входящих в словарь анализатора, может быть гораздо больше, например, в одной из используемых автором текстовых тестовых коллекций было 65 млн. различных слов, не входящих в словарь морфологического анализатора. О том, как обрабатывать такие слова, написано далее.

Кэширование для слов, не входящих в словарь морфологического анализатора

Подход, описанный в предыдущем подразделе, мы не можем использовать для слов, не входящих в словарь анализатора. В данном случае можно реализовать некие стандартные стратегии кэширования, например, при необходимости добавления нового блока в кэш выгружать из кэша блок, запись в который не производилась дольше всего.

Автором также был реализован подход, описанный далее. Слова, не входящие в состав морфологического анализатора, как правило, встречаются очень редко. Соответственно, суммарный объем информации об их вхождениях достаточно мал.

Предлагается сохранять информацию о вхождениях для нескольких слов в одном списке блоков. При этом, при записи информации о вхождении, вместе с ней сохраняется тег, определяющий, к какому слову она относится.

Данный метод значительно сокращает количество операций записи при создании индекса, при этом эффективность поиска снижается незначительно.

Например, пусть в одном списке блоков сохраняется информация о вхождениях слов: ааа, ббб, ввв. Присвоим этим словам номера 1, 2, 3. Если мы ищем слово ааа, то мы считываем весь этот список блоков. При чтении из списка блоков информации о каждом вхождении считываем также и тег, соответствующий данному вхождению. Рассматриваем только те записи, у которых тег равен 1.

При создании индекса контролируется, сколько раз уже встретилось слово. Если оказывается, что слово встречается слишком часто, то для него создается новый список блоков, в который переносится информация о вхождениях данного слова, далее в данном списке блоков будет сохранена информация только об этом слове.

Автором был проведен следующий эксперимент. Было зафиксировано число 4096, и для слов, число вхождений которых превышало данное число, создавались отдельные списки блоков.

Для тех слов, число вхождений которых меньше или равно 4096, в один список блоков помещалась информация сразу о 32 различных словах.

Обрабатывалось 35 Гб текстов, содержащих 69 млн. различных слов.

В результате оказалось, что максимальное количество записей в списке блоков, для списков, содержащих несколько слов, было равно 19 431.

Данное число показывает, что эффективность не пострадала, т. к. объем указанных списков блоков не превышает нескольких десятков килобайт (т. к. одна запись занимает примерно 2-4 байта).

Слов, которые встречаются больше чем 4096 раз, оказалось всего 19 593.

Для повышения быстродействия также используется алгоритм вставки пакета словоформ в B-дерево, подробно описанный в одной из работ автора диссертации.

Использование кэша при создании индекса

Для повышения производительности разделилим все базовые формы слов на группы, и записывать группы по отдельности. Определяем число K_GROUPS – количество групп, исходя из размера доступной оперативной памяти. Пусть K_FORMS – количество базовых форм известных слов (на данный момент около 200000), тогда:

1. число кластеров в кэше $GROUP\_SIZE$ = объем кэша / K.
2. $K\_GROUPS = K\_FORMS/GROUP\_SIZE$, с округлением в большую сторону.

Будем использовать кэш и для неизвестных словоформ. Задаем число U_GROUPS – число групп для неизвестных словоформ. Номер группы здесь определим с помощью хеш- функции $HASH$, которая, например, суммирует все коды символов словоформы и делит сумму на U_GROUPS.

Всего групп $GROUPS = K\_GROUPS + U\_GROUPS$. Введем $GROUPS$ временных файлов. Первые K_GROUPS файлов для известных словоформ, остальные для неизвестных словоформ, нумерация идет от нуля.

При чтении текстов информация о словах в текстах сохраняется во временных файлах. Информация для каждой группы базовых форм слов сохраняется в своем временном файле. После завершения чтения каждая группа обрабатывается и сохраняется в индексе отдельно, с использованием своего кэша.

Обработка часто встречающихся слов и схемы кодирования

Предлагается также новый подход для обработки наиболее часто встречающихся слов, который позволяет более эффективно выполнять точный поиск фраз, содержащих такие слова.

Рассмотрены схемы кодирования.

Основные теоретические результаты

Введем обозначения:

Пусть d – доля известных слов в текстах, R – текущее количество записей о вхождении в CLB-дереве.

K – размер кластера, H – размер страницы используемого B-дерева.

Теорема 1. Для вставки N записей о вхождении в CLB-дерево достаточно $O(d \cdot N/K + (1-d) \cdot N \cdot (1 + log_H((1-d) \cdot (R+N))))$ обращений к внешней памяти.

Теорема 2. Для поиска набора слов из N-элементов в CLB-дереве достаточно $O(N \cdot (log_H((1-d) \cdot R) + occ/K))$ обращений к внешней памяти (здесь occ – количество вхождений данных слов в текстах).

Теорема 3. Размер файла с кластерами для CLB-дерева составляет $O(S)$, где S – суммарный объем сохраненных в индексе данных.

На практике можно использовать не строгую оценку объема файла с кластерами: $d \cdot S \cdot 2 + (1-d) \cdot S \cdot 4$. Следует отметить, что данная оценка, хотя и не является характеристикой худшего случая, но весьма близка к нему, а в среднем в проведенных экспериментах получается меньший объем файла.

В третьей главе дается описание разработанного программного комплекса и приводятся результаты экспериментов. подтверждающие теоретические результаты и возможность применения данной структуры и алгоритмов на практике.

Рассматривается архитектура комплекса. Описываются возможности конфигурации.

Программный комплекс может индексировать файлы в различных форматах, например RTF, PDF, CHM, HTML, DJVU и кодировках, например UNICODE, UTF-8, CP-1251, ASCII, KOI-8. Поддерживается обработка архивов форматов ZIP, CAB, RAR, 7Z, ARJ, TAR, и др. Комплекс реализован в виде COM-сервера для операционных систем Windows.

Состав комплекса

1. Ядро, осуществляет создание индекса и поиск.

2. Модуль поддержки морфологии.

3. Модуль распознавания кодировки. При распознавании кодировки также учитывается морфология.

4. Модуль поддержки форматов файлов. Поддержка форматов файлов и архивов реализована с помощью подключаемых дополнительных модулей, которые могут быть реализованы в виде динамических библиотек или написаны на Java. Модуль поддержки форматов файлов реализован в виде отдельного процесса для повышения надежности системы.

5. Модуль атрибутов документов, для сохранения описания документов.

6. Модуль репозитария для сохранения текстов документов. Создается для того, чтобы при поиске можно было быстро получать фрагмент текста, содержащий найденную фразу. Тексты в репозитарии могут сохраняться с использованием различных алгоритмов сжатия.

7. Модуль COM осуществляет доступ к остальным модулям извне с помощью COM, что позволяет использовать программный комплекс в различных языках программирования.

Отдельно проведены сравнительные исследования производительности в 32-битной и 64-битной среде. Дается сравнение скорости работы алгоритма создания индекса с другими структурами данных. Приводится сравнение скорости работы с рядом широко используемых программных комплексов других разработчиков, предназначенных для решения рассматриваемой задачи. Приводится сравнение с инвертированными файлами – основной структурой данных, которая используется для построения поисковых индексов.

Проведенные эксперименты показывают, что разработанный программный комплекс строит индекс быстрее чем программы других разработчиков. При сравнении с инвертированными файлами эксперименты показали на порядок более быстрое обновление индекса, созданного с помощью CLB-дерева.

В четвертой главе рассмотрены вспомогательные компоненты и особенности реализации, в частности подсистема управления выделением памяти и подсистема интерфейса пользователя, на базе WindowSystemObject (WSO) – специальной библиотеки для создания интерфейсов пользователя, также разработанной автором работы.

Разработанный комплекс программ может применяться для поиска в сети Интернет, базах данных, системах документооборота, больших коллекциях электронных документов. Комплекс программ может быть полезен при организации поиска в быстро обновляющихся массивах текстовых данных, например, электронных газет и журналов.

ПУБЛИКАЦИИ ПО ТЕМЕ ДИССЕРТАЦИИ

Основные результаты диссертации опубликованы в следующих работах. Статьи, опубликованные в ведущих рецензируемых научных журналах, определенных ВАК:

1. Веретенников А. Б., Лукач Ю. С. Еще один способ индексации больших массивов текстов. Известия Уральского государственного университета. Серия «Компьютерные науки», 2006. №43. с. 103-122, 1,2 п. л. (лично автором 0,7 п. л.).

2. Веретенников А.Б. Эффективная индексация текстовых документов с использованием CLB-деревьев. Системы управления и информационные технологии, 2009, 1.1(35). - С. 134-139.

Другие публикации:

3. Веретенников А. Б., Лукач Ю. С. CLB-деревья: новый способ индексации больших массивов текстов. Международная алгебраическая конференция: К 100-летию со дня рождения П. Г. Конторовича и 70-летию Л. Н. Шеврина. Тез. докл. Екатеринбург: Изд-во Урал. ун-та, 2005, с. 173-175, 0,2 п. л. (лично автором 0,12 п. л.).

4. Веретенников А. Б. Новый подход к быстрому выделению памяти в программах на C++. Проблемы теоретической и прикладной математики: Труды 37-й Региональной молодежной конференции. Екатеринбург: УрО РАН, 2006, с. 413-417.

5. Веретенников А. Б. Эффективное создание текстовых индексов. Проблемы теоретической и прикладной математики: Труды 39-й Всероссийской молодежной конференции. Екатеринбург: УрО РАН, 2008. с. 348-350.

6. Веретенников А. Б. Создание легко обновляемых текстовых индексов. Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды Десятой Всероссийской научной конференции «RCDL'2008». Дубна: ОИЯИ, 2008. с. 149-154.

7. Веретенников А. Б. Библиотека для создания оконных интерфейсов на любых скриптовых языках в операционной системе Windows. Информационно-математические технологии в экономике, технике и образовании: Тезисы докладов Третьей международной научной конференции. Екатеринбург: УГТУ-УПИ, 2008. с. 220-221.

Подписано в печать 03.11.2009 г. Формат 60x84/16
Бумага офсетная. Уч.-изд. л. 1,5 Усл. печ. л. 1,5
Тираж 100 экз. Заказ №
Отпечатано в ИПЦ «Издательство УрГУ».
620000, г. Екатеринбург, ул. Тургенева, 4