Раздел II. Математические модели и методы

УДК 681.33

Ю.М. Вишняков, С.Н. Юрчук

МОДЕЛИ И МЕТОДЫ ПОСТРОЕНИЯ ИНДЕКСОВ ИНФОРМАЦИОННО-ПОИСКОВЫХ СИСТЕМ

Рассматриваются основные подходы к реализации инвертированного индекса информационно-поисковой системы (ИПС). Приводятся структуры данных для организации индексного хранилища и результаты исследования алгоритмов организации словарей. В статье освещены некоторые структуры данных для организации словарей и приведены результаты исследований их эффективности. Предлагается несколько способов организации инвертированного индекса. Приведены популярные алгоритмы сжатия данных и их анализ. Предлагается подход к выбору таких алгоритмов для сжатия инвертированного индекса.

Информационный поиск; индексирование; индекс ИПС; алгоритмы сжатия; организация словарей.

Y.M. Vishnyakov, S.N. Yurchuk

MODELS AND METHODS OF CONSTRUCTION OF INDEXES INFORMATION RETRIEVAL SYSTEMS

In this paper, the basic approaches to implementing an inverted index information-retrieval system (IRS). Provides a data structure for organizing the storage of the index and the results of the study of algorithms dictionaries organization. The article highlights some of the data structure for the organization of dictionaries and the results of studies of their effectiveness. There are several ways to organize the inverted index. Shows popular data compression algorithms and their analysis. An approach to the choice of algorithms to compress the inverted index.

Information retrieval; indexing documents; index IRS; compression algorithms; design dictionaries.

Под задачей информационного поиска понимается широкий класс подзадач. В настоящее время говорить об актуальности поиска информации не приходится в силу массовости использования популярных поисковиков таких, как Google, Yandex и т.д. Однако, в общем случае, задача поиска возникает не только при поиске в ВЕБ-е, но и в специализированных интеллектуальных системах документооборота различных организаций. При проектировании систем, поддерживающих функцию поиска информации, возникает ряд проблем, связанных с эффективностью реализации такой функции. Одной из главных проблем является время отклика системы на поисковый запрос пользователя. Для решения этой проблемы в системе создают образ документа, позволяющий существенно сократить скорость обработки запроса. В свою очередь, при создании такого образа возникает проблема его эффективного хранения и доступа к нему на внешнем устройстве хранения данных. Для решения этих проблем создается специализированная система или модуль системы, называемые – поисковым индексом (далее просто индекс). На сегодняшний день существует много различных подходов построения индексов. Среди них: инвертированный индекс, сигнатурный индекс, индекс, построенный на основе patricia-деревьев, и т.д. [1]. В данной работе речь будет идти об инвертированном индексе. Этот тип индекса исследовался авторами не только потому, что он явля-

ется самым популярным в использовании в настоящее время, но и потому, что он позволяет осуществлять поиск по всему тексту всех проиндексированных документов, другими словами, дает возможность полнотекстового поиска.

Модели поисковых индексов. Вначале рассмотрим общую структуру поискового инвертированного индекса. Введем понятие поискового образа документа. Образ документа – это информация, которая необходима поисковой системе для осуществления функции поиска [1]. В системе обычно каждому документу присваивается идентификационный номер. Набор образов документов и функции их обработки составляют поисковый индекс. Введем еще два понятия, нужных для определения образа документа. Лексема – это конкретное слово документа, ограниченное с двух сторон пробелами, причем в документе оно находится в конкретной позиции. Слово само по себе может содержать буквы алфавита естественного языка (ЕЯ) и специальные символы такие, как знаки препинания. В данной работе исследуются только слова, содержащие буквы ЕЯ. Терм – это определенное слово, которое может встречаться во всем индексе, т.е. он объединяет в себе все одинаковые лексемы. Например, лексема «информатика» встречается в нескольких документах по несколько раз. В каждом конкретном месте документа это будет лексема, а в целом – это терм «информатика». Терм «информатики» при этом является другим термом.

Сам инвертированный индекс представляет собой, в общем случае, список термов, которые встречались хотя бы в одном документе, список документов для каждого терма и список позиций для каждого терма в конкретном документе. Формально это можно представить следующим образом. Имеется набор <T, idDoc, <Pos1, Pos2, …, PosN> >, представляющий собой тройку из конкретного терма, документа и списка позиций, в которых этот терм встречается в данном документе (так называемые пост-листы). Данная модель структуры данных изображена на рис. 1.

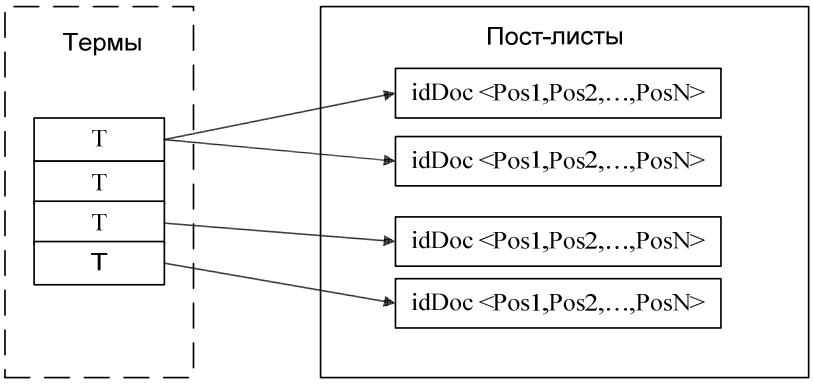

Рис. 1. Модель инвертированного индекса

Таких термов и соответствующих им пост-листов, естественно, может встречаться достаточно много. Как правило, индексы сохраняются в файлы. Здесь и встает проблема эффективности хранения данных. Если индекс сохранить сразу же в файл, то его объем будет просто не приемлем. Поэтому желательно сделать оптимизацию еще на уровне проектирования индекса. Рассмотрим список позиций терма в конкретном документе. Список позиций – это возрастающая последовательность чисел-позиций. Соответственно можно хранить не все числа в явном виде, а сократить их за счет разницы между соседними позициями. Таким образом, введем понятие длины интервала $L_i = P_i - P_{i-1}$. Кроме того, индекс нужно хранить в нескольких файлах с целью ускорения доступа к самому индексу. Приведем модель оптимизированного инвертированного индекса:

файл 1:
[countTerm] <$Term_i$, countDoc>
файл 2:
[countTerm] < [CountDoc(T_i)] < idDoc, countTerm(idDoc,T_i) > >
файл 3:
[countDoc] < [countTerm(T_i)$_{idDoc}$] < P_0,L_1,L_2,…,L_n > >
файл 4:
[countDoc] < offset$_{idDoc}$ >

Здесь квадратными скобками обозначены счетчики. Они в файл не записываются и здесь отображены для наглядности. Все, что находится в угловых скобках, записывается в файл, причем вложенность означает n-мерность. Данный индекс работает относительно быстро и хорошо сжимается, если использовать хорошие методы компрессии. Приведем еще одну модель индекса, которую можно использовать для небольших документов:

файл 1:
[countTerm] <$Term_i$, countDoc>
файл 2:
[countTerm] < [CountDoc(T_i)] < idDoc, countTerm (idDoc) > >
файл 3:
[countDoc] < [countTerm(idDoc)] < T_0,T_1-T_0,T_2-T_1,…,T_n-T_{n-1} > >
файл 4:
[countDoc] < offset$_{idDoc}$ >

То есть во втором случае мы храним уже не позиции, а сами термы, а точнее, разницу между ними для экономии памяти. Таким образом, скорость индексирования существенно повышается, а скорость поиска, естественно, падает.

Реализация словарей. В модели индекса использовались термы, а точнее, индексы термов. Здесь требуется функция получения индекса терма, исходя из самого терма. Имеется достаточно методов для реализации словарей с высокой скоростью работы. Один из популярных – это хеширование строк [2]. Здесь требуется $O(n) = 1 + \frac{\alpha^2}{2}$ в случае неудачного поиска и $O(n) = 1 + \frac{\alpha}{2}$ в случае успешного поиска, где α – коэффициент заполнения таблицы. Метод хеширования – один из самых эффективных методов, но у него есть недостаток. Сами строки не сохраняются, что может вызвать проблемы в некоторых специализированных функциях, например, нечетких запросах [1]. Можно также использовать B-деревья [2], здесь оценка $O(n) = 2 + log_{m/2}\left\lceil \frac{n}{b} \right\rceil$, где m – порядок дерева, b – количество записей в листе. Есть еще более эффективное решение – это использование префиксных деревьев [3] или похожая структура данных – m-путевое дерево [2]. Такая структура позволяет искать слово с количеством операций, равным длине слова. Но эту структуру нужно несколько модифицировать в связи с тем, что у нее есть два существенных недостатка. Первый состоит в том, что узлы дерева имею ограниченную длину символов. Второй недостаток заключается в том, что из-за существования множества узлов, содержащих мало букв, тратится довольно много памяти попусту, причем настолько много, что к использованию структуры в таком явном виде прибегают редко. Авторами предлагается использовать TST (ternary search trie)-дерево [2]. Данное дерево позволяет выполнять операцию поиска-вставки за $O(n) = log_m n + O(n)$. Это время почти совпадает со временем m-путевых деревьев. Разница состоит в том, что в TST-дереве на каждом уровне нужно пройти

некоторое количество букв для перехода на следующий уровень. Результаты с данными о среднем количестве операций поиска слова в таком дереве приведены в табл. 1. Эксперимент проводился на тексте объемом ~ 500 млн лексем. Общее количество слов, используемых при поисковых запросах, ~ 3 млн. Из результатов эксперимента видно, что даже при неудачном поиске TST-дерево довольно быстро находит слова в словаре. При этом устраняются указанные выше недостатки.

Таблица 1

Кол-во. букв в слове/ эксп.	Уд. поиск	Неуд. поиск	Уд. поиск	Неуд. поиск	Уд. поиск	Неуд. поиск
1	4	0	4	0	4	0
2	6	0	7	0	7	0
3	7	19	9	15	9	15
4	11	19	11	19	11	19
5	13	19	13	19	13	20
6	15	20	15	20	15	20
7	16	20	16	20	16	20
8	17	20	17	20	17	20
9	18	21	18	21	18	21
10	20	21	19	21	19	21

Сжатие индекса. Возвращаясь к модели индекса, нужно отметить, что в образе документа, а именно позиций, находится довольно много избыточной информации. Для сокращения этой информации необходимо использовать эффективные алгоритмы компрессии. Причем их эффективность должна состоять не только в плане сжатия, но и в плане скорости. Алгоритмы компрессии бывают универсальные и энтропийные [4]. Универсальные алгоритмы подразделяются на параметрические и непараметрические. Параметрические алгоритмы выгодно использовать, когда известны характеристики входных данных. Энтропийные методы сжимают достаточно хорошо без этой информации, но требуют больше вычислительных затрат.

Приведем на рис. 2 анализ некоторых универсальных непараметрических кодов. Нас интересует длина кода в битах в зависимости от входного числа. Ниже в табл. 2 даны функции представленных кодов.

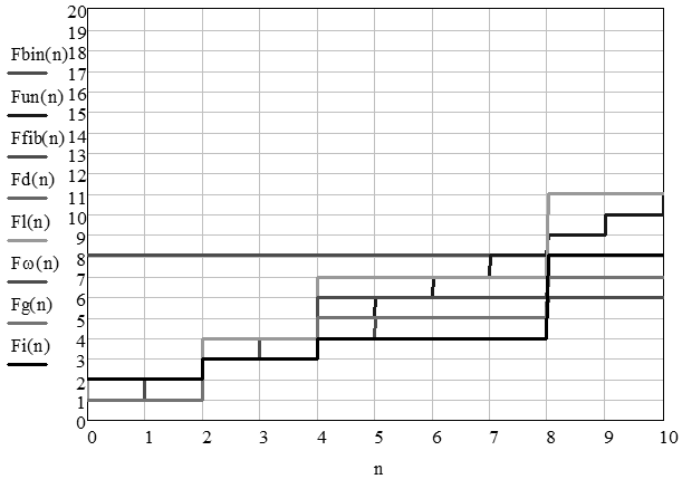

Рис. 2. График аналитических функций универсальных кодов

Таблица 2

№	Название кода	Название функции
1	Байтовое кодирование	Fbin(N)
2	Унарное кодирование	Fun(N)
3	Код Фибоначчи	Ffib(N)
4	Дельта код	Fd(N)
5	Код Левенштейна	Fl(N)
6	Омега-код Элиаса	Fω(N)
7	Гамма-код Элиаса	Fg(N0
8	Код Ивэна-Роде	Fi(N)

Из графика функций видно, что самый предпочтительный код, если значения маленькие, это гамма-код либо код Фибоначчи. Последний эффективно работает и при больших числах. Но в случае, например, второй модели индекса, когда числа будут в своем большинстве не маленькие, лучше использовать параметрический код Райса [4]. Он, во-первых, работает быстрее, чем энтропийные коды и кодирование Голомба [4], во-вторых, дает сжатие кодов в 1,5 раза лучше универсальных кодов. Также для второй модели неплохо подойдет кодирование Хаффмана [4].

В данной работе были рассмотрены основные подходы к организации инвертированных индексов. Было приведено два подхода к организации индексов. Рассмотрены и предложены структуры данных организации словарей. Приведены результаты экспериментов по сжатию индексной информации и соответственно сделаны выводы о выборе наиболее эффективных алгоритмов компрессии.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Маннинг Кристофер Д.* Введение в информационный поиск / Маннинг Кристофер Д., Рагхаван Прабхакар, Шютце Хайнрих: Пер. с англ. – М.: ООО «И.Д. Вильямс», 2011. – 528 с.
2. *Кнут Дональд Э.* Искусство программирования. Т. 3. Сортировка и поиск / Кнут Дональд Э. – 2-е изд.: Пер. с англ. – М.: Вильямс, 2005. – 824 с.
3. *Вишняков Ю.М.* Системное программирование: Конечные распознаватели / Вишняков Ю.М.: Учебное пособие. – Таганрог: ТРТИ, 1991. – 74 с.
4. *Salomon D.* Data compression / Salomon David – 3 edt. – Northridge, CA, USA: Springer, 2004. – 904 с.

Статью рекомендовал к опубликованию д.т.н., профессор Е.А. Башков.

Юрчук Станислав Николаевич
Вишняков Юрий Мусович
Технологический институт федерального государственного автономного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: vishn@tsure.ru.
347928, г. Таганрог, пер. Некрасовский, 44.
Тел.: 88634371885.

Yurchuk Stanislav Nikolaevich
Vishnyakov Yuriy Musovich
Taganrog Institute of Technology – Federal State-Owned Autonomy Educational Establishment of Higher Vocational Education “Southern Federal University”.
E-mail: vishn@tsure.ru.
44, Nekrasovskiy, Taganrog, 347928, Russia.
Phone: +78634371885.