Раздел IV. Математические методы искусственного интеллекта

УДК 681.3.07

И.С. Злыгостев

ПРОСТРАНСТВЕННО-ВРЕМЕННАЯ ОПТИМИЗАЦИЯ СТРУКТУРЫ ДАННЫХ ЕСТЕСТВЕННОГО ЯЗЫКА С ДОСТУПОМ ПО КЛЮЧАМ

В работе проведена оптимизация структуры PATRICIA-дерева в контексте использования ее для хранения данных, ключами доступа к которым являются слова русского языка с целью уменьшения объема, занимаемого структурой при минимальных потерях в скорости работы с ней. Оптимизация проведена на основе данных статистически словаря русского языка. Приведены алгоритмы итераций по ключам структуры, отсортированным в лексикографическом порядке. Оптимизация структуры привела к сокращению размера дерева в 25 раз и сделала доступным использование структуры как с целью быстрого поиска, так и для управления содержащимися в ней данными.

Структура данных; PATRICIA дерево; статистика; словарь; русский язык; лексикографический порядок; итерация; оптимизация по объему.

I.S. Zlygostev

SPATIO-TEMPORAL OPTIMIZATION OF DATA STRUCTURE IN NATURAL LANGUAGE WITH ACCESSING BY-KEYS

In this work the PATRICIA-tree structure optimization in the context of using it for data storage was investigated. The access keys for data are the words of Russian language. Optimization made by volume contraction of structure with the minimum loss of work speed. Optimization was done on basis of statistical data of Russian language vocabulary. The algorithms of direct and reverse iterations by structure keys that sorted lexicographically were considered. Research makes it possible to decrease the structure size 25 times and makes available the usage of the structure not only for the quick search, but also for the full data control.

Data structure; PATRICIA-tree; statistic optimization; dictionary; Russian language; lexicographical grading; iteration on tree; volume optimization.

Автоматизация задач обработки текста на естественном языке привела к развитию компьютерной лингвистики [3]. В ней проводится формализация модели правил естественного языка. Существует целый ряд задач автоматизированной обработки текста, для реализации которых необходимо хранение и использование словаря естественного языка [2]. В частности, такими задачами являются системы автоматического перевода, поисковые системы, различного рода словари, системы извлечения знаний из текста, системы лингвистической обработки текста, генераторы естественной речи, интеллектуальный системы text-to-speech. Чем глубже и точнее проводится в системе лингвистический и смысловой анализ текста, тем вероятней, что система содержит словарь языка, с которым она работает. Слова в таких словарях являются ключами доступа к некоторым данным системы. Так, для компьютерного морфологического анализа языка такими данными могут быть

морфологические характеристики слова. В поисковых системах слову ключу соответствует строчка инвертированного индекса со списком оценки значимости слова для документов, по которым поисковая система ведет поиск.

Количество слов развитого естественного языка достаточно велико [1]. Современные поисковые системы насчитывают в своих словарях порядка 10 миллионов слов русского языка. Различные по написанию слова в тексте могут являться словоформами одной леммы языка [1]. Количество слов в словаре естественного языка, как правило, значительно превышает количество лемм языка, и разница между этими двумя величинами тем сильней, чем развитей морфология словообразования языка. Морфология русского языка считается весьма развитой в сравнении с морфологией других языков [2]. Словоформы одной леммы отличаются аффиксами [7]. Аффиксы бывают левыми и правыми. Множество аффиксов естественного языка ограничено и количество элементов в нем незначительно в сравнении с количеством лемм в языке. Таким образом, существует множество слов в словаре отличающихся только аффиксами.

Современный этап развития программных средств ведет к построению программ на основе уже имеющихся компонент. Компоненты реализуют некоторые функциональности и предоставляют интерфейс для их использования. Одна компонента может использоваться несколькими программами, запущенными на различных компьютерах и предоставлять им сервис использования своих функциональностей параллельно.

Развиваются программные компоненты, реализующие задачи компьютерной лингвистики, развиваются системы управления базами данных. В такого рода компонентах может обслуживаться множество запросов и содержаться большие массивы данных. Возникает задача оптимального использования ресурсов оперативной памяти компьютера и времени процессора при работе со структурами-контейнерами в качестве ключей доступа к данным, которых используются слова естественного языка.

В качестве реализации структуры могут быть выбраны массивы, списки, векторы с записями вида слово-ключ и соответствующими ему данными. Также структурами могут быть выбраны карты, хэш-массивы [6], различные реализации деревьев [4] с ключами доступа к данным в виде слов естественного языка.

Электронный «Грамматический словарь русского языка» А.А. Зализняка [1] является одним из наиболее полных открытых источников перечня всех словоформ лемм [2] русского языка в цифровом виде. Данный факт послужил причиной выбора его в качестве основы для исследования статистических характеристик слов русского языка и источника слов-ключей для проведения экспериментов над исследуемыми в работе структурами данных.

Автором получена следующая общая статистика о словаре: словарь содержит W_{all} = 3469277 словоформ W_{nf_all} = 100000 лемм русского языка, максимальная длина слова в словаре W_{len} = 27 букв, среднее количество словоформ одной леммы словаря W_{aver_wf} = 34,69. Без учета регистра букв слова, все слова в словаре построены на основе 34-х символов. Размер словаря 45 876 КБ.

Для упрощения процедуры подсчета размера памяти, занимаемого различными реализациями структур данных, которые возможно использовать для хранения словаря Зализняка, определим размер в словаре Зализняка, занимаемый символами слов. В файле пустые строки между блоками слов в словаре занимают W_{nf_all}*2 = 100 000(слов)*2 = 200 000 байт ≈ 195 КБ. Символы окончания строк в файле занимают W_{all}*2 = 3 469 277*2 = 6 938 554 байт≈6775 КБ. То есть символы, составляющие слова словаря в файле имеют размер C_{size}=45876-6755-200=38921 КБ.

Получены численные характеристики распределения количества слов в словаре по их длине. Ниже приведена табл. 1 с количеством и долей слов определенной длины в «Грамматическом словаре русского языка» А.А. Зализняка.

Таблица 1

Распределение слов в словаре Зализняка по количеству символов

Длина слова	Количество слов	Процент слов	Длина слова	Количество слов	Процент слов
1	27	00,0008 %	15	217280	06,2630 %
2	279	00,0080 %	16	133446	03,8465 %
3	3317	00,0956 %	17	78427	02,2606 %
4	13662	00,3938 %	18	42439	01,2233 %
5	39538	01,1397 %	19	23481	00,6768 %
6	83785	02,4151 %	20	12105	00,3489 %
7	161169	04,6457 %	21	6887	00,1985 %
8	252656	07,2827 %	22	3033	00,0874 %
9	338656	09,7616 %	23	1437	00,0414 %
10	424512	12,2363 %	24	695	00,0200 %
11	453602	13,0748 %	25	227	00,0065 %
12	461646	13,3067 %	26	145	00,0042 %
13	401986	11,5870 %	27	28	00,0008 %
14	314812	09,0742 %	28	0	00,0000 %

В результате выявлено, что распределение слов по числу символов в словаре нормальное и средняя длина слова в словаре $W_{aver\_len} = 11{,}49$ символа.

В силу достаточно большого объема содержащихся в «Грамматическом словаре русского языка» А.А. Зализняка слов – можно считать приведенную статистику приблизительным отражением распределения слов по длине в русском языке.

Перед началом исследования структур, необходимых для хранения данных словаря, сформулируем предъявляемые к их реализации требования.

1. Возможность внесения данных по схеме «строка-ключ / значение».
2. Наличие программного интерфейса по внесению элементов в контейнер.
3. Наличие программного интерфейса для удаления элементов из контейнера.
4. Поиск элементов в контейнере.
5. Проверка наличия элементов в контейнере.
6. Интерфейс для прямого и обратного итеративного перехода по элементам контейнера. Итератор организует переход по элементам контейнера, упорядоченного в лексикографическом порядке по значениям строки-ключа.
7. Определение конечного и начального итератора.

Небольшой алфавит символов, сравнительно небольшая средняя и максимальная длина слова и большое количество повторяющихся начальных частей слов привели автора работы к мысли о возможности применения для хранения словаря слов русского языка в методе морфологического анализа древовидной структуры типа patrie-дерева.

Для аргументации выбора проведем характеристики основных альтернативных структур данных.

Если строки со словами словаря будут храниться в минимальной для несжатого хранения строк структуре C-строках, они будут занимать $S_{c\_st} = C_{size}+W_{all}\approx 42308$ КБ памяти.

Определимся с данными, хранимыми в структурах в проводимых исследованиях. Пусть ключами поиска в контейнерах будут С-строки со словами на русском языке, а в качестве данных будут 4-байтовые ссылки. Ссылка может быть на данные любого вида. То есть такая структура универсальна к данным словаря. Будем вычислять в рамках исследования размеры некоторой реализации структуры вышеописанного вида, инициализированной данными по всем словами-ключами из словаря.

Линейный список

Рассмотрим вариант организации словаря словоформ русского языка в виде двусвязного списка. В словаре должны быть 2 ссылки на предыдущий и следующий элемент списка + слово + ссылка на данные. Без учета слова структура записи в контейнере имеет размер 2*4 байта + 4 байта = 12 байт. Размер всего списка $S_{list} = 12*W_{all}+S_{c\_str}$ = 82964 KB = 81 MB.

Сложность алгоритмов поиска нужного элемента в списке O(N* M), где N – количество словоформ в словаре, M – длина слова. Операция вставки или удаления элемента в упорядоченном списке производится алгоритмом со сложностью O(N*M). Вставка или удаления указанного элемента, будет производиться в одну операцию после поиска его места в списке. На базе списка нет проблем в реализации итераторов.

Массив записей

Элементами массива являются структуры данных вида {Строка, ссылка на данные}. По причине оптимальности хранения строк в С-формате, элементы массива будут иметь разную длину. В этом случае массив словоформ слов русского языка может быть оптимально организован в виде массива ссылок на структуры данных.

Посчитаем размер массива. Он будет составлен из суммы размера массива ссылок на элементы массива, размера ссылок на данные, и строки:

$$S_{ar}=8*W_{all}+S_{c\_str} = 69412 \text{ KB} = 68 \text{ MB}.$$

В процессе инициализации массива может возникнуть проблема в выделении участка памяти размером $8*W_{all}$ = 27 MB единым сегментом по причине фрагментации.

Операция поиска слова из словаря в оптимальной реализации строкового массива имеет сложность $O(\log(N^M))$ (метод дихотомии). Здесь N – количество словоформ в словаре, M – длина слова.

По причине необходимости переноса всех последующих элементов после вставки или удаления какого-нибудь элемента массива, операции вставки и удаления элементов будут иметь сложность $O(\log(N^M)+N)$. На базе массивов нет проблем в реализации итераторов.

Patricia-дерево

Patricia-дерево [5] составлено из вершин. Вершины соединены двунаправленными связями. В каждой вершине могут храниться данные структуры, есть одна исходящая связь на узел предка и n ссылок на все возможные дочерние узлы. Здесь, n = [*число символов в алфавите*]+1. Если дочерних узлов нет, то ссылка обнулена.

Каждая значимая ссылка от узла отца к узлу сына помечается символом перехода. У графа есть корневая вершина. У нее нет вершины отца. Переходя от корневой вершины по ссылкам ветви Patricia-дерева и собирая в кортеж символы, соот-

ветствующие последовательным переходам в дереве, осуществляется поиск вершин, соответствующих кортежам ключей. В данной работе в качестве таких кортежей-ключей используются слова русского языка.

Получается, что все строки patricia-дерева, имеющие общее начало, располагаются в одном поддереве. Каждое ребро помечено некоторым символом. Терминальным вершинам («листьям») соответствуют некоторые слова-ключи структуры (рис. 1).

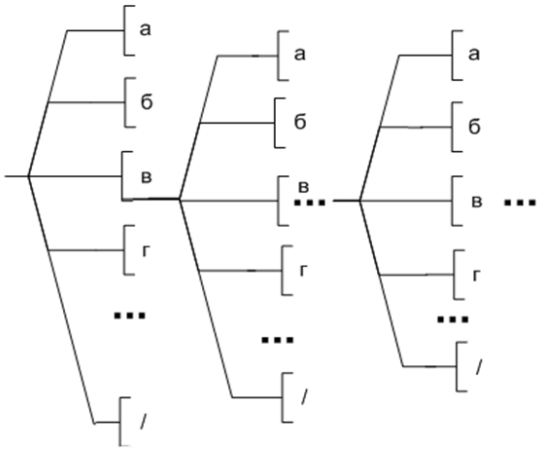

Рис. 1. Структура связей между узлами в patricia-дереве

Таким образом, получается, что в программной модели, если мы используем входную С-строку в качестве ключа, то каждый узел дерева имеет 256 ссылок на дочерние узлы. Для определенности назовем программный класс, реализующий функциональности по работе с одной вершиной CTreeNode. Класс, организующий работу с деревом в целом и содержащий вершину корня, назовем CTree. В этих двух классах можно обеспечить реализацию работы с patricia-tree.

Для вставки нового элемента по ключу в дерево необходимо:

a) найти максимальную подстроку в дереве, соответствующую ключу, начиная с начала слова;
b) дополнить подстроку до полного совпадения с ключом;
c) произвести вставку значение в конечный узел строки ключа.

Сложность первой и второй операции в patricia-деревьях O(M). Третья операция производится в одно действие. Таким образом, сложность всей операции вставки элемента в patricia-дерево O(M).

Для удаления элемента из структуры необходимо:

a) найти конечную вершину ключа;
b) произвести удаления данных в узле, по необходимости с цепочкой пустых узлов, следующих к нему.

Сложность обеих операций O(M). Следовательно, и сложность всей операции удаления элемента из patricia-дерева O(M).

В рамках исследования проведена инициализация классической реализации patricia-дерева данными по словам-ключам словаря А.А. Зализняка.

В такой реализации patricia-дерева слова-ключи составлены из однобайтовых символов. Каждый узел дерева содержит следующие данные:

- Массив указателей на корни узлов поддеревьев детей (1024 байта).
- Указатель на данные дерева (4 байта).

♦ Указатель на узел родителя (4 байта).

То есть каждый узел дерева занимает V_{node_size} = 1032 байта.

В случае отсутствия у рассматриваемого узла дочерних узлов, узла родителя или значения – ссылки на соответствующие элементы обнулены.

Доступ к patricia-дереву производится по уникальным ключам. В русском языке слова омонимы могут иметь одинаковое написание, но разное значение. Проблему использования patricia-деревьев в словарях, содержащих слова-омонимы легко решить, поместив в качестве значений дерева, к примеру, массив или список возможных значений по ключу.

При тестировании patricia-дерева использовались только уникальные ключи. Поэтому слова омонимы в текстовый вариант patricia-дерева не были включены.

В результате тестирования классической реализации patricia-дерева на словах электронного словаря А.А. Зализняка получены новые статистические данные. Структура дерева содержит T_{nodes} = 3 239 042 узлов. Число омонимов в словаре – $N_{homonyms}$ = 1 339 701. Слова омонимы составляют 38,62 % словаря русского языка. Количество узлов, содержащих данные T_{values} = 2 129 576. То есть в словаре содержится T_{values} уникальных по написанию слов.

Структура данных, содержащая только словарь Зализняка, имеет размер V_{node_size} * T_{nodes} = 3 342 691 344 bytes = 3 188 MB = 3,1 Gb. Это величина на 2 порядка превышает размер других реализаций структур данных хранения словаря А.А. Зализняка.

Несмотря на хорошие характеристики в скорости поиска, удаления и вставки элементов классической реализации patricia-дерева, структура признана автором непригодной для использования в хранении больших объемов данных по словам-ключам русского языка. Сравнительно большой объем используемой оперативной памяти вводит жесткие ограничения на применение данной структуры в реальных задачах.

Тем не менее высокая скорость работы, с инициализированными данными структуры в контексте решаемых в работе задач, привела автора к мысли о возможности оптимизации структуры. Действительно, скорость доступа к данным по слову-ключу в дереве на порядки выше всех рассмотренных реализаций структур данных решающих подобную задачу.

При реализации и исследовании структуры данных классического patricia-дерева, автором был получен ряд результатов, которые смогут помочь при оптимизации структуры.

Пусть размер данных, помещенных в узлы дерева равен V_{size}. В рамках тестирования V_{size} = 4 байта. Формула, по которой можно определить размер памяти в структуре patricia-дерева, содержащей данные $T_{value\_size} = T_{values} * V_{size}$. Тогда данные в patricia-дереве, составленном из слов-ключей словаря А.А. Зализняка займут всего T_{value_size} = 8 518 304 байта = 8,12 MB. Размер классической реализации patricia-дерева 3188 MB. Таким образом, основная часть рабочего пространства памяти занята под структуру ключей дерева.

В классической реализации patricia-дерева в каждом дереве присутствует 256 ссылок на дочерние элементы дерева. Возможно, что словарь дерева будет меньших размеров. Все же не факт, что алфавит дерева будет расположен в одном интервале ASCII символов и что он вместится в диапазон 256 элементов.

Оптимизация patricia-tree сокращением словаря символов структуры

Программный модуль patricia-дерева был снабжен дополнительным классом CSymbolsManager, реализующим работу с выборкой из всех возможных символов

в типе char (размер byte) только тех, с которыми предполагается работать patricia-дереву.

Алфавит patricia-дерева считывается из файла при инициализации класса. Всем символам словаря присваиваются внутри класса в соответствие номера таким образом, что каждый символ имеет уникальный номер, номера символов следуют подряд и начинаются с единицы.

В программном модуле оптимизированного patricia-дерева присутствует всего 1 экземпляр класса CSymbolsManager. Ссылка на него есть во всех узлах дерева. Цель работы класса – делать прямое и обратное преобразование ASCII символов d соответствующего им номера. В результате добавления CSymbolsManager число ссылок в узлах дерева уменьшилось по причине уменьшения размера алфавита словаря.

Тестирование программного модуля проводилось на дереве, использующем словарь размером в 61 символ (русские, английские буквы нижнего регистра и символы “/”, ”-“).

Было реализована функциональность в дереве, делающая его регистронезависимым. Все слова-ключи – переводятся в нижний регистр перед использованием в дереве.

Таблица 2

Число узлов, соответствующих некоторому символу в patricia-дереве. Дерево составлено на основании словаря А.А. Зализняка

Символ	Найдено в дереве	Процент узлов	Символ	Найдено в дереве	Процент Узлов
«с»	571522	17,645 %	«ш»	37535	01,159 %
«я»	535206	16,524 %	«л»	26200	00,809 %
«е»	312693	09,654 %	«к»	22207	00,686 %
«м»	201171	06,211 %	«щ»	10412	00,321 %
«и»	194572	06,007 %	«р»	9564	00,295 %
«й»	176957	05,463 %	«ч»	6326	00,195 %
«ь»	155384	04,797 %	«ц»	4525	00,139 %
«о»	151724	04,684 %	«д»	3890	00,120 %
«у»	151270	04,670 %	«з»	3104	00,096 %
«ю»	141641	04,373 %	«п»	2329	00,072 %
«а»	127725	03,943 %	«ж»	2205	00,068 %
«х»	99021	03,057 %	«б»	2020	00,062 %
«т»	87185	02,692 %	«ф»	937	00,029%
«г»	60283	01,861 %	«-»	406	00,013 %
«ы»	50734	01,566 %	«ё»	144	00,004 %
«н»	45169	01,395 %	«э»	117	00,004 %
«в»	44793	01,383 %	«ъ»	70	00,002 %

После оптимизации структура дерева в памяти заняла 823,62 MB. Таким образом, объем памяти, занимаемый структурой дерева, уменьшился почти в 4 раза. В силу использования в CSymbolsManager таблиц переходов, скорости работы алгоритма за счет оптимизации фактически не изменилось. Все узлы дерева, как и в прошлой реализации, содержат одинаковое количество ссылок.

Тем не менее размер структуры на порядок больше размеров списков и массивов, содержащих те же элементы, что и в patricia-дереве. Применение структуры

дерева для использования ее в лингвистических модулях с большим числом словключей остается затруднительным.

Для оптимизации полученной структуры, автор работы провел дополнительные статистические исследования, результаты которых представлены ниже в виде двух таблиц. Таблицы исследуют структуру дерева с хранящимися в нем данными словаря А.А. Зализняка.

Таблица 2 содержит численные характеристики, отражающие количество узлов дерева, соответствующих определенной букве алфавита.

Таблица 3 отражает количественные характеристики связей в дереве. В таблице представлены данные о числе узлов patricia-дерева, содержащих ссылки на определенное количество сыновей. Дерево составлено на базе данных из электронного словаря Зализняка.

Таблица 3

Количество узлов patricia-дерева с заданным числом значимых связей в нем. Дерево составлено на основании словаря А.А. Зализняка

Число Связей	Узлов в дереве	Процент Узлов	Число Связей	Найдено в дереве	Процент Узлов
0	1258631	38,85812 %	17	34	0,00105 %
1	1481678	45,74433 %	18	32	0,00099 %
2	198904	6,14083 %	19	20	0,00062 %
3	40040	1,23617 %	20	26	0,00080 %
4	124821	3,85364 %	21	20	0,00062 %
5	103126	3,18384 %	22	19	0,00059 %
6	16954	0,52343 %	23	7	0,00022 %
7	5721	0,17662 %	24	17	0,00053 %
8	6152	0,18993 %	25	6	0,00019 %
9	1022	0,03155 %	26	6	0,00019 %
10	880	0,02717 %	27	6	0,00019 %
11	368	0,01136 %	28	7	0,00022 %
12	207	0,00639 %	29	5	0,00015 %
13	134	0,00414 %	30	2	0,00006 %
14	77	0,00237 %	31	0	0,00000 %
15	66	0,00204 %	32	0	0,00000 %
16	54	0,00167 %	33	0	0,00000 %

На основании того, что электронный словарь А.А. Зализняка – один из наиболее полных источников перечня всех словоформ слов русского языка, следует предположить, что представленная автором работа в таблицах процентная статистика достаточно точно отражает предположительную долю узлов patricia-дерева, составленного по всем словам русского языка во всех их словоформах.

Оптимизация patricia-дерева отбрасыванием ссылок на дочерние узлы в листьях дерева

На основании анализа данных, приведенных в таблицах, автор задался мыслью об оптимизации структуры хранения ссылок на дочерние узлы. В дереве, в 84% случаев узлы либо не имеют дочерних узлов, либо имеют один. Вероятность встречи различных символов в узле – различна.

В классе реализации узла CTreeNode, вносятся изменения в структуру patricia-дерева. Вместо статического массива ссылок на дочерние узлы вводится ссылка на динамическую структуру данных, хранящую эти узлы.

Если у узла нет дочерних, то ссылка на структуру обнуляется. В результате, мы на порядок уменьшаем размер (39%) узлов дерева (табл. 3).

Оптимизация управления памятью в узлах patricia-дерева

Следующая оптимизация произведена в реализации класса управления символами в узлах дерева CSymbolsManager. Ранее упоминалось, что символы входного алфавита дерева индексируются в классе, начиная с единицы. Приведем описание структуры данных ссылок, которыми управляет класс.

Структура представлена в виде упорядоченного массива одноразмерных элементов – ссылок. Номер ссылки в массиве соответствует индексу символа алфавита в CSymbolsManager, по которому осуществляется переход. Ссылки на несуществующие дочерние узлы обнулены.

Добавим в массив ссылок элемент с нулевым индексом. Пусть он содержит самый большой номер элемента массива, содержащего не нулевую ссылку. По сути, в элементе с индексом 0 массива содержится длина массива со значащими ссылками. Если искомая ссылка с узла по букве имеет индекс меньший числа, содержащегося в элементе массива с индексом 0, то ссылку можно не искать в массиве – ее значение 0.

Получается, если под динамически инициализированную структуру данных выделять памяти ровно до последнего значащего элемента в массиве, т.е. размером, указанным в элементе с индексом 0 массива ссылок, то структура останется полнофункциональной, хотя уменьшится в размерах.

При реализации вышеописанных изменений возникли небольшие проблемы в методе по добавлению новых ссылок в существующий динамический массив: если индекс добавляемой ссылки превышает значение нулевого элемента в массиве, то возникает необходимость в динамическом расширении памяти в массиве. По причине того, что эффективней для организации быстрой скорости работы с массивом память под элементы в массиве выделять одним блоком, возникла необходимость в выполнении ряда дополнительных операций. Для добавления новой ссылки в массив необходимо:

1. Создать новую структуру данных для ссылок, размер которой определяется индексом новой добавляемой ссылки.
2. Скопировать старую структуру данных в новую.
3. Изменить нулевое значение ссылки на индекс добавляемого элемента.
4. Поместить новую ссылку в массив ссылок.
5. Освободить память, занимаемую старой структурой.

Копирование и освобождение небольших блоков памяти, таких как в рассматриваемом случае (4 байта * 32 символа = 128 байт), для современных ПК не трудоемкая операция, она займет сравнительно не очень много времени. Операция вставки и удаления элементов в новой структуре данных останется все также на порядок быстрее, чем вставка элементов, к примеру, в упорядоченный массив.

Оптимизация patricia-дерева изменением порядка символом алфавита в узлах

В рамках описываемой реализации patricia-дерева произведена еще одна оптимизация. Она касается переупорядочивания символов входного алфавита. Как следует из табл. 2, встреча различных букв в переходах с узла на узел не равновероятна. Инициализируем в CSymbolsManager алфавит, упорядоченный по убыванию частоты встреч узлов в дереве (табл. 2). В результате получается, что наиболее вероятный для встречи в узле переход по символу будет иметь меньший индекс в массиве ссылок на дочерние узлы.

Цель оптимизации порядка алфавита – уменьшить размер дерева. В результате применения переупорядоченного алфавита, значения максимальных индексов в узлах дерева в массиве ссылок на дочерние деревья стали в среднем значительно меньше. По причине оптимизации алгоритма работы с памятью динамического массива ссылок в узле (смотрите первую оптимизацию для структуры хранения ссылок на дочерние узлы в этом разделе), максимальный индекс в узлах определяет размер, занимаемый массивом ссылок в узле. Чем меньше максимальный значимый индекс в массиве ссылок, тем меньше и размер структуры со ссылками в целом.

Приведем результаты тестирования реализации оптимизаций структуры хранения ссылок на дочерние узлы. Размер, занимаемый структурой, уменьшился в 6,59 раза в сравнении с прошлой оптимизацией и в 25,5 раз в сравнении с классической реализацией patricia-дерева. Время инициализации и деинициализации новой структуры данных уменьшилось в 2,5 раза в сравнении с прошлой реализацией дерева. Оптимизированный размер patricia-tree структуры составил 125 MB.

Открытым остается вопрос об организации итеративного процесса по словам-ключам структуры, отсортированным в лексикографическом порядке. Автор не нашел в открытых источниках реализации данного алгоритма и предложил свой. Приведем его.

Алгоритм прямого шага итератора по patricia-дереву

Если текущий итератор не лист ветви дерева, до тех пор, пока текущий итератор не попадет в узел, содержащий значение.

- Выбираем первую дочернюю ветвь поддерева в узле и делаем ее корень текущим узлом итератора.

По завершению цикла отдаем итератор на узел. Операция завершена.

Иначе – текущий итератор лист ветви дерева.

Пока у текущего узла итератора есть родитель и узел соответствует последнему поддереву в списке поддеревьев родителя.

Делаем узел родитель текущим.

a. Если родителя больше нет – то узел, который был в вызове прямого шага итерации, является конечным, его и возвращаем, как результат работы алгоритма.

b. Если ветвь на текущий узел не последняя в поддеревьях узла родителя, переходим к следующей ветви родителя и выполняем цикл.

До тех пор, пока текущий итератор не попадет в узел, содержащий значение.

- Выбираем первую дочернюю ветвь поддерева в узле и делаем ее корень текущим узлом итератора.

По завершению цикла отдаем итератор на узел. Операция завершена.

Алгоритм обратного шага итератора по patricia-дереву

Ищем у родителя поддерево, ссылающееся на текущий узел.

Пока поддерево на текущий узел у родителя является первым в списке и в узле-родителе не содержится значения.

Ищем у родителя поддерево, ссылающееся на текущий узел.

- Если родителя нет, то мы при входе в алгоритм находились в начальном узле дерева. И, следовательно, возвращаем результат неизмененный входной итератор.
- Если поддерево на текущий узел у родителя является первым в списке и в узле-родителе не содержится значения, устанавливаем узел родитель текущим в итераторе.

- Если поддерево на текущий узел у родителя не является первым в списке:
  1. Выбираем предыдущее поддерево в списке родителя.
  2. Устанавливаем его корневой узел текущим в итераторе.

  Пока текущий элемент в итераторе не будет листом в дереве.
  а. Выбираем самое последнее (правое) поддерево текущего узла и переходим по нему в дочерний узел, устанавливая его текущим.
- Если текущий узел является листом дерева, возвращаем итератор на него, как результат работы алгоритма.
- Если поддерево у родителя на текущий узел итератора является первым в списке поддеревьев родителя и в узле-родителе есть значение – передаем в качестве результата работы алгоритма итератор на узел родителя.

Выводы

Структура patricia-дерева построена таким образом, что обеспечивает минимальное число операций для осуществления поиска значения по ключу и небольшое число операций, для осуществления вставки значения в дерево по ключу или удаления значения. Выбор данной структуры стал причиной оптимальности работы структуры по времени. Ширина дерева ограничена алфавитом, составляющим слова-ключи дерева.

Глубина дерева невелика – в силу того, что ключами в дереве являются слова русского языка. Количество узлов в дереве будет меньше, чем число символов в словах из-за сжатия данных, по причине совпадения большого количества начал в словах.

Предложенные итерационные алгоритмы по ключам дерева не являются рекурсивными. Выполнение одного шага итерации в любую сторону имеет сложность O(M), где M – максимальная длина слова в словаре. Существенным плюсом patricia-tree структуры, имеющей в запасе предложенные итеративные алгоритмы является то, что данная структура для проведения итераций по отсортированным в лексикографическом порядке данным не нуждается в предварительной их сортировке. Это положительно влияет на сложность алгоритмов, поддерживающих работу со структурой.

В результате проведенных автором оптимизаций реализации patricia-дерева, размер, занимаемый структурой уменьшился в 25 раз. При этом характеристики скорости выполнения базовых операций с данными структуры остались практически неизменными. Размер patricia-tree структуры составил 125 MB. Пространственная оптимизация структуры дала результат сравнимый с альтернативными структурами данных, время поиска и вставки элементов в предложенную структуру на порядки больше, чем в альтернативных структурах.

По причине сравнительно большой эффективности выполнения базовых операций работы со структурой слово-значение в оптимизированном в рамках работы автором patricia-дереве, структура является достаточно оптимальной по скорости и приемлемой по занимаемой памяти для хранения данных по словам-ключам словаря русского языка.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Зализняк, А.А.* Грамматический словарь русского языка (словоизменение). 2-е изд. – М.: Русский язык, 1980.
2. *Мельчук, И.А.* Курс общей морфологии. – М.: ЯРК, 1998. – С. 175.
3. *Мельчук, И.А.* Русский язык в модели «Смысл-Текст»/ И.А. Мельчук. – Москва – Вена: Школа «Языки русской культуры», Венский славистический альманах, 1995. – XXVIII. – С. 682.

4. *E. Ukkonen*. Approximate String Matching over Suffix-Trees. In Proceedings of the Fourth Annual Symposium on Combinatorial Pattern Matching, Padova, Italy, June. 1993. – P. 229-242,
5. *D.R. Morrison*. PATRICIA - practical algorithm to retrieve information coded in alphanumeric. Journal of the ACM, 15(4):514-o34 (1968).
6. *Kahveci, T.M*. Proceedings of the 27th International Conferenc on Very Large Databases // T. Kahveci, Ambuj K. Singh // An Efficient Index Structure for String Databases. 2001. – P. 351-360..
7. *Resnikoff, H.L*. The Nature of Affixing in Written English. Part 1, in Mechanical Translation, 8, No. 3 (1965), Part 11 in Mechanical Translation 9, No. 2 (1966).
8. *Shang, H.G*. Tries for Approximate String Matching – H. Shang T.H. Merret – In IEEE Transactions on Knowledge and Data Engineering, volume 8(4). 1996. – P. 540 – 547,

Злыгостев Илья Сергеевич
Технологический институт федерального государственного образовательного учреждения высшего профессионального образования «Южный федеральный университет» в г. Таганроге.
E-mail: workmaking@gmail.com.
347928, г. Таганрог, пер. Некрасовский, 44.
Тел.: 8(8634)371-606.
Кафедра высшей математики; аспирант.

Zlygostev Iliya Sergeevich
Taganrog Institute of Technology – Federal State-Owned Educational Establishment of Higher Vocational Education “Southern Federal University”.
E-mail: workmaking@gmail.com.
44, Nekrasovskiy, Taganrog, 347928, Russia.
Phone: 8(8634)371-606.
The Department of Higher Mathematics; post-graduate student.

УДК 004.932.72

А.Ю. Антипова, В.В. Губарев

ПРИМЕНЕНИЕ МЕТОДА МОМЕНТОВ В ЗАДАЧЕ ГЕОМЕТРИЧЕСКОГО ВЫРАВНИВАНИЯ ЛИЦ НА ИЗОБРАЖЕНИЯХ*

Данная работа посвящена методу улучшения алгоритмов обработки цифровых изображений лиц. Для определения угла наклона лица в плоскости изображения предлагается модель, основанная на понятии эллипса рассеяния и вычислении моментов изображения. Проанализирована применимость этого метода для различных цветовых пространств и методов фильтрации. Приведены результаты вычислительных экспериментов.

Изображения; детекция лиц; метод моментов; геометрическое выравнивание; цветовые пространства; фильтрация.

A.Y. Antipova, V.V. Gubarev

APPLICATION OF METHOD OF MOMENTS IN PROBLEM OF GEOMETRICAL ALIGNMENT OF FACES IN IMAGES

This work focuses on methods of improving algorithms of processing digital images of individuals. To determine the angle of inclination of face in image’s plane a model based on the notion of concentration ellipse and computation of image moments is proposed. Application of this

* Работа выполнена при финансовой поддержке РФФИ, проект №08-07-00129, №07-07-00067.