P. B. Mognonov, D. N. Chimitov, N. V. Yaryshkina

IMPLEMENTATION OF THE INTERPRETER WITH THE DELAYED CALCULATION

One of methods of optimization of operation of algorithms, by means of the interpreter with the delayed calculation, is considered. The authors make an attempt to implement algorithm of operation of such interpreter on the basis of expanded functional language LISP. Value of the delayed calculations for procedure adaptation is shown.

Keywords: the delayed calculations, the interpreter, lambda-calculation, LISP, polymorphism.

УДК 004.75

Н. А. Распопин, М. В. Карасева, П. В. Зеленков, Е. В. Каюков, И. В. Ковалев

МОДЕЛИ И МЕТОДЫ ОПТИМИЗАЦИИ СБОРА И ОБРАБОТКИ ИНФОРМАЦИИ*

Рассмотрена структура представления данных, отвечающих определенным требованиям. Данная дополнительная структура данных разработана исходя из особенностей поисковой системы и требуемой функциональности системы.

Ключевые слова: оптимизация, поисковая система, расширенная база, сбор и обработка информации.

Сегодня редко можно встретить человека, который не использовал бы поисковые технологии. Поиск информации в информационных сетях начал зарождаться задолго до появления глобальной сети. Прообразом первых сетевых поисковиков считается картотечная система наподобие той, что мы встречали в библиотеках, когда большое количество документов классифицируется по особому признаку, каковым может быть жанр произведения, отрасль науки, автор, название произведения; используется и упорядочение по алфавиту. Классификация может быть одноуровневая, когда мы производим ее только по одному признаку, или многоуровневая, когда признаков несколько. Например, мы классифицируем всю литературу по отраслям науки, а потом каждую отрасль науки классифицируем по направлению или по автору и т. д. Система каталогов просуществовала в бумажном виде долгое время, а впоследствии она была перенесена в электронный вариант. В Интернете, в частности, можно было встретить сайты-каталоги, где все многообразие веб-страниц было строго сортировано по тематике, и для того чтобы найти «нужный» сайт, приходилось просматривать десятки, а то и сотни ресурсов. С тех пор в сфере поисковых технологий изменилось очень многое.

Первые поисковые системы сталкивались с большим количеством проблем – прежде всего технического характера [1]. В то время как математические модели позволяли реализовывать достаточно сложные по содержанию алгоритмы, которые могли обеспечить приемлемый для того времени уровень сервиса, аппаратные средства не могли предоставить платформы для реализации подобного рода идей. В значительной степени не хватало мощности процессора для обеспечения производительности, однако основной проблемой была нехватка памяти как оперативной – для реализации самого процесса поиска, так и дисковой – для хранения пространства данных. Разработчики были вынуждены прибегать к многократной оптимизации, что сказывалось на качестве конечного продукта – приходилось чем-то жертвовать. С развитием вычислительной техники подобного рода проблемы ушли на второй план, уступив место новым.

Главная из них – огромный объем информации, которая ко всему прочему имеет очень низкий уровень формализации, что в значительной степени затрудняет поиск на пространстве данных [1]. Под уровнем формализации понимается степень смысловой и любой иной неоднозначности данных. Также необходимо учитывать и лингвистическую неоднозначность. Это прежде всего грамматические, синтаксические, семантические ошибки, которые искажают пространство данных, приводя в конечном счете к ошибочным результатам работы поисковой системы.

Лавинообразное накопление цифровой информации требует совершенно новых подходов к решению проблемы поиска по информационному пространству. Под информационным пространством понимается вся совокупность данных в цифровом виде, доступная посредством различных носителей информации, а также местных локальных и глобальных сетей [2].

*Работа выполнена при финансовой поддержке ФЦП «Научные и научно-педагогические кадры инновационной России на 2009–2013 гг.»

Для эффективной работы алгоритмов обработки и поиска информации прежде всего необходимо создать структуру представления данных, отвечающую определенным требованиям, среди которых основные – это максимальное быстродействие поисковых алгоритмов, простота реализации по структуре данных алгоритмов синтаксического, морфологического и семантического анализов. Разработка структуры представления данных является достаточно сложной и ответственной задачей, от успешной реализации которой зависит дальнейшее функционирование всей системы [3]. Структура представления данных являет собой тот фундамент, на который по кирпичику укладываются различные модули и алгоритмы системы, обеспечивающие выполнение запросов конечных пользователей.

Акцентируем свое внимание на первом требовании, озвученном выше – максимальном быстродействии поисковых алгоритмов. Самым оптимальным вариантом является случай, при котором на поиск подстроки в строке требуется не больше операций, чем содержит подстрока символов. Например, нам требуется найти слово «объем» в одном или нескольких документах. Само слово содержит пять символов, документ при этом может содержать неограниченное количество символов. Рассмотрим структуру данных, образованную из исходного документа путем проведения операции предварительной поисковой оптимизации, которая включает в себя построение линейного дерева (рис. 1) и последующую синхронизацию (рис. 2).

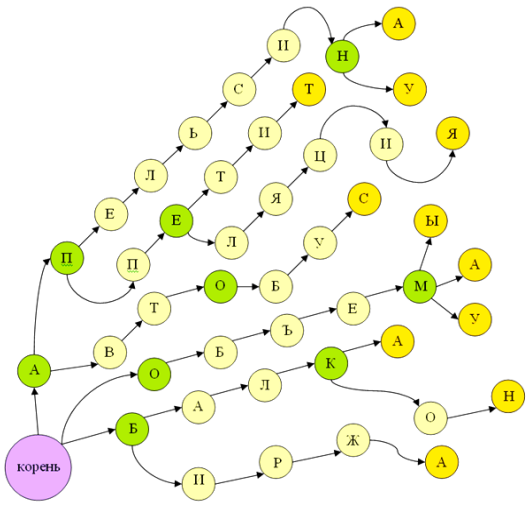

Рис. 1. Линейное дерево

Данная структура представляет собой синхронизированное дерево анализа, состоящее из несовпадающих слов исходного документа, связанных между собой определяющими узлами ветвей и основным корнем дерева. Основной корень дерева является указателем на первые буквы слов, содержащихся в исходном документе. Определяющие узлы ветвей представляют собой узлы в символьной цепочке дерева (их наличие говорит о присутствии однокоренных слов или различных словоформ одного слова), у которых имеется хотя бы один потомок и которые остаются в дереве, если все дочерние узлы по одной из ветвей одновременно удалить. Если же при удалении дочерних узлов определяющий узел тоже удаляется, теряется смысл или не существует словоформы – будем называть такой узел транзитным. Также показаны конечные узлы, не имеющие потомков.

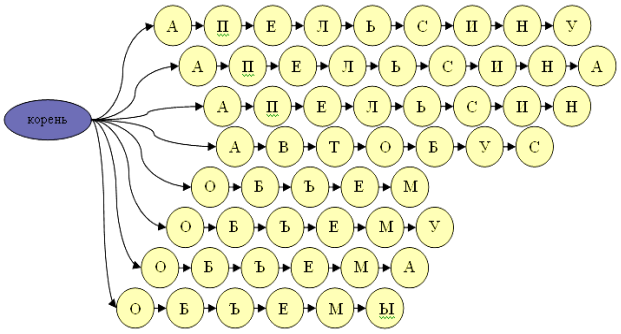

Рис. 2. Синхронизированное линейное дерево

Рассмотрим более подробно правило построения структуры, представленной на рис. 2. Исходный документ разбивается на отдельные составляющие – слова (элементы), затем каждое слово побуквенно накладывается на уже имеющуюся структуру так, как показано на рисунке. Например, если в тексте 10 раз встречается слово «объем», 15 раз слово «объему», 20 раз «объемы» и 25 раз «объема», которые все вместе содержат в себе 410 символов – в структуре они займут всего лишь восемь символов.

Процедура поиска слова «объема» по структуре данных (рис. 3) заключается в последовательном переходе от узла-корня структуры к узлу-потомку с посимвольным сравнением содержания соответствующего узла с буквой в слове, отстоящей по порядку от начала слова на величину уровня узла потомка. Введем функцию $z(t)$, которая в каждый момент времени t будет обозначать букву в искомом слове, находящуюся на $t+1$ месте от начала. Введем функцию $p(z(t+1), r(t))$, которая будет возвращать указатель на потомка узла структуры, если у текущего узла есть потомок, который содержит $z(t)$, и будет возвращать 0, если у узла нет потомков или ни один потомок не содержит $z(t)$, где $r(t)$ – указатель на текущий узел в момент времени t.

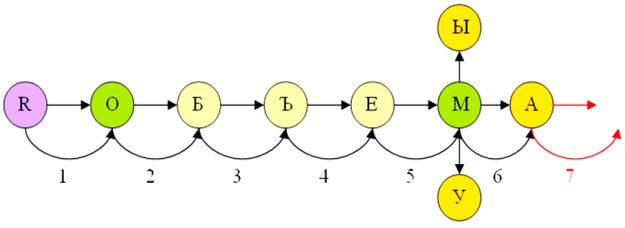

Рис. 3. Процедура поиска по структуре данных

В момент $t = 0,\ z(1) = «o»,\ p(z(1), root) = «o»$. Функция вернула указатель на узел, который является дочерним для корня и содержит искомую букву. В момент $t = 1,\ z(2) = «б»,\ p(z(2), «o») = «б»$. В момент времени $t = 5,\ z(6) = «a»,\ p(z(6), «м») = «a»$. В момент времени $t = 6,\ z(7) = \text{NULL}$ – это означает, что проверка закончилась, и все буквы слова присутствуют в структуре данных в том порядке, в котором они идут в слове. Для того чтобы подтвердить нахождение слова в документе, достаточно, чтобы последний узел был конечным или определяющим. Если операция поиска закончится на транзитном узле – это недостаточное условие, искомого слова в документе нет. В нашем случае это конечный узел, т. е. искомое слово найдено.

Итак, для представленной структуры данных (см. рис. 2), мы определили поисковый алгоритм, который позволяет найти в ней искомое слово. В случае поиска подстроки или более крупного фрагмента текста будет необходимо проверять его наличие поэлементно. Рассмотрение этого момента не является целью данной статьи. В данном случае можно показать, что максимальное количество операций $P_{\max}(x)$ для поиска слова занимает не больше операций, чем длина $N(x)$ самого слова плюс единица:

$$P_{\max}(x) = N(x) + 1;$$
$$P(x) <= N(x) + 1;$$
$$P(x) > 0.$$

Выражение $0 < P(x) <= N(x)$ говорит о том, что количество операций, затрачиваемое на поиск слова, может быть меньше, чем $P_{\max}(x)$ (однако всегда больше 0). Это возможно в том случае, если искомого слова в документе нет. Выражение $P(x) > 0$ следует из того, что для получения отрицательного результата от поискового алгоритма необходимо произвести как минимум одну операцию – определение нулевого указателя на узел с первой буквой слова. Математически поисковый алгоритм будет выглядеть следующим образом:

$$\sum_{i=0}^{(N(x)-1)} p(z(i+1), r(i)) = \sum_{i=0}^{(N(x)-1)} z(i+1),$$

Если равенство соблюдается – элемент присутствует в структуре, если нет – элемент в структуре не содержится.

Мы рассмотрели простейший случай поискового запроса – фактически это бинарная операция по определению присутствия или отсутствия слова (элемента) в тексте. На практике потребности в поисковых алгоритмах довольно редко ограничиваются столь примитивной формой. Почти всегда бывает необходимо определить наличие в тексте группы слов, выражений или даже целых текстовых фрагментов, также может потребоваться наложение некоторых ограничений на искомые элементы: дату их создания или последнего редактирования, расположение в документе, регистр символов и т. д. В конечном счете, может быть поисковый запрос, содержащий в себе десятки и сотни элементов и ограничений. Однако каким бы сложным он ни был, его решение всегда сводится к разбиению на простейшие составляющие элементы, каждый из которых обрабатывается независимо друг от друга. Данные, полученные от каждого элементарного поискового запроса, формируются в единое целое, которое представляет собой ответ системы на первоначальный поисковый запрос. С учетом вышесказанного наш алгоритм будет выглядеть следующим образом:

$$\sum_{k=1}^{N(y)} \sum_{i=0}^{(N(x_k)-1)} p(z(i+1), r(i)) = \\ = \sum_{k=1}^{N(y)} \sum_{i=0}^{(N(x_k)-1)} z(i+1),$$

где $N(y)$ – количество элементарных запросов, на которые разбит первоначальный x. Следует отметить, что данное выражение позволяет определить только бинарное состояние каждого из $k = 1...y$ запросов. Обработка запросов более высокого уровня, например проведение семантического анализа, будет рассмотрена далее.

Несмотря на кажущуюся простоту и легкость работы со структурой, изображенной на рис. 2, в ней есть существенный недостаток. В подавляющем большинстве случаев пользователя интересует не столько положительная или отрицательная реакция поисковой системы, сколько получение конкретного документа или указания места в документе, которому соответствует его поисковый запрос. Предложенная структура в общем виде не удовлетворяет условиям, при которых можно было бы реализовать дополнительный функционал. Более того, преобразование документа в структуру необратимо, так как произвести обратное преобразование и получить документ из структуры невозможно. Это означает, что не существует однозначного отображения оптимизированной структуры в исходный документ. Соответственно, получив по ходу работы поискового алгоритма элемент в оптимизированной структуре, будет невозможно анализировать его в связи с первоначальным документом. Для устранения указанных ограничений нам понадобится ввести некоторые изменения в существующий алгоритм оптимизации, а именно: дополнительную структуру данных, которую мы назовем *расширенной базой*. Структура расширенной базы должна разрабатываться исходя из особенностей поисковой системы: чем больше возможностей должна иметь поисковая система, тем сложнее и больше будет расширенная база.

Положим, наша система должна иметь возможность перевести курсор на позицию, в которой находится искомый элемент, если таковой существует; также должна быть реализована возможность поиска подстроки, состоящей из произвольного количества элементов. Под элементом понимается отдельное слово или любое другое цифробуквенное выражение, отстоящее от других на один знак-разделитель

или более (по умолчанию разделителем будем считать пробел). Для того чтобы иметь возможность перевести курсор на позицию элемента, необходимо, чтобы в структуре была представлена информация о местоположении каждого отдельного слова. Для всех узлов структуры, являющихся *конечными* или *определяющими*, создается отдельная таблица, в которой хранится информация о каждом элементе.

В предложенном случае необходимо хранить следующую информацию об элементе:

– принадлежность элемента к документу $\varepsilon \in P$, где ε – индекс элемента (он же индекс таблицы), P – индекс документа. Следует отметить, что $\varepsilon = \mu$, где μ – индекс узла. Индекс узла назначается всем *конечным* и *определяющим* узлам структуры, транзитным узлам индекс не назначается;

– индекс предыдущего и индекс следующего элемента в данном документе $\varepsilon_{i-1} < \varepsilon_i < \varepsilon_{i+1}$. Эти данные нам понадобятся для выполнения сложных запросов и осуществление лингвистического анализа текста;

– положение элемента ε в документе P в виде смещения относительно начала документа – обозначим его как θ. По этому параметру, исходя из $\varepsilon \in P$, можно будет однозначно восстановить исходный документ из оптимизированной структуры. Таким образом, ограничение необратимости преобразования, о котором мы говорили выше – снимается.

Содержание расширенной базы напрямую зависит от требуемой функциональности системы. В случае дополнительных требований информативность расширенной базы может быть увеличена путем добавления необходимых полей в таблицу элемента (это может быть время создания документа, лингвистические характеристики: род, число, падеж и т. д.). Чем информативнее расширенная база, тем больше будет возможностей для проведения анализа, тем качественнее может быть работа всей поисковой системы в целом.

Библиографические ссылки

1. Талантов М. Поиск в Интернете: подводные камни // КомпьютерПресс. 1999. № 9. С. 46–52.

2. Мультилингвистическая модель распределенной системы на основе тезауруса / С. В. Рогов, П. В. Зеленков, И. В. Ковалев, М. В. Карасева // Вестник СибГАУ. 2008. Вып. 1 (18). С. 26–28.

3. Карцан И. Н., Лохмаков П. М., Цветков Ю. Д. Интеллектуализация поиска информации в корпоративных системах // Вестник НИИ СУВПТ. 2006. Вып. 23. С. 141–156.

N. A. Raspopin, M. V. Karasyova, P. V. Zelenkov, E. V. Kayukov, I. V. Kovalyov

MODELS AND METHODS OF DATA COLLECTING AND PROCESSING

The paper considers the structure of data presentation, which meet certain requirements. The given additional data structure is developed, proceeding from requirements of the retrieval system and necessary system functionality.

Keywords: optimization, retrieval system, extended base, data collecting and processing.

УДК 81`322

К. В. Сафонов, Д. В. Личаргин

НЕКОТОРЫЕ ПРИНЦИПЫ АВТОМАТИЧЕСКОЙ ГЕНЕРАЦИИ УЧЕБНЫХ МАТЕРИАЛОВ НА ОСНОВЕ БАЗ ЗНАНИЙ И ЛИНГВИСТИЧЕСКОЙ КЛАССИФИКАЦИИ

Рассматриваются модели и средства генерации осмысленного подмножества естественного языка учебных курсов. В частности, задается семантическое понятийное пространство слов языка. Ставится цель построить модель генерации текстов для учебных курсов по английскому языку, формулируются задачи ее применения на основе порождающих грамматик над ориентированным лесом строк. Делается вывод о специфике и структуре модели генерации учебных курсов.

Ключевые слова: генерация естественного языка, семантические признаки, классификации слов и понятий языка, генерация учебных материалов.

На современном этапе актуальной является проблема автоматизации систем письменного и устного перевода для различных языков, экспертных, поисковых систем и систем реферирования. Для решения данных задач успешно реализуются различные теории, концепции и программные системы. Многочисленные работы в области семантики, дискретной математики, лингвистики и искусственного интеллекта дают надежду на решение в ближайшем будущем многих проблем формализации естественного языка и прохождения теста Тьюринга во все более жестких для тестовых систем условиях. Особенно важной