Усовершенствованный метод поиска слов для систем обработки данных

Маталыга А.А.
ИПиЭ, ФКП
Белорусский государственный университет информатики и радиоэлектроники
Минск, Беларусь
e-mail: kadosha1@rambler.ru

***Аннотация* — В данной статье представлен оригинальный метод поиска информации, использующий методы ассоциативного и бинарного поиска.**

Ключевые слова: поиск; ассоциативная память; бинарное дерево; ошибки; опечатки

Сегодня в системе интернет существует большое количество сайтов, связанных с различными аспектами деятельности человека. Имеется также большое количество поисковых машин (Google, Yandex, Nigma). Google.com является самой корректной поисковой машиной. Yandex.ru - быстрее всех индексирует. Разработчики поисковых сервисов стремятся разработать наиболее совершенный алгоритм отбора результатов поиска в интернете. У большинства поисковых сервисов есть поисковые системы. Каждая система обладает собственной методикой отбора правильных результатов. Несмотря на имеющиеся результаты, до сих пор не был предложен наиболее оптимальный механизма поиска, который бы справился с любыми запросами пользователя, при этом не важно, как они сформулированы и какие в них содержатся ошибки.

Набирая поисковый запрос, пользователи нередко ошибаются, например: пропускают или добавляют лишние буквы, пишут слова с орфографическими ошибками, пишут слова разговорным языком (включая слэнг), пишут слова не переключив клавиатуру на нужный язык. Процент ошибок пользователей настолько значителен, что многие компании создают на своих сайтах страницы, “заточенные” под разного рода ошибки и опечатки.

Две трети всех ошибок - орфографические, четверть - связана с лишними или пропущенными пробелами, а так же со смысловыми ошибками. Примерно 5% ошибок возникает из-за неверной раскладки клавиатуры [1].

Приведем несколько примеров типичных опечаток и грамматических ошибок, встречающихся в тексте. Чаще всего пользователи поисковых систем допускают следующие ошибки и опечатки:

– Добавление лишних элементов. Повторение одной и той же буквы; нажатие соседней клавиши, вместо необходимой (чаще всего по горизонтали).

– Пропуски элементов. Особенно при соединении букв, включающих одинаковый элемент.

– Недописывание элементов букв. Связано с недоучетом их количества: Л вместо М; Х вместо Ж и т.д.

– Потеря или первой, или последней буквы в слове.

– Зеркальное написание букв. Буквы переставлены местами при наборе.

При формулировании запроса самой распространённой ошибкой является написание сайта "Однокласники" с одной буквой «с». Второе место среди ошибочных написаний уверенно держит слово "агенство", без буквы «т» [1]. Далее следуют ошибки в словах:

- русификатор вместо руссификатор ;
- шпаргалки вместо шпоргалки ;
- видео вместо видио и т.д.

Таким образом, выделив типичные ошибки в поисковых запросах, следует их учитывать при разработке алгоритма поиска.

Как мы знаем, в ассоциативной памяти поиск реализуется аппаратно путем параллельного сравнения слова-эталона со всеми записанными словами [2]. Для этого каждый набор элементов хранения программных объектов дополняется схемами сравнения. При схемной реализации ассоциативной памяти доступ к данным осуществляется очень быстро, их поиск по любому фрагменту не представляет труда, если этот фрагмент точно определен. Но, если фрагмент был видоизменен (написан с ошибкой или опечаткой), то ассоциативный поиск в таком случае будет безрезультатен [3]. В этой связи будет целесообразно использовать возможности бинарного поиска, который может осуществлять поиск не точно заданного ключа.

На основании выше изложенного можно предположить, что оптимальным будет поиск, который соединит в себе ассоциативный поиск и поиск по дереву. Дерево дает возможность использования ASCII кода. Буквы кодируются в соответствии с кодовым набором ASCII, для их представления и поиска используются 5 младших бит кода [4].

В предлагаемом нами методе используется аналог метода динамики средних (идея нивелирования влияния случайных отклонений при ошибках в записи ключей), полученный список поиска дает вероятностные результаты (определяемый документ не обязательно тот, поиск которого задумал клиент сайта).

Рассмотрим более детально алгоритм поиска. Сначала подсчитывается среднее значение ASCII-кода запроса. Далее сравнивается среднее значение запроса с элементом массива, т.е. сравнивается значение массива с диапазоном среднего значения ключа. Первая запись входной последовательности сопоставляется с диапазоном значений корня дерева.

Для каждой следующей записи ключ сначала сравнивается с диапазоном значений ключа корня, т.е. входит ли ключ записи в диапазон значений ключа корня дерева. Если он меньше чем диапазон значений ключа корня, то далее он сравнивается с диапазоном значений ключа правого потомка и т.д. до тех пор, пока потомок не будет отсутствовать. Место отсутствующего потомка занимает новая вершина, с которой сопоставляется очередная запись.

Данные действия повторяются до тех пор, пока не будет просмотрена вся входная последовательность записей.

Поиск считается успешно завершенным, если ключ искомого элемента входит в диапазон значений узла. Если поиск завершается неудачей, т.е. ключ не вошел в диапазоны, приписанные узлам дерева, то выбираем

тот узел, где степень близости значений оказалась наибольшей.

Более подробно алгоритм предлагаемого нами метода показан на Рис. 1.

Рассмотрим алгоритм поиска на примере (Рис.2). В строку запроса введено слово «грепп». В начале осуществляется поиск введенного слова (ключевого слова) путем параллельного сравнения со всеми хранимыми в памяти словами. Поиск по ключу оказался безрезультатным, далее поиск автоматически продолжается по дереву. Слово «грепп» имеет ключ 1174, среднее значение ASCII-кода запроса - 234,8. Сравнивается среднее значение запроса с элементом массива, т.е. идет сравнение значения массива с диапазоном среднего значения ключа. Соответственно производится поиск диапазона значений ключа по дереву. Первая запись входной последовательности сопоставляется с диапазоном значений корня дерева.

Для каждой следующей записи ключ сначала сравнивается с диапазоном значений ключа корня, т.е. входит ли ключ записи в диапазон значений ключа корня дерева. Если он меньше чем диапазон значений ключа корня, то далее ключ сравнивается с диапазоном значений ключа правого потомка и т.д. до тех пор, пока потомок не будет отсутствовать. Место отсутствующего потомка занимает новая вершина, с которой сопоставляется очередная запись.

В рассматриваемом примере ответом будет узел p4 с диапазонам значений (230; 240) в который попадает ключ искомого слова.

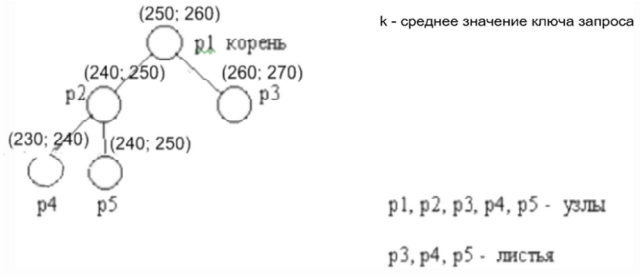

Рис. 2. Пример алгоритма поиска

Таким образом, соединение ассоциативного поиска с поиском по дереву позволит улучшить алгоритм ассоциативного поиска, что, в свою очередь, приведет к уменьшению затрат времени на получение необходимого пользователю объема информации.

[1] Поиск в интернете: самые популярные запросы, что чаще всего ищут в интернете, интересные запросы и опечатки [Электронный ресурс]. – Электронные данные.– Режим доступа: http://www.phorumka.ru/forum/75 - 8452-1/

[2] Кохонен, Т. Ассоциативная память / Т. Кохонен – М.: Мир, 1980. – 240 с.

[3] Власова, А.Е. Алгоритм формирования ассоциативных связей и его применение в поисковых системах / А.Е. Власова, В.И. Шабанов // Тезисы докладов международной конференции «Диалог 2003» – М., Московский государственный лингвистический университет, 2003. – 6 с.

[4] Прохождение и поиск по бинарным деревьям [Электронный ресурс]. – Электронные данные.– Режим доступа: http://rk6.bmstu.ru/electronic_book/posapr/zadanpo/bintree.htm, свободный.

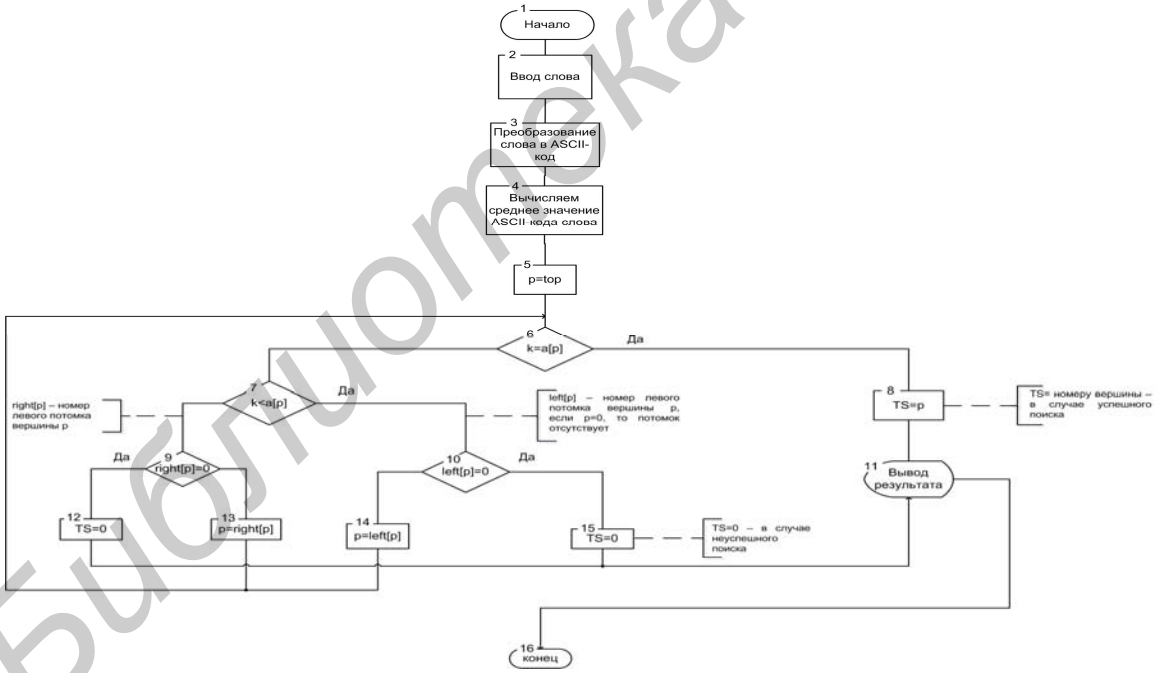

Рис. 1. Блок-схема алгоритма