электронный журнал
МОЛОДЕЖНЫЙ
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана»

УДК 004.021

Алгоритм нечеткого фонетического поиска на основе простых чисел

В. В. Ставровиецкий, Ю.Е. Гапанюк, В.А. Галкин

slava.lion25@gmail.com, sfm2007@yandex.ru, galkinvalery@rambler.ru

Введение

Нечёткий поиск – это одна из важнейших функций поисковой системы. Он используется в таких задачах, как проверка орфографии, исправление опечаток, поиск похожих значений или поиск вхождения подстроки в тексте. В то же время нечеткий фонетический поиск предоставляет ряд дополнительных возможностей, таких как поиск слова по звучанию, когда есть неоднозначность написания, или поиск слов, когда вообще не известно написание слова (иностранные и заимствованные слова), транслитерации. [1] Основное направление данной работы - это поиск человека по фамилии, имени и отчеству в базе данных. Практическая ценность решения этой проблемы достаточно высока. Во многих случаях такая, казалось бы, несложная задача не может быть решена с достаточной скоростью и точностью. Одна из самых распространённых проблем – это опечатки либо в тексте поискового запроса, либо в базе данных. Вторая проблема – это запись ФИО транслитерацией. Третья, которая и будет рассмотрена в этой статье, - когда точной написание не известно (ведь при знакомстве не предоставляют паспорт, а представляются устно). Как пример, фамилия Соколов может произноситься или слышаться как Сокалов или Сакалов. В таких случаях приходит на помощь нечеткий фонетический поиск.

Организация фонетического поиска

При фонетическом поиске каждому слову сопоставляется определённый код, который одинаков для схожих по звучанию лингвистических единиц. На сегодняшний день существует достаточно алгоритмов, так или иначе решающих эту задачу. Самые известные и популярные - это Soundex, Daitch-Mokotoff Soundex, NYSIIS, Metaphone, Double Metaphone, русский Metaphone[2], Caverphone. Одна из особенностей этой работы - это поиск русскоязычных фамилий, имен и отчеств. Фонетические алгоритмы основаны на языковых особенностях, таких как: «оглушение» букв в зависимости от их местоположения; чередование корней; парные согласные («д» - «т», «ж» - «ш» и т.д.), которые имеют одинаковое звучание, и ряд других правил. По этой причине алгоритмы имеют свою специфику и в основном предназначены для конкретных алфавитов и языков. Исходя из этого, предпочтительными алгоритмами являются Daitch-Mokotoff Soundex, разработанный для восточно-европейских языков, и русская адаптация Metaphone. Однако Daitch-Mokotoff Soundex использует для представления кода слова число, что является неинформативным и не даёт хорошей возможности осуществить нечёткий поиск после фонетического кодирования.[3] Русский Metaphone сопоставляет слову его фонетический код в виде набора символов. После преобразования вполне возможен нечёткий поиск. Именно этот вариант и будет и будет взят за основу. Для сохранения информативности будем кодировать и окончания, а не заменять их на цифры, как это предлагает русский Metaphone.

Организация нечеткого поиска

Именно алгоритмы нечёткого поиска и помогают распознать возможную ошибку, находят похожие слова или места вхождения подстроки в текст. Существует достаточное количество алгоритмов и методов для решения этой задачи. Чаще всего схожесть слов определяют через расстояние Дамерау-Левенштейна, также используют алгоритм Bitap, алгоритм расширения выборки, метод N-грамм, хеширование по сигнатуре, BK-деревья.

В статье предлагается новый подход для решения этой задачи.

Каждое слово характеризуется определённым набором букв. Часто ошибка ввода слова связана с перестановкой букв местами или повторным вводом одной и той же буквы. Определение схожести двух слов предлагается осуществлять на основе схожести их числового представления. Предлагается подход, в основу которого положено предположение, выдвинутое автором, что для нечеткого поиска не так важен порядок букв в слове и можно характеризовать искомое выражение набором уникальных букв. Для рассматриваемого примера (фамилия Соколов) – это С, О, К, Л, В.

Очевидно, что поиск числового значения, соответствующего определенному набору букв, будет значительно быстрее, чем поиск строки и вхождения определенных букв в эту строку. Именно поэтому в работе рассматривается подход на основе числовой идентификации набора символов.

Каждой букве алфавита предполагается ставить в соответствие свой уникальный числовой идентификатор. Возникает вопрос выбора числового ряда, который бы лучше всего удовлетворял этой задаче. Не представляется возможным использовать идущие один за другим значении вида А – 1, Б – 2, В – 3 и т.д. Необходимо минимизировать число слов, описываемых одним числом. Числа, которые расположены достаточно близко друг к другу не дают такой возможности, так как комбинации букв АБ будет иметь то же значении, что и В. Это не допустимо.

После продолжительных исследовании автором статьи был найден числовой ряд, максимально удовлетворяющий заданной задаче. Этот ряд представлен ниже.

Таблица 1

Числовой ряд

А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П
3	5	7	11	13	17	19	23	29	31	37	41	43	47	53	59	61

Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ы	Ъ	Э	Ю	Я
67	71	73	79	83	89	97	101	103	107	109	113	127	131	137	139

Это ряд простых чисел. Ни одна буква при заданных значениях чисел не может быть представлена суммой значений других букв. Описанная выше проблема возможной неточности при таком представлении исключена.

Одновременно на этапе кодирования можно учесть наработки в сфере фонетических алгоритмов и более точно распределить буквы по числам. Начать можно с очевидных вещей, например, не кодировать буквы Ъ и Ь, а буквы Ё и Й заменять на Е и И соответственно. Также предлагается использовать созвучие некоторых букв, таких как Б – П, З – С, Д – Т, В – Ф, Г – К, ТС – Ц, О – А, Е – И, Э – И, Ю – У, Ш – Щ. (этот же подход используется для получения фонетического ключа в русском Metaphone, оттуда и был заимствован с небольшими изменениями) Таким образом таблица соответствия символов и их кодов принимает следующий вид.

Таблица 2

Числовой ряд после анализа

Е, Ё	И, Й, Ы	Э		Ю	У		О	А		Я

3	5	7		11	13		17	19		23

Б	**П**	Ж	**З**	**С**	Л	М	**Д**	**Т**
29	**31**	37	**41**	**43**	47	53	**59**	**61**

Н	**В**	**Ф**	Р				**Г**	**К**	Ч	Ц		Х		**Ш**	**Щ**
67	**71**	**73**	79	83	89	97	**101**	**103**	107	109	113	127	131	**137**	**139**

В приведенной выше таблице номера выбраны не случайно. Все описанные в правилах буквы отличаются друг от друга ровно на значении 2. Все остальные отличаются на большее значение. Таким образом, возвращаясь к примеру, фамилия Соколов будет закодирована согласно этой таблице, как С – 43, О – 17 (т.к. возможно А, запоминаем отклонение +2), К – 103 (тут отклонение можно не запоминать, т.к. Г преобразуется к К, а не наоборот), О – уже было (согласное выдвинутому предположению запоминаем только уникальные символы), Л – 47, О – опять пропускаем, В – 71 (запоминаем отклонение +2 в сторону Ф). Путем сложения всех чисел, получаем: 43 + 17 + 103 + 47 + 71 = 281. Возможные отклонения +2, если оно только одно, и +4, если ошибка в обоих случаях: О и В. Поиск по отклонениям будет осуществляться, если пользователь не удовлетворен результатами работы алгоритма.

Возможные изменения метода расчета:

- Сразу преобразовывать буквы по указанным правилам и запоминать только отрицательное отклонение. В этом случае будет большая группировка похожих слов по одному числовому ключу.
- Применять нечеткий поиск после работы фонетического алгоритма и искать по выданному этим алгоритмом ключу. (рассмотрено ниже)

Организация нечеткого поиска после работы фонетического алгоритма

В данной части статьи описывается применения нечеткого поиска после формирования фонетического ключа, который устранит ряд возможных неточностей, связанных с незнанием правильного написания слова. Соответственно для приведенного выше примера (фамилия Соколов) будет осуществляться нечёткий поиск по ключу САКАЛАФ, который будет характеризовать ряд схожих фамилий, таких как Соколов, Сокалов, Сакалов и др.

Алгоритм нечеткого поиска, описанный выше, работает по тем же принципам. Однако поскольку на вход он получает уже готовый фонетический ключ, никаких дополнительных правил преобразования использовано не будет. Для рассматриваемого примера результат будет следующий: С – 43, А – 19, К – 103, Л – 47, Ф – 73, итог - 285. Дальнейший поиск будет осуществляться, разумеется, без учета каких-либо отклонений.

Сложность предложенного алгоритма

Так как суть алгоритма состоит в обработке определённого количества символов и суммировании их кодов, то можно видеть, что не зависимо от того, какой символ обрабатывается, последовательность и количество операций всегда одинаковы. Таким образом, время работы алгоритма линейно зависит от длины слова, поданного на вход. Сложность предложенного алгоритма описывается функцией O(N), где N – количество символов на входе, а O – сложность алгоритма.

Было бы любопытно сравнить эту сложность с существующими алгоритмами. Расчёт расстояния Левенштейна в базовой версии этого алгоритма подразумевает вычисление матрицы и имеет сложность O(mn), где m – длина искомой строки, поданной на вход, а n – длина строки, в которой осуществляется поиск. Кроме того маловероятно найти универсальное слово, после вычисления расстояния до которого, можно было бы использовать полученный результат, как ключ для поиска в БД. Bitap (также известный как Shift-Or или Baeza-Yates-Gonnet, и его модификация от Wu-Manber), также производящий расчет расстояния Левенштейна, но другим методом, имеет сложность O(kn), где k – количество ошибок, а n – длина искомой строки. Алгоритм расширения выборки, являющийся алгоритмом нечеткого поиска с индексацией, в своей основной версии имеет сложность $O((m|A|)^k \cdot m \cdot \log n)$. Метод хеширования по сигнатуре - $O(|H|^k \cdot n / (2^{|H|}))$

Требования алгоритма к памяти

Алгоритм очень нетребователен к памяти. На вход поступает строка и память нужна только для сложения чисел, соответствующих символам в строке. Соответственно зависимость количества памяти, которая необходима для успешной работы, тоже линейная - O(N), где N – количество символов на входе.

Точность работы алгоритма

В силу особенностей алгоритма, в результате его работы в конечное множество результатов попадают нежелательные элементы. Это происходит из-за численного представления слова. Для примера: Соловов → солв → 43+17+47+71 = 178 и Иванов → ивано → 5+71+19+67+17 = 179. Алгоритм запоминает возможные отклонения, таким образом, при поиске одной из этих фамилий, будут найдены обе. Во избежание этого предлагается использовать на конечном этапе ограничение, накладываемое на расстояние Дамерау – Левенштейна между искомой комбинацией слов и результатами работы алгоритма. Если взять ограничение "расстояние ≤ 3", то останутся только релевантные результаты поиска.

Итоги

Был разработан принципиально новый метод нечеткого поиска, который может успешно включать в себя некоторые черты фонетического поиска. Таким образом, один алгоритм выполняет сразу несколько задач, раннее выполняемых отдельно и поэтапно. Сложность предложенного алгоритма и его требования к памяти минимальны. Эта разработка может быть успешно применена для нечеткого фонетического поиска в базах данных, содержащих персональные данные, а также результаты могут быть обобщены для любой структурированной информации (картотеки, списки и т.д.).

Список литературы

[1] Билл Смит, Методы и алгоритмы вычислений на строках, Вильямс, 2006 г.

[2] «Адаптация английской версии алгоритма Metaphone к русскому алфавиту» Петр Каньковски, журнал «Программист», 2002 г., 8-ой выпуск,
(http://forum.aeroion.ru/topic461.html - электронная версия)

[3] Ресурс, посвященный нечеткому поиску, Бойцов Леонид,
http://ftp.forsys.ru/literature/itman/index.htm