МЕТОДЫ ПОИСКА КЛЮЧЕВЫХ СЛОВ В ПОЛНОТЕКСТОВЫХ БАЗАХ ДАННЫХ

К.Я. Кудрявцев, П.Д. Волков

Во многих случаях, поиск требуемой информации в полнотекстовых базах данных осуществляется по ключевым словам, которые определяются автором и отражают основные положения документа. Задание ключевых слов определяется на этапе внесения информации о печатном издании в базу данных и является довольно ответственным моментом, т.к. отсутствие какого-либо ключевого слова приведет к тому, что информация о печатной работе не будет выведена поисковыми системами. Аналогичные подходы используются поисковыми системами интернет, когда информация о сайте включается в ответ на запрос при наличии определенных ключевых слов, связанных с сайтом и заданных разработчиком сайта. Как видно проблема заключается в необходимости предварительного задания набора ключевых слов и отсутствие некоторых из них исключает печатную работу из списка поиска.

Другим способом быстрого поиска ключевых слов в текстовых документах является создание специальных, довольно громоздких структур данных, таких как B+ – деревья [3], суффиксные деревья [1], GiST – обобщенное дерево поиска [2] и др. Построение и хранение перечисленных структур данных требует довольно больших аппаратно-временных ресурсов.

Было бы интересно создать такой механизм поиска информации в полнотекстовых базах данных, для которого не требовалось бы предварительного задания списка ключевых слов и не требовалось больших объемов памяти для хранения вспомогательных данных.

В данной работе предлагается подход создания поисковых систем без предварительного задания набора ключевых слов для каждой печатной работы. Подход строится на основе подхода аналогичного построению дискретного вейвлет преобразования и сглаживания по методу скользящего сведения.

1. Описание основных соотношений.

Представим текст из файла в виде ряда *{x(n)}* длиной *L (n=0,1,…,L-1)*. Текст целесообразно привести к нижнему регистру и удалить из него пробелы, знаки препинания и прочие незначащие символы. Значениями *x(n)* можно считать, например ASCII-коды символов (хотя можно использовать и собственную кодировку, например, числа из диапазона [-1,+1]). На представленном рисунке 1 на верхнем графике изображен именно такой текстовый ряд *{x(n)}*.

Будем считать, что ключевое слово, имеющее длину *N*, присутствует в тексте и расположено, начиная с позиции n_0.

Вейвлет преобразование представляет собой интегральное преобразование исходного сигнала с помощью вейвлет функций. Оно несколько похоже на преобразование Фурье, но при этом позволяет локализовать частотные изменения сигнала во времени.

Накладывая исходный сигнал на масштабируемый вейвлет и проводя интегрирование по всей временной области получают новое двухмерное представление о исходном сигнале. Новое представление в ряде случаев позволяет выявить определенные закономерности в сигнале, произвести его уплотнение и фильтрацию. Дискретное вейвлет преобразование задает новое представление сигнала, состоящее из усреднений и детализаций.

В данной работе предлагается из исходного сигнала получить «карту усреднений». Для ее получения будем рассматривать прямоугольный «вейвлет» пременной длины N, который будем перемещать вдоль сигнала. Как видно из рисунка при достижении расстояния n_0 произойдет наложение «вейвлета» на ключевое слово и, следовательно, оно будет обнаружено. Таким образом, необходимо выбрать диапазон ширины прямоугольника «вейвлета» и сопоб вычисления «вейвлет преобразования». В качестве способа вычисления «вейвлет преобразования» целесообразно вычислять среднее значение для дипазона попавщего внутрь «вейвлета». Данный процесс аналогичен вычислению по методу скользящего среднего. Следует начать с трехбуквенных

усреднений, далее получить четырех-буквенные, затем пяти-буквенные и т.д. до восьми-буквенных усреднений. Находить более длинные усреднения по-видимомму нецелесообразно, т.к. длина большинства слов сосредоточена в этом диапазоне. Таким образом, будет получена некоторая карта (матрица) состоящая из 6 строк и L-2 столбцов, причем стобцы L-3, L-4,…, L-7 в нижних строках равны нулю.

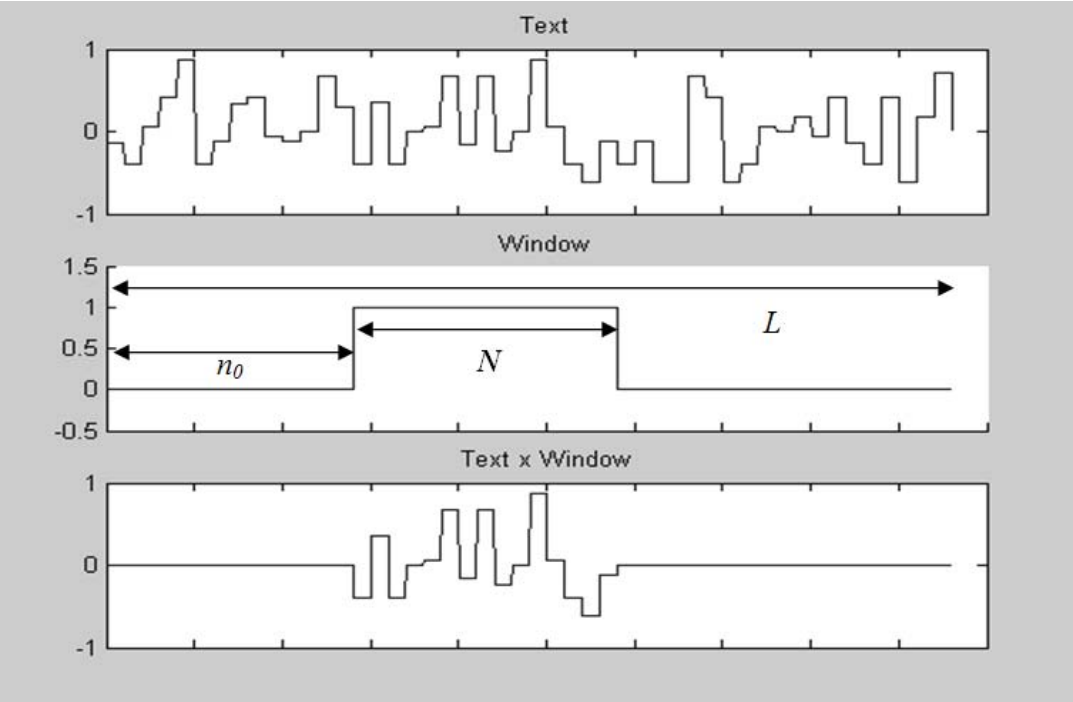

Рис. 1. Представление текста в виде сигнала {x(n)}

Рассмотрим на примере создание «карты усреднений».

Пусть имеется строка из 10 символов – **abcdefghjk**. Тогда карта (матрица) усреднений будет выглядеть следующим образом:

abc	bcd	cde	def	efg	fgh	ghj	hjk
abcd	bcde	cdef	defg	efgh	fghj	ghjk	-
abcde	bcdef	cdefg	defgh	efghj	fghjk	-	-
abcdef	bcdefg	cdefgh	defghj	efghjk	-	-	-
abcdefg	bcdefgh	cdefghj	defghjk	-	-	-	-
abcdefgh	bcdefghj	cdefghjk	-	-	-	-	-

В реальной ситуации вместо букв abc будет стоять некоторое число равное:

abc = [Код(a) + Код(b) + Код(c)] / 3

или

cdefg = [Код(c) + Код(d) + Код(e) + Код(f) + Код(g)] / 5

Данная карта усреднений строится один раз и в дальнейшем используется для поиска ключевых слов.

Например, требуется определить, содержится ли слово «bcdefg» в тексте.

Для этого определяем «среднее» значение данного слова:

bcdefg = [Код(b) + Код(c) + Код(d) + Код(e) + Код(f) + Код(g)] / 6

выбираем 6-ю строку (т.к. длина слова равна 6 символам) и производим сравнение с элементами данной строки. Если совпадение обнаружено, то слово присутствует, если нет, то слово отсутствует.

Как видно, «карта усреднений» не приводит к существенному увеличению ресурсов памяти, по-сравнению, например, с индексами всех слов.

В ряде случаев могут быть ошибки. Например, слова «МИР» и «РИМ» будут иметь одинаковые средние значения, но при реальном поиске обычно задаются довольно длинные слова (словосочетания) и вероятность совпадений становится небольшой. При этом следует отметить, что лучше выдать избыточную информацию, чем пропустить какой либо документ.

2. Алгоритм поиска ключевых слов.

На основании вышеизложенного, можно предложить следующий алгоритм поиска ключевых слов в файлах или в полнотекстовых базах данных:

1. Создание карты усреднений M[8 x L-2], где *L* длина текста, исходного, модифицированного текста. Данную операцию следует проделать один раз на начальном этапе построения базы данных или при создании файла.

2. Вычисление «среднего» значения для заданного ключевого слова длины *N*

$$s = \frac{\sum_{n=0}^{N-1} k(n)}{N} \quad (1)$$

где *{k(n)}* коды символов ключа длины *N*.

3. Выбор строки N-2 карты усреднений.

4. Проверка условия

$$s = M(N-2, j), j = 1,2,..., L - N + 1 \quad (2)$$

Если при каком-либо *j* условие (2) выполнено, то ключевое слово присутствует в тексте.

Кроме того, значение *j* при котором выполнется условие (2) определяет и место положения ключевого слова в тексте, что в ряде случаев является важным.

Анализ современных поисковых систем, работающих с текстом, показывает, что они строятся на индексировании текста, т.е. построении специального словаря, состоящего из всех слов текстового файла. Сама по себе процедура построения такого словаря является очень громоздкой и требует большого количества ресурсов. Помимо этого требуется применять сложные процедуры лексического и морфологического анализа, т.к. одно и то же слово может иметь много окончаний, словоформ и т.д.

Предлагаемый метод работает с картой усреднения текста, в которой содержится в новом качественном виде содержимое текстового файла. Само понятие словоформы отходит на второй план и для поиска ключевых слов выполняются операции сложения и сравнения.

В ряде случаев могут быть ошибки. Например, слова «МИР» и «РИМ» будут иметь одинаковые значения *s* вычисленное по выражению (1), но при реальном поиске обычно задаются довольно длинные слова (словосочетания) и вероятность совпадений становится небольшой. При этом следует отметить, что лучше выдать избыточную информацию, чем пропустить какой либо документ.

Предложенный подход прошел экспериментальную проверку, которая подтвердила его высокую эффективность.

Литература

1. Андрианов, И.А. Применение неплотных суффиксных деревьев для поиска наибольшей общей подстроки // Методы и системы обработки информации / Муромский ин-т (филиал) Владимирского гос. ун-та. М.: 2004. С. 77-82.

2. Кудрявцев К.Я. Спектральный метод поиска ключевых слов в полнотекстовых базах данных // Информационные технологии. 2010. №4. С. 2-8.

3. Алгоритмы: построение и анализ / Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. М.: Вильямс, 2006. С. 536.

4. Hellerstein J.M., Naughton J.F., Pfeffer A. Generalized Search Trees for Database Systems, Proc. 21st Int'l Conf. on Very Large Data Bases, Zürich, Switzerland, Sep. 1995, 562-573.