ОБ ОПТИМИЗАЦИИ ПОИСКА ФРАЗ И НАБОРОВ СЛОВ В ПОЛНОТЕКСТОВОМ ИНДЕКСЕ

Веретенников А.Б.
e-mail: alexander@veretennikov.ru

Для поиска в Интернет и текстовых электронных библиотеках используются инвертированные файлы [1, 2]. Инвертированный файл представляет собой набор записей вида (ID,P), ID — идентификатор документа, P – позиция, например порядковый номер, слова в документе. Все записи, соответствующие одному слову, хранятся последовательно для их быстрого чтения при поиске. Слова в документах встречаются с различной частотой. Информация о часто используемых словах, таких как предлоги "и", "или"(стоп слова), как правило вообще не включается в индекс, т. к. они считаются малоинформативными (и их слишком много).

Но есть много слов, которые встречаются достаточно часто, и которые включаются в индекс. При индексации коллекций текстов суммарного размера 100-200 Гб. подобное слово может встретиться несколько десятков миллионов раз.

Даже если вся информация сохранена в оперативной памяти, на анализ такого количества записей при поиске требуется значительное время. Вместе с тем поиск должен быть осуществлен быстро, типичные требуемые времена поиска – меньше секунды. В результате на одном компьютере может осуществляться поиск по индексу весьма ограниченного размера.

В [3] рассмотрен метод ускорения поиска фраз, содержащих стоп слова. Он позволяет осуществлять точный поиск фраз, включающих такие слова, без хранения информации о всех вхождениях для них.

При поиске набора слов, как правило, требуется осуществлять поиск с учетом расстояния, т. е. находятся те документы, в которых данные слова находятся рядом. Частным случаем поиска с учетом расстояния является точный поиск – требуется найти документы, содержащие заданную фразу (т. е. между искомыми словами не должно быть других слов).

Предполагается, что если пользователь ввел несколько слов, то значит эти слова для него по смыслу связаны, и пользователь скорее всего ищет документ, где они располагаются рядом.

При этом, скорее всего, если расстояние между искомыми словами в конкретном документе достаточно большое, то связи между ними нет. Т. е. чем ближе друг к другу искомые слова в документе, тем этот документ является более релевантным по отношению к данному запросу. Современные поисковые системы выводят для каждого найденного вхождения заданных слов фрагмент текста, включающий данные слова. Длина данного фрагмента обычно не превосходит 20-30 слов (2-3 строки).

Т. е. можно свести задачу поиска по расстоянию к 1) поиску документов в которых искомые слова располагаются близко друг к другу (в пределах 20-30 слов) и 2) поиску документов, включающих искомые слова (данный вид поиска требует сохранения в индексе только первого вхождения каждого слова в документе, он также называется поиском без учета расстояния).

Выберем число $ProcessingDistance$ (например, 32 или 64 слова), которое будет соответствовать тому максимальному расстоянию между словами в тексте, для которого между этими словами мы будем предполагать наличие смысловой связи.

Выберем число $MaxWords$ (например, 100). Выберем число $MaxFrequency$ (например, 200). Пусть $TotalWords$ – суммарное число слов в текстах. $Margin = TotalWords\ /\ MaxFrequency$.

Пусть все слова упорядочены по количеству вхождения их в текстах. Т. е. есть список пар (w_1,c_1), (w_2,c_2) ..., w_i – слово, c_i – число его вхождений, $c_i \geq c_{i+1}\ \forall i$.

Первые (например, 100-200) из них соответствуют стоп словам. Рассмотрим следующие $MaxWords$ из этого списка.

Разделим их на группы $G_1, ..., G_q$, в каждой группе минимум 1 элемент и суммарное количество записей (ID,P), соответствующих словам группы, меньше $Margin$ (в худшем случае имеем $MaxWords$ групп). Для каждой группы создадим расширенный индекс.

Определение 1. Расширенный индекс (Advanced Index) для набора G из k слов (w_{i_1},...w_{i_k}) представляет собой индекс, в котором для каждого вхождения (ID,P) каждого слова индексируемых текстов, с условием что существует вхожение (ID,P2) некоторого w_{i_y} из G в текстах, $|P - P2| \leq ProcessingDistance$, сохранена запись (ID,P2,y).

Пусть дано слово w из группы G_i, для него создан расширенный индекс, и дано произвольное слово x. Для слова x в расширенном

индексе сохранен набор записей (ID,P,y), которые соответствуют тем вхождениям в текстах слова w, когда близко к нему было слово x, где y – порядковый номер w в его группе слов.

Т. е. из расширенного индекса мы можем получить информацию о тех вхождениях слова w, когда вблизи слова w было слово x. Эти записи располагаются в файле последовательно.

Пусть мы ищем набор слов (x_1,...,x_n). Пусть x_i слово, для которого существует расширенный индекс I. Вместо анализа списка всех вхождений x_i в текстах мы можем использовать:

1) Для каждого слова x_j, $j \neq i$ извлекаем список записей (ID,P,y) из расширенного индекса I, соответствующих x_j. Где y – равен номеру x_i в группе слов расширенного индекса.

2) Список первых вхождений слова x_i в каждом документе.

Суммарная длина данных списков скорее всего существенно меньше чем список всех вхождений x_i.

Недостаток метода заключается в том, что ориентировочный размер индекса примерно в 25 раз больше обычного полнотекстового индекса при приведенных значениях параметров.

Однако, современные диски обладают как большим размером, так и большой скоростью чтения/записи (HDD – до 3 Тб., 100 Мб./с., SSD – до 500 Гб., 1Гб./с.). Таким образом метод может быть полезен, несмотря на кажущиеся большими (психологически) затраты дискового пространства.

В настоящее время данный метод частично реализован автором и производятся различные эксперименты с ним.

Литература

[1] *Prywes, N. S., Gray, H. J.* The organization of a Multilist-type associative memory. IEEE Trans. on Communication and Electronics, 68 (1963), 488-492.

[2] *Zobel, Justin, and Alistair Moffat* Inverted files for text search engines. ACM Computing Surveys 38(2), 2006, Article 6.

[3] *Веретенников А.Б.* Программный комплекс и эффективные методы организации и индексации больших массивов текстов. Диссертация на соискание ученой степени кандидата физико-математических наук. Екатеринбург, 2009.