ГИБКИЙ ПОДХОД К ПРОБЛЕМЕ ПОИСКА ПОХОЖИХ ДОКУМЕНТОВ

Веретенников А.Б.
e-mail: abvmt@e1.ru

Поиск похожих документов является весьма важной задачей. Рассмотрим электронные библиотеки: над большим объемом данных работают параллельно много людей, которые одновременно могут добавить в библиотеку один и тот же документ. Еще больше похожих документов в сети Интернет. Многие страницы копируются с одного сайта на другие, возникают новые версии документов с небольшими изменениями. Некоторые сайты помещают в URL документа ID сессии, таким образом, загруженный два раза в разные сессии один и тот же документ имеет разный URL.

Разными авторами используется понятие шингл (shingle) или его аналоги. Шингл – последовательность длины k, элементами которой обычно являются слова или символы текста. Рассматриваются все такие последовательности, например, рассмотрим следующий текст

$a \quad rose \quad is \quad a \quad rose \quad is \quad a \quad rose$

При $k = 4$ получаем следующие шинглы

$$\begin{array}{llllll} a & rose & is & a & & \\ & rose & is & a & rose & \\ & & is & a & rose & is \\ & & & a & rose & is & a \\ & & & & rose & is & a & rose \end{array}$$

На основании последовательности вычисляется некоторое значение (например с помощью хеширования), далее обрабатываются уже эти значениями, а не сами шинглы.

В работе [1] для документа вычисляются все шинглы, затем из этого множества выбирается подмножество шинглов (всех шинглов очень много), например берутся те шинглы, у которых последние m бит равны 0. Похожесть двух файлов определяется по количеству общих элементов в их наборах шинглов.

В работе [4] для определения похожести документов выбирается некоторое множество из N слов, затем для каждого документа вычисляется N-мерный двоичный вектор, i-й элемент вектора равен 1, если i-е слово достаточно часто встречается в документе. На основании сравнения данных векторов определяется похожесть документов.

В данной работе похожесть документов понимается как значительное совпадение фрагментов текста в рассматриваемых документах. В частности решается задача поиска практически одинаковых документов.

Основное требование: гарантированное нахождение похожих документов, а также определение степени похожести документов.

При определении похожести документов автор использует анализ морфологии. Использовался морфологический анализатор [6].

Анализатор позволяет по словоформе получить набор ее базовых форм. Для большинства словоформ существует только одна базовая форма, однако для многих словоформ существует несколько базовых форм. Количество базовых форм слов в анализаторе – около 200 000.

Рассмотрим возможные операции изменения документа

1) Удаление фрагмента, состоящего из нескольких предложений.
2) Перестановка предложений.
3) Удаление слов или вставка слов в предложение.
4) Переформатирование текста.
5) Изменение формы слова.
6) Перестановка слов в предложении местами.
7) Замена слова на синоним.

Разработанный алгоритм определяет документ A и документ B, который получен из A путем указанных 1-3 операций, как похожие. Степень похожести зависит от количества примененных операций.

Для операций 4-6 поставлено условие: при применении этих операций к документу, полученный документ должен рассматриваться как практически не изменившийся.

Пункт 5 в настоящее время учтен с некоторыми ограничениями, в частности не учитывается изменение формы слова, не входящего в словарь морфологического анализатора.

Операция 7 как 4-6 рассматривается как не меняющая документ. По мнению автора это может быть учтено с помощью словаря синонимов, но в настоящее время не реализовано.

Можно ли использовать частоты слов? Если из документа удалить половину – получим два документа, новый документ полностью совпадает с частью первоначального документа, при этом скорее всего частоты слов в них разные. В то же время автор считает, что фрагмент документа должен считаться похожим для исходного документа.

Поэтому как базовый элемент анализа выбрано предложение.

Введем две хеш функции: WH ставит в соответствие слову число, SH ставит в соответствие предложению число.

Для SH поставим следующие условия:

$SH.1$) При изменении порядка слов предложения SH не меняется.

$SH.2$) При изменении формы слова SH не меняется.

Чтобы вычислить хеш предложения вначале с помощью WH вычислим хеш для каждого слова.

Если слово входит в словарь анализатора, то анализатор возвращает список ID базовых форм слова. В качестве значения WH в этом случае берем сумму этих ID или сумму квадратов ID.

Если слово не входит в словарь анализатора, определим WH на основании кодов символов слова. Возьмем сумму возведенных в 4-ю степень кодов символов. Показатель степени выбран из соображений соразмерности хеша слова входящего в словарь анализатора и хеша слова не входящего в словарь (код символа находится в диапазоне 0 – 255, ID базовой формы в диапазоне от 1, до, примерно, 200 000).

SH определим как сумму значений WH слов предложения. $SH.1$ выполнено. $SH.2$ выполнено частично, не учитывается изменение формы слова, не входящего в словарь анализатора, и возможные различия в наборах базовых форм для разных словоформ, порожденных от одной базовой формы.

Документ A это последовательность предложений: $A = A_1, ..., A_n$. Вычислим массив 32-битных чисел: $Seq_1(A) = SH(A_1), ..., SH(A_n)$. Рассмотрим другую последовательность:

$Seq_2(A) = SH(A_1) + SH(A_2), SH(A_2) + SH(A_3), ..., SH(A_n) + 0,$

которая более подходит нам с учетом того, что при изменении порядка предложений похожесть документов уменьшается.

Пусть $Count(X, L)$ – количество вхождений значения X в массиве чисел L. Введем для A множество $DSH(A) = \{Seq_2(A)[i] | i = 1, ..., n\}$.

Определим степень похожести A с документом B как число хешей A, которые встречаются в наборе хешей B, с учетом возможных повторений.

$X(A, B) = \sum_{V \in DSH(A)} min(Count(V, Seq_2(A)), Count(V, Seq_2(B))$.

Автор разработал эффективный алгоритм для решения задачи.

Далее приведены результаты работы алгоритма.

Эксперименты сделаны на следующей конфигурации. Процессор: Intel Core 2 Duo E6700, 2.66 GHz, кэш: L1 Seq – 2 x 32 кб, L1 inst. 2 x 32

кб, L2 – 4096 кб. Оперативная память: 4 гб, DDR2 800. HDD: Seagate Barracuda 7200.10, 7200 RPM, кэш 16 мб., объем 750 гб. FSB 1066 MHz. В экспериментах использовалось 1 гб. оперативной памяти.

Обрабатывались 400 тысяч документов общим объемом 86 гб. Время поиска похожих документов: 1 ч. 9 мин. (не включает времени чтения файлов, т. к. оно зависит от разных факторов, не имеющих отношения к алгоритму, и времени вычисления SH, т. к. оно мало).

Был проведен ряд экспериментов, показывающих высокое качество работы алгоритма, например, в одном из них рассмотрены документы:

1) Братья Карамазовы, 1.7МБ, 6300 строк.

2) Братья Карамазовы 2-й документ, другое форматирование, 1,69 мб, 31 тысячи строк.

3) Братья Карамазовы (фрагмент), 17 кб., 64 строк.

4) Братья Карамазовы (фрагмент), 51 кб., 184 строк.

5) Братья Карамазовы (фрагмент), 800 кб., 307 строк.

6) Еще указанная ранее база из 86 гб. документов.

В данной таблице в строке с номером i показаны похожие документы для i-го документа, определенные алгоритмом.

	1	2	3	4	5
1		+			
2	+				
3	+	+			
4	+	+			+
5	+	+			

В пункте 6 обнаружено 10 файлов, содержащих текст книги "Братья Карамазовы" с различным форматированием.

Был проведен также следующий эксперимент:

Взято 639 файлов, 593 мб., для каждого файла создано 100 фрагментов. Начало и размер фрагмента определены с помощью генератора псевдослучайных чисел. Размер фрагмента не менее 8 кб. Суммарный размер (с фрагментами) 4,28 гб.

Результаты, которые хотелось бы получить: 1) Каждый фрагмент должен быть определен алгоритмом как похожий своему исходному документу. 2) Каждый фрагмент одного документа похож фрагменту другого документа, только если оба исходных документа похожи. (Отсутствие ошибок поиска).

Полученные результаты: время работы 54 с., пункт 1 выполнен для всех фрагментов. Пункт 2 также выполнен для всех фрагментов. Примечание: обнаружены похожие фрагменты для 19 разных исход-

ных документов, но в результате ручной проверки определено, что соответствующие исходные документы практически одинаковые.

Эксперимент показывает высокое качество работы алгоритма, На данном наборе алгоритм проработал идеально.

Достоинство алгоритма в том, что по сути он вычисляет величину похожести по указанной формуле, и если, два документа похожи между собой (в рассмотренном смысле), то их похожесть гарантированно будет обнаружена. Скорость работы зависит от суммарного объема документов, и в гораздо меньшей степени от их количества.

Список литературы

[1]. *U. Manber.* Finding Similar Files in a Large File System. Winter USENIX Technical Conference, 1994.

[2]. *A. Broder, S. Glassman, M. Manasse and G. Zweig.* Syntactic clustering of the Web. Proc. of the 6th International World Wide Web Conference, April 1997.

[3]. *S. Brin, J. Davis, H. Garcia-Molina.* Copy Detection Mechanisms for Digital Documents. Proceedings of the ACM SIGMOD Annual Conference, San Francisco, CA, May 1995.

[4]. *Sergey Ilyinsky, Maxim Kuzmin, Alexander Melkov, Ilya Segalovich* An efficient method to detect duplicates of Web documents with the use of inverted index. THE ELEVENTH International World Wide Web Conference, Sheraton Waikiki Hotel Honolulu, Hawaii, USA 7-11 May 2002.

[5]. *Зеленков Ю.Г, Сегалович И.В.* Сравнительный анализ методов определения нечетких дубликатов для Web-документов. Труды 9ой Всероссийской научной конференции «Электронные библиотеки: перспективные методы и технологии, электронные коллекции» — RCDL'2007, Переславль-Залесский, Россия, 2007.

[6]. *Лукач Ю. С.* Быстрый морфологический анализ флективных языков. Международная алгебраическая конференция: К 100-летию со дня рождения П. Г. Конторовича и 70-летию Л. Н. Шеврина. Тез. докл. Екатеринбург: Изд-во Урал. ун-та, 2005, с. 182-183.