Порождение кластеров документов-дубликатов: подход, основанный на поиске частых замкнутых множеств признаков.

Сергей О. Кузнецов, Дмитрий И. Игнатов,
Сергей А. Объедков, Михаил В. Самохин

ВИНИТИ РАН
serge@viniti.ru

Аннотация

Множество документов в Интернете имеют дубликаты, в связи с чем необходимы средства эффективного вычисления кластеров документов-дубликатов [1-5, 8-10, 13-14]. В работе исследуется применение алгоритмов Data Mining для поиска кластеров дубликатов с использованием синтаксических и лексических методов составления образов документов. На основе экспериментальной работы делаются некоторые выводы о способе выбора параметров методов.

Generating Clusters of Duplicate Documents: An Approach Based on Frequent Closed Itemsets

Sergei O. Kuznetsov, Dmitrii I. Ignatov,
Sergei A. Obiedkov, Mikhail V. Samokhin

Abstract

A vast amount of documents in the Web have duplicates, which necessitates creation of efficient methods for computing clusters of duplicates [1-5, 8-10, 13-14]. In this paper some algorithms of Data Mining are used for constructing clusters of duplicate documents (duplicates), documents being represented by both syntactic and lexical methods. Series of experiments suggest some conclusions about choosing parameters of the methods.

1.Введение

Около 30% документов в Интернете имеют дубликаты, в связи с чем поисковые машины должны обладать эффективными средствами вычисления кластеров дубликатов. Происхождение дубликатов может быть разным — от дублирования компаниями собственной информации на разных серверах (создание зеркал) до злонамеренных — обмана программ индексаторов веб-сайтов, незаконного копирования и спамерских рассылок.

Обычно дубликаты документов определяются на основе отношения сходства на парах документах: два документа сходны, если некоторая числовая мера их сходства превышает некоторый порог (см, например [1-3]). По отношению сходства вычисляются кластеры сходных документов, например, по транзитивному замыканию отношения сходства [3].

При нахождении множеств (полу)дубликатов документов основными являются следующие стадии (см., например [3]): представление документов множеством признаков, составление образа документа путем выбора подмножества признаков, определение отношения сходства на образах документов и вычисление кластеров сходных документов. Вначале, после снятия HTML-разметки, документы, как линейные последовательности слов (символов), преобразуются во множества. Здесь двумя основными схемами (определяющими весь возможный спектр смешанных методов) являются синтаксический (выбор последовательностей символов, слов, или предложений) и лексический метод (выбор представительных слов).

К синтаксическим относится метод шинглирования (от англ. shingle – чешуйка), когда документ, очищенный от разметки (и, возможно, пробелов и знаков препинания), сперва представляется набором всех подцепочек символов (или слов и т.п.) определенной длины. Каждой такой цепочке, называемой шинглом,

сопоставляется хеш-код (равенство цепочек гарантирует равенство кодов). Из множества хеш-кодов цепочек, в соответствии с некоторой схемой рандомизации, выбирается подмножество, служащее образом документа таким образом, что совпадение образов документов говорит о «сходстве» самих документов (см. раздел 2.1). Данный метод находил применение в поисковой системе AltaVista и, если судить по патенту [14], также и в Google.

Среди способов выбора подцепочек известны методы выбора фиксированного (или логарифмического, как в Яндекс-почте) от длины текста числа подцепочек, выбор каждой k-й цепочки и т.д.

В методах лексического типа реализуется отбор множества представительных слов исходя из показателей значимости этих слов. Обычно показателями значимости служат частотные характеристики: отбираются слова (кроме слов из заранее фиксированного стоп-списка), чьи частоты лежат в некотором интервале (так как высокочастотные слова могут быть неинформативными, а низкочастотные — опечатками или случайными словами).

В лексических методах, таких как, например, I-Match [5], используют большой текстовый корпус для порождения лексикона, то есть набора представительных слов. Документ представляется множеством тех его слов, которые входят в лексикон. При порождении лексикона отбрасываются самые низко- и высоко-частотные слова. I-Match порождает сигнатуру документа (множество термов), а по ней — хеш-код документа, причем два документа получают один хеш-код с вероятностью равной их мере сходства (по метрике косинуса). Как указывается в [10], I-Match порой неустойчив к изменениям текста, например, к рандомизации по существу одних и тех же спамерских сообщений. Для устранения этого недостатка, помимо стандартной сигнатуры, создается еще K сигнатур, каждая из которых получается случайным удалением некоторой доли всех термов из исходной сигнатуры (т.о. все новые сигнатуры суть подмножества исходной). Два документа считаются почти дублями, если их наборы из K + 1 сигнатуры имеют большое пересечение хотя бы по одной из них. В [10] указывается на сходство такого подхода с подходом на основе супершинглов (конкатенация шинглов), когда сходство документов определяется как совпадение хотя бы одного супершингла.

В лексическом методе [9] большое внимание уделяется построению словаря — набора дескриптивных слов, который

должен быть небольшим, но хорошо покрывать коллекцию, а присутствие каждого из слов в образе документа устойчиво по отношению к малым изменениям документов.

На втором этапе из документа, представленного множеством синтаксических или лексических признаков, выбирается подмножество признаков, образующее краткое описание (образ) документа. В синтаксических методах такого рода отбор чаще всего осуществляется с помощью схем рандомизации [1-3], в лексических методах — с помощью тех или иных методов выбора *существенных* слов, например, на основе заранее созданных словарей, или на основе какой-либо меры существенности слова для текста.

На третьем этапе определяется отношение сходства на документах. Чаще всего для этого используется некоторая метрика сходства, сопоставляющая двум документам число в интервале [0, 1], характеризующее сходство и некоторый параметр — порог, превышение которого свидетельствует о большом сходстве документов или о том, что документы являются (почти) дубликатами друг друга.

На основе отношения сходства документы объединяются в кластеры (полу-)дубликатов. Определение кластера также может варьироваться. Одно из возможных определений, часто используемых на практике (например, в компании AltaVista), но наиболее слабых, упоминается в обзоре [3]: если документам Интернета сопоставить граф, вершины которого соответствуют самим документам, а ребра – отношению «быть (почти) дубликатом», то кластером объявляется компонента связности такого графа. Достоинством такого определения является эффективность вычислений: компоненту связности можно вычислить за линейное время от числа ребер. Недостаток такого подхода очевиден: отношение «быть (почти) дубликатом» не является транзитивным, поэтому в кластер сходных документов могут попасть абсолютно разные документы. Противоположным — «самым сильным» — определением кластера, исходя из отношения «быть (почти) дубликатом», было бы рассмотрение в его качестве клик графа (максимальных по вложению полных подграфов). При этом каждый документ из кластера должен быть сходным со всеми другими документами того же кластера. Такое определение кластера более адекватно передает представление о групповом сходстве, но, к сожалению, практически не применимо в масштабе Интернета, в силу того, что поиск клик в графе – классическая труднорешаемая задача.

Два приведенных выше противоположных по силе определения кластера допускают целый спектр промежуточных формулировок, в которых можно было бы находить необходимый баланс между адекватностью определения кластеров (с точки зрения их соответствия множествам «в самом деле» сходных документов) и сложностью вычисления кластеров.

Другие методы определения кластеров основаны на вариациях стандартных методов кластерного анализа, например, когда при отнесении очередного объекта к кластеру используется дистанция до «центров масс» кластеров. Такого рода методы существенно зависят от последовательности поступления объектов, образующих кластеры. Применительно к задаче выявления дубликатов это означает, в частности, что документы, попавшие в поле зрения раньше, сильнее определяют структуру кластера, чем документы, ставшие известными позднее.

В данной работе мы рассматриваем сходство не как отношение на множестве документов, а как операцию, сопоставляющую двум документам множество общих элементов их сокращенных описаний, где в качестве элементов описания выступают либо синтаксические единицы («шинглы») либо лексические единицы («представительные слова»). Кластер сходных документов определяется как множество всех документов с определенным множеством общих элементов описания. Кластер дубликатов определяется как множество документов, у которых число общих элементов описания превышает определенный порог. В работе приводятся результаты экспериментальной проверки данного метода на основе сравнения результатов его применения (для разных значений порогов) со списком дубликатов, составленным на основе результатов применения других методов к тому же множеству документов.

Мы исследовали влияние следующих параметров модели на результат:

- использование синтаксических или лексических методов представления документов,
- использование методов «n минимальных элементов в перестановке» или «минимальные элементы в n перестановках» [1-3], (Второй метод, обладая лучшими теоретическими свойствами, вычислительно более трудоемок.)
- параметры шинглирования,
- величина порога сходства образов документов.

Одной из задач проекта было связать вычисление попарного сходства образов документов с построением кластеров документов, так чтобы, с одной стороны, получаемые кластеры были бы независимы от порядка рассмотрения документов (в отличие от методов кластерного анализа), а с другой стороны гарантировали бы наличие реального попарного сходства всех образов документов в кластере. При решении этой задачи нами использовался метод построения «частых замкнутых множеств признаков» (frequent closed itemsets), ставших в последние годы одной из центральных тем исследования в Data Mining [12], в том числе, при решении задач анализа данных с очень большим размером входа. В следующем разделе мы рассматриваем математическую модель, лежащую в основе методов составления образов документов и методов поиска кластеров сходных документов. В третьем разделе мы даем краткое описание программных средств и структуры эксперимента. В четвертом разделе мы обсуждаем результаты компьютерного эксперимента и ставим дальнейшие задачи.

2. **Описание вычислительной модели**

2.1. Образ документа

В качестве методов представления документов (создания образа документа) мы использовали стандартные синтаксические и лексические подходы с разными параметрами.

В рамках синтаксического подхода была реализована схема шинглирования и составление краткого образа (скетча) документов на основе методов «n минимальных элементов в перестановке» и «минимальные элементы в n перестановках», описание которого можно найти, например, в [Broder 1998, Broder 2000]. Программа shingle с двумя параметрами length и offset порождает для каждого текста набор последовательностей слов (шинглов) длины length, так что отступ от начала одной последовательности до начала другой последовательности в тексте имеет размер offset. Полученное таким образом множество последовательностей хэшируется, так что каждая последовательность получает свой хэш-код.

Далее из множества хэш-кодов, соответствующему документу, выбирается подмножество фиксированного (с помощью параметра) размера с использованием случайных перестановок, описанных в работах [1-3]. При этом вероятность того, что минимальные

элементы в перестановках хэш-кодов на множествах шинглов документов A и B (эти множества обозначаются через F_A и F_B, соответственно) совпадут, равна мере сходства этих документов *sim*(A,B):

$$P[\min\{\pi(F_A)\} = \min\{\pi(F_B)\}] = \frac{|F_A \cap F_B|}{|F_A \cup F_B|} = sim(A, B)$$

Перестановки (которые можно представлять как перенумерации шинглов) реализуются с помощью произведений векторов-строк, представляющих множество хэш-кодов шинглов документа в двоичном виде, на случайные бинарные матрицы, соответствующие перестановкам. Для каждого хэш-кода из множества хэш-кодов документа вычисляется номер в каждой из случайных перестановок. Номер в перестановке получается как произведение хэш-кода, представленного в виде двоичного вектора, на случайно порожденную бинарную матрицу, соответствующую перестановке. Число используемых перестановок также является параметром. Для каждой из перестановок (задаваемой случайной матрицей с определенными свойствами) выбирается минимальный элемент, то есть шингл (а точнее его хэш-код), получивший при перестановке минимальный номер. Образом документа в методе «n минимальных элементов в перестановке» есть n минимальных (первых) хэш-кодов для случайной перестановки, а в методе «минимальные элементы в n перестановках» есть множество минимальных элементов для n случайных перестановок. В обоих методах образы всех документов имеют фиксированную длину n.

В рамках лексического подхода нами применялся элементарный метод, основанный на представлении документа множеством слов с частотой из заданного интервала частот (такая процедура используется, например, в [5]), точнее использовалось инвертированное представление (естественным образом получаемое из индекса коллекций), в котором для слова указываются URL, в которых оно встречается.

В индексе коллекции (основная информация в котором – о вхождении слова в документы коллекции, стоп-слова удалены), предоставленного Яндексом, были оставлены
слова с относительной частотой по коллекции в интервале [0.00001, 0.5] от слова с максимальной частотой, затем для каждого слова из индексного файла удалялись повторяющиеся идентификаторы документов, содержащих данное слово. Это делается за один проход

по документу. Таким образом получался инвертированный список «слово-документ», который можно подавать на вход следующего этапа (программы, вычисляющей частые замкнутые множества)

2.2. Определение сходства и кластеров сходства на основе поиска частых замкнутых множеств

Основные определения, связанные с частыми замкнутыми множествами, естественно даются в терминах анализа формальных понятий (АФП) [6]. Контекстом в АФП называют тройку K = (G, M, I), где G — множество объектов, M — множество признаков, а отношение $I \subseteq G \times M$ говорит о том, какие объекты какими признаками обладают. Для произвольных $A \subseteq G$ и $B \subseteq M$ определены операторы Галуа:

$A' = \{m \in M \mid \forall g \in A\ (g\ I\ m)\}$;
$B' = \{g \in G \mid \forall m \in B\ (g\ I\ m)\}$.

Оператор '' (двукратное применение оператора ') является *оператором замыкания*: он идемпотентен (A'''' = A''), монотонен ($A \subseteq B$ влечет $A'' \subseteq B''$) и экстенсивен ($A \subseteq A''$). Множество объектов $A \subseteq G$, такое, что A'' = A, называется *замкнутым*. Аналогично для замкнутых множеств признаков - подмножеств множества M. Пара множеств (A, B), таких, что $A \subseteq G$, $B \subseteq M$, A' = B и B' = A, называется *формальным понятием* контекста K. Множества A и B замкнуты и называются *объемом* и *содержанием* формального понятия (A, B) соответственно. Для множества объектов A множество их общих признаков A' служит описанием сходства объектов из множества A, а замкнутое множество A'' является кластером сходных объектов (с множеством общих признаков A'). Для произвольного $B \subseteq M$ величина $|B'| = |\{g \in G \mid \forall m \in B\ (g\ I\ m)\}|$ называется *поддержкой* (support) B и обозначается sup(B). Нетрудно видеть, что множество B замкнуто тогда и только тогда когда для любого $D \supset B$ имеет место $sup(D) < sup(B)$. Именно это свойство используется для определения замкнутости в методах Data Mining. Множество $B \subseteq M$ называется *k-частым* если $|B'| > k$ (то есть множество признаков B встречается в более чем k объектах), где k – параметр. Вычисление частых замкнутых множеств признаков (содержаний) приобрело важность в Data Mining благодаря тому, что по этим множествам эффективно вычисляются множества всех ассоциативных правил [12].

Хотя теоретически размер множества всех замкнутых множеств признаков (содержаний) может быть экспоненциальным относительно числа признаков, на практике, таблицы данных сильно «разрежены» (то есть среднее число признаков на один объект весьма мало) и число замкнутых множеств невелико. Для таких случаев существуют весьма эффективные алгоритмы построения всех наиболее частых замкнутых множеств признаков (см. также наш обзор по алгоритмам построения всех замкнутых множеств [11]). В последние годы проводился ряд соревнований по быстродействию таких алгоритмов на серии международных семинаров под общим названием FIMI (Frequent Itemset Mining Implementations). Пока лидером по быстродействию считается алгоритм FPmax* [7], показавший наилучшие результаты по быстродействию в соревновании 2003 года. Мы использовали этот алгоритм для построения сходства документов и кластеров сходных документов. При этом в роли объектов выступали элементы описания (шинглы или слова), а в роли признаков – документы. Для такого представления «частыми замкнутыми множествами» являются замкнутые множества документов, для которых число общих единиц описания в образах документов превышает заданный порог.

3. Программная реализация метода

Программные средства для проведения экспериментов в случае синтаксических методов включали следующие блоки:

1.Парсер формата XML (предоставлен Яндексом) для коллекции ROMIP
2. Снятие html-разметки.
3. Нарезка шинглов с заданными параметрами
4. Хэширование шинглов
5. Составление образа документов путем выбора подмножества (хэш-кодов) шинглов с помощью методов «n минимальных элементов в перестановке» и «минимальные элементы в n перестановках».
Класс NElementsPageHandler реализует метод «n минимальных элементов в перестановке».
Последовательно обрабатываются данные файлов в формате xml из коллекции ROMIP. Каждый документ, имеющий URL_Id, шинглируется с заданными параметрами <длина_шингла>

<смещение> <размер_образа>, результаты шинглирования помещаются в выходной файл(ы) <shingles.fps>. По умолчанию из множества всех хешей шинглов данного документа выбирается сто минимальных (в смысле перестановок) хешей. После выборки в выходной файл в поле Fingerprint записывается выбранный хеш соответствующего шингла.
Класс MatrixPageHandler реализует «минимальные элементы в n перестановках».
Последовательно обрабатываются данные файлов в формате xml коллекции ROMIP. Каждый документ, имеющий URL_Id, шинглируется с заданными параметрами <длина_шингла> <смещение> <размер_образа>. После разбиения документа на шинглы и хеширования используется техника перестановок, а именно случайная выборка заданного количества отпечатков для конечного представления документа из множества имеющихся отпечатков. Для получения выборки порождаются случайные матрицы определенного вида, на которые затем умножается вектор хешей для каждого документа. Результаты шинглирования помещаются в выходные файлы <shingles.fps> . Отпечаток (хеш) данного шингла для документа DocId записывается в выходной файл в поле Fingerprint.

6. Составление по результатам методов 4-5 инвертированной таблицы «список идентификаторов документов - шингл» - подготовка данных к формату программ вычисления замкнутых множеств .
Класс SortAndMerge.
Модуль сортирует данные из файлов <shingles.fps> по полю Figerprint в порядке убывания, затем идущие подряд записи с одинаковым значением поля Fingerprint объединяются в списки вида <DocId1, DocId2,…, DocIdn> и помещаются в выходной файл <shingles.txt.out>. В одной строке нет повторяющихся DocId.
7. Вычисление частых замкнутых множеств с заданным порогом общего числа документов, в которое входит данное множество шинглов: программа MyFim (реализующая алгоритм FPmax*)
Алгоритм, лежащий в основе этого модуля, позволяет быстро находить максимально частые подмножества среди множества набора признаков с порогом на общее число объектов имеющих данный признак. Это означает, что в результате обработки файла <shingles.txt.out> с записями вида <DocId1, DocId2,…, DocIdn> получим выходной файл <duplicates.txt> вида <DocId1, DocId2,…, DocIdn, CountCommonFinerpints>. Здесь CountCommonFingerprints

есть число общих отпечатков документов из списка DocId1, DocId2,…, DocIdn (в списке нет повторяющихся DocId).

Блоки 1-7.
Вход : файлы коллекции ROMIP в формате XML
Выход: набор строк, первые элементы (через пробел) – идентификаторы документов, последний элемент - число общих шинглов у документов из предшествующего списка.

8. Сравнение со списком дубликатов РОМИП – программа Comparator.
Вход: выход программы MyFim (см. выше) и файл со списком пар документов-дубликатов РОМИП: каждая строка файла содержит пару идентификаторов документов, являющихся дублями.
Выход:
Текстовый лог-файл с пятью строками
1) число всех пар дублей ROMIP; 2) число всех пар для нашей реализации методов; 3) число уникальных пар дублей ROMIP; 4) число уникальных пар найденных нашими методами; 5) число общих пар.
Список уникальных пар дублей ROMIP и список уникальных пар дублей, найденных нами, в формате CSV.

Для «лексического» метода основные блоки таковы:

1.Утилита printkeys:
Выход: файл записей вида
<слово>,
<DocId1,
… ,
DocId2,
… ,
DocIdn>,
с повторением идущих подряд идентификаторов документов.
2. Класс Lex.
Выход:
файл с записями вида <DocId1, DocId2,…, DocIdn>, т.е. списки документов, имеющих данное общее слово.
3. Модуль алгоритма FPmax (программа myFim.exe)
В результате обработки входного файла с записями вида <DocId1, DocId2,…, DocIdn> получим выходной файл <duplicates.txt> вида <DocId1, DocId2,…, DocIdn, CountCommonWords>, где CountCommonWords есть число общих слов документов из списка

DocId1, DocId2,…, DocIdn. Естественно в списке нет повторяющихся DocId.
Блоки 1-3.
Вход: файлы индекса коллекции ROMIP indexkey и indexinv.
Выход: набор строк, первые элементы (через пробел) – идентификаторы документов, последний элемент - число общих слов у документов из предшествующего списка
3. Сравнение со списком дубликатов РОМИП – программа Comparator.

4. Эксперименты с данными РОМИП

В качестве экспериментального материала нами использовалась URL-коллекция РОМИП, состоящая из 52 файлов, общего размера 4,04 Гб. Для проведения экспериментов коллекция разбивалась на несколько частей, включающих от трех до двадцати четырех файлов (приблизительно от 5% до 50% от размера всей коллекции). Параметры шинглирования, использовавшиеся в экспериментах: число слов в шингле 10 и 20, отступ между началом соседних шинглов 1. Данное значение отступа означает, что начальное множество шинглов включало все возможные последовательности цепочек слов.

В экспериментах с синтаксическим методом представления исследовались оба способа составления образа документа, описанные в разделе 2.1: «n минимальных элементов в перестановке» и «минимальные элементы в n перестановках».

Размеры результирующих образов документов выбирались в пределах от 100 до 200 шинглов. В случае «лексического» представления, описанного в разделе 2.1, для каждого документа в образ документа отбирались только слова, попавшие в итоговый словарь (множество дескриптивных слов).

В качестве порогов, определявших «частые замкнутые множества» (то есть числа общих шинглов в образах документов из одного кластера) нами исследовались различные значения в интервалах, правая часть которых совпадала с числом шинглов в образе документа, например интервал [85, 100] для образов документов размером 100 шинглов, интервал [135, 150] для образов документов размером 150 шинглов, и т.д. Очевидно, что при выборе в качестве порога верхнего значения из указанных интервалов в кластер дубликатов попадали только те документы, образы которых совпадали полностью.

Для значений параметров из указанных интервалов исследовалось соответствие кластеров дубликатов списку дубликатов РОМИП. Для каждой пары из списка дубликатов РОМИП искался кластер «частых замкнутых множеств», в котором содержится данная пара, и обратно, для каждого кластера замкнутых множеств, каждая пара документов из какого-либо кластера частых замкнутых множеств искалась в списке дубликатов РОМИП. Выходом данного модуля являлась таблица с указанием общего числа пар дубликатов ВИНИТИ (пар дубликатов, имеющихся в списке РОМИП) и число уникальных пар дубликатов ВИНИТИ (то есть пар, не входящих в список дубликатов РОМИП). Эти числа позволяют подсчитывать полноту и точность наших методов относительно списка пар РОМИП.

Замерялось время, затрачиваемое на стадии шинглирования, составления образов документов и порождения кластеров сходных документов. Эксперименты проводились на персональном компьютере P-IV HT с тактовой частотой 3.0 ГГц, объемом оперативной памяти 1024 Мб и операционной системой Windows XP Professional. Результаты экспериментов и время, затраченное на их проведение, частично приводятся в следующих таблицах

1. Результаты работы метода “n минимальных элементов в перестановке”.

MyFim		**Все пары дублей**		**Уникальные пары дублей**		**Общие пары**
Вход	**Порог**	**Яндекс**	**ВИНИТИ**	**Яндекс**	**ВИНИТИ**	
b_1_20_s_100_n1-6.txt	100	33267	7829	28897	3459	4370
b_1_20_s_100_n1-6.txt	95	33267	11452	26729	4914	6538
b_1_20_s_100_n1-6.txt	90	33267	17553	22717	7003	10550
b_1_20_s_100_n1-6.txt	85	33267	22052	21087	9872	12180
b_1_20_s_100_n1-12.txt	100	105570	15072	97055	6557	8515
b_1_20_s_100_n1-12.txt	95	105570	20434	93982	8846	11588
b_1_20_s_100_n1-12.txt	90	105570	30858	87863	13151	17707
b_1_20_s_100_n1-12.txt	85	105570	41158	83150	18738	22420
b_1_20_s_100_n1-	100	191834	41938	175876	25980	15958

24.txt						
b_1_20_s_100_n1-24.txt	95	191834	55643	169024	32833	22810
b_1_20_s_100_n1-24.txt	90	191834	84012	155138	47316	36696
b_1_20_s_100_n1-24.txt	85	191834	113100	136534	57800	55300
b_1_10_s_120_n1-6.txt	120	33267	7725	29065	3523	4202
b_1_10_s_120_n1-6.txt	115	33267	11763	26586	5082	6681
b_1_10_s_120_n1-6.txt	110	33267	11352	26547	4632	6720
b_1_10_s_150_n1-6.txt	150	33267	6905	28813	2451	4454
b_1_10_s_150_n1-6.txt	145	33267	9543	27153	3429	6114
b_1_10_s_150_n1-6.txt	140	33267	13827	24579	5139	8688
b_1_10_s_150_n1-6.txt	135	33267	17958	21744	6435	11523
b_1_10_s_150_n1-6.txt	130	33267	21384	19927	8044	13340
b_1_10_s_150_n1-6.txt	125	33267	24490	19236	10459	14031
b_1_10_s_180_n1-6.txt	170	33267	9834	27457	4024	5810
b_1_10_s_180_n1-6.txt	130	33267	38402	20142	25277	13125
b_1_10_s_180_n1-6.txt	120	33267	55779	19966	42478	13301
b_1_10_s_200_n1-6.txt	200	33267	5083	29798	1614	3469
b_1_10_s_200_n1-6.txt	195	33267	6700	28661	2094	4606
b_1_10_s_200_n1-6.txt	190	33267	8827	27516	3076	5751
b_1_10_s_200_n1-6.txt	170	33267	12593	25866	5192	7401
b_1_10_s_200_n1-6.txt	135	33267	48787	19987	35507	13280
b_1_10_s_200_n1-6.txt	130	33267	57787	19994	44514	13273

2. Результаты работы метода “минимальные элементы в n перестановках”.

MyFim		**Все пары дублей**		**Уникальные пары дублей**		**Общие пары**
Вход	**Порог**	**Яндекс**	**ВИНИТИ**	**Яндекс**	**ВИНИТИ**	
m_1_20_s_100_n1-3.txt	100	16666	4409	14616	2359	2050
m_1_20_s_100_n1-3.txt	95	16666	5764	13887	2985	2779
m_1_20_s_100_n1-3.txt	90	16666	7601	12790	3725	3876
m_1_20_s_100_n1-	85	16666	9802	11763	4899	4903

3.txt						
m_1_20_s_100_n1-6.txt	100	33267	13266	28089	8088	5178
m_1_20_s_100_n1-6.txt	95	33267	15439	26802	8974	6465
m_1_20_s_100_n1-6.txt	90	33267	19393	24216	10342	9051
m_1_20_s_100_n1-12.txt	100	105570	21866	95223	11519	10347
m_1_20_s_100_n1-12.txt	95	105570	25457	93000	12887	12570

3. Время вычисления образа документов для метода “n минимальных элементов в перестановке” и метода “минимальные элементы в n перестановках”.

Подколлекция	**Число документов**	**Метод**	**Время работы, с**	**Параметры шинглирования**	
				Длина шингла	**Размер образа**
narod.1-3.xml	26077	n-min	312	20	100
narod.1-6.xml	53539	n-min	622	20	100
narod.1-12.xml	110997	n-min	1360	20	100
narod.1-24.xml	223804	n-min	2435	20	100
narod.1-3.xml	26077	min в n	924	20	100
narod.1-6.xml	53539	min в n	1905	20	100
narod.1-12.xml	110997	min в n	3617	20	100
narod.1-24.xml	223804	min в n	7501	20	100
narod.1-3.xml	26077	n-min	277	10	100
narod.1-6.xml	53539	n-min	563	10	100
narod.1-12.xml	110997	n-min	1118	10	100
narod.1-24.xml	223804	n-min	2348	10	100
narod.1-3.xml	110997	n-min	315	10	120
narod.1-6.xml	223804	n-min	622	10	120
narod.1-3.xml	110997	n-min	388	10	150
narod.1-6.xml	223804	n-min	745	10	150
narod.1-6.xml	223805	n-min	1312	10	180
narod.1-6.xml	223805	n-min	2585	10	200

narod.1-6.xml	223805	n-min	541	5	100
narod.1-6.xml	223806	n-min	611	15	100
narod.1-6.xml	223807	min в n	2101	10	200

4. Время работы MyFim для метода “n минимальных элементов в перестановке”.

Вход	Порог	Время работы, с
b_1_20_s_100_n1-6.txt	100	1,2
b_1_20_s_100_n1-6.txt	95	2,0
b_1_20_s_100_n1-6.txt	90	3,1
b_1_20_s_100_n1-6.txt	85	5,3
b_1_20_s_100_n1-12.txt	100	3,0
b_1_20_s_100_n1-12.txt	95	9,0
b_1_20_s_100_n1-12.txt	90	14,2
b_1_20_s_100_n1-12.txt	85	25,7
b_1_20_s_100_n1-24.txt	100	16,1
b_1_20_s_100_n1-24.txt	95	120,0
b_1_20_s_100_n1-24.txt	90	590,4
b_1_20_s_100_n1-24.txt	85	1710,6
b_1_10_s_150_n1-6.txt	150	1,75
b_1_10_s_150_n1-6.txt	145	3,265
b_1_10_s_150_n1-6.txt	140	19,609
b_1_10_s_150_n1-6.txt	135	97,046
b_1_10_s_150_n1-6.txt	130	36,609
b_1_10_s_150_n1-6.txt	125	11,75

5. Время работы MyFim для метода “минимальные элементы в n перестановках”.

Вход	Порог	Время работы, с
m_1_20_s_n1-3.txt	100	3,4
m_1_20_s_n1-3.txt	95	3,9
m_1_20_s_n1-3.txt	90	7,0
m_1_20_s_n1-6.txt	100	16,4

m_1_20_s_n1-6.txt	95	4479,6
m_1_20_s_n1-6.txt	90	7439,4
m_1_20_s_n1-12.txt	100	291,36
m_1_20_s_n1-12.txt	95	40418,796

Последние строки таблицы напоминают о том, что теоретическая временная сложность (в худшем случае) работы алгоритма экспоненциальна.

6. Время работы MyFim для “лексического» метода.

Вход	**Порог**	**Время работы, с**
lex.txt	500	6150,2
lex.txt	600	3589,561
lex.txt	700	2317,25
lex.txt	800	96,531
lex.txt	900	54,953
lex.txt	1000	23,453
lex.txt	1500	9,953
lex.txt	2000	9,125

Файл вида b*.txt представляет собой последовательность записей вида <DocId1, DocId2, …, DocIdn>,где список DocId состоит из идентификаторов документов имеющих данный общий отпечаток FingerPrintId. Для файлов с именем типа b_1_20_s_100_n1-6.txt b (от Broder) означает метод “n min”, 1- отступ между соседними шинглами, 20 - длину шингла в словах, s – признак очистки от разметки, 100 – длину образа документа в отпечатках, n1-6 – номера используемых подколлекций РОМИП с 1 по 6. Аналогичный формат имеют названия файлов, полученных методом “min в n”, и имеющие префикс m (от matrix). Данные файлы соответствуют описанию <shingles.txt.out> в разделе 3.7. Файл lex.txt состоит из записей вида <DocId1, DocId2,…, DocIdn>, являющимися списками документов с данным общим словом. Этот файл соответствует входному файлу шага 2 из раздела 3, где описывается “лексичекийметод”.

5. Заключение

По результатам наших экспериментов по использованию методов порождения частых замкнутых множеств в сочетании с традиционными синтаксическими и лексическими средствами можно сделать следующие выводы.

- Методы порождения частых замкнутых множеств представляют эффективный способ определения сходства документов одновременно с порождением кластеров сходных документов.
- На результаты синтаксических методов определения дубликатов значительное влияние оказывает параметр «длина шингла». Так в наших экспериментах результаты для длины шингла, равной 10, были существенно ближе к списку дублей РОМИП чем для длины шингла, равной 20, 15 и 5.
- В экспериментах для всех значений параметров не было обнаружено существенного влияния использования метода «минимальные элементы в n перестановках» на качество результатов. По-видимому, случайности, задаваемой отбором шинглов с помощью метода «n минимальных элементов в перестановке» достаточно на практике.
- Необходимы дальнейшие эксперименты с использованием различных значений параметров синтаксических методов, и их сравнение с результатами лексических методов, использующих инвертированные индексы коллекций.

Список литературы

1. A. Broder, On the resemblance and containment of documents, in Proc. Compression and Complexity of Sequences (SEQS: Sequences'97) .
2. A. Broder, M. Charikar, A.M. Frieze, M. Mitzenmacher, Min-Wise Independent Permutations, in Proc. STOC, 1998.
3. A. Broder, Identifying and Filtering Near-Duplicate Documents, in Proc. Annual Symposium on Combinatorial Pattern Matching, 2000.
4. J. Cho, N. Shivakumar, H. Garcia-Molina, Finding replicated web collections, 1999
5. A. Chowdhury, O. Frieder, D.A.Grossman, and M.C. McCabe, Collection statistics for fast duplicate document detection, ACM Transactions on Information Systems, 20(2): 171-191, 2002
6. B. Ganter and R. Wille, Formal Concept Analysis: Mathematical Foundations, Springer, 1999.
7. G. Grahne and J. Zhu, Efficiently Using Prefix-trees in Mining Frequent Itemsets, in Proc. FIMI Workshop, 2003.

8 T. H. Haveliwala, A. Gionis, D. Klein, and P. Indyk ,
Evaluating Strategies for Similarity Search on the Web, in Proc. WWW'2002, Honolulu, 2002.
9. S. Ilyinsky, M.Kuzmin, A. Melkov, I. Segalovich, An efficient method to detect duplicates of Web documents with the use of inverted index, in Proc. 11th Int. World Wide Web Conference (WWW'2002).
10. A. Kolcz, A. Chowdhury, J. Alspector, Improved Robustness of Signature-Based Near-Replica Detection via Lexicon Randomization, in Proc. KDD'04, Seattle, 2004.
11. S.O. Kuznetsov and S.A. Obiedkov, Comparing Performance of Algorithms for Generating Concept Lattices, Journal of Experimental and Theoretical Artificial Intelligence, vol. 14 (2002), pp. 189-216.
12. N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Efficient Mining of Association Rules Using Closed Itemset Lattices, Inform. Syst., 24(1), 25-46, 1999.
13. N. Shivakumar, H. Garcia-Molina, Finding near-replicas of documents on the web, 1998
14.W. Pugh, M. Henzinger, Detecting duplicate and near-duplicate files United States Patent 6658423 (December 2, 2003).