УДК 004.912

А.В. Цимбалов[1]
О.В. Золотарев[2]

МЕТОД ШИНГЛОВ

В данной статье рассматриваются методы по определению дублирования документов с целью недопущения их включения в коллекции; анализируются подходы для поиска нечетких дубликатов на основе метода шинглов с целью определения спама в электронной почте, поиска плагиата, очистки коллекций документов от дубликатов. В данной работе представлен разбор методики реализации и выбора параметров алгоритма шинглов, выработаны критерии для выбора функции контрольных сумм (сигнатур), разработана программа для определения дубликатов, предложены критерии выбора параметров оптимизации алгоритма шинглов с применением MinHash и алгоритма супершинглов.

Ключевые слова: *шинглы, супершинглы, нечеткие дубликаты, подобие текстов, алгоритм шинглов.*

A.V. Zimbalov
O.V. Zolotarev

SHINGLE METHOD

This article discusses the methods for identifying duplicate documents to prevent their inclusion into collections; it also analyses the approaches for finding near-duplicate documents based on the method of shingles to determine spam e-mail, search of plagiarism, to clean collections' documents from duplicates. This paper also presents analysis methods and parameter selection of shingle algorithm, criteria of the checksums (signatures) selection. There was developed a program for identifying duplicates, and proposed the criteria for selecting the optimization algorithm of shingles with using MinHash and supershingles algorithm.

Keywords: *shingles, supershingles, fuzzy duplicates, similarity of texts, algorithm of shingles.*

Введение

Информационное общество продолжает свое развитие.

Ключевой технологией, помогающей удовлетворять информационные потребности человека, остаются поисковые системы. Они реализуют механизмы информационного поиска, ищут различные объекты информации, очень часто это текстовые документы.

Одной из задач по улучшению качества информационного поиска текстовой информации является решение проблемы дублирования информации.

В поисковой выдаче после отбора и сортировки по релевантности может оставаться множество похожих документов, хотя в силу того, что они равнозначны для пользователя, достаточно было бы включить в список только один из них, либо отобрать все похожие документы в отдельную группу.

Такие равнозначные для пользователя по содержанию документы называют *нечеткими дубликатами.*

Существует класс алгоритмов для определения нечетких дубликатов, который позволяет исключить их из поисковой выдачи и улучшить тем самым качество результатов информационного поиска.

Помимо качества поиска, меры подобия текстов, рассчитанные такими алгоритмами, можно использовать как критерий качества информации и в других задачах управления информацией: очистки коллекции от дубликатов, определение спама в электронной почте, определение плагиата.

[1] Магистрант АНО ВО «Российский новый университет».

[2] Кандидат технических наук, доцент, заведующий кафедрой информационных систем в экономике и управлении АНО ВО «Российский новый университет».

Задачи

В нашей работе мы создали реализацию алгоритма для контроля за уникальностью текстов коллекции. Решаемая задача состояла в том, чтобы не допустить добавления в хранилище текстовых документов со слишком схожим содержанием. Характер документов: статьи различной тематики, предназначенные для размещения на веб-страницах в рекламных целях. Чтобы обеспечить наибольшую аудиторию читателей таких статей они должны соответствовать определенным критериям качества. Одним из важных критериев качества является уникальность содержания, так как в случае если несколько статей слишком похожи, поисковая система не будет предоставлять дубликаты в поисковой выдаче, что ограничит аудиторию читателей. С целью обеспечения уникальности было решено определить характеристики схожести для каждого документа в хранилище, слишком взаимно схожие документы объединить в группы дубликатов. Все новые добавляемые в хранилище документы должны сразу же проверяться на уникальность.

В случае если уникальность документа слишком низкая, его добавление в хранилище не допускается, либо документ добавляется с пометкой группы дубликатов, при этом количество текстов для добавления в такую группу ограничено.

Один из способов определения нечетких дубликатов и сравнениях текстов – применение алгоритма шинглов. Алгоритм получил распространение и имеет множество прикладных реализаций в различных системах, и также упоминается в академических и научных работах. Однако ряд аспектов и параметров реализации в теоретических и практических работах часто спорен и плохо обоснован.

Для повышения эффективности алгоритма мы рассмотрели теоретические и практические аспекты и параметры реализации алгоритма шинглов.

Цели:

• разработка эффективного алгоритма для сравнения текстов на основе алгоритма шинглов;

• разработка эффективного алгоритма для поиска дубликатов текстов;

• разработка методики выбора и обоснования параметров алгоритма для особенностей конкретных задач.

Обзор работ по алгоритму шинглов

Алгоритм получил распространение благодаря работе [1], в связи с чем его вполне можно назвать алгоритмом Бордера, он предлагает сравнивать тексты, представляя каждый из них в виде множества *n*-грамм, – всех последовательностей слов текстов фиксированной длины *n*, которые были названы *шинглами*. Для увеличения эффективности шинглы из текстовых подстрок преобразовывались в числа – дактилограммы (fingerprints), вычисляемые по алгоритму Рабина – Карпа. Если количество схожих элементов в двух множествах, представляющих таким образом тексты, было существенно больше количества различных, то такие тексты признавались схожими.

Позже этот метод получил развитие в исследовании [2], где подробно приводятся параметры его реализации. Авторы также предлагают способы по уменьшению вычислительной сложности алгоритма. В частности, переход от сравнения полных множеств шинглов двух текстов к сравнению усеченных множеств фиксированной длины, составленных из исходных с помощью вычисления минимумов случайных инъективных (одно-однозначных) функций.

Также среди русскоязычных ученых и инженеров популярно исследование на русском языке [3], описывающее сравнительные результаты реализации различных алгоритмов поиска дубликатов. В этом исследовании алгоритм описывается кратко со ссылкой на исследование [2], предлагаются оригинальные параметры реализации. О результатах исследования Зеленков Ю.Г. и Сегалович И.В. сообщают, что алгоритм показал на тестовых данных сравнительно плохие результаты, что, возможно, связано с неудачной параметризацией.

Несмотря на многочисленные описания алгоритма в различных работах, конкретные параметры его реализации представляются нам не всегда достаточно подробно описанными и обоснованными.

Примером параметра, который часто заимствуется в работах по данному алгоритму, является «магическое» число 84. Именно 84 случайные функции предлагают авторы исследования [2] для усечения множеств хешей шинглов. Однако в статье [2] написано: “We drew the 84 functions from a smaller space, which does not seem to be a problem in practice”, что в вольном переводе звучит: «Мы использовали 84 функции для экономии места (на диске), что на практике не выглядит проблемой». То есть, данное число было выбрано исходя из конкретных ограничений аппаратной среды.

Заметны также противоречия в трактовке, какие именно действия над данными нужно осуществлять при реализации алгоритма, какие из

них и для чего необходимы и как они влияют на результат.

Мы постарались восполнить пробелы в обосновании и детализации параметров алгоритма и выбрать оптимальные параметры для нашей задачи, а также сформулировать общий подход к их выбору для подобных задач.

Принцип работы

Коэффициенты подобия

В основе работы алгоритма лежит определение скалярной меры подобия текстов.

Данная мера вычисляется отношениями количественных характеристик множеств признаков двух текстов.

При таком подходе можно выделить две меры, характеризующие подобие текста: сходства (выражение (1)) и меру вхождения (меру включения) (выражение (2)):

$$rs(A,B)=\frac{|A\cap B|}{|A\cup B|} \quad (1)$$

– выражение 1 – коэффициент сходства,
где A – множество признаков первого текста;
B – множество признаков второго текста;

$$cnt(A,B)=\frac{|A\cap B|}{|A|} \quad (2)$$

– выражение 2 – коэффициент вхождения,
где A – множество признаков первого текста;
B – множество признаков второго текста.

Мера сходства является симметричной (выражение (1) коммутативно) и независима от размеров сравниваемых множеств, т.е. длины сравниваемых текстов. Её удобство в том, что подобие документов можно записать всего одним коэффициентом, что удобнее для обработки данных и хранения. Однако в некоторых случаях этот коэффициент дает неполное представление о «качестве подобия». Например, если текст сравнивается с его же фрагментом, т.е. $A\in B$, то коэффициент подобия будет характеризоваться отношением (3). То есть, чем меньше $|A|$ относительно $|B|$, тем более различными будут выглядеть тексты по этой мере, в то время как очевидно, что качество текста, представленного множеством A для большинства задач, выглядит очень низким, то есть в данном случае как критерий качества мера использоваться не может. Для таких случаев больше применима мера вхождения, рассчитываемая отдельно для каждого из документов:

$$A\in B;\ rs(A,B)=\frac{|A|}{|B|}. \quad (3)$$

В некоторых хранилищах документов (и в нашем случае) ситуация, когда $|A|$ много меньше $|B|$, невозможна. Это связано с ограничениями на минимальную и максимальную длину текста. В нашей системе минимальная длина текста документа ограничена 2000 символов, а максимальная не превышает 6000, что делает достаточным применение симметричного коэффициента.

Несмотря на то что пара коэффициентов вхождения полноценно характеризует подобие текстов, их расчет применительно к алгоритму шинглов возможен не всегда, так как для повышения эффективности алгоритма множества сравниваемых признаков усекаются до фиксированной мощности, что делает возможным рассчитать только меру сходства.

Шинглы

В качестве множества признаков, по которым сравниваются тексты, используется набор шинглов.

Шинглы – это упорядоченные множества слов фиксированной длины (длина измеряется в словах), на которые текст «разрезается внахлест». Соответственно, шинглы сохраняют тот же порядок слов, в котором слова следуют в тексте.

В качестве примера в таблице (1) приведено множество шинглов размером 3 из строфы стихотворения С. Есенина «Берёза».

Таблица 1

Строфа из стихотворения, разбитая на шинглы размером 3

белая берёза под
берёза под моим
под моим окном
моим окном принакрылась
окном принакрылась снегом
принакрылась снегом точно
снегом точно серебром

Исследование [2] характеризует размер шингла в словах как существенный параметр алгоритма.

Замена слова «кумкват» на «хурма» в документе из N слов (при условии, что «кумкват» и «хурма» больше нигде в документе не встречаются) дает меру схожести $(N – K) / (N + K)$ между оригинальным и модифицированным документами. Это означает, что не следует выбирать K слишком большим, чтобы даже небольшие изменения не приводили к низкому показателю

подобия. С другой стороны, не следует выбирать K слишком маленьким. Как экстремальный пример можно рассмотреть случай установки K, равным единице, это приведет к сравнению лексикона документов, сделав метрику полностью нечувствительной к порядку слов.

Так как шинглы выбираются «внахлест» и каждый последующий отличается от предыдущего лишь на два слова (первое и последнее), то количество шинглов N_{sd} в документе сопоставимо с количеством его слов N_w и вычисляется по формуле (4), где S_{len} – длина шингла в словах.

$$N_{sd} = N_w - S_{len} + 1. \quad (4)$$

В качестве конкретного коэффициента подобия можно взять представление выражения (1) в виде выражения, которое называется коэффициентом Сёренсена (5), где a – количество шинглов первого текста; b – количество шинглов второго текста, c – количество совпадений шинглов:

$$K_s = \frac{2c}{a+b}. \quad (5)$$

Порядок действий

Таким образом, для базовой принципиальной реализации алгоритма шинглов без оптимизаций необходимо совершить всего три действия:

1) выделить все для двух сравниваемых текстов множества всех шинглов размера N;

2) попарно сравнить шинглы двух множеств, подсчитав общее количество и число совпадений;

3) с учетом общего количества и количества совпадений элементов рассчитать коэффициент подобия.

В данном виде алгоритм уже вполне применим для задач сравнения отдельных текстов, когда не требуется производительности.

Все прочие операции не являются необходимыми, они служат для уменьшения вычислительной сложности алгоритма.

Повышение эффективности алгоритма

Хеширование

Алгоритм Рабина – Карпа

Первой распространенной оптимизацией базового алгоритма является введение функции расчета контрольных сумм шинглов. После разбивки документа на шинглы, они представлены строками, то есть упорядоченным набором символов. Для сравнения двух строк требуется $L_1 \times L_2$ операций, где L_1, L_2 – длины соответствующих строк в символах. С возрастанием длин строк сложность возрастает как $O(L_1 L_2)$.

Для увеличения эффективности можно представить каждую строку шингла в виде сигнатуры (числа, контрольной суммы), уменьшив сложность до O(1), однако это потребует затрат машинных операций на предварительный расчет сигнатуры, сложность которых зависит от алгоритма расчета.

В исследованиях [1] и [2] для хеширования шинглов перед сравнением используется алгоритм дактилограмм по методу Рабина – Карпа, ранее предложенному в работе [5] для поиска строки в подстроке. Идея алгоритма состоит в последовательном вычислении для строки суммы кодов отдельных её символов, взятых по модулю большого простого числа. Такой принцип преобразования называют *хешированием*, а осуществляющие его функции – *хеш-функциями*, итог в виде контрольной суммы для краткости можно назвать *хешем*. Функцию, реализующую преобразования по алгоритму Рабина – Карпа, можно назвать *RK-hash*.

Сильной стороной алгоритма является время вычисления хеша O(n) [6], где n – длина строки (в нашем случае шингла), а также «закольцованности хеша», свойстве, позволяющем вычислить следующую дактилограмму на основе предыдущей. Если известен хеш одного шингла, то хеш следующего за ним, который отличается двумя словами (одно слово убрано в начале и одно добавлено в конце), можно вычислить путем вычитания из первого хеша кодов символов первого слова и добавлением символов кодов последнего. Таким образом, если для первого шингла количество операций расчета сопоставимо с длиной его строки в символах, то на каждый следующий хеш будет затрачено лишь $L_{w1} + L_{w2}$ операций, где L_{w1} – длина первого «вычитаемого» слова, L_{w2} – второго «добавляемого» слова.

Проблема коллизий

При сравнении сигнатур шинглов, рассчитанных функцией хеширования, возможны ошибки сравнения, возникающие в ситуации, когда для строк S_1 и S_2 при RK-hash(S_1) = RK-hash(S_2), тем не менее $S_1 \neq S_2$, то есть, имеет место совпадение контрольных сумм при несовпадении строк. Такая ситуация называется *коллизией хешей*. Частота коллизий зависит от общего количества уникальных шинглов, а также от диапазона значений контрольной суммы, которая, как правило, выбирается фиксированной.

Чтобы минимизировать количество коллизий, важно выбрать функцию с равномерным распределением значений контрольных сумм, а также выбрать достаточно большой размер диапазона значений хеша, исходя из предполагаемого количества исходных значений, в нашем случае – уникальных шинглов.

Выбор размера сигнатуры

В работе [2] для шинглов вычислялись дактилограммы размером 64 bit, т.е. диапазон значений хеша составлял 2^{64}. В работе [8] используются сигнатуры размером 40 bit, или 5 байт, что дает 2^{40} значений. Во многих работах, например в [12], предлагается использовать для вычисления функцию CRC32, выдающую значения 32 bit и имеющую 2^{32} значений, соответственно. Возникает вопрос, как выбрать функцию для минимизации коллизий?

Вероятность коллизий можно оценить, применив схему парадокса дней рождения [11].

Так, если предположить, что функция хеширования достаточно равномерна, то для общего числа уникальных шинглов N_s и размера хеша N_r вероятность совпадения P_c хотя бы двух хешей разных шинглов можно оценить выражением (6)

$$P_c = 1 - \frac{N_s!}{N_s^{N_r}\left(N_s - N_r\right)!}. \quad (6)$$

Видно, что чем больше шинглов, тем большая длина хеша требуется для уменьшения коллизий.

Для простоты расчетов целесообразно использовать аппроксимацию (6), полученную разложением экспоненты в ряд Тейлора. В итоге получим (7), следствием которого является (8):

$$P_c\left(N_s, N_r\right) \approx 1 - e^{-\frac{N_s^2}{2 \times N_r}} \quad (7)$$

$$N_s = \sqrt{2N_r \times ln\left(\frac{1}{1-P_c}\right)}. \quad (8)$$

Существует еще один грубый метод оценки вероятности коллизий: это правило, которое гласит, что хотя бы одна коллизия с высокой вероятностью появляется при $N_s \geq \sqrt{N_r}$ (8). Оно описывается во многих работах по криптографии, например в [11].

Расчет значения оценок для популярных размеров хеш-функций приведен в таблице (2).

Таблица 2

Оценка минимального количества шинглов, приводящих к коллизии хеш-функции

Размер значения хеш-функции	**Оценка** $N_s \geq \sqrt{N_r}$	**Оценка по формуле (8) для** P_c **= 0,01**	**Оценка по формуле (8) для** P_c **= 0,1**	**Оценка по формуле (8) для** P_c **= 0,4**
2^{32}	65536	9291,486835	30083,8817	66241,66888
2^{40}	1048576	148663,7894	481342,1072	1059866,702
2^{48}	16777216	2378620,63	7701473,715	16957867,23
2^{56}	268435456	38057930,08	123223579,4	271325875,7
2^{64}	4294967296	608926881,2	1971577271	4341214012
2^{72}	68719476736	9742830100	31545236336	69459424188

Из таблицы 2 видно, что лучшая оценка количества шинглов без коллизий для хеш-функции со значениями в пределах 32 bit составляет не более 67000. Это указывает на нецелесообразность использования любых функций со значениями такого размера для множеств более 67000 элементов. Например, использование функции CRC32 на таких объемах должно неизбежно приводить к коллизиям.

Обрабатываемые в рамках задачи данные содержали 54035 текстов, из которых было выделено 10376876 шинглов. Часть этих шинглов – дубликаты, т.е. количество уникальных шинглов составило менее 10^7. Для существующих данных, таким образом, согласно грубой оценке, вполне достаточным выглядел бы размер хеша 48 bit, однако с ростом количества добавляемых в систему текстов вероятность коллизий могла бы сильно возрасти.

Возможный рост объема данных в будущем был оценен как не более 1 порядка, т.е. он не превысит 10^8 шинглов. Таким образом, оценка показывает, что наиболее подходящим для конкретной задачи является размер хеша 56 bit.

Чтобы убедиться в правильности расчетов, был проведен эксперимент: была написана программа, выполняющая расчеты сигнатур всех 10376876 шинглов различными хеш-функциями со значениями различной длины и выполняющая подсчет коллизий. Были произведены вычисления сигнатур по алгоритму Рабина – Карпа с размером хеша 32, 48, 56 и 64 bit, а также для сравнения сигнатур CRC32 и CRC64.

Алгоритм Рабина – Карпа (т.н. RK-hash) при-

менялся в виде собственной реализации на языке высокого уровня с применением библиотеки GMP для корректных операций умножения с беззнаковыми целыми числами. Результаты представлены в таблице 3.

Таблица 3

Количество коллизий для контрольных сумм различной длины

Функция расчета сигнатуры	Количество коллизий на 10376876 значениях
RK-hash 32 bit	13089
CRC32	12320
RK-hash 48 bit	0
RK-hash 56 bit	0
RK-hash 64 bit	0
CRC64	0

Таким образом, для конкретной задачи для представления сигнатур шинглов была выбрана реализация алгоритма Рабина – Карпа с длиной контрольной суммы 56 bit.

Переход к упорядоченным множествам и устранение дублей

После преобразования наборов шинглов каждого из документов к набору хешей, задача сравнения двух текстов сводится к вычислению пересечения множеств их хешей. Для неупорядоченных множеств сложность составляет O(m*n), где m – количество шинглов первого документа, n – количество шинглов второго документа. Такая сложность всё еще выглядит неприемлемой, однако можно её уменьшить, если представить шинглы упорядоченными множествами. Для этого достаточно отсортировать их по величине контрольной суммы и избавиться от дублей. После этого задача сравнения документов сводится к сравнению упорядоченных множеств, сложность которой составляет уже O(m + n).

Изменение порядка шинглов никак не влияет на меру схожести. Избавление от дублей изменит коэффициент, в случае если в двух образцах будет разное количество повторов.

Например, сравнивая тексты “A B C” и “A C C” с дублями с размером шингла 1, мы получим по формуле (5) $a = 3$; $b = 3$; $c = 2$ и коэффициент Сёренсена 4/6, а в случае с игнорированием дублей: $a = 3$; $b = 2$; $c = 2$ и коэффициент 4/5. Однако интуитивно понятно, что на практике повторы шинглов – явление редкое, и на коэффициент схожести будет влиять незначительно ввиду большого количества неповторяющихся шинглов.

Уменьшение количества операций за счет ограничения набора признаков

При сравнении большого количества текстов с учетом вышеописанных оптимизаций сложность сравнения может быть слишком велика для организации вычислений за приемлемое время.

В таком случае одним из решений будет уменьшение количества операций за счет сравнения не всех, а только части из набора признаков документа, т.е. произвести усечения множеств сигнатур шинглов каждого документа, выбрав из них несколько, ограничив множество признаков фиксированной длиной.

Такой метод подробно описан в исследовании [4] и используется во многих реализациях алгоритма, например в работах [2] и [3], в настоящее время метод часто называют MinHash.

Идея преобразования состоит в том, чтобы задать множество случайных инъективных функций размером N_{rf} и получить их значения для всех сигнатур каждого документа. После этого для сравнения отбираются только те сигнатуры, для которых одна из функций возвращает минимальное значение. Таким образом, для сравнения из множества контрольных сумм каждого документа выбирается только N_{rf} контрольных сумм. Путь для документов T_1 и T_2 выбрали две сигнатуры, которые дали минимумы случайной функции Fj, соответственно $m(T_1, F_j)$ и $m(T_2, F_j)$. Утверждается, что такая пара сигнатур имеет вероятность совпадения, равную мере подобия документов, как показано в равенстве (9). Получив таким способом достаточно большое количество контрольных сумм и посчитав количество их совпадений, можно с определенной точностью судить о совпадении исходных множеств. Доказательство и более подробные требования к членам равенства содержатся в работе [10]:

$$P_p\left(m\left(T_1, F_j\right) = m\left(T_2, F_j\right)\right) = \frac{\left|T_1 \cap T_2\right|}{\left|T_1 \cup T_1\right|}. \qquad (9)$$

Таким образом, посчитав количество совпадений отобранных сигнатур, можно судить о подобии исходных множеств. Чем больше взято функций и чем больше отобрано сигнатур, тем более точной получится эта оценка. Максимальная ожидаемая вероятность ошибки сравнения множеств в случае, когда отобранные значения не характеризуют исходные множества, согласно [10] составит $O\left(\frac{1}{\sqrt{N_{rf}}}\right)$.

Переход от шинглов к супершинглам. Выбор параметров

В исследовании [2] $N_{rf} = 84$, то есть использу-

ется семейство из 84 случайным образом определенных и после зафиксированных функций для выбора соответствующего количества сигнатур каждого документа. Чтобы увеличить эффективность сравнения, сигнатуры объединяются в 6 групп по 14 в каждой. Такие группы получили название «супершинглы».

Исходя из (9), можно рассчитать зависимость вероятности совпадения хотя бы одного супершингла от подобия исходных множеств. Для общего случая, когда выбрано M_m групп по N_m шинглов, вероятность совпадения хотя бы одного супершингла может быть описана по формуле (10):

$$P_{1sh} = 1 - \left(1 - P_p^{N_m}\right)^{M_m}. \quad (10)$$

Таким образом, для 84 сигнатур при $M_m = 6$ и $N_m = 14$ для подобия двух документов, составляющего 0,9, вероятность совпадения хотя бы одного супершингла составит примерно 0,98. Сравнивая только множества супершинглов, можно найти все документы с подобием, близким 0,9.

Однако для подобия 0,7 вероятность составляет всего лишь 0,04, а для подобия 0,5 – только 0,0003. Это означает, что с высокой вероятностью для подобных документов с коэффициентом 0,7 и ниже найти их сравнением групп шинглов не удастся.

Кратко эта особенность рассматривается в работе [3], где предложены другие параметры алгоритма (36 сигнатур, $M_m = 6$, $N_m = 6$), однако упоминается, что они показали не очень хорошие характеристики качества. В большом количестве работ используются параметры из исследования [2] без изменений. Согласно работе [10], вероятность ошибки составляет $\frac{1}{\sqrt{N_{rf}}}$, где N_{rf} – количество отобранных MinHash сигнатур. Исходя из такой оценки, можно предположить, что авторы [2] выбрали 84 как наименьшее число функций, дающее вероятность ошибки примерно 0,1.

Из вышеописанных особенностей алгоритма напрашивается вывод, что при выборе размера (N_m) и количества (M_m) супершинглов нужно в первую очередь ориентироваться на минимально требуемый для поиска коэффициент подобия документов.

Существуют еще два значимых ограничения для выбора параметров: это количество операций сравнения и вероятность ошибки сравнения в связи с усечением множества.

Количество операций сравнения можно оценить как количество размещений M_m по 2.

С увеличением количества супершинглов количество сравнений возрастает, что можно считать критерием, ограничивающим количество и размер супершинглов сверху.

Вероятность ошибки с уменьшением произведения N_m *и* M_m увеличивается, согласно [10] и (10), что ограничивает возможность уменьшить набор функций MinHash.

Нашим ориентиром является совпадение хотя бы одной группы шинглов при подобии документа 0,7, для которых параметры, предложенные в исследованиях, не подходят. При беглом анализе более эффективными могли бы стать параметры $N_m = 6$, $M_m = 14$, то есть 14 групп по 6 шинглов вместо 6 групп по 14. При таких параметрах вероятность совпадения супершингла при подобии 0,7 составляет примерно 0,83.

На данный момент расчеты и реализация оптимизации MinHash не завершена. В дальнейшем планируется выбрать параметры алгоритма и экспериментально проверить их эффективность.

Выводы

В ходе проделанной работы нам удалось:

- детально разобрать методику реализации и выбора параметров алгоритма шинглов;
- выработать критерии для выбора функции контрольных сумм (сигнатур);
- разработать программу и, обработав данные, подтвердить критерии экспериментально;
- наметить критерии выбора параметров оптимизации алгоритма шинглов с применением MinHash и алгоритма супершинглов.

В будущем мы рассчитываем реализовать более эффективный и гибкий алгоритм и полностью описать методику выбора его параметров в зависимости от области применения.

Литература

1. Broder, A. Some applications of Rabin's fingerprinting method // R. Capocelli, A. De Santis, and U. Vaccaro, editors, Sequences II: Methods in Communications, Security, and Computer Science. – Springer-Verlag, 1993.

2. Fetterly, Dennis, Manasse, Mark, Najork, Marc, and Wiener, Janet. A Large-Scale Study of the Evolution of Web Pages. Software // Practice & Experience, Wiley, 2004. – February.

3. Зеленков Ю.Г., Сегалович И.В. Сравнительный анализ методов определения нечетких дубликатов для WEB-документов // Труды 9-й Всероссийской научной конференции RCDL'2007. – Переславль-Залесский, 2007.

4. Broder, A. On the resemblance and containment of documents // SEQS: Sequences '91, 1998.

5. Rabin, M. Fingerprinting by random polynomials. Report TR-15-81, Center for Research in Computing Technology. – Harvard University, 1981.

6. Фофанов О.Б. Алгоритмы и структуры данных. – Томск : Национальный исследовательский Томский политехнический университет, 2014.

7. https://ru.wikipedia.org/wiki/SSE4

8. Broder, A., Glassman, S., Manasse, M., Zweig, G. Syntactic Clustering of the Web // Comput. Netw. ISDN Syst. – 1997. – Vol. 29. – Pp. 1157–1166.

9. Цимбалов А. Код реализации алгорима шинглов. – https://github.com/luckybeggar/text_search_cmp

10. Chum, Ondrej, Philbin, James, Zisserman, Andrew. Near Duplicate Image Detection: MinHash and tf-idf Weighting. – British Mashine Vision Conference, 2008.

11. Mihir Bellare, Tadayoshi Kohno. Hash Function Balance and its Impact on Birthday Attacks //EUROCRYPT '04, Lecture Notes in Computer Science. – Vol. 3027 / C. Cachin and J. Camenisch eds. – Springer-Verlag, 2004.

12. Серов С.С., Андреев А.Е., Кравченя П.Д., Гущин Р.И., Чеботарев П.П. Сокращение времени оценки схожести текстовых документов на неоднородной многопроцессорной вычислительной системе // Инженерный вестник Дона. Волгоградский государственный технический университет. – 2015. – № 2. – Ч. 2.