САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

КАФЕДРА КОМПЬЮТЕРНОГО МОДЕЛИРОВАНИЯ

И МНОГОПРОЦЕССОРНЫХ СИСТЕМ

Фаустов Богдан Андреевич

Выпускная квалификационная работа бакалавра

Поиск похожих изображений

Направление 010300

Фундаментальная информатика и информационные технологии

Научный руководитель,

кандидат техн. наук,

доцент

Гришкин В. М.

Санкт-Петербург

2016

Содержание

Введение

Различные алгоритмы поиска похожих изображений широко применяются в средствах массовой информации, поисковых системах, социальных сетях, букинговых системах и т. п. С помощью поиска похожих изображений эффективно решаются задачи классификации изображений, удаления дубликатов, отслеживания нарушений авторских прав.

Для сравнения изображений различных размеров можно использовать приведение размеров к некоторому эталонному разрешению, например, 16×16 пикселей. С помощью сравнения получаемых эталонов можно быстро найти класс вероятно похожих изображений, среди которых можно провести более детальный анализ, например, сравнить их по эталонам других размеров.

Сравнение эталонов можно проводить с помощью простых процедур, таких как вычисление нормы разности матриц, состоящих из закодированных числами цветов пикселей изображений. Однако подобные подходы требуют значительных затрат времени и по этой причине не приспособлены для решения задач классификации изображений.

Эффективным подходом к сравнению эталонных изображений является использование представления изображений в виде неупорядоченных множеств некоторых заранее выделенных элементов. Например, можно разбить все возможные цвета на несколько групп, и построить по заданному изображению множество всех групп цветов его пикселей. Для оценки близости множественных представлений изображений применяются различные коэффициенты сходства множеств, например, коэффициент Жаккара. Данный подход можно использовать также для сравнения похожести других объектов, допускающих представление в виде множеств, таких как текстовые документы, видеозаписи, популяции организмов, химические вещества и т. п.

Вычисление коэффициентов сходства изображений напрямую является достаточно трудоемкой задачей. Для быстрого приближенного вычисления коэффициента Жаккара можно использовать метод MinHash. При применении

этого метода множество, соответствующее изображению, описывается с помощью вектора, каждая компонента которого представляет собой минимальное значение хэш-функции среди ее значений для всех элементов множества. Каждой компоненте вектора соответствует своя хэш-функция. Векторы, получаемые с помощью метода MinHash, как правило, имеют большую размерность.

Locality-sensitive hashing (LSH, локально-чувствительное хэширование) – вероятностный метод понижения размерности многомерных данных. Этот метод позволяет строить структуру для быстрого приближенного (вероятностного) поиска n-мерных векторов, «похожих» на искомый шаблон. Он часто используется, когда имеются чрезвычайно большие объемы данных, которые должны быть сравнены. LSH имеет большое количество применений. Он используется в таких областях, как распознавание образов, вычислительная геометрия и сжатие данных, в классификации и кластеризации, в поисках плагиата, в работе с большими системами документооборота, в электронных библиотеках и т. д.

В данной работе обосновывается использование метода MinHash и LSH для сигнатур, получаемых с помощью этого метода. Рассматриваются общие классы хэш-функций, которые можно использовать для понижения размерности данных. Изучается реализация LSH для Google VisualRank. Реализованы в виде программы в среде Wolfram Mathematica 10.4.1 некоторые методы определения похожих изображений и проведено сравнение времени их работы.

Постановка задачи

В данной работе были поставлены следующие задачи.

1. Обосновать использование метода MinHash.
2. Обосновать использование LSH для сигнатур метода MinHash.
3. Рассмотреть общие классы локально-чувствительных хэш-функций, которые можно использовать для понижения размерности данных.
4. Изучить и обосновать реализацию LSH для Google VisualRank.
5. Реализовать в виде программы в среде Wolfram Mathematica 10.4.1 методы определения похожих изображений, изучить их варианты для различных параметров и сравнить время их работы.
   5.1. Методы, основанные на сравнении представлений изображений в виде упорядоченных векторов, состоящих из закодированных числами цветов пикселей.
   5.2. Методы, основанные на сравнении представлений изображений в виде множеств с использованием коэффициента Жаккара.
   5.3. Методы, основанные на сравнении представлений изображений в виде множеств с использованием MinHash.
   5.4. Методы, использующие LSH сигнатур метода MinHash.

Обзор литературы

Первые понятия, относящиеся к LSH, были предложены Уди Манбером в 1994 году в работе, посвященной поиску похожих файлов в большой файловой системе [1]. Он предложил идею шинглов – выделенных из текста для проверки схожести последовательностей идущих друг за другом слов. Наряду с этим были использованы также контрольные суммы. Сами понятия шинглов и «похожести» двух документов были сформулированы позднее в 1997 году Андреем Бродером в работе о сходстве и локализации документов [2]. Под k-шинглами для документа понимается последовательность, состоящая из k подряд идущих токенов, которыми могут быть символы или слова, встречающиеся в документе. Документы, которые состоят из большого числа одинаковых шинглов, считаются похожими, даже если текст встречается в другом порядке. Затем Аристид Гионис, Петр Индик и Раджив Мотвани в своих работах 1997–98 годов [3, 4] предложили поиск сходства посредством использования хэширования, т. е. преобразования по определенному алгоритму входного массива данных произвольной длины в выходную битовую строку фиксированной длины. Они предложили применение алгоритма LSH к задаче приближенного поиска ближайшего соседа. Позднее А. Андони и П. Индик показали, что приближенный поиск можно считать достаточно точным в задачах, где ближайший сосед находится проверкой всех приближенных соседей [5]. Также в 1998 году A. Broder, M. Charikar, A. Frieze и M. Mitzenmacher представили метод MinHash, использующий взаимно однозначные независимые (min-wise independent) функции [6]. Эти концепции были разработаны в результате работы авторов над алгоритмом поисковой системы AltaVista, который был использован для обнаружения дубликатов веб-страниц и устранения их из результатов поиска.

LSH может использоваться для идентификации и поиска видео. В этом случае векторы признаков строятся из видеокадров с использованием определенных цветовых гистограмм. В работе [7] авторы рассматривают недостатки

применения LSH для этой цели. В 2005 году S. Hu в своей работе [8] предложил новую схему для идентификации видео с применением LSH. В 2012–2013 годах появились статьи, в которых авторы показывают, что LSH может использоваться в поиске изображения [9, 10].

Глава 1. Алгоритмы поиска похожих изображений

Основная цель данной главы – описать и обосновать методы MinHash и LSH. Здесь приводятся основные определения и известные результаты, на которых основан данный алгоритм и которые используются в дальнейшем. Основу изложенного материала составляют лекции Anil Maheshwari [11].

1.1. Сигнатуры множеств с сохранением сходства

Пусть S и T – два непустых одновременно множества. Определим *коэффициент Жаккара* для множеств S и T как отношение числа элементов, входящих в пересечение множеств S и T к числу элементов, входящих в объединение этих множеств, т. е. по формуле

$$\frac{|S \cap T|}{|S \cup T|}.$$

Рассмотрим естественное представление множества. Пусть U – универсальное множество, элементы которого произвольно упорядочены. Пусть множество $S \subset U$. Множество S может быть представлено вектором длины $|U|$ из нулей и единиц, в котором 1 означает, что соответствующий элемент из универсального множества присутствует в S, и 0 означает отсутствие этого элемента в S. С конечным числом множеств из U можно связать *характеристическую матрицу*, где каждый столбец представляет собой вектор, соответствующий определенному множеству и каждая строка соответствует элементу U.

	S_1	S_2	S_3	S_4
Круиз	1	0	0	1
Лыжи	0	0	1	0
Курорты	0	1	0	1
Сафари	1	0	1	1
Остаться дома	0	0	1	0

Таблица 1.1. Характеристическая матрица

Например, рассмотрим таблицу 1.1. Здесь четыре множества S_i (представляющие семьи) и универсальное множество U, состоящее из пяти элементов (варианты отдыха). Из характеристической матрицы следует, что множество S_1 предпочитает круиз и сафари, а S_2 любит посещать только курорты.

Одним из эффективных способов нахождения сигнатуры для набора множеств является применение *метода MinHash*: сначала строки характеристической матрицы случайным образом переставляются, затем для каждого множества S_i (столбец в характеристической матрице), можно найти MinHash-значение h как номер (начиная с 0) первой строки, в которой есть 1 (если в столбце нет единиц, в качестве значения h можно взять $|U|$) .

Используя предыдущий пример, предположим, что результат перестановки строк представлен в таблице 1.2.

	S_1	S_2	S_3	S_4
Лыжи	0	0	1	0
Сафари	1	0	1	1
Остаться дома	0	0	1	0
Курорты	0	1	0	1
Круиз	1	0	0	1

Таблица 1.2. Характеристическая матрица после перестановки строк

Тогда значения MinHash для соответствующих множеств

$$h(S_1) = 1,\ h(S_2) = 3,\ h(S_3) = 0 \text{ и } h(S_4) = 1.$$

Следующая лемма показывает, что значение коэффициента Жаккара двух множеств совпадает с вероятностью того, что после случайной перестановки строк характеристической матрицы их значения MinHash окажутся равными.

***Лемма* 1.1.** Для любых двух множеств S_i и S_j вероятность того, что $h(S_i) = h(S_j)$ равна коэффициенту Жаккара этих множеств.

Доказательство. Представим множества S_i и S_j в виде столбцов перед случайной перестановкой. Тогда строки можно разделить на три типа:

(а) строки имеют в обоих столбцах 0;

(b) строки имеют в обоих столбцах 1;

(c) строка имеет в одном столбце 1, в другом столбце – 0.

Пусть X – число строк типа (b), Y – число строк типа (c). Заметим, что коэффициент Жаккара множеств S_i и S_j равен $\frac{X}{X+Y}$, так как X – размер пересечения $S_i \cap S_j$, $X + Y$ – размер объединения $S_i \cup S_j$.

Теперь рассмотрим вероятность того, что $h(S_i) = h(S_j)$. Пусть строки переставляются в случайном порядке. Затем будем просматривать строки сверху вниз. Тогда вероятность встретить тип (b) перед типом (c) равна $\frac{X}{X+Y}$. Но если первая строка сверху, кроме (a) строк является типом (b), то, очевидно, что $h(S_i) = h(S_j)$. С другой стороны, если первая строка будет типа (c), то множество с 1 получает эту строку в качестве MinHash, а множество с 0 в этой строке, имеет 1 в другой строке, идущей дальше. Следовательно, $h(S_i) \neq h(S_j)$, если первая строка будет типа (c).

Таким образом, показано, что вероятность того, что $h(S_i) = h(S_j)$ равна $\frac{X}{X+Y}$, что совпадает с коэффициентом Жаккара для множеств S_i и S_j. Лемма доказана.

Рассмотрим MinHash-сигнатуры, которые имеют меньшие размеры матрицы, полученные путем неоднократного применения метода MinHash к характеристической матрице. Пусть M обозначает характеристическую матрицу для множества системы S универсального множества U. Выберем набор n случайных перестановок строк характеристической матрицы.

Для каждого множества в S можно вычислить его h-значение по отношению к каждому набору из n перестановок. Это приводит к *сигнатурной матрице* – матрице, состоящей из $|S|$ столбцов и n строк, где (i, j)-му элементу соответствует сигнатура j-го множества относительно i-й хэш-функции. Обычно сигнатурная матрица должна быть значительно меньшего размера, чем M.

Приведем пример минимальной хэш-сигнатуры, полученной из таблицы

1.1 с помощью $n = 2$ перестановок (задающихся функциями $h_1(r) = r + 1 \bmod 5$, и $h_2(r) = 3r + 1 \bmod 5$, где r – номер строки таблицы 1.1, начиная с 0).

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Таблица 1.3. Сигнатурная матрица для четырех множеств

Вычисление сигнатурной матрицы должно быть сделано в электронном виде: используя описанный метод, необходимо провести n случайных перестановок характеристической матрицы M достаточно большого размера, и это является трудоемкой задачей. Можно использовать n случайных хэш-функций, переставляющих строки из M. Будем считать, что отображение из $|U|$ элементов на себя не вызовет много совпадений, однако некоторые совпадения могут быть.

Таким образом, $h(r)$ для строки с номером r (начиная с 0) из матрицы M обозначает новый номер строки после перестановки. Ниже приведено краткое описание алгоритма для вычисления сигнатурной матрицы.

Шаг 1. Выбрать n случайных хэш-функций $h_1, \dots, h_n$.

Шаг 2. Создать массив H из n строк и $|S|$ столбцов для хранения сигнатурной матрицы и заполнить его ∞.

Шаг 3. Выполнить следующие шаги для каждой строки с номером r из матрицы M.

1. Вычислить значения $h_1(r), \dots, h_n(r)$.
2. Для каждого $c = 1, \dots, |S|$, если $M[r, c] = 1$, то присвоить $H[i, c] = \min\big(H[i, c], h_i(r)\big)$ для $i = 1, \dots, n$.

Заметим, что для каждого ненулевого элемента из M, проводится $O(n)$ вычислений, и, кроме памяти для сохранения n хэш-функций, никакая дополнительная память не требуется.

Несмотря на то, что сигнатурная матрица имеет меньший размер по сравнению с характеристической матрицей, ее размер по-прежнему может быть достаточно большим. К тому же, если документы сравниваются попарно, то это займет большое количество времени.

Для быстрого попарного сравнения можно применить LSH-метод хэширования, увеличивающий вероятность коллизии объектов, которые имеют шансы быть схожими.

1.2. LSH для MinHash-сигнатур

LSH – вероятностный метод понижения размерности многомерных данных. Основная идея состоит в таком подборе хэш-функций для некоторых измерений, чтобы похожие объекты с высокой степенью вероятности попадали в одну корзину. Документы должны хэшироваться многократно для увеличения вероятности совпадений или различий. Ключевым моментом является то, что только документы, которые попадают в одну и ту же корзину, рассматриваются как потенциально похожие. Если два документа не отображаются в одну и ту же корзину при любом хэшировании, то они больше не рассматриваются по этой методике. Очевидно, нужно минимизировать количество разнородных документов, которые могут попасть в одну корзину. В то же время подобные документы по крайней мере один раз должны хэшироваться в одну и ту же корзину.

Пусть найдена MinHash-сигнатурная матрица для множества документов, как показано в предыдущем параграфе. Разделим строки этой матрицы на $b = \frac{n}{r}$ групп, где каждая группа имеет r последовательных строк (таблица 1.4). Будем считать, что n делится на r.

Группа 1	⋯	1	0	0	0	2	⋯
	⋯	3	2	1	2	2	⋯
	⋯	0	1	3	1	1	⋯
Группа 2	...			...			...
Группа 3	...			...			...

Группа 4	...	...	...

Таблица 1.4. Сигнатурная матрица, разделенная на четыре группы из трех строк каждая

Для каждой группы определим хэш-функцию h: $\mathbb{R}^r \to Z$, которая вектор-столбец длины r отображает в целое число (т. е. корзину). Можно выбрать одну и ту же хэш-функцию для всех групп, но корзины должны быть различны для каждой группы. Теперь, если два вектора длины r в некоторой группе хэш-функцией отображаются в одну и ту же корзину, то соответствующие документы потенциально похожи. Покажем это.

***Лемма* 2.1.** Пусть s – коэффициент Жаккара для двух документов. Вероятность того, что MinHash-сигнатурная матрица совпадает во всех строках по крайней мере одной из групп для этих двух документов (того, что они будут похожи после хэширования и разбиения на группы) равна $1 - (1 - s^r)^b$.

Доказательство. Из леммы 1.1 следует, что вероятность того, что MinHash для этих двух документов совпадают в какой-либо конкретной строке сигнатурной матрицы, равна s. Таким образом, вероятность того, что сигнатуры совпадают во всех строках одной конкретной группы, есть s^r. Вероятность того, что сигнатуры не совпадут, по крайней мере, одной из строк такой группы, равна $1 - s^r$.

Теперь вычислим вероятность того, что в каждой группе сигнатуры нет совпадений, по крайней мере, в одной из строк. Так как имеется b групп, эта вероятность равна $(1 - s^r)^b$. Следовательно, вероятность того, что сигнатуры совпадут, по крайней мере, в одной из групп есть $1 - (1 - s^r)^b$. Лемма доказана.

Можно построить график функции $1 - (1 - s^r)^b$, откладывая по оси абсцисс значения s, а по оси ординат – значения функции с фиксированными значениями b и r (рисунок 2). График этой функции имеет форму S-образной кривой. На самом деле, кривая будет иметь такую форму независимо от значений b и r. Из графика видно, что если коэффициент Жаккара s приближается к 1, то значение этой функции – вероятность попадания входных элементов в одну

и ту же корзину – также приближается к 1. Несмотря на это одним из важных аспектов этой кривой есть то, что самый крутой наклон происходит для значения s, приблизительно равного $t = b^{-\frac{1}{r}}$. Другими словами, если коэффициент Жаккара для двух документов превышает порог t, то вероятность быть похожими с помощью LSH очень высока.

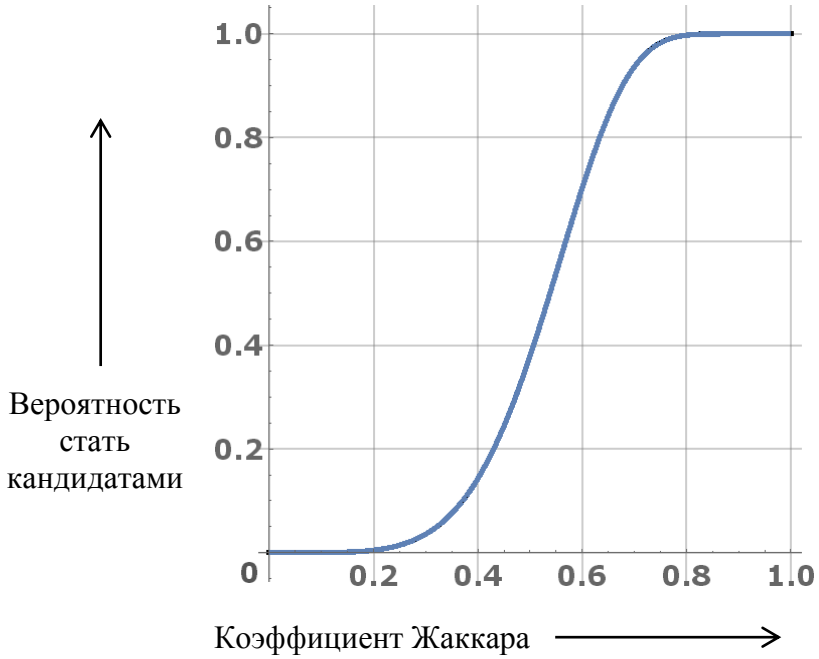

Рисунок 1.1. *S*-кривая

Таким образом, использование LSH позволяет сравнить документы (или их MinHash-сигнатуры) и выяснить, являются ли они на самом деле похожими.

1.3. Локально-чувствительные функции

В этом параграфе рассматривается семейство LS-функций (локально-чувствительных функций), которое обозначается F. Например, MinHash-функции образуют семейство таких функций.

Напомним, что мера расстояния (метрика) является неотрицательной симметричной функцией, удовлетворяющей неравенству треугольника. Этим свойствам удовлетворяют, например, евклидово расстояние между точками в пространстве, расстояние Жаккара и расстояние Хэмминга.

***Определение* 3.1.** Пусть d – мера расстояния и пусть для двух расстояний d_1 и d_2 этой меры выполняется $d_1 < d_2$. Семейство функций F называется

(d_1, d_2, p_1, p_2)-*чувствительным*, если для любой функции $f \in F$ выполняются следующие два условия:

1) если $d(x, y) \leq d_1$, то вероятность $P[f(x) = f(y)] \geq p_1$;

2) если $d(x, y) \geq d_2$, то вероятность $P[f(x) = f(y)] \leq p_2$.

Если два элемента находятся близко друг к другу по отношению к функции расстояния, то вероятность того, что они хэшируются в одну и ту же корзину, будет высокой. И наоборот, если эти два элемента являются далекими друг от друга, то вероятность того, что они хэшируются в одну и ту же корзину, будет низкой.

В качестве примера рассмотрим расстояние Жаккара для двух множеств, которое определяется как 1 минус коэффициент Жаккара этих множеств. Пусть $0 \leq d_1 < d_2 \leq 1$. Заметим, что семейство MinHash-сигнатур является $(d_1, d_2, 1 - d_1, 1 - d_2)$-чувствительным.

Предположим, что коэффициент Жаккара для двух документов не меньше, чем s. Тогда их расстояние не может быть большим чем $d_1 = 1 - s$. По лемме 1.1 вероятность того, что MinHash-сигнатуры будут равны, не может быть меньше $s = 1 - d_1$.

Предположим, что мера расстояния между двумя документами больше или равна d_2. Отсюда их коэффициент Жаккара самое большее равен $s = 1 - d_2$, и вероятность того, что MinHash-сигнатуры совпадут, не может быть больше чем $s = 1 - d_2$.

Рассмотрим (d_1, d_2, p_1, p_2)-чувствительное семейство функций F. Можно построить новое семейство G следующим образом. Каждая функция $g \in G$ формируется из множества r независимо выбранных функций из F, например $f_1, f_2, \ldots, f_r$ для некоторого фиксированного значения r, и $g(x) = g(y)$ тогда и только тогда, когда для всех $i = 1, \ldots, r$ выполняется $f_i(x) = f_i(y)$. Это семейство называют *AND-семейством*.

***Предложение* 3.1.** AND-семейство является (d_1, d_2, p_1^r, p_2^r)-чувствительным семейством функций.

Доказательство. Степень r в формулировке предложения появляется потому, что вычисляется вероятность одновременного осуществления r независимых событий.

Можно также построить G как *OR-семейство*. Тогда каждая функция g из G строится следующим образом. Из F независимо друг от друга выбираются b функций $f_1, f_2, \dots, f_b$. И $g(x) = g(y)$ тогда и только тогда, когда $f_i(x) = f_i(y)$ хотя бы для одной из функций из $\{f_1, f_2, \dots, f_b\}$.

***Предложение* 3.2.** OR-семейство является $(d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$-чувствительным семейством функций.

Доказательство. Формулы преобразования вероятностей в формулировке предложения появляются потому, что вычисляется вероятность осуществления по крайней мере одного из b независимых событий. Эта вероятность вычисляется через вероятность того, что ни одно из b событий не происходит.

Рассмотрим некоторые значения b и r для того, чтобы увидеть результат таких семейств, которые показаны выше. Можно так же объединить функции этих семейств через композицию.

Сначала построим *AND*-семейство для определенного значения r, потом – *OR*-семейство для определенного значения b, затем – композицию.

Рассмотрим таблицу 1.5 вероятностей для различных значений p.

p	p^5 (F_1)	$1-(1-p)^5$ (F_2)	$1-(1-p^5)^5$ (F_3)	$(1-(1-p)^5)^5$ (F_4)
0,2	0,00032	0,67232	0,00159	0,13703
0,4	0,01024	0,92224	0,05016	0,66714
0,6	0,07776	0,98976	0,33285	0,94983
0,8	0,32768	0,99968	0,86263	0,99840
0,9	0,59049	0,99999	0,98848	0,99995

Таблица 1.5. Иллюстрация четырех семейств, полученных при различных значениях p

Здесь F_1 – *AND*-семейство для $r = 5$, F_2 – *OR*-семейство для $b = 5$, F_3 – *AND-OR* семейство для $r = 5$ и $b = 5$, F_4 – *OR-AND* семейство для $r = 5$ и

$b = 5$.

Прокомментируем данные таблицы 1.5. Рассмотрим столбцы. Пусть F является $(0{,}2; 0{,}6; 0{,}8; 0{,}4)$-чувствительной MinHash-функцией семейства. Это означает, что если расстояние между двумя документами x и y меньше или равно $0{,}2$, то с вероятностью больше или равной $0{,}8$, они будут хэшироваться в одну и ту же корзину. Точно так же, если расстояние между x и y больше или равно $0{,}6$, то маловероятно, что они будут хэшироваться в одну и ту же корзину (вероятность хэширования в ту же самую корзину меньше или равна $0{,}4$).

Теперь рассмотрим две строки, соответствующие вероятностям $p_2 = 0{,}4$ и $p_4 = 0{,}8$. Заметим, что для *AND*-семейства p^5 существенно меньше, чем p, но p_2^5 близко к 0, а p_4^5 находится далеко от 0. Для *OR*-семейства $1 - (1 - p)^5 \geq p$, но все же значение, соответствующее p_4 близко к 1, а значение, соответствующее p_2, далеко от 1. Обратим внимание на то, что *AND-OR* семейство (F_3) соответствует семейству LSH-функций для MinHash-сигнатуры, рассмотренного в параграфе 1.2. В этом случае значение, соответствующее p_4 близко к 1, а значение, соответствующее p_2, ближе к 0. Отметим, что значения функции $1 - (1 - p^5)^5$ образуют S-образную кривую с неподвижной точкой $p \approx 0{,}7549$, которую можно найти из уравнения $p = 1 - (1 - p^5)^5$. Отсюда следует, что для значений p, значительно меньших $0{,}75$, *AND-OR* семейство близко к 0; аналогично, для значений p, больших чем $0{,}75$, – близко к 1.

Отметим, что LS-семейства можно построить не только для меры расстояния Жаккара, но и для евклидова расстояния, расстояния Хэмминга и других метрик.

1.4. Использование LSH в алгоритме Google VisualRank

С помощью локально-чувствительного хэширования можно выполнять поиск похожих изображений. Часто изображения обрабатываются с помощью глобальных дескрипторов, таких как цветные гистограммы, для получения

векторов, которые затем сравниваются с использованием различных мер расстояния. Одним из очевидных приложений сходства изображений является веб-поиск. Однако большинство поисковых систем не используют сходство изображений в поиске; вместо этого они полагаются на информацию, собранную из окружающего изображение текста на веб-странице вместе с метаданными изображения. В этом параграфе покажем, как Google в алгоритме VisualRank использует для поиска похожих изображений метод LSH вместе с несколькими локально-чувствительными семействами функций, которые были предложены в качестве меры сходства изображений.

В алгоритме VisualRank наряду с LSH используются методы компьютерного зрения. Рассмотрим поиск изображения с помощью текстового запроса. Существующий метод поиска получает первоначальные изображения-кандидаты, которые наряду с другими изображениями группируются согласно их сходству. Далее в каждой группе производится измерение «важности» изображений, что приводит к наиболее подходящим запросу результатам. При этом требуется достигнуть соглашения между пользователями интернета о том, что считать похожими изображениями. Мера сходства изображений имеет решающее значение для VisualRank, так как она определяет структуру используемого графа.

Изображения считают схожими, если их локальные признаки хэшируются в одну и ту же корзину хэш-таблицы. В качестве примера рассмотрим рисунок 1.2. В этом примере синие и зеленые круги – корзины, в которые хэшируются признаки; B подобно (схоже с) C, и C подобно D.

Вначале система VisualRank извлекает из изображений векторы локальных дескрипторов (признаков) с использованием масштабно-инвариантной функции преобразования (SIFT). Вместо цветовых гистограмм, о которых говорилось выше, используются локальные дескрипторы, так как они позволяют рассматривать сходство изображений с потенциалом вращения, масштаба и преобразований. Затем к векторам признаков применяется LSH с использованием p-устойчивого распределения.

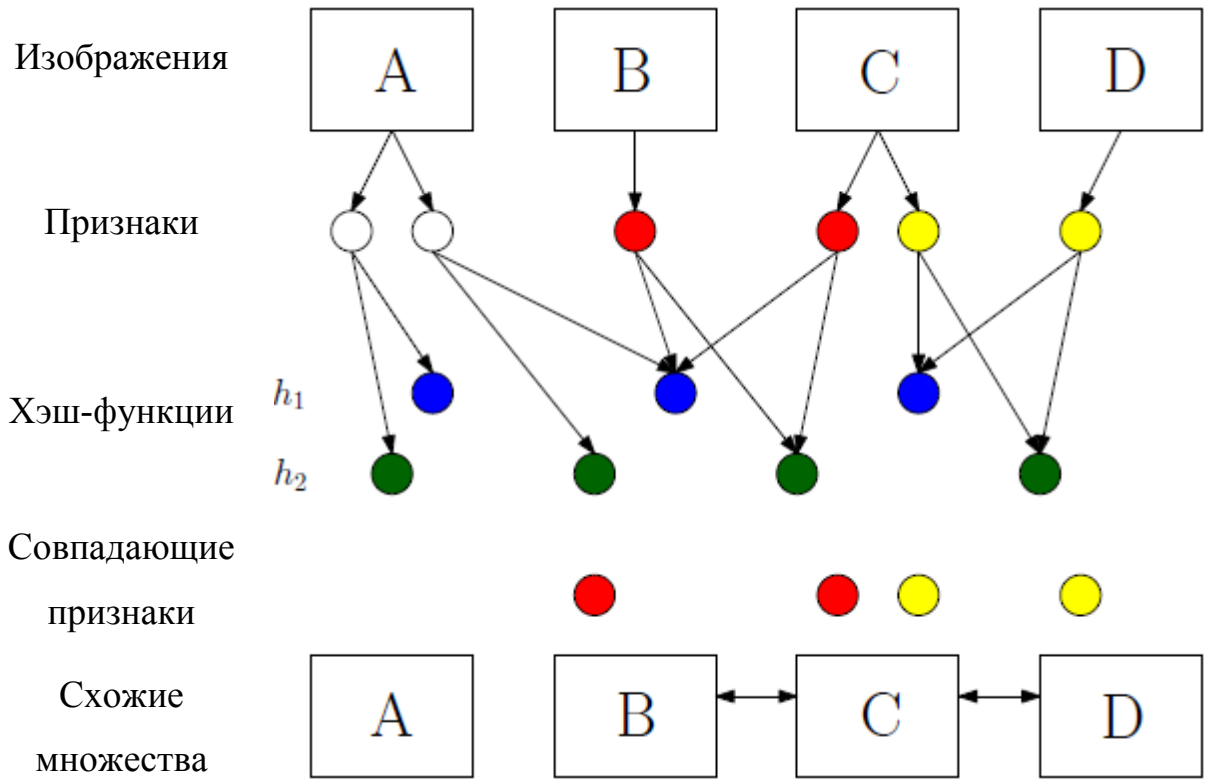

Рисунок 1.2. Пример схемы хэширования признаков изображений

***Определение* 4.1.** Распределение D называется p-*устойчивым* при $p > 0$, если для любых n вещественных чисел $v_1, v_2, \dots, v_n$ и независимых одинаково распределенных случайных величин $X_1, X_2, \dots, X_n$ с распределением D случайная величина $v_1X_1 + v_2X_2 + \dots + v_nX_n$ имеет такое же распределение, как величина $(|v_1|^p + |v_2|^p + \dots + |v_n|^p)^{\frac{1}{p}}X$, где X является случайной величиной с распределением D.

Заметим, что значение случайной величины $v_1X_1 + v_2X_2 + \dots + v_nX_n$ может быть получено как скалярное произведение вектора v на вектор a, полученный из p-устойчивого распределения X. Из определения следует, что эта случайная величина имеет такое же распределение, как p-норма вектора v, умноженная на случайную величину из того же распределения. Суть *метода эскизов* состоит в том, что для оценки $\|v\|_p$.вычисляется скалярное произведение va.

Нормальное распределение с нулевым математическим ожиданием является 2-устойчивым. В рамках применяемой схемы используется нормальное

распределение.

LSH использует устойчивые распределения для того, чтобы применить метод эскизов к векторам признаков большой размерности. Однако вместо того, чтобы использовать его для оценки нормы векторов, метод эскизов используется для получения значений хэш-функций от них. Эти значения отображают каждый вектор в точку на вещественной оси, которая разделена на равные по ширине сегменты длиной r, представляющие корзины. Определим каждую хэш-функцию из семейства следующим образом:

$$h(v) = \left\lfloor \frac{va + b}{r} \right\rfloor,$$

где v – исходный d-мерный вектор локальных признаков, a – d-мерный случайный вектор с элементами, независимо выбранными из p-устойчивого распределения, b – случайное число, которое выбирается из равномерного распределения на отрезке $[0, r]$.

Обозначим через $f_p(t)$ функцию плотности вероятности абсолютного значения рассматриваемого устойчивого распределения, и пусть $c = \|u - v\|_p$ для двух векторов u и v. Так как случайный вектор a из p-устойчивого распределения, $ua - va$ имеет распределение cX, где X является случайной величиной из устойчивого распределения. Поэтому вероятность совпадения значений хэш-функции

$$p(c) = P[h(u) = h(v)] = \frac{1}{c}\int_0^r \left(1 - \frac{t}{r}\right) f_p\left(\frac{t}{c}\right) dt.$$

Если два вектора u и v близки, то они будут иметь небольшую p-норму разности c и должны попасть в одну корзину с высокой вероятностью.

При фиксированном r вероятность коллизий $p(c)$ монотонно убывает по c. Таким образом, семейство хэш-функций является $\left(d_1, d_2, p(d_1), p(d_2)\right)$-чувствительным. Заметим, что для каждого c можно выбирать конечное r так, чтобы сделать p как можно меньше.

Глава 2. Реализация и тестирование алгоритмов

Цель данной главы – описать программную реализацию различных методов определения похожих изображений и сравнить результаты их работы.

Была рассмотрена *база* из 5586 изображений в формате JPEG с размерами по ширине от 112 до 1000 пикселей и высоте от 38 до 1000 пикселей включительно. Средняя ширина изображения составила 571 пиксель, а средняя высота – 459 пикселей. Среди данных изображений были группы изображений, отличающиеся друг от друга только различной обрезкой, надписями и по небольшим фрагментам. Параметры алгоритмов, такие как количество диапазонов значений компонент цветов, выбирались таким образом, чтобы такие похожие изображения успешно находились.

Для сравнения изображений различных размеров использовалось приведение размеров к эталонному разрешению (получение *эскизов*). Опытным путем было установлено, что для определения похожих изображений в данной базе хорошо подходит эталонное разрешение 16×16 пикселей.

Цвет каждого пикселя полученного эскиза кодировался методом TrueColor (24 бит на пиксель). По этому методу цвет пикселя представляется с использованием 256 уровней для каждой из трех компонент модели RGB: красного (R), зеленого (G) и синего (B), что в результате позволяет закодировать 16 777 216 (2^{24}) различных цветов. Таким образом, каждое изображение описывалось массивом целых чисел от 0 до 255 включительно с размерами $16 \times 16 \times 3$ (*цветовой матрицей*). Такое описание одного изображения занимает 768 байт.

Система Wolfram Mathematica 10.4.1 была выбрана для реализации алгоритмов потому, что она содержит удобные встроенные функции для работы с изображениями, обработки, анализа и визуального представления данных. Также стоит отметить, что используемый высокоуровневый мультипарадигмальный язык программирования Wolfram Language и удобная система встроенных функций позволяют получать очень короткий и интуитивно понятный

код, что уменьшает время разработки и облегчает понимание работы программы.

2.1. Сравнение цветовых матриц

Пусть задано изображение (*образец*), для которого требуется найти наиболее похожие изображения из имеющейся базы. Представим данное изображение описанным выше методом в виде цветовой матрицы размером $16 \times 16 \times 3$. Для решения задачи поиска похожих изображений нужно обеспечить хранение таких матриц для всей имеющейся базы изображений.

После этого для поиска похожих на образец изображений достаточно обойти в цикле все цветовые матрицы в базе, для каждой вычислить норму разности между этой матрицей и цветовой матрицей образца и найти в этом цикле изображение из базы, на котором достигается наименьшее значение нормы. В качестве нормы была использована сумма модулей всех компонент цветовой матрицы, так как она вычисляется наиболее просто и учитывает отклонения всех компонент.

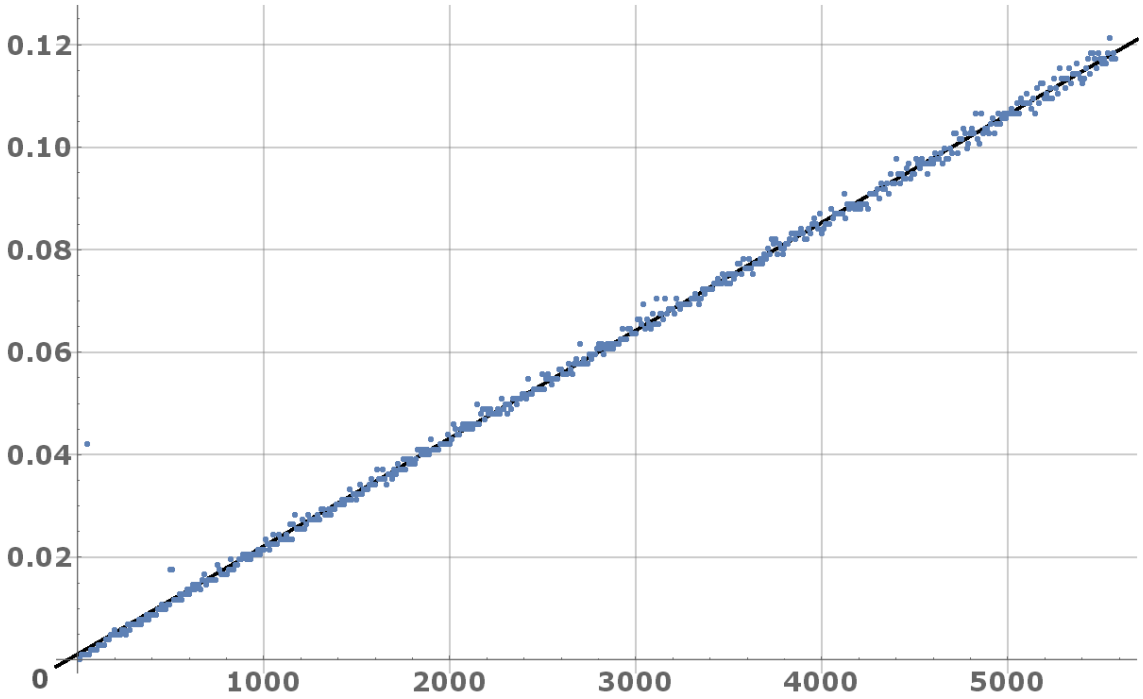

Рисунок 2.1. Время работы алгоритма сравнения норм разности цветовых матриц в зависимости от количества изображений в базе

На рисунке 2.1 с помощью синих точек, соответствующих запускам программы с различным размером базы со случайными образцами, взятыми из нее, представлено измеренное время (в секундах) работы программы в зависимости от количества изображений в базе (от 1 до 5586). Очевидно, время работы цикла поиска наименьшего значения нормы разности цветовых матриц в среднем прямо пропорционально количеству изображений в базе.

2.2. Коэффициент Жаккара

Теперь представим эскиз данного изображения-образца в виде *множества цветов*, состоящего из 16×16 цветов его пикселей. В качестве *цвета* пикселя будем использовать 3-мерный вектор целых чисел – компонент RGB, причем будем использовать числа от 0 до 15, разбив диапазон чисел от 0 до 255 из цветовой матрицы на 16 групп для увеличения вероятности совпадения цветов в изображениях.

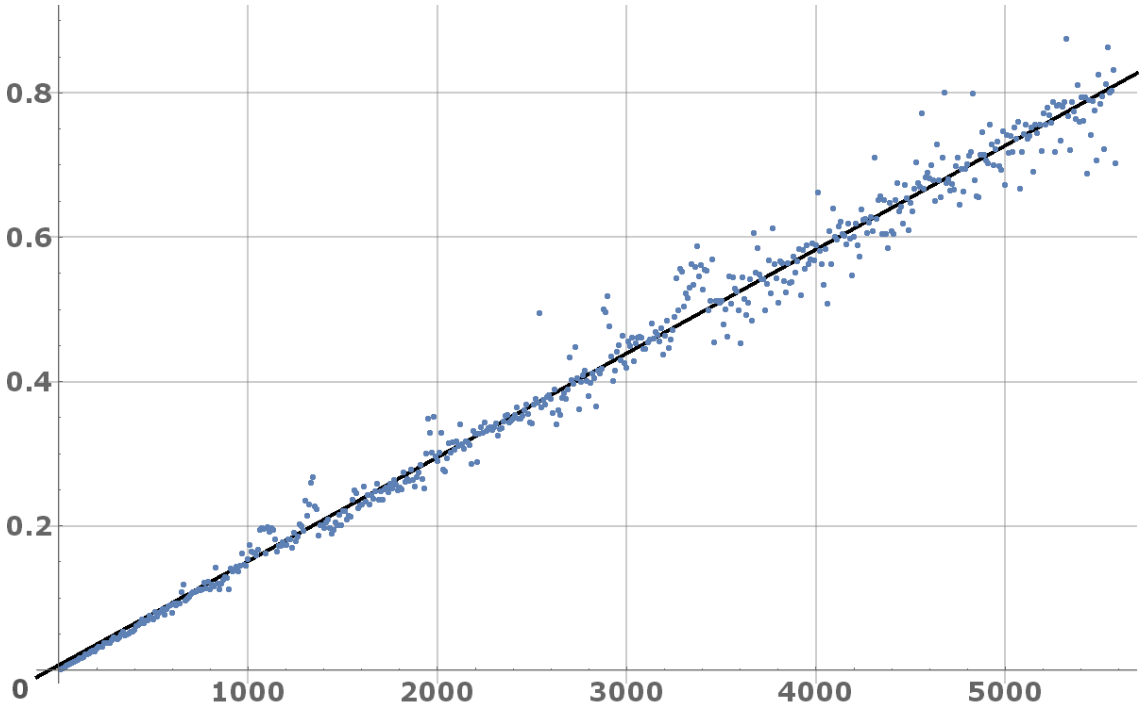

Рисунок 2.2. Время работы алгоритма сравнения цветовых множеств с помощью коэффициента Жаккара в зависимости от количества изображений в базе

Тогда для поиска похожих на образец изображений можно обойти в цикле все множества цветов в базе, вычислить коэффициент Жаккара между каждым

таким множеством и множеством цветов образца и найти в этом цикле изображение из базы, на котором достигается наибольшее значение коэффициента Жаккара.

2.3. Метод MinHash

Чтобы ускорить вычисление коэффициентов Жаккара для рассмотренных цветовых множеств, воспользуемся приближенным методом MinHash.

Для этого нужно предварительно вычислить матрицу сигнатур. Для ее получения зададим 1000 случайных хэш-функций со значениями от 0 до $16^3 = 4096$ (для имитации перестановки), отображающих цвет (вектор с 3 компонентами от 0 до 15) в число. Далее для каждого изображения из базы применим эти функции к каждому цвету из множества цветов данного изображения, и таким образом найдем наименьшее значение каждой хэш-функции для каждого изображения.

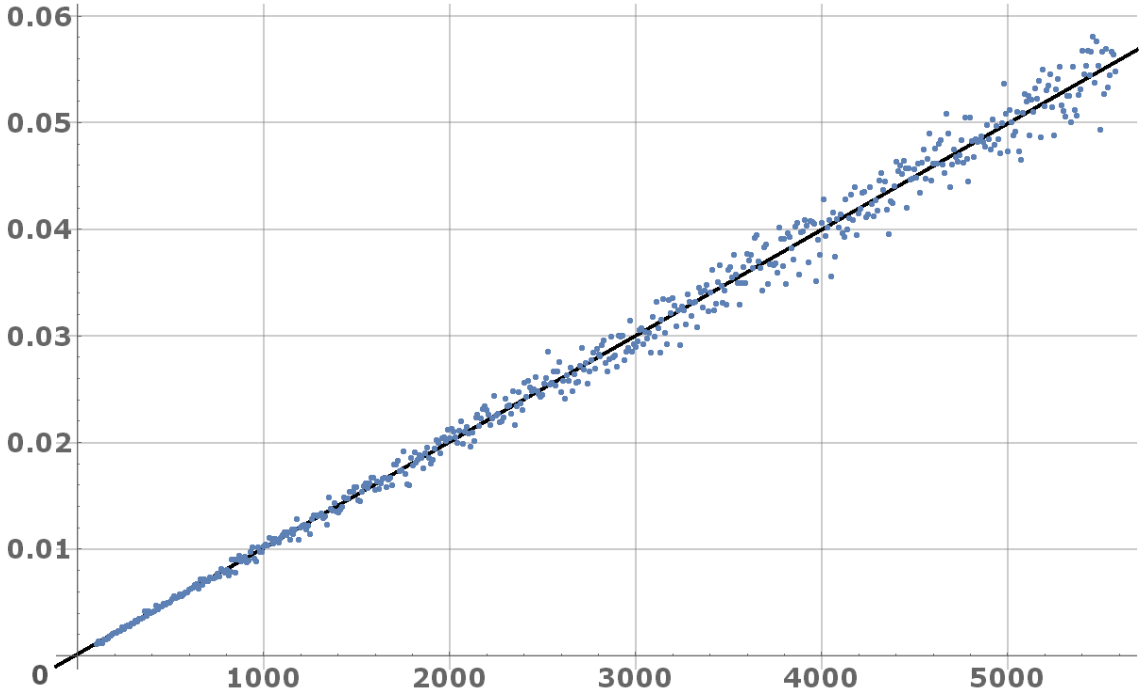

Рисунок 2.3. Время работы алгоритма сравнения MinHash-сигнатур в зависимости от количества изображений в базе

После этого для поиска похожих на образец изображений достаточно вычислить вектор сигнатур для множества цветов данного образца, и затем найти в цикле изображение из базы, столбец из матрицы сигнатур которого имеет

наибольшее количество одинаковых компонент с вектором сигнатур образца.

2.4. Метод LSH для сигнатур метода MinHash

Для ускорения работы метода сравнения MinHash-сигнатур используем метод понижения размерности LSH. Для работы с этим методом без перестроения корзин требуется изображение-образец выбирать из базы, по которой были построены корзины.

Разобьем матрицу сигнатур на группы по 2 строки. Далее в каждой группе выделим некоторое количество корзин, каждая из которых будет содержать по крайней мере 2 (этот параметр можно увеличивать для получения меньшего количества малозначимых корзин) изображения, которые имеют одинаковое значение хэш-функции от вектора из 2 значений MinHash, находящихся в данной группе.

После того, как все корзины построены, для поиска похожих на образец изображений можно выбрать все корзины, содержащие данный образец. Затем можно выполнить поиск наиболее часто встречающихся изображений из этих корзин.

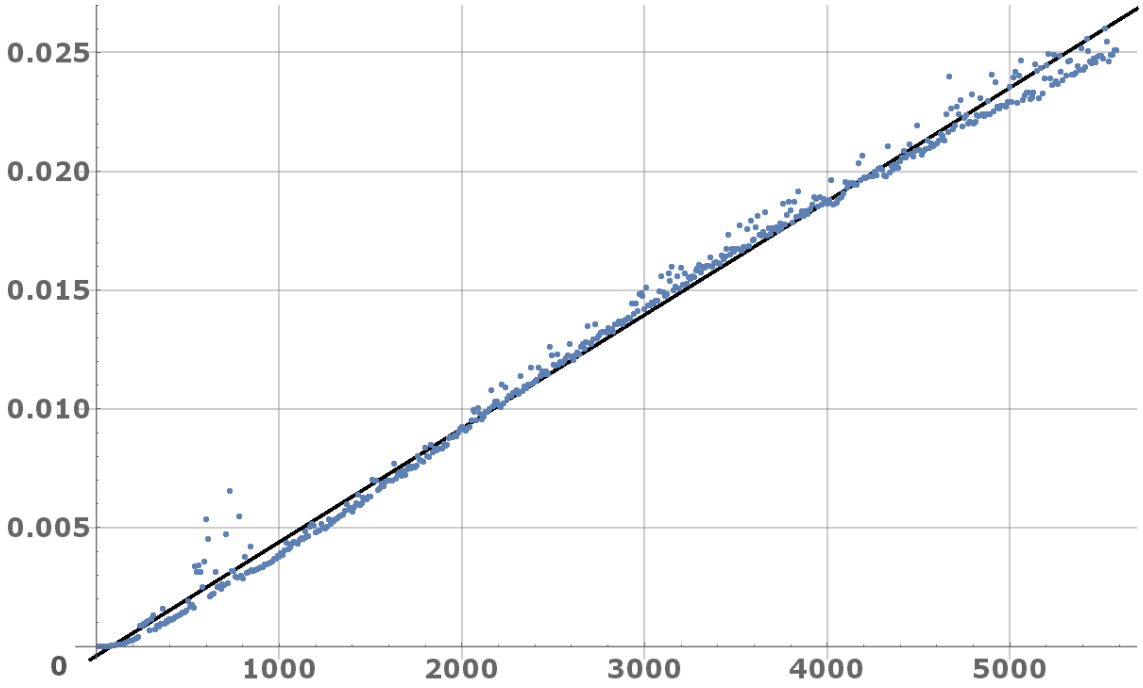

Рисунок 2.4. Время работы алгоритма LSH для MinHash-сигнатур в зависимости от количества изображений в базе

Выводы

После изучения работы программных реализаций описанных алгоритмов и выбора параметров были сформулированы следующие выводы.

1. Использование приведения размеров изображений к эталонным размерам эскизов позволяет быстро сравнивать их с помощью матричных представлений и уменьшает затраты памяти, которая требуется для работы алгоритмов.
2. Сравнение цветовых матриц по норме требует значительных затрат времени на вычисление норм разностей матриц, что не позволяет эффективно использовать данный алгоритм для решения задач классификации изображений. В отличие от других рассмотренных методов, данный метод использует представление изображения в виде упорядоченных элементов и, таким образом, позволяет сравнивать изображения с сохранением их структуры.
3. Прямое вычисление коэффициентов Жаккара для сравнения множественных представлений изображений занимает очень много времени. Метод MinHash представляет собой быстрое и эффективное приближение такого способа и пригоден для решения задач классификации изображений.
4. В случае, если размерность векторов сигнатур высока и время работы алгоритма MinHash неприемлемо, ее можно уменьшить, используя LSH. На время работы алгоритма непосредственно влияет количество построенных корзин. Использование корзин при правильном подборе параметров также позволяет автоматически решать задачи разделения изображений на группы похожих между собой. Метод LSH на имеющихся данных показал лучшее время работы из рассматриваемых алгоритмов.

Заключение

В данной работе были рассмотрены теоретически основы методов MinHash и LSH и с помощью оценок вероятностей обосновано их использование. Описаны общие классы локально-чувствительных хэш-функций, изучена реализация метода LSH для векторов локальных дескрипторов Google VisualRank.

В среде Wolfram Mathematica 10.4.1 реализованы алгоритмы сравнения эскизов изображений как матриц цветов и с использованием множественного представления на основе цветов. Получены значения параметров, при которых LSH уменьшает время работы метода MinHash с сохранением качества определения похожих изображений.

Таким образом, все поставленные задачи были выполнены.

Список литературы

1. Udi Manber. Finding similar files in a large file system. In in Proceedings of the USENIX Winter 1994 Technical Conference, pages 1–10, 1994.

2. A.Z. Broder. On the resemblance and containment of documents. In Compression and Complexity of Sequences 1997. Proceedings, pages 21–29, 1997.

3. Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Similarity search in high dimensions via hashing. pages 518–529, 1997.

4. Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. pages 604–613, 1998.

5. A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In Foundations of Computer Science, 2006. FOCS ’06. 47th Annual IEEE Symposium on, pages 459–468, 2006.

6. Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations. Journal of Computer and System Sciences, 60:327–336, 1998.

7. Zixiang Kang, Wei Tsang Ooi, and Qibin Sun. Hierarchical, non-uniform locality sensitive hashing and its application to video identification. In Multimedia and Expo, 2004. ICME ’04. 2004 IEEE International Conference on, volume 1, pages 743–746 Vol.1, 2004.

8. S. Hu. Efficient video retrieval by locality sensitive hashing. In Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP ’05). IEEE International Conference on, volume 2, pages 449–452, 2005.

9. B. Kulis and K. Grauman. Kernelized locality-sensitive hashing. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34(6):1092–1104, 2012.

10. Tanmoy Mondal, Nicolas Ragot, Jean-Yves Ramel, and Umapada Pal. A fast word retrieval technique based on kernelized locality sensitive hashing. In Document Analysis and Recognition (ICDAR), 2013 12th International Conference on, pages 1195–1199, 2013.

11. Maheshwari A. Topics in Algorithm Design-Lecture Notes for COMP 5703 [Электронный ресурс]/ A. Maheshwari. – School of Computer Science Carleton University, 2015. – 142 с. – Режим доступа: доступ свободный.

http://people.scs.carleton.ca/~maheshwa/courses/5703COMP/Notes/edited-notes.pdf