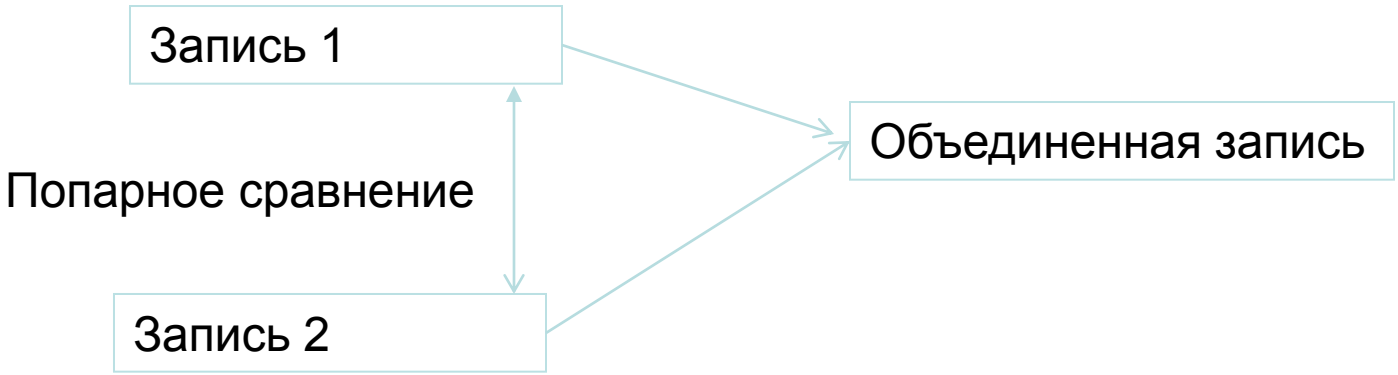
Слияние большого числа библиографических записей

Косолапов К.А. (ВМК)
Серебряков В.А. (ВЦ ФИЦ)
Теймуразов К.Б. (ВЦ ФИЦ)
Шорин О.Н. (РНБ)

Постановка задачи

21 млн записей



Попарное сравнение 21 млн * 21 млн

Постановка задачи

Вместо попарного сравнения записей желательно:

1. Сравнивать не записи, а их короткие «заменители»

2. Каким-то образом заранее разбить все множество записей на группы кандидатов предположительно имеющих сходство.

Меры близости

- Мера Jaccard схожести множеств S и T - это $|S \cap T|/|S \cup T|$, т.е. Отношение числа элементов в пересечении к числу элементов в объединении.
- Косинусная мера: скалярное произведение векторов.
- Угловая мера: 1-угол между векторами/180
- Расстояние Хэмминга между битовыми векторами: |S XOR T|.
- Меру схожести между S и T будем обозначать SIM(S,T).

Представление множеств матрицами

Признак	S1	S2	S3	S4
a	1	0	0	1
b	0	0	1	0
c	0	1	0	1
d	1	0	1	1
e	0	0	1	0

Minhashing

Broder, Andrei Z.; Charikar, Moses; Frieze, Alan M.; Mitzenmacher, Michael (1998), "Min-wise independent permutations", Proc. 30th ACM Symposium on Theory of Computing (STOC '98), New York, NY, USA: Association for Computing Machinery, pp. 327–336

Нам надо заменить большие множества значительно меньшими представлениями, называемыми «Следами» (“fingerprints”, “signatures”, “sketches”). Основное свойство следов – возможность сравнения следов двух множеств для оценки меры схожести Jaccard исходных множеств. Ясно, что они не могут дать точного значения меры схожести.

Далее предполагается, что след – результат вычисления hash-функции. Для вычисления minhash представим множество колонкой характеристической матрицы и выберем перестановку строк. Значение minhash колонки – это номер первой строки перестановки, в которой колонка имеет 1. Каждая перестановка определяет свою hash-функцию. На предыдущем слайде h(S1)=1, h(S2)=3, h(S3)=2, h(S4)=1. Для перестановки (3,1,5,4,2) получаем матрицу

Для перестановки (3,1,5,4,2) получаем матрицу

Признак	S1	S2	S3	S4
b	0	0	1	0
e	0	0	1	0
a	1	0	0	1
d	1	0	1	1
c	0	1	0	1

И соответственно h(S1)=3, h(S2)=5, h(S3)=1, h(S4)=3

Minhashing и мера Jaccard

Имеет место замечательная связь между Minhashing и мерой Jaccard:

Вероятность, что функция Minhash для случайной перестановки строк дает одно и то же значение для двух множеств, равна мере Jaccard этих множеств.

Рассмотрим матрицу для двух множеств. Строки можно разделить на три категории:

X. 1 в обеих колонках

Y. 1 в одной колонке и 0 в другой

Z. 0 в обеих колонках.

Поскольку строки типа Z нас не интересуют, вероятность того, что при движении сверху-вниз будет встречена строка вида X, равна |X|/(|X|+|Y|), а это и есть мера Jaccard.

Minhash множества

Пусть теперь мы сделали n перестановок h1, h2, . . . , hn. Этот набор перестановок будем называть minhash. Для каждой колонки S назовем minhash следом S вектор [h1(S), h2(S), . . . , hn(S)]. Т.о. из характеристической матрицы мы формируем матрицу следов, в которой i-я колонка – это minhash след i-й колонки характеристической матрицы. Попарная близость колонок hi(Sj) = hi(Sk) совпадает с мерой Jaccard множеств Sj и Sk.

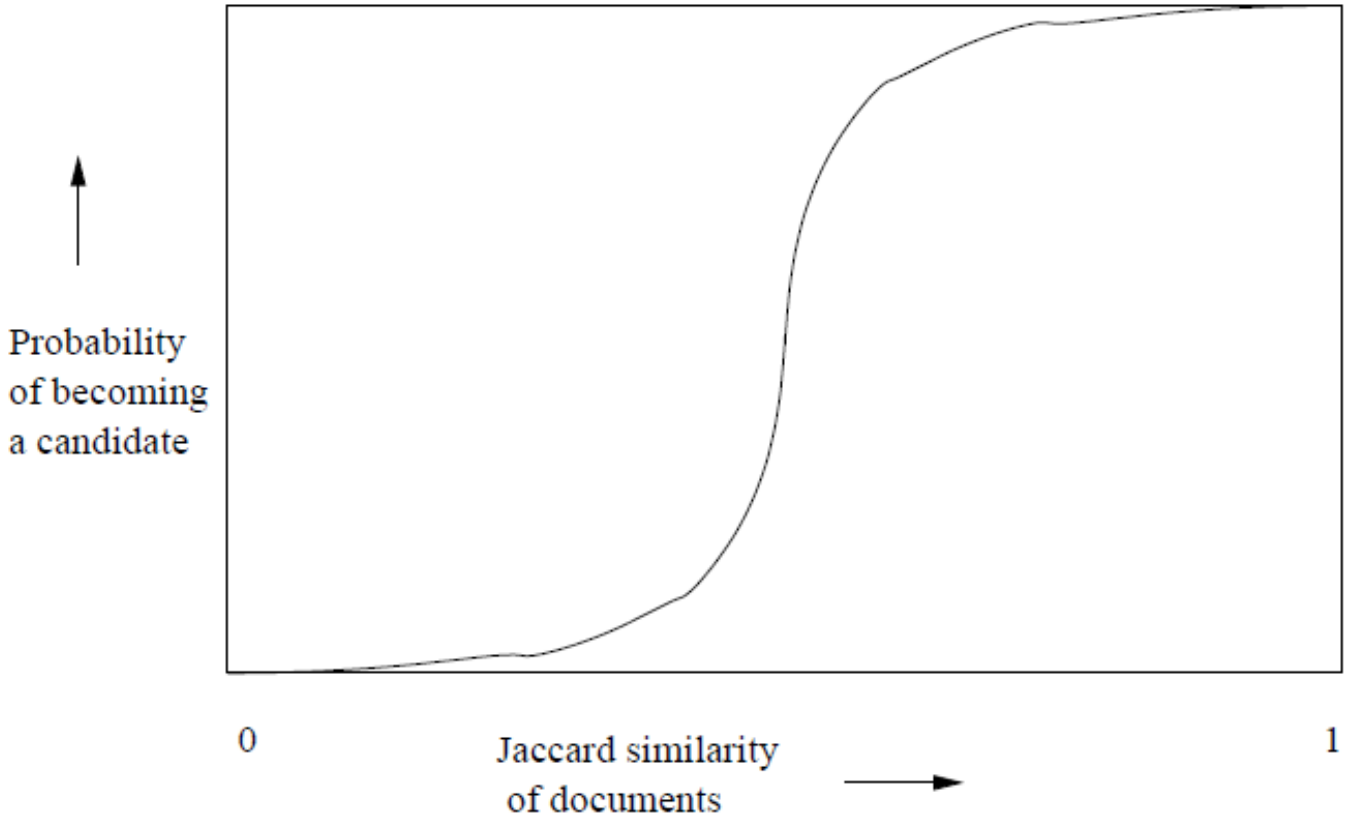
Локально чувствительное хеширование

Indyk, Piotr.; Motwani, Rajeev. (1998). "Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality.". Proceedings of 30th Symposium on Theory of Computing.
Gionis, A.; Indyk, P.; Motwani, R. (1999). "Similarity Search in High Dimensions via Hashing". Proceedings of the 25th Very Large Database (VLDB) Conference.

При поиске дубликатов даже попарное сравнение следов может оказаться нереализуемым. Но часто нас интересует не точное совпадение, а совпадение с некоторой точностью.
Разобьем матрицу следов на b блоков, состоящих каждый из r строк.
Предположим, что пара документов имеет меру Jaccard s. Тогда:

1. Вероятность того, что следы равны во всех строках одного блока – s^r.
2. Вероятность того, что следы НЕ равны по крайней мере в одной строке блока – $1-s^r$.
3. Вероятность того, что следы НЕ равны по крайней мере в одной строке каждого блока – $(1-s^r)^b$.
4. Вероятность того, что следы равны во всех строках по крайней мере одного блока и поэтому становятся кандидатами на близость – $1-(1-s^r)^b$.

Эта функция имеет вид S-кривой



Порог, т.е. значение схожести s, при котором вероятность стать кандидатом равна ½, это функция b и r.

Алгоритм слияния

1. Выбрать n для следов.
2. Выбрать порог t, который определяет документы как близкие. Выбрать b число блоков и r число строк такие, что br = n и порог t примерно равен $(1/b)^{1/r}$.
3. Отобрать пары-кандидаты.
4. Проверить каждую пару кандидатов и определить действительно ли доля компонент, в которой они равны, не меньше t.
5. Опционально: сравнить сами документы.

SimHash - Сравнение на основе расстояний

[6] G. S. Manku, A. Jain, and A. Das Sarma, \Detecting near-duplicates for web crawling," *Proc. ACM WWW, pp. 141{150, 2007.*
[7] M. S. Charikar, \Similarity estimation techniques from rounding algorithms," *Proc. ACM STOC, pp. 380{388, 2002.*

Расстояние d должно удовлетворять следующим аксиомам:
1. $d(x, y) \geq 0$ (неотрицательно).
2. $d(x, y) = 0$ титтк $x = y$ (0 только для совпадающих точек).
3. $d(x, y) = d(y, x)$ (симметрия).
4. $d(x, y) \leq d(x, z) + d(z, y)$ (свойство треугольника).

Евклидово расстояние $d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = (\sum_i (x_i - y_i)^2)^{1/2}$

Расстояние Jaccard $d(x, y) = 1 - SIM(x, y)$.

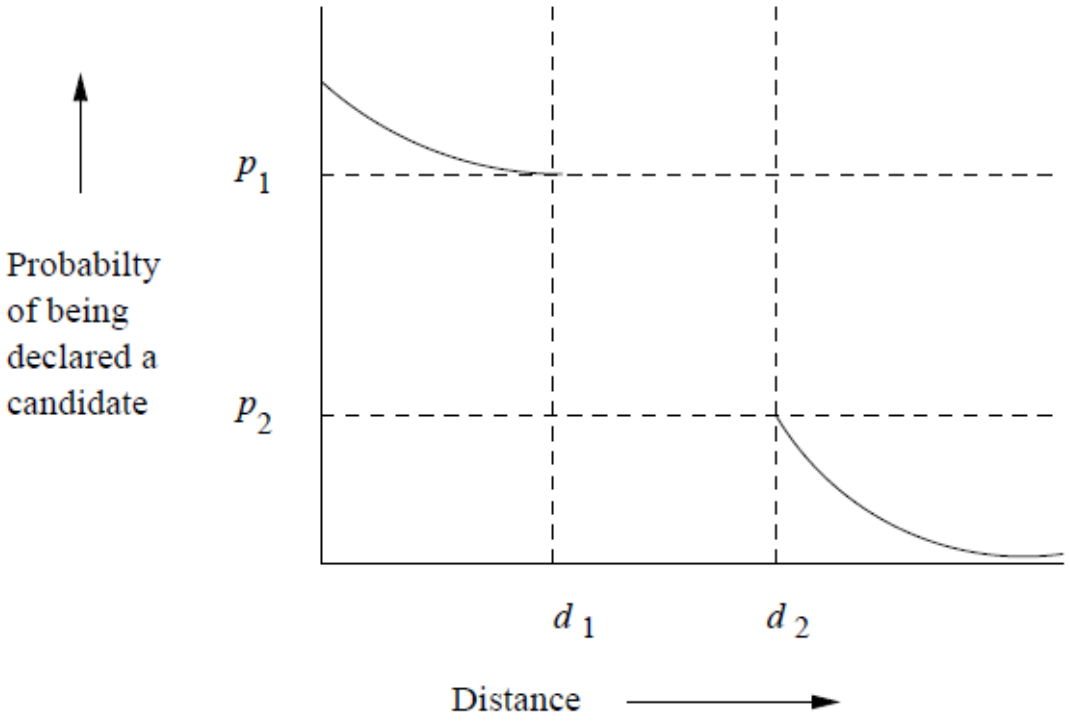
Косинус угла $d([x_1, x_2, \ldots, x_n], [y_1, y_2, \ldots, y_n]) = \sum_i x_i y_i / (|x| \cdot |y|)$

Расстояние Хэмминга – элементы вектора 0 и 1, расстояние – число несовпадающих значений координат.

Семейство функций

Для этого применяется локально чувствительное хеширование (LSH) или поиск близких соседей. Пусть d1 < d2 - два расстояния в соответствии с некоторым расстоянием d. Говорят, что семейство F функций(d1, d2, p1, p2)-чувствительно, если для любой f ∈F:

1. Если d(x, y) ≤ d1, то вероятность, что f(x) = f(y) не меньше p1.
2. Если d(x, y) ≥ d2, то вероятность, что f(x) = f(y) не больше p2.



Эквивалентное определение

Схема локально-чувствительного хэширования – это семейство F хэш функций над множеством объектов такое, что для двух объектов x, y,

$Pr_{h \in F}[h(x) = h(y)] = sim(x, y)$

OR расширение локально чувствительного семейства функций

Если определить f(x) = f(y) тттк fi(x) = fi(y) для одного или более значений i, получим OR-конструкцию – по (d1, d2, p1, p2)-чувствительному семейству F строится (d1, d2, 1 – (1 – p1)^b, 1 – (1 – p2)^b)-чувствительное семейство F′.

Если f ∈ F′– это множество {f1, f2, . . . , fb} элементов F, будем говорить, что f(x) = f(y) тогда и только тогда, когда fi(x) = fi(y) для одного из i = 1, 2, . . . , b. OR-конструкция отражает комбинирование блоков: (x, y) пара кандидатов, если (x, y) пара кандидатов в каком-нибудь блоке.

LSH для расстояния Hamming

Пусть h(x, y) расстояние Хэмминга между векторами x и y. Можно определить функцию $f_i(x)$ как i-й бит вектора x. Тогда $f_i(x) = f_i(y)$ титтк векторы x и y совпадают в i-й позиции. Тогда вероятность того, что $f_i(x) = f_i(y)$ для случайно выбранного i равна $1 – h(x, y)/d$ (d – число бит), т.е. доле совпадающих позиций. Это в точности совпадает с ситуацией с minhashing. Т.о. семейство F, состоящее из функций $\{f_1, f_2, \ldots, f_d\}$, - это $(d_1, d_2, 1 – d_1/d, 1 – d_2/d)$-чувствительное семейство хэш-функций для любых $d_1 < d_2$.

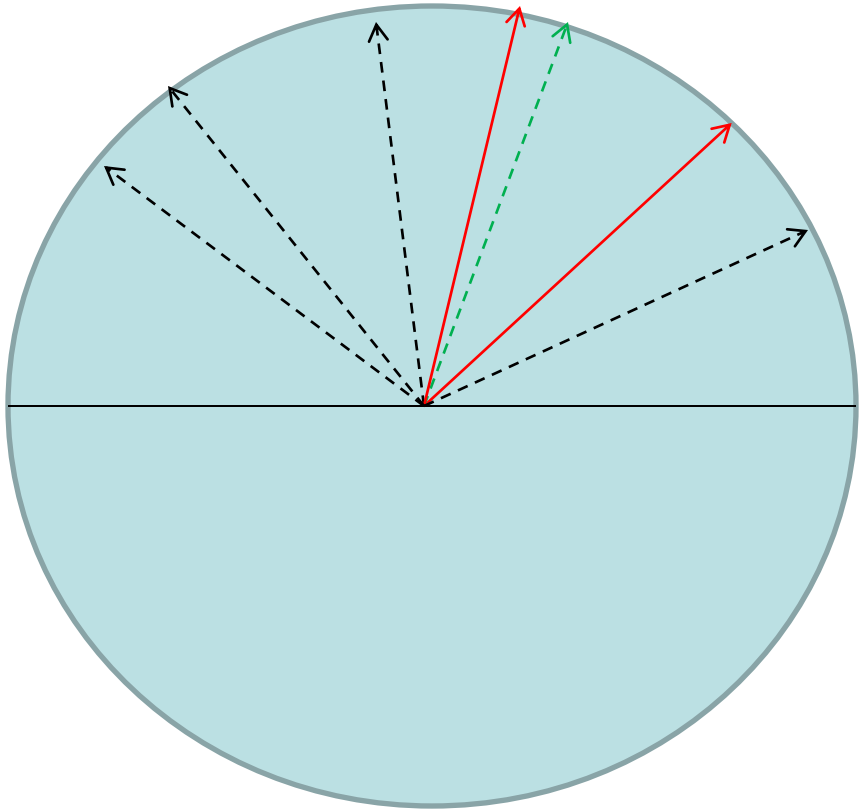
Случайные гиперплоскости и угловое расстояние

Similarity Estimation Techniques from Rounding Algorithms

Moses S. Charikar Dept. of Computer Science
Princeton University

Построим семейство функций следующим образом: Каждая функция f строится по случайно выбранному вектору v. Если даны два вектора x и y, будем говорить, что f(x) = f(y) титтк когда скалярные произведения (v,x) и (v,y) имеют один и тот же знак. Тогда F – локально-чувствительное семейство с угловым (косинус) расстоянием. Его параметры те же, что и для семейства с расстоянием Jaccard, за исключением того, что расстояние меняется 0–180, а не 0–1. Т.е., F (d_1, d_2, (180 – d_1)/180, (180 – d_2)/180)-чувствительное семейство хэш-функций.

Угловое расстояние



Вычисление SimHash

Для каждого документа SimHash = (вектор характеристик, случайный вектор). Случайный вектор – это хэш-функция f. Плоскость задается вектором характеристик документа. Знак скалярного произведения (т.е. элемента SimHash) определяет положение случайного вектора относительно плоскости. SimHash –то AND локально-чувствительное семейство хэш функций.

Разные знаки скалярного произведения означают, что случайные векторы расположены по разные стороны от плоскости. Если в векторах SimHash двух разных документов стоят разные знаки, значит соответствующий случайный вектор расположен по разные стороны относительно гиперплоскостей двух документов.

Ищем между ними расстояние Хэмминга.

SimHash

Число характеристик F

Число разрядов d

Число документов N

Матрица
[документ N, характеристики F]

Матрица
[число характеристик,
число разрядов]

Элементы матрицы –
случайные векторы

Число характеристик F

Значение скалярного произведения	-10	95	66	-4	..	..	..	..	..	..
SimHash i	0	1	1	0	..	..	..	..	..	..
SimHash	1	1	0	0	..	..	..	..	..	..

Интеграция библиографических записей

Имеем N биграмм. Для каждой биграммы b имеем вектор h(b) и вес w(b). h(b) – это случайный вектор. Получаем матрицу M: Биграмма **x** h(b). Заменяем в матрице 0 на -1, получаем столбцы – векторы из 1 и -1 – это нормальные векторы для гиперплоскостей. Берем случайный вектор из значений биграмм данного множества (документа), скалярно умножаем на каждый из векторов матрицы, получаем либо >=0, либо <0. Это определяет, лежит ли вектор из значений биграмм данного множества (документа) по одну или другую сторону от гиперплоскости, определяемой ее вектором. Заменяем значение >=0 на 1, значение <0 на 0, получаем вектор следов.

Строго говоря, получаем косинус угла, а не сам угол, что требуется по Теореме, но они не сильно расходятся.

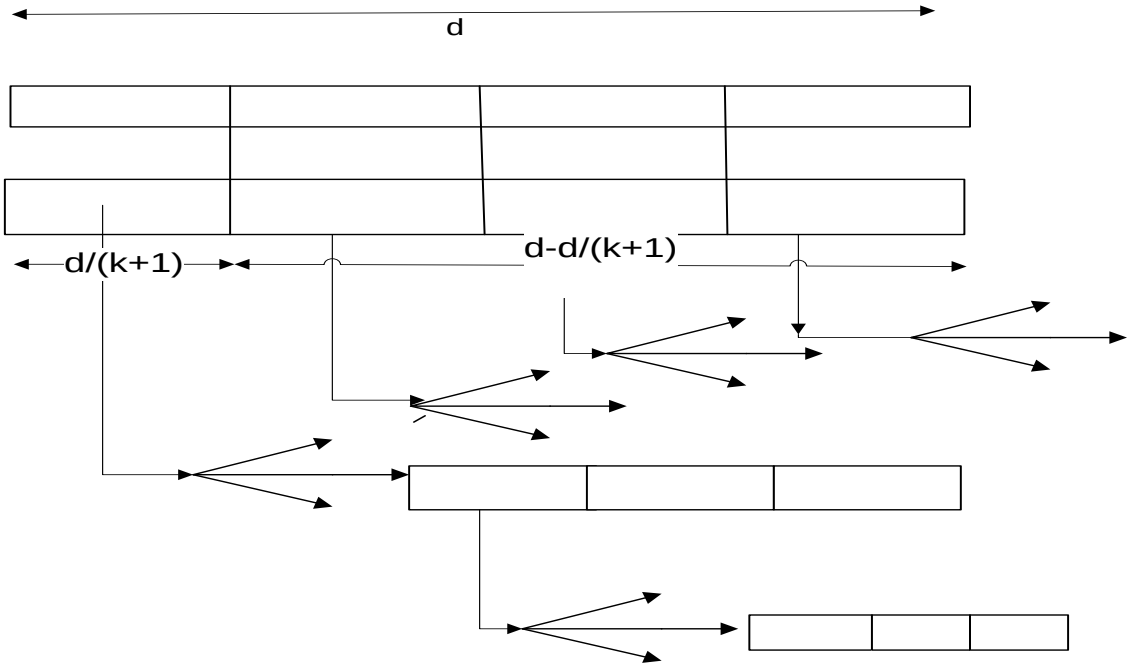
Проблема расстояния Хэмминга

Пусть d – длина simhash, в нашем случае 64. Пусть S – общее число записей, т.е. «задействованных» simhash, в нашес случае примерно 2^{25}. Обозначим $S=2^f$, т.е. в нашем случае f=25. Пусть k – критерий поиска, т.е. по скольким битам мы ищем различия, т.е при заданном simhash q мы ищем во множестве S такие, которые отличаются от q не более, чем в k битах.
Идея заключается в том, чтобы разбить d на группы. Выделенные биты образуют входы в таблицу. Остальные биты сравниваем на расстояние k.
Необходимо и достаточно, чтобы таких групп было k+1.
K берем исходя из меры Jaccard. Если это 0.75, то k=16.
На самом деле это вопрос. Отношение «расстояние Jaccard» не транзитивно, так что не есть отношение эквивалентности. Мы не можем отталкиваясь от записи набрать все, отстоящие на J. Если взять слишком большое k, то такие группы будут очень большими, если маленькое – можем не поймать нужные.
С другой стороны, поскольку мы ищем эквивалентные записи, то содержательно должны быть группы эквивалентности. Формально это определить нельзя.

Дерево поиска

Detecting NearDuplicates for Web Crawling
Gurmeet Singh Manku Google Inc.
Arvind Jain Google Inc.
Anish Das Sarma
Stanford University

Как найти элементы, отличающиеся на k бит?



Попарное сравнение

На предыдущем шаге для каждой записи выбираются записи-кандидаты на попарное сравнение. Затем пары сравниваются. Сравнение заключается в построении биграмм и сопоставлении множеств этих биграмм.

Сложность

Для данного q сложность поиска равна (число таблиц)*(lg длины таблицы, т.е. число бит, по которым строится таблица) * (сложность поиска во множестве).
При $n \to \infty$ это все равно n^2 , но реально при $n <= 2^{64}$

Результаты

Сервер:
2. Процессор – Intel(R) Xeon(R) CPU E5645 @ 2.40GHz x2, два процессора. 24 виртуальных ядра, 12 реальных ядер.
3. Оперативная память – 98238 MBytes DDR3. 1066Mhz
4. Жесткие диски – RAID0 Intel Multi-Flex SCSI Disk Device. Средняя скорость чтения данных – 150 MB/s

Windows Server 2012 Standard.

Результаты

Алгоритм выявления дублетных библиографических записей обработал 21 313 009 записей. Для обработки такого объема данных на указанном сервере потребовалось 90 часов. В результате работы алгоритма было выявлено, что 1 239 293 библиографические записи являются дублетами какой-нибудь другой записи.
Уникальных записей, т.е. записей, для которых не существует ни одного дублета, оказалось 19 066 057. Статистика по количеству дублетных записей записей
0 - 19 066 057
1 - 699 687
2 - 101 490
3 - 29 247
4 - 11 655
5 - 6 777
...
71 - 500

Спасибо за внимание