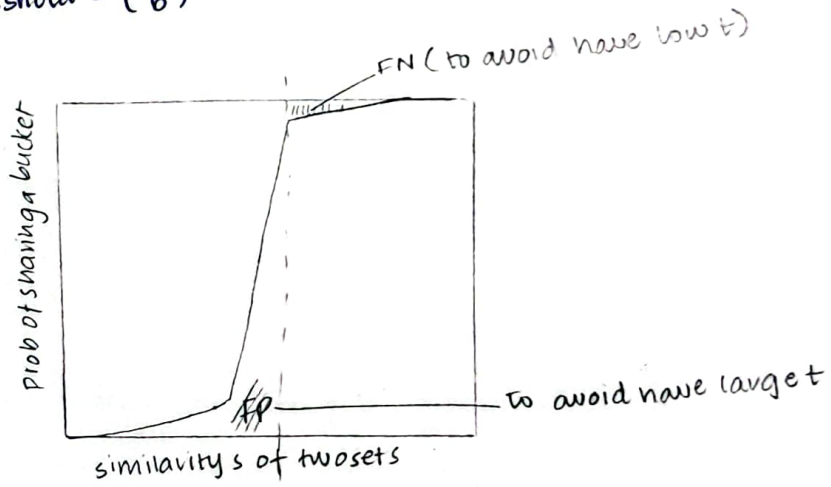
BDA

$(C_1, C_2) = \frac{C_1 \cap C_2}{C_1 \cup C_2}$

- signature matrix M from random permutations.
- comparing original matrix and matrix after random permutation or hash functions.
- if hash function not given, then take $x+1 \bmod 5$, $3x+1 \bmod 5$ where x = row number.

LOCALITY SENSITIVE HASHING

- hash columns to many buckets and make elements of the same bucket candidate pairs.
- divide matrix M into bands of r rows.
- k = buckets, should be large.
- probability that the signatures agree in all rows of one particular band is s^r
- probability that the signatures disagree in that atleast one row of a particular band is $1-s^r$
- probability that the signatures disagree in that atleast one row of each band is $(1-s^r)^b$
- probability that the signatures agree in all the rows of atleast one band and candidate pair is $1-(1-s^r)^b$.
- threshold = $(\frac{1}{b})^{1/r}$



LSH involves tradeoff

1. number of bands.
   → increasing number of bands, then false positive increases as candidate pair increases.

2. increasing number of buckets
   → False positive decrease as it reduces the likelihood of dissimilar items being hashed to same bucket.

3. decreasing number of buckets.
   → False positive: increase higher likelihood of dissimilar item being hashed to the same bucket.
   → False negative decreases, similar items likely to hash in same bucket and reduces chance of missing out on genuine similarities.

4. increasing number of hash function.
   → False positive decreases as less chances of dissimilar item to be considered similar.

5. decreasing number of hash function
   → False positive increases, and increasing likelihood of dissimilar items being hashed to same bucket

HADOOP

server: computers that provide services to clients.

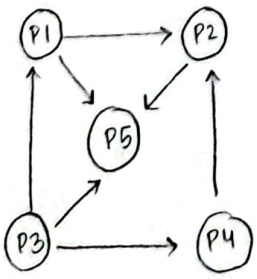
data centre: a facility used to house computer systems and components such as networking and storage systems.

PAGERANK

- the importance of a page is distributed to pages that it points to.

$$r(P_i) = \frac{\sum r(P_j)}{|P_j|}$$

→ rank of previous iteration.
→ outgoing links



Pages	P1	P2	P3	P4
Pagerank	3.5	1.2	4.2	1.0

pagerank of P_5 computed as:

$$r(P_5) = \frac{3.5}{2} + \frac{1.2}{1} + \frac{4.2}{3} = 4.35$$

→ P5 ki jitni incoming links hain unke ranks ko add kiye hain.

$$= \frac{\text{pagerank}}{\text{outgoing links}}$$

0:

initialization: $\pi^0 = \left(\frac{1}{n} \quad \frac{1}{n} \quad \frac{1}{n} \cdots \frac{1}{n}\right)$

$H' =$

$$\begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

$= \left[\frac{1}{6} \quad \frac{1}{6} \quad \frac{1}{6} \quad \frac{1}{6} \quad \frac{1}{6} \quad \frac{1}{6}\right]$

$= \frac{1}{6} \times \frac{1}{3} = 0.0556$

$= \frac{1}{6} \times \frac{1}{2} + \frac{1}{6} \times \frac{1}{3} = 0.1389$

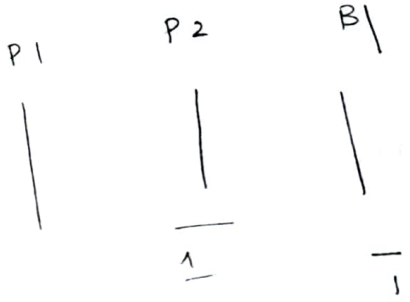
$= \frac{1}{6} \times \frac{1}{2} = 0.0833$

$= \frac{1}{6} \times \frac{1}{2} + \frac{1}{6} \times 1 = 0.25$

$= \frac{1}{6} \times \frac{1}{3} + \frac{1}{6} \times \frac{1}{2} = 0.1389$

$= \frac{1}{6} \times \frac{1}{2} + \frac{1}{6} \times \frac{1}{2} = 0.1667$

$\pi' = [0.0556 \quad 0.1389 \quad 0.0833 \quad 0.25 \quad 0.1389 \quad 0.1667]$



$\pi^2 = \pi' \times H$

$= 0.0833 \times \frac{1}{3} = 0.0277$

$= 0.0556 \times \frac{1}{2} + 0.0833 \times \frac{1}{3} = 0.05556$

$= 0.0556 \times \frac{1}{2} = 0.0277$

$= 0.1389 \times \frac{1}{2} + 0.1667 = 0.23615$

$= 0.0833 \times \frac{1}{3} + 0.25 \times \frac{1}{2} = 0.1528$

$= 0.25 \times \frac{1}{2} + 0.1389 \times \frac{1}{2} = 0.1944$

$\pi^2 = [0.0277 \quad 0.0556 \quad 0.0277 \quad 0.236 \quad 0.1528 \quad 0.194]$

final pagerank is = [0,0,0, 0.2667, 0.133, 0.2]

0.4 lost; to normalize divide all answers by 0.6

iterations of P2 doesn't add up to 1.

• dangling pages have no outgoing links and random surfer cannot proceed forward from these pages.

row-normalized: all non-zero rows sum to 1

stochastic matrix: all rows sum to 1

row x column

1x1 5x5

1x5 5x5

$$H' = \begin{bmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

4 repeating matrix multiplication $\pi^{(k+1)} = \pi^{(k)} H$,
no pagerank is lost.

5 • allow teleporting to any page at any time.

$G = \alpha H' + (1-\alpha)\left(\frac{1}{n}\right) I$

$\pi^{k+1} = \pi^{k} G$ π^0

$\alpha = 0.85 \rightarrow$ google.

$\frac{1}{1-\alpha}$ → average number of steps before jumping to random pages.

d

R

•

FREQUENT PATTERN MINING

$$\text{confidence}(I \to j) = \frac{\text{support}(I \cup j)}{\text{support}(I)}$$

$$\text{interest}(I \to j) = \text{conf}(I \to j) - \Pr[j]$$

NAIVE ALGORITHM

generate pairs = $\frac{n(n-1)}{2}$ total bytes = $2n^2$

fail = $(\text{number of items})^2$ exceeds main memory.

approach 1 : count all pairs using matrix

approach 2 : requires 4 bytes per pair. – triangular matrix
Keep a table of triples.

A-PRIORI ALGORITHM

pass 1 : read baskets and count in main memory the occurrence of each individual item.

pass 2 : read baskets again and count in main memory only those elements whose pairs are frequent.

* first write all possible combination then discard those which are less than support.

PARK-CHEN YU ALGORITHM

pass 1 : in addition to item count, maintain a hash table.
Keep a count for each bucket into which pairs of items are hashed.

pass 2 : only count pairs that hash to frequent buckets.

* replace bucket with bit vector, 1 means bucket count exceeded the support

SON ALGORITHM

pass 1 : find candidate itemsets. Map(F,1) Reduce : union all the (F,1).

pass 2 : frequent itemsets Map (c,v) Reduce : add all the (c,v)

pass 1 map task : divide data into chunks, for each chunk support = support/chunks, find frequent items, discard < support and output key value (F,1) for remaining.

pass 1 reduce task : produces key greater than threshold, basically union of both chunks.

pass 2 map task : takes output of pass 1 reduce task and writes its support as input
outputs item and its support for both chunks.

pass 2 reduce task : calculates sum of both chunks that are greater than support.

STREAMING DATA

apache kafka : distributed publish - subscribe messaging system.

synchronous replication: producers write data to primary storage and replica simultaneously.

asynchronous replication: producers write data to the primary storage first and then copy data to the replica.

: it costs less, requires less bandwidth.

broker : kafka node on the cluster

topics: a unique name.

producer : push message into kafka topic

consumer : pulls message off a kafka topic

role of breakers → producers send message to Brokers, store messages, consumers subscribe messages from brokers.

- messages should be small to achieve maximum I/o efficiency
- topics are partitioned among multiple ~~folders~~ brokers.
- each partition is a separate folder.
- replication factor defines the number of copies of the partition.
- in-sync replicas are the subset of all the replicas for a partition having same messages as the leader.

FILTERING DATA STREAMS.

- a bloom filter will declare that certain URL have been seen before
- but they can be false positive as well. It can declare a URL has been seen before when it hasn't.
- $h_1(x)$ = take odd numbered bits from right $?_0|_011 = ?$
  $h_2(x)$ = take even numbered bits from right $?_0|_011 = ?$

filter contents → 0 1 2 3 . . .

example 25 = 11001 $h_1(x) = 101$ $h_2(x) = 10$

$5_0|_011 = 5$ $2_0|_011 = 2$

stream element ... 3, 2, 1 ←

2nd and 5th bit is 1 in filter contents.

- to lookup element, all bits shall be 1

example lookup element is 118 → 1110110 $h_1(x) = 1110$ $h_2(x) = 101$

$14_0|_011 = 3$ $5_0|_011 = 5$

...lity of false positive depends upon number of 1's

(fraction of 1) ^ # of hash functions. → $1 - e^{-m/n}$ or $1 - e^{-d/t}$

...rgets = bits/bucket

... darts, n targets.

d darts, t targets.

- probability of hitting anyone of the targets → $(\frac{1}{n})$ or $(\frac{1}{t})$
- probability of not hitting → $(1 - \frac{1}{t})$ or $(1 - \frac{1}{n})$
- probability of not hitting targets by all m/d darts → $(1 - \frac{1}{n})^m$ or $(1 - \frac{1}{t})^d$ or $e^{-m/n}$ or $e^{-d/t}$
- throwing k.m darts at n so fraction of 1 is $(1 - e^{-km/n})$ k = number of hash functions

$\downarrow$

$(1 - e^{-km/n})^k$

- optimal value of $k = \frac{n}{m} \ln(2)$