NOT IN THE EXAM

15-826: Multimedia Databases and Data Mining

Lecture #32: BONUS LECTURE

Approximate Counting

C. Faloutsos

1

Material

- Christopher Palmer, Phillip B. Gibbons and Christos Faloutsos,*ANF: A Fast and Scalable Tool for Data Mining in Massive Graphs*, KDD 2002
- *Efficient and Tunable Similar Set Retrieval*, by Aristides Gionis, Dimitrios Gunopulos and Nikos Koudas, SIGMOD, 2001.
- *New sampling-based summary statistics for improving approximate query answers*, by Phillip B. Gibbons and Yossi Matias, ACM SIGMOD, 1998.

2

Outline

- **Flajolet-Martin (and Cohen) – vocabulary size (Problem #1)**
- Application: Approximate Neighborhood function (ANF)
- other, powerful approximate counting tools (Problem #2, #3)

3

Problem #1

- Given a multiset (eg., words in a document)
- find the vocabulary size (#, after dup. elimination)

A A A B A B A C A B

Voc. Size = 3 = |{A, B, C}|

4

Carnegie Mellon

Thanks to

- Chris Palmer (Vivisimo->IBM)

Carnegie Mellon

Problem #2

- Given a multiset
- compute approximate high-end histogram = hot-list query = (*k* most common words, and their counts)

A A A B A B A C A B D D D D D

(for k=2:
A#: 6
D#: 5)

Carnegie Mellon

Problem #3

- Given two documents
- compute quickly their similarity (#common words/ #total-words) == Jaccard coefficient

7

Carnegie Mellon

Problem #1

- Given a multiset (eg., words in a document)
- find the vocabulary size V (#, after dup. elimination)
- using space O(V), or O(log(V))

(Q1: Applications?)

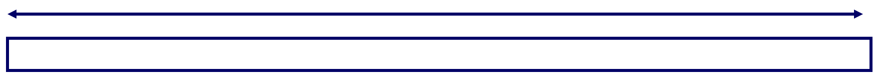
(Q2: How would you solve it?)

8

Carnegie Mellon

Basic idea (Cohen)

large bit string, initially all zeros



A

A

C

9

Carnegie Mellon

Basic idea (Cohen)

large bit string, initially all zeros

A

hash!

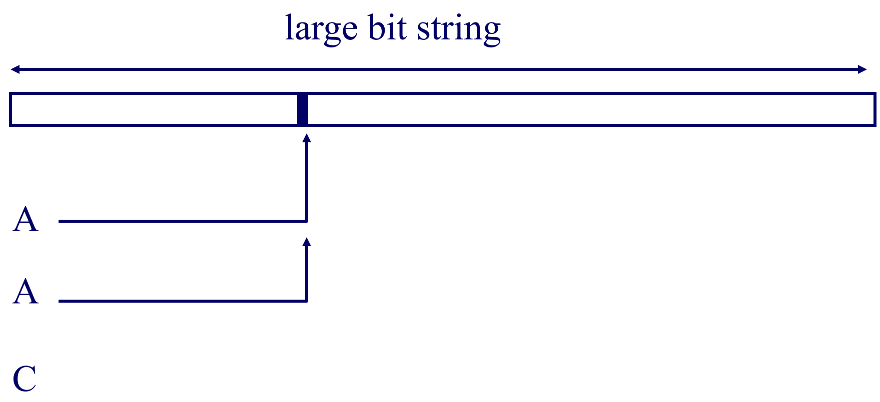
A

C

10

Carnegie Mellon

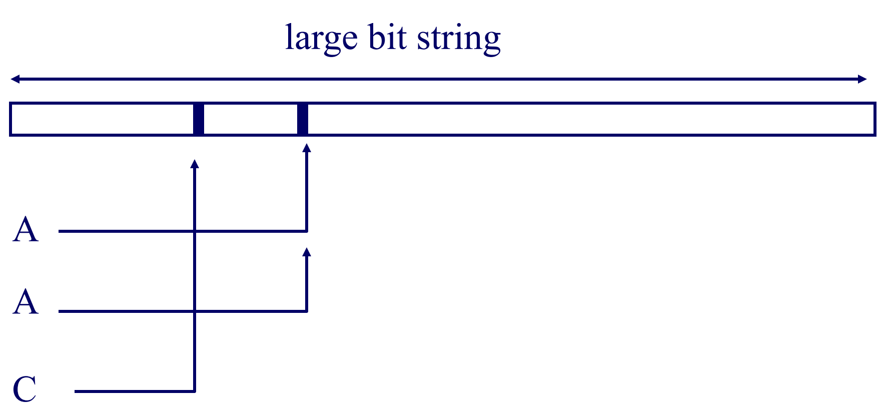
Basic idea (Cohen)



11

Carnegie Mellon

Basic idea (Cohen)

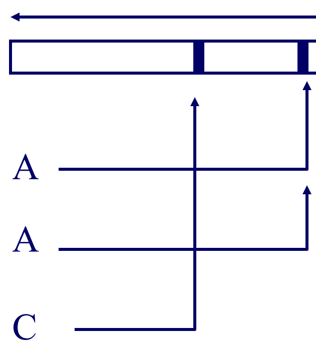


12

Carnegie Mellon

Basic idea (Cohen)

large bit string



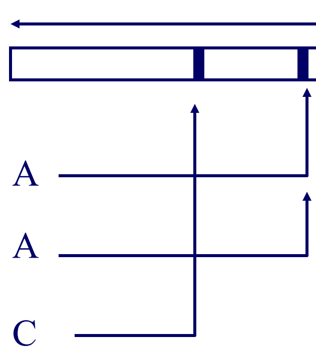
the rightmost position depends on the vocabulary size
(and so does the left-most)

Repeat, with several hashing functions, and merge the estimates

Carnegie Mellon

Basic idea (Cohen)

large bit string

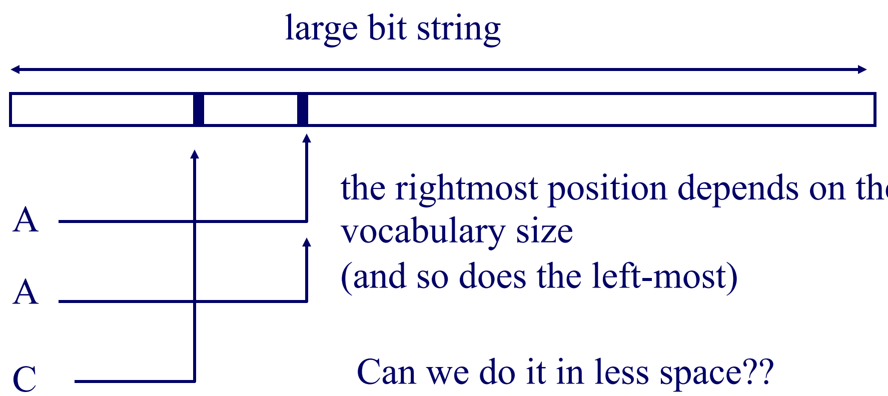


the rightmost position depends on the vocabulary size
(and so does the left-most)

Can we do it in less space??

Carnegie Mellon

Basic idea (Cohen)



the rightmost position depends on the vocabulary size
(and so does the left-most)

Can we do it in less space??
YES

15

Carnegie Mellon

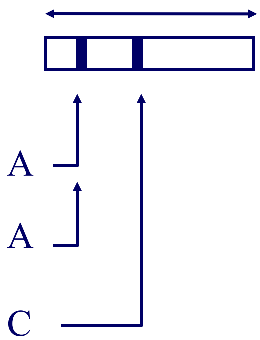
How?

16

Carnegie Mellon

Basic idea (Flajolet-Martin)

O(log(V)) bit string (V: voc. size)



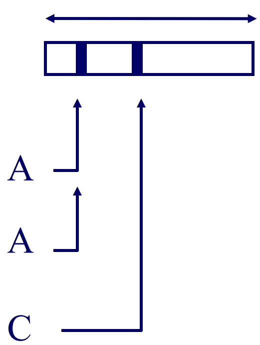
first bit: with prob. ½
second: with prob. ¼
...
i-th: with prob. ½**i

17

Carnegie Mellon

Basic idea (Flajolet-Martin)

O(log(V)) bit string (V: voc. size)



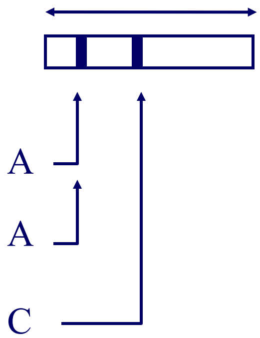
again, the rightmost bit
'reveals' the vocabulary size

18

Carnegie Mellon

Basic idea (Flajolet-Martin)

O(log(V)) bit string (V: voc. size)



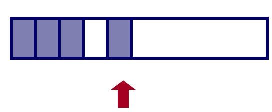
again, the rightmost bit 'reveals' the vocabulary size

Eg.: V=4, will probably set the 2nd bit, etc

Carnegie Mellon

Flajolet-Martin

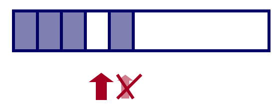
- Hash multiple values of *X* to same signature
  - Hash each *x* to a bit, using exponential distr.
  - ½ map to bit 0, ¼ map to bit 1, …
- Do several different mappings and average
  - Gives better accuracy
  - Estimate is: 2^b / .77351 / *BIAS*
    - *b* ~ rightmost '1', and actually:



Carnegie Mellon

Flajolet-Martin

- Hash multiple values of *X* to same signature
  - Hash each *x* to a bit, using exponential distr.
  - ½ map to bit 0, ¼ map to bit 1, …
- Do several different mappings and average
  - Gives better accuracy
  - Estimate is: $2^b / .77351 / BIAS$
    - *b : average least zero bit in the bitmask*
    - *bias : 1+.31/k for k different mappings*
- Flajolet & Martin prove this works



21

Carnegie Mellon

FM Approx. Counting Alg.

Assume $X = \{ 0, 1, \ldots, V\text{-}1 \}$
FOR $i = 1$ to k DO *bitmask[i] = 0000…00*
Create k random hash functions, $hash_i$
FOR each element x of M DO
    FOR $i = 1$ to k DO
        $h = hash_i(x)$
        bitmask[i] = bitmask[i] LOR h
Estimate: b = average least zero bit in *bitmask[i]*
            $2^b/.77351/(1+.31/k)$

- How many bits? *log V* + small constant
- What hash functions?

22

Carnegie Mellon

Random Hash Functions

- Can use linear hash functions. Pick random (a_i, b_i) and then the hash function is:
  - $lhash_i(x) = a_i * x + b_i$
- Gives uniform distribution over the bits
- To make this exponential, define
  - $hash_i(x)$ = least zero bit in $lhash_i(x)$
- Hash functions easy to create and fast to use

Carnegie Mellon

Conclusions

- Want to measure # of distinct elements
- Approach #1: (Flajolet-Martin)
  - Map elements to random bits
  - Keep bitmask of bits
  - Estimate is $O(2^b)$ for least zero-bit b
- Approach #2: (Cohen)
  - Create random permutation of elements
  - Keep least element seen
  - Estimate is: $O(1/le)$ for least rank le

Carnegie Mellon

Approximate counting

- Flajolet-Martin (and Cohen) – vocabulary size
- **Application: Approximate Neighborhood function (ANF)**
- other, powerful approximate counting tools

25

Carnegie Mellon

details

Fast Approximation of the "neighborhood" Function for Massive Graphs

Christopher R. Palmer
Phillip B. Gibbons
Christos Faloutsos

KDD 2001

26

Carnegie Mellon

details

Motivation

- What is the diameter of the Web?
- What is the effective diameter of the Web?
- Are the telephone caller-callee graphs for the U.S. similar to the ones in Europe?
- Is the citation graph for physics different from the one for computer science?
- Are users in India further away from the core of the Internet than those in the U.S.?

27

Carnegie Mellon

details

Proposed Tool: neighborhood

Given graph $G=(V,E)$

$N(h)$ = # pairs within h hops or less

= **neighborhood function**

28

Carnegie Mellon

details

Proposed Tool: neighborhood

Given graph $G=(V,E)$

$N(h)$ = # pairs within h hops or less

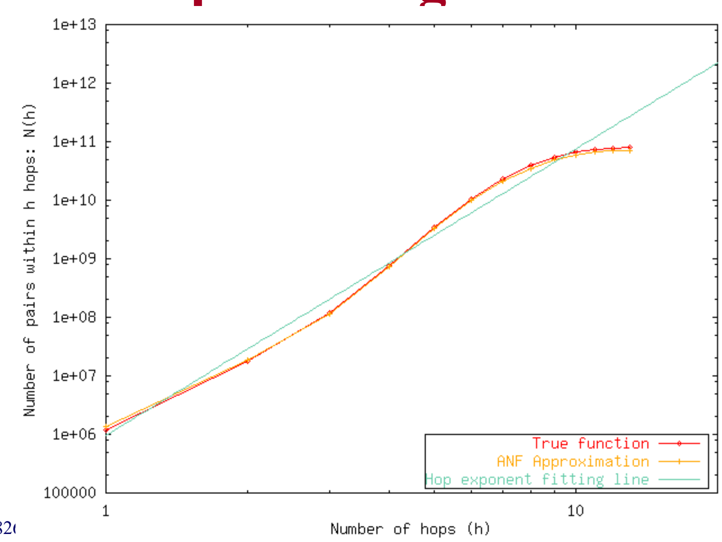
= **neighborhood function**

$N(u,h)$ = # neighbors of node u, within h hops or less

Carnegie Mellon

details

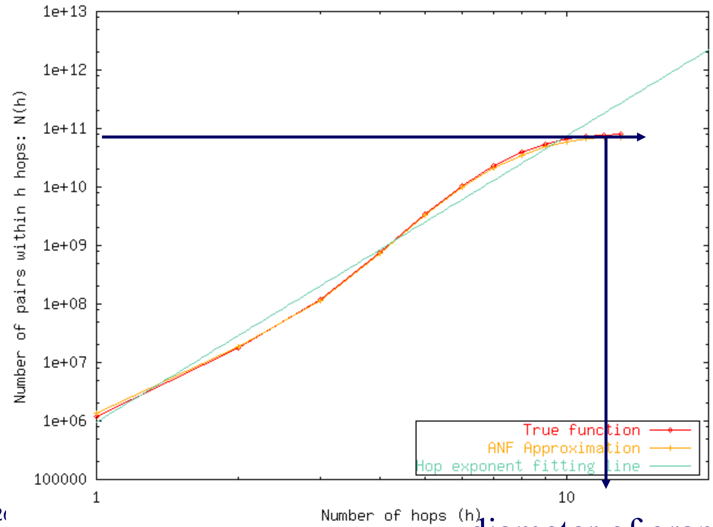
Example of neighborhood



Carnegie Mellon

details

Example of neighborhood



~diameter of graph

15-82 31

31

Carnegie Mellon

details

Requirements (for massive g

- *Error guarantees*
- *Fast*: (and must scale linearly with graph)
- *Low storage requirements*: massive graphs!
- *Adapts to available memory*
- *Sequential scans of the edges*
- Also estimates *individual neighborhood functions* |S(u,h)|
  – These are actually quite useful for mining

15-826 Copyright (c) 2019 C. Faloutsos 32

32

Carnegie Mellon

details

How would you compute it?

- Repeated matrix multiply
  - Too slow $O(n^{2.38})$ at the very least
  - Too much memory $O(n^2)$
- Breadth-first search

  FOR each node u DO

  bf-search to compute $S(u,h)$ for each h
  - Best known exact solution!
  - We will use this as a reference
- Approximations? Only 1 that we know of which we will discuss when we evaluate it.

33

Carnegie Mellon

details

Intuition

- Guess what we'll use?
  - Approximate Counting!
- Use very simple algorithm:

FOR each node u DO $S(u,0) = \{ (u,u) \}$ initialize to self-only

FOR $h = 1$ to *diameter* of G DO

FOR each node u DO $S(u,h) = S(u,h-1)$ can reach same things

FOR each edge (u,v) in G DO and add one more step

$S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h-1) \}$

34

Carnegie Mellon

details

Intuition

- Guess what we'll use?
  - Approximate Counting!
- Use very simple algorithm:

(distinct) neighbors of u, within h hops

FOR each node u DO $S(u,0) = \{ (u,u) \}$ initialize to self-only

FOR $h = 1$ to *diameter* of G DO

FOR each node u DO $S(u,h) = S(u,h\text{-}1)$ can reach same things

FOR each edge (u,v) in G DO and add one more step

$S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h\text{-}1) \}$

(distinct) neighbors of v, within h-1 hops

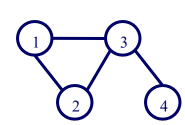
35

Carnegie Mellon

details

Trace

h=0



{(1,1)}

{(2,2)}

{(3,3)}

{(4,4)}

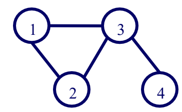
36

Carnegie Mellon

details

Trace

h=0	h=1
{(1,1)}	{(1,1)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}



15-826 37

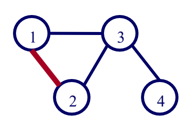
37

Carnegie Mellon

details

Trace

h=0	h=1
{(1,1)}	{(1,1)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}



15-826 38

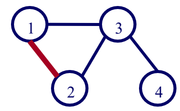
38

Carnegie Mellon

details

Trace

h=0 h=1



h=0	h=1
{(1,1)}	{(1,1), (1,2)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}

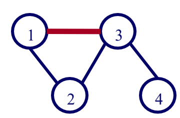
15-826 Copyright (c) 2019 C. Faloutsos 39

39

Carnegie Mellon

details

Trace



h=0	h=1
{(1,1)}	{(1,1), (1,2), (1,3)}
{(2,2)}	{(2,2)}
{(3,3)}	{(3,3)}
{(4,4)}	{(4,4)}

15-826 Copyright (c) 2019 C. Faloutsos 40

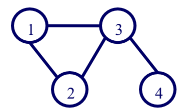
40

Carnegie Mellon

details

Trace

h=0 h=1



h=0	h=1
{(1,1)}	{(1,1), (1,2), (1,3)}
{(2,2)}	{(2,2), (2,1), (2,3)}
{(3,3)}	{(3,3), (3,1), (3,2), (3,4)}
{(4,4)}	{(4,4), (4,3)}

15-826 41

41

Carnegie Mellon

details

Intuition

(distinct) neighbors of u, within h hops

- Guess what we' ll use?
  - Approximate Counting!
- Use very simple algorithm:

FOR each node *u* DO $S(u,0) = \{ (u,u) \}$ initialize to self-only

FOR $h = 1$ to *diameter* of *G* DO

    FOR each node *u* DO $S(u,h) = S(u,h-1)$ can reach same things and add one more step

    FOR each edge (u,v) in *G* DO

        $S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h-1) \}$

15-826 42

42

details

Intuition

- Guess what we'll use?
  - Approximate Counting!
- Use very simple algorithm:

(distinct) neighbors of u, within h hops

FOR each node u DO $S(u,0) = \{ (u,u) \}$ initialize to self-only
FOR $h = 1$ to *diameter* of G DO
    FOR each node u DO $S(u,h) = S(u,h\text{-}1)$ can reach same things
    FOR each edge (u,v) in G DO and add one more step
        $S(u,h) = S(u,h) \cup \{ (u,v') : (v,v') \in S(v,h\text{-}1) \}$

- Too slow and requires too much memory
- Replace expensive set ops with bit ops

15-826 43

43

details

ANF Algorithm #1

FOR each node, u, DO
    $M(u,0)$ = concatenation of $\boldsymbol{k}$ bitmasks of length $log\ n + \boldsymbol{r}$
        each bitmask has 1 bit set (exp. distribution)
DONE

FOR $h = 1$ to *diameter* of G DO
    FOR each node, u, DO $M(u,h) = M(u,h\text{-}1)$
    FOR each edge (u,v) in G DO
        $M(u,h) = (M(u,h)$ OR $M(v,h\text{-}1))$

    Estimate $N(h) = Sum(N(u,h)) = Sum\ 2^{b(u)} / .77351 / (1+.31/k)$
        where $b(u)$ = average least zero bit in $M(u,it)$
DONE

15-826 44

44

details

ANF Algorithm #1

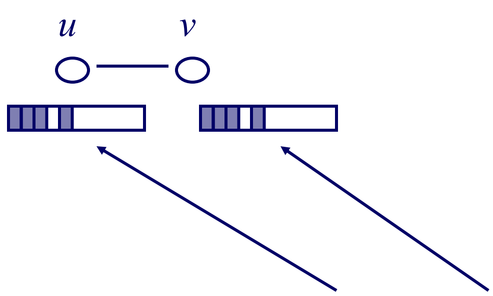
FOR each node, u, DO
  $M(u,0)$ = concatenation of $\boldsymbol{k}$ bitmasks of length $log\ n$ + $\boldsymbol{r}$
    each bitmask has 1 bit set (exp. distribution)
DONE

FOR $h = 1$ to *diameter* of G DO
  FOR each node, u, DO $M(u,h) = M(u,h-1)$
  FOR each edge (u,v) in G DO
    $M(u,h) = (M(u,h)$ OR $M(v,h-1))$

  Estimate $N(h) = Sum(N(u,h)) = Sum\ 2^{b(u)} / .77351 / (1+.31/k)$
    where $b(u)$ = average least zero bit in $M(u,it)$
DONE

details

ANF Algorithm #1



whatever u can reach with h hops plus whatever v can reach with h-1 hops
Duplicates: automatically eliminated!

$M(u,h) = (M(u,h)$ OR $M(v,h-1))$

Carnegie Mellon

details

Properties

- **Has error guarantees**: (from F&M)
- **Is fast**: $O((n+m)d)$ for n nodes, m edges, diameter d (which is typically small)
- **Has low storage requirements**: $O(n)$
- **Easily parallelizable**: Partition nodes among processors, communicate after full iteration
- **Does sequential scans of edges**.
- **Estimates individual neighborhood functions**
- DOES NOT work with limited memory

Carnegie Mellon

Conclusions

- Approximate counting (ANF / Martin-Flajolet) take minutes, instead of hours
- and discover interesting facts quickly

Carnegie Mellon

Outline

- Flajolet-Martin (and Cohen) – vocabulary size (Problem #1)
- Application: Approximate Neighborhood function (ANF)
- other, powerful approximate counting tools (**Problem #2**, #3)

15-826 49

49

Carnegie Mellon

Problem #2

- Given a multiset
- compute approximate high-end histogram = hot-list query = (*k* most common words, and their counts)

A A A B A B A C A B D D D D D

(for *k*=2:
A#: 6
D#: 5)

15-826 50

50

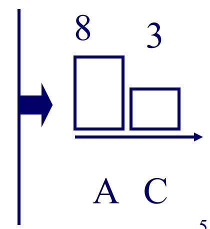
Carnegie Mellon

Hot-list queries

- Given a stream of product ids (with duplicates)
- Compute
  - the *k* most frequent products,
  - and their counts
- with a SINGLE PASS and O(*k*) memory

A A B A C A B C A A D E A C A

k=2



15-826 Copyright (c) 2019 C. Faloutsos 51

51

Carnegie Mellon

Applications?

15-826 Copyright (c) 2019 C. Faloutsos 52

52

Carnegie Mellon

Applications?

- Best selling products
- most common words
- most busy IP destinations/sources (DoS attacks)
- summarization / synopses of datasets
- high-end histograms for DBMS query optimization

15-826 53

53

Carnegie Mellon

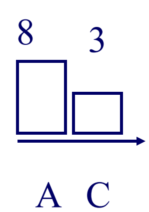
Hot-list queries

•Given a stream of product ids (with duplicates)
•Compute
  •the *k* most frequent products,
  •and their counts
•with a SINGLE PASS and O(*k*) memory

A A B A C A B C A A D E A C A

Exact: impossible
Thus: **approximate**

k=2



15-826 54

54

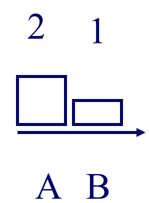
Carnegie Mellon

Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count

A A B A C A B C A A D E A C A

k=2



15-826 Copyright (c) 2019 C. Faloutsos 55

55

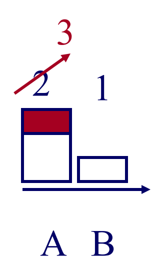
Carnegie Mellon

Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count

A A B A C A B C A A D E A C A

k=2



15-826 Copyright (c) 2019 C. Faloutsos 56

56

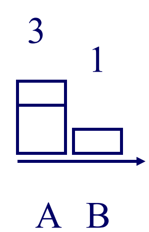
Carnegie Mellon

Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count
  - else ??

A A B A C A B C A A D E A C A

k=2



15-826 57

57

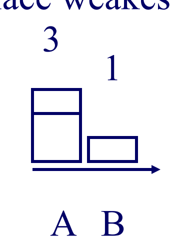
Carnegie Mellon

Hot-list queries - idea

- Keep the (approx.) *k* best so far, plus counts
- for a new item, if it is in the hot list
  - increment its count
  - else TOSS a coin, and possibly displace weakest

A A B A C A B C A A D E A C A

k=2



15-826 58

58

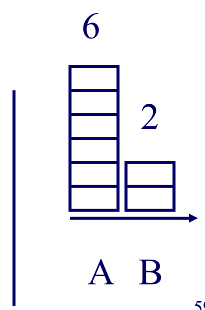
Carnegie Mellon

Hot-list queries - idea

- Biased coin - what are the Head/Tail prob.?

A A B A C A B C A A D E A C A

k=2



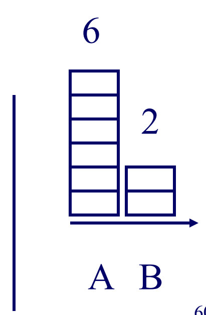
Carnegie Mellon

Hot-list queries - idea

- Biased coin - what are the Head/Tail prob.?
- A: depends on count(weakest)

A A B A C A B C A A D E A C A

k=2



Carnegie Mellon

Hot-list queries - idea

- Biased coin - what are the Head/Tail prob.?
- A: depends on count(weakest)
- and the new item ('D'), if it wins, it gets the **count of the item it displaced**.

Carnegie Mellon

Hot-list queries - idea

- See [Gibbons+Matias 98] for proofs

Carnegie Mellon

Outline

- Flajolet-Martin (and Cohen) – vocabulary size (Problem #1)
- Application: Approximate Neighborhood function (ANF)
- other, powerful approximate counting tools
  - Problem #2,
  - **Problem #3**

63

Carnegie Mellon

Problem #3

- Given two documents
- compute quickly their similarity (#common words/ #total-words) == Jaccard coefficient

64

Carnegie Mellon

Problem #3'

- Given a query document *q*
- and many other documents
- compute quickly the *k* nearest neighbors of *q*, using the Jaccard coefficient

D1: {A, B, C} q: {A, C, D, W}
D2: {A, D, F, G}
…

Carnegie Mellon

Applications?

Carnegie Mellon

Applications?

- Set comparisons eg.,
  - snail-mail address (set of trigrams)
- search engines - 'similar pages'
- social networks: people with many joint friends (facebook recommendations)

Carnegie Mellon

Problem #3'

- Given a query document *q*
- and many other documents
- compute quickly the *k* nearest neighbors of *q*, using the Jaccard coefficient

- Q: how to extract a fixed set of numerical features, to index on?

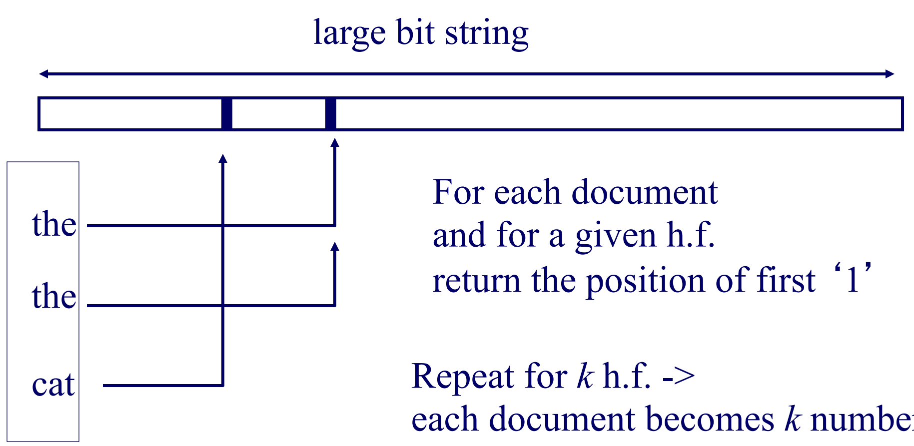
Carnegie Mellon

Answer

- Approximation / hashing - Cohen:

Carnegie Mellon

Basic idea (Cohen)



For each document
and for a given h.f.
return the position of first '1'

Repeat for *k* h.f. ->
each document becomes *k* numbers

Carnegie Mellon

Idea

- Doc1: n1, n2, ... nk
- Doc2: n1', n2', ... nk'

Carnegie Mellon

Idea

- Doc1: n1, n2, ... nk
- Doc2: n1', n2', ... nk'

1 *m*

- say they agree on *m* values

Carnegie Mellon

Idea

- Doc1: n1, n2, ... nk
- Doc2: n1', n2', ... nk'

- say they agree on *m* values,
- then

Jaccard(Doc1, Doc2) ~ *m/k*

15-826 Copyright (c) 2019 C. Faloutsos 73

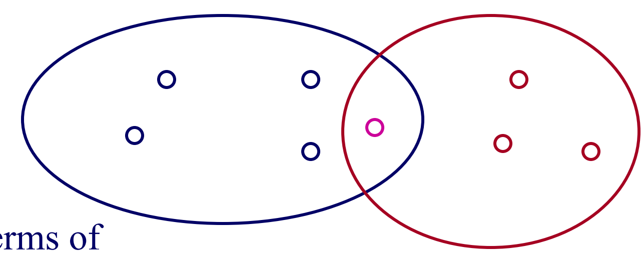
73

Carnegie Mellon

Intuition behind proof

Andrew Tomkins

- Venn diagram



voc. terms of Doc.#1

voc. terms of Doc.#2

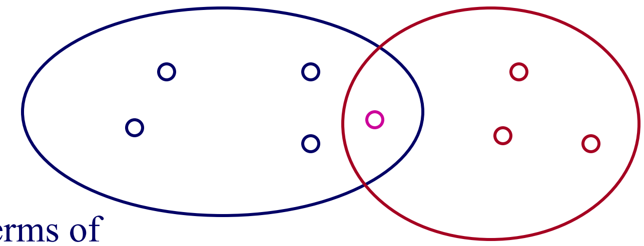
15-826 Copyright (c) 2019 C. Faloutsos 74

74

Carnegie Mellon

Intuition behind proof

- Venn diagram

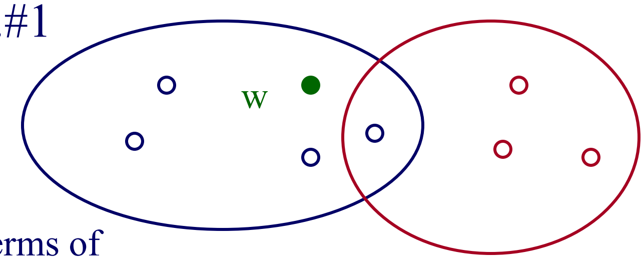


75

Carnegie Mellon

Intuition behind proof

- Venn diagram - let w be the voc. word with the overal smallest hash value, for h.f.#1

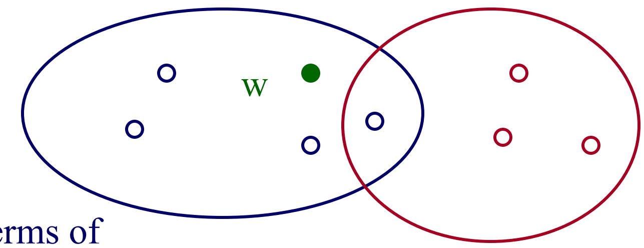


76

Carnegie Mellon

Intuition behind proof

- Prob. that w is smallest on both is exactly Jaccard: #common / #union



voc. terms of Doc.#1

voc. terms of Doc.#2

Carnegie Mellon

Conclusions

- Approximations can achieve the impossible!
- MF and ANF for neighborhood function
- hot-lists
- Jaccard coeff. / ‘similar pages’

Carnegie Mellon

References

E. Cohen. Size-estimation framework with applications to transitive closure and reachability. *Journal of Computer and System Sciences*, 55(3):441-453, December 1997. http://www.research.att.com/~edith/Papers/tcest.ps.Z

Phillip B. Gibbons, Yossi Matias, *New sampling-based summary statistics for improving approximate query answers*, ACM SIGMOD, 1998 Seattle, Washington, pp 331 - 342

Carnegie Mellon

References (cont' d)

Aristides Gionis, Dimitrios Gunopulos, Nikos Koudas, *Efficient and Tunable Similar Set Retrieval,* ACM SIGMOD 2001, Santa Barbara, California

M. Faloutsos, P. Faloutsos, and C. Faloutsos. *On power-law relationships for the internet topology*. SIGCOMM, 1999.

Carnegie Mellon

References (cont'd)

P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31:182-209, 1985.

C. R. Palmer, P. B. Gibbons and C. Faloutsos. *Fast approximation of the "neighborhood" function for massive graphs*. KDD 2002

Carnegie Mellon

References (cont'd)

C. R. Palmer, G. Siganos, M. Faloutsos, P. B. Gibbons and C. Faloutsos. *The connectivity and fault-tolerance of the internet topology*. NRDM 2001.

