Data stream statistics

Filippo Geraci, CNR, Pisa

A set of problems on twitter data

1. How many different users accessed the server this week?
2. Was john among them?
3. How many times john accessed the server?
4. What is the usage trend on the server?
5. Who are the most active users on this server?

Cardinality Estimation

- Easy when I want to count all
- Counting the distinct elements of a stream:
  - Sort data and find unique keys
  - Use hash tables

- Sorting takes O(n log n) time
- Both require O(n) space

Cardinality Estimation: Linear Counting

```
1  class LinearCounter {
2      BitSet mask = new BitSet(m) // m is a design parameter
3
4      void add(value) {
5          int position = hash(value) // map the value to the range 0..m
6          mask.set(position) // sets a bit in the mask to 1
7      }
8  }
```

- Estimation of c' can be adjusted according to the the number n of bits and the number c of bits set to 1

$$c' = -m \, ln \frac{m - c}{m}$$

How big should be the bit vector?

Number of elements in the stream	Size for an error rate of 1%
100	5034
1000	5329
7000	7132
8000	7412
10000	7960
100000	26729
1000000	154171
10000000	1096582
100000000	8571013

- http://dblab.kaist.ac.kr/Publication/pdf/ACM90_TODS_v15n2.pdf

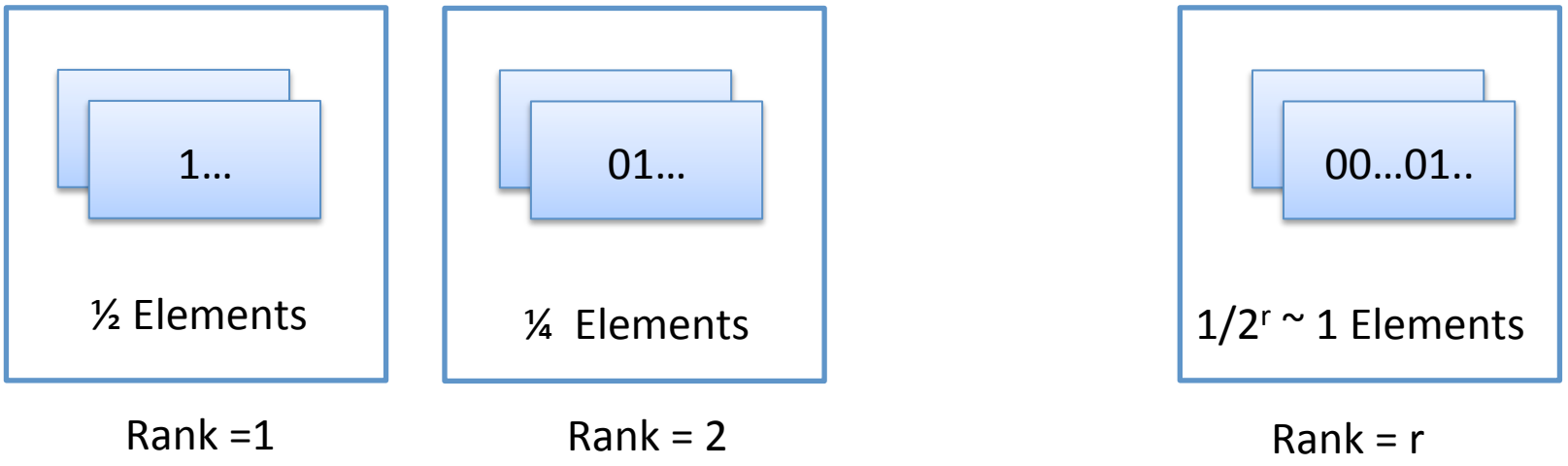
Cardinality Estimation: Linear Counting – Complex queries

- Case study: I have tweets tagged with country and language
  - Question: how many tweets from Italy are in English?

- I can keep two counters one for country and one for language
  - Answer: OR of the two counters

Loglog counters

- Assuming each element is hashed as a H bit vector
  - Let ρ(y) the rank (i.e. the position of the leftmost bit set to 1) of the hash of the element y



Loglog counters

- Given a hash function where the bits are uniformly distributed we can estimate that:

$$|X = \{y' : \rho(y') = r\}| = \frac{1}{2^r} \cong 1$$

Imply

$$\max \rho\,(y) = log_2 n$$

thus

$$n = 2^{\max \rho\,(y)}$$

Loglog counters

```
1  class LogLogCounter {
2     int H            // H is a design parameter
3     int m = 2^k          // k is a design parameter
4     etype[] estimators = new etype[m] // etype is a design parameter
5
6     void add(value) {
7         hashedValue = hash(value)
8         bucket = getBits(hashedValue, 0, k)
9         estimators[bucket] =
10            max (estimators[bucket], rank( getBits(hashedValue, k, H) );
11        )
12    }
13    int count (void) {
14        int sum = 0;
15        for (i=0; i < m; i ++) sum += estimators[i];
16        return m * 2 ^ (1/m * sum);
17    }
18 }
```

Loglog counters - performance

- Given m=256 (k=8) H=16 -> max rank () stored in 4 bits
  - The data structure is 256 * 4bit = 128 bytes
  - Count the number of distinct words in Shakespeare's writings with an error rate of 9.4%
  - 30,897 instead of 28,239
- The HyperLogLog algorithm can count > 10^9 elements using 1.5kB of memory with error rate less than 2%

Resources

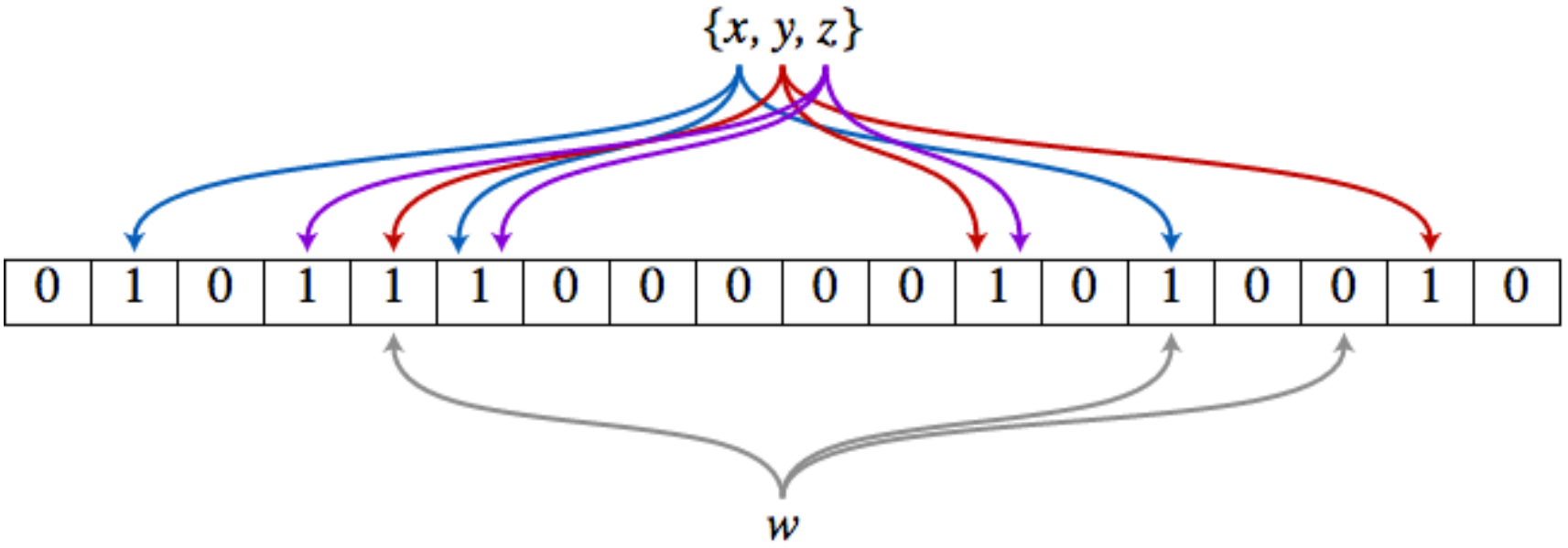
- Python Imlementation of Loglogcounters
  - https://github.com/svpcom/hyperloglog
- Original work:
  - http://algo.inria.fr/flajolet/Publications/DuFl03-LNCS.pdf
- Several references can be found in the Wikipedia article
  - https://en.wikipedia.org/wiki/HyperLogLog

A step further

- Now I know how many different elements in a multiset.
- I want to know if an element belongs to the set

- Bloom filters answer:
  - I strongly think the element is in set
  - Definitely not in set

Bloom filters



The bloom filter version of the spell checker

```
1  from pybloom import BloomFilter
2  import sys
3
4  bf = BloomFilter(capacity=466544, error_rate=0.01)
5
6  f = open ("english.txt")
7  for line in f:
8      line = line[:-1]
9      bf.add (line)
10 f.close ()
11
12 f = open (sys.argv[1])
13 for line in f:
14     line = line[:-1]
15     line = line.split (" ")
16     for elem in line:
17         if elem in bf:
18             print elem, "True"
19         else:
20             print elem, "False"
21 f.close ()
22
```

Number of words in the dictionary

Practical usage

- A python implementation:
  - https://github.com/jaybaird/python-bloomfilter
- Two parameters:
  - Capacity (i.e. expected number of elements to insert)
  - Error rate (> 0, < 1)
- Compare speed versus space of bloom filters and hash sets