Analysis of Bloom Filters

Matteo Dell'Amico

Distributed Computing

Outline

1. Probability of False Positives

2. Choosing m and k

Outline

Probability of False Positives (1)

- Assume hash functions select array positions with equal probability
- We have m bits, k hash functions and n elements in our Bloom filter

Probability of False Positives (1)

- Assume hash functions select array positions with equal probability
- We have m bits, k hash functions and n elements in our Bloom filter

- The probability that a bit is not set to 1 by a single hash function for a single element is

$$1 - \frac{1}{m}$$

Probability of False Positives (1)

- Assume hash functions select array positions with equal probability
- We have m bits, k hash functions and n elements in our Bloom filter

- The probability that a bit is not set to 1 by a single hash function for a single element is

$$1 - \frac{1}{m}$$

- For the k hashing functions, assuming they're independent the probability that a bit is not set to 1 for each of them is

$$\left(1 - \frac{1}{m}\right)^k$$

Probability of False Positives (2)

- After inserting n elements, a bit is still set to 0 with probability

$$\left(1 - \frac{1}{m}\right)^{kn}$$

so, it is 1 with probability

$$1 - \left(1 - \frac{1}{m}\right)^{kn}$$

Probability of False Positives (2)

- After inserting n elements, a bit is still set to 0 with probability

$$\left(1 - \frac{1}{m}\right)^{kn}$$

so, it is 1 with probability

$$1 - \left(1 - \frac{1}{m}\right)^{kn}$$

- We get a false positive when k random bits are set to 1, hence with probability

$$p_{err} = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^{k}$$

Probability of False Positives (3)

- We can use the fact that

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m = \frac{1}{e}$$

to get, for large m values

$$p_{err} = \left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k = \left(1-\left(\left(1-\frac{1}{m}\right)^m\right)^{kn/m}\right)^k \approx \left(1-\left(\frac{1}{e}\right)^{kn/m}\right)^k$$

$$p_{err} \approx \left(1-e^{-kn/m}\right)^k .$$

Probability of False Positives (3)

- We can use the fact that

$$\lim_{m\to\infty}\left(1-\frac{1}{m}\right)^m=\frac{1}{e}$$

to get, for large m values

$$p_{err}=\left(1-\left(1-\frac{1}{m}\right)^{kn}\right)^k=\left(1-\left(\left(1-\frac{1}{m}\right)^m\right)^{kn/m}\right)^k\approx\left(1-\left(\frac{1}{e}\right)^{kn/m}\right)^k$$

$$p_{err}\approx\left(1-e^{-kn/m}\right)^k.$$

- In real-world cases, the approximation is good and this is the value we'll be using.

Outline

Optimal Number of Hash Functions

- To minimize p_{err}, the optimal number of hash functions (derivation) is

$$k = \frac{m}{n} \ln 2$$

(we disregard the fact that that value shouldn't be an integer)

Optimal Number of Hash Functions

- To minimize p_{err}, the optimal number of hash functions (derivation) is

$$k = \frac{m}{n} \ln 2$$

(we disregard the fact that that value shouldn't be an integer)

- Note that with $k = \frac{m}{n} \ln 2$, the probability that a given bit is 0 is

$$\left(1 - \frac{1}{m}\right)^{kn} \simeq e^{-kn/m} = e^{-\ln 2} = \frac{1}{e^{\ln 2}} = \frac{1}{2}$$

Optimal Number of Hash Functions

- To minimize p_{err}, the optimal number of hash functions (derivation) is

$$k = \frac{m}{n} \ln 2$$

(we disregard the fact that that value shouldn't be an integer)

- Note that with $k = \frac{m}{n} \ln 2$, the probability that a given bit is 0 is

$$\left(1 - \frac{1}{m}\right)^{kn} \simeq e^{-kn/m} = e^{-\ln 2} = \frac{1}{e^{\ln 2}} = \frac{1}{2}$$

- This means that a Bloom filter is most efficient when half of the bits are 0s and half are 1s
- Intuitively, it makes sense: the data structure is carrying as much information as possible!

How Big Given an Error Rate?

- With optimal k, the false positive rate is

$$p_{err} \approx \left(1 - e^{-kn/m}\right)^k = \left(1 - e^{-\ln 2}\right)^{\frac{m}{n}\ln 2} = \frac{1}{2}^{\frac{m}{n}\ln 2} = \left(e^{-\ln 2}\right)^{\frac{m}{n}\ln 2} = e^{-\frac{m}{n}(\ln 2)^2}$$

How Big Given an Error Rate?

- With optimal k, the false positive rate is

$$p_{err} \approx \left(1 - e^{-kn/m}\right)^k = \left(1 - e^{-\ln 2}\right)^{\frac{m}{n}\ln 2} = \frac{1}{2}^{\frac{m}{n}\ln 2} = \left(e^{-\ln 2}\right)^{\frac{m}{n}\ln 2} = e^{-\frac{m}{n}(\ln 2)^2}$$

- If we fix $p_{err} = \epsilon$, we get

$$\ln \epsilon = -\frac{m}{n}(\ln 2)^2 ,$$

hence

$$m = -\frac{n \ln \epsilon}{(\ln 2)^2}$$

Bits Per Item

- The optimal number of bits per item is

$$\frac{m}{n} = -\frac{\ln \epsilon}{(\ln 2)^2} \approx -2.08 \ln \epsilon$$

- Let's change the base:

$$ln\epsilon = \frac{\log_{10} \epsilon}{\log_{10} e} \approx 2.30 \log_{10} \epsilon,$$

hence

$$\frac{m}{n} \approx -4.79 \log_{10} \epsilon$$

Bits Per Item: Interpretation

$$\frac{m}{n} \approx -4.79 \log_{10} \epsilon$$

- For an error rate of 10%, $\epsilon = 0.1$ and $\log_{10} \epsilon = -1$, so we have 4.79 bits per item—less than one byte per item

Bits Per Item: Interpretation

$$\frac{m}{n} \approx -4.79 \log_{10} \epsilon$$

- For an error rate of 10%, $\epsilon = 0.1$ and $\log_{10} \epsilon = -1$, so we have 4.79 bits per item—less than one byte per item

- For 1%, $\epsilon = 0.01$ and $\log_{10} \epsilon = -2$, so we have 9.59 bits per item: a bit more than 1 byte per item
- For 0.1%, $\epsilon = 0.001$ and $\log_{10} \epsilon = -3$, we have 14.38 bits per item: less than 2 bytes per item
- Every additional 0 in ϵ only adds 5 bits per item…

Bits Per Item: Interpretation

$$\frac{m}{n} \approx -4.79 \log_{10} \epsilon$$

- For an error rate of 10%, $\epsilon = 0.1$ and $\log_{10} \epsilon = -1$, so we have 4.79 bits per item—less than one byte per item

- For 1%, $\epsilon = 0.01$ and $\log_{10} \epsilon = -2$, so we have 9.59 bits per item: a bit more than 1 byte per item
- For 0.1%, $\epsilon = 0.001$ and $\log_{10} \epsilon = -3$, we have 14.38 bits per item: less than 2 bytes per item
- Every additional 0 in ϵ only adds 5 bits per item…

- If you're interested in even more, look up cuckoo filters :)