algoritme

cascadetrap met $C_1, C_2, ..., C_n$

$C_1 \Rightarrow BF_1(C_2, C_3, ..., C_n) \Rightarrow C'_1$

$C_2 \Rightarrow BF_2(C'_1, C_3, C_3, ..., C_n) \Rightarrow C'_2$

$C_3 \Rightarrow BF_3(C'_1, C'_2, C_4, C_5, ..., C_n) \Rightarrow C'_3$

...

$C_n \Rightarrow BF_n(C'_1, C'_2, ..., C'_{n-1}) \Rightarrow C'_n$

Bij BF_i als het antwoord "neen" is, dan weten we dat het sowieso niet in de bijhorende C_i zit.

Zo hebben we al veel elementen kunnen uitsluiten.

Dan de volgende cascade doen we met $C'_1, C'_2, ..., C'_n$

bits en hashfuncties

hashfuncties optimaal: $k = \frac{n}{m} * \ln 2$

kans op false positives: $(1 - e^{\frac{-km}{n}})^k$

kans bij optimale k: $(1 - e^{-\ln 2})^{\frac{n}{m} * \ln 2} = (1 - \frac{1}{2})^{\frac{n}{m} * \ln 2} = (\frac{1}{2})^{\frac{n}{m} * \ln 2}$

hoeveel moet n i.f.v. met m zijn om de kans $\leq x$ te maken?

$$(\frac{1}{2})^{\frac{n}{m} * \ln 2} \leq x \Leftrightarrow 2^{\frac{n}{m} * \ln 2} \geq \frac{1}{x} \Leftrightarrow \frac{n}{m} * \ln 2 \geq \ln(\frac{1}{x}) \Leftrightarrow n \geq -m * \frac{\ln x}{\ln 2}$$

neem voor x telkens een negatieve macht van 2: $x = 2^{-p}$ met $p = 1, 2, 3, ...$

dan wordt onze formule:

$n \geq m * p$

Dus als we kans op false positives 1/32 willen en m = 1 MB (1 000 000 B) dan hebben we:

$n \geq 5$ MB

en bij $n = 5$ MB hebben we:

$k = \lfloor 5 * \ln 2 \rfloor = \lfloor 3.465... \rfloor = 3$

deze p zullen we meegeven aan de bloomfilter om de n mee te berekenen

met benchmarks kunnen we dan achteraf een goede heuristiek voor p vinden voor mijn algoritme

bestandsformaat cascade

vooraf: - aantal categoriëen (32 bits) = aantal bloomfilters per trap - namen van categoriëen in juiste volgorde (8 bits lengte + 8 bits * len)

elke cascade trap: - /

elke bloomfilter binnen de trap: - aantal hashfuncties in bloomfilter (8 bits) - seeds voor hashfuncties (8 bits seeds * aantal hashfuncties) - aantal bits in bloomfilter / 8 (32 bits) - bloomfilter bits (veelvoud van 8)

indien de categorie horende bij een bloomfilter leeg is, zet je het aantal hashfuncties gewoon 0 en ga je verder, dit is beter dan elke keer alle nieuwe lege categoriëen aan te kondigen of nutteloze bloomfilters aan te maken

- einde van bloomfilters aankondigen met 8 x 1-bits, aangezien 8 x 0-bits een lege categorie betekent
- dan eindigen met de laatste niet lege categorie naam (8 bits lengte + 8 bits * len) (indien in dezelfde trap alle niet-lege categoriëen simultaan leeg geworden zijn, dan is de lengte 0)

limitaties

Voor de categorie namen is er een limitatie van 256 chars bij zowel `train` als `classify`

Indien een ingegeven element bij `classify` niet in de volledige verzameling zit en niet alle categoriëen op hetzelfde moment leeg zijn geworden bij het bouwen van de cascade, dan zal deze geclassificeerd worden als de laatste niet-lege categorie na het bouwen van de cascade. Dus op deze manier wel false positives, maar ja ge moet maar geen elementen ingeven die niet in de volledige verzameling zitten.

Voor de nummers (aantal elementen, aantal bits ...) is er een limitatie van 32 bits (0 - 4,294,967,295)

Bv. het aantal elementen in `large.txt` is 25,000,000 en pas vanaf we een $p \geq 172$ kiezen zal het aantal bits daar groter zijn dan de limitatie, maar dit is een onrealistische situatie, in quasi alle gevallen zal $p \leq 10$ zijn en dan kunnen we 429,496,729 elementen hebben, dus deze limiet is voldoende.

Ik kan het verhogen naar 64 bit unsigned integers, maar dat zal het aantal geheugen onnodeloos vergroten, aangezien de laatste 32 bits quasi nooit gebruikt zullen worden en we veel zo'n nummers zullen gebruiken.

Ook zal ik daardoor een andere hash functie moeten vinden, aangezien de implementatie van murmur has die ik op het internet vond 32 bit unsigned integers retourneert.