Mining Data Streams

Motivacija

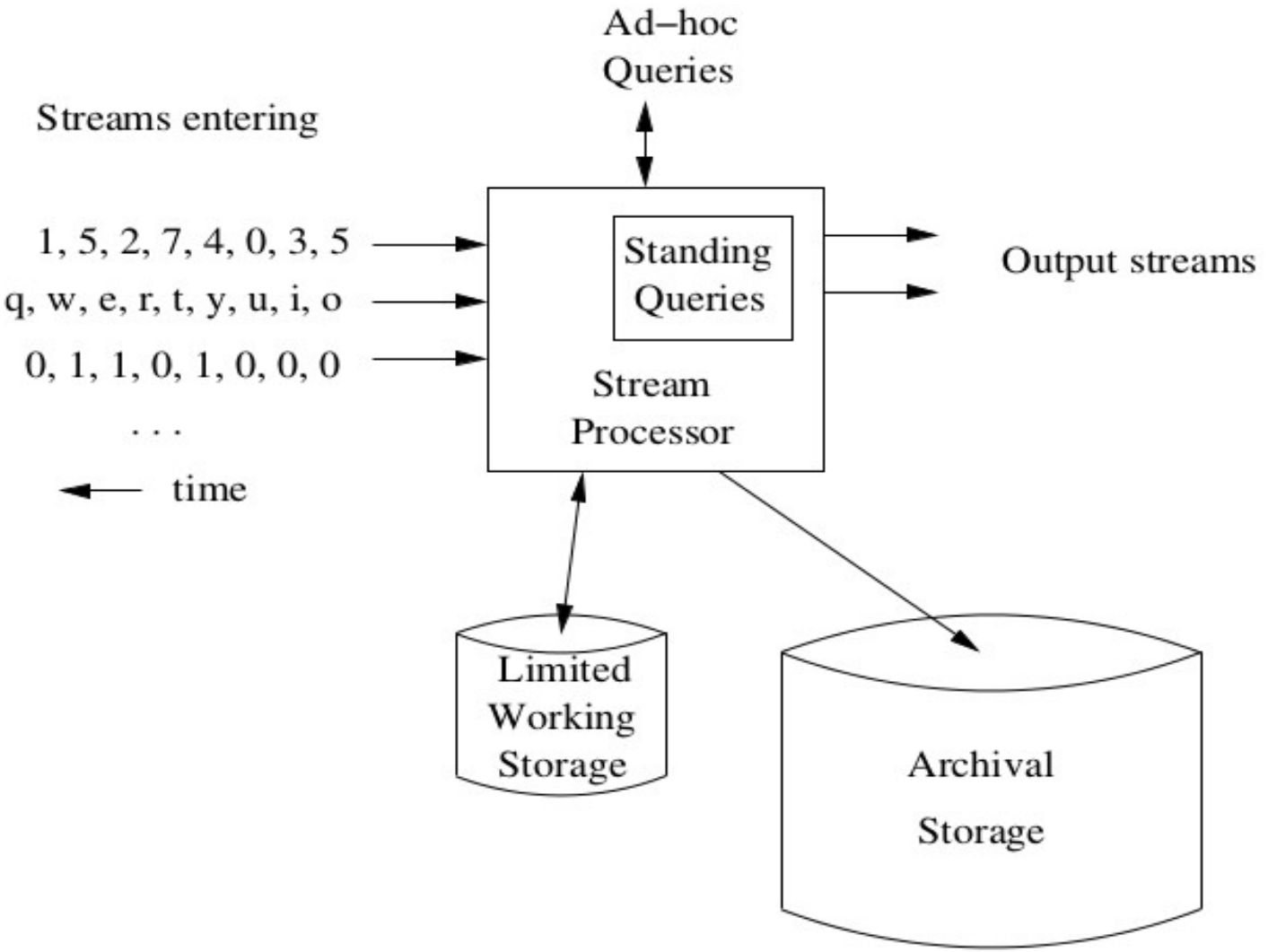
- Podaci nijesu unaprijed poznati i smješteni u bazu podataka, već “dolaze” kao “tok” ili više “tokova”
  - Ukoliko se ne procesiraju direktno podaci se gube
  - Brzina generisanja podataka je prevelika da be se oni upisivali u tradicionalne baze podataka
- Sumarizacija toka podataka
  - Kreiranje uzorka uz filtriranje da bi se eliminisali “nepotrebni” elementi, procjena broja različitih elemenata u toku
  - Formiranje “prozora” sa n posljednjih elemenata, za veliko n, “rudarenje prozora”, procjena broja 1-ica u toku bitova (bit stream)

Stream queries

- Standing queries – unaprijed definisani, sačuvani, izvršavaju se konstantno i generišu izlaz u određenim vremenskim intervalima
  - Prikazati upozorenje kada element iz toka pređe zadatu vrijednost
  - Prikazati prosječnu vrijednost posljednih n elemenata ili prosječnu vrijednost za čitavi tok
  - Prikazati najveći element
- Ad-hoc queries – proizvoljni upit vezan za trenutno stanje toka
  - Sliding window – posljednih n elemenata ili svi elementi pristigli u posljednjih t sekundi, paralela sa relacijom
  - Broj različitih korisnika nekog sajta u posljednjih mjesec dana, relacija Logins(name, time)

```
SELECT COUNT(DISTINCT(name))
FROM Logins
WHERE time >= t
```

Data Stream Management System



Data Stream Management System (2)

- Stream processor – data management system
  - Proizvoljan broj tokova dolazi u sistem, različiti po tipu i/ili učestalosti elemenata, u opštem slučaju ni u okviru istog toka elementi ne stižu konstantnom frekvencijom
- Archival storage – skladište velikog kapaciteta, ali nepovoljno za direktno izvršavanje upita nad tokom podataka (time consuming retrieval process)

Data Stream Management System (3)

- Limited Working Storage – RAM ili HD, namijenjeno za smještanje samo djelova toka podataka i izvršavanje upita
- Primjeri tokova podataka
  - Podaci sa senzora, ako senzor šalje realni broj (4 Bajta) na svaku milisekundu u toku jednog dana generiše se 3.5MB podataka, ako je potrebno milion senzora generiše se 3.5TB za 24 sata
  - Snimci sa satelita

Specifičnosti stream mining-a

- Elementi obično pristižu velikom brzinom ili je veliki broj tokova koji se moraju paralelno obraditi, procesiranje u realnom vremenu, dizajnirati algoritme tako da rijetko koriste podatke sa spoljašnjih memorija
- Rješenje problema kapaciteta RAM memorije
  - Aproksimativni algoritmi
  - Kreiranje uzorka

Uzorkovanje - sampling

- Uzorak sadrži podskup elemenata iz toka podataka, napraviti uzorak tako da je rezultat upita nad uzorkom približan rezultatu nad čitavim tokom
- Primjer, Internet pretraživač za svakog korisnika obrađuje postavljene upite pretrage, tok se sastoji od trojki (user, query, time), traži se procenat ponovljenih upita u posljednjih mjesec dana, moguće je sačuvati samo 1/10 elemenata toka u RAM memoriji

Uzorkovanje – sampling (2)

- Algoritam za formiranje uzorka: generisati slučajan broj iz segmenta 0-9, ako je broj 0 sačuvati element, prema zakonu velikih brojeva očekujemo da za većinu korisnika procenat sačuvanih upita bude približno 1/10
- Drugi algoritam: izabrati 1/10 korisnika, pa sačuvati sve njihove upite
- Koji je algoritam ispravan?

Opšti problem kreiranja uzorka

- Neka se tok podataka sastoji od n-torki, neka je key ključ
- Algoritam za kreiranje uzorka veličine a/b
  - Primijeniti heš funkciju sa b mogućih ishoda nad ključem key
  - Uključiti element u uzorak ako je dobijena vrijednost manja od a
- Prilagođavanje veličine uzorka, obezbijediti da uzorak u svakom trenutku sadrži elemente za koje važi h(Key) < t, inicijalno je t = b, sa povećanjem obima uzorka t se smanjuje sa neko delta

Filtriranje toka podataka

- Selekcija elemenata iz toka koji zadovoljavaju zadati uslov, selektovani elementi opet obrazuju tok
- Različiti kriterijumi za selekciju
  - Prva komponenta treba da je veća od 10
  - Uslov je pripadnost nekom skupu (koji ne može da se u cjelosti učita u RAM memoriju)
  - Bloom filtering

Motivacioni primjer

- Zadat je skup S od milijardu dozvoljenih mejl adresa – adresa za koje se vjeruje da nijesu spam
- Tok podataka sastoji se od parova (mejl adresa, poruka)
- Uobičajeno je mejl adresa > 20B, skup S je smješten na disku

Bloom filtering

- RAM memorija se organizuje kao bit vektor, radi jednostavnost neka je na raspolaganju 1GB operativne memorije = 8 milijardi bita
- Odabrati heš funkciju h koja hešira mejl adrese u 8 milijardi baketa, heširati skup S i postaviti 1 na odgovarajuće bite
  - Kako je $|S|=10^9$ za očekivati je da 1/8 bita bude postavljeno na 1
- Odluka o spamu: heširati tekuću mejl adresu, ako je bit koji se dobije heširanjem postavljen na 1, onda mejl nije spam, inače jeste spam jer smatramo da ne pripada S
- Nedostatak je da će ovaj pristup “propustiti” neke spam poruke, jer se neke poruke koji nijesu u S ipak heširaju u 1

Bloom filtering (2)

- Bloom filter sadrži
  - Niz od n bita, inicijalno svi postavljeni na 0
  - Kolekcija heš funkcija h_1, h_2, …, h_k. Svaka heš funkcija preslikava vrijednost ključa u jedan od n baketa koji odgovaraju prethodnom bit vektoru
  - Skup S sa m ključeva
  - Zadatak bloom filtera je da “propusti” elemente iz toka podataka čiji su ključevi u skupu S

Bloom filtering (3)

- Svaki ključ iz S se hešira sa svakom od k heš funkcija
- Postaviti 1 za svaki bit $1 \leq j \leq n$ za koji postoji K iz S tako da je $h_i(K)=j$ za neko $1 \leq i \leq k$
- Za testiranje na spam nove mejl adrese K potrebno je izračunati $h_1(K)$, …, $h_k(K)$, ako su svi 1 onda mejl nije spam, inače jeste

Analiza za bloom filtering

- Ako je element iz S onda sigurno prolazi Bloom filter
- Ako nije iz S postoji mogućnost da bude false positive, procjena vjerovatnoće false positive odgovara problemu sa x meta koje se ciljaju y puta i pitanju koliko meta će biti pogođeno makar jednom
  - Vjerovatnoća da nećemo pogoditi određenu metu u jednom pokušaju je w=(x-1)/x
  - Vjerovatnoća da nećemo pogoditi određenu metu u svim pokušajima je w^y
  - Koristeći formulu (1 – eps)^(1/eps) = 1/e za malo eps, dobija se w^y=e^(-y/x)
  - U bloom filteringu je x = n, y = m, pa je vjerovatnoća false positive 1 – e^(-m/n), sa k heš funkcija dobija se (1 – e^(-km/n))^k

Prebrojavanje različitih elemenata u toku podataka

- Elementi toka pripadaju univerzalnom skupu U, zadatak je izbrojati koliko se različitih elemenata pojavljuje u toku, računajući od “početka toka” ili određenog trenutka u prošlosti
  - prebrojavanje jedinstvenih korisnika nekog sajta za dati mjesec dana, skup U sadrži sva korisnička imena dodijeljena na tom sajtu, element toka generiše se poslije svakog uspješnog logovanja
  - Umjesto korisničkog imena može da se koristi URL, tehnički postoji 4 milijarde različitih URL

Prvo rješenje

- U RAM memoriji čuvaju se svi elementi toka, koriste se heš tabele ili stabla traženja radi efikasnog unošenja novog elementa i provjere da li element već postoji
- Moguće rješenje samo ako je broj jedinstvenih elemenata ”dovoljno mali”, u suprotnom
  - Više mašina
  - Eksterna memorija

Flajolet-Martin algoritam

- Ideja je da se heširaju elementi univerzalnog skupa u bit-string koji je dovoljno velike dužine, tj. da bude više mogućih rezultata heš funkcije nego što je broj elemenata u univerzalnom skupu
- Kada se primijeni heš funkcija h na element a, bit-string h(a) završava sa određenim brojem 0, tail length, sa R označimo najveći tail length do sada, procjena broja različitih elemenata u toku je 2^R

Estimating moments

- Generalizacija problema prebrojavanja jedinstvenih elemenata
- Pretpostavlja se da je univerzalni skup uređen
- Sa m_i označava se broj pojavljivanja i-tog elementa u toku podataka
- K moment je suma $(m_i)^k$ za svako I
  - 0 moment je broj različitih elemenata u toku
- Direktno rješenje ako možemo da čuvamo broj pojavljivanja svakog elementa u toku podataka

Alon-Matias-Szegedy algoritam

- Računanje 2. momenta uz pretpostavku da je dužina toka n i da ne postoji mogućnost čuvanja svih m_i
- Uz ograničenje memorije dobijamo aproksimativno rješenje, sa više memorije bolja aproksimacija
- Računamo nekoliko promjenljivih X1, …, Xs na sljedeći način
  - Sa X.element označavamo pridruženi element univerzalnog skupa
  - Sa X.value označavamo vrijednost promjenljive. Vrijednost se određuje tako što se na slučajan način bira pozicija u toku između 1 i n, pa se dalje postavlja X.element na nađeni element a X.value na 1. Prilikom čitanja toka svaki put kada se naiđe na X.element inkrementira se X.value
  - Procjena drugog momentuma je n * (2 * X.value - 1) za bilo koje X

Analiza

- Dakazuje se da je očekivana vrijednost za ma koju promjenljivu konstruisanu na opisani način zapravo 2. momentum
- Sa e(i) označava se element na poziciji i, sa c(i) broj pojavljivanja elementa e(i) na pozicijama i, i + 1, ...

$$E(X.value) = \frac{1}{n}\sum_{i-1}^{n} n \times (2 \times c(i) - 1)$$

$$E(X.value) = \sum_{i=1}^{n}\big(2c(i) - 1\big)$$

$$E(X.value) = \sum_{a} 1 + 3 + 5 + \cdots + (2m_a - 1)$$

Domaći

- Procjena momentuma reda > 2
- Procjena momentuma kada dužina toka nije ograničena na n

Prebrojavanje 1

- Dat je binarni tok, posmatramo “prozor” dužine N, potrebno je odgovoriti na upit koliko ima 1 u posljednjih k bita, za svako k <= N
- Exact count, potrebno je čuvati N bita, dokazati

DGIM algoritam

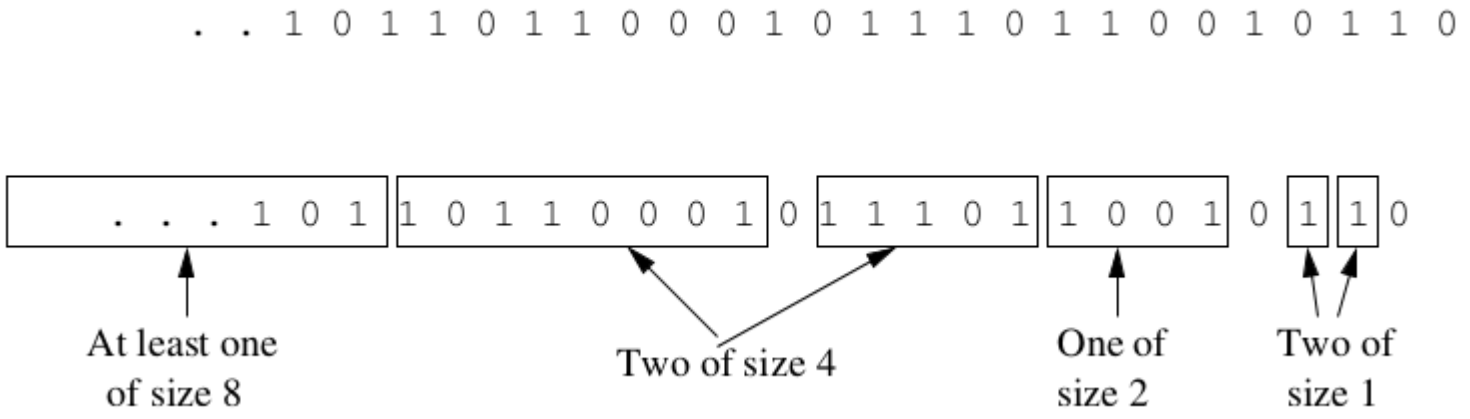
- Koristi se $O(\log^2 N)$ bita za predstavljanje prozora dužine N, procjena broja 1 je sa greškom <= 50%
- Poboljšanje, ograničava se graška na Eps > 0, uz potrošnju $O(\log^2 N)$ bita (ali konstanta raste ako smanjujemo Eps)
- Svaki bit dobija vremensku oznaku (timestamp), prvi bit ima oznaku 1, drugi 2 itd.

DGIM (2)

- Pošto se posmatra prozor dužine N, oznake se računaju po modulu N, pa je dovoljno $\log_2 N$ bita za njihovo predstavljanje
- Tok bitova dijeli se na bakete koji sadrže
  - Vremensku oznaku posljednjeg elementa
  - Broj 1 – veličina baketa, pri čemu taj broj mora biti stepen 2
  - Za predstavljanje baketa potrebno je $\log_2 N$ bita za vremensku oznaku i $\log_2 \log_2 N$ bita za veličinu

DGIM (3)

- Predstavljanje toka pomoću baketa
  - Baket završava sa 1
  - Baketi su disjunktni
  - Postoje najviše 2 baketa jednake veličine
  - Veličine su stepeni 2
  - Idući prema lijevo veličine baketa ne opadaju



Analiza

- Za predstavljanje jednog baketa potrebno je $\log_2 N$ bita, broj baketa je $<= 2*\log_2 N = O(\log_2 N)$, pa je ukupno potrebno $O(\log^2 N)$ bita

Procjena broja 1 u DGIM

- Traži se broj 1 u posljednjih k <= N bita
  - Pronaći baket b sa najmanjom vremenskom oznakom koji uključuje makar jedan od posljednjih k bita
  - Procjena broja 1 je suma veličina svih baketa poslije b + polovina veličine baketa b
- Ako je tačan odgovor c, koliko je DGIM procjena lošija?

Održavanje DGIM uslova za bakete

- Kada algoritam pročita još jedan bit
  - Provjerava se krajnji lijevi baket, ako je njegova oznaka van prozora veličine N, taj se baket briše
  - Ako je novi bit 1, kreira se novi baket sa tekućom vremenskom oznakom i veličinom 1, ako time kreiramo tri baketa veličine 1 onda spajamo dva starija u jedan veličine 2 sa vremenskom oznakom “desnog” od njih dvojice
  - Postupak se moguće nastavlja kombinujući dva baketa veličine 2 u jedan veličine 4 itd.
  - Za obradu novog bita potrebno je O(log N) vremena

Most-Common elements

- Tok podataka sastoji se od karti za bioskop prodatih svuda u svijetu, element toka sadrži ime filma, zadatak je naći najpopularnije filmove “trenutno”
- Da li je film za koji je prodato n ulaznica u svakoj od posljednjih 10 nedjelja popularniji od filma za koji je prodato 2n ulaznica samo u posljednjoj nedjelji?

Direktno rješenje

- Formirati tok bitova za svaki film, sa elementom 1 ako je ulaznica za taj film, inače je element 0
- Formirati prozor veličine N, broj ulaznica na osnovu kojih se procjenjuje popularnost
- DGIM algoritam da prebrojim ulaznice za svaki film, pa filmove sortiramo po broju prodatih ulaznica
- Nepogodno ako je broj filmova preveliki

Decaying window

- Tok podataka sastoji se od a1, a2, …, at, gdje je a1 prvi element, at je tekući element u toku
- Za c <= 10^{-w} decaying window definiše se kao

$$\sum_{i=0}^{t-1} a_{t-i}(1-c)^i$$

Sljedeći element a_{t+1}, obrađuje se u dva koraka

- Pomnoži se tekuća suma sa 1-c
- Sabere se sa a_{t+1}
- Ova suma određuje “popularnost filma”