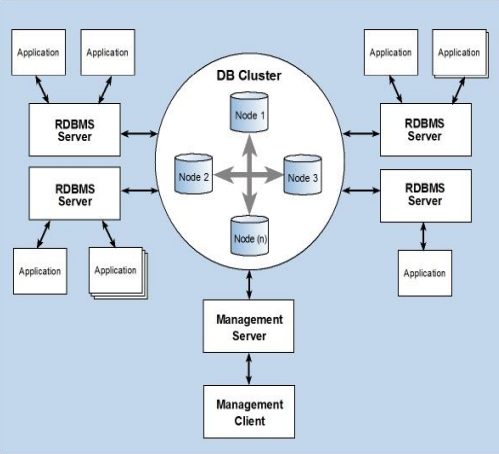
1. **Raspodijeljena pohrana velike količine strukturiranih podataka**

Strukturirani podaci :

* najčešća definicija: podaci koji prate definiranu shemu te čije vrijednosti imaju smisleni smještaj unutar neke strukture
* jednostavnije razmišljati o tabličnim/netabličnim podacima
* alternativno pitanje: da li možemo podacima pristupiti SQL-ovskim jezikom?

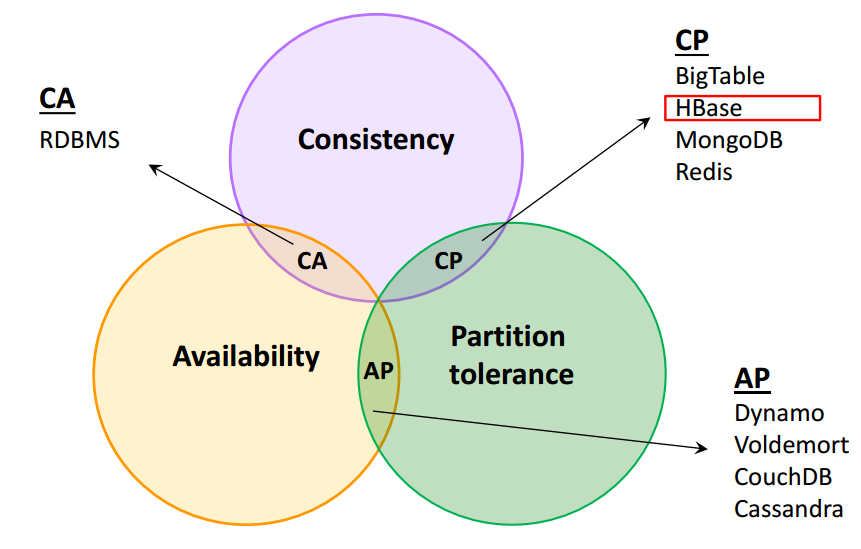
Raspodijeljena pohrana velike količine strukturiranih podataka

* tradicionalno strukturirane podatke pohranjujemo u relacijskim bazama podataka (RDBMS)
* relacijske baze podataka također mogu imati raspodijeljenu arhitekturu; klijent ne zna da je raspodijeljena jer ima karakteristike klasične relacijske baze
  + podaci se pohranjuju u centralno upravljanom grozdu



* Dobra svojstva RDBMS-ova
  + Normalizirana pohrana podataka – uklanja nepotrebnu redundanciju
  + Stroga integritetska pravila i transakcijska obrada (ACID) – visoka konzistentnost podataka
  + Rigidna struktura i apstrakcija fizičke i logičke razine omogućuje lako upravljanje podacima uz pomoć formalnog upitnog jezika (SQL)
  + Pohranjene procedure omogućuju dodatnu apstrakciju, rasterećenu aplikacijsku logiku, smanjuju mrežni promet
  + Sekundarni indeksi povećavaju učinkovitost izvođenja upita
* Mogući problemi RDBMS-ova kod velike količine podataka
  + Spajanje tablica postaje resursno zahtjevno – denormalizacija čime se gubi 3NF
  + RDBMS-ovi nisu dizajnirani za horizontalnu skalabilnost (moguće uz sharding, ali jako komplicirano)
  + Očuvanje konzistentnosti postaje usko grlo – stroga politika zaključavanja i kontrole pristupa
  + Sekundarni indeksi usporavaju sustav kod čestih izmjena

Cap teorem:

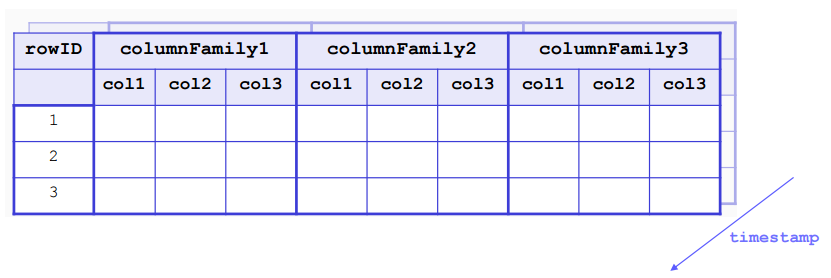


Stupčano orijentirane baze podataka (jedna n-torka je jedan stupac baze, a ne redak):

* Analize zasnovane na sadržaju pojedinih stupaca puno učinkovitije
  + Smanjuje se broj I/O operacija
  + Bolja kompresija podataka (stupci sadrže slične podatke)
* Pogodno za tablice s velikim broje stupaca (široke tablice)
* Učinkovito upravljanje rijetko popunjenim tablicama
  + Row-oriented: moraju pohraniti NULL
  + Column-oriented: preskaču prazne vrijednosti

Apache HBase:

* Apache HBase je distribuirana, stupčano-orijentirana baza podataka (column-family oriented) predviđena za polustrukturirane i strukturirane podatke te dizajnirana za horizontalnu skalabilnost
* Što nam omogućuje HBase a da već nije omogućeno HDFS-om?
  + Pristup individualnim zapisima (HDFS radi u batch modu – čita sve podatke odjednom)
  + Nisku latenciju u radu s podacima (HDFS-u to nije prioritet)
  + Nasumični pristup podacima i indeksiranje (HDFS – samo sekvencijalni pristup)
* Za razliku od klasičnog RDBMS-a, baza HBase:
  + Ne poznaje koncept sheme već se fleksibilno dodaju obitelji stupaca
  + Nema tipova podataka
  + Dizajnirana za horizontalnu skalabilnost
  + Ne radi u transakcijskom modu
  + Pohranjuje podatke u denormaliziranom obliku
  + Ne koristi SQL
* HBase pohrana podataka
  + Tablica = skup redaka
  + Redak = vrijednosti unutar skupa obitelji stupaca
  + Obitelj stupaca = skup stupaca
  + Stupac = skup parova ključ-vrijednost (vrijednosti mogu biti verzinirane)



* Pohrana podataka – HFile
  + Neizmjenjive datoteke – HFile
    - Svaki HFile je uređena mapa parova (K,V)
    - Sortirani po leksikografskom redu identifikatora redaka
    - Na kraju datoteke pohranjuje se indeks blokova pomoću kojeg HBase pronalazi traženi zapis (slično B-stablima)
  + Kod izmjene podataka ne izmjenjuje se HFile već se izmjene prvo pišu u HLog ili WAL (write-ahead log) a potom pohranjuju u memoriju (memstore)
    - Korisniku se poslužuju podaci kao kombinacija trenutnih podataka u memoriji i HFile-u
    - Kada se memstore prepuni stvara se novi HFile uz pomoć WAL-a
  + Minor i major compaction – stapanje manjih datoteka u veće
* HBase – arhitektura sustava
  + Poput HDFS-a, HBase koristi distribuiranu master/slave arhitekturu
  + Poslužitelji: master(upravljački) i region(regionalni) – slično kao HDFS
  + HBase koristi usluge:
    - HDFS-a za fizičku pohranu podataka
    - Zookeeper-a za evidenciju i koordinaciju resursa (u ovom slučaju regionalnih poslužitelja)

HBase i MapReduce:

* HBase se lagano integrira sa MapReduce infrastrukturom nudeći wrapperse koji pretvaraju tablice u ulazne i izlazne podatke za MapReduce
* Map se izvodi za svaku regiju podataka
* Moguće filtriranje redaka/stupaca radi bržih performansi

Rad s HBase konozolom:

* Moguće pisati skripte za provedbu kroz HBase konzolu no za to je potrebno poznavati jezik JRuby u kojem je konzola implementirana

HBase Java Api:

* HBase Java API ima posebno definirane objekte upravo za rad s Hadoop MapReduce-om

1. **Sustavi za preporučivanje objekata u raspodijeljenom okruženju**

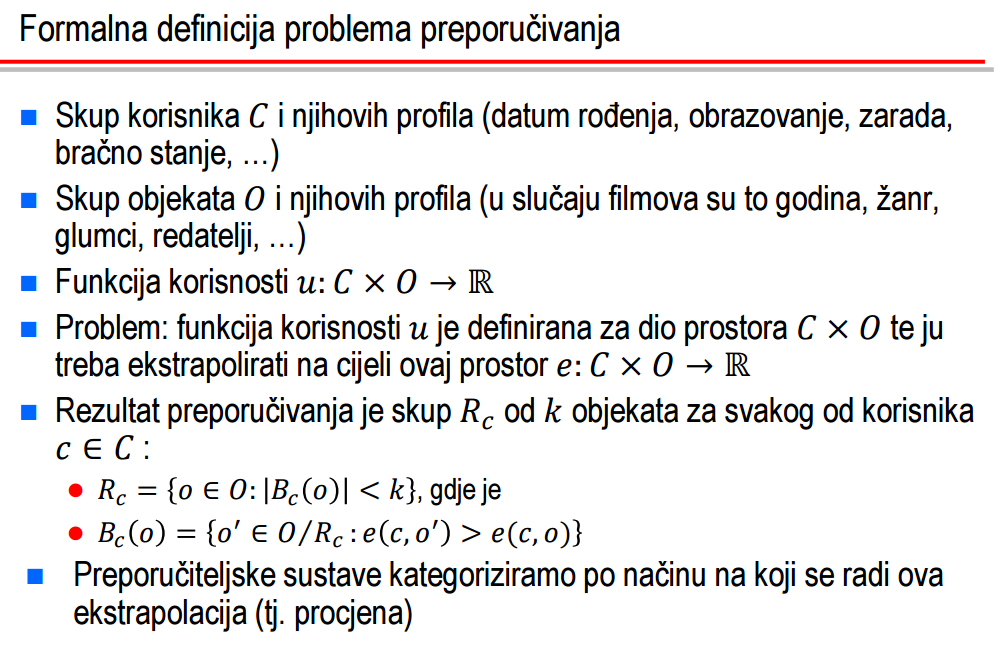
Općenito o preporučivanju objekata:

* Postoje dvije vrste preporučivanja
  + Preporučivanje najpopularnijih objekata bez znanja o specifičnim interesima korisnika
  + Personalizirano preporučivanje objekata na osnovu interesa korisnika
    - Interesi se saznaju preko: ključnih riječi, sličnih pretraživanja, ocjenjivanja sličnih objekata, preko društvenih mreža, poruka e-maila
    - Ovime se mi bavimo
* Preporučivanjem se pokušava unaprijed predvidjeti interes korisnika za njemu nepoznate objekte čime se pokušava zadržati korisnika ili zaraditi od njega (trgovanje podacima o interesima korisnika)
* *Recommender systems are personalized information filtering technology used to either predict whether a particular user will like a particular item (prediction problem) or to identify a set of k items that will be of interest to a certain user (top-k recommendation problem)*
* *The recommendation problem is the problem of estimating ratings for the items that have not been seen by a user*

Zipfova razdioba – modelira odnos frekvencije pojavljivanje veličine koju mjerimo i diskretne veličine poredane po padajućoj frekvenciji pojavljivanja (tj. služi za rankiranje)

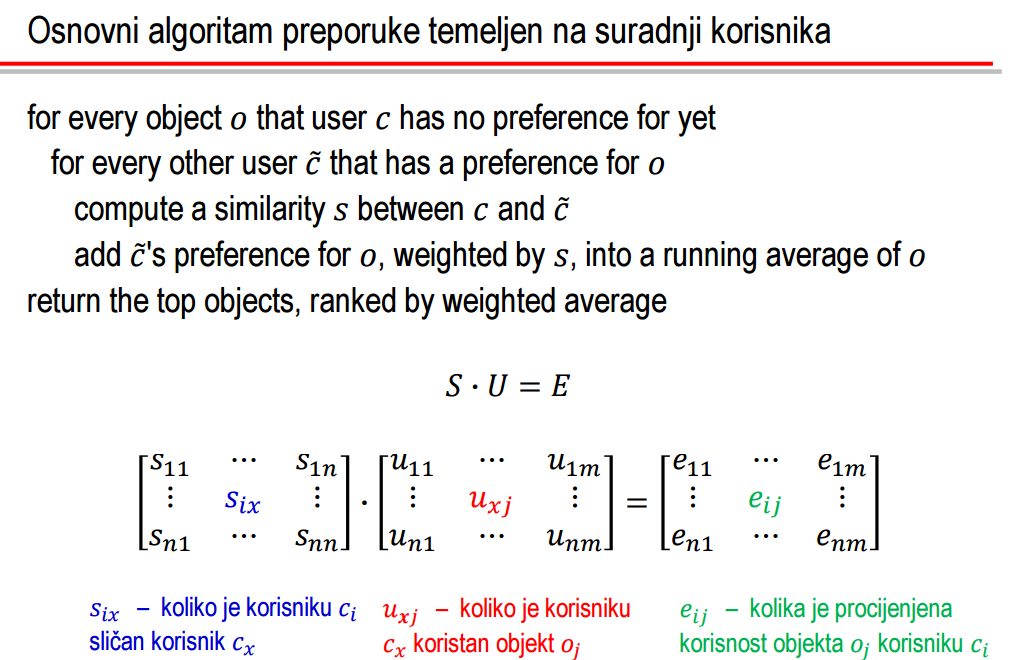


* Što preporučivati? Npr. svi znaju za popularne filmove; treba preporučivati rijetko gledane filmove specifične za interes pojedinog korisnika

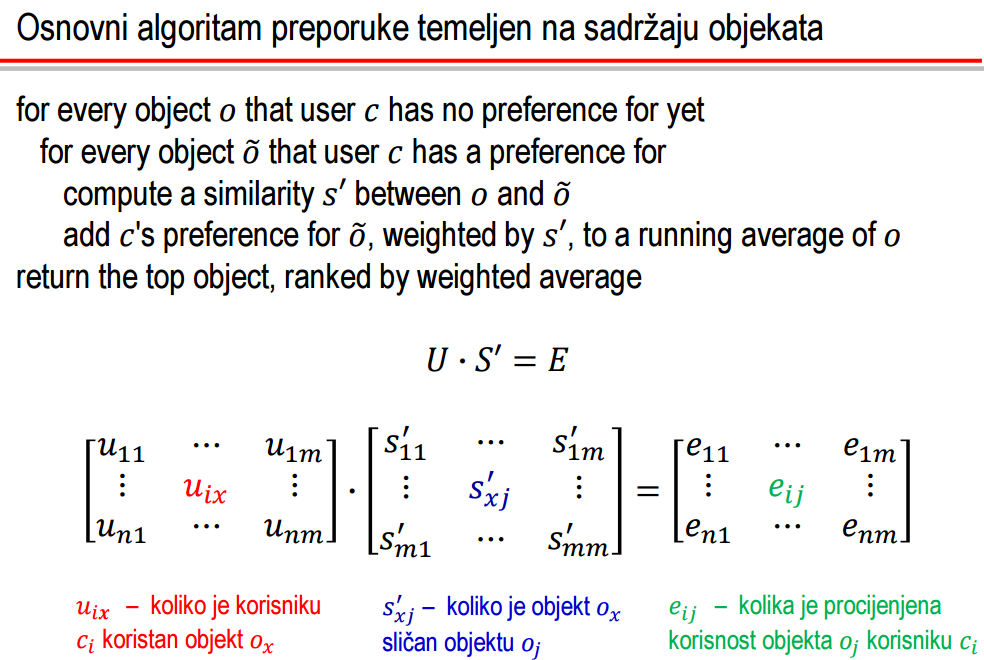


Kategorizacija:

* Preporučiteljski sustavi temeljeni na suradnji korisnika (preporučuju se objekti koji su bili zanimljivi sličnim korisnicima)
  + Procijenjena korisnost 𝑒 𝑐, 𝑜 objekta 𝑜 za korisnika 𝑐 se računa na osnovu njegove korisnosti korisnicima 𝑐 ′ ∈ 𝐶′ koji su slični (tj. sličnih profila) korisniku *c*



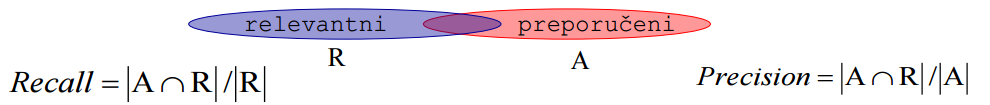
* + Problemi:
    - Performanse algoritma su loše u slučaju velikog broja korisnika
    - Pri izračunu preporuke korisniku ci za svaki objekt oj moramo zbrojiti n umnožaka dva broja pa za sve objekte imamo: m\*(n + n) operacija između 2 broja
    - Za sve korisnike stoga imamo: n\*m\*(n+n) = 2\*m\*n2
    - Rješenje – koristiti samo slične korisnike: 2\*n\*m\* |C'|
    - Problem novog korisnika/novog objekta/kritične mase korisnika
* Preporučiteljski sustavi temeljeni na sadržaju (preporučuju se objekti koji su slični objektima za koje je korisnik prethodno iskazao interes)
  + Procijenjena korisnost 𝑒 𝑐, 𝑜 objekta 𝑜 za korisnika 𝑐 se računa na osnovu korisnosti objekata 𝑢 𝑐, 𝑜 koje je korisnik 𝑐 eksplicitno dao objektima 𝑜 ′ ∈ 𝑂′ koji su slični (tj. sličnih profila) objektu *c*



* + Problemi
    - Performanse algoritma su loše u slučaju velikog broja objekata
    - Pri izračunu preporuke korisniku ci za svaki objekt oj moramo zborjiti m umnožaka dva broja pa za sve objekte imamo m\*(m+m) operacija između 2 broja
    - Za sve korisnike: n\*m\*(m+m) = 2\*n\*m2
    - Rješenje – za svaki objekt koristiti samo njemu l najsličnijih objekata: 2\*n\*m\*l
    - Problem novog korisnika/prevelika specijalizacija/izgradnje profila objekta
  + Preporuka temeljena na sličnosti objekata (item-based recommendation)
    - Algoritam je identičan algoritmu preporuke temeljenom na sadržaju objekata, ali se sličnost objekata određuje na drugi način
    - Mjere za određivanje sličnosti profila korisnika i objekata:
      * Vektorski profili (Euklidksa udaljenost, pearsonova korelacija, log-likelihood…)
      * Tekstualni profili
      * Hibridni profili (za određivanje sličnosti svakog dijela profila koristi se zasebna mjera; ukupna sličnost se dobije kombinacijom)
    - Zašto je potrebna nomralizacija matrica sličnosti?
      * Normalizacija: vrijednosti u intervalu [0, 1] i matrica je simetrična
      * Postiže se da matrica ima smisla:
        + Tako da sličnost A i B itema bude ista kao sličnost B i A itema (simetrija matrice) što ne mora vrijediti općenito u podacima
        + Ukoliko se kombiniraju ili uspoređuju dvije matrice sličnosti, vrijednosti u njima moraju biti usporedive (npr. tablica 1 ima raspon vrijednosti [0, 100] dok tablica 2 ima [0, 10]; ako u tablici 1 neki zapis ima vrijednost 17, a isti taj zapis u tablici 2 ima zapis 3 njihovom usporedbom bi zaključili da je zapis u tablici 1 označava veću sličnost... normalizacijom ćemo uspoređivati 17/100 i 3/10 te zaključiti da zapis u tablici 2 ima veću sličnost)
* Hibridni preporučiteljski sustavi (kombinacija gornja dva)

Evaluacija preporučitelja:

* Puno mogućnosti za izgradnju preporučitelja – prilikom izgradnje produkcijskog preporučitelja je potrebno odabrati najboljeg za točno odgovarajuću primjenu
* Evaluacija nam govori koji je preporučitelj bolji za koje ulazne podatke
  + Usporedba predviđene i stvarne korisnosti (ulazni podaci se razdvajaju na dio za treniranje i dio za testiranje)
  + Usporedba preciznosti i odziva
    - Preciznost: koliko je relevantnih objekata među preporučenim
    - Odziv: koliko je preporučenih među relevantnim



Apache Mahout:

* Skup algoritama strojnog učenja (suradno filtriranje, klasterizacija, klasifikacija)
* Podržava centralizirano i raspodijeljeno izvođenje: Apache Hadoop, Apache Spark…

1. **Raspodijeljena obrada toka podataka programskim okvirom Apache Spark**

Kolekcijski tokovi:

* Kolekcije u Javi 8 možemo promatrati kao tok pohranjenih elemenata
* Tok se prijelaznim (engl. intermediate) operacijama može transformirati u tok nekih drugih elemenata koji će se računati temeljem originalnih elemenata
* Na kraju se tok teminalnim (engl. terminal) operacijama pretvara u kolekciju ili reducira u neki drugi rezultat
* Prijelazne operacije se lijeno izvode – provode se tek po izvođenju terminalne operacije
* Tok ima ulogu cjevovoda, nije spremište podataka – stvaranjem toka puštamo podatke u cjevovod koji smo prethodno složili

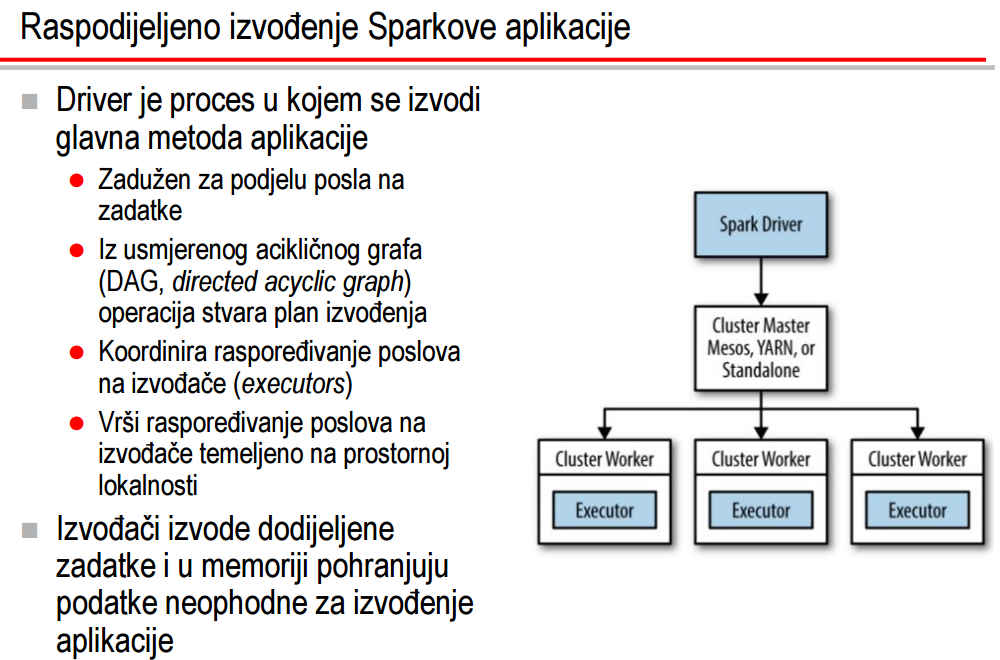
Lambda izrazi:

* Uvedeni u Javi 8
* Lambda izraz je anonimna metoda (tj. funkcija bez klase)
* Oblik funkcijskog programiranja, smanjuje broj linija, povećava čitljivost
* Uvedena mnoga funkcijska sučelja od kojih mnoga imaju samo jednu metodu – idealno za lambda izraze
* Uveden operator :: (double colon) za lako referenciranje metode

Apache Spark:

* Programski okvir za obradu velikih podataka (izvodi se na Hadoopu, samostalno, u oblaku…)
* Za razliku od Hadoopa podržava obradu podataka u memoriji
  + 100 puta brža obrada iz memorije, 10 puta brža obrada s diska
* Sastoji se od 5 djelova:
  + Spark Core - osnovne komponente za raspodjelu poslova i upravljanje resursima
  + Spark SQL – komponenta za rad sa strukturiranim podacima
  + Spark Streaming – komponenta za rad s tokovima podataka
  + Spark MLib – komponenta s algoritmima strojnog učenja
  + Spark GraphX – komponenta za rad s velikim grafovima
* Podržava različite načine pohrane podataka: loklano, HDFS, HBase,..

Raspodijeljeno izvođenje Sparkove aplikacije:



RDD (Resilient Distributed Dataset):

* Osnovna apstrakcija za rad s podacima u Sparku je klasa JavaRDD
* Nastaje učitavanjem vanjskog skupa podataka ili iz postojeće kolekcije objekata u memoriji
* Raspodijeljena kolekcija podataka
  + Nepromjenjiva (immutable, read-only)
  + Particionirana na čvorovima izvođačima
  + Otporna na ispade (resilient) – ponovno se može rekonstruirati nakon ispada čvora
  + Podržava paralelni pristup
* Podržava dvije vrste operacija
  + Transformacije stvaraju novi RDD
  + Akcije konzumiraju RDD
* Kao kolekcijski tok u Javi 8
* Transformacije se lijeno izvode – provode se tek po izvođenju akcije
* Razlika u odnosu na Javu 8: dohvaćanje prvih k elemenata je u Javi prijelazna operacija, dok je u Sparku terminalna

Obrada toka podataka:

* *A data stream is a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items. It is impossible to control the order in which items arrive, nor is it feasible to locally store a stream in its entirety.*
* Potencijalno beskonačan slijed podataka
* Osnovni operatori s karakteristikama: imaju unutarnje stanje i blokirajući su (blokiraju isporuku rezultata)
  + Agregacija (aggregation) – kao rezultat vraća jednu vrijednost (sum, count, max, min, avg,..)
  + Združivanje (join) – kao rezultat vraća novi tok združenih objekata
  + Top-k – kao rezultat vraća najboljih k objekata iz toka podataka
* Prozor
  + Problemi s gornjim operacijama ukoliko se radi s beskonačnim slijedom (npr. koji objekti su top-k u beskonačnom slijedu? Koja je srednja/maksimalna vrijednost beskonačnog slijeda?)
  + Uveden pojam prozora kako bi se osnovni operatori odblokirali
  + Vremenski prozor (npr. svi objekti u zadnjih 20 minuta)
  + Prozor temeljen na broju objekata (npr. zadnjih 10 objekata)
* Brzina obrade
  + Podaci se trebaju obrađivati brže nego što pristižu u sustav
  + Vremenska složenost algoritama je jako bitna
  + Podatkovni objekti se moraju pohranjivati u memoriju
  + Ne smiju se koristiti spore vanjske jedinice (memorijske kartice, čvrsti diskovi)
* Količina memorije
  + Koliko objekata treba pohraniti u memoriji da bi se deterministički odredilo top-k objekata u prozoru veličine n?
    - Min k, max n, k \* ln(n/k) ukoliko su vrijednosti atributa podatkovnog objekta međusobno neovisne
    - Što ako je veličina memorije premala za objekte?

Obrada toka podataka programskim okvirom Apache Spark:

* Komponenta Spark Streaming - rad s tokovima podataka u stvarnom vremenu
* Osnovna apstrakcija za rad s tokovima podatak au Sparku je klasa JavaDstream
  + Tokovi podataka se u Sparku nazivaju diskretizirani tokovi i sastoje se od niza RDD-ova
  + Podržavaju transformacije i izlazne operacije
* Mnogobrojni načini stvaranje ulaznih podatkovnih tokova
  + Transformacija iz jednog toka u drugi
  + Tok podataka s TCP socket-a
  + Tok podataka s HDFS-a
  + Flume, Kafka, Akka actors, MQTT…
* Apliakcije za obradu toka podataka se izvode 24/7

Arhitektura Spark Streaminga

* Stvarnovremenost je privida jer se podaci iz dolaznih tokova podataka obrađuju u mikro-skupinama (micro-batch) RDD-ova
* Nove mikro-skupine RDD-ova za obradu nastaju u regularnim intervalima (batchInterval)
* Rezultati se predaju vanjskim sustavima također u mikro-skupinama RDD-ova
* Vremenski prozori:
  + windowDuration – koliko zadnjih intervala čini prozor
  + slideDuration – koliko često prozor klizi u vremenu
  + oba su višekratnici batchIntervala

1. **Analiza poveznica i velikih mreža**

Problemi koje pokušavamo riješiti u velikim mrežama:

* Pretraživanje grafa
  + Pronalaženje vođe u društvenom grafu
  + Preporuke prijatelja u društvenoj mreži
* Planiranje putova u grafu
  + Usmjeravanje paketa u računalnoj mreži
  + Usmjeravanje vozila
* Grupiranje čvorova
  + Pronalaženje zajednica u društvenoj mreži

Prikaz podataka iz mreža:

* G(V,E): usmjereni ili neusmjereni graf
* V: skup čvorova, predstavljaju objekte
* E: skup grana, označavaju veze među objektima

Povezanost grafa:

* Povezani graf: usmjereni graf u kome postoji put za svaki par čvorova
* Slabo povezani graf: onaj graf koji postaje povezan ako sve njegove usmjerene grane zamijenimo neusmjerenima

Matrica susjedstva (adjacency matrix)

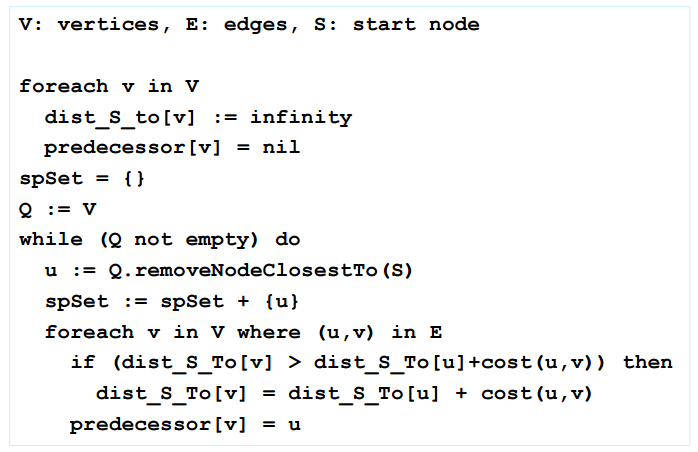
* Graf s n čvorova može se prikazati kvadratnom matricom M
* Element matrice cij > 0 označava granu od čvora ni do nj
* Za grane bez definiranih težina: 1 – grana postoji, 0 – grana ne postoji
* Za grane s definiranim težinama: u matrici su zapisane težine grana
* Neusmjereni graf: koristi gornju trokutastu matricu
* Prednost: pojednostavljuje obradu jer koristimo matematičke operacije nad matricama
* Nedostatak: veličina zapisa matrice 0(n2) – velike mreže su sparse (rijetke)

Lista susjedstva (adjacency list)

* Lista koja je pridružena svakom čvoru, a u kojoj pišu susjedni čvorovi i težine bridova do njih, ako je graf težinski
* Zapisuje samo grane koje postoje
* Zapis neusmjerenih grana:
  + Zapiši svaku granu 2 puta: čvor 2 se pojavljuje u listi susjednosti čvora 1 i čvor 1 se pojavljuje u listi susjednosti čvora 2
  + Ako postoji slijednost čvorova, zapisuje se samo prvi čvor u redoslijedu, a drugi se podrazumijeva: čvor 2 se pojavljuje u listi susjednosti čvora 1 ali čvor 1 se ne pojavljuje u listi susjednosti čvora 2
* Prednost: sažeti zapis
* Nedostatak: izvedba operacija nad grafom je složenija pomoću liste susjedstva
* Pogodnija za zapis velikih mreža:
  + Koristi puno manje memorije

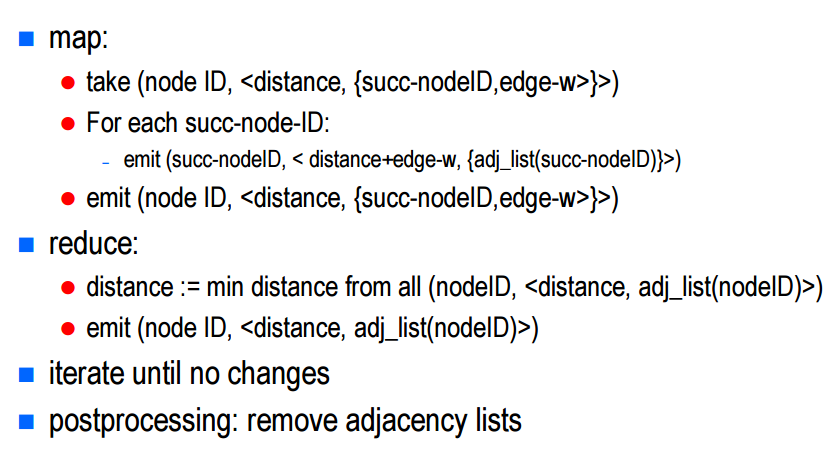
Najkraći putovi od zadanog izvorišta (Dijkstra)

* Za graf G = (V,E) u kome grana e ima težinu w(e) koja predstavlja duljinu grane e potrebno je izračunati najkraće putove od definiranog izvorišnog čvora v do svih ostalih čvorova grafa G



* Kako pralelizirati Dijkstrin algoritam?
  + Teško
  + U jednoj iteraciji MapReduce analiziraju se čvorovi na udaljenosti 1 od izvoršta, zatim na udaljenosti 2… dok rješenje ne konvergira
  + Broj iteracija je malen s obzirom na svojstvo grafova da su svi čvorovi najviše na udaljenosti 6 od izvorišnog čvora (Milgramov eksperiment) ???

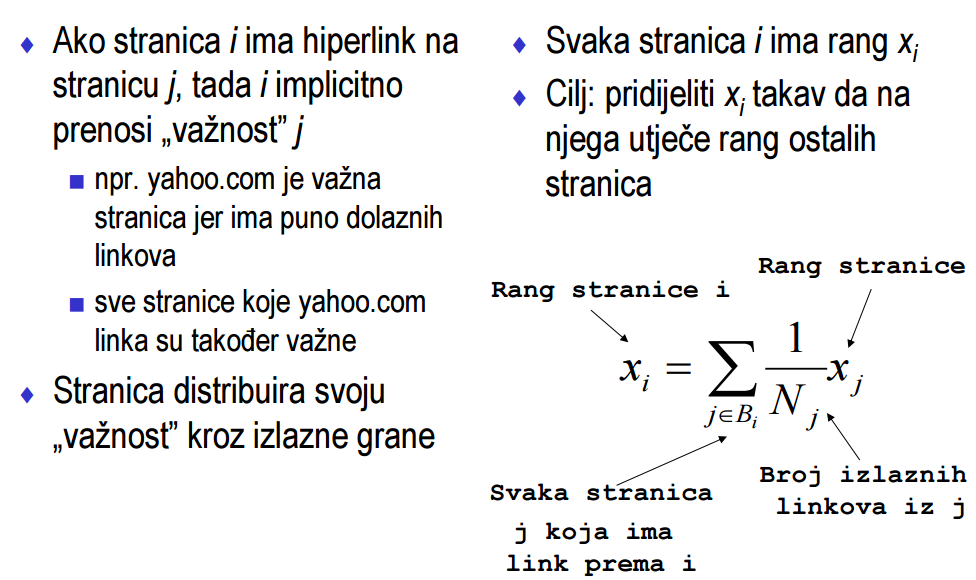
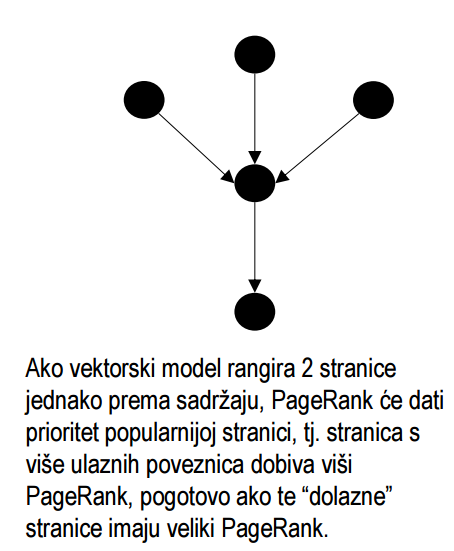
Skica MapReduce algoritma:

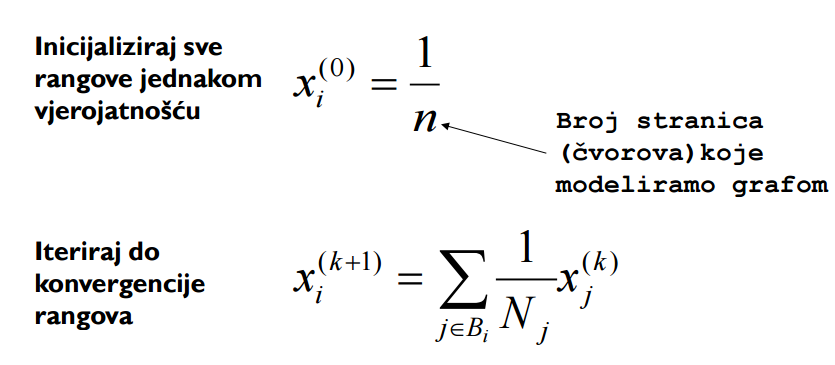


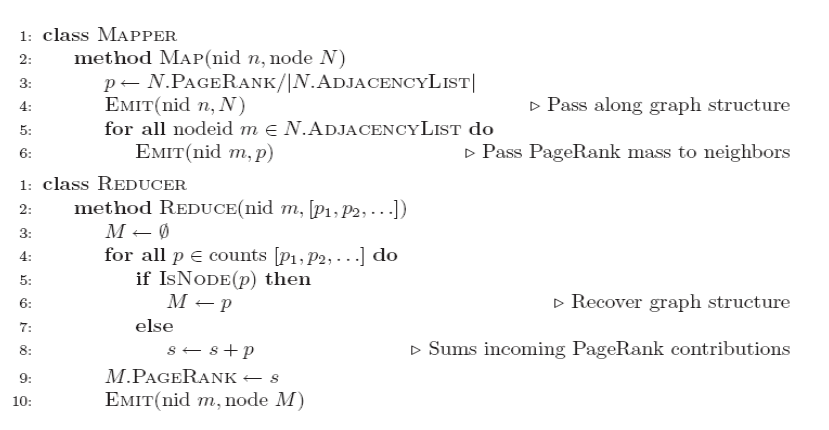
Slučajna šetnja na webu:

* Modeliranje slučajnog istraživanja weba (random surfer model)
  + Korisnik započinje istraživanje weba na slučajno odabranoj stranici
  + Korisnik na slučajan način odabire link na stranici
  + Postoji vjerojatnost da korisnik slučajno odabere bilo koju stranicu na webu umjesto da slijedi link na stranici na kojoj se trenutno nalazi

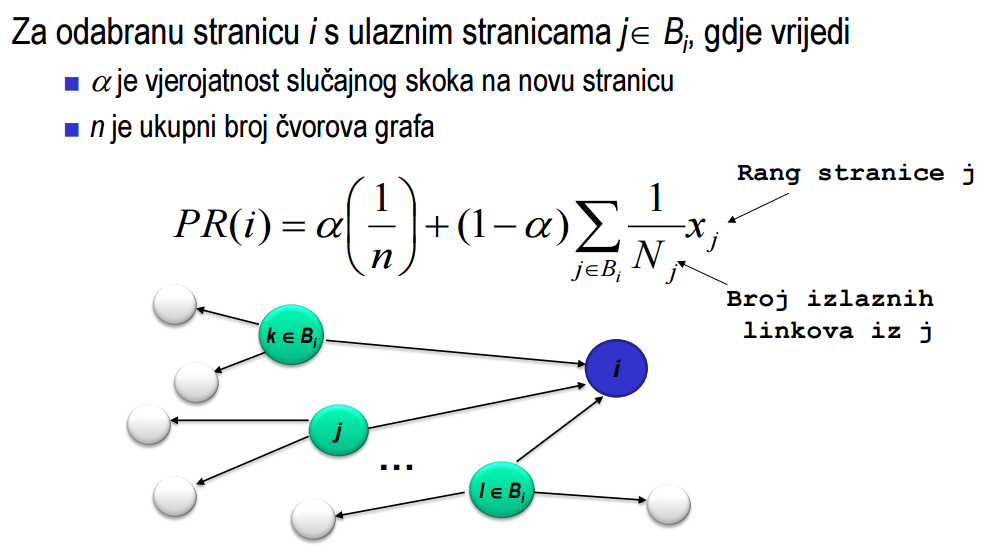
PageRank:

* Karakterizira količinu vremena koju korisnik povede na nekoj web stranici, može se usporediti s popularnošću/utjecajem stranice
* Jedna oda brojnih komponenti koja utječe na rang stranice prilikom pretraživanja
* Matematički: raspodjela vjerojatnosti za web-stranice
* Mjera popularnosti web-resursa, neovisna o tekstualnom sadržaju dokmenta
* Algoritam koji je učinio Google najpopularnijom tražilicom
* Modelira web (resurse i poveznice) usmjerenim grafom
* Koristi ulazne i izlazne poveznice radi rangiranja relevantnih stranica s obzirom na njihovu popularnost
* Neovisan o upitu
* Iterativni izračun PageRanka
  + Pretpostavka: korisnik započinje istraživanje na slučajno odabranoj stranici (jednake vjerojatnosti za sve stranice)
  + Pojednostavljenje: zanemarujemo vjerojatnost prelaska na novu stranicu koja nije linkana s postojeće stranice

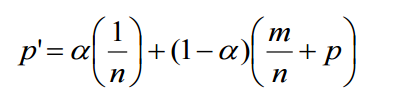




Potpuna definicija PageRanka:



* Dva dodatna proširenja:
  + Uzeti u obzir čvorove bez izlaznih grana (dangling nodes), znamo koliko ih ima tek nakon što pobrojim u prvom koraku kada iterativno računamo rangove (pojednostavljeno)
  + Uključiti vjerojatnost za slučajni skok na bilo koju stranicu
* Rješenje:
  + U drugom koraku se redistribuira „missing PageRank mass“ i uzimaju u obzir slučajni skokovi
  + P – PageRank iz prvog koraka
  + M (missing PageRank mass) – ovisi o broju dangling nodes



Zaključak:

* Primjena liste susjedstva, iterativna propagacija
* U svakom čvoru se provodi izračun lokalnih vrijednosti koje se prenose susjedima
* Nedostatak MapReducea: nedostupnost globalnih struktura podataka, složenost O(n2) pa nije pogodan za paralelnu obradu gustih mreža

1. **Dubinska analiza podataka u raspodijeljenom okruženju**

Uvod u strojno učenje:

* Strojno učenje je polje računalne znanosti koje se bavi specifičnim načinom programiranja u kojem očekujemo da računalo samostalno dođe do određenih spoznaja na osnovu dostupnih podatkovnih skupova i odabrane metode “učenja”
* pretpostavka: skup podataka odražava neka svojstva, odnose ili zakonitosti iz stvarnog svijeta koje uz odgovarajuće metode možemo prepoznati, opisati i na određeni način iskoristiti
* problem - složenost stvarnog svijeta, velika količina "šuma"

Nadzirano učenje (supervised learning)

* podvrsta strojnog učenja kod koje podatkovni skup dijelimo na ulaze i izlaze i potom tražimo način koji će omogućiti
  + transformaciju ulaza u dovoljno točnu procjenu izlaza
  + dobivanje uvida u prirodu povezanosti ulaza i izlaza
* najčešće se svodi na stvaranje „prediktivnog“ modela koji na osnovu matematičke formula ili algoritma pretvara ulazne varijable (prediktore) u izlazne (predikcije)
* kod nadziranog učenja često razmatramo dva aspekta
  + prediktivna moć modela – koliko su točne predikcije dobivenog modela
  + interpretabilnost modela – dobivanje uvida u prirodu povezanosti ulaza i izlaza
* subjektivna važnost pojedinog aspekta je jedan od kriterija odabira metode stvaranja prediktivnog model
  + white-box modeli daju jasan uvid u način kako funkcioniraju te prirodu povezanosti prediktora i predikcija (skupno ili pojedinačno)
  + black box modeli nisu interpetabilni (najčešće zbog visoke složenosti interne strukture) ali mogu imati visoku prediktivnu moć
* priroda ciljne varijable u općenitom slučaju može biti:
  + numerička (kontinuirana)
  + kategorijska (diskretna)
* prema tome metode nadziranog strojnog učenja možemo podijeliti na:
  + regresijske – linearna regresija, ridge/lasso/elasticnet regresija…
  + klasifikacijske – LDA, QDA, kNN, SVM, klasifikacijska stabla
* podjela nije ograničavajuća – neke metoda su univerzalne (npr. neuronske mreže), a neke koje su prirodno regresijske se često adaptiraju za primjenu nad klasifikacijskim problemima i obrnuto

Nenadzirano učenje (unsupervised learning)

* tip strojnog učenja kod kojeg imamo samo ulaze tj. nemamo definirane konkretne izlaze – ciljane varijable
* nenadzirano učenje usredotočeno je na pronalaženje interesantnih uzoraka i odnosa između elemenata podatkovnog skupa, tj. odgovara na pitanja kao što su:
  + koji podaci su "slični" jednim drugima? Što karakterizira tu "sličnost"?
  + koji podaci drastično "iskaču" od ostalih podataka?

Karakteristike podatkovnog skupa za analizu:

* najčešće trebamo podatke organizirane u denormaliziranu tabličnu strukturu – ovo često zovemo podatkovni okvir (data frame)
* skup za treniranje i skup za testiranje

Linearna regresija:

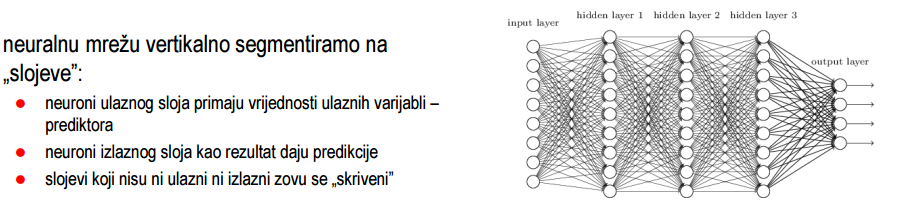
* jedna od najjednostavnijih i najraširenijih metoda nadziranog učenja
* upotreba: kada između prediktora i predikacija postoji linearna veza – moguće provjeriti matematički (Pearsonov koeficijent kolinearnosti)
* prediktivni model u ovom slučaju predstavlja pravac
  + jednadžba pravca definira metodu transformacije ulaza u izlaza
* kod traženja najboljeg pravca najčešće se koristi metoda najmanjih kvadrata
  + traži se pravac kod kojeg će zbroj kvadrata odstupanja ciljeva od pravca (reziduali) biti najmanji
  + reziduali će se kod linearne regresije jednoliko rasipati oko pravca
* pogrešku najčešće evaluiramo uz pomoć RMSE mjere (root mean square error)
  + kvadriramo greške, sumiramo, podijelimo brojem obzervacija i korijenujemo
  + RMSE opisuje koliko u prosjeku predikcije odstupaju od stvarnih veličina

kNN klasifikacija:

* K najbližih susjeda
* Klasificiramo nove obzervacije usporedbom njihove sličnosti sa postojećim obzervacijama
* Kod ove metode sam ulazni skup podataka predstavlja prediktivni model
  + Za novu obzervaciju pronalazi se k obzervacija koji su najbliži novoj
  + Većinskim glasanjem određuje se koju klasu pridijeliti novoj obzervaciji
* Moramo paziti na definiciju udaljenosti – normalizacija prediktora
* kNN ima hiperparametar – broj susjeda k
  + hiperparametar ili parametar za podešavanje (tuning parameter) je parametar kojeg analitičar sam mora odabrati prije primjene metode
  + znatno utječe na rezultate
* postoji više načina procjene kvalitete klasifikacijskog modela
  + najjednsotavnije je računati točnost (accuracy) tj. omjer točnih klasifikacija i ukupnog broja novih obzervacija
  + mjera točnosti često nije najbolji pokazatelj kvalitete modela

Neuralne mreže:

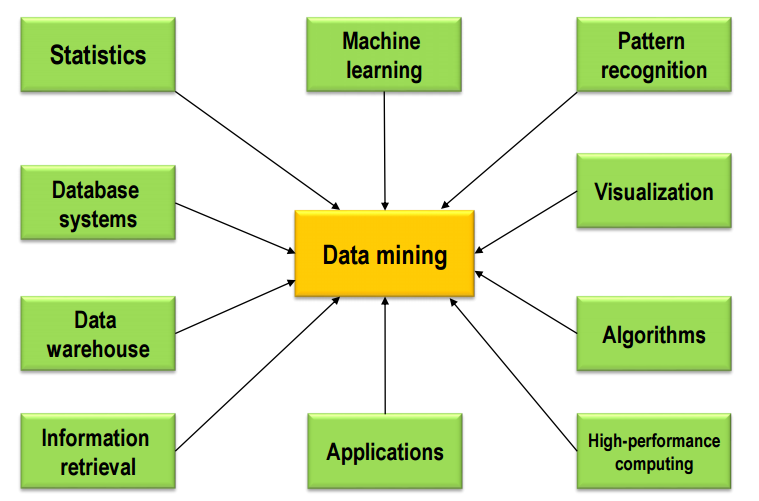
* neuralne mreže su metoda strojnog učenja inspirirana biološkim neurološkim sustavima
* često se primjenjuju za probleme gdje je „teško” eksplicitno objasniti vezu između ulaza i izlaza
  + prepoznavanje rukopisa, raspoznavanje lica, obrada prirodnog jezika…
* element neuralne mreže je neuron – element koji provodi elementarnu pretvorbu ulaza u izlaze

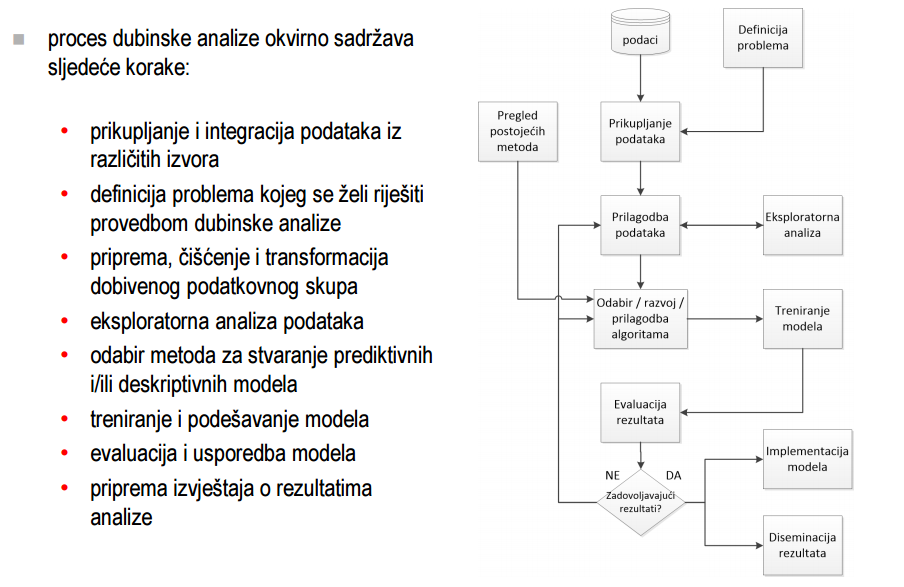


* pojednostavljeno, svaki neuron ima sljedeće parametre:
  + težine ulaza wi
  + iznos praga b
* neuron prosljeđuje signal ako je težinska suma ulaza veća od zadanog praga
* treniranje neuralne mreže svodi se na podešavanje težina i pragova kako bi izlazi bili što bliži očekivanim vrijednostima
* klasičan algoritam treniranja: feed-forward backpropagation
  + počinje se sa nasumičnim vrijednostima težina i pragova
  + signal se „propusti” kroz neuralnu mrežu
  + kretanjem od izlaznog sloja prema ulaznog iterativno se rade „podešavanja” težina i pragova kako bi se smanjila ukupna greška
* neuralne mreže imaju niz mogućih hiperparametara
  + broj skrivenih slojeva
  + broj neurona u skrivenim slojevima
  + brzina učenja
  + broj iteracija učenja
  + broj ulaznih obzervacija koji se koristi u jednoj iteraciji učenja
* u novije vrijeme je aktualan deep learning
  + istraživanje mogućnosti dubokih neuralnih mreža sa većim brojem skrivenih slojeva

Dubinska analiza podataka (data mining)

* pojam dubinske analize nije strogo definiran te mu se značenje mijenja u ovisnosti o kontekstu
* u općenitom slučaju može se smatrati da je to skup metoda za izvlačenje korisnih informacija iz nekog (najčešće velikog) skupa podataka
* dubinska analiza podataka između ostalog uključuje





* priprema podataka (dana preparation, dana munging)
  + često najzahtjevniji dio procesa analize podataka (50-80% ukupnog vremena za analizu)
  + uključuje:
    - uređivanje podataka – svaki redak jedna obzervacija, svaki stupac jedno svojstvo, čišćenje podataka, uklanjanje smeća, uvođenje konzistentnosti
    - upravljanje nedostajućim vrijednostima – zašto nedostaju? Da li nedostaju nasumične ili sistematične vrijednosti? Koncentrirane samo u određenim retcima/stupcima?
    - Upravljanje iskočnicama (outliers) – kako ih prepoznati i interpretirati? Što napraviti s njima?
    - Provedba transformacija, uočavanje potrebe za dodatnim podacima…
* Eksploratorna analiza podataka
  + Razumijevanje podataka je nužno za provođenje daljnjih korak analize
  + U pravilu označava:
    - Statističku obradu podataka
    - Stvaranje vizualizacija
  + S njom dobivamo:
    - Uvid u razdiobe varijabli
    - Naslućujemo prirode odnosa između varijabli
    - Uočavamo potencijalne iskočnice
    - Stvaramo temelje za odluke o daljnjem smjeru analitičkog procesa
* Treniranje, podešavanje i evaluacija modela
  + U procesu modeliranja analitičar odabire:
    - Ciljnu varijablu
    - Prediktore
    - Metodu strojnog učenja
    - (hiper)parametre metode
    - Način treniranja
    - Mjeru evaluacije
  + Podaci moraju biti prilagođeni prirodi odabrane metode
    - Jako puno metoda strojnog učenja voli normalnu razdiobu numeričkih prediktora
    - neke metode su "osjetljive" na iskočnice, različite skale, kategorijske varijable sa disbalansiranim zastupljenostima kategorija, "nakrivljene" razdiobe i sl.
    - često su potrebne dodatne transformacije ulaza i/ili izlaza kao što je npr. centriranje, skaliranje, logaritmiranje, BoxCox transformacije i sl.
    - puno metoda je jako osjetljivo na korelirane ulaze (slučaj više stupaca sadržavaju istu informaciju o izlazima a metoda ih tretira kao neovisne)
  + kod analize često imamo dostupan samo jedan skup podataka
    - obično razdvajamo na trening skup (~70%) i testni skup (~30%)
      * to možemo napraviti nasumično, kronološki, stratificirano nasumično..
  + evaluacija metode na treningu skupu nije objektivan pokazatelj kvalitete modela
    - tzv. Overfitting – model dobro predviđa na trening skupu ali loše generalizira
  + danas vrlo često ne radimo "jedno" rezanje već se koristimo nekom od odabranih metoda "ponovnog uzorkovanja" (engl. resampling):
    - Monte Carlo resampling: odabrani broj puta ponavljamo "rezanje"
    - k-fold crossvalidation: dijelimo skup na k dijelova i onda k puta treniramo i testiramo, pri čemu svaki "fold" jednom postaje testni skup
    - leave-one-out-crossvalidation (LOOCV) – posebni slučaj k-fold CV u kojem svaki put ostavljamo samo jednu obzervaciju kao testni skup
    - bootstrap – slično kao Monte Carlo, ali umjesto običnog "rezanja" radimo slučajni odabir pri čemu dozvoljavamo da se ista obzervacija odabere više puta
  + ovakvi pristupi danas su lako izvedivi zbog dostupnosti računala visokih performansi a omogućuju
    - veću objektivnost evaluacije
    - uvid u stabilnost odabranih modela
    - procjenu učinkovitosti modela kod različitih odabira hiperparametara



Dubinska analiza podataka u raspodijeljenom okruženju

* Problem skalabilnosti alata za dubinsku analizu podataka
  + dubinska analiza podataka očekivano radi sa velikim podatkovnim skupovima
  + alati za dubinsku analizu intenzivno koriste radnu memoriju
    - memorija ne čuva samo inicijalni podatkovni skup nego i brojne međurezultate nastale u procesu pripreme podataka
  + alati prirodno ne podržavaju paralelizam
    - metode su uglavnom dizajnirane za centralizirano izvođenje
    - paralelizam, ako je nužan, često zahtijeva korištenje zasebnih objekata predviđenih upravo za to uz vrlo intenzivno niskorazinsko upravljanje
* Raspodijeljena obrada i metodologija dubinske analize
  + analitičari podataka (u novije vrijeme "podatkovni znanstvenici", engl. data scientist) imaju specifični skup znanja koji su vezani uz matematiku, statistiku i računarstvo
    - podatkovni znanstvenici često nisu primarno programeri
    - raspodijeljena obrada podataka često naglasak stavlja na programerska znanja te opća znanja o informacijskim sustavima
  + analize se često provode interaktivno, po principu pitanje – odgovor, uz detaljno proučavanje međurezultata i kontinuirano donošenje odluka o daljnjem smjeru analize
    - kod raspodijeljene obrade podataka često smo usredotočeni na implementaciju unaprijed specificiranog procesa obrade koji se nakon postavljanja na sustav provodi automatski
  + analiza podataka često je vrlo usko povezan uz ustanovljeni radni proces koji uključuje organizaciju analize, metodologiju, korištene alate i sl.
    - niskorazinski detalji kao što je detaljno upravljanje paralelizmom, pisanje tzv. "boilerplate" programskog koda, pisanje integracijskih skripti i sl. često su vrlo disruptivni

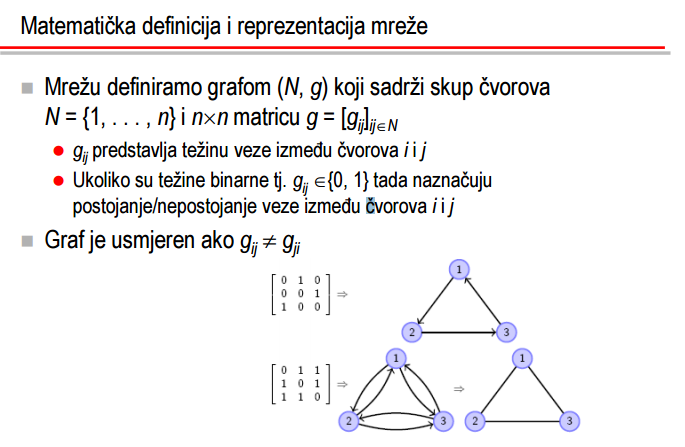
Apache Spark + MLib

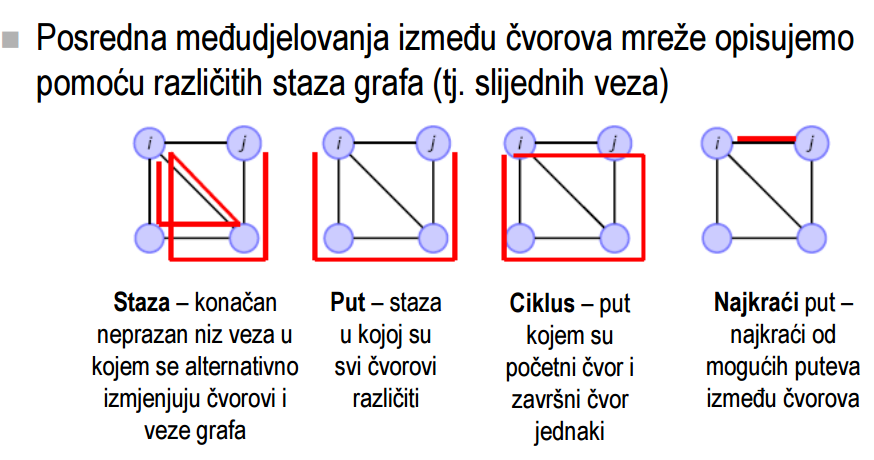
* MLlib is Apache Spark's scalable machine learning library
* orijentiran programskim jezicima Java, Scala, Python i R
* omogućuje skalabilno izvođenje algoritama strojnog učenja u okruženju velikih podataka
* MLib kao takav nije nužno dizajniran za paralelizam kakav nam često treba u strojnom učenju tj. dubinskoj analizi
  + proces analize često uključuje treniranje većeg broja modela nad istim podatkovnim skupom
  + MLib / Spark je dizajniran za paralelizam u smislu obrade jednog, velikog, raspodijeljnog podatkovnog skupa
* Kada koristiti Spark + Mlib?
  + ako smo suočeni sa klasičnim Big Data scenarijem
  + ako želimo proširiti mogućnosti postojećih alata
* Negativne strane
  + tehnologija je vrlo nova i iznimno turbulentna
  + na tržištu se pojavljuje veliki broj paralelnih, konkurirajućih rješenja
  + integracijska rješenja i dalje nisu na visokoj razini glede jednostavnosti, stabilnosti i transparentnosti

1. **Raspodijeljena analiza društvenih mreža**

Društvena mreža:

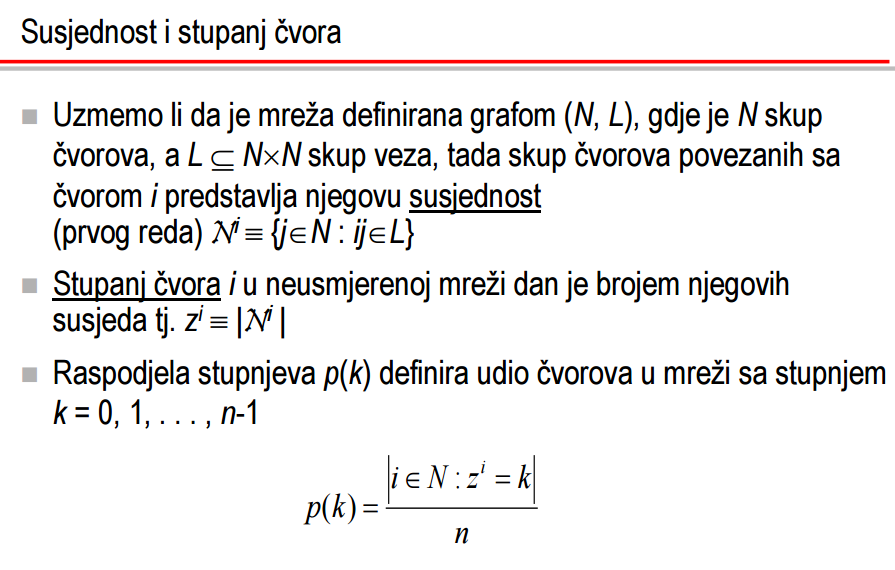
* Skup entiteta međusobno povezanih odnosima
  + Entiteti: ljudi, grupe ljudi, organizacije, brandovi
  + Odnosi: poznanstvo, obiteljska povezanost, privlačnost
* Mreža: skup čvorova međusobno povezanih granama



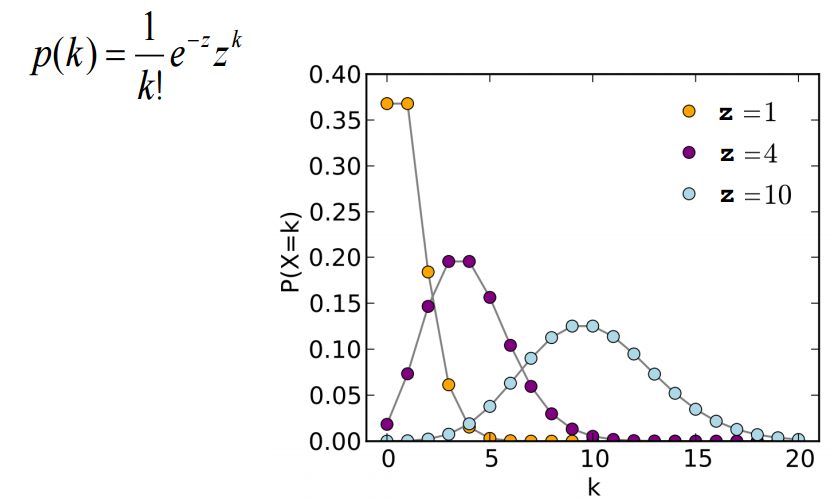


Analiza društvenih mreža:

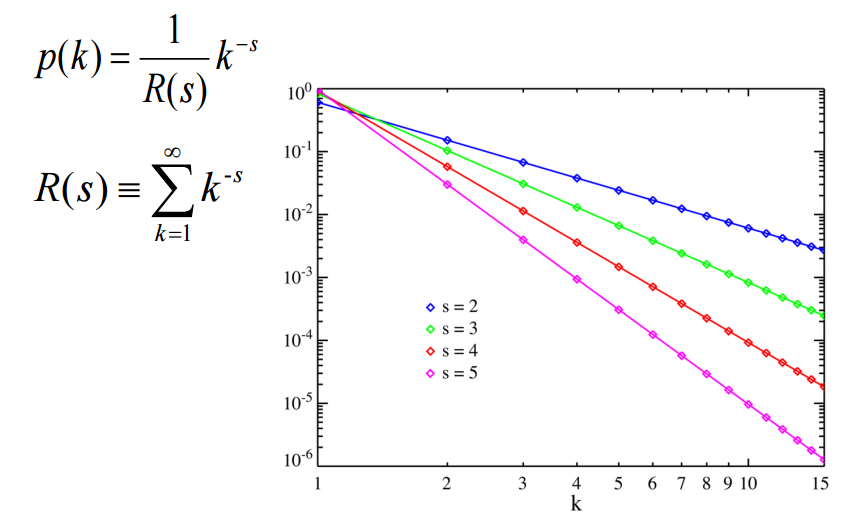
* Zasnovana na teoriji grafova i analizi kompleksnih mreža
* Kompleksne mreže karakterizira
  + Veliki broj čvorova u mreži
  + Složena raspodjela veza među čvorovima
* Analiziraju se statističke mjere korelacije veza i grupiranja korisnika (članova) mreže
  + Raspodjela stupnja čvora
  + Prosječni najkraći put
  + Koeficijent grupiranja
  + Centralnost čvorova
* Programski okviri za raspodijeljenu analizu grafova
  + Apache Giraph
  + Apache Spark GraphX
  + Apache Flink Gelly
* Apache Giraph
  + Temeljen na Googleovom sustavu Pregel
  + Koristi Hadoopovu implementaciju MapReduce-a
  + Napisan u Javi
  + Proceduralni programski model – MapReduce nad grafom
* Apache Spark GraphX
  + Temeljen na Googleovom sustavu Pregel
  + Proširenje Apache Sparka za rad s grafovima
  + Čvorovi i grane su RDD-ovi; deklarativni programski model
  + Napisan u Scali – loše dokumentiran i nije prilagođen Javi
* Apache Flink Gelly
  + Proširenje Apache Flinka za rad s grafovima
  + Deklarativni programski model
  + Napisan u Javi
    - Dobro dokumentiran
    - Prilagođen Javi
    - Dolazi s više algoritama od GraphXa
    - Lambda izrazi i parametrizacija rade samo s prevoditeljem Eclipse JDT (BOoHOo!)
* Susjednost i stupanj čvora



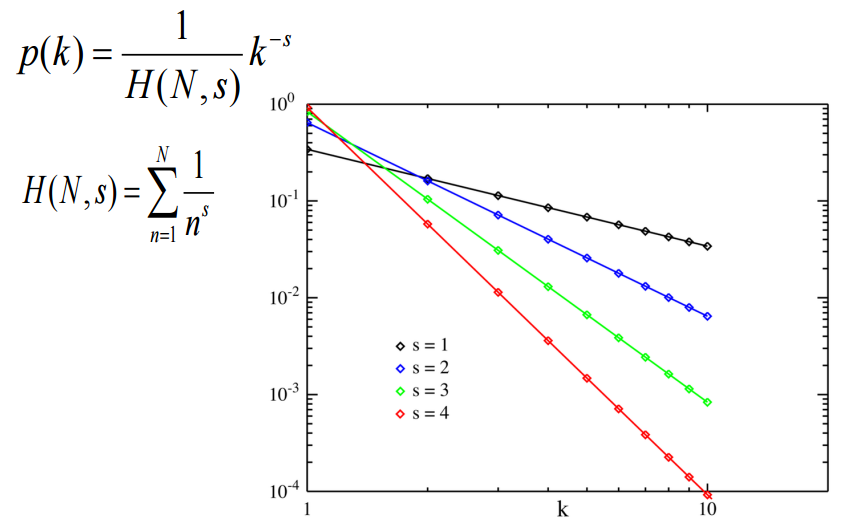
* Poissonova raspodjela stupnjeva



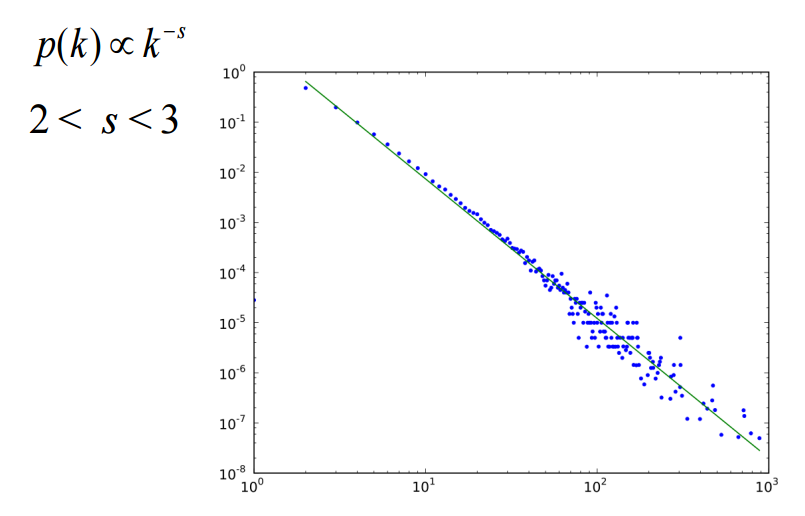
* Zeta raspodjela stupnjeva

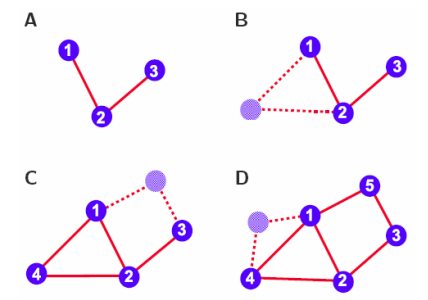
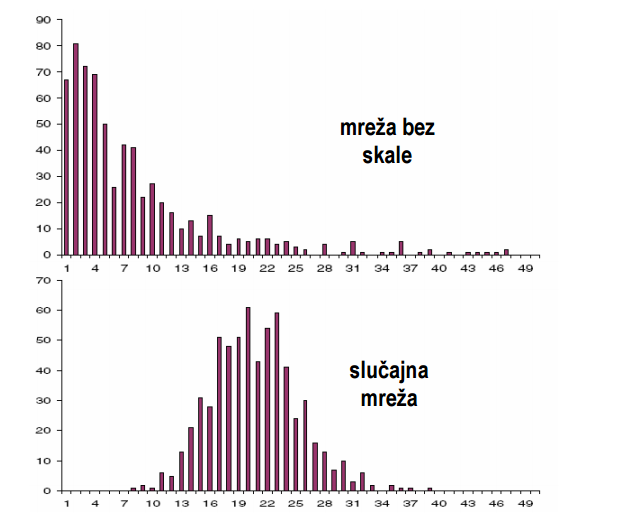


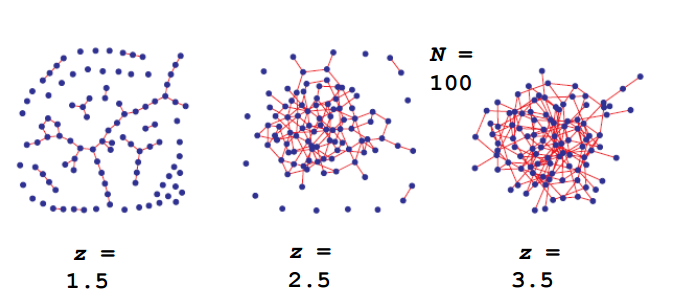
* Zipfova raspodjela stupnjeva



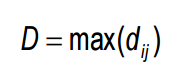
* Raspodjela stupnjeva bez skale (scale-free)

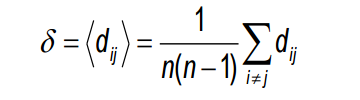


* + Polinomna raspodjela stupnjeva karakteristična je za društvene mreže
  + Pojavljuju se čvorovi koji imaju stupnjeve za red veličine veće od prosječnog stupnja čvora
  + Čvorovi sa visokim stupnjem su od iznimne važnosti u mnogim procesima u mreži (fragmentacija, širenje informacije)
  + Model Barabasi-Albert
    - Opisuje formiranje mreže bez skale putem preferencijalnog povezivanja
    - Novi čvor se povezuje s dva čvora odabrana na osnovu vjerojatnosti proporcionalnoj stupnjevima čvorova
* Slučajne mreže
  + Model Erdos-Renyi
    - Poissonova raspodjela stupnjeva
    - Prosječan stupanj čvora (u mreži sa ukupno n čvorova i vjerojatnošću povezivanja parova čvorova q) iznosi z=qn
    - Postojanje gigantske komponente u mreži je dano uvjetom q > 1/n

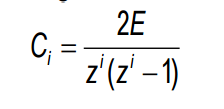
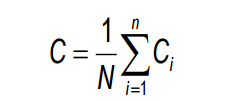


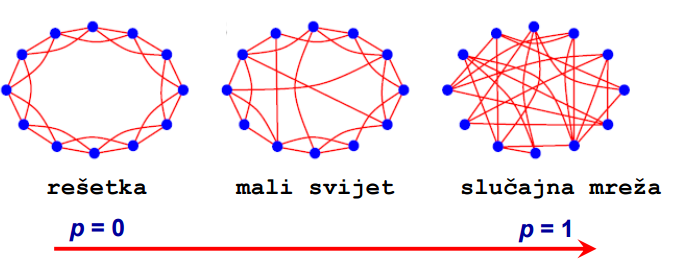
Prosječni najkraći put:

* Udaljenost dij između čvorova i i j definiramo kao najkraći put (minimalni broj veza na putu od i do j)
* Prosječni najkraći put δ predstavlja prosječnu udaljenost između svih parova čvorova
* Dijametar mreže D je maksimalna udaljenost među parovima čvorova u mreži



Koeficijent grupiranja (local clustering coefficient):

* Koeficijent grupiranja čvora i, Ci , predstavlja omjer broja međusobnih veza među susjedima (prvoga reda) čvora i, E, i ukupnog broja njihovih mogućih međusobnih veza
* Koeficijent grupiranja nam kaže koliko su "gusto" povezani prvi susjedi nekog čvora
* Koeficijent grupiranja mreže (network average clustering coefficient) je prosječni koeficijent grupiranja čvorova u mreži
* Efekt malog svijeta
  + Model Watts-Strogatz
    - Veze u rešetki se nasumično prespajaju (vjerojatnost prespajanja p)
    - Prečaci drastično smanjuju prosječni najkraći put
    - Mali svijet: mreža s visokim koeficijentom grupiranja i malim prosječnim najkraćim putom



Centralnost čvorova:

* Centralnost pokušava identificirati najvažnije čvorove u grafu
  + Utjecajne osobe u društvenoj mreži ili najbitniji čvorovi na Internetu ili najveći raznositelji zaraze
* Različite vrste centralnosti čvorova
  + Centralnost po položaju (betweenness centrality) – nalazi se na puno najkraćih puteva
  + Centralnost po blizini (closeness centrality) – kratki najkraći putevi
  + Centralnost po svojstvenom vektoru (eigenvector centrality)
  + Centralnost po stupnju (degree centrality)