Sustavi za davanje preporuka

Damijanić, Zlatko

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Zagreb, Faculty of Science / Sveučilište u Zagrebu, Prirodoslovno-matematički fakultet**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:217:058735

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-02-27**

Repository / Repozitorij:

Repository of the Faculty of Science - University of Zagreb

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Zlatko Damijanić

SUSTAVI ZA DAVANJE PREPORUKA

Diplomski rad

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Zlatko Damijanić

SUSTAVI ZA DAVANJE PREPORUKA

Diplomski rad

Voditelj rada:
izv. prof. dr. sc. Luka Grubišić

Zagreb, 2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

”Ne idi utabanim putem. Umjesto toga, kreni tamo gdje puta nema i ostavi trag.” (n. a.)

Zahvaljujem svom mentoru izv. prof. dr. sc. Luki Grubišiću na strpljenju, pomoći i vodstvu pri izradi ovog diplomskog rada.

Srdačno zahvaljujem ravnatelju i kolegama Srednje strukovne škole u Samoboru te ravnateljici i kolegama Srednje škole Zvane Črnje u Rovinju na iskazanom povjerenju, razumijevanju i potpori.

Također, hvala prijateljima i odbojkaškoj ekipi ŽMOK koji su mi produžili i olakšali studiranje te ga učinili najljepšim dijelom moga života.

Na kraju, najveće hvala mojim roditeljima i obitelji na sveobuhvatnoj podršci tijekom cijelog školovanja.

Sadržaj

Uvod

Koja je razlika između pretrage i preporuke? Pretraživanje je kada korisnik traži neku informaciju, a preporuka je kada informacija pronađe korisnika koji nju smatra relevantnom. Informacijsko doba uslijedilo je nakon industrijskog doba i obilježeno je naglim razvitkom i rasprostranjenosti informacijske tehnologije pomoću kojih informacije postaju sve dostupnije. Internet i digitalna tehnologija pružaju nove prilike za učenje, omogućuju pojedincima direktnu prodaju ideja, usluga ili proizvoda i pristup informacijama. Povećanjem količine informacije koja nas okružuje javlja se potreba za što boljim upravljanjem informacijama. Bitan aspekt tog upravljanja je filtriranje informacija koje se postiže pretragom ili preporukom. Kod pretrage korisnik upisuje ključne riječi te se na temelju unesenih riječi informacije filtriraju i prikazuju korisniku. Najpoznatiji primjer filtriranja je web tražilica Google koja pretražuje web stranice, a sustavi za davanje preporuka otkrivaju sadržaj na Internetu koji bi mogao biti zanimljiv korisniku na temelju njegovih prethodnih aktivnosti.

U ovom radu analizirat ćemo sustave za davanje preporuka na webu. Dana je njihova podjela i opisani su najkorišteniji algoritmi pojedine metode. Navedeni su njihovi nedostaci i prednosti te su opisane mjere za evaluaciju. Za implementaciju i evaluaciju korišten je programski okvir ”Apache Mahout”.

Poglavlje 1.

Opis sustava za davanje preporuka na webu

Sustav za davanje preporuka (eng. recommendation system) pripada području filtriranja informacija i čini ga skup metoda za analizu kako bi predvidio što bi se moglo svidjeti, što bi moglo zanimati ili što je relevantno za korisnika. Najčešći oblik preporuka je lista objekata za koje je sustav predvidio da su najrelevantniji za korisnika. Primarni razlog potrebe za takvim sustavima je previše raspoloživih informacija (eng. information overload) i raznih mogućnosti interakcije (eng. interaction overload), na primjer: pregledavanje, kupovanje, označavanje, ocjenjivanje, komentiranje, praćenje itd. Web mjesta kako bi korisnicima olakšali pronalaženje i dostupnost informacija te time povećali posjećenost i profitabilnost implementiraju sustave za davanje preporuka.

Primjeri web sjedišta koje daju preporuke svojim korisnicima i područje njihove primjene:

- movielens.org, netflix.com, imdb.com, themoviedb.org - filmovi
- last.fm, pandora.com, spotify.com, napster.com - glazba
- flickr.com, instagram.com, pinterest.com, imgur.com - slike
- amazon.com, ebay.com, aliexpress.com, alibaba.com - e-trgovina
- youtube.com, dailymotion.com, vimeo.com, metacafe.com - video
- del.icio.us, diigo.com, raindrop.io, getpocket.com - web oznake
- facebook.com, myspace.com, linkedin.com, gotinder.com - prijatelji

- airbnb.com, booking.com, trivago.com, tripping.com - smještaj
- bibsonomy.org, citeulike.org, mendeley.com, zotero.org - publikacije
- google.com/adsense, propellerads.com, media.net, infolinks.com - oglasi

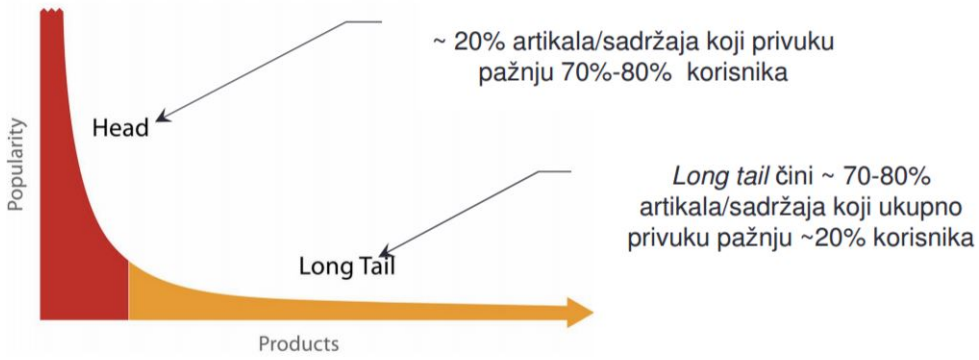
Problem preporuka definiran je 90-tih godina, ali se danas još razvija i proširuje se područje njegove primjene. Godine 2006. problem je populariziran natjecanjem "Netflix Prize". Netflix je američko poduzeće koje ima web sjedište za gledanje filmova, serija, emisija i TV kanala na zahtjev. Natjecanje je bilo otvorenog tipa i cilj je bio pronaći algoritam koji će poboljšati predikcije Netflix-ovog algoritma. Trebalo je predvidjeti koju će ocjenu korisnik dodijeliti nekom filmu samo na temelju prijašnjih ocjena korisnika, bez dodatnih informacija o korisnicima i filmovima. Nagradu od 1.000.000 američkih dolara 2009. godine odnio je tim "BellKor's Pragmatic Chaos" koji je poboljšao predikcije za 10.06%. Netflix je 2010. godine objavio da neće biti drugog natjecanje zbog federalne tužbe radi narušavanja privatnosti. Naime, bez obzira na to što su podaci o korisnicima i filmovima bili anonimni, moglo se povezati pojedine korisnike s ocjenama filmova pomoću web sjedišta "imdb.com" (International Movie Database).

Sustave za davanje preporuka dijelimo na nepersonalizirane i personalizirane. Nepersonalizirani sustavi su najjednostavniji i ne uzimaju u obzir preferencije korisnike te daju svim korisnicima jednake preporuke. Prednost je što su jednostavni za implementaciju i lako je prikupiti podatke. Zbog širokog spektra primjene, riječ objekt odnosi se na knjige, filmove, video, proizvode i ostalo što sustav preporučuje.
Primjeri lista preporuka nepersonaliziranih sustava:

- trenutno pregledavani objekti
- najbolje ocijenjeni objekti (prosječna ocjena svih korisnika)
- najnoviji objekti
- najprodavaniji objekti
- najpopularniji objekti

Prednost personaliziranih sustava je što analiziraju podatke o korisnicima, njihovim aktivnostima u prošlosti, njihove ocjene proizvoda, njihove odnose s drugim korisnicima, podatke o samom objektu te tako daju prilagođene preporuke za pojedinog korisnika. Kvaliteta sustava mjeri se kroz njegovu sposobnost davanja preporuka iz "dugog repa" (eng.

Long Tail) distribucije. Nepersonalizirani sustavi imaju tendenciju preporučivanja najpopularnijih objekata koji postaju sve popularniji. Najpopularniji objekti uglavnom nisu relevantni za pojedinog korisnika. Stoga je važno da sustav daje preporuke ostalih objekata kojih je mnogo više, ali koji su relevantniji korisniku. Slika 1.1. prikazuje odnos tih objekata.

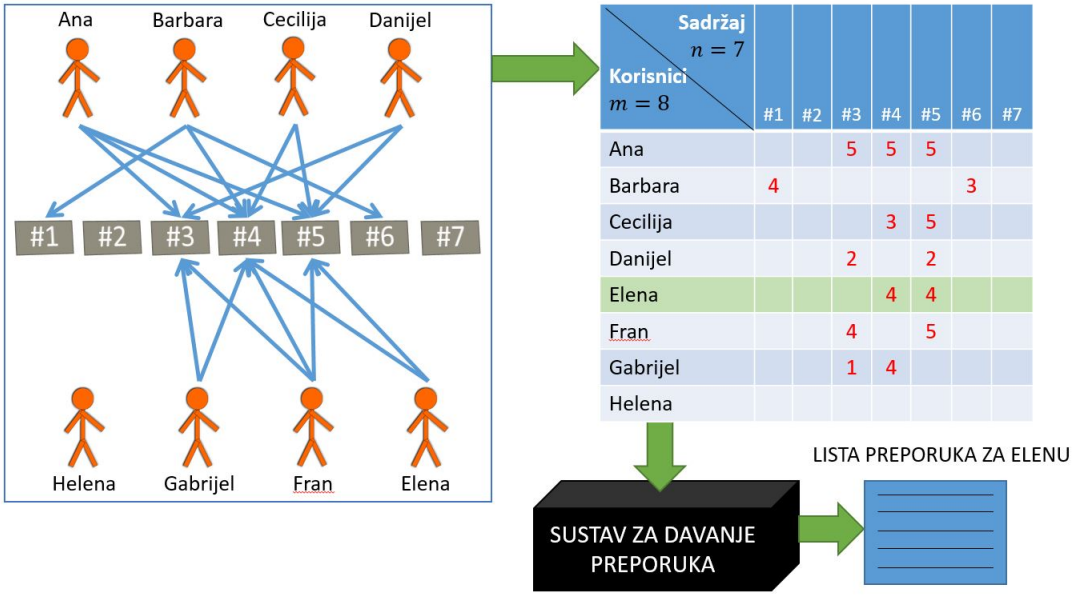


Slika 1.1. Preporučivanje objekata iz Long Tail-a (izvor: J. Jovanović, 2014.)

1.1. Nedostaci personaliziranih sustava

Najčešći nedostaci personaliziranih sustava su: problem previše rijetkog uzorkovanja (eng. sparsity), premalo ocjena korisnika na početku rada (eng. cold start), problem novih korisnika ili objekata (eng. new user, new item), korisnik ili objekt koji nema sličnosti s niti jednim drugim korisnikom ili objektom (eng. black sheep), problem skalabilnosti (eng. scalability), nefleksibilnost (eng. inflexibility), problem raznolikosti ili pretreniranja (eng. diversity ili overfitting), izolacija preporuka (eng. filter bubble) i problem manipulacije sustava (shilling attacks) te pitanje privatnosti.

Da bi se bolje razumjelo i opravdalo uvođenje novih i izmjena postojećih metoda za davanje personaliziranih preporuka, potrebno je proučiti njihove nedostatke. Nedostaci su opisani na primjeru sa slike 1.2.. U tom primjeru nalazi se 8 korisnika koji su ocijenili 7 objekata.



Slika 1.2. Primjer sustava u kojem su korisnici dodijelili ocjene objektima

Najveći nedostatak je problem previše rijetkog uzorkovanja. Neki objekti imaju mali broj ocjena dok je druge objekte ocijenila većina korisnika. Na slici 1.2. može se vizualno vidjeti da raspodjela ocjena nije povoljna za izračun predikcija. Ovoj kategoriji nedostataka pripada i problem malog broja ocjena na početku rada sustava. Također problem nastaje kod novih korisnika ili objekata, jer nemaju ocjena, te je takvom korisniku onemogućeno davanje personaliziranih preporuka, a novi objekt neće se nikome preporučiti. U sustavu se mogu pojaviti korisnici koji su dodijeli ocjene objektima kojima je dodijeljena samo njihova ocjena. Za takve korisnike se kaže da su ”crne ovce”. Na slici 1.2. vidimo novog korisnika ”Helena” i novi objekt ”#7” koji nemaju ocjena te crnu ovcu ”Barbaru” koja je ocijenila objekte koji imaju samo njezinu ocjenu.

Algoritmi koji su razvijeni na sustavu s malim brojem korisnika i objekata ne moraju nužno dovoljno brzo raditi u većim sustavima. Taj nedostatak naziva se problem skalabilnosti. Kako bi se povećala skalabilnost, razvijaju se sustavi temeljeni na modelu, ali se kod njih javlja problem nefleksibilnosti. To znači da sustav ne mijenja dovoljno brzo svoj model s obzirom na nove podatke, pa se gubi na točnosti predikcija.

U slučaju izolacije preporuka korisniku se preporučuje samo dio objekata koji su za njega predviđeni kao najrelevantniji. Za takav sustav kaže se da je došlo do pretreniranja.

Treba se voditi briga da u sustav ne dođu lažni korisnici samo kako bi ciljano ocijenili neke objekte lošom ili dobrom ocjenom, te tako utjecali na izračun predikcija. Također vrlo je bitna i privatnost korisnika, tj. kome su dostupne informacije o tome što su oni kupovali, pregledavali, slušali, pretraživali i slično.

1.2. Podjela personaliziranih sustava

Prve metode za davanje personaliziranih preporuka bazirane su na ocjenama koje korisnici dodjeljuju objektima. Za pojedinog korisnika na temelju njegovih ocjena pronalazi se k-najsličnijih korisnika (eng. k-nearest neighbours - kNN), za razliku od nepersonaliziranih sustava koji uzimaju u obzir sve ocjene. Predikcija s kojom ocjenom bi promatrani korisnik ocijenio neki objekt proizlazi agregacijom ocjena njegovih k-najsličnijih susjeda. Zatim se korisniku ispisuje lista preporuka koja sadrži n objekata s najboljim predikcijama ocjena (eng. top n recommendation). Bez obzira na jednostavnost implementacije i lakoće razumijevanja metode, nije ju moguće koristiti u praksi, jer korisnik želi preporuke u stvarnom vremenu. Kako je broj korisnika i objekta u praksi vrlo velik, složenost ovakvih metoda onemogućuje rad u stvarnom vremenu.

Da bi se smanjila složenost, umjesto da se računa sličnost između korisnika, računa se sličnost između objekata te se preporučuju objekti koji su najsličniji onima koje je korisnik najbolje ocijenio. Sličnost između objekata je puno stabilnija od sličnosti korisnika pa je moguće unaprijed odrediti sličnosti između objekata neznatno smanjujući točnost što daje veću skalabilnost ovoj metodi. Ove dvije metode pripadaju grupi sustava temeljenih na memoriji (eng. memory-based) jer se direktno primjenjuju na podatke bez pretprocesiranja. One mogu imati različite implementacije ovisno kojom skalom korisnik ocjenjuje objekte, kako mjerimo sličnost između korisnika i na koji način računamo predikciju ocjena.

Prethodno opisane metode za ulaz koriste sirove podatke. Svaki put potrebno je računati predikcije na svim podacima, što usporava sustav i potrebno je mnogo memorije za pohranu sličnosti. Sustavi temeljeni na modelu (eng. model-based) nad podacima izrađuju model te kasnije ne temelju njega, a ne svih podataka, daje preporuke čime se ubrzava rad

sustava. Modeli se izrađuju pomoću algoritama strojnog učenja (eng. machine learning) i rudarenja podacima (eng. data mining). Sustavi temeljeni na memoriji i modelu pripadaju skupini sustava temeljenih na suradnji (eng. collaborative filtering) jer se baziraju na podacima dobivenim zajedničkim radnjama korisnika, kao što su ocjenjivanje, kupovanje, pregledavanje i sl. Njihov formalni opis i primjeri različitih implementacija odrađeni su u 2. poglavlju.

Ako se u sustavu nalazi puno objekata, a korisnici ih jako malo ocjenjuju javlja se problem previše rijetkog uzorkovanja. Zbog toga se razvijaju sustavi temeljeni na sadržaju objekata (eng. content-based). Pomoću funkcija sličnosti izračunava se sličnost između objekata na temelju njegovih atributa i ključnih riječi. Korisniku se preporučuju najsličniji objekti onima koje je korisnik najbolje ocijenio. Metode su opisane u 3. poglavlju.

Kada se sustavi primjenjuju na objekte koje se vrlo rijetko kupuju i ocjenjuju, kao što su automobili, smještaj, digitalne kamere, tada ne možemo primijeniti navedene metode zbog male količine podataka. Da bi se u takvim slučajevima davale preporuke, potrebno je pohraniti sve relevantne podatke o objektu, za takve sustave kaže se da su temeljeni na znanju (eng. knowledge-based). Korisnik može pretraživati objekte tako da unosi ograničenja preko tražilice, koja može imati različite oblike unosa. Preporuke moraju zadovoljavati dana ograničenja, pa se za takve sustave kaže da su temeljeni na ograničenjima (eng. constraint-based). U sustavima temeljenima na slučajevima (eng. case based) preporuke se temelje na sličnosti prethodno zadanih zahtjeva za pretragu.

Širenjem društvenih mreža (eng. social web) i spajanjem različitih uređaja na internet (eng. Internet of thinks) dolazi do razvoja novih metoda preporuka (eng. novel methods). Metode temeljene na spolu, nacionalnosti i sličnih atributa pripadaju sustavima temeljeni na demografiji (eng. demographic-based). Korisnici na društvenim mrežama imaju prijatelje ili pratitelje, preporuke koje se baziraju na takvim odnosima čine sustav temeljen na zajednici (eng. social-based ili community-based). Također, u takvim mrežama moguće je izraditi profil korisnika te na neki način odrediti njegovu osobnost pa se za takve sustave kaže da su temeljeni na osobnosti. Na nekim web sjedištima omogućeno je korisnicima slobodno označavanje objekata oznakama ili ključnim riječima (eng. folksonomy), što omogućuje izradu sustava temeljenog na oznakama (eng. tag-based ili keyword-based). Sustavi koje u obzir uzimaju kontekst u kojem se korisnik nalazi (lokacija, vrijeme, ras-

položenje, dan u tjednu, svrha pretrage i sl.) temeljeni su na kontekstu (eng. context-based).

Za implementaciju odabran je programski okvir "Apache Mahout" koji sadrži efikasne algoritme koji se mogu izvoditi paralelno i distribuirano. Kod je pisan u Javi u programskom okruženju Eclipse. Implementacije se nalaze u 4. poglavlju.

Zbog raznih dostupnih metoda i njezinih različitih implementacija, potrebno ih je evaluirati kako bi se za pojedinu domenu primjene izabrala najprikladnija. Mjere za njihovu evaluaciju i evaluacije sustava izrađenih različitim algoritmima pomoću Apache Mahout-a na "MovieLens" bazama podataka dane su u 5. poglavlju.

Sažeti pregled podjele personaliziranih sustava:

- klasične (eng. classic methods) ili tradicionalne metode (eng. traditionals methods)
  - temeljeni na suradnji (eng. collaborative filtering)
    - temeljeni na memoriji (eng. memory-based): temeljeni na sličnosti korisnika (eng. user-based ili user-user collaborative filtering), temeljeni na sličnosti objekata (eng. item-based ili item-item collaborative filtering)
    - temeljeni na modelu (eng. model-based)
  - temeljeni na sadržaju objekata (content-based)
  - temeljeni na znanju (eng. knowledge-based)
    - temeljeni na ograničenjima (eng. constraint-based)
    - temeljeni na slučajevima (eng. case-based)
- nove metode (eng. novel methods)
  - temeljeni na demografiji (eng. demographic-based)
  - temeljeni na zajednicama (eng. social-based ili community-based)
  - temeljeni na osobnosti (eng. personality-based)
  - temeljeni na oznakama (eng. tag-based ili keyword-based)
  - temeljeni na kontekstu (eng. context-based)
- hibridne metode (eng. hybrid methods)

Poglavlje 2.

Sustavi temeljeni na suradnji

Među najpopularnijim sustavima koji se koriste su oni temeljeni na suradnji. Kod takvih sustava korisnici ocjenjuju objekte te na temelju njih kreiraju preporuke. Dijelimo ih na sustave temeljene na memoriji i one temeljene na modelu. Sustavi temeljeni na memoriji koriste sve dostupne podatke bez pretprocesiranja, dok oni temeljni na modelu prvo izrađuju model pa kasnije preporuke baziraju na njemu.

2.1. Sustavi temeljeni na memoriji

Sustavi temeljeni na sličnosti korisnika

Ocjena koju bi korisnik dao nekom objektu, kojeg još nije ocijenio, predviđa se na temelju k najsličnijih korisnika koje zovemo k-najbliži susjedi. Način predviđanja ocjena najčešće je određen sljedećom definicijom.

Definicija 2.1.1. Neka je U skup korisnika (eng. users), a I skup objekata (eng. items), korisnici ocjenjuju pojedini objekt što je dano funkcijom $r : U \times I \to \mathbb{N}$. Vrijednosti te funkcije zapisane su u matrici ocjena R. Ako korisnik nije ocijenio neki objekt, tada je vrijednost funkcije r jednaka nuli. $N_{u,i} \subseteq U$ je skup od najviše k korisnika koji su najsličniji korisniku u i koji su ocijenili objekt i. Predviđanje koju ocjenu bi dao korisnik u objektu i računa se aritmetičkom sredinom s težinama, pri čemu su težine zadane funkcijom sličnosti

korisnika $s : U \times U \to \mathbb{R}$.

$$p : U \times I \to \mathbb{R}^+, p(u,i) = \overline{r_u} + \frac{\sum_{u' \in N_{u,i}} s(u,u')(R(u',i) - \overline{r_{u'}})}{\sum_{u' \in N_{u,i}} s(u,u')}$$

Pri čemu je $\overline{r_u}$ aritmetička sredina svih ocjena koje je dao korisnik u.

$$\overline{r_u} = \frac{\sum_{i \in I} R(u,i)}{card(\{i \in I | R(u,i) \neq 0\})}$$

Prema prethodnoj definiciji može se napisati općeniti algoritam za davanje liste preporuke nekom korisniku na temelju ocjena njemu najsličnijih korisnika.

Algoritam 2.1.2. *(izrada liste preporuka u sustavima temeljenima na sličnosti korisnika)*
Ulaz: Broj korisnika u, matrica ocjena R, funkcija sličnosti između korisnika s, broj najsličnijih susjeda k, broj preporuka n
Izlaz: Lista preporuka s n objekata koji imaju najbolju predikciju ocjena.

1. Računanje sličnosti korisnika u s ostalim korisnicima.
2. Odabir objekta i kojeg korisnik u nije ocijenio i za kojeg nije izračunata predikcija, ako takvog nema odlazak na korak 6.
3. Odabir k-najsličnijih susjeda koji su ocijenili objekt i.
4. Računanje predikcija ocjena za objekt i na temelju ocjena k-najsličnijih susjeda.
5. Povratak na 2. korak.
6. Ispis n objekata s najvećim predikcijama ocjena.

Za ovaj model moguće je na razne načine zadati funkciju sličnosti između korisnika i izračunati predikciju ocjene, odnosno napraviti agregaciju ocjena najsličnijih korisnika. U definiciji 2.1.1 uzeta je u obzir srednja ocjena korisnika, jer neki korisnici nemaju dobru raspodjelu ocjena, tj. daju samo visoke ili niske ocjene. Ako se dobije predikcija ocjene manja od najmanje moguće ocjene ili veća od najveće ocjene koju korisnik može dodijeliti nekom objektu u sustavu tada se predikcija ocjena postavlja na najmanju odnosno najveću ocjenu. Također, moguće ju je zaokružiti na određeni broj decimala, ali navedeno se treba koristiti samo kod ispisa predikcije ocjene kako bi se dobile konzistentne ocjene u sustavu, jer za izradu liste preporuka potrebne su što točnije predikcije. Slijede primjeri funkcija sličnosti i njihove definicije.

Primjeri funkcija sličnosti:

- L1 sličnost (eng. Manhattan similarity)
- Euklidska sličnost (eng. Euclidean similarity)
- kosinusova sličnost (eng. cosine similarity)
- prilagođena kosinusova sličnost (eng. adjusted cosine similarity)
- Pearsonov koeficijent korelacije (eng. Pearson correlation)
- Spearmanov koeficijent korelacije (eng. Spearman rank correlation)

Definicija 2.1.3. *(L1 i Euklidova sličnost)*
Korisnici su reprezentirani s vektorima pri čemu i-ta pozicija predstavlja ocjenu koju je korisnik dao objektu i. Neka su r_u i r_v vektori koji predstavljaju korisnike u i v. Vektori sadržavaju ocjene samo onih objekata koju su ocijenili korisnik u i v. Formule možemo izraziti i pomoću elemenata matrice ocjena R, pri čemu $R(u, i)$ označava ocjenu koju je korisnik u dodijelio objektu i.
L1 udaljenost je definira s:

$$d_{manhattan} : U \times U \to \mathbb{R}^+, d_{manhattan}(u, v) = \|r_u - r_v\|_1 = \sum_i |R(u, i) - R(v, i)|$$

Euklidova udaljenost je definira s:

$$d_{euclidean} : U \times U \to \mathbb{R}^+, d_{euclidean}(u, v) = \|r_u - r_v\|_2 = \sqrt{\sum_i (R(u, i) - R^2(v, i))}$$

Funkcije udaljenosti poprimaju pozitivne vrijednosti. Što je udaljenost manja to je sličnost veća i obratno, pa na temelju tih udaljenosti definiraju se sljedeće sličnosti.

$$s_{manhattan} : U \times U \to [0, 1], s_{manhattan}(u, v) = 1 + \frac{1}{1 + d_{manhattan}(u, v)}$$

$$s_{euclidean} : U \times U \to [0, 1], s_{euclidean}(u, v) = 1 + \frac{1}{1 + d_{euclidean}(u, v)}$$

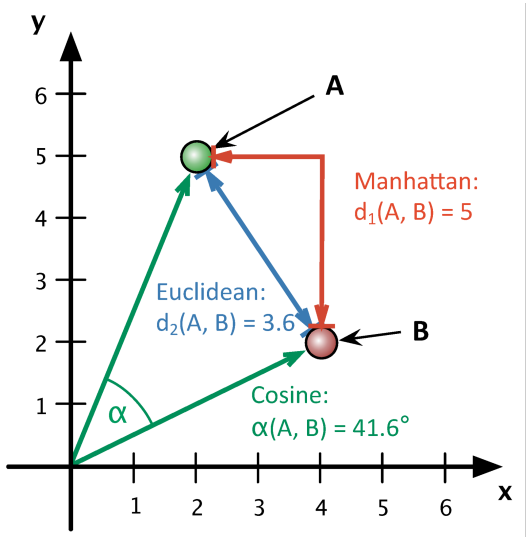
Definicija 2.1.4. *(kosinusova i prilagođena kosinusova sličnost)*
Za razliku od prethodne definicije vektori r_u i r_v sadržavaju sve ocjene. Ako korisnik nije dao ocjenu objektu i, tada se na toj poziciju zapisuje 0. Kosinusova sličnost zadana je formulom:

$$s_{cosine} : U \times U \to [0,1], s_{cosine}(u,v) = \frac{r_u \cdot r_v}{\|r_u\|_2 \, \|r_v\|_2} = \frac{\sum_i R(u,i)\, R(v,i)}{\sqrt{\sum_i R^2(u,i)}\ \sqrt{\sum_i R^2(v,i)}}$$

U prilagođenoj kosinusovoj sličnosti uzima se u obzir i prosječna ocjena korisnika. Zadana je sljedećom formulom:

$$adjcosine : U \times U \to [-1,1],$$
$$adjcosine(u,v) = \frac{(r_u - \overline{r_u}) \cdot (r_v - \overline{r_v})}{\|(r_u - \overline{r_u})\|_2 \, \|(r_v - \overline{r_v})\|_2} = \frac{\sum_i (R(u,i) - \overline{r_u})(R(v,i) - \overline{r_v})}{\sqrt{\sum_i (R(u,i) - \overline{r_u})^2}\ \sqrt{\sum_i (R(v,i) - \overline{r_v})^2}},$$

pri čemu je $\overline{r_u}$ aritmetička sredina svih ocjena korisnika u i $\overline{r_v}$ je aritmetička sredina svih ocjena korisnika v.



Slika 2.1. Interpretacija različitih funkcija udaljenosti (preuzeto sa: `http://dh2016.adho.org/abstracts/253` [pristupljeno 17. kolovoza 2017.])

Kosinusova sličnost poprima vrijednosti između 0 i 1, gdje 1 označava potpunu podudarnost (kut od 0 stupnjeva), a 0 najmanju sličnost (pravi kut). Ne može poprimiti negativne vrijednosti, jer pretpostavljamo da korisnici dodjeljuju samo pozitivne ocjene.

Prilagođena kosinusova sličnost poprima vrijednosti između −1 i 1, a može se normalizirati tako da poprima vrijednosti između 0 i 1.

$$s_{adjcosine} : U \times U \to [0,1], s_{adjcosine} = \frac{1}{2}(adjcosine(u,v) + 1)$$

Definicija 2.1.5. *(Pearsonov koeficijent korelacije)*
Neka su $u, v \in U$ neki korisnici i I_u, I_v skup objekata koje je ocijenio korisnik u, odnosno v. C je skup objekata koji su ocijenili oba korisnika, $C := I_u \cap I_v$. Pearsonov koeficijent korelacije zadan je na sljedeći način.

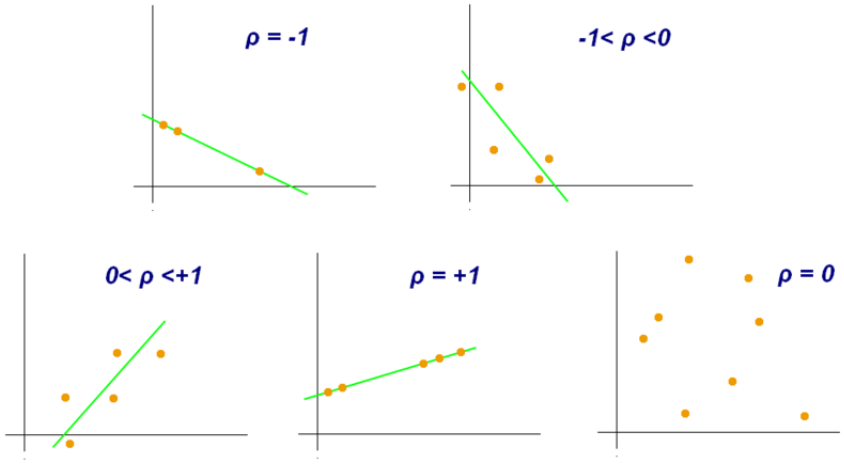
$$pearson : U \times U \to [-1,1],$$

$$pearson(u,v) = \frac{n\sum_C R(u,i)R(v,i) - \sum_C R(u,i)\sum_C R(v,i)}{\sqrt{n\sum_C R^2(u,i) - (\sum_C R(u,i))^2}\sqrt{n\sum_C R^2(v,i) - (\sum_C R(v,i))^2}}$$

Pearsonov koeficijent korelacije je mjera linearne korelacije između varijabli X i Y. Raspon vrijednosti je od −1 do 1, gdje 0 predstavlja da nema linearne korelacije, dok 1 predstavlja potpunu pozitivnu linearnu korelaciju, odnosno −1 potpunu negativnu lineranu korelaciju. Kad bi se u prilagođenoj kosinusovoj sličnosti računalo samo sa zajedničkim ocjenama dobio bi se upravo Pearsonov koeficijent. Problem je što se kod malog broja zajedničkih ocjena dobiva velika sličnost. Za jednu zajedničku ocjenu sličnost nije definirana, za dvije dobiva se vrijednost 1. Zato se koeficijent množi s faktorom $\frac{min(|C|,50)}{50}$ (Herlocker i sur., 1999.), pri čemu je $|C|$ zajednički broj ocjena. Kod prilagođene kosinusove sličnosti ne javlja se ovaj problem jer se uzimaju u obzir sve ocjene pri čemu se smokorigira s faktorom $\frac{C}{\sqrt{|I_u|}\sqrt{|I_v|}}$. Pearsonov koeficijent nije definiran ako su ocjene zajedničkih objekata jednog od korisnika sve jednake. Funkcija se može normalizirati tako da poprima vrijednosti između 0 i 1.

$$s_{pearson} : U \times U \to [0,1], s_{pearson} = \frac{1}{2}(pearson(u,v) + 1) \cdot \frac{min(|C|, 50)}{50}$$

Spearmanov koeficijent korelacije računa se kao Pearsonov koeficijent korelacije, samo su ulazne vrijednosti rangovi ocjena, a ne same ocjene. Najvećoj ocjeni dodijeljen je rang 1, sve manjim ocjenama dodijeljeni su sve veći rangovi.



Slika 2.2. Prikaz interpretacije vrijednosti Pearsonove korelacije (preuzeto sa: `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient` [pristupljeno 16. kolovoza 2017.])

Sustavi temeljeni na sličnosti objekata

Od sustava za davanje preporuka očekuje se da u što kraćem vremenu kreiraju preporuke. Kod sustava s većim brojem korisnika i objekata radi ubrzanja poželjno je unaprijed izračunati sličnosti. Sličnosti između korisnika su podložnije promjenama pa je bolje unaprijed izračunati sličnosti između objekata. Objekti su sličniji ako imaju podjednake ocjene korisnika. Dana je definicija predikcije kod sustava temeljenih na sličnosti objekta i općeniti algoritam za njezino računanje.

Definicija 2.1.6. Neka je U skup korisnika, I skup objekata. Korisnici ocjenjuju pojedini objekt što je dano funkcijom $r : U \times I \to \mathbb{N}$. Vrijednosti te funkcije zapisane su u matrici ocjena R. Ako korisnik nije ocijenio neki objekt, tada je vrijednost funkcije r jednaka 0. $N_{i,u} \subseteq I$ je skup k-najsličnijih objekata objektu i koje je ocijenio korisnik u. Predikcija ocjene koju bi dao korisnik u objektu i računa se aritmetičkom sredinom s težinama, pri čemu su težine zadane funkcijom sličnosti objekata $s : I \times I \to \mathbb{R}$.

$$p : U \times I \to \mathbb{R}^+, p(u,i) = \frac{\sum_{i' \in N_{i,u}} s(i,i')R(u,i')}{\sum_{i' \in N_{i,u}} s(i,i')}$$

Algoritam 2.1.7. *(izrada liste preporuka u sustavima temeljenima na sličnosti objekata)*
Ulaz: Broj korisnika u, matrica ocjena R, funkcija sličnosti između objekata s, broj najsličnijih susjeda k, broj preporuka n
Izlaz: Lista preporuka s n objekata koji imaju najbolju predikciju ocjena.

1. Za svaki objekt odrediti k-najsličnijih objekata na temelju zadane funkcije sličnosti s.
2. Odabir objekta i kojeg korisnik u nije ocijenio i za kojeg nije izračunata predikcija, ako takvog nema odlazak na korak 5.
3. Računanje predikcija ocjena za objekt i na temelju sličnosti k-najsličnijih objekata i ocjena koje je korisnik u dodijelio tim objektima.
4. Povratak na 2. korak.
5. Ispis n objekata s najboljim predikcijama.

I u ovom modelu moguće je na razne načine definirati funkciju sličnosti. Definiraju se analogno kao u sustavima temeljenim na sličnosti korisnika. U praksi se najbolje pokazala prilagođena kosinusova sličnost.

Definicija 2.1.8. *(prilagođena kosinusova sličnost za određivanje sličnosti između objekata)*
Neka su i i j neki objekti. Skup $U_{i,j} \subseteq U$ sadrži sve korisnike koji su ocijenili oba objekta. Prilagođena kosinusova sličnost zadana je formulom:

$$s_{adjcosine} : I \times I \to [0, 1],$$

$$s_{adjcosine}(i, j) = \frac{1}{2}\left(\frac{\sum_{u \in U_{i,j}} (R(u, i) - \overline{r_u})(R(u, j) - \overline{r_u})}{\sqrt{\sum_{u \in U_{i,j}} (R(u, i) - \overline{r_u})^2}\ \sqrt{\sum_{u \in U_{i,j}} (R(u, j) - \overline{r_u})^2}} + 1\right),$$

pri čemu je $\overline{r_u}$ aritmetička sredina svih ocjena korisnika u.

Računanje najsličnijih susjeda

Kod izračuna predikcije može se unaprijed zadati koliko će se najsličnijih susjeda uzeti u obzir ili se može zadati neki prag (eng. threshold). Svi korisnici ili objekti koji su sličniji od zadanog praga ulaze u izračun predikcije. Postavlja se pitanja koji prag uzeti, prag ovisi i funkciji udaljenosti te postoji mogućnost da neće dovoljno korisnika ili objekata prijeći

prag, pa se u praksi češće upotrebljava računanje k-najsličnijih susjeda. Postoje razni algoritmi za njihovo određivanje. Dijelimo ih na egzaktne (eng. exact) i približne (eng. approximate). Kod egzaktnih metoda računaju se sve sličnosti i pamti se k najsličnijih. Često se koriste stabla particije prostora, kao što su "metric-trees", "Kd-trees", "ball-trees" i sl. Koriste se još i razna stabla pretrage i hrpe. Postoji razne implementacije koje se razvijaju i poboljšavaju jer je njihova primjena vrlo široka. Približni algoritmi se koriste kako bi se još više ubrzao rad sustava, najpoznatiji je "lokalno-osjetljivo haširanje" (eng. locality-sensitive hashing), ali se gubi na točnosti predikcije.

"Offline" računanje sličnosti između objekata

Da bi se dodatno ubrzao rad sustava, moguće je unaprijed izračunati sličnosti između korisnika ili objekata. Korisnici imaju relativno mali broj zajednički ocijenjenih objekata, pa mali broj novih ocjena može znatno utjecati na njihove sličnosti. Dok objekti imaju veći broj zajedničkih ocjena što čini sličnost između objekata stabilnijom. Stoga je bolje unaprijed računati sličnosti između objekata kako se ne bi znatno izgubilo na točnosti predikcija. Da bi se smanjila vremenska i prostorna složenost, moguće je spremati sličnosti samo k-najsličnijih objekata. Tada postoji mogućnost da se za neke korisnike i neki objekt neće moći izvršiti predikcija jer korisnik nije ocijenio niti jedan njegov najsličniji objekt ili ih je ocijenio jako malo pa će se smanjiti točnost predikcija. Kako bi se sadržala što veća točnost, a smanjila vremenska i prostorna složenost, razvijene su metode kod kojih se na temelju svih ocjena prvo izradi model pa se na temelju njega računaju predikcije i kreiraju preporuke.

2.2. Sustavi temeljeni na modelu

Dva prethodno opisana sustava temelje svoje predikcije na svim ocjenama korisnika, tj. na cijeloj bazi korisnika i objekata, pa za takve sustave kažemo da su temeljeni na memoriji. Sustavi temeljeni na modelu izrađuju model pomoću algoritama rudarenja podataka (eng. data mining) i strojnog učenja (eng. machine learning) nad podacima. Prednost ovakvog pristupa je što se postiže bolja skalabilnost (brže vrijeme izvršavanja i manje memorije kod većih sustava) i dobivaju se bolje predikcije na raspršenim podacima. Nedostatak je taj što je potrebna zahtjevna izrada modela i potrebno je osvježavati model s novim podacima.

Neki od algoritama za izradu modela:

- matrična faktorizacija (eng. matrix factorization) - redukcija dimenzije (eng. dimensionality reduction) - analiza značajki (eng. factor analysis, latent search)
  - dekompozicija singularnih vrijednosti - SVD (eng. singular value decomposition)
  - analiza glavnih componenti - PCA (eng. principle component analysis)
  - stohastički gradijentni spust - SGD (eng. stochastic gradient descent)
  - alternirajući najmanji kvadrati - ALS (eng. alternating least square)
  - nenegativna matrična faktorizacija - NMF (eng. nonnegative matrix factorization)
- grupiranje podataka (eng. clustering)
- asocijacijska pravila (eng. association rule)
- klasifikacija (eng. classification)
- stabla odluke (eng. decision tree)
- Baysove mreže (eng. Bayesian network)
- neuronske mreže (eng. neural network)

2.3. Matrična faktorizacija

Metode matrične faktorizacije upotrebljavaju se kako bi se pronašao skup latentnih (skrivenih) faktora i karakterizirali objekti i preferencije korisnika pomoću tih faktora. Time se dobiva redukcija dimenzije i ubrzanje rada i smanjenje prostora. Kod filmova to mogu biti i neki očiti faktori, poput žanra filma, ali mogu biti i neki teži za interpretacije. Najpopularnija metoda matrične faktorizacije je SVD dekompozicija. Koristila se kod otkrivanja latentnih faktora kod dokumenata (Deerwester i dr., 1990.) u metodi LSA (eng. latent semantic analysis) ili LSI (latent semantic indexing). Korisnik je zadao upit pomoću ključnih riječ pa je trebalo pronaći odgovarajuće dokumente. Problem nastaje zbog sinonima (za jedno značenje više riječi) i polisemija (jedna riječ više značenja), što je riješeno SVD dekompozicijom koja se kasnije primjenjuje i na druga područja.

Općenito kod matrične faktorizacije $n \times m$ matrice M želi se rastaviti na $n \times k$ matricu P i $k \times m$ matricu Q tako da je $M = PQ^T$. Kod primjene faktorizacije matrice ocjena R dobije se prikaz korisnika i objekata u prostoru s k značajki. Ako se želi reducirati dimenzije, tada se matrica R aproksimira $n \times d$ matricom $\tilde{R}$ i $d \times m$ matricu $\hat{R}$, pa je $R = \tilde{R}\hat{R}^T$ (eng. low rank matrix factorization).

Dekompozicija singularnih vrijednosti

Kako sustav ima puno korisnika i objekata matrica R je velikih dimenzija. Redukcijom dimenzija dobivamo bolje performanse i rješavamo se takozvanog šuma te se mogu otkriti skrivene značajke. Dana je definicija singularne dekompozicije matrice i kako se primjenjuje u redukciji dimenzije i davanju preporuka.

Teorem 2.3.1. *(dekompozicija singularnih vrijednosti)*
Neka je M realna matrica, $M \in \mathbb{R}^{n \times m}$. M se može zapisati kao $M = U\Sigma V^*$, pri čemu su $U \in \mathbb{R}^{n \times n}$ i $V* \in \mathbb{R}^{m \times m}$ ortogonalne matrice. Matrica Σ je $n \times m$ dijagonalna matrica pri čemu su dijagonalni elementi σ_i nenegativni realni brojevi poredani od najvećeg prema najmanjem i zovemo ih singularnim vrijednostima matrice M. Matrica ranga k ima k singularnih vrijednosti, pa je $\sigma_i = 0$ za $i > k$.
Stupce matrice U zovemo lijevi singularni vektori matrice M, a stupce matrice V desni singularni vektori matrice M. Lijevi singularni vektori jednaki su ortonormiranim svojstvenim vektorima matrice $MM*$, desni singularni vektori jednaki su ortonormiranim svojstvenim vektorima matrice $M * M$, a singularne vrijednosti su korijeni nenegativnih svojstvenih vrijednosti matrice $MM*$ (ili isto dobivamo ako promatramo matricu $M * M$).
Kako je M realna matrica tada je $V^* = V^T$ i V je ortogonalna, $VV^T = 1$.

Neka su u daljnjem tekstu n broj korisnika, a m broj objekata. Matrica ocjena $\overline{R}$ je normalizirana matrica (svakom retku se oduzme aritmetička sredina pripadnog retka). Pomoću SVD dekompozicije otkrivamo d značajki i dobivamo matrice korisnik-značajke i matricu objekt-značajke. Koristeći Frobeniusovu normu ($||M||_F := \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |m_{i,j}|^2}$) kao mjeru greške dobije se da je singularna dekompozicija najbolja aproksimacija s matricom ranga d.

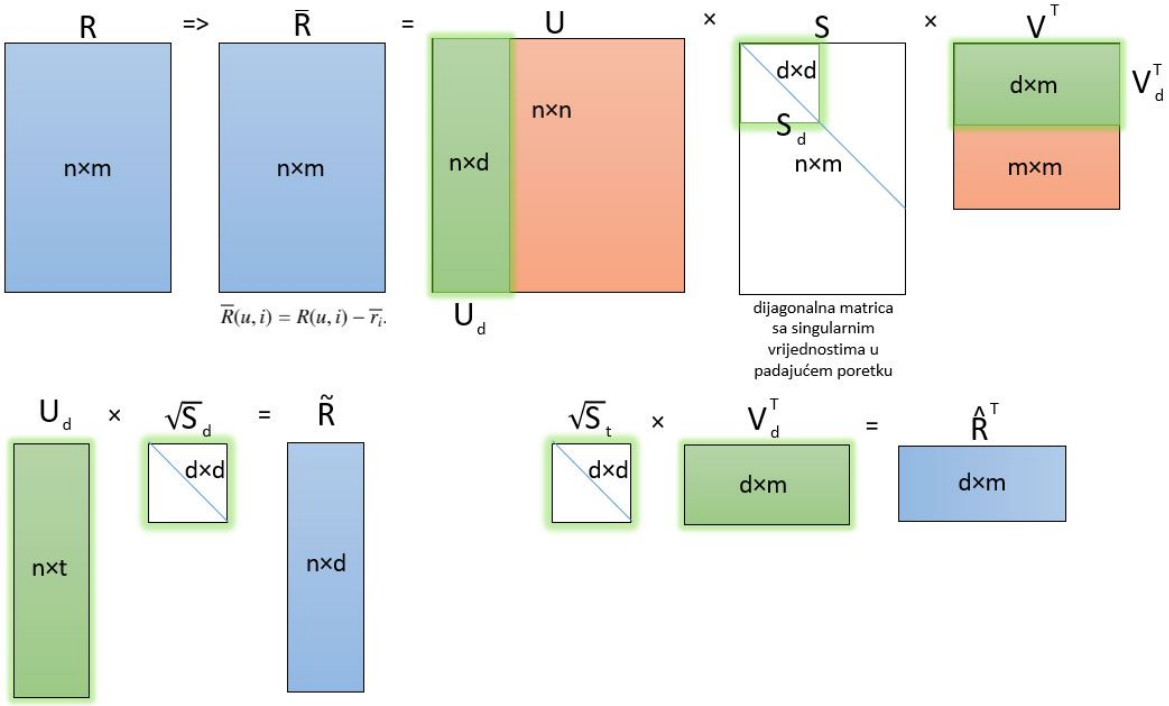
Teorem 2.3.2. *(aproksimacija matrice dekompozicijom singularnih vrijednosti)*
Neka je $\overline{R}$ matrica dimenzija $m \times n$. Prema teoremu 2.3.1 $\overline{R}$ se može zapisati kao $\overline{R} = U\Sigma V^T$.

Neka je d neki prirodni broj, $d < rang(\overline{R})$. Aproksimacija matrice $\overline{R}$ dekompozicijom singularnih vrijednosti dana je s $\overline{R} \approx U_d S_d V_d^T = \tilde{R}$, pri čemu matrica U_d sadrži prvih d stupaca matrice U, S_d je dijagonalna matrica koja sadrži d najvećih singularnih vrijednosti i matrica V_d^T sadrži prvih d redaka matrice V^T.

Potrebno je definirati preslikavanje objekata i korisnika u novi d-dimenzionalni prostor. Dobije se matrica korisnik-značajke $\tilde{R}$ i matrica objekt-značajke $\hat{R}$.

Definicija 2.3.3. *(SVD redukcija dimenzije matrice ocjena R)*
Neka je R matrica ocjena, $\overline{R}$ njena normalizirana matrica. $\overline{R}$ se može aproksimirati s $\overline{R} \approx \tilde{R}\hat{R}^T$. Gdje je d dimenzija prostora značajki. Korisnici u prostoru značajki $\tilde{R}$ i objekti u tom prostoru $\hat{R}$ dani su s:

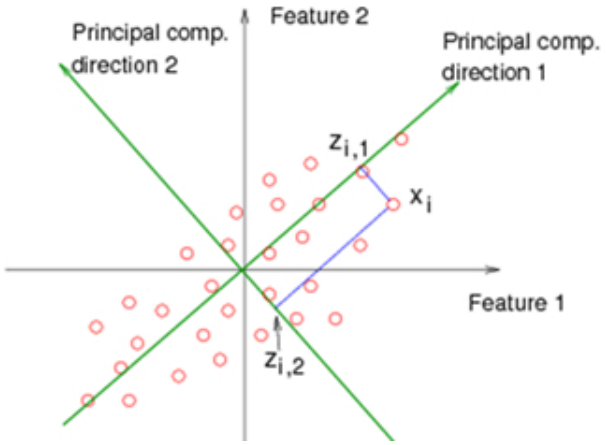
$$\tilde{R} = U_d \sqrt{S_d}, \ \hat{R}^T = \sqrt{S_d} V^T$$



Slika 2.3. Prikaz redukcije dimenzije matrice ocjena pomoću dekompozicije singularnih vrijednosti

Analiza glavnih komponenti

Karl Person prvi je dao opis ove metode 1901. godine. Ovom metodom se dobiva redukcija dimenzije kao kod SVD dekompozicije samo u terminima multivarijatne statistike. Cilj je pronaći novi koordinatni sustav s manjim brojem dimenzija od izvornog s najmanjim gubitkom podataka.



Slika 2.4. Za novi koordinatni sustav može se uzeti samo prva glavna komponenta s najmanjim gubitkom podataka (preuzeto sa: `https://onlinecourses.science.psu.edu` [pristupljeno 25. kolovoza 2017.])

Za primjenu analize glavnih komponenti u sustavima za davanje preporuka dane su interpretacije pojmova iz statistike. Neki korisnik i ocjenjuje objekte i to zapisujemo kao slučajnu varijablu X_i, tj. X_i je vektor dimenzije m pri čemu su njegovi elementi ocjene korisnika. $S = (X_1, X_2, ..., X_n)$ je skup uzoraka (eng. samples) koji sadrži n slučajnih varijabli (eng. random variable ili stochastic variable). μ_i označava aritmetičku sredinu $\mu_i = \frac{1}{n}\sum_{j=1}^{n} X_j(i)$, to jest aritmetičku sredinu ocjena koje su dodijeljene objektu i, dok je $\mu = (\mu_1, \mu_2, ..., \mu_m)$ vektor koji sadrži te aritmetičke sredine.

Definicija 2.3.4. *(varijanca skupa uzoraka)*
Varijanca slučajne varijable X je $Var(X) = E[(X - E[X])^2]$. Kako se radi o diskretnoj slučajnoj varijabli s uniformnom distribucijom vjerojatnosti tada je

$$Var(X) = \frac{1}{n}\sum_{i=1}^{n}(X(i) - \overline{X}))^2,$$

gdje je $\overline{X}$ aritmetička sredina slučajne varijable X.
Kako varijanca uzorka nije dobra, procjena varijance populacije definira se korigirana varijanca (eng. biased sample variance) kao

$$Var'(X) = \frac{1}{n-1}\sum_{i=1}^{n}(X(i) - \overline{X}))^2.$$

Varijanca skupa uzoraka S je m-dimenzionalni vektor V dan izrazom:

$$V = \begin{bmatrix} Var'(X_1) \\ Var'(X_1) \\ \vdots \\ Var'(X_n) \end{bmatrix} = \begin{bmatrix} \frac{1}{n-1}\sum_{i=1}^{n}(X_i(1) - \mu(1))^2 \\ \frac{1}{n-1}\sum_{i=1}^{n}(X_i(2) - \mu(2))^2 \\ \vdots \\ \frac{1}{n-1}\sum_{i=1}^{n}(X_i(m) - \mu(m))^2 \end{bmatrix}$$

Komponente vektora V su mjere raspršenosti skupa uzoraka duž svake osi. Potrebno je pronaći os (komponentu) duž koje je varijanca podataka najveća kako bi gubici bitnih podataka bili najmanji, a tu os nazivamo prva glavna komponenta. Svaka sljedeća glavna komponenta je okomita na prethodnu čime dobijemo ortogonalnu bazu. Odabiremo t glavnih komponenti čime smo dobili d- dimenzionalni prostor. Za određivanje glavnih komponenti potrebno je odrediti kovarijacijsku matricu uzoraka C te odrediti njene svojstvene vrijednosti i jedinične svojstvene vektore.

Definicija 2.3.5. *(kovarijacijska matrica uzorka)*
Za dvije slučajne varijable X i Y definirana je kovarijanca uzoraka (eng. sample covariance) s:

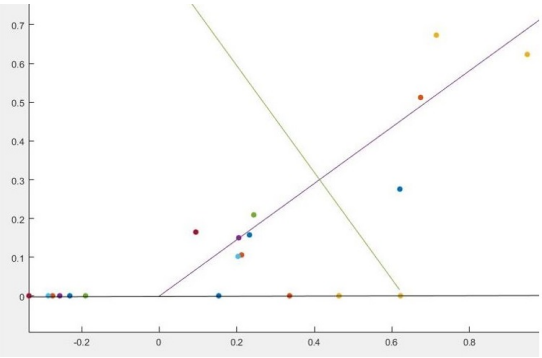
$$Cov(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(X(i) - \overline{X})(Y(i) - \overline{Y})$$

Matrica kovarijanci C uzorka S dana je s:

$$C(i, j) = Cov(X_i, X_j) = \frac{1}{n-1}\sum_{l=1}^{n}(X_l(i) - \mu(i))(X_l(j) - \mu(j))$$

Formula za matricu C zapisana pomoću matrice ocjena R je:

$$C(i, j) = \frac{1}{n-1}\sum_{u=1}^{n}(R(u, i) - \overline{r_i})(R(u, j) - \overline{r_j}),\ \ C = \frac{1}{n-1}\overline{R}^T\overline{R},$$



Slika 2.5. Primjer redukcije deset 2-dimenzionalnih točaka u 1-dimenzionalni prostor. Ljubičasti pravac prikazuje prvu glavnu komponentu, a zeleni koji je okomiti na nju drugu komponentu. Vrši se ortogonalna projekcija točaka na glavnu komponentu te se na x-osi (crni pravac) prikazuju njene vrijednosti u 1-dimenziji.

pri čemu je $\overline{r_i}$ aritmetička sredina svih ocjena koje je dobio objekt i, a $\overline{r_j}$ aritmetička sredina svih ocjena koje je dobio objekt j:

$$\overline{r_i} = \frac{\sum_{u \in U} R(u,i)}{card(u \in U | R(u,i) \neq 0)}$$

$\overline{R}$ je normalizirana matrica matrice R gdje je $\overline{R}(u,i) = R(u,i) - \overline{r_i}$.

Definicija 2.3.6. *(svojstveni vektor i svojstvena vrijednost)*
Netrivijalni vektor x dimenzije n ($x \in \mathbb{R}^n$ i $x \neq 0$) zovemo svojstveni vektor realne kvadratne matrice A, $A \in \mathbb{R}^{n \times n}$, ako vrijedi $Ax = \lambda x$ za neki skalar $\lambda \neq 0$. Skalar λ ako postoji onda je jedinstven i zovemo ga svojstvena vrijednost matrice A pridružena svojstvenom vektoru x.

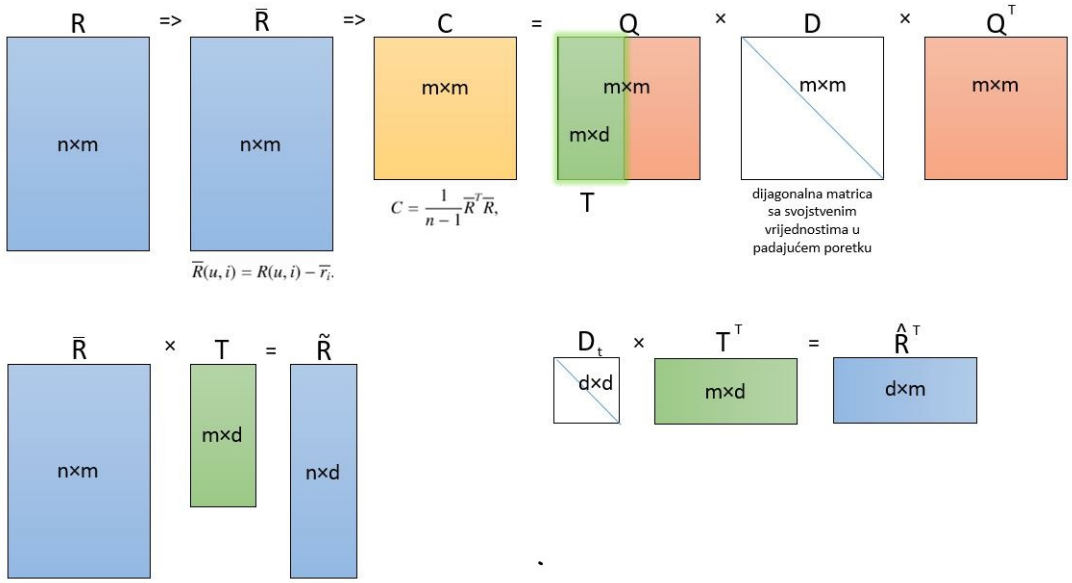
Teorem 2.3.7. *(dekompozicija svojstvenih vrijednosti ili spektralna dekompozicija, eng. eigenvalue decomposition ili spectral decomposition)*
Neka je A realna kvadratna matrica, $A \in \mathbb{R}^{n \times n}$, koja ima n linearno nezavisnih svojstvenih vektora $q_i, i = 1, 2, 3, ..., n$. Tada se matrica A može zapisati kao $A = Q\Lambda Q^{-1}$, pri čemu je Q matrica, sa stupcima koji su jednaki svojstvenim vrijednostima, a Λ je dijagonalna matrica gdje je i-ti dijagonalni element jednak svojstvenoj vrijednosti λ_i matrice A pridružena svojstvenom vektoru q_i.
Ako je A realna simetrična kvadratna matrica tada se ona može zapisati kao $A = Q\Lambda Q^T$,

gdje je Q ortogonalna matrica i svojstvene vrijednosti jednake su singularnim vrijednostima.

Matrica C je realna simetrična kvadratna matrica i može se prema teoremu 2.3.7 zapisati kao $C = Q\Lambda Q^T$. Bazu d-dimenzionalnog prostora čine jedinični svojstveni vektori koji pripadaju d najvećim svojstvenim vrijednostima koji se postavljaju u stupce matrice T pomoću koje se vrši transformacija u novi prostor.

Definicija 2.3.8. *(PCA redukcija dimenzije matrice R)*
Neka je R matrica ocjena, $\overline{R}$ njena normalizirana matrica i C njena kovarijacijska marica uzorka koja se može zapisati kao $C = QDQ^T$ tako da D sadrži svojstvene vrijednosti u padajućem poretku. Matrica transformacije T sadrži prvih d stupaca matrice Q, odnosno d ortonormiranih svojstvenih vektora koji pripadaju d najvećim svojstvenim vrijednostima. Matrica korisnik-značajke $\tilde{R}$ dimenzije $n \times d$ i matrica objekt-značajke $\hat{R}$ dimenzije $n \times d$ zadane su s:

$$\tilde{R} = \overline{R}T, \;\; \hat{R}^T = D_d T^T$$



Slika 2.6. Prikaz redukcije dimenzije matrice ocjena pomoću analize glavnih komponenti

Davanje preporuka u reduciranom prostoru

Dobiveni $d \times d$ reducirani prostor zovemo prostor značajki. Matricu ocjena R naziva se još matrica korisnik-objekti jer redci predstavljaju korisnike, a stupci objekte. Dobivena matrica $\tilde{R}$ je matrica korisnik-značajke. U prostoru značajki možemo prikazati i objekte pomoću matrice $\hat{R}$. Redukcijom dimenzije mogu se otkriti značajke i otkloniti šumovi. Ovisno o domeni primjene treba eksperimentalno odrediti na koju dimenziju je poželjno reducirati da se pritom izgubi što manje podataka kako bi predikcija bila što točnija, a istodobno da je algoritam što efikasniji. SVD ili PCA redukcija se vrši u "offline" fazi. Nakon što smo prikazali korisnike i/ili objekte u prostoru značajki moguće je napraviti predikciju ocjena na više načina:

(1) Dobiveni reducirani prostor koristiti se za izračun sličnosti kao u sustavima temeljenim na memoriji. U "ofline" fazi mogu se odrediti i susjedi, ali postoji mogućnost da nema dovoljno korisnika koji su ocijenili neki objekt potrebnan za predikciju.
(1.a) Matrica $\tilde{R}$ može se koristiti za brže određivanje sličnosti između korisnika pa se na temelju k najsličnijih korisnika u "online" fazi vrši predikcija ocjena kao kod sustava temeljenih na sličnosti korisnika.
(1.b) Matrica $\hat{R}$ može se koristiti za brže računanje sličnosti između objekata pa se na temelju k-najsličnijih objekata u "online" fazi vrši predikcija ocjena kao kod sustava temeljenih na sličnosti objekata.

(2) Kako se korisnici i objekti mogu prikazi u istom prostoru značajki, moguće je odrediti sličnosti između korisnika i objekta nekom od funkcija sličnosti, te se preporučuju n najsličniji objekti. U ovom načinu ne vrši se predikcija ocjena.

(3) Kod SVD metode množenjem matrica korisnik-značajki $\tilde{R}$ i matrice $\tilde{R}$ dobivamo aproksimaciju matrice R te još trebamo dodati prijašnje oduzete aritmetičke sredine stupaca. Umjesto nula sad imamo neke vrijednosti koje možemo koristiti kao predikciju ocjena.

$$p(i, j) = r_j + \tilde{R}\hat{R}^T$$

Kod PCA metode predikcija se dobiva na sljedeći način: $p(i, j) = r_j + \tilde{R}T^T$

Popunjavanje nedostajućih vrijednosti matrice ocjena

U praksi javlja se problem da matrica ocjena sadrži mnogo nula. Nula u matrici znači da korisnik nije ocijenio taj objekt. Kod SVD i PCA metode možemo postaviti na nedostajuća mjesta (umjesto nule) prosječna ocjena objekta. Kod PCA metode prilikom izrade kovarijacijske matrice mogu se ignorirati ta mjesta, ali tada dobivena matrica ne mora biti simetrična i možda nije moguće napraviti dekompoziciju na svojstvene vrijednosti. Stoga su razvijene metode poput vjerojatnosne analize glavnih komponenti (eng. probabilistic principal component analysis) i iterativna analiza glavnih komponenti (eng. iterative pca). Kod metoda gradijentnog spusta i alternirajućih najmanjih kvadrata za nedostajuće vrijednosti postavlja se slučajne brojeve te je cilj iterativnim postupkom smanjiti grešku koja se promatra samo na skupu ocijenjenih objekata.

Stohastički gradijentni spust

Matricu ocjena R želimo aproksimirati matricom $\hat{R}$ ($R \approx \hat{R} = PQ^T$). Za razliku od prošlih metoda tu se promatra greška na skupu ocijenjenih objekata kako nedostajuće vrijednosti ne bi preuzele vodeću ulogu i tako smanjile točnost predikcija. Definira se kvadratna greška jer predikcija može biti veća ili manja od stvarne ocjene.

Definicija 2.3.9. *(kvadratna greška ili kvadrat reziduala)*
Neka je R matrica ocjena, a $\hat{R}$ matrica predikcija ocjena. Kvadratna greška zadana je s:

$$e_{i,j}^2 = (r_{i,j} - \hat{r}_{i,j})^2 = (r_{i,j} - p_i q_j^T)^2 = (r_{i,j} - \sum_{k=1}^{d} p_{i,k} q_{k,j})^2$$

Sada trebamo odrediti smjer kojim ćemo mijenjati vrijednosti u matricama kako bi optimizirali, tj. pronašli najmanju grešku. Odabire se smjer duž kojeg se greške najbrže smanjuju što nam daje gradijent odnosno parcijalne derivacije.

$$\frac{\partial}{\partial p_{i,k}} e_{i,j}^2 = -2(r_{i,j} - p_{i,j}) q_{k,j} = -2e_{i,j} q_{k,j}$$

$$\frac{\partial}{\partial q_{k,j}} e_{i,j}^2 = -2(r_{i,j} - p_{i,j}) p_{i,k} = -2e_{i,j} p_{i,k}$$

Nakon što smo odredili smjer promjene, potrebno je odrediti način računanja novih vrijednosti:

$$p'_{i,k} = p_{i,k} + \alpha \frac{\partial}{\partial p_{i,k}} e^2_{i,j} = p_{i,k} + 2\alpha e_{i,j} q_{k,j}$$

$$q'_{i,k} = q_{k,j} + \alpha \frac{\partial}{\partial q_{k,j}} e^2_{i,j} = q_{k,j} + 2\alpha e_{i,j} p_{i,k}$$

Konstanta α označava brzinu promjene koje vršimo. Ako je brzina prevelika, može doći do osciliranja oko minimuma, uobičajeno $\alpha \approx 0.0002$. Kad bi se ukupna greška E računala kao suma svih grešaka $e^2_{i,j}$ tada bi se kao minimum dobila matrica R te bi opet na nedostajućim mjestima bila nula. Ukupna greška koju treba minimizirati računa se kao zbroj kvadrata grešaka ili reziduala (eng. sum of square error - SSE ili residual square error - RSR). Da bi se smanjila greška samo na skupu ocijenjenih objekata uvode se težine $w_{i,j}$. Takvu metodu zovemo stohastički gradijentni spust s težinama (eng. stochastic gradient descent with weights). Smanjenjem zbroja kvadrata grešaka smanjuje se i korijen aritmetičke sredine kvadrata grešaka (eng. root-mean-square error - RMSE ili root-mean-square deviation - RMSE) koja se koristi i za evaluaciju sustava za davanje preporuka.

Definicija 2.3.10. *(zbroj kvadrata grešaka)*
Zbroj kvadrata grešaka (SSE) zadan je sljedećom formulom:

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{m} e^2_{i,j} w_{i,j},$$

$$w_{i,j} = \begin{cases} 0, & R(i,j) = 0 \\ 1, & R(i,j) \neq 0 \end{cases}$$

Kako bi se izbjeglo pretreniranje, uvodi se novi parametar λ, $\lambda \approx 0.001$. Takva metoda naziva se regularizirani stohastički gradijentni spust s težinama (eng. regularized stochastic gradient descent with weights):

$$e^2_{i,j} = (r_{i,j} - p_i q_j^T)^2 + \lambda (\sum_{i=1}^{n} (\|p_i\|^2 + \sum_{j=1}^{m} \|q_j\|^2)$$

$$p'_{i,k} = p_{i,k} + \alpha \frac{\partial}{\partial p_{i,k}} e^2_{i,j} = p_{i,k} + 2(\alpha e_{i,j} q_{k,j} - \lambda p_{i,k})$$

$$q'_{i,k} = q_{k,j} + \alpha \frac{\partial}{\partial q_{k,j}} e^2_{i,j} = q_{k,j} + 2(\alpha e_{i,j} p_{i,k} - \lambda q_{k,j})$$

Neki korisnici daju učestalo visoke ili niske ocjene, a isto tako vrijedi i za objekte. U model je potrebno uvesti pristranost ocjenama. μ predstavlja prosjek svih ocjena, $\overline{r}_j$ prosjek ocjena korisnika i, a $\overline{r}_j$ prosjek ocjena objekata j. Dobije se metoda nazvana pristrani regularizirani stohastički gradijentni spust s težinama (eng. biased regularized stochastic gradient descent with weights) te su predikcije i kvadratne greške dane s:

$$\hat{r}_{i,j} = p_i q_j^T + \mu + \overline{r}_i + \overline{r}_j$$

$$e^2_{i,j} = (r_{i,j} - \hat{r}_{i,j} - \mu - \overline{r}_i - \overline{r}_j)^2 + \lambda(\sum_{i=1}^{n}(||p_i||^2 + \sum_{j=1}^{m} ||q_j||^2 + b_i^2 + b_j^2)$$

Alternirajuća metoda najmanjih kvadrata

Problem stohastičkog gradijentnog spusta nije konveksan, što znači da može imati više lokalnih minimuma. Kod većih dimenzija metoda se pokazala prespora. Alternirajuća metoda najmanjih kvadrata bazira se na tome da se fiksira matrica P i pronalazi optimum mijenjajući matricu Q, zatim se matrica Q fiksira, a mijenja se matrica P, postupak se ponavlja do konvergencije. Fiksiranjem jedne matrice dobije se konveksni kvadratni problem najmanjih kvadrata koji se može riješiti optimalno i efikasno. Na početku prvi redak matrice P postavlja se na prosjek ocjena objekata, dok se preostale vrijednosti postavljaju na male slučajne vrijednosti. Zatim se svaki redak matrice Q postavlja po sljedećem pravilu:

$$q_j = (P^T W^{(j)} P + \lambda I_n)^{-1} P^T W^{(j)} r_j$$

Nakon toga fiksira se matrica Q, a redci matrice P računaju se:

$$p_i = (Q^T W^{(i)} Q + \lambda I_m)^{-1} P^Q W^{(i)} r_i$$

Pri tome je r_i vektor ocjena korisnika i, a r_j vektor ocjena objekata j, $W^{(i)}$ je $n \times n$ dijagonalna matrica s elementima na dijagonali $w^{(i)}_{j,j} = \delta_{i,j}$ i $W^{(j)}$ je $m \times m$ dijagonalna matrica s elementima na dijagonali $w^{(j)}_{i,i} = \delta_{i,j}$. Metoda sadrži težine i parametar λ pa je puni naziv regularizirana alternirajuća metoda najmanjih kvadrata s težinama (eng. regularized alternating least squares with weights).

Nenegativna matrična faktorizacija

Kod prijašnjih matričnih faktorizacija nisu bili zadani posebni zahtjevi na matrice P i Q. Kod sustava za davanje preporuka poželjno je nenegativnu matricu ocjena R aproksimirati nenegativnim matricama korisnik-značajke P i objekt-značajke Q ($R \approx PQ^T, p_{i,j} \geq 0, q_{i,j} \geq 0$), pa je zato razvijena metoda nenegativne matrične faktorizacije. Također želi se smanjiti greška samo na skupu ocijenih objekata, pa se uvode težine i parametar λ kako ne bi došlo do pretreniranja. Problem nje konveksan, ali se može riješiti pomoću alternirajućih najmanjih kvadrata s ograničenjima (eng. alternating constrained least square) ili "blok koordinatni spust" (eng. block coordinate descent).

2.4. Grupiranje podataka

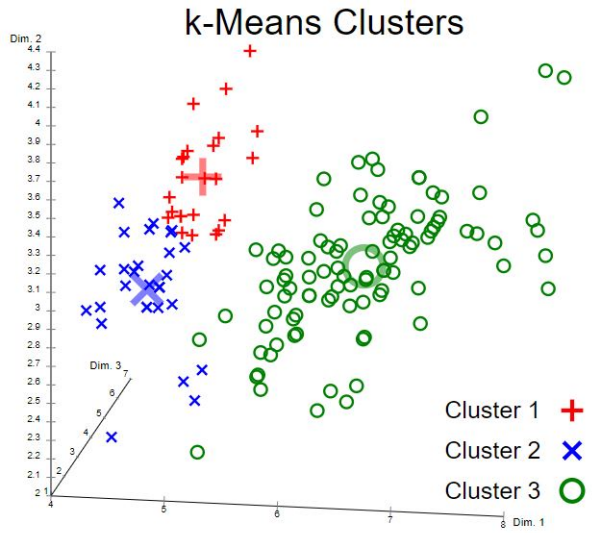
Još jedan mogući način ubrzanja sustava je da se korisnici grupiraju prema sličnosti u takozvane klastere (eng. clusters) te se preporuke daju na razini cijele grupe. Kod takvog pristupa gubi se na personaliziranosti preporuka. Drugi pristup je da se nakon grupiranja koriste metode kao kod sustava temeljenih na memoriji, ali onda umjesto da se računaju k-najsličniji korisnici promatraju se sličnosti korisnika unutar klastera, čime se dobije na ubrzanju. Isto tako mogu se grupirati i objekti, a moguće je prvo izvršiti redukciju dimenzije, a zatim grupirati podatke. Tada je moguća i grupacija korisnika i objekata koji su prikazani u istom prostoru značajki.

Najpoznatija metoda za grupiranje je "k-means". Cilj je podijeliti n podataka u k klastera pri čemu svaki podatak pripada onom klasteru čijem je najbliži promatrajući njegovu sredinu. Problem je NP-težak, ali postoje heurističke metode koje brzo konvergiraju u lokalni optimum. Grupiranje se vrši u "offline" fazi, a kod dodavanja novog korisnika ili objekata određuje se njegova pripadnost klasteru čija je sredina njemu najbliža te se ponovno računa sredina tog klastera.

Definicija 2.4.1. *("k-means" grupiranje podataka)*
Neka je $S = (x_1, x_2, ..., x_n)$ skup m-dimenzionalnih vektora. Cilj "k-means" grupiranja je podjela n-članog skupa u particiju od k skupa, $C = (C_1, C_2, ..., C_k)$ pri čemu se želi

minimizirati sljedeća suma:

$$\sum_{i=1}^{k} \sum_{x \in C_i} ||x - \mu_i||^2 = \sum_{i=1}^{k} |C_i| Var(C_i)$$



Slika 2.7. Prikaz grupiranja u 3 klustera (preuzeto sa: `https://en.wikipedia.org/wiki/K-means_clustering` [pristupljeno 31. kolovoza 2017.])

Predstavnik ove metode je Lloyd-ov algoritam. Algoritam ponavlja dva koraka - pridruživanje i ponovno računanje sredina. Neka je t korak iteracije, a $c_1^{(1)}, c_2^{(1)}, ..., c_k^{(1)}$ početne vrijednosti sredina k klastera.
(1) pridruživanje:

$$C_i^{(t)} = \{x_p : ||x_p - c_i^{(t)}||^2 \leq ||x_p - c_j^{(t)}||^2, \forall j, 1 \leq j \leq k\}$$

(2) ponovno računanje sredina:

$$c_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x_j \in C_i^{(t)}} x_j$$

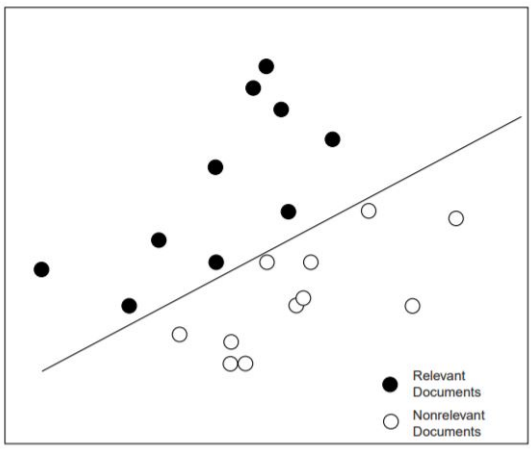
U navedenom algoritmu korisnici se grupiraju po Euklidovoj udaljenosti. Ako se koristi neka druga udaljenost, tada algoritam ne mora konvergirati pa je razvijen algoritam "k-medoids". Za brže računanje u distribuiranim sustavima koristi se algoritam "k-means||". Običan algoritam za odabir početne inicijalizacije na slučajan način s uniformnom distribucijom bira sredine, što može dovesti do lošijeg lokalnog minimuma, stoga se za inicijalizaciju koristi "k-means" algoritam.
(1) Odaberi jednu sredinu c_1 na slučajan način iz S s uniformnom distribucijom.
(2) Odaberi novu sredinu c_i na slučajan način iz S s vjerojatnosti: $\frac{minD^2(x)}{\sum_{x \in S} minD^2(x)}$, minD(x) je najmanja udaljenost između vektora x i već odabranih sredina.
(3) Ponavljaj 2. korak dok nije odabrano k sredina.

Predstavnici algoritama koji se za određivanje klastera baziraju na gustoći podataka su "DBSCAN" (end. density-based spatial clustering of applications with noise) i "OPTICS" (eng. ordering points to identify the clustering structure). Ovi algoritmi svakom podatku pridružuju jedan klaster, što je jednoznačno (tvrdo) grupiranje podataka. Moguće je i meko grupiranje podataka (eng. fuzzy clustering) pri čemu je pripadnost pojedinom klasteru određeno s realnim brojem između 0 i 1. Predstavnici su "C-means" i "FLAME" (eng. fuzzy clustering by local approximation of memberships).

Poglavlje 3.

Sustavi temeljeni na sadržaju objekata

Problem koji se javlja kod sustava temeljenih na suradnji je problem ”cold-start” kada korisnici nisu još ocijenili objekte. Isto tako neće se preporučivati niti novi objekti jer nemaju ocjene za usporedbu. Kako bi se riješio taj problem, izrađuju se sustavi temeljeni na sadržaju objekata. Objekti su opisani pomoću atributa i ključnih riječi. Kod filmova ti atributi mogu biti na primjer žanr, godina, redatelj i glumci. Kod takvih sustava i dalje ostaje otvoreni problem novih korisnika koji nemju izrađeni profil. Takvi korisnici mogu eksplicitno izraziti svoje preferencije.



Slika 3.1. Linearni klasifikator (D. Jannach i dr., Recommender systems - An introduction. 2011.)

Problem hoće li se nekom korisniku svidjeti neki objekt može se opisati kao problem klasifikacije. Cilj je pojedini dokument klasificirati kao ”sviđa mi se” ili ”ne sviđa mi se” ili kao ”relevantan” ili ”nije relevantan”. Klasifikacija se može postići linearnom klasifikatorom pri čemu se pravcem razdvajaju te dvije klase. U kategoriju linearnih klasifikatora pripada naivni Bays (eng. naive Bays), Rocchijeva metoda, Widrow-Hoff algoritam, metoda potpornih vektora - SVM (eng. support vector machines). Klasifikacija pomoću k-najbliži susjeda ne pripada linearnim klasifikatorima.

Kod preporučivanja filmova, knjiga, glazbe, video uradaka potrebno je navesti atribute i ključne riječi koji opisuju njihov sadržaj. Ključne riječi koje opisuju neki dokument mogu se automatski izraditi.

Prikaz dokumenata

Za određivanje sličnosti između dokumenata moguće je napraviti automatsku obradu ključnih riječi u dokumentu. Može se za svaku riječ zapisati 0 da se ne pojavljuje u dokumentu ili 1 da se pojavljuje, čime se dobije binarni vektor. No, nema svaka riječ istu važnost u dokumentu, neke riječi se češće pojavljuju od ostalih. Zato se koristi standard ”TF-IDF” (eng. term frequency-inverse document frequency). Dokumenti se prikazuju u višedimenzionalnom Euklidskom prostoru gdje dimenzije čine ključe riječi. Frekvencija riječi $freq(t, j)$ (eng. frequency) označuje broj pojavljivanja riječi t u dokumentu j. Kako duljina dokumenta ne bi utjecala na relevantnost frekvencija, mjera $TF(t, j)$ (eng. term frequency) je omjer $freq(t, j)$ i maksimalne frekvencijom ostalih riječi $maxOther(t, j) = max\{freq(w, j) : w \in D_j\}$, gdje D_j označava skup svih riječi koje se nalaze u dokumentu j.

$$\text{TF}(t, j) = \frac{freq(t, j)}{maxOther(t, j)}$$

Neka m označava broj dokumenta, a $m(t)$ broj dokumenata koji sadrže riječ t. Inverzna frekvencija dokumenta (eng. inverse document frequency) je mjera koja manju važnost stavlja na riječi koje se pojavljuju na više dokumenata, jer tada znači da su one manje relevantne za pojedini dokument. Mjera je zadana sjedećom formulom:

$$\text{IDF} = log\frac{m}{m(t)}$$

Kombinirana mjera ”TF-IDF” računa se kao umnožak ”TF” i ”IDF” mjere:

$$\text{TF-IDF}(t, j) = TF(t, j) \cdot IDF(t, j)$$

Dokument se tada umjesto binarnog vektora prikazuje pomoću vektora s ”TF-IDF” mjerom za pojedinu riječ. Problem koji se javlja je da taj vektor može biti jako velik. To se može riješiti na različite načine: izbacivanjem veznika, računanjem mjere za frazu umjesto za riječi, morfološkom analizom i slično. Vektor se može normalizirati tako da je njegova Euklidska norma jednaka 1. Nakon prikaza dokumenata u ovom obliku najčešće se koristi kosinusova sličnost.

Mjere sličnosti za binarne vektore

Sadržaj objekata je opisan ključnim riječima. Ako objekti sadrže što više ključnih riječi, to su oni sličniji. Napisane su definicije sličnosti kod sustava temeljenih na sličnosti sadržaja objekata.

Definicija 3.0.1. Neka je A skup riječi koji opisuju sadržaj objekta i, a B skup riječi koji opisuju sadržaj objekta j.
Tanimoto / Jaccard indeks sličnosti:

$$J = \frac{|A \cap B|}{|A \cup B|}$$

Dice indeks sličnosti:

$$D = \frac{2|A \cap B|}{|A| + |B|}$$

Definirane sličnosti poprimaju vrijednosti između 0 i 1. Moguće je napraviti izračun i pomoću binarnih vektora i logičkih operatora ”AND” i ”OR”. Može se izraditi profil korisnika koje ključne riječi najviše preferira ili sam pretražuje prema njima, pa tada odrediti sličnost između profila i objekta. Također, može se koristiti i metoda kao kod sustava temeljenih na sličnosti objekta.

Poglavlje 4.

Implementacija u Apache Mahout-u

Postoje razni programski jezici i paketi koji imaju implementirane efikasne algoritme za strojno učenje. Među poznatijima su programski jezik R, paket scikit-learn za Python, Apache Mahout koji koristi Javu, Matlab, Weka, RapidMinder, GraphLabs. Programski paket (eng. framework) otovrenog koda Apache Hadoop koristi distribuirano spremanje i obradu velikih podatka (eng. big data) temeljenim na programskom modelu "MapReduce". Taj model je na počeku razvijao i koristio Google, a zatim je dao kod na slobodno korištenje. Danas je zbog velike količine podataka (petabytes, 10^{15}), napustio ovaj model i koristi "Cloud Dataflow".

Mahout nudi izvršavanje na jednom računalu kao i distribuirano računanje pomoću Apache Hadoop. Sadrži efikasne i skalabilne algoritme za davanje preporuka, grupiranje i klasifikaciju. Sastoji se od Javinih klasa te je jednostavan za uporabu i lako je napraviti funkcionalni početni primjer. Također, može se razviti u funkcionalan sustav za primjenu te ga se može integrirati što ga čini dobrim izborom za implementaciju sustava za davanje preporuka. U ovom radu koristi se verzija 0.13 iz 17. travnja 2017.

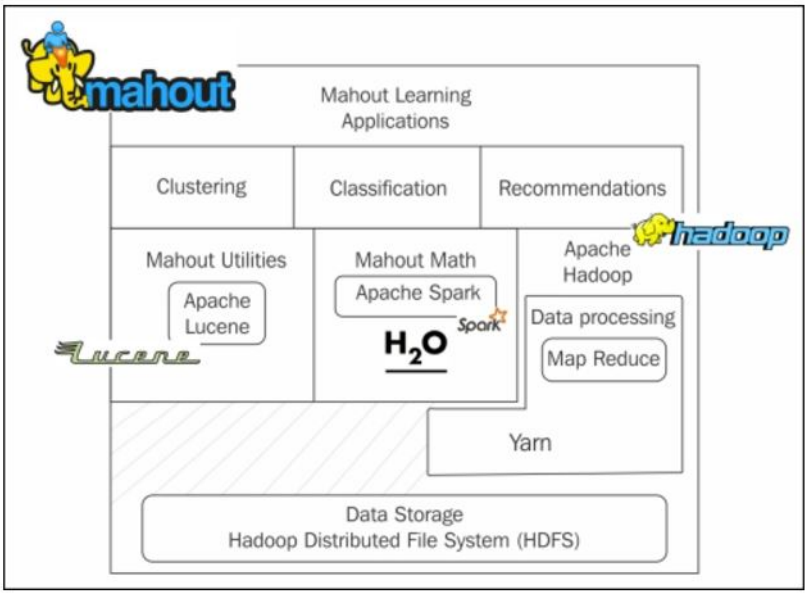
Paket Hadoop sadrži sljedeće module:

- Hadoop Common - sadrži biblioteke koje koriste drugi moduli
- Hadoop Distributed File System (HDFS) - distribuirano spremanje podataka
- Hadoop YARN - platforma odgovorna za podjelu resursa u klasteru i podjela zadataka klijentima

Slika 4.1. Hadoop i Mahout logo (preuzeto sa: `http://mahout.apache.org/` [pristupljeno 01. rujna 2017.])

- Hadoop MapReduce - implementacija MapReduce programskog modela za procesiranje velikih podataka

Uz Apache Hadoop za distribuirano procesiranje Mahout koristi i Apache Spark, za pretraživanje i obradu informacija Apache Lucene te za analizu i rudarenje velikih podataka H_20.



Slika 4.2. Prikaz modula programskog okvira Apache Mahout (preuzeto sa: `https://www.packtpub.com` [pristupljeno 01. rujna 2017.])

MovieLens baze podataka

Istraživački odjel "GroupLens" sveučilišta u Minnesoti specijaliziran je za proučavanje sustava za davanje preporuka. Počeli su s radom 1992. godine. Razvili su mrežni sustav za davanje preporuka "MovieLens" i objavljuju njegove baze za daljnja istraživanja. Na njihovom web sjedištu (`https://grouplens.org/datasets/`) nalaze se 4 baze koje sadrže ocjene i oznake korisnika. Ocjene su od 1.0 do 5.0 s razmakom od 0.5. Podijeljeni su prema broju ocjena koje sadrže. Slučajno su izabrani korisnici koji su ocijenili više od 20 filmova te je njihov pravi identitet skriven. Predstavljeni su brojevima od 1 do, onoliko koliko ih je odabrano za pojedinu bazu. Filmovi su predstavljeni brojevima koji odgovaraju identifikatoru na "MovieLens" sustavu. Podaci iz baze podataka transformirani su tako da ne zadrže retke zaglavlja, da su svi prikazani u "csv" (eng. comma separated value) formatu, te ne sadržavaju oznaku vremena (eng. timestamp) kao originalna baza. Kako je broj identifikatora filmova mnogo veći nego broj filmova koji su korisnici ocijenili, promijenjeni su tako da počinju od 1 do broja filmova koji su ocijenjeni, odnosno koji imaju oznake prema redoslijedu koji se pojavljuje u datoteci s ocjenama. U datoteci "movie_transformed spremaju se originalni identifikatori filmova da bi se kasnije moglo doći do njihovog naziva i originalnog identifikatora. Ako je u i-tom retku zapisan broj j, tada se film koji se u starim podacima pojavljuje kao i sad u novim pojavljuje kao i.

MovieLens baze podataka	ml-100k	ml-1M	ml-10M	ml-20M	ml-26M
Broj ocjena	100.004	1.000.209	10.000.054	20.000.263	26.024.289
Broj korisnika	671	6.040	71.567	138.493	270.896
Broj filmova s ocjenama	9.066	3.706	10.677	26.744	45.115
Broj oznaka	1.296	-	95.580	465.457	753.170
Broj filmova s oznakama	689	-	7.601	19.545	18.393
Broj filmova s ocjenama ili oznakama	9.125	-	10.677	26.744	45.115
Broj filmova s ocjenama i oznakama	630	-	7.601	19.545	18.393
Podaci od	09.01.1995.	09.01.1995.	09.01.1995.	09.01.1995.	09.01.1995.
Podaci do	16.10.2016.	01.02.2003.	01.01.2009.	31.03.2015.	04.08.2017.
Minimalno ocjena	20	20	20	20	20
Raspon ocjena	1.0 - 5.0	1.0 - 5.0	1.0 - 5.0	1.0 - 5.0	1.0 - 5.0
Naziv datoteke s ocjenama	ratings_transformed	ratings_transformed	ratings_transformed	ratings_transformed	ratings_transformed
Ekstenzija datoteke	.csv				
Broj redaka zaglavlja	0				
Podaci	userID, movieID, rate				
Naziv datoteke s oznakama	tags_transformed	tags_transformed	tags_transformed	tags_transformed	tags_transformed
Ekstenzija datoteke	.csv				
Broj redaka zaglavlja	0				
Podaci	userID, movieID, tag				

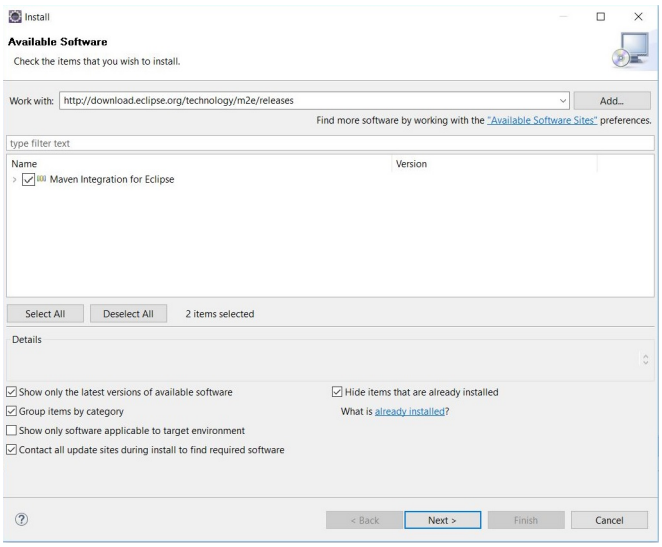
Slika 4.3. Karakteristike uređenih MovieLens baza

Razvojno okruženje Eclipse i Apache Maven

Jedno od najpoznatijih integriranih razvojnih okruženja za pisanje Jave je besplatan Eclipse. Sadrži podršku i za ostale programske jezike. Zadnja stabilna verzija je 4.7. nazvana "Oxygen" od 28. lipnja 2017. godine koja se može preuzeti na stranici `https://www.eclipse.org/downloads/`.

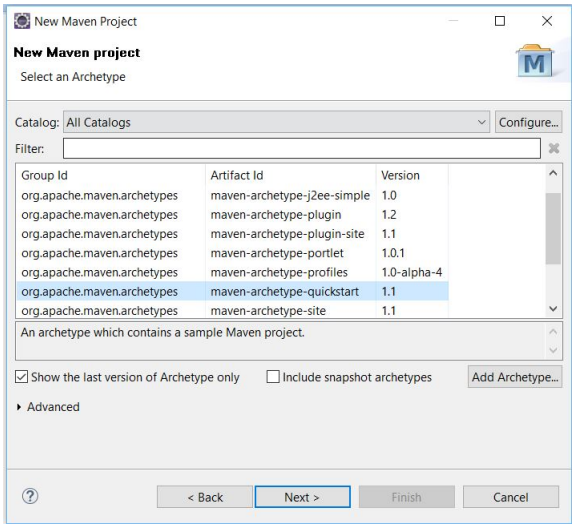
Slika 4.4. Logo besplatnog razvojnog okruženja Eclipse

Za automatiziranu izradu i provjeru binarnog koda Apache Mahout koristi Maven. Koristi se i za upravljanje ovisnostima među modulima i ostalih programa te njihovih verzija. Ostali poznatiji takvi alati su Apache Ant i Gradle. Integracija Mavena u Eclipsu vrši se preko opcije "install New Software..." koja se nalazi u izborniku "Help". Potrebno je zatim upisati adresu "`http://download.eclipse.org/technology/m2e/releases`".

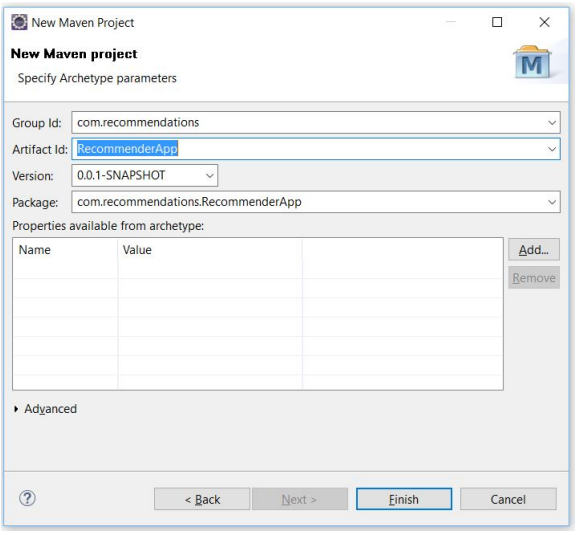


Slika 4.5. Instalacija Mavena unutar Eclipsa

Kreiranje novog Maven projekta vrši se opcijom ”New” u izboriku ”File” i zatim se odabire ”Maven Project”.

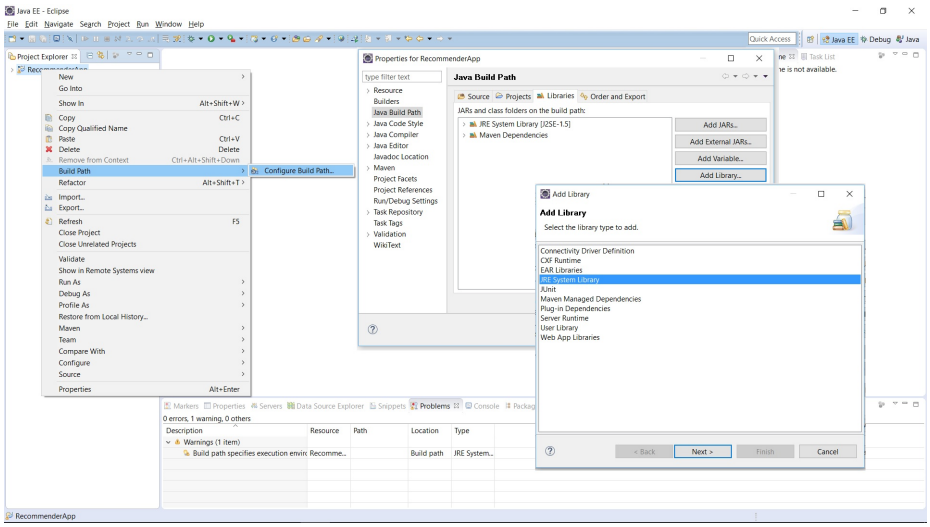


Slika 4.6. Izrada novog Maveb projekta.



Slika 4.7. Postavke novog projekta

Potrebno je ispravno postaviti biblioteku ”jre” (eng. java routine environment).



Slika 4.8. Postavljanje ”jre” biblioteke

Zatim se u datoteku ”pom.xml” između oznaka ”<dependencies>” i ”</dependencies>” zapisuje sljedeći tekst:

<dependency>
<groupId>org.apache.mahout</groupId>
<artifactId>mahout-mr</artifactId>
<version>0.13.0</version>
</dependency>

Naposljetku treba napraviti novu datoteku u ”log4j.properties”. Datoteka se treba nalaziti u ”src/main/java”. U njoj treba zapisati postavke kako bi se ispisale poruke na konzoli za lakše otkrivanje grešaka u programu. Sadržaj datoteke je sljedeći:

# Root logger option
log4j.rootLogger=INFO, stdout
# Direct log messages to stdout
log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.Target=System.out
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p {1}:%L - %m%n

Mapa ”data” sadrži MovieLens baze podataka. Kod se piše u datoteci ”App.java”. Kada se ta datoteka otvori, u njoj je zapisana osnovna klasa s glavnom funkcijom ”main”.

Za implementaciju jednostavnog sustava za preporuke u pomoću Apache Mahoutu potrebno je nekoliko linija koda. Prvo se učitavaju podaci iz MovieLens baze podataka. Zatim se određuje koja se mjera koristi za određivanje sličnosti između korisnika, pa način na koji se računaju najbliži susjedi, te se na temelju toga izrađuje preporučitelj.

Programski kod 4.1. Korisniku ”1” se na temelju sličnosti između 20 najsličnijih korisnika koristeći Pearsonov koeficijent korelacije daje 10 objekata s najboljom predikcijom ocjena

```
1   public static void main( String[] args ) throws IOException, TasteException
2   {
3       DataModel model = new FileDataModel(new File("data/ml-100K/ratings_transformed
    .csv"));
4       UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
5       UserNeighborhood neighborhood = new NearestNUserNeighborhood(20, similarity,
    model);
6       UserBasedRecommender recommender = new GenericUserBasedRecommender(model,
    neighborhood, similarity);
7       List<RecommendedItem> recommendations = recommender.recommend(1, 10);
8       for (RecommendedItem recommendation : recommendations)
9       {
10          System.out.println(recommendation);
11      }
12  }
```

Pomoću tipkovničkog prečaca ”SHIFT + CTRL + O” učitavaju se sve potrebne klase za ispravan rad programa. Prikazan je kod unutar ”main” funkcije. U prikazanom primjeru model čine svi podaci iz datoteke ”rating_transformed.csv” koja sadrži ocjene korisnika. Osim modela mogu se mijenjati i drugi parametri. Umjesto prilagođene kosinusove sličnosti koristi se Pearsonov koeficijent korelacije.

Parametri sustava za davanje preporuka temeljenih na sličnosti korisnika:

- mjera sličnosti
  - CityBlockSimilarity → za binarne vektore
  - LogLikelihoodSimilarity → za binarne vektore
  - TanimotoCoefficientSimilarity → za binarne vektore
  - EuclideanDistanceSimilarity

  - UncenteredCosineSimilarity
  - PearsonCorrelationSimilarity → centrira podatke
  - SpearmanCorrelationSimilarity → centrira podatke
- određivanje susjeda
  - NearestNUserNeighborhood → kao parametre prima broj najsličnijih susjeda i mjeru sličnosti
  - ThresholdUserNeighborhood → kao parametre prima prag na temelju kojeg određuje najbliže susjede i mjeru sličnosti
- broj preporuka → prirodan broj

Za izradu sustava temeljenih na sličnosti objekata mogu se koristiti iste klase za računanje sličnosti između objekata osim "SpearmanCorrelationSimilarity". Klase su implementirane tako da same prepoznaju kada se računa sličnost za korisnike, a kada za objekte.

Programski kod 4.2. Korisniku "1" se na temelju sličnosti između objekata koristeći Pearsonov koeficijent korelacije ispisuje 10 objekata s najboljom predikcijom ocjena

```
1     public static void main( String[] args ) throws IOException, TasteException
2     {
3         DataModel model = new FileDataModel(new File("data/ml-100K/ratings_transformed
      .csv"));
4         ItemSimilarity similarity = new PearsonCorrelationSimilarity(model);
5         ItemBasedRecommender recommender = new GenericItemBasedRecommender(model,
      similarity);
6         List<RecommendedItem> recommendations = recommender.recommend(1, 10);
7         for (RecommendedItem recommendation : recommendations)
8         {
9             System.out.println(recommendation);
10        }
11    }
```

Implementirani su i sljedeći algoritmi za matričnu faktorizaciju:

- SVDPlusPlusFactorizer → koristi unaprijeđeni algoritam SVD++
- RatingSGDFactorizer → osnovni stohastički gradijentni spust koji uzima u obzir pristranost ocjena korisnika i objekata
- ParallelSGDFactorizer → stohastički gradijentni spust za paralelno izvršavanje
- ALSWRFactorizer → regularizirana alternirajuća metoda najmanjih kvadrata s težinama

Programski kod 4.3. Popis konstruktora matričnih faktorizacija

```
1  public SVDPlusPlusFactorizer(DataModel dataModel, int numFeatures, int numIterations)
2  public SVDPlusPlusFactorizer(DataModel dataModel, int numFeatures, double learningRate
       , double preventOverfitting, double randomNoise, int numIterations, double
       learningRateDecay)
3
4  public RatingSGDFactorizer(DataModel dataModel, int numFeatures, int numIterations)
5  public RatingSGDFactorizer(DataModel dataModel, int numFeatures, double learningRate,
       double preventOverfitting, double randomNoise, int numIterations, double
       learningRateDecay)
6
7  public ParallelSGDFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numEpochs)
8  public ParallelSGDFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numIterations, double mu0, double decayFactor, int stepOffset, double
       forgettingExponent)
9  public ParallelSGDFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numIterations, double mu0, double decayFactor, int stepOffset, double
       forgettingExponent, int numThreads)
10 public ParallelSGDFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numIterations, double mu0, double decayFactor, int stepOffset, double
       forgettingExponent, double biasMuRatio, double biasLambdaRatio)
11 public ParallelSGDFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numIterations, double mu0, double decayFactor, int stepOffset, double
       forgettingExponent, double biasMuRatio, double biasLambdaRatio, int numThreads)
12
13 public ALSWRFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numIterations, boolean usesImplicitFeedback, double alpha, int numTrainingThreads)
14 public ALSWRFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numIterations, boolean usesImplicitFeedback, double alpha)
15 public ALSWRFactorizer(DataModel dataModel, int numFeatures, double lambda, int
       numIterations)
```

Konstruktorima je zajednički model podataka (DataModel) na kojem će napraviti faktorizaciju, broj značajki (numFeatures), broj iteracija (numIterations) te kod većine regularizacijski parametar (lambda).

Poglavlje 5.

Evaluacija sustava

U radu su opisani i implementirani razni sustavi za davanje preporuka. Svaki od njih može se izraditi s različitim parametrima. Kako bi sustav bio što točniji i brži, potrebno je napraviti analizu i evaluaciju sustava za specifičnu domenu primjene. Navedene su najkorištenije mjere za vrednovanje sustava te kako se evaluacije implementiraju u Apache Mahoutu te rezultati tih analiza na MovieLens bazama podataka.

Dvije mjere koje se najčešće koriste u literaturi za točnost sustava su sredina apsolutnih grešaka - MAE (eng. mean absolute error) i korijen sredine kvadrata grešaka - RMSE (eng. root-mean-square error). Oznaka MSE (eng. mean-square error) predstavlja sredinu kvadrata grašaka. Što je vrijednost bliža nuli, to je točnost predikcija veća. Iz zadane baze odabire su skup ocjena na kojem izrađujemo sustav, a ono što preostaje je skup $T \subset U \times I$ na kojem testiramo sustav. Na tom skupu vrši se predikcija ocjena. Neka je $r_{u,i}$ stvarna ocjena korisnika u objektu i, a $\hat{r}_{u,i}$ predikcija te ocjene. Mjere su zadane formulama:

$$\text{MAE} = \frac{\sum_{(u,i)\in T} |R(u,i) - \hat{R}(u,i)|}{card(T)}, \quad \text{RMSE} = \sqrt{MSE} = \sqrt{\frac{\sum_{(u,i)\in T} (R(u,i) - \hat{R}(u,i))^2}{card(T)}}$$

U Apache Mahoutu implementirane su klase "AverageAbsoluteDifferenceRecommenderEvaluator", koja računa MAE i "RMSRecommenderEvaluator", koja računa RMSE. Zadani parametri su postotak (decimalni broj između 0 i 1) ocjena svakog korisnika koji se zadržava za treniranje i postotak korisnika na kojima će se vršiti predikcija ocjena. Skup za treniranje bira se uniformnom distribucijom nad ocjenama korisnika. Kako se radi o slučajnom izboru, poželjno je nekoliko puta ponoviti evaluaciju pa uzeti prosjek mjera.

Osim točnosti važna je i brzina davanja preporuka. ATR (eng. average time per predictions) mjeri prosječno vrijeme potrebno za predikciju ocjene. Izvršavanjem evaluacije u Apache Mahoutu u konzoli ispisat će se i prosječno vrijeme. Prvo prosječno vrijeme je najveće, dok se preostala neznatna jer, Mahout već izračunate sličnosti pamti pa ih nije potrebno ponovno računati.

Implementirana je i klasa "IRStatistic" (eng. IR = information retrieval) koja računa točnost (eng. accuracy - ACC) i osjetljivost (en. sensitivity ili recall ili true positive rate - REC). Ove mjere najlakše je razumjeti pomoću matrice konfuzije koja se sastoji od dva retka i dva stupca. Po recima se nalazi broj objekata koji su predviđeni modelom kao pozitivni (relevantni) i negativni (irelevantni), a po stupcima broj objekata koji su stvarno pozitivni (relevantni) i negativni. S "TP" (eng. true positive) označava se broj objekata za koje se predviđa da su pozitivni, i stvarno jesu, a s "FP" (eng. false positive) oni za koje se predviđa da su pozitivni, a nisu. "FN" (end. false negative) su oni koji su predviđeni pozitivni, a nisu, te "TN" (eng. true negative) označava broj objekata koji su predviđeni i jesu negativni.

Matrica konfuzije (confusion matrix)		**Stvarna klasa**	
		G (+) Pozitivni	**NG (-) Negativni**
Predviđeno modelom *h*	**G (+) Pozitivni**	TP	FP
	NG (-) Negativni	FN	TN

Slika 5.1. Matrica konfuzije (Tomislav Šmuc, 2011.)

Točnost predstavlja omjer točno klasificiranih primjera u odnosu na ukupan broj primjera, a preciznost udio stvarno pozitivnih primjera u svima koji su modelom predviđeni kao pozitivni:

$$ACC = \frac{TP + TN}{TP + FP + TN + FN}, \quad REC = \frac{TP}{TP + FN}$$

Sustav daje listu preporuka korisniku. Za te objekte na listi predvidio je da su relevantni za korisnika. Računanje točnosti i osjetljivosti vrši se tako da se uklone ocjene relevantnih objekata (većom ocjenom od nekog praga, obično aritmetičke sredine ocjena korisnika) i

sustav napravi listu te se promatra nalaze li se unutar nje ti uklonjeni objekti. Kod sustava gdje korisnici imaju mali broj ocijenjenih objekata ove mjere nisu primjenjive, jer za njih postoji mnogo relevantnih objekata koji nisu ocijenjeni.

Evaluacija sustava temeljnog na memoriji

Primjer evaluacije sustava dan je bazama MovieLens. Prvo se promatra sustav temeljen na sličnosti korisnika koji određuje k-najsličnijih susjeda. Zabilježene se relevantne mjere se ovakav sustav MAE i RMSE.

Programski kod 5.1. Evaluacija sustava temeljenog na sličnosti korisnika računajući 50 najsličnijih susjeda - 95% ocjena se zadržava i na 5% korisnika se vrši testiranje

```
1     public static void main(String[] args) throws Exception {
2         int numEvaluations = 1;
3       double percentageOfTrainingData = 0.95;
4       double percentageOfTestData = 0.5;
5       final int k = 50;
6
7       RandomUtils.useTestSeed();
8       DataModel model = new FileDataModel(new File("data/ml-100K/ratings_transformed.
          csv"));
9       RecommenderEvaluator MAE_evaluator = new
          AverageAbsoluteDifferenceRecommenderEvaluator();
10      RecommenderEvaluator RMSE_evaluator = new RMSRecommenderEvaluator();
11
12      RecommenderBuilder EuclideanDistanceRecBuilder = new RecommenderBuilder() {
13        public Recommender buildRecommender(DataModel model) throws TasteException {
14            UserSimilarity similarity = new EuclideanDistanceSimilarity(model);
15            UserNeighborhood neighborhood = new NearestNUserNeighborhood(k, similarity
          , model);
16            return new GenericUserBasedRecommender(model, neighborhood, similarity);
17        }
18      };
19
20      RecommenderBuilder UncenteredCosineRecBuilder = new RecommenderBuilder() {
21        public Recommender buildRecommender(DataModel model) throws TasteException {
22            UserSimilarity similarity = new UncenteredCosineSimilarity(model);
23            UserNeighborhood neighborhood = new NearestNUserNeighborhood(k, similarity
          , model);
24            return new GenericUserBasedRecommender(model, neighborhood, similarity);
25        }
26      };
27
28      RecommenderBuilder PearsonCorrelationRecBuilder = new RecommenderBuilder() {
29        public Recommender buildRecommender(DataModel model) throws TasteException {
30            UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
31            UserNeighborhood neighborhood = new NearestNUserNeighborhood(k, similarity
          , model);
32            return new GenericUserBasedRecommender(model, neighborhood, similarity);
33        }
34      };
```

```
35
36          RecommenderBuilder SpearmanCorrelationRecBuilder = new RecommenderBuilder() {
37            public Recommender buildRecommender(DataModel model) throws TasteException {
38                UserSimilarity similarity = new SpearmanCorrelationSimilarity(model);
39                UserNeighborhood neighborhood = new NearestNUserNeighborhood(k, similarity
                , model);
40                return new GenericUserBasedRecommender(model, neighborhood, similarity);
41            }
42          };
43
44          double totalEuclideanDistanceRecMAE = 0, totalEuclideanDistanceRecRMSE = 0,
              avgEuclideanDistanceRecMAE, avgEuclideanDistanceRecRMSE;
45          double totalUncenteredCosineRecMAE = 0, totalUncenteredCosineRecRMSE = 0,
              avgUncenteredCosineRecMAE, avgUncenteredCosineRecRMSE;
46          double totalPearsonCorrelationRecMAE = 0, totalPearsonCorrelationRecRMSE = 0,
              avgPearsonCorrelationRecMAE, avgPearsonCorrelationRecRMSE;
47          double totalSpearmanCorrelationRecMAE = 0, totalSpearmanCorrelationRecRMSE = 0,
              avgSpearmanCorrelationRecMAE, avgSpearmanCorrelationRecRMSE;
48          for( int i = 0; i < numEvaluations; i++)
49          {
50            totalEuclideanDistanceRecMAE = totalEuclideanDistanceRecMAE + MAE_evaluator.
              evaluate(EuclideanDistanceRecBuilder, null, model, percentageOfTrainingData,
              percentageOfTestData);
51            totalEuclideanDistanceRecRMSE = totalEuclideanDistanceRecRMSE + RMSE_evaluator
              .evaluate(EuclideanDistanceRecBuilder, null, model, percentageOfTrainingData,
              percentageOfTestData);
52
53            totalUncenteredCosineRecMAE = totalUncenteredCosineRecMAE + MAE_evaluator.
              evaluate(UncenteredCosineRecBuilder, null, model, percentageOfTrainingData,
              percentageOfTestData);
54            totalUncenteredCosineRecRMSE = totalUncenteredCosineRecRMSE + RMSE_evaluator.
              evaluate(UncenteredCosineRecBuilder, null, model, percentageOfTrainingData,
              percentageOfTestData);
55
56            totalPearsonCorrelationRecMAE = totalPearsonCorrelationRecMAE + MAE_evaluator.
              evaluate(PearsonCorrelationRecBuilder, null, model, percentageOfTrainingData,
              percentageOfTestData);
57            totalPearsonCorrelationRecRMSE = totalPearsonCorrelationRecRMSE +
              RMSE_evaluator.evaluate(PearsonCorrelationRecBuilder, null, model,
              percentageOfTrainingData, percentageOfTestData);
58
59            totalSpearmanCorrelationRecMAE = totalSpearmanCorrelationRecMAE +
              MAE_evaluator.evaluate(SpearmanCorrelationRecBuilder, null, model,
              percentageOfTrainingData, percentageOfTestData);
60            totalSpearmanCorrelationRecRMSE = totalSpearmanCorrelationRecRMSE +
              RMSE_evaluator.evaluate(SpearmanCorrelationRecBuilder, null, model,
              percentageOfTrainingData, percentageOfTestData);
61
62          }
63          avgEuclideanDistanceRecMAE = totalEuclideanDistanceRecMAE / numEvaluations;
64          avgEuclideanDistanceRecRMSE = totalEuclideanDistanceRecRMSE / numEvaluations;
65          System.out.println("avgEuclideanDistanceRecMAE = " + avgEuclideanDistanceRecMAE)
              ;
66          System.out.println("avgEuclideanDistanceRecRMSE = " +
              avgEuclideanDistanceRecRMSE);
67
68          avgUncenteredCosineRecMAE = totalUncenteredCosineRecMAE / numEvaluations;
69          avgUncenteredCosineRecRMSE = totalUncenteredCosineRecRMSE / numEvaluations;
70          System.out.println("avgUncenteredCosineRecMAE = " + avgUncenteredCosineRecMAE);
71          System.out.println("avgUncenteredCosineRecRMSE = " + avgUncenteredCosineRecRMSE)
```

```
            ;
72
73          avgPearsonCorrelationRecMAE = totalPearsonCorrelationRecMAE / numEvaluations;
74          avgPearsonCorrelationRecRMSE = totalPearsonCorrelationRecRMSE / numEvaluations;
75          System.out.println("avgPearsonCorrelationRecMAE = " +
              avgPearsonCorrelationRecMAE);
76          System.out.println("avgPearsonCorrelationRecRMSE = " +
              avgPearsonCorrelationRecRMSE);
77
78          avgSpearmanCorrelationRecMAE = totalSpearmanCorrelationRecMAE / numEvaluations;
79          avgSpearmanCorrelationRecRMSE = totalSpearmanCorrelationRecRMSE / numEvaluations
              ;
80          System.out.println("avgSpearmanCorrelationRecMAE = " +
              avgSpearmanCorrelationRecMAE);
81          System.out.println("avgSpearmanCorrelationRecRMSE = " +
              avgSpearmanCorrelationRecRMSE);
82        }
```

Database		ml-100k					
Similarity	k	50		100		200	
	Error	MAE	RMSE	MAE	RMSE	MAE	RMSE
Euclidean Distance Similarity		0.686	0.913	0.697	0.915	0.712	0.931
Uncentered Cosine Similarity		0.761	0.994	0.768	1.008	0.743	0.968
Pearson Correlation Similarity		0.826	1.075	0.804	1.043	0.813	1.085
Spearman Correlation Similarity		0.775	1.010	0.749	0.974	0.770	1.017

Slika 5.2. Vrijednosti MAE i RMSE prema različitim parametrima sustava temeljenog na sličnosti korisnika pomoću najsličnijih susjeda

Programski kod 5.2. Evaluacija sustava temeljenog na sličnosti korisnika uzimajući u obzir najsličnije korisnike koji prelaze prag *t*

```
1           UserNeighborhood neighborhood = new ThresholdUserNeighborhood(t,
          similarity, model);
```

Database		ml-100k					
Similarity	t	0.95		0.9		0.85	
	Error	MAE	RMSE	MAE	RMSE	MAE	RMSE
Euclidean Distance Similarity		0.682	0.960	0.763	0.986	0.775	1.011
Uncentered Cosine Similarity		0.759	0.974	0.761	0.978	0.769	0.990
Pearson Correlation Similarity		0.872	1.145	0.840	1.074	0.871	1.120
Spearman Correlation Similarity		0.862	1.113	0.895	1.139	0.873	1.118

Slika 5.3. Vrijednosti MAE i RMSE prema različitim parametrima sustava temeljenog na sličnosti korisnika pomoću određivanja susjeda preko praga

Kada se sličnost računa pomoću Euklidove udaljenosti, tada se smanjenjem broja najbližih susjeda i povećanjem praga smanjuje ukupna greška sustava. Kod preostalih sličnosti razlike ukupnih grešaka su male pa je za brži radi bolje uzeti manji broj sličnih korisnika.

Programski kod 5.3. Evaluacija sustava temeljnog na matričnoj faktorizaciji

```
1       int numEvaluations = 10;
2       double percentageOfTrainingData = 0.95;
3       double percentageOfTestData = 0.05;
4       final int numFeatures = 20;
5       final int numIterations = 1;
6       final double lambda = 0.2;
7
8     RandomUtils.useTestSeed();
9       DataModel model = new FileDataModel(new File("data/ml-100K/ratings_transformed.
          csv"));
10      RecommenderEvaluator MAE_evaluator = new
          AverageAbsoluteDifferenceRecommenderEvaluator();
11      RecommenderEvaluator RMSE_evaluator = new RMSRecommenderEvaluator();
12
13      RecommenderBuilder SVDPlusPlusRecommenderBuilder = new RecommenderBuilder() {
14        public Recommender buildRecommender(DataModel model) throws TasteException {
15            return new SVDRecommender(model, new SVDPlusPlusFactorizer(model,
          numFeatures, numIterations));
16        }
17      };
18      RecommenderBuilder RatingSGDRecommenderBuilder = new RecommenderBuilder() {
19        public Recommender buildRecommender(DataModel model) throws TasteException {
20            return new SVDRecommender(model, new RatingSGDFactorizer(model,
          numFeatures, numIterations));
21        }
22      };
23      RecommenderBuilder ParallelSGDRecommenderBuilder = new RecommenderBuilder() {
24        public Recommender buildRecommender(DataModel model) throws TasteException {
25            return new SVDRecommender(model, new ParallelSGDFactorizer(model,
          numFeatures, lambda, numIterations));
26        }
27      };
28      RecommenderBuilder ALSWRRecommenderBuilder = new RecommenderBuilder() {
29        public Recommender buildRecommender(DataModel model) throws TasteException {
30            return new SVDRecommender(model, new ALSWRFactorizer(model, numFeatures,
          lambda, numIterations));
31        }
32      };
33
34      double totalSVDPlusPlusMAE = 0, totalSVDPlusPlusRMSE = 0, avgSVDPlusPlusMAE,
          avgSVDPlusPlusRMSE;
35      double totalRatingSGDMAE = 0, totalRatingSGDRMSE = 0, avgRatingSGDMAE,
          avgRatingSGDRMSE;
36      double totalParallelSGDMAE = 0, totalParallelSGDRMSE = 0, avgParallelSGDMAE,
          avgParallelSGDRMSE;
37      double totalALSWR_MAE = 0, totalALSWR_RMSE = 0, avgALSWR_MAE, avgALSWR_RMSE;
38      for( int i = 0; i < numEvaluations; i++)
39      {
40        totalSVDPlusPlusMAE = totalSVDPlusPlusMAE + MAE_evaluator.evaluate(
          SVDPlusPlusRecommenderBuilder, null, model, percentageOfTrainingData,
```

```
            percentageOfTestData);
41          totalSVDPlusPlusRMSE = totalSVDPlusPlusRMSE + RMSE_evaluator.evaluate(
            SVDPlusPlusRecommenderBuilder, null, model, percentageOfTrainingData,
            percentageOfTestData);
42
43          totalRatingSGDMAE = totalRatingSGDMAE + MAE_evaluator.evaluate(
            RatingSGDRecommenderBuilder, null, model, percentageOfTrainingData,
            percentageOfTestData);
44          totalRatingSGDRMSE = totalRatingSGDRMSE + RMSE_evaluator.evaluate(
            RatingSGDRecommenderBuilder, null, model, percentageOfTrainingData,
            percentageOfTestData);
45
46          totalParallelSGDMAE = totalParallelSGDMAE + MAE_evaluator.evaluate(
            ParallelSGDRecommenderBuilder, null, model, percentageOfTrainingData,
            percentageOfTestData);
47          totalParallelSGDRMSE = totalParallelSGDRMSE + RMSE_evaluator.evaluate(
            ParallelSGDRecommenderBuilder, null, model, percentageOfTrainingData,
            percentageOfTestData);
48
49          totalALSWR_MAE = totalALSWR_MAE + MAE_evaluator.evaluate(
            ALSWRRecommenderBuilder, null, model, percentageOfTrainingData,
            percentageOfTestData);
50          totalALSWR_RMSE = totalALSWR_RMSE + RMSE_evaluator.evaluate(
            ALSWRRecommenderBuilder, null, model, percentageOfTrainingData,
            percentageOfTestData);
51        }
52        avgSVDPlusPlusMAE = totalSVDPlusPlusMAE / numEvaluations;
53        avgSVDPlusPlusRMSE = totalSVDPlusPlusRMSE / numEvaluations;
54        System.out.println("avgSVDPlusPlusMAE = " + avgSVDPlusPlusMAE);
55        System.out.println("avgSVDPlusPlusRMSE = " + avgSVDPlusPlusRMSE);
56
57        avgRatingSGDMAE = totalRatingSGDMAE / numEvaluations;
58        avgRatingSGDRMSE = totalRatingSGDRMSE / numEvaluations;
59        System.out.println("avgRatingSGDMAE = " + avgRatingSGDMAE);
60        System.out.println("avgSRatingSGDRMSE = " + avgRatingSGDRMSE);
61
62        avgParallelSGDMAE = totalParallelSGDMAE / numEvaluations;
63        avgParallelSGDRMSE = totalParallelSGDRMSE / numEvaluations;
64        System.out.println("avgParallelSGDMAE = " + avgParallelSGDMAE);
65        System.out.println("avgParallelSGDRMSE = " + avgParallelSGDRMSE);
66
67        avgALSWR_MAE = totalALSWR_MAE / numEvaluations;
68        avgALSWR_RMSE = totalALSWR_RMSE / numEvaluations;
69        System.out.println("avgALSWR_MAE = " + avgALSWR_MAE);
70        System.out.println("avgALSWR_RMSE = " + avgALSWR_RMSE);
```

Iz tablice vrijednosti ukupnih grešaka kod sustava temeljenih na matričnoj faktorizaciji, može se zaključiti da se povećavanjem broja iteracije i povećanjem broja značajki smanjuje ukupna greška. Što ima više značajki, to je gubitak informacija manji pa je time i točnost predikcije veća, ali se produžuje vrijeme za davanje preporuka. Također i kod povećanja broja iteracija. Kada je parametar *lambda* oko 0.2, tada se postižu najbolji rezultati.

Database			ml-100k					
Factorizer	numF.		10		20		50	
	numIt.	Lambda	MAE	RMSE	MAE	RMSE	MAE	RMSE
SVD Plus Plus	5	-	0.747	0.951	0.723	0.968	0.721	0.955
	10	-	0.718	0.949	0.717	0.948	0.716	0.947
	50	-	0.703	0.936	0.701	0.934	0.699	0.929
Rating SGD	5	-	0.744	0.91	0.717	0.919	0.692	0.895
	10	-	0.681	0.876	0.681	0.883	0.681	0.883
	50	-	0.667	0.866	0.663	0.861	0.661	0.858
Parallel SGD	5	0.5	0.707	0.911	0.706	0.910	0.706	0.910
	5	0.2	0.709	0.906	0.716	0.898	0.704	0.913
	5	0.02	0.705	0.910	0.705	0.910	0.705	0.910
	10	0.5	0.693	0.899	0.693	0.898	0.693	0.899
	10	0.2	0.694	0.899	0.694	0.898	0.694	0.898
	10	0.02	0.693	0.898	0.692	0.897	0.690	0.894
	50	0.5	0.683	0.888	0.683	0.683	0.683	0.889
	50	0.2	0.683	0.889	0.6829	0.889	0.683	0.889
	50	0.02	0.687	0.904	0.687	0.903	0.679	0.885
ALSWR	5	0.5	0.880	1.060	0.880	1.060	0.880	1.060
	5	0.2	0.874	1.106	0.851	1.112	0.764	0.949
	5	0.02	0.745	0.974	0.785	1.036	0.843	1.119
	10	0.5	0.856	1.037	0.855	1.037	0.856	1.037
	10	0.2	0.743	0.935	0.743	0.935	0.743	0.935
	10	0.02	0.758	0.999	0.791	1.058	0.819	1.084
	50	0.5	0.855	1.036	0.855	1.037	0.855	1.036
	50	0.2	0.735	0.928	0.735	0.928	0.735	0.928
	50	0.02	0.752	0.995	0.773	1.018	0.809	1.057

Slika 5.4. Vrijednosti MAE i RMSE prema različitim parametrima sustava temeljenog na matričnoj faktorizaciji

Bibliografija

[1] Aberger, Christopher R: *Recommender: An Analysis of Collaborative Filtering Techniques*. 2014.

[2] Asanov, Daniar: *Algorithms and Methods in Recommender Systems*. 2011.

[3] Barbieri, Nicola, Giuseppe Manco i Ettore Ritacco: *Probabilistic Approaches to Recommendations*. 2014.

[4] Bobadilla, J, F Ortega, A Hernando i A Gutiérrez: *Knowledge-Based Systems Recommender systems survey*. Knowledge-Based Systems, 2013.

[5] Bogers, Toine i Antal Van Den Bosch: *Recommender Systems Handbook*. 2009.

[6] Burnik, Konrad: *Implementacija efikasnog algoritma za izvlačenje rezultata iz sustava za davanje automatskih preporuka*, 2016.

[7] Cacheda, Fidel, Víctor Carneiro, Diego Fernández i Vreixo Formoso: *Comparison of collaborative filtering algorithms*. ACM Transactions on the Web, 2011.

[8] Ĉavka, Marko: *Sustavi za preporuĉivanje*.

[9] Durao, Frederico i Peter Dolog: *A Personalized Tag-Based Recommendation in Social Web Systems*. 2012.

[10] Ekstrand, Michael D., John T. Riedl i Joseph A. Konstan: *Collaborative Filtering Recommender Systems*. International Journal of Electronic Business, 2007.

[11] Ilin, Alexander i Tapani Raiko: *Practical approaches to principal component analysis in the presence of missing values*. Journal of Machine Learning Research, 2010.

[12] Jannach, Dietmar i Gerhard Friedrich: *Tutorial: Recommender Systems*, 2013.

[13] Jannach, Dietmar, Markus Zanker, Alexander Felfering i Gerhard Friedrich: *Recommender systems - An introduction*. 2011.

[14] Leskovec, Jure, Rajaraman Anand i Jeff Ullman: *Recommendation Systems*. Mining of Massive Datasets, 2011.

[15] Portugal, Ivens, Paulo Alencar i Donald Cowan: *The Use Of Machine Learning Algorithms In Recommender Systems: A Systematic Review*. arXiv, 2015.

[16] Segaran, Toby: *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. 2007.

[17] Setnam, Alag: *Collective Intelligence in Action*. 2013.

[18] Simović, Aleksandar: *Sistemi preporuke u e-trgovini*. Proceedings of the 1st International Scientific Conference - Sinteza 2014, 2014.

[19] Šitum, Mirjam: *Rangiranje prijatelja u društvenoj mreži facebook zasnovano na korisničkim akcijama*. Disertacija, Sveučilište u Zagrebu, 2012.

[20] Šumečki, Josip: *Sustavi za preporučivanje*, 2010.

[21] Vozalis, Manolis G. i Konstantinos G. Margaritis: *Applying SVD on item-based filtering*. Proceedings - 5th International Conference on Intelligent Systems Design and Applications 2005, ISDA '05, 2005.

[22] Zanardi, Valentina i Licia Capra: *A scalable tag-based recommender system for new users of the social web*. Lecture Notes in Computer Science, 2011.

Sažetak

U ovom diplomskom radu analiziran je sustav za davanje preporuka na webu. Navedeni su primjeri, zadaće, prednosti i nedostaci te njihova podjela. Detaljnije su opisani sustavi temeljeni na suradnji korisnika i onih temeljenih na sadržaju objekata. Dane su upute za instalaciju potrebnih programa i paketa za implementaciju opisanih sustava u Apache Mahout-u. Priloženi su kodovi i opisani su potrebni parametri. Evaluacija sustava izvršena je prema različitim metodama na MovieLens javno dostupnim bazama podataka te je mapravljena analiza dobivenih podataka.

Summary

In this graduate thesis, a system for giving recommendations on the web is described. Examples, goals, advantages, disadvantages and their classification are listed. User-based and content-based systems are described in more details. The instructions for installing the required programs and packages for deploying the described systems in Apache Mahout are given. The implementation codes are enclosed and parameters for possible modification are described. The evaluation was performed according to various methods on MovieLens publicly available databases and analysis of the obtained data was made.

Životopis

Rođen sam 18. kolovoza 1991. godine u Puli. U Rovinju sam završio Osnovnu školu Jurja Dobrile. Već u osnovnoj školi pokazujem interes za matematiku i informatiku, posebice za programiranje te sam sudjelovao na brojnim županijskim i državnim natjecanjima. Bio sam član Udruge tehničke kulture "Galileo Galilei" u Rovinju, a kasnije za Udrugu vodim radionicu "Izrada web stranica" za osnovnoškolce. Uspješnim polaganjem državne mature 2010. godine završio sam Prirodoslovno-matematičku gimnaziju u Srednjoj školi Zvane Črnje u Rovinju. Iste godine upisujem sveučilišni preddiplomski studij matematike na Prirodoslovno-matematičkom fakultetu u Zagrebu, zatim 2015. godine upisujem sveučilišni diplomski studij Računarstva i matematike. Tijekom studija radio sam kao nastavnik matematike i računarstva u Srednjoj školi Zvane Črnje u Rovinju i u Srednjoj strukovnoj školi u Samoboru te sam bio demonstrator iz kolegija Građa računala. Za fakultet sam redovito sudjelovao u sportskim natjecanjima iz dvoranske odbojke i odbojke na pijesku.