Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij matematike i računarstva

Mirna Marković

Kolaborativno filtriranje

Diplomski rad

Osijek, 2016.

Sveučilište J.J. Strossmayera u Osijeku
Odjel za matematiku
Sveučilišni diplomski studij matematike i računarstva

Mirna Marković

Kolaborativno filtriranje

Diplomski rad

Mentor : doc. dr. sc. Zoran Tomljanović

Osijek, 2016.

Sadržaj

1 Uvod

Preporuka je, prema definiciji iz [21], povoljno pismeno ili usmeno mišljenje, odnosno povoljna ocjena o svojstvima koga ili čega. Svakodnevno se, pri donošenju i najjednostavnijih odluka, oslanjamo na preporuke osoba iz svoje okoline. Bilo da se radi o prehrambenoj namirnici, odjevnom predmetu, knjizi ili novom filmu, uvijek nas unaprijed zanima isplati li se na to utrošiti vrijeme i novac. Kako bismo dobili informacije o novim stvarima najčešće o njima raspravljamo s osobama iz okoline. Ukoliko znamo da nam je netko blizak po ukusu, njegovu preporuku uvažavamo više i obratno, te u skladu s tim rangiramo stvari i odlučujemo.

Devedesetih godina prošlog stoljeća trgovine, knjižare i kina pronašli su svoje mjesto u virtualnom svijetu što im je omogućilo veću ponudu proizvoda od one na fizičkim policama (engl. *Long Tail phenomen* [1]). Veliki broj proizvoda i ograničen prikaz informacija o proizvodima korisnicima je otežao pretragu i odlučivanje o njima. Također, trgovcu u klasičnoj trgovini moguće je postaviti pitanje kako bi nam preporučio proizvod sličan onom koji smo nedavno kupili, ali gotovo je nemoguće bilo postaviti isto pitanje internet trgovini i dobiti željeni odgovor. Već na samom početku razvoja internet trgovina, kina i knjižara bilo je očito da standardni način preporuke i otkrivanja novih stvari ima velike nedostatke i ograničenja, a posebice pri razmjeni novih informacija. Za više informacija vidi [1].

Upravo je to potaknulo istraživače na kreiranje automatiziranih sustava koji bi pomogli korisniku pretražiti proizvode i ubrzati *on-line* kupnju, a istovremeno bili pouzdani i točni. Kako bi ti sustavi postali što sličniji pravim trgovcima i korisnikovim poznanicima trebali bi naučiti ukus svakog pojedinog korisnika ili prepoznati korisnike sa sličnim ukusom te na temelju tih informacija svakom korisniku pružiti personaliziranu preporuku o onome što bi sljedeće mogao pregledati i kupiti.

Srećom, takvi sustavi su ubrzo razvijeni i danas se nalaze u pozadini gotovo svake internetske trgovine, knjižare, aplikacije za gledanje filmova ili slušanje glazbe. Prilikom posjeta *YouTube-u*, *Amazon-u* ili *Netflix-u* već na početnoj stranici aplikacije prikazuje se sadržaj koji bi nam se mogao svidjeti, a u njihovoj pozadini nalaze se spomenuti sustavi koje nazivamo *sustavi za preporuke*.

Sustavi za preporuke zajedničko je ime za algoritme koji na temelju informacija o korisnicima i proizvodima daju korisnicima preporuke za nove proizvode koji bi im se mogli svidjeti. Dvije su osnovne kategorije sustava za preporuke, kolaborativno filtriranje (engl. *Collaborative Filtering*, CF) i kontekstno bazirani sustavi za preporuke (engl. *Content Based*, CB). Algoritmi iz skupine kontekstno baziranih sustava za preporuke stvaraju preporuku na temelju proizvoda koje je korisnik kupio ili pozitivno rangirao, te preporučuju proizvode sličnoga tipa. Suprotno tome, sustavi za preporuke bazirani na kolaborativnom filtriranju stvaraju preporuku na temelju usporedbe s ostalim korisnicima i preporučuju proizvode koje su njemu slični korisnici kupili ili pozitivno rangirali. Kombinaciju kontekstno baziranih sustava i kolaborativnog filtriranja nazivamo hibridni sustavi za preporuke.

U ovom radu opisani su specijalno sustavi za preporuke bazirani na kolaborativnom filtriranju, odnosno jedan tip takvih sustava. Drugo poglavlje pruža uvid u povijesni pregled sustava za preporuke te su navedene njihove opće karakteristike i podjela. U trećem poglavlju opisan je jedan tip sustava za preporuke, točnije kolaborativno filtriranje bazirano na

[1] Prema [10] *Long Tail phenomen* je glavni razlog pojave sustava za preporuke, te predstavlja razliku između stvarnog i virtualnog svijeta, trgovina. Naziv dolazi od grafičkog prikaza gdje vertikalna osi predstavlja popularnost proizvoda, a horizontalna proizvode poredane prema popularnosti. Trgovine mogu ponuditi kupcima samo najpopularnije proizvode (krajnje lijeve uz y-os), dok virtualne trgovine otvaraju vrata proizvodima iz "repa".

modelu orijentiranom prema proizvodu (engl. *Item model based collaborative filtering RS*) i njegov algoritam temeljen na faktorizaciji matrice i smanjenju dimenzije.
Dodavanje novog korisnika u sustav i kreiranje preporuka za njega opisano je također u trećem poglavlju, a u četvrtom poglavlju je dana kratka analiza implementiranih algoritama iz trećeg poglavlja. Na kraju rada priloženi su Matlab kodovi implementiranih algoritama.

2 Sustavi za preporuke

2.1 Povijesni pregled

Povijesno gledano, možemo reći da je začetnik automatiziranih sustava za preporuke *Grundy*, računalni program za knjižnicu (1979. godine). *Grundy* je o čitateljskom tipu korisnika zaključivao na temelju korisnikovih odgovora na zadana pitanja. Prema odgovorima *Grundy* bi svrstavao korisnika u neki od poznatih stereotipa te mu prema dodijeljenom tipu pružao preporuke. Unatoč *Grundijevoj* primitivnosti on se smatra prvim korakom u svijet sustava za preporuke. Za više informacija vidi [18]. *Xerox PARC* razvio je 1992. godine *Tapestry*, platformu koja je korisnicima omogućila označavanje emailova i pretraživanje istih na temelju tih oznaka. Iako ni ovaj sustav nije bio automatiziran, važan je korak u razvoju sustava za preporuke jer se tada po prvi put spominje pojam kolaborativno filtriranje.
Prvi automatizirani sustav za preporuku je *GroupLens*, sustav kojeg je 1994. godine razvio istoimeni istraživački tim za preporuke za *UseNet News*[2]. *GroupLens* istraživački tim je 1997. godine razvio i sustav *MovieLens* za preporuke filmova korisnicima na temelju njihovih ocjena.(vidi [1])
Pandora.com[3] 2000. godine je predstavila još jedan važan sustav za preporuke *Music Genome Project* koji prepoznaje sličnosti žanrova, umjetnika i pjesama i na temelju naučenog korisnikova ukusa preporučuje mu iduće pjesme koje bi mogao poslušati.
Tijekom 90-ih godina sustavi za preporuke postali su sve zanimljivija tema na području strojnog učenja, interakcije čovjeka i računala i njihove razmjene informacija, što je očito iz velikog broja domena koje su u svojoj pozadini počele koristiti neki od takvih sustava. Naravno, u sustavima za preporuke uočen je potencijal za povećanjem prodaje i poboljšanjem iskustava kupaca pri posjetu internetskoj trgovini pa su postali i marketinška tema. Najpoznatija primjena sustava za preporuke je ona na domeni *Amazon.com*[4]. Sustav u pozadini trgovine *Amazon.com* na temelju kupljenih proizvoda, povijesti pregleda i trenutne akcije korisnika preporučuje korisniku proizvode koje bi sljedeće mogao kupiti. Osim što taj sustav olakšava korisniku pretragu jako velikog broja proizvoda[5] pridonosi i zaradi *Amazon-a* jer daje precizne podatke za lakše donošenje bitnih marketinških odluka.
Sustavi za preporuku najveću pozornost privukli su 2006. godine kada je *Netflix*[6] objavio *Netflix Prize*, nagradu od milijun dolara za kreiranje algoritma koji bi za 10% ubrzao njihov tadašnji algoritam.[7] Danas najpoznatiji online sustav za iznajmljivanje filmova, *Netflix*, pokrenut je 1999. godine, a njegov izvorni sustav za preporuke poznat je pod imenom *Cinematch*. *Netflix prize* problem riješen je tek 2009. godine kada su snage ujedinili natjecatelji koji su do tada bili najbliži rješenju u tim pod imenom *BellKor's Pragmatic Chaos*, a njihov algoritam detaljno je opisan u [15].
Danas poznajemo nekoliko osnovnih vrsta sustava za preporuke i njihovih tipova koji su predstavljeni u nastavku.

[2] `https://usenet-news.net`
[3] `https://www.pandora.com`
[4] `https://www.amazon.com`
[5] Prema [20] Amazon je u 2015. godini imao 488 milijuna proizvoda samo na Američkom tržištu.
[6] `https://www.netflix.com`
[7] Ubrzanje za 10% odnosilo sa na poboljšanje pogreške algoritma za 10% na uzorku iz Netflix baze, za više vidi [12].

2.2 Općenito o sustavima za preporuke i osnovni pojmovi

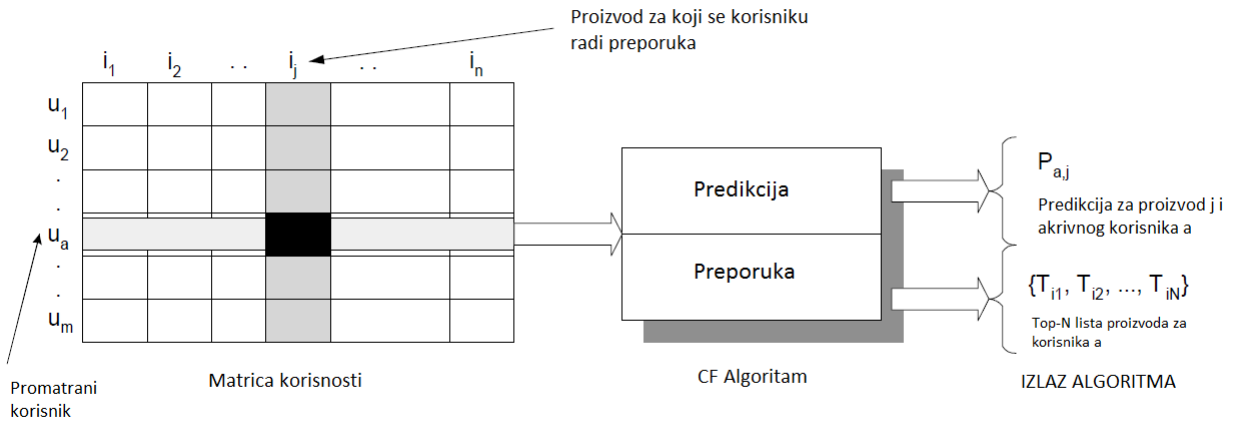
Sustavi za preporuke razvili su se kako bi se što bolje suočilo s eksponencijalnim rastom informacija, u smislu filtriranja podataka. Oni uče iz akcija korisnika o njihovom ukusu kako bi im u skladu s tim predložili nove proizvode, stoga ih istovremeno možemo promatrati kao dio rudarenja podataka i strojnog učenja. Području rudarenja podataka pripadaju zbog svoje namjene, olakšavaju korisniku pretraživanje podataka, a u područje strojnog učenja možemo ih smjestiti jer na temelju podataka o korisniku i proizvodima, uče o njima i prema tome generiraju preporuke.
Iako su sustavi za preporuke intenzivno razvijaju već tridesetak godina, tehnike koje se koriste u algoritmima za generiranje preporuke ostale su relativno jednostavne. Kako bismo objasnili tehnike korištene u algoritmima za preporuke i same algoritme potrebno je prvo navesti osnovne pojmove koji se u njima koriste. Korisnici i proizvodi su dvije osnovne klase entiteta i povezani su tako da korisnici imaju preference prema određenim proizvodima, a o njima se zaključuje iz podataka o korisniku ili ih daju sami korisnici. Podatci o vezi korisnika i proizvoda spremaju se u matricu koju nazivamo matrica korisnosti (engl. *Utility/rating matrix*), vidi tablicu 1. Svaki element matrice korisnosti predstavlja jedan par korisnik-proizvod

	Proizvod 1	Proizvod 2	Proizvod 3	Proizvod 4
Korisnik 1	4	?	3	5
Korisnik 2	?	5	4	?
Korisnik 3	5	4	2	?

Tablica 1: Matrica korisnosti

i on označava razinu koliko se promatranom korisniku sviđa dani proizvod. Razina sviđanja najčešće se prikazuje vrijednostima iz uređenog skupa elemenata kao što su prirodni brojevi od 1 do 5 koji mogu predstavljati rangiranje proizvoda zvjezdicama. Također, moguće je koristiti binarnu ili ternarnu skalu[8], dok je unarna skala nabolja za sustave internet trgovina jer jasno predstavlja povijest korisnikove kupnje. Ako korisnik i proizvod nisu povezani tada se na pripadnom mjestu u matrici nalazi nepoznata vrijednost, odnosno ne postoji izražen stupanj sviđanja korisnika prema proizvodu. Pretpostavka je da je matrica korisnosti rijetko popunjena matrica (engl. *sparse matrix*), a glavni cilj sustava je predvidjeti vrijednosti na praznim mjestima u matrici korisnosti, što je prikazano na slici 2.1. Pri modeliranju sustava za preporuke mogu se promatrati i svojstva proizvoda te iz tih sličnosti predvidjeti vrijednosti na praznim mjestima matrice korisnosti. Vrlo je bitno da nije potrebno izračunati svaku nepoznatu vrijednost u matrice korisnosti, nego je dovoljno otkriti samo neke vrijednosti za svaki redak, koje bi potencijalno mogle biti velike. Točnije, kako je navedeno u [12] bitno je pronaći takav podskup nepoznatih vrijednosti za svaki redak za koje predviđamo da će njihove vrijednosti biti među najvećima.

[8]Binarna skala je ona koja samo prikazuje je li se nešto svidjelo korisniku ili nije, ternarna uvodi i dodatnu vrijednost ukoliko je korisnik pogledao neki proizvod i nije ga rangirao, a unarna označava samo je li korisnik nešto pogledao/kupio ili ne.



Slika 2.1:
Ilustracija algoritma za preporuku

2.3 Kontekstno bazirani sustavi za preporuke i sustavi za preporuke bazirani na kolaborativnom filtriranju

Dvije su osnovne vrste sustava za preporuke, kontekstno bazirani i kolaborativno filtriranje. Kontekstno bazirani sustavi za preporuke generiraju preporuke tako da za svakog korisnika pronalaze proizvode slične onima koje je korisnik pregledao ili kupio i takvi proizvodi postaju dobri kandidati za preporuku. Kod ovakvih sustava rangiranje je manje bitno, puno je bitnije pronaći sličan sadržaj u skupu svih proizvoda. Ukoliko je takav sustav namijenjen npr. internet videoteci on će generirati preporuke korisniku s obzirom na žanr filmova koje je već pogledao i preporučiti mu bliske filmove. Tehnika kojom se pronalaze slični proizvodi u skupu svih proizvoda jest TF-IDF (engl. *Term Frequency – Inverse Document Frequency*) koja predstavlja mjeru koliko je važan neki izraz u skupu dokumenata. TF-IDF se u kontekstno baziranim sustavimma koristi kako bi sustav mogao zaključiti o ukusu pojedinog korisnika iz njegovih akcija. Ukoliko neki korisnik često gleda trilere, prema [1] TF-IDF analizom će se u bazi filmova filtrirati oni koji su po žanru trileri.
Ovdje se svaki korisnik promatra kao vektor čiji elementi predstavljaju svojstva proizvoda u sustavu. Vrijednost svakog elementa u tom vektoru su težine za pripadno svojstvo u odnosu na promatranog korisnika. Dobar primjer domene koja koristi kontekstno bazirani sustav jest prethodno spomenuta *Pandora.com*. Ovakvi sustavi za preporuke korisniku uzastopno preporučuju proizvode koji su slični po svojstvima, što je ujedno njihova prednost i mana (engl. *over-specialized search*)[9]. Korisnik mora na neki način sam pokazati interes prema nekom novom tipu proizvoda kako bi mu sustav u buduće mogao preporučiti proizvode s tim svojstvima, što nije slučaj kod kolaborativnog filtriranja.
Kod sustava za preporuke baziranih na kolaborativnom filtriranju (u nastavku CF sustavi) preporuke se generiraju na temelju korisnikova rangiranja proizvoda, a ne na temlju svojstava proizvoda. Kako je navedeno u [1] snovni cilj CF sustava je učiti o korisnikovu ukusu na temelju neke mjere (skale), te usporediti promatranog korisnika s ostalim korisnicima tog sustava i pronaći njemu slične korisnike. Važno je da skala, kojom će se rangirati proizvodi, dovoljno dobro odražava stav korisnika prema proizvodima koji se trebaju rangirati. Glavna

[9]Sustav neće pronaći proizvode koji nisu slični proizvodima za koje je korisnik do sada pokazao interes, nedostatak faktora iznanađenja u preporuci (engl. *Serendipity*, [13])

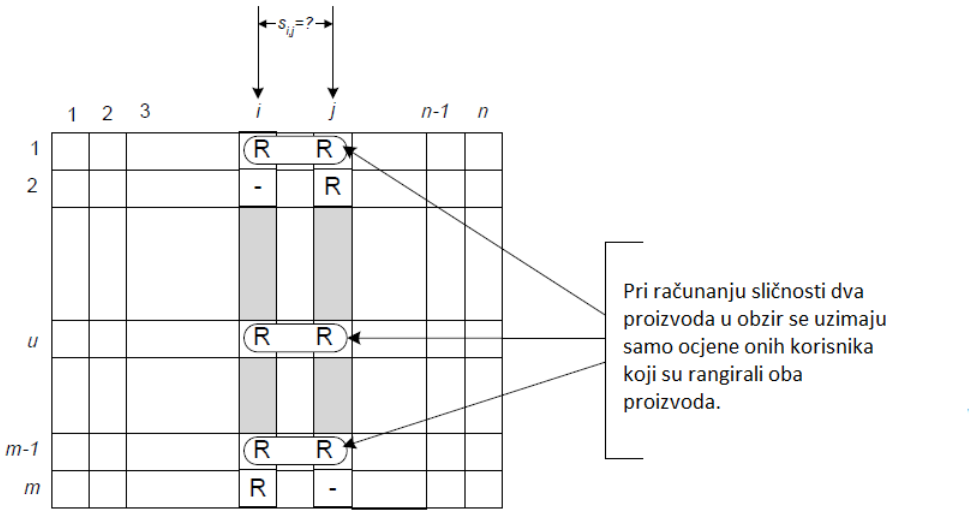
pretpostavka na kojoj se temelje CF sustavi jest da ukoliko isti proizvod dva korisnika isto rangiraju, oni će vjerojatno dijeliti isto mišljenje o još nekim proizvodima. Odnosno, ako se jednom od dva slična korisnika sviđa neki proizvod za koji drugi korisnik ne zna, prethodna tvrdnja govori da taj proizvod možemo preporučiti drugom korisniku i on će mu se vjerojatno svidjeti.
CF sustav koji se baziraju na pronalasku sličnih korisnika za nekog promatranog korisnika nazivamo sustavi orijentirani prema korisniku (engl. *User Based*), dok CF sustave koji pronalaze slične proizvode za neki promatrani proizvod nazivamo sustavi orijentirani prema proizvodu (engl. *Item Based*). Bez obzira na tip sustava, svi koriste funkciju sličnosti (engl. *similarity function*) za pronalazak sličnih korisnika ili proizvoda. Mnogo je različitih načina na koji se može definirati funkcija sličnosti, a njezina zadaća je da numerički predstavi udaljenost između dva proizvoda ili korisnika kako bi se moglo definirati njihovo susjedstvo. Neke od najčešće korištenih funkcija sličnosti su Euklidska udaljenost, Pearsonova korelacija i Cosinus udaljenost.
Također, CF sustave možemo podijeliti na one bazirane na modelu (engl. *Model Based*) i memorijski bazirane (engl. *Memory Based*). Memorijski bazirane CF sustave smatramo one kod kojih algoritam računa sličnosti u memoriji, bez prethodno generiranog modela i često se za njih kaže da su bazirani na susjedstvu (engl. *Neighbourhood Based*) jer u skupu najboljih susjeda pronalaze najbolje kandidate. CF sustavi bazirani na modelu generiraju model sustava i koriste matričnu faktorizaciju za pronalazak skrivenh faktora u matrici sličnosti što je detaljno objašnjeno u poglavlju 2.5.

2.4 CF sustavi orijentirani prema korisniku i orijenitirani prema proizvodu

Kako je već spomenuto, s obzirom na dimenziju po kojoj pronalaze sličnosti, po korisnicima ili proizvodima, CF sustavi se mogu podijeliti na one orijentirane prema korisniku ili proizvodu. U ovom poglavlju navedene su neke prednosti i nedostatci oba tipa CF sustava. Kako im ime govori algoritmi CF sustava orijentirani prema korisniku tragaju za sličnim korisnicima vodeći se pretpostavkom da *„slični korisnici vole slične proizvode“*, dok oni orijentirani prema proizvodu tragaju za proizvodima koji su slično rangirani od više korisnika vodeći se pretpostavkom da *„slične proizvode vole slični korisnici“*, kao na slici 2.2 gdje je grafički prikazana funkcija sličnosti. Iako su ova dva pristupa vrlo slična, ovisno o sustavu unutar kojeg su implementirani mogu uvelike utjecati na njegovu skalabilnost [10]. Ukoliko promatramo sustav sa puno većim brojem korisnika od broja proizvoda, algoritmi orijentirani prema korisniku su loš odabir jer će pronalazak najbližih susjeda biti puno teži u dimenziji korisnika nego što bi mogao biti u dimenziji proizvoda, kojih je manje u bazi. Nadalje, može se reći da je dimenzija korisnika najčešće više dinamička od dimenzije proizvoda, odnosno skup korisnika se češće mijenja od skupa proizvoda, pa je pouzdanije računati sličnosti unutar skupa proizvoda. No, ako promatramo sustav za preporuke u sklopu internetskih novina, tada se skup proizvoda (novosti) sigurno češće mijenja i veći je od skupa korisnika i u tom slučaju algoritmi orijentirani prema korisniku imaju prednost nad algoritmima orijentiranim prema proizvodu.
Još jedna prednost algoritama orijentiranih prema proizvodu jest da korisniku mogu dati jasnije objašnjenje preporuke. Činjenica da je neki proizvod preporučen korisniku jer je sličan ostalim proizvodima koji su mu se svidjeli je korisniku ugodnija od one da je proizvod

[10] Sposobnost sustava da se dobro ponaša pri povećanju korisnika i sve većem broju zahtijeva

preporučen jer se svidio sličnom korisniku.



Slika 2.2:
Grafički prikaz funkcije sličnosti

2.5 CF sustavi bazirani na modelu i memorijski bazirani

Uz podjelu CF sustava na orijentirane prema proizvodu i orijentirane prema korisniku, CF sustave možemo podijeliti na one bazirane na modelu i memorijski bazirane. Obje vrste algoritama moraju izračunati susjedstvo korisnika ili proizvoda, no razlika je što memorijski bazirani algoritmi računaju sličnosti unutar susjedstva u memoriji (engl. *in-memory*), dok algoritmi bazirani na modelu konstruiraju model i na temelju njega generiraju preporuke.
Memorijski bazirani algoritmi podatke o korisnicima i proizvodima drže u memoriji i direktno računaju preporuke, tj. sličnost među susjedstvom se računa *on-line* svaki put pri generiranju preporuke.
Kod pristupa baziranom na modelu model se kreira *off-line*, a preporuka se generira *on-line* na temelju modela. Ova vrsta algoritma doživjela je procvat tijekom istraživanja za *Netflix prize* 2006. godine, a pobjednički algoritam koristio je tehniku baziranu na modelu. Pri generiranju modela koristi se faktorizacija matrice korisnosti korisnika i proizvoda na dvije matrice u kojima je moguće zaključiti o nekim skrivenim faktorima u sustavu (npr. žanr filma), ali ti faktori često nisu izravno povezani sa stvarnim svojstvima proizvoda ili korisnika. Iako imaju dosta nedostataka, danas su najviše koriste CF sustavi bazirani na modelu koji koriste faktorizaciju matrice korisnika i proizvoda za kreiranje modela. Najveći nedostatak algoritama baziranih na modelu jest da bi se model trebao računati svaki puta kada dođe do veće promjene u početnoj matrici, odnosno svaki puta kada korisnik rangira neki od proizvoda ili se doda novi korisnik u sustav.

2.6 Izazovi CF sustava

Sustavi za preporuke mogu imati vrlo veliku ulogu u marketingu organizacije koja ga koristi, ukoliko je on pouzdan i brzo daje preporuke visoke kvalitete. Može se reći da je

porast dobiti organizacije koja koristi sustav za preporuke poželjna posljedica točnih i brzih preporuka. Takvi sustavi najčešće su integrirani u okruženjima s mnogo korisnika i proizvoda zbog čega naizgled laki zadatci postaju vrlo zahtjevni, a primjer tome su domene kao što su *Youtube*, *Netflix*, *Amazon*, *eBay*[11] i slično. U nastavku su navedeni neki od problema s kojima se, prema [5] sustavi za preporuke moraju nositi u realnom svijetu.

Slaba popunjenost podataka (engl. *Data sparsity*): Matrice sustava najčešće su vrlo velikih dimenzija zbog mnogo korisnika i proizvoda u okruženju. No, veliki broj korisnika i proizvoda nisu povezani pa su te matrice najčešće jako slabo popunjene (engl. *extremely sparse*). Slaba popunjenost najveći problem predstavlja pri dodavanju novog korisnika u sustav (engl. *cold start problem*), tj. kada korisnik još nema niti jedan rangiran proizvod i nema dovoljno informacija o njemu (engl. *new user problem*). Također, ako se doda novi proizvod, nemoguće je predložiti ga kao preporuku ukoliko ga nitko još nije rangirao (engl. *new item problem*).
Problem popunjenosti podataka može se riješiti smanjenjem dimenzije sustava, a najčešće korištena tehnika za to je SVD dekompozicija matrice sustava, definirana u poglavlju 3.1, pri čemu se uklanjaju manje bitni korisnici ili proizvodi. Taj pristup može dovesti do gubitka informacija o korisnicima ili proizvodima te posljedično pogoršati kvalitetu preporuke, no postoje načini kako se to može spriječiti.

Skalabilnost (engl. *Scalability*): Problem skalabilnosti sustava za preporuke nastaje ukoliko broj korisnika i proizvoda jako brzo raste. Primjerice, za sustav sa deset milijuna korisnika (M) i milijun proizvoda (N), kada bi CF algoritam bio složenosti $O(N)$ on bi bio prespor uzimajući u obzir da algoritam mora raditi jako brzo u stvarnom vremenu. Ovaj problem se isto može riješiti redukcijom dimenzije pomoću SVD dekompozicije. Problem nastaje kod dodavanja novog korisnika ili podataka u matricu, jer ono zahtijeva ponovno računanje dekompozicije dopunjene matrice, ali i on se može vješto riješiti inkrementalnim računanjem novog modela.
Što je sustav brži to je skalabilnost sustava veća, ali posljedica tome je smanjena kvaliteta preporuke. Upravo je to najveći izazov sustava za preporuke, tj. kako u isto vrijeme zadržati što veću kvalitetu preporuke i povećati skalabilnost sustava.

Sinonimi (engl. *Synonymy*): Pod sinonime podrazumijevamo proizvode koji su isti ili jako slični, ali imaju drugačiji naziv. Sustavi za preporuke ne mogu prepoznati takva svojstva, te sinonime tretiraju kao različite proizvode. Postoji mnogo tehnika koje pokušavaju automatski prepoznati sinonime i otkloniti pojavu ovog problema, no sve one utječu na performanse sustava za preporuke. Tehnika kojom se brzo i jednostavno može riješiti problem sinonima jest LSI (engl. *Latent semantic indexing*) koja se temelji na SVD dekompoziciji matrice korisnik-proizvod, a najefikasnija je pri velikim dimenzijama.

***Gray Sheep*:** *Gray Sheep* je izraz koji se koristi za korisnika koji nema konzistentno mišljenje, odnosno on se niti slaže niti ne slaže sa određenim skupinama korisnika. Za takve korisnike možemo reći da su indiferentni, a sustavi za preporuke im nisu od pomoći.
Korisnike za koje je zbog njihovog oprečnog mišljenja gotovo nemoguće generirati preporuku nazivamo *Black Sheeps*. Takvi korisnici predstavljaju problem i u stvarnom

[11] `http://www.ebay.com`

svijetu, stoga su prihvaćeni kao poznata iznimka i u sustavima za preporuke. Hibridni sustavi za preporuke mogu riješiti i ovakve probleme.

Shilling Attacks U sustavima gdje se ocjene proizvoda prikupljaju od svih korisnika, moguća je pojava zloupotrebe ocjenjivanja. Točnije, neki korisnik može dati mnogo dobrih ocjena za proizvode koji su mu iz nekog razloga od interesa, ali suprotno može nekim proizvodima namjerno dati loše ocjene kako bi im smanjio rejting (npr. *Youtube*). Istraživanjima je pokazano kako su sustavi orijentirani prema proizvodu manje podložni takvim napadima nego sustavi orijentirani prema korisniku, kako je navedeno u [5].

U nastavku ovoga rada opisana je inkrementalni CF algoritam za preporuke koji se temelji na SVD dekompoziciji matrice korisnik-proizvod, a ima tendenciju visoke skalabilnosti i visoke kvalitete preporuke.

3 SVD algoritmi za CF sustave bazirane na modelu orijentiranom prema proizvodu

3.1 Dekompozicija na singularne vrijednosti (SVD)

Dekompozicija na singularne vrijednosti ili SVD (engl. *Singular Value Decomposition*) je metoda matrične dekompozicije koja danu matricu $A \in \mathbb{R}^{m\times n}$ faktorizira na matrice U, S i V tako da je $A = USV^T$ i matrice $U \in \mathbb{R}^{m\times m}$ i $V \in \mathbb{R}^{n\times n}$ ortogonalne, a matrica $S \in \mathbb{R}^{m\times n}$ dijagonalna. Prema [11] vrijede sljedeće tvrdnje.

Definicija 1. Neka je $A \in \mathbb{R}^{m\times n}$ za $m, n \in \mathbb{N}$. Rastav matrice

$$A = USV^T$$

zovemo *dekompozicija na singularne vrijednosti matrice* A, ako su $U \in \mathbb{R}^{m\times m}$ i $V \in \mathbb{R}^{n\times n}$ ortogonalna, a $S \in \mathbb{R}^{m\times n}$ dijagonalna

$$S = diag(\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)}),$$

pri čemu vrijedi $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{min(m,n)} \geq 0$, a brojeve $\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)}$ zovemo singularne vrijednosti matrice A. Stupce matrice U zovemo lijevi, a stupce matrice V desni singularni vektori matrice A. Na slici 3.1 grafički je prikazana SVD dekompozicija matrice A.

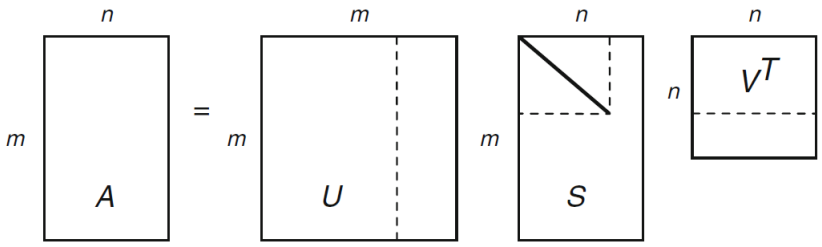
Za matricu $Q \in \mathbb{R}^{n\times n}$ kažemo da je ortogonalna ukoliko je $Q^TQ = I$, odnosno ako su stupci matrice Q ortogonalni s obzirom na standardni skalarni umnožak.
Sljedeći teorem govori o egzistenciji dekompozicije na singularne vrijednosti.

Teorem 1. *Ako je* $A \in \mathbb{R}^{m\times n}$*, tada postoje ortogonalne matrice* $U \in \mathbb{R}^{m\times m}$ *i* $V \in \mathbb{R}^{n\times n}$ *tako da je*

$$A = USV^T,\ S = diag(\sigma_1, \sigma_2, \ldots, \sigma_{\min(m,n)}), \tag{1}$$

pri čemu vrijedi $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_{\min(m,n)} \geq 0$.

Dokaz Za više vidi [11], str.112 Teorem 7.2 .



Slika 3.1:
Ilustracija SVD dekompozicije matrice A.

Napomena 1. Sukladno formuli (1) matricu A sada možemo zapisati kao $A = \sum_{i=1}^{\min(m,n)} u_i\sigma_i v_i^T$, gdje su vektori u_i i v_i vektori stupci matrica U i V i σ_i i-ta singularna vrijednosti matrice A. Ova tvrdnja biti će korisna u nastavku, točnije u teoremu 3.

Primjer 1. *Prema definiciji 1 i teoremu 1 sada može se izračunati SVD dekompozicija matrice* $A \in \mathbb{R}^{3\times 2}$ *zadane s:* $A = \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ \sqrt{2} & -\sqrt{2} \end{bmatrix}$ *i ona je dana s:*

$$A = USV^T = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & \sqrt{2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$$

Teorem 2. *Neka je* $A \in \mathbb{R}^{m\times n}$ *matrica sa dekompozicijom na singularne vrijednosti* $A = USV^T$ *i singularnim vrijednostima* $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0$. *Tada vrijedi:*

1. *rang matrice A je r, tj.* $rang(A) = r$,
2. $\mathcal{R}(A) = span(u_1, \ldots, u_r)$, *odnosno stupci matrice U,* $u_1, \ldots, u_r$ *razapinju sliku*[12] *od A*
3. $\mathcal{N}(A) = span(v_{r+1}, \ldots, v_n)$, *odnosno stupci matrice V,* $v_{r+1}, \ldots, v_n$ *razapinju jezgru* [13] *od A*

Dokaz:

1. Mratrice U i V^T su regularne, pa je tada

$$rang(A) = rang(S) = r.$$

2. Znamo da je $\mathcal{R}(A) = span(Av_1, \ldots, Av_n)$, jer je $v_1, \ldots, v_n$ baza i svaki linearni operator je jedinstveno određen svojim djelovanjem na bazu. Stoga je $span(Av_1, \ldots, Av_n) \subseteq \mathcal{R}(A)$. No, za $1 \leq i \leq r$ vrijedi:

$$Av_i = USV^T v_i = USe_i = \sigma_i U e_i = \sigma_i u_i \text{ i } \sigma_i > 0. \tag{2}$$

   Vektori u_i su linearno nezavisni, pa potprostor

$$span(\sigma_1 u_1, \ldots, \sigma_r u_r) = span(u_1, \ldots, u_r)$$

   ima dimenziju r kao i $\mathcal{R}(A)$, odnosno $\mathcal{R}(A) = span(u_1, \ldots, u_r)$.

3. Analogno kao u raspisu (2) imamo da je $Av_i = 0$, za $r+1 \leq i \leq n$, tada je

$$span(v_{r+1}, \ldots, v_n) \subseteq \mathcal{N}(A).$$

   Prema teoremu o rangu i defektu iz [19] koji kaže da je $defekt(A) = n - range(A) = n - r$ slijedi da je dimenzija potprostora $span(v_{r+1}, \ldots, v_n)$ ista kao i dimenzija od $\mathcal{N}(A)$. ■

[12] Slika operatora $A \in L(V, W)$ je potprostor definiran kao $\mathcal{R}(A) = ImA = \{Av : v \in V\} \leq W$, a još se naziva područje vrijednosti operatora A

[13] Jezgra operatora $A \in L(V, W)$ je potprostor definiran kao $\mathcal{N}(A) = KerA = \{v \in V : Av = 0\} \leq V$, a još se naziva nulprostor operatora A

Uočimo, ukoliko je matrica $A \in \mathbb{R}^{m\times n}$ ranga $r = n$ tada vektori stupci matrice V čine bazu za $\mathbb{R}^n$ i vektori stupci matrice U čine bazu za $\mathbb{R}^m$, a zapis operatora A, koji preslikava vektor $x \in \mathbb{R}^n$ u vektor $Ax = y \in \mathbb{R}^m$, u tom paru baza je dijagonalna matrica S.
Za matricu A^TA lako se može vidjeti da su svojstvene vrijednosti oblika σ_i^2, odnosno kvadrati pripadnih svojstvenih vrijednosti matrice A, a odgovarajući svojstveni vektori su desni singularni vektori matrice A, v_i.

$$A^TA = VS^TU^TUSV^T = VS^2V^T$$

Slično se može pokazati i za matricu AA^T, tj.

$$AA^T = USV^TVS^TU^T = US^2U^T.$$

Jasno je da su svojstvene vrijednosti matrice AA^T tada jednake σ_i^2, a svojstveni vektori za pripadne svojstvene vrijednosti su lijevi singularni vektori matrice A, tj. u_i za $i = 1, \ldots, \min(m, n)$.
U poglavlju 2.6 navedeno je kako se problemi sustava za preporuke vezani uz skalabilnost efikasno rješavaju redukcijom dimenzije matrice korisnika i proizvoda. Prema [17] u nastavku je naveden Teorem 3 koji govori da je pomoću SVD dekompozicije uistinu moguće matricu aproksimirati matricom nižeg ranga tako da je navedena aproksimacija ujedno i najbolja moguća, a pogreška aproksimacije je dana u smislu dva matrične norme.

Teorem 3. ***[Eckart–Young–Mirsky]*** *Neka je dana matrica $A \in \mathbb{R}^{m\times n}$ i dekompozicija na singularne vrijednosti matrice A sa $A = USV^T$ kao u Teoremu 1. Neka je $r = rang(A) \leq p = \min(m, n)$ i definiramo*

$$A_k = \sum_{i=1}^{k} u_i\sigma_i v_i^T \tag{3}$$

gdje je $k < r$. Tada je

$$\min_{rang(B)=k} \|A - B\|_2^2 = \|A - A_k\|_2^2 = \sigma_{k+1}^2 \tag{4}$$

Dokaz (Prema [3]) Potrebno je pokazati dvije tvrdnja, prvo da je $\|A - A_k\|_2 = \sigma_{k+1}$ i drugo da je A_k najbolja aproksimacija ranga k za matricu A. Neka u_i i v_j označavaju odgovarajuće stupce matrica U i V, redom.

- Matrica A_k je prema konstrukciji ranga k te vrijedi da je $\|A-A_k\|_2 = \|\sum_{i=k+1}^{n} \sigma_i u_i v_i^T\|_2 = \sigma_{k+1}$
- Pretpostavimo da je matrica B proizvoljna matrica ranga k, $B \in \mathbb{R}^{m\times n}$. Mogu se pronaći ortogonalni vektori $x_1, \ldots, x_{n-k}$ tako da oni razapinju jezgru od B, tj. $\mathcal{N}(B) = span\{x_1, \ldots x_{n-k}\}$. Pogledamo li presjek:

  $$span\{x_1, \ldots, x_{n-k}\} \bigcap span\{v_1, \ldots, v_{k+1}\} \neq \emptyset.$$

  Neka je z jedinični vektor koji se nalazi u navedenom presjeku. Kako je $Bz = 0$ i $Az = \sum_{i=1}^{k+1} \sigma_i(v_i^T z)u_i$ slijedi:

  $$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2(v_i^T z)^2 \geq \sigma_{k+1}^2.$$

  Ovime je pokazano da je matrica A_k najbolja aproksimacija ranga k za matricu A.
  Uočimo još da $\|A - B\|_2^2 \geq \|(A - B)z\|_2^2$ vrijedi zbog definicije inducirane norme $\|\cdot\|$ na $\mathbb{R}^{n\times n}$ (za detalje pogledati [11] str. 9).■

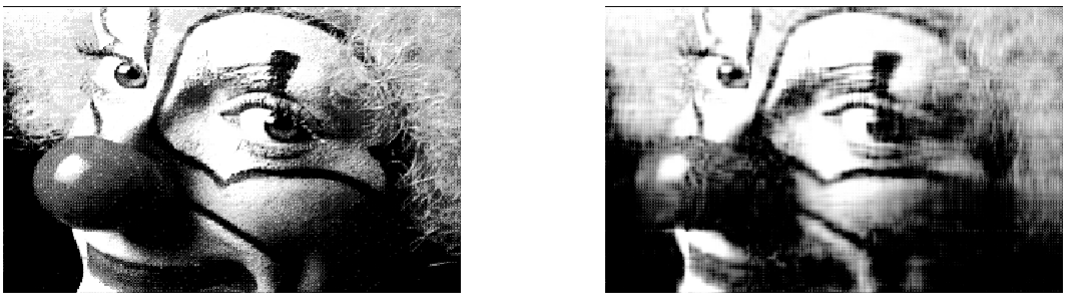
Matrica A_k sastavljena je od trojki koje pripadaju k-najvećim singularnim vrijednostima, (u_i, σ_i, v_i^T) i to je aproksimacija nižeg ranga matrice A u smislu 2-norme. Odnosno, to je najbolja k-aproksimacija nižeg ranga matrice A za bilo koju unitarnu invarijantnu normu[14], štoviše vrijedi i da je:

$$\min_{rang(B)=k} \|A - B\|_F^2 = \|A - A_k\|_F^2 = \sigma_{k+1}^2 + \ldots + \sigma_p^2.$$

Ovaj teorem je od velike važnosti za CF sustave jer pokazuje da je moguće zadržati samo najvećih k od r singularnih vrijednosti, za $k \ll r$ i daje najbolji mogući način kako to učiniti. Matrica S tada se očito aproksimira dijagonalnom matricom S_k kojoj su na dijagonali preostalih k vrijednosti, dok se matrice U i V aproksimiraju matricama U_k i V_k. Matrica U_k nastaje uklanjanjem $(r-k)$ stupaca iz matrice U, dok uklanjanjem $(r-k)$ redaka matrice V nastaje matrica V_k i tada je $A_k = U_k S_k V_k^T$. Na slici 3.1 iscrtane linije označavaju nove dimenzije matrica nakon redukcije.

Napomena 2. U praksi se često javljaju matrice koje se mogu efikasno aproksimirati matricom nižeg ranga. Najbolji takav primjeri jest kompresija slike jer se u tom slučaju javlja veliki broj kolinearnosti, što je prikazano u primjeru 2 te se i za vrlo male k ne gubi značajno na kvaliteti.

Primjer 2. *Ovaj primjer ilustrira svojstva iz teorema 3. Za potrebe primjera korišten je programski paket Matlab i njegova ugrađena slika clown.mat, a ideja je prikazati koliko se razlikuju učitana slika pohranjena u matrici* $X \in \mathbb{R}^{200\times320}$, $rank(X) = 200$ *i slika pohranjena u matrici* X_k *koja je dobivena najboljom aproksimacijom nižeg ranga matrice* X *za* $k = 20$. *Apsolutna pogreška vrijednosti dobivene formulom (4) i k+1. singularne vrijednosti matrice* X *iznosi* 2.2737×10^{-13}. *Na slici 3.2 su prikazane učitana slika pohranjena u matrici* X *(lijevo) i slike nakon aproksimacije pohranjene u matrici* X_k *(desno). Za detalje implementacije vidi Prilog 1.*



Slika 3.2:
Ilustracija najbolje matrične aproksimacije nižeg ranga na primjeru slike clown.mat. Lijeva slika je pohranjena u matrici $X \in \mathbb{R}^{200\times320}$, *a desna je pohranjena u matrici* X_{20} *koja je najbolja aproksimaciji ranga* 20 *za matricu* X.

3.2 Ocjena kvalitete preporuke

Jedna od najbitnijih svojstava sustava za preporuke je kvaliteta preporuke. Kvaliteta preporuke može se mjeriti na više različitih načina, a prema [5] osnovni tipovi su: *metrike obzirom na predikciju* (engl. *predictive accurancy metrics*), *metrike obzirom na rang* (engl.*rank*

[14]Prema [11] unitarna matrica u $\mathbb{R}^{n\times n}$ je isto što i ortogonalna, a za normu $\|\cdot\|$ kažemo da je unitarno invarijantna ako je $\|UAV\| = \|A\|$ za unitarne matrice U i V.

accurancy metrics) i *normalizirana metrika obzirom na udaljenost* (engl.*normalized distance-based performance metric*).
Tip metrike koji će se koristiti ovisi o namijeni sustava za preporuke. Najčešće korištena metrika je *prosječna apsolutna pogreška* (engl. *Mean Absolute Error*, MAE), iz skupa metrika obzirom na predikciju, koja računa prosječnu apsolutnu razliku između procijenjene vrijednosti i prave vrijednosti.
Ako se točnost računa za proizvod j i m je broj korisnika, $prediction_{ij}$ je procijena za element r_{ij} matrice sustava R, odnosno za i-tog korisnika i j-ti proizvod, tada se MAE računa po formuli:

$$MAE_j = \frac{\sum_{i=1}^{m} |prediction_{ij} - r_{ij}|}{m}, \text{ gdje je } j = 1, \ldots, n \tag{5}$$

Što je vrijednost za MAE manja, to je preporuka točnija, odnosno kvalitetnija. Iako ovakvi tipovi metrike mogu pomoći kako bi se odredila kvaliteta sustava, odnosno njegove preporuke, korisnicima ponekad nisu od velike koristi kao u slučaju kada korisnik želi nekakav novi, neočekivani rezultat preporuke koji nije u skladu s njegovim dotadašnjim preferencama. Za takve slučajeve mogu se koristiti metrike nekog drugog tipa, a u ovom radu koristi se navedena MAE metrika pri analizi kvalitete preporuke u ovisnosti o parametrima promatranog algoritma.

3.3 Algoritam zasnovan na SVD dekompoziciji s redukcijom dimenzije

3.3.1 Modeliranje i algoritam

Standardni algorimi za preporuke podijeljeni su u dva osnovna koraka. Prvi korak predstavlja *off-line* izgradnju modela sustava (engl. *model-building step*), dok drugi korak (engl. *execution step*) radi *on-line* i računa preporuke. Prema [4], algoritam koji će biti predstavljen u nastavku sastoji se od tri osnovna koraka: definiranje matrice korisnika i proizvoda, formuliranje susjedstva i generiranje preporuke.
Prije iskaza samoga algoritma potrebno je uvesti osnovne oznake:

- u_i označava i-tog korisnika u sustavu, a i_j označava j-ti proizvod u sustavu tako da je $i \in \{1, \ldots, m\}$ i $j \in \{1, \ldots, n\}$.
- $R \in \mathbb{R}^{m \times n}$ je matricu m korisnika i n proizvoda sa elementima r_{ij} koji predstavljaju numeričku oznaku koliko se korisniku u_i sviđa proizvod i_j. Ukoliko korisnik nije rangirao neki proizvod, tada se na to mjesto u matrici R dogovorno stavlja 0.

Za korisnika u_a potrebno je kreirati preporuku, odnosno predvidjeti koliko će mu se svidjeti proizvod i_j, što se može postići sljedećim postupkom.

Algoritam 1 (prema [4])

1. Konstruirati matricu R korisnika i proizvoda, dimenzije $m \times n$ sa elementima r_{ij} koji predstavljaju numeričku oznaku koliko se korisniku u_i sviđa proizvod i_j
2. Eliminirati nepoznata polja u matrici R na sljedeći način:
   - Izračunati prosjek redaka, r_i za $i = 1, 2, \ldots, m$, $\overline{r_i}$ i prosjek svih r_j za $j = 1, 2, \ldots, n$, $\overline{r_j}$ matrice R tako da je $r_{ij} \neq 0$

- Nepoznate vrijednosti u svakom stupcu r_j (one koje su jednake 0) zamijeniti sa prethodno izračunatim prosjekom toga stupca, $\overline{r_j}$ i rezultat ovog koraka je popunjena $R_{filled-in}$ matrica
- Svakom elementu matrice $R_{filled-in}$ oduzeti prosjek $\overline{r_i}$ pripadnog retka r_i matrice R i rezultat ovog koraka je normalizirana R_{norm} matrica

3. Izračunati SVD dekompoziciju matrice R_{norm}, $R_{norm} = USV^T$ gdje su U, S i V redom dimenzija $m \times m$, $m \times n$ i $n \times n$.

4. Reducirati dimenziju matrice R_{norm} tako da se zadrži samo k najvećih singularnih vrijednosti, $k \ll rang(R_{norm})$, a aproksimacija matrice R_{norm} u novoj dimenziji je matrica $R_k = U_k S_k V_k^T$ sa elementima $\hat{r}_{ij}$, gdje su $U_k \in \mathbb{R}^{m \times k}$, $S_k \in \mathbb{R}^{k \times k}$ i $V_k \in \mathbb{R}^{k \times n}$

5. Izračunati $\sqrt{S_k}$, te $U_k\sqrt{S_k}^T$ koja predstavlja m korisnika i $\sqrt{S_k}V_k^T$ koja predstavlja n proizvoda u k dimenzionalnom prostoru. Matrica $\sqrt{S_k}V_k^T$ dimenzije $k \times n$, k pseudo korisnika i n proizvoda, je od velikog značaja jer njezini elementi w_{ij} predstavljaju numeričku oznaku koliko se pseudo korisniku u_i sviđa proizvod i_j

6. Kreirati susjedstvo za promatrani proizvod:

   - Izračunati sličnost između proizvoda i_j i i_f pomoću funkcije sličnosti, prilagođene cosinus sličnosti[15] (engl. *Adjusted Cosine Similarity*):

$$sim_{jf} = adjcosr_{jf} = \frac{\sum_{i=1}^{k} w_{ij} \cdot w_{if}}{\sqrt{\sum_{i=1}^{k} w_{ij}^2 \sum_{i=1}^{k} w_{if}^2}}$$

   - Obzirom na rezultate prilagođene cosinus sličnosti kreirati l-susjedstvo promatranog proizvoda tako da susjedstvo čine oni proizvodi najsličniji promatranom (kao na slici 3.3).

7. Izračunati predikciju za korisnika u_a i proizvod i_j prema sljedećoj formuli:

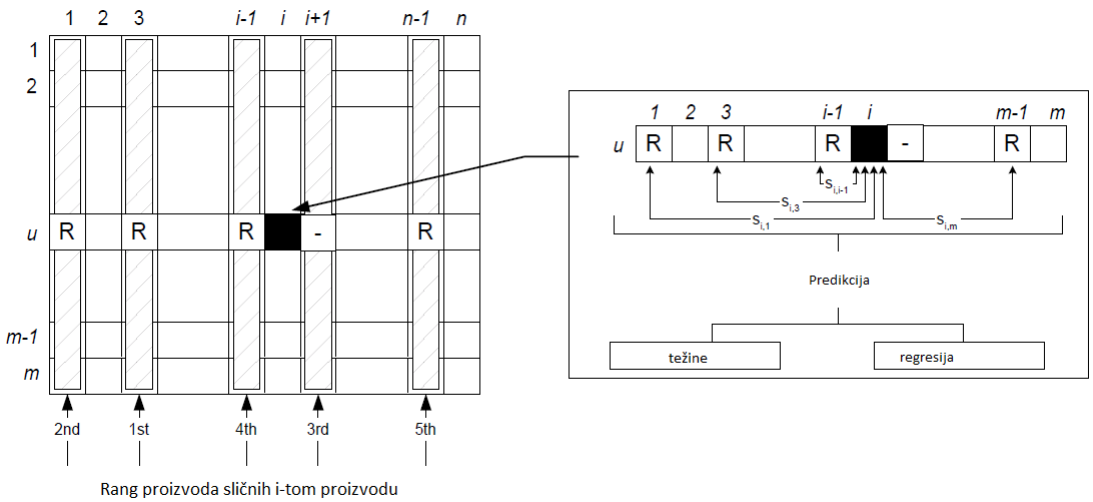
$$prediction_{aj} = \frac{\sum_{k=1}^{l} sim_{jk} \cdot (\hat{r}_{ak} + \overline{r_a})}{\sum_{k=1}^{l} |sim_{jk}|}$$

Algoritam 1 implementiran je u programskom paketu *Matlab*,a za više detalja pogledati Prilog 2.

3.3.2 Korektnost algoritma

U 1. i 2. koraku algoritma matrica sustava se konstruira i priprema za daljnji postupak normalizacijom. 3. korak algoritma računa SVD dekompoziciju matrice što je moguće zbog Teorema 1. Zbog tvrdnje iz Teorema 3 korektan je i 4. korak algoritma jer na taj način dobijemo najbolju aproksimaciju nižeg ranga početne matrice sustava. U skladu sa tvrdnjama iz Teorema 2 u 5. koraku algoritma možemo promatrati matricu $\sqrt{S_k}V_k^T$ dimenzije $k \times n$ kao matricu k pseudo korisnika i n proizvoda. Zbog svojstva SVD dekompozicije iz Teorema

[15] Standardna funkcija cosinus sličnosti dana je formulom: $cosr_{jf} = \frac{\sum_{i=1}^{k}(r_{ij}-\overline{r_j})\cdot(r_{if}-\overline{r_f})}{\sqrt{\sum_{i=1}^{k}(r_{ij}-\overline{r_j})^2 \sum_{i=1}^{k}(r_{ij}-\overline{r_f})^2}}$, gdje je $\overline{r_j}$ prosjek supca j matrice R



Slika 3.3:
Ilustracija izračunavanja 5-susjedstva za proizvod i

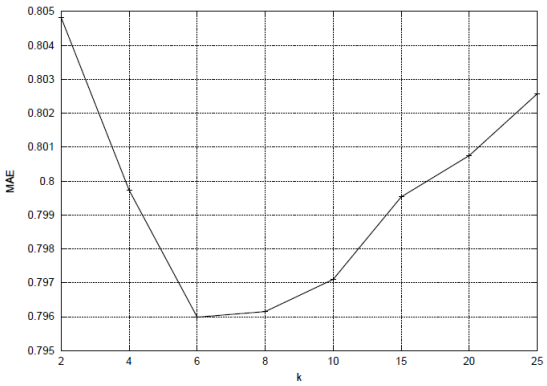
2 i Teorema 3, svi SVD bazirani CF algoritmi za preporuke zadržat će nakon dekompozicije i redukcije dimenzije svojstva početne matrice. 6. korak algoritma koristeći prilagođenu cosinus funkciju sličnosti računa sličnosti između proizvoda i susjedstvo zadanog proizvoda, dok 7. korak prema danoj formuli računa predikciju za zadanog korisnika i proizvod. U tom zadnjem koraku predikcija se za danog korisnika u_a i proizvod i_j računa uzimajući u obzir proizvode iz skupa l-susjeda za proizvod i_j, a svakom pripadnom pseudo korisniku $\hat{r}_{ak}$, za $k = 1, \ldots, l$ mora se dodati prosjek retka za pripadnog korisnika, tj. $\overline{r_a}$, koji je bio oduzet u 2. koraku pri normalizaciji.

3.3.3 Svojstva algoritma

Prema [4] složenost standardnog algoritma bez koraka redukcije je $\mathcal{O}(mn^2)$, a s redukcijom dimenzije na $k \ll m$, kako je navedeno u 4. koraku prethodnog algoritma, složenost je $\mathcal{O}(kn^2)$. Stoga se može zaključiti da je zbog redukcije početne matrice, što je moguće zbog Teorema 3, povećana skalabilnost sustava, smanjena potreba za količinom *on-line* memorije i ubrzano izvršavanje algoritma.
Zbog popunjavanja praznih mjesta u 2. koraku riješen je problem slabe popunjenosti podataka pa je popunjenost matrice korisnosti uvijek 100%.
Dva su osnovna parametra koja mogu utjecati na kvalitetu ovoga algoritma, a to su: parametar k koji određuje dimenziju reduciranog sustava i parametar l koji određuje dimenziju susjedstva promatranog proizvoda.
Kako je prethodno navedeno, parametar k određuje broj singularnih vrijednosti matrice S koje će se zadržati u matrici S_k, odnosno određuje rang matrice R_{red} i broj pseudo korisnika u reduciranom sustavu. Parametar k bira se tako da bude što manji kako bi utjecao na poboljšanje brzine i efikasnosti, a ujedno mora biti takav da reducirani sustav zadrži što više svojstava početnog sustava. Fiksiranjem parametra l može se eksperimentom za različite k odrediti onaj koji prema MAE daje najbolju preporuku. Slika 3.4 prikazuje rezultate eksperimenta koji je u [4] proveden na sustavu od 943 korisnika i 1682 proizvoda za različite vrijednosti parametra k, $k = 2, 4, 6, 8, 10, 15, 20, 25$ i fiksan parametar $l = 60$. Obzirom na promjenu parametra k mijenja se kvaliteta preporuke mjerena metrikom MAE. U početku

povećanjem parametra k MAE brzo doseže minimum za $k = 6$, odnosno kvaliteta doseže maksimum za $k = 6$, daljnjim povećanjem parametra k MAE raste i kvaliteta se smanjuje, pa se za najbolji k uzima $k = 6$.

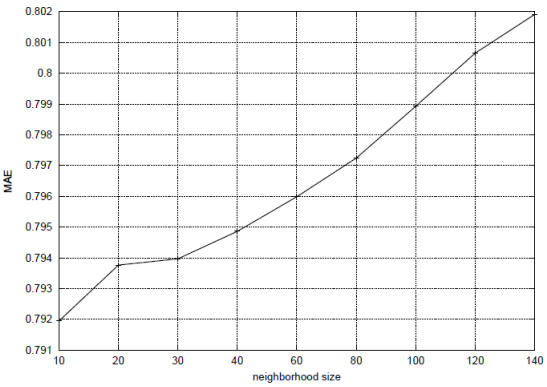
Veličina susjedstva, l, koje se promatra za određeni proizvod također ima utjecaj na kva-



Slika 3.4:
Eksperiment za različite vrijednosti parametra k

litetu preporuke. Najčešće slučaj je da je za blisko susjedstvo kvaliteta prema MAE niska, zatim proširenjem susjedstva MAE dostiže minimum, odnosno postiže se maksimalna kvaliteta za određeni broj susjeda. Nakon toga MAE postepeno raste što je prikazano na slici 3.5, a to znači da kvaliteta predikcije opada za veliko susjedstvo. Eksperiment sa slike proveden je u [4] na sustavu koji sadrži 943 korisnika i 1682 proizvoda. Za različite vrijednosti parametra l, $l = \{10, 20, 30, 40, 60, 80, 100, 120, 140\}$ i $k = 6$ ispitana je kvaliteta sustava u odnosu na MAE metriku, a za najbolje susjedstvo određeno je ono od 10 proizvoda. Optimalna vrijednost za oba parametra, k i l, ovisit će o skupu podataka nad kojima se koriste.

Nedostatak CF algoritama baziranih na SVD dekompoziciji jest da se vrijeme računanja



Slika 3.5:
Eksperiment za različite vrijednosti parametra l

SVD dekompozicije povećava povećanjem broja proizvoda i korisnika u sustavu. Nadalje, dodavanjem novog korisnika u sustav ili pri promjeni početne matrice zbog novih vrijednosti iznova se mora *off-line* računati SVD dekompozicija sustava, što nije trivijalan postupak. U nastavku je opisan algoritam koji inkrementalno osvježava početni sustav nakon dodavanja novog korisnika.

3.4 Inkrementalni algoritmi za osvježavanjem sustava bez ponovnog računanja SVD dekompozicije

3.4.1 Osvježavanje sustava *Folding-in* metodom

Glavni problem prethodnog algoritma jest kako dodati korisnika u sustav bez ponovnog računanja modela sustava. U matričnom jeziku, za dani vektor r_{new} potrebno je izračunati SVD matrice $[R; r_{new}]$ u ovisnosti o R. U [16] je dan postupak kako je moguće u sustav dodati novi vektor korisnika $r_{new} \in \mathbb{R}^{1\times n}$ bez ponovnog računanja cijele SVD dekopozicije.

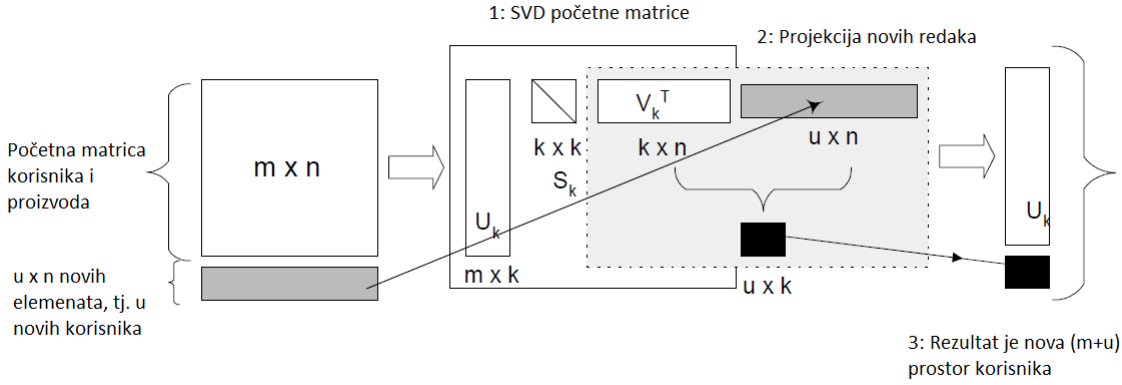
Algoritam 2

1. Novi vektor korisnika najprije je potrebno popuniti tako da se na prazna mjesta stavi prosjek pripadnog stupca. Srednje vrijednosti stupaca prethodno se spremaju u vektor $r_{mean} \in \mathbb{R}^{1\times n}$, a srednja vrijednost za novodobivene stupce računa se po sljedećoj formuli:
$$\overline{r'_j} = \frac{m \cdot \overline{r_j} + \sum_{i=m+1, r_{ij}\neq 0}^{(m+p)} r_{ij}}{m + \sum_{i=m+1, r_{ij}\neq 0}^{m+p} 1} \tag{6}$$
gdje je p broj redaka koji se dodaju u sustav, $\overline{r_j}$ prosjek j-tog stupca matrice R. Još je potrebno elemente u svakom novom retku normalizirati s prosjekom tog retka.

2. Vektor r_{new} aproksimirati koristeći k-dimenzionalan prostor kako bi se dobio vektor r_{new_k}, tj. izračunati $r_{new_k} = r_{new} V_k S_k^{-1}$.

3. Aproksimacija vektora r_{new}, odnosno r_{new_k} se dodaje postojećoj SVD dekompoziciji matrice R (engl. *folded-in*) tako što se doda kao posljednji redak matrice U_k. Rezultat je modificirana matrica $\hat{U}_k \in \mathbb{R}^{(m+1)\times n}$, dok matrice V_k i S_k ostaju nepromijenjene.

Navedeni postupak iz algoritma ilustriran je na slici 3.6. Nakon *folding-in* metode ponovno se izvršava postupak iz algoritma 1.



Slika 3.6:
Ilustracije postupka osvježavanja SVD dekompozicije matrice A nakon dodavanja novog korisnika, u ovom slučaju njih u

Postupak je moguće ponavljati više puta pri dodavanju jednog ili više korisnika. Nakon nekoliko iteracija, ovisno o količini i strukturi podataka koji su dodani, sustav će izgubiti na kvaliteti te će se morati ponovno *off-line* izračunati SVD dekompozicija za matricu R

i nove podatke. To je vrlo jeftin i dobar način osvježavanja za sustave u kojima se novi korisnici ne dodaju često. No, kako se matrice V_k i S_k nikada ne mijenjaju, kvaliteta preporuke se postupno smanjuje dodavanjem sve većeg broja korisnika, a ponovno računanje SVD dekompozicije sustava je neizbježno, ponekad i nakon samo par novih korisnika. Primjer dodavanja novog korisnika u sustav dan je u poglavlju 4.3, a detalji implementacije dostupni su u Prilogu 3.

3.4.2 Osvježavanje sustava *Update* metodom

Update metoda osvježavanja sustava pri dodavanju novog korisnika je složenija od *folding-in* metode, no krajnji rezultat ove metode je točna SVD dekompozicija novog sustava, do na pogrešku zaokruživanja, bez ponovnog računanja cijele SVD dekompozicije. Ovim postupkom se pri dodavanju novog korisnika u sustav odmah osvježava i prostor proizvoda u sustavu.
Pretpostavimo da u sustav želimo dodati novog korisnika kojeg možemo prikazati pomoću vektora $r_{new} \in \mathbb{R}^{1\times n}$. Matrica sustava je prethodno izračunata matrica $R_k = U_k S_k V_k^T$ i vektor r_{mean} je vektor sa srednjim vrijednostima svih stupaca matrice R. Zadaća *update* algoritma je osvježiti matrice U_k, S_k, V_k novim podatcima i vektor r_{mean}, te dodati nove podatke u sustav. Prema [12],[16] i [14] algoritam za *update* metodu možemo iskazati na sljedeći način:

Algoritam 3

1. Nove podatke potrebno je dopuniti i normalizirati. Vektor novog korisnika najprije je potrebno popuniti tako da se na prazna mjesta stavi prosjek pripadnog stupca. Srednje vrijednosti za novodobivene stupce računaju se po sljedećoj formuli:

$$\overline{r_j'} = \frac{m \times \overline{r_j} + \sum_{i=m+1, r_ij\neq 0}^{(m+p)} r_{ij}}{m + \sum_{i=m+1, r_{ij}\neq 0}^{m+p} 1},$$

   osvježava se vektor r_{mean} i novi redak normalizira se s prosjekom tog retka.

2. Izračunati projekciju $\hat{r}_{new} = (I_n - V_k V_k^T) r_{new}^T$, te SVD dekompoziciju matrice

$$\hat{S} = \begin{bmatrix} S_k & 0 \\ r_{new} V_k & \|\hat{r}_{new}\| \end{bmatrix} = U_k' S_k' V_k'^T$$

3. Izračunati $\hat{U_k} = \begin{bmatrix} U_k & 0 \\ 0 & 1 \end{bmatrix} V_k'$

4. Izračunati $\hat{V_k} = \begin{bmatrix} V_k & \frac{\hat{r}_{new}}{\|\hat{r}_{new}\|} \end{bmatrix} V_k'$

Napomena 3. Algoritam 3 nalaže računanje SVD dekompozicije matrice $\hat{S}$ pri svakom novom osvježavanju sustava. Iako je SVD dekompozicija inače skup postupak u ovom slučaju radi se o relativno maloj matrici čija dimenzija ovisi o parametru k i broju novih korisnika koji se dodaju u sustav.

Lako se može zaključiti da i ova metoda osvježavanja ima nedostatke, pogotovo ukoliko se odjednom u sustav dodaje mnogo korisnika. Prema [16] navedeni problem može se riješiti paralelizacijom algoritma, no to nije u opsegu ovog rada.
Metoda koja može smanjiti trošak osvježavanja sustava i zadržati kvalitetu preporuke je *folding-up* hibridna metoda koja je kombinacija *folding-in* i *update* metode. Ideja metode je da u poečtku koristi *folding-in* metodu do unaprijed određenog udijela novih korisnika u sustavu. Nakon toga da se pokreće *update* metoda nad novim korisnicima kako bi se sustav vratio u ravnotežu i osvježio prostor proizvoda, taj proces se iterativno nastavlja kombiniranjem tih metode.

4 Testiranje implementiranih algoritama

4.1 Podatci za testiranje

Osim teorijskih iskaza SVD algoritama za CF sustave, u ovome radu provedeni su eksperimanti za dane algoritme. Uz pomoć programskog paketa *Matlab*, implementirani je algoritam zasnovan na SVD dekompoziciji s redukcijom dimenzije i algoritam za osvježavanje sustava *folding-in* metodom.
Prije same implementacije, potrebno prikupiti i pripremiti podatke za testiranje. Testiranje algoritama provedeno je na bazi od 11 korisnika i 11 proizvoda, a podatci za matricu korisnosti R, prikazanu tablicom 2, prikupljeni su pomoću *Google ankete*[16]. Ispitani korisnicu su trebali ponuđene filmove rangirati ocjenom iz skupa $\{1, 2, 3, 4, 5\}$, a filmove koje nisu pogledali označiti sa 0.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
K1	3	0	5	0	3	4	0	0	0	0	0
K2	0	3	5	3	0	4	0	3	0	0	3
K3	4	0	5	3	3	4	4	0	4	0	0
K4	5	3	5	0	5	5	0	0	0	0	5
K5	4	4	5	5	4	5	0	3	0	3	3
K6	4	4	4	4	5	5	0	0	0	0	0
K7	0	5	0	0	4	0	0	3	0	0	0
K8	0	4	5	0	5	3	0	0	0	5	0
K9	5	4	0	5	3	4	5	0	5	4	0
K10	0	5	5	0	4	2	0	0	0	0	0
K11	3	0	5	4	5	0	4	5	2	0	5

Tablica 2: Matrica korisnosti za testiranje algoritama

Filmovi navedeni u matrici su redom: *The Martian(2015)*, *The Illusionist(2006)*, *Now You See Me(2013)*,*The Intouchables(2011)*, *Ice Age(2002)*, *Ted(2012)*, *The Man Who Knew Infinity(2015)*,*The Choice(2016)*, *The Secret Life Of Pets(2016)*, *Me Before You(2016)* i *Race(2016)*.

4.2 Algoritam zasnovan na SVD dekompoziciji s redukcijom dimenzije

Algoritam zasnovan na SVD dekompoziciji s redukcijom dimenzije implementiran je u programskom paketu *Matlab* prema Algoritmu 1. Za detalje implementacije vidi Prilog 2
U ovom poglavlju promatraju se rezultati implementiranog algoritma i njegova točnost. Prvo su navedeni osnovni rezultati algoritma neophodni za predikciju, zatim su dani primjeri predikcije za zadanog korisnika u_a i film i_j i svojstva predikcije za slične proizvode i korisnike te je na kraju algoritam testiran za raličite parametre k i l kako bi se odredili najbolji.

[16] Anketa: `https://goo.gl/forms/QPu36O4PK7FJznEt1`,
Rezultati ankete:
`https://docs.google.com/spreadsheets/d/1CL0j43vxyF6RVU0cp4pDvfJMGdME1fHtdm52QORUs94/edit?usp=sharing`

Kako je navedeno u algoritmu prvi korak kod generiranja preporuke zasnovane na SVD dekompoziciji s redukcijom dimenzije jest popunjavanje i noramlizacija matrice korisnosti. Funkcija koja normalizira matricu sustava prikazanu u tablici 2, u ovom slučaju `calculateRnorm` funkcija navedena u Prilogu 2, za danu matricu testnog sustava računa matricu R_{norm} kao:

$$R_{norm} = \begin{bmatrix} -0.75 & 0.25 & 1.25 & 0.25 & -0.75 & 0.25 & 0.583 & -0.25 & -0.083 & 0.25 & 0.25 \\ 0.5 & -0.5 & 1.5 & -0.5 & 0.6 & 0.5 & 0.833 & -0.5 & 0.167 & 0.5 & -0.5 \\ 0.143 & 0.143 & 1.143 & -0.857 & -0.857 & 0.143 & 0.143 & -0.357 & 0.143 & 0.143 & 0.143 \\ 0.333 & -1.667 & 0.333 & -0.6667 & 0.333 & 0.333 & -0.333 & -1.167 & -1 & -0.667 & 0.333 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0.333 & -1 & -0.333 & -1 & -1 \\ -0.333 & -0.333 & -0.333 & -0.333 & 0.667 & 0.667 & 0 & -0.833 & -0.667 & -0.333 & -0.333 \\ 0 & 1 & 0.889 & 0 & 0 & 0 & 0.333 & -1 & -0.333 & 0 & 0 \\ -0.4 & -0.4 & 0.6 & -0.4 & 0.6 & -1.4 & -0.067 & -0.9 & -0.733 & 0.6 & -0.4 \\ 0.625 & -0.375 & 0.514 & 0.625 & -1.375 & -0.375 & 0.625 & -0.875 & 0.625 & -0.375 & -0.375 \\ 0 & 1 & 1 & 0 & 0 & -2 & 0.333 & -0.5 & -0.333 & 0 & 0 \\ -1.125 & -0.125 & 0.8750 & -0.125 & 0.875 & -0.125 & -0.125 & 0.875 & -2.125 & -0.125 & 0.875 \end{bmatrix}$$

Prema algoritmu sljedeći korak je SVD dekompozicija matrice R_{norm} i redukciju dimenzije na prethodno zadanu dimenziju obzirom na parametar k. Metoda `KdimReduction` iz Priloga 2 računa k-dimenzionalnu redukciju matrice R_{norma} i rezultat je matrica $R_{red} = U_k S_k V_k^T$. Za $k = 4$ i prethodno dobivenu matricu R_{norm}, matrica R_{red} je dana s:

$$R_{red} = U_k S_k V_k^T = \begin{bmatrix} -0.2851 & -0.1262 & -0.1178 & 0.4282 \\ -0.3844 & -0.1205 & 0.1729 & -0.0128 \\ -0.2486 & -0.1736 & -0.0588 & 0.0629 \\ -0.3128 & 0.2237 & 0.5132 & -0.4167 \\ -0.3501 & -0.2353 & 0.3780 & 0.3606 \\ -0.1226 & 0.1704 & 0.3441 & -0.0788 \\ -0.3328 & -0.1106 & -0.1221 & 0.1566 \\ -0.3713 & 0.2016 & -0.2069 & -0.4905 \\ -0.1765 & -0.5131 & 0.0201 & -0.1395 \\ -0.3548 & -0.0353 & -0.5996 & -0.2072 \\ -0.2607 & 0.6983 & -0.1022 & 0.4185 \end{bmatrix} \begin{bmatrix} 3.8664 & 0 & 0 & 0 \\ 0 & 3.6269 & 0 & 0 \\ 0 & 0 & 3.2647 & 0 \\ 0 & 0 & 0 & 2.4602 \end{bmatrix}$$

$$\begin{bmatrix} 0.0658 & 0.0536 & -0.7301 & 0.0787 & -0.0512 & 0.1276 & -0.2113 & 0.5018 & 0.3571 & 0.0456 & 0.0928 \\ -0.3197 & -0.1508 & -0.1158 & -0.2074 & 0.4954 & -0.0745 & -0.2266 & 0.2738 & -0.6057 & 0.0383 & 0.2701 \\ 0.1327 & -0.5293 & -0.1323 & -0.0111 & 0.1231 & 0.7110 & -0.0550 & -0.2446 & -0.0663 & -0.2772 & -0.1410 \\ -0.3223 & 0.4019 & 0.3122 & 0.3174 & -0.1262 & 0.5609 & 0.1559 & 0.3840 & -0.1137 & -0.0980 & 0.1072 \end{bmatrix} \quad (7)$$

i to je model testnog sustava.

Primjer 3.

a) *Promatra li se općenito sličnost proizvoda u matričnom zapisu, vidi se da je matrica sličnosti proizvoda simetrična matrica s jedinicama na dijagonali. Iz tablice 2 može se vidjeti da su filmovi $F3$ i $F7$, $F1$ i $F9$ i $F8$i $F11$ najsličniji, što potvrđuje i funkcija sličnosti* `AdjustedCosSim` *navedena u Prilogu 2 i vrijedi:*

$$sim_{3,7} = 0.8223 \ , \ sim_{1,9} = 0.7467, \ i \ sim_{8,11} = 0.7884.$$

b) *Nakon što je izračunata sličnost proizvoda, može se odrediti l-susjedstvo za svaki film. Primjerice, za fiksni $k = 4$ 3-susjedstvo filma $F2$ čine filmovi $F5$, $F1$ i $F6$, te se film $F2$ nalazi u 3-susjedstvu filmova $F1$ i $F6$. Susjedstvo filmova određeno je pomoću funkcije* `neighborhood` *navedene u Prilogu 2.*

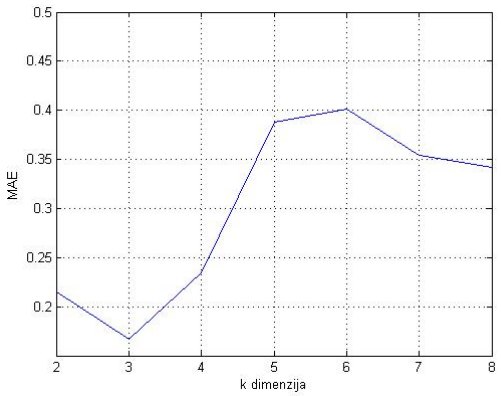
c) *Napokon je moguće izračunati predikciju za danog korisnika u_a i filma i_j, neka je promatrani korisnik $K7$ i film $F1$, The Martian(2015). Funkcija* `svdBasedPrediction` *navedena u Prilogu 2 uz parametre $k = 2$ i $l = 2$ za rezultat daje $svdBasedPrediction_{7,1} = 4.0177$. Točnije, algoritam predviđa de će korisnik $K7$ rangirati sa 4 film The Martian(2015).*

Ukoliko se susjedstvo povećava ili se parametar k poveća dobiva se nešto drugačiji rezultat, odnosno za $k = 3$ rezultat je 3.5215, a za $l = 3$ je 3.7676. Ovaj rezultat naveden je kao testni primjer u Prilogu 2.

Iz primjera 3.c očito je da parametri k i l bitno utječu na rezultat algoritma pa je važno provesti analizu za oba parametra i pronaći one koji najbolje odgovaraju sustavu.

Analiza parametra k:

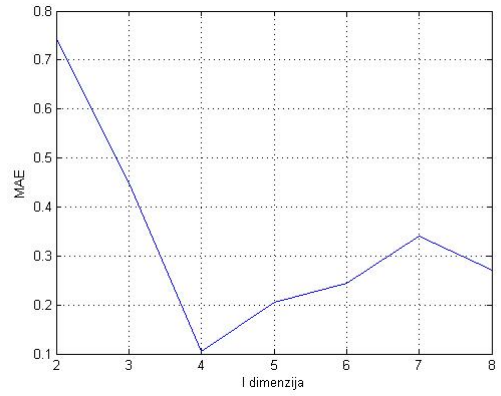
Testiranje za parametar k provedeno je za $k = 2, \ldots, 8$ i fiksan $l = 3$, a rezultat je prikazan na slici 4.1. Sukladno analizi, za najprikladniji k može se uzeti $k = 3$, odnosno najbolja aproksimacije matrice R je matrica ranga $k = 3$.



Slika 4.1:
Test za parametar k

Analiza parametra l:

Testiranje za parametar l provedeno je za $l = 2, \ldots, 8$ i fiksan $k = 3$, a rezultat je prikazan na slici 4.2. Analiziom je pokazano da je najprikladnije za parametar l uzeti $l = 4$. Točnije, ovdje je za susjedstvo nekog filma najprikladnije promatrati njemu četiri najsličnija filma.



Slika 4.2:
Test za parametar l

4.3 *Folding-in* metoda osvježavanja sustava

Pri dodavanju svakog novog korisnika u sustav, dodaje se novi redak u matricu korisnosti, no ideja *Folding-in* metode nije ponovno računanje SVD dekompozicije nego dodavanje korisnika u već reducirani sustav.

Primjer 4. *Ukoliko se u matricu sustava prikazanu u tablici 2 dodaje novi vektora* $r_{new} = K11$*, pretpostavka je da će tada preporuke za korisnika* r_{new} *biti ista kao i preporuka za postojećeg korisnika* $K11$*.*
Predikcija za korisnika $K11$ *i film* $F3$*, tj. Now You See Me(2013) iznosi* 2.7514*, a predikcija za novog korisnika* r_{new} *i film* $F3$ *iznosi* 2.6268*. U Prilogu 3 dana je implementacija ovog primjera.*

Odstupanje predikcije za novog korisnika rasti će dodavanjem sve većeg broja korisnika te bi se za sustav mogao promatrati i maksimalan broj korisnika koji se može dodati bez značajnog gubitka na kvaliteti. Kod većih sustava taj broj bi se mogao razmatrati, no u malim sustavima kao što je sustav iz primjera opadanje kvalitete je vidljivo već u samom startu. Osvježavanje sustava omogućuje dodavanje novog korisnika u sustav, a time rješava i *cold start* problem u CF sustavima.
Također, može se promotriti razliku u predikciji ukoliko se novi korisnik doda *folding-in* metodom ili se on dodaje u početni sustav ponovno računajući cijelu SVD dekompoziciju, što je prikazano u sljedećem primjeru.

Primjer 5. *Sustavu se dodaje novi korisnik* u_{12} *zadan s* $u_{12} = [4, 5, 4, 0, 0, 4, 4, 0, 4, 3, 3]$ *i promatra se predikcija za tog novog korisnika* $K12$ *i film* $F4$*, tj. film The Intouchables(2011). Korisnik se može dodati folding-in metodom, te u tom slučaju predikcija za novog korisnika i dani film iznosi* 3.3156 *. Ukoliko se novi korisnik dodaje standardnom metodom u sustav i ponovno se računa cijela SVD dekompozicija, predikcija iznosi* 3.4851*. Ovakav rezultat upućuje da dodavanje jednog novog korisnika folding-in metodom ne utječe bitno na kvalitetu predikciju. No, dodavanjem više korisnika kvaliteta se smanjuje pogotovo za male sustave kao što je ovaj testni primjer te je neophodno koristiti i metode koje pravilno osvježavaju cijeli sustav kao što je update ili još bolje folding-up metoda.*

5 Zaključak

Počevši od internetskih trgovina, preko aplikacija za preuzimanje glazbe do internet videoteka, sustavi za preporuke danas su neizbježan alat za komunikaciju virtualnih trgovina sa korisnicima i vrlo su moćan alat u modernoj razmijeni dobara. Sustavi za preporuke pomažu korisnicima pronaći proizvode koji bi im se mogli svidjet i preporučiti im proizvode slične onima koje su već kupili. Osim što su korisnicima od velikog značaja pri pretraživanju velikog broja proizvoda, sustavi za preporuke vrlo su korisni i samoj domeni koja ih koristi te mogu uvelike utjecati na zaradu i poboljšanje trgovine.
Razvojem interneta i povećanjem broja internetskih domena sve je više korisnika koji očekuju brz pristup podatcima koji su im od koristi. Može se reći da su sustavi za preporuke pod velikim pritiskom zbog eksponencijalnog rasta podataka i korisnika u virtualnom svijetu te se očekuju njihova učestala poboljšanja. Stoga je očito da su najveći izazovi današnjih sustava za preporuke baš skalabilnost i kvaliteta sustava, odnosno sposobnost sustava da se dobro ponaša pri povećanju broja korisnika i proizvoda i zadrži očekivanu kvalitetu. Različiti su načini kako poboljšati skalabilnost te je pokazano kako SVD dekompozicija matrice sustava sadrži bitna svojstva koja mogu na to utjecati.
U ovom radu promatran je specijalno jedan tip sustava za preporuke, odnosno kolaborativnom filtriranje bazirano na modelu orijentiranom prema proizvodu koji za generiranje preporuke koristi SVD dekompoziciju matrice sustava i njezina svojstva. Koristeći taj pristup sustav se može podijeliti na *off-line* i *on-line* dio kako bi se u potpuosti iskoristile sve prednosti SVD dekompozicije i ublažili njeni nedostatci. *Off-line* dio sustava zadužen je za zahtijevni dio računanja SVD dekompozicije matrice sustava i pripremu za brzo *on-line* računanje predikcije. Ukoliko se u tom dijelu sustava primijeni svojstvo SVD dekompozicije o najboljoj aproksimaciji nižeg ranga može se bitno utjecati na skalabilnost sustav. *On-line* dio sustava zadužen je za računanje najbližeg susjedstva nekog korisnika ili proizvoda i generiranje preporuke za danog korisnika i proizvod. Redukcijom dimenzije sustava za preporuke smanjuje se potreba za količinom *on-line* memorije i poboljšavaju se *on-line* performanse sustava, a time i performanse cijelog sustava.
Također, koristeći inkremenatalne SVD algoritme kao što je *folding-in* metoda moguće je brzo i efikasno dodati nove korisnike ili proizvode u sustav. Iako inkrementalni načini dodavanja novih korisnika ili proizvoda zahtjevaju malo vremena i online memorije, oni mogu dovesti do gubitka kvalitete te ih je potrebno kombinirati sa algoritmima koji u potpunosti zadržavaju kvalitetu sustava nauštrb brzini i memoriji, kao što je *update* metoda.
Premda je SVD dekompozicija vrlo snažan alat u algoritmima za preporuke, postoje i situacije u kojima su njezine mane izraženije od vrlina. Nadalje, postavlja se i pitanje koliko često i u kojem trenutku je potrebno ponovno pokrenuti *off-line* dio sustava te je li *update* metoda dovoljno brza i kvalitetna za velike sustava, što se ostavlja za buduća razmatranja.

6 Prilog

Prilog 1. Matlab kod za Primjer 2:

```
load clown % učitavanje clown.mat slike
whos % provjera dimenzije
[U S V a]=KdimReduction(X,20);
rank(X)
Xk=U*S*V;
cmap1 = contrast(X);
cmap2 = contrast(Xk);
subplot(1,2,1), imshow(X,colormap(cmap1))
subplot(1,2,2), imshow(Xk,colormap(cmap2))
format short
errorCalc = norm(X-Xk);
diff = abs(a-errorCalc)
```

Prilog 2. U nastavku su navedene funkcije implementirane u Matlabu za Algoritam 1.

```
% funkcija svdBasedPrediction za racunanje predikcije pomocu SVD
% dekompozicije racuna predikciju za zadanog korisnika "user"
% i proizvod "item" iz matrice "A" uz zadane paremetre:
% "kpred" parametar za novu dimenziju i "nn" za broj susjeda

    function [pred] = svdBasedPrediction (A, user, item, kpred,nn)
    Afil=fillInMatrix(A);
    NR=calculateRnorm(Afil,A);
    kpred=min(kpred,rank(A));
    [M N P]=KdimReduction(NR,kpred);
    Rred=M*N*P;
    [US SV]=calcSqMatrices(M,N,P);
    SimMat=AdjustedCosSim(SV);
    a=neighborhood(SimMat,item,nn);
    kk=length(a);
    topSum=0;
    absSum =0;
    for iter=1:kk
        topSum=topSum+SimMat(item,iter)*(Rred(user,iter)+vectorMean(A(user,:)));
        absSum=absSum+abs(SimMat(item,iter));
    end
    pred= topSum/absSum;

% funkcija fillInMatrix za danu matricu A racuna Rfil matricu koja je
% popunjena, tj.svaki element matrice A koji je jendak 0 u matrici
% Rfil se zamjenjuje aritmetickom sredinom stupca kojemu taj element
% pripada, a ostali elementi ostaju jednaki
```

```
    function [Rfil] = fillInMatrix (A)
    [nf mf]=size(A);
    Rfil = zeros(nf,mf);
    for ii=1:mf
        temp=vectorMean(A(:,ii));
        for jj=1:nf
            if A(jj,ii)==0
                Rfil(jj,ii)=temp;
            else
                Rfil(jj,ii)=A(jj,ii);
            end
        end
    end

% funkcija calculateRnorm za ulaznu matricu F i njenu popunjenu matrci
% Ffill racuna normaliziranu matricu Rnorm

    function [Rnorm] = calculateRnorm (Ffill, F)
    [n m]=size(F);
    Rnorm =zeros(n,m);
    for ii=1:n
        % svakom retku matrice Ffill oduzima se aritmeticka sredina
        % pripadnog retka matrice F
        Rnorm(ii,:) = Ffill(ii,:)-(ones(1,m)*vectorMean(F(ii,:)));
    end

% funkcija KdimReduction racuna SVD dekompoziciju matrice Am i
% za zadani kk vraca matrice Uk, Sk i Vk reducirane na dimenziju kk:

    function [U S V] = KdimReduction (Am, kk)
    [U S V]=svd(Am);
    V=V';
    U=U(:,1:kk);
    S=S(1:kk,1:kk);
    V=V(1:kk,:);

% funkcija calcSqMatrices racuna produkte matrica UK, SK, VK
% za 5. korak algoritma

    function [UKSK, SKVK] = calcSqMatrices(UK, SK, VK)
    UKSK=UK*(sqrtm(SK))';
    SKVK=sqrtm(SK)*VK;

% funkcija za danu matricu SVK racuna matricu slicnosti po
% "Adjusted Cosine Similarity" principu

    function [Asim]=AdjustedCosSim (SVK)
    [nsv msv]=size(SVK);
```

```
    Asim=zeros(msv,msv);
    for pp=1:msv
        for tt=1:msv
        Asim(pp,tt)=dot(SVK(:,pp),SVK(:,tt))/(norm(SVK(:,pp))*norm(SVK(:,tt)));
        end
    end

% funkcija racuna najblizih "neig" susjeda za proizvod "item"
% prema matrici slicnosti "Sim" i vraca vektor "neighbors" sa
% indeksima susjednih proizvoda poredanih silazno

    function [neighbors] = neighborhood (Sim, item, neig)
    [maxN ind]=sort(Sim(item,:));
    neighbors=ind;
    for ee=length(maxN):-1:neig+1
        neighbors(ee)=[];
    end

% funkcija racuna i vraca aritmeticku sredinu pozitivnih
% elemenata vektora a

    function [vecMean] = vectorMean (a)
    n=length(a);
    count=0;
    vecMean=0;
    vecSum=0;
    for ii=1:n
        if a(ii)~= 0
           count=count+1;
           vecSum = vecSum + a(ii);
        end
    end
    vecMean=vecSum/count;

% funkcija MAE računa kvalitetu preporuke za proizvod "item"

    function [rez] = MAE (A,item,kk,nn,brU)
    rez=0;
    vecSum=0;
    for ii=1:brU
            pred=svdBasedPrediction(A,ii,item,kk,nn);
            rr=A(ii,item);
            vecSum = abs(pred-rr);
    end
    rez=vecSum/brU;

% primjer računanja predikcije za korisnika K7 i film F1
% uz kpred=2 i nn=2
```

```
% Matrica A predstavlja matricu korisnik-film

    A=[
        3 0 5 0 3 4 0 0 0 0 0
        0 3 5 3 0 4 0 3 0 0 3
        4 0 5 3 3 4 4 0 4 0 0
        5 3 5 0 5 5 0 0 0 0 5
        4 4 5 5 4 5 0 3 0 3 3
        4 4 4 4 5 5 0 0 0 0 0
        0 5 0 0 4 0 0 3 0 0 0
        0 4 5 0 5 3 0 0 0 5 0
        5 4 0 5 3 4 5 0 5 4 0
        0 5 5 0 4 2 0 0 0 0 0
        3 0 5 4 5 0 4 5 2 0 5
     ];
    pr=svdBasedPrediction(A,7,1,2,2) % pr=4.0177
    pr=svdBasedPrediction(A,7,1,3,2) % pr=3.5215
    pr=svdBasedPrediction(A,7,1,2,2) % pr=3.7676
```

Prilog 3. U nastavku su navedene funkcije implementirane u Matlabu za Algoritam 2. Funkcije koje su identične funkcijama korištenim za Algoritam 1 navedene su u Prilogu 2.

```
% primjer dodavanja novog korisnika u sustav
% zadamo novi vektor korisnika "a" i novu dimenziju "k"
% zadamo broj susjeda, "neig"
    a=[3 0 0 4 5 0 4 5 2 0 5];
    k=3;
    neig=4;
% 1.korak
    [rCol rRow M N P]=SVDPreCalculation(A,k);
% 2.korak (predikcija za početnu matricu A)
    pr1=svdBasedPrediction(M,N,P,11,3,neig,rRow)
% 3.korak (dodavanje novog vektora)
    [rCol rRow M N P] = AddingNewVector(b,rCol,rRow,M,N,P);
% predikcija za novi vektor
    pr2 = svdBasedPrediction(M,N,P,12,3,neig,rRow)


% funkcija SVDPreCalculation priprema sustav za racunanje predikcije

    function [rCol, rRow, U, S, V]= SVDPreCalculation(A,kpred)
    [Afil rCol]=fillInMatrix(A);
    [NR rRow]=calculateRnorm(Afil,A)
    %kpred=min(kpred,rank(A));
    [U S V]=KdimReduction(NR,kpred)
```

```
% prije ove metode pozvati:
% 1) [rCol rRow M N P]=SVDPreCalculation(A,kpred);
% ukoliko zelimo dodati novog korisnika, potrebno je
% prije pokretanja funkcije za predikciju pozvati metodu 1)
% (samo jedanput),a zatim:
% [rCol rRow M N P] = AddingNewVector(a,rCol,rRow,M,N,P)

    function [pred] = svdBasedPrediction (M,N,P,user,item,nn,rRow)
    Rred=M*N*P;
    [US SV]=calcSqMatrices(M,N,P);
    SimMat=AdjustedCosSim(SV);
    a=neighborhood(SimMat,item,nn);
    kk=length(a);
    topSum=0;
    absSum =0;
    for iter=1:kk
       topSum=topSum+ SimMat(item,iter)*(Rred(user,iter)+rRow(user));
       absSum=absSum+abs(SimMat(item,iter));
    end
    pred= topSum/absSum;

% funkcija AddingNewVector dodaje novi vektor u postojeci sustav

    function [rCol rRow M N P] = AddingNewVector(a,rCol, rRow,M,N,P)
    [newa rCol rRow]=vcalculateVecNorm(a,rCol,rRow);
    M=UpdateU(M,N,P',newa);

% funkcija vcalculateVecNorm za zadani novi vektor "a" osvjezava
% vektore "rCol" i "rRow"  i normira vektor "a"

   function [a, rCol, rRow]= vcalculateVecNorm (a,rCol,rRow)
    temp = vectorMean(a)
    for iter=1:length(a)
        if(a(iter)==0)
            a(iter)=rCol(iter);
         else
             rCol=(rCol*length(rRow) + a(iter))/(length(rRow)+1)
        end
        a(iter)=a(iter)-temp;
    end
    rRow=[rRow;temp];

% funkcija UpdateU racunati novi "a" kao: anew=a*OldV*inv(OldS)
% i dodaje ga kao zadnji redak matrice "U"

    function [U] = UpdateU(U,OldS,OldV,a)
    a=a*OldV/OldS;
```

```
    U=[U;a];

% funkcija fillInMatrix popunjava matricu "A", odnosno elemente
% koji su jednaki 0 zamjenjuje sa prosjekom pripadnog stupca i racuna
% vektor rCol koji sadrzi prosjeke stupaca

    function [Rfil, rCol] = fillInMatrix (A)
    [nf mf]=size(A);
    Rfil = zeros(nf,mf);
    rCol = zeros(1,mf);
    for ii=1:mf
        temp=vectorMean(A(:,ii));
        rCol(ii)=temp;
        for jj=1:nf
            if A(jj,ii)==0
                Rfil(jj,ii)=temp;
            else
                Rfil(jj,ii)=A(jj,ii);
            end
        end
    end

% funkcija za ulaznu "F" i njenu popunjenu matrci "Ffill" racuna
% normaliziranu matricu "Rnorm" i vektor prosjeka redaka "rRow"

    function [Rnorm, rRow] = calculateRnorm (Ffill, F)
    [n m]=size(F);
    Rnorm =zeros(n,m);
    rRow = zeros(n,1);
    for ii=1:n
        rRow(ii) = vectorMean(F(ii,:));
        Rnorm(ii,:) = Ffill(ii,:)-(ones(1,m)*vectorMean(F(ii,:)));
    end
```

7 Literatura

Literatura

[1] Claudio Adrian Levinas, *An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals*, Master of Science Thesis (2014.)

[2] Badrul Sarwar, George Karypis, Joseph Konstan, John Ried, *Application of Dimensionality Reduction in Recommender System - A Case Study*, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA (2000.)

[3] James W. Demmel, *Applied Numerical Linear Algebra, University of California*, Berkeley, Clifornia (1997.)

[4] Manolis G. Vozalis, Konstantinos G. Margaritis, *Applying SVD on Generalized Item-based Filtering*, University of Macedonia, Thessaloniki, Greece (2006.)

[5] Taghi M. Khoshgoftaar, Xiaoyuan Su, *A Survey of Collaborative Filtering Techniques*, Department of Computer Science and Engineering, Florida Atlantic University, USA (2009.)

[6] Michael D. Ekstrand, Joseph A. Konstan, John T Riedl, *Collaborative Filtering Recommender Systems*,Department of Computer Science and Engineering, University of Minnesota, Minneapolis (2011.)

[7] Badrul Sarwar, George Karypis, Joseph Konstan, John Ried, *Incremental Singular Value Decomposition Algorithm for Highly Scalable Recommender Systems* , Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA (2002.)

[8] Badrul Sarwar, George Karypis, Joseph Konstan, John Ried, *Item-based Collaborative Filtering Recommendation Algorithms*, Department of Computer Science and Engineering, University of Minnesota, Minneapolis (2001.)

[9] Gene H. Golub, Charles F. Van Loan, *Matrix computations (Johns Hopkins Studies in Mathematical Sciences)*, The Johns Hopkins University Press (1996.)

[10] Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman, *Mining of Massive Datasets, Second Edition* (2014.)

[11] Ninoslav Truhar, *Numerička linearna algebra*, Odjelu za matematiku, Sveučilišta J.J. Strossmayera, Osijek (2010.)

[12] Alexander Paprotny, Michael Thess, *Realtime Data Mining, Self-Learning Techniques for Recommendation Engines*, Applied and Numerical Harmonic Analysis, Birkhäuser (2013.)

[13] Charu C. Aggarwal, *Recommender Systems:The Textbook*, Springer (2016.)

[14] Xiwei Wang, Jun Zhang, *SVD-based Privacy Preserving Data Updating in Collaborative Filtering*, Proceedings of the World Congress on Engineering, Vol I, London (2012.)

[15] Yehuda Koren, *The BellKor Solution to the Netflix Grand Prize* (August 2009.)

[16] Raymond J. Spiteri, Jane E. Tougas, *Updating the Partial Singular Value Decomposition in Latent Semantic Indexing*, NSERC Canada (2006.)

[17] M.W. Berry, S.T. Dumais, G.W. O'Brein, *Using Linear Algebra for Intelligent Information Retrieval*, Computer Science Department, University of Tennessee, Knoxville (1994.)

[18] Elaine Rich, *User Modeling via Stereotypes* (1979).

[19] Hrvoje Kraljević, *Vektorski prostori*, Odjelu za matematiku, Sveučilišta J.J. Strossmayera, Osijek (2008.)

[20] https://export-x.com/2015/12/11/how-many-products-does-amazon-sell-2015/ ExportX, 28.10.2016.

[21] http://hjp.znanje.hr, Hrvatski Jezični portal, 27.10.2016.

Sažetak:
U radu je opisan jedan tip sustava za preporuke, točnije kolaborativno filtriranje bazirano na modelu orijentiranom prema proizvodu i pripadni algoritam baziran na SVD dekompoziciji matrice sustava.
Ukratko je dan kratak povijesni pregled sustava za preporuke te su navedeni osnovni pojmovi i opisani su najbitniji tipovi sustava za preporuke. Zatim je iskazan centralni algoritam rada, algoritam za preporuke zasnovan na SVD dekompoziciji s redukcijom dimenzije te je obrazložena njegova korektnost. Kroz svojstva navedenog algoritma naglašena je korisnost SVD dekompozicije matrica u stvarnom svijetu. Nakon toga opisana su dva načina za osvježavanje sustava novim korisnicima, *folding-in* metoda i *update* metoda te su dani pripadni algoritmi.
U zadnjem dijelu rada opisani su testovi provedeni na implementiranim algoritmima, algoritmu za preporuke zasnovanom na SVD dekompoziciji s redukcijom dimenzije i *folding-in* metodi, a pripadni Matlab kodovi priloženi su u Prilogu 2 i 3.

Ključne riječi: sustavi za preporuke, kolaborativno filtriranje, sustavi bazirani na modelu, sustavi orjentirani prema proizvodu, predikcija, SVD, redukcija dimenzije, folding-in metoda

Abstract:
The thesis is dealing with a special type of recommender system, with the main focus on the item and model based collaborative filtering recommender system including corresponding SVD based algorithm.
The historical overview of recommender systems is briefly illustrated, key terms are listed and the most important types of recommender systems are described. Afterwards, the central algorithm of the thesis, the SVD algorithm for recommender system with dimensionality reduction, has been stated and its utility is motivated. The benefit of SVD decomposition in the real world has been outlined while describing properties of the central algorithm. After that, the folding-in method and the update method for updating system with new users have been described and related algorithms are given.
The last section of the thesis displays the test examples of implemented algorithms, the central algorithm of the thesis and the folding-in method. Code implementation using Matlab is appended at the end of the thesis in Appendix 2 and Appendix 3.

Key words: recommender systems, collaborative filtering, model based RS, item based RS, prediction, SVD, dimension reduction, folding-in method

Životopis:

Rođena sam 17. siječnja 1992. godine u Našicama. Osnovnu školu pohađala sam u Osnovnoj školi Ivane Brlić Mažuranić u Orahovici te sam osnovnoškolsko obrazovanje završila 2006.godine. Iste godine upisala sam opću gimnaziju u Srednjoj školi Stjepana Ivšića, također u Orahovici gdje 2010. godine završavam srednjoškolsko obrazovanje. Tijekom osnovne i srednje škole sudjelovala sam na općinskim i županijskim natjecanjima iz više nastavnih predmeta uključujući i matematiku. Preddiplomski studij matematike na Odjelu za matematiku Sveučilišta J.J. Strossmayera u Osijeku upisala sam 2011. godine te sam ga uspješno završile 2014. godine. Daljenje obrazovanje nastavila sam na Odjelu za matematiku gdje sam iste, 2014. godine, upisala diplomski studija matematike, smjer Matematika i računarstvo. Tijekom prve godine diplomskog studija odradila sam jednomjesečnu stručnu praksu u osječkoj tvrtki Inchoo d.o.o, a tijekom druge godine diplomskog studija obavljala sam višemjesečnu praksu u tvrtki Span d.o.o.