math.e

Hrvatski matematički elektronički časopis

Kolaborativno filtriranje

kolaborativno filtriranje predikcija redukcija dimenzije sustavi za preporuke SVD dekompozicija

Mirna Marković i Zoran Tomljanović

Sažetak

U ovom članku opisan je 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 naglašene su neke korisnosti SVD dekompozicije matrice koja ima mnoge primjene 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.

Riječani 73
33515, Orahovica
m.mirna08@gmail.com (M. Marković)
Odjel za matematiku
Sveučilište Josipa Jurja Strossmayera u Osijeku
Trg Lj. Gaja 6
31000, Osijek
ztomljan@mathos.hr (Z. Tomljanović)

1 Uvod

Preporuka je, prema definiciji iz [17], 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*, u nastavku CF sustavi) 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.

Povijesno gledano, možemo reći da je začetnik automatiziranih sustava za preporuke *Grundy*, računalni program za knjižnicu (1979. godine) (za više informacija vidjeti [15]). *Xerox PARC* razvio je 1992. godine *Tapestry*, platformu koja je korisnicima omogućila označavanje emailova i pretraživanje istih na temelju tih oznaka. Najpoznatija primjena sustava za preporuke je ona na domeni *Amazon.com*[2]. Sustavi za preporuku najveću pozornost privukli su 2006. godine kada je *Netflix*[3] objavio *Netflix Prize*, nagradu od milijun dolara za kreiranje algoritma koji bi za 10% ubrzao njihov tadašnji algoritam.[4] 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 [12]. Danas su poznate različite vrste sustava za preporuke i njihovi tipovi, a neke od njih predstavit ćemo u ovom radu.

Ovaj rad baziran je na diplomskom radu [5], gdje se može pronaći detaljnija implementacija nekih algoritama koji su opisani u nastavku. U ovom radu opisani su specijalno sustavi za preporuke bazirani na kolaborativnom filtriranju, odnosno jedan tip takvih sustava. U

drugom poglavlju su navedene opće karakteristike i podjela sustava za preporuke. U trećem poglavlju opisan je jedan tip sustava za preporuke, točnije kolaborativno filtriranje bazirano na modelu orijentiranom prema proizvodu (engl. *item model based collaborative filtering RS*) i njegov algoritam temeljen na faktorizaciji matrice i redukciji dimenzije. Dodavanje novog korisnika u sustav i kreiranje preporuka za njega opisano je također u trećem poglavlju, a u četvrtom poglavlju su testirani opisani algoritmi koji su implementirani u programskom paketu Matlab.

2 Sustavi za preporuke

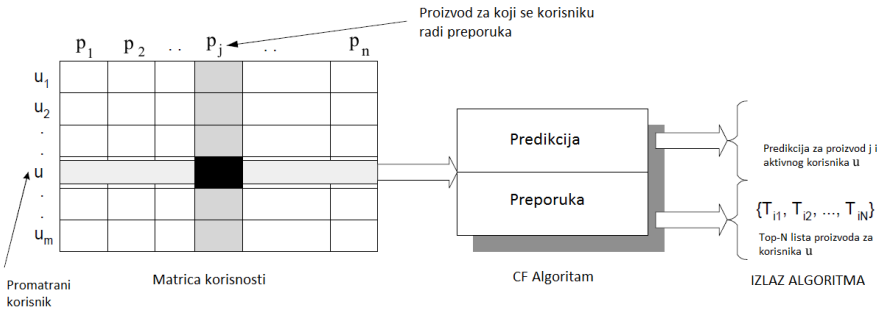
Sustavi za preporuke razvijeni su kako bi se što bolje suočilo s eksponencijalnim rastom informacija, u smislu filtriranja podataka. Oni uče iz akcija korisnika o njihovom ukusu kako bi im u skladu s tim predložili nove proizvode, stoga ih istovremeno možemo promatrati kao dio rudarenja podataka i strojnog 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 se 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 preferencije 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.

	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

Svaki element matrice korisnosti predstavlja jedan par korisnik-proizvod 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[5], dok je unarna skala najbolja 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 1.



Slika 1: *Ilustracija algoritma za preporuku*

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 [9], bitno je pronaći takav podskup nepoznatih vrijednosti za svaki redak za koje predviđamo da će njihove vrijednosti biti među najvećima.

2.1 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 i ovdje se koristi kako bi sustav mogao zaključiti o ukusu pojedinog korisnika iz njegovih akcija. U takvom sustavu svaki korisnik se 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.

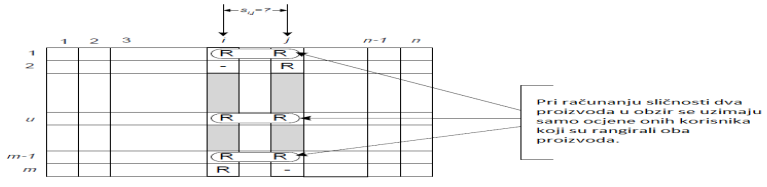
Dobar primjer domene koja koristi kontekstno bazirani sustav jest *pandora.com*. Ovakvi sustavi za preporuke korisniku uzastopno preporučuju proizvode koji su slični po svojstvima, što je ujedno njihova prednost i mana (engl. *over-specialized search*)[6]. Korisnik mora na neki način sam pokazati interes 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 preporuke se generiraju na temelju korisnikova rangiranja proizvoda, a ne na temelju svojstava proizvoda. Kako je navedeno u [1], osnovni 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.

2.2 CF sustavi orijentirani prema korisniku i orijentirani prema proizvodu

Glavna pretpostavka na kojoj se temelje svi 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. Sličnost korisnika ili proizvoda računa se unaprijed definiranom funkcijom koju nazivamo funkcija sličnosti (engl. *similarity function*), a skup najsličnijih korisnika ili proizvoda u odnosu na promatranog korisnika ili proizvod nazivamo susjedstvo (engl. *neighbourhood*). Zadaća funkcije sličnosti je da numerički predstavi udaljenost između dva korisnika ili proizvoda kako bi se moglo definirati njihovo susjedstvo, odnosno najbliži korisnici ili proizvodi. Neke od najčešće korištenih funkcija sličnosti su euklidska udaljenost, Pearsonova korelacija i kosinus udaljenost. CF sustavi 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*).

Algoritmi CF sustava orijentirani prema korisniku vode se pretpostavkom da *"slični korisnici vole slične proizvode"*, dok se oni orijentirani prema proizvodu vode se pretpostavkom da *"slične proizvode vole slični korisnici"*, kao na Slici 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[7]. 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 preporučen jer se svidio sličnom korisniku.



Slika 2: *Grafički prikaz funkcije sličnosti*

2.3 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 (engl. *model based*) i memorijski bazirane (engl. *memory based*).

Memorijski baziranima CF sustavima 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. Takvi algoritmi podatke o korisnicima i proizvodima drže u memoriji i direktno računaju preporuke, odnosno sličnost među susjedstvom računa se svaki put kada se korisniku želi dati preporuka, još se kaže da je to *on-line* generiranje preporuke.

CF sustavi bazirani na modelu generiraju model sustava i na temelju njega generiraju preporuku. Kod pristupa baziranom na modelu kažemo da se model kreira *off-line* odnosno unaprijed, 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.

2.4 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*[8] i slično.

U stvarnim primjerima i primjenama sustavi za preporuke se moraju nositi s različitim problemima, a istaknuli bismo neke od njih: slaba popunjenost podataka, sinonimi, gray sheep i shilling attacks. Više detalja vezano za spomenute probleme može se pronaći u [5, 4].

U okviru ovog rada istaknuli bismo da se slaba popunjenost podataka pojavljuje jer najčešće matrice sustava 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.

Nadalje, 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.

U sljedećem poglavlju opisat ćemo algoritme za CF sustave bazirane na modelu orijentiranom prema proizvodu koji koriste dekompoziciju na singularne vrijednosti (SVD).

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

3.1 Dekompozicija na singularne vrijednosti

Prije nego što uvedemo dekompoziciju na singularne vrijednosti, prisjetimo se da za matricu $Q \in \mathbb{R}^{n\times n}$ kažemo da je ortogonalna ukoliko je $Q^TQ = I$. Analogno, za $U \in \mathbb{C}^{n\times n}$ kažemo da je unitarna ukoliko je $U^*U = I$.

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 $U \in \mathbb{R}^{m\times m}$, $S \in \mathbb{R}^{m\times n}$ i $V \in \mathbb{R}^{n\times n}$ čija su svojstva iskazana Teoremom 1.

U ovom poglavlju iskazat ćemo neke od osnovnih svojstava singularne dekompozicije matrice, a više detalja može se pronaći u [2, 8, 6].

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*

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

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

Stupce matrice U zovemo lijevi, a stupce matrice V desni singularni vektori matrice A, dok dijagonalne elemente matrice S nazivamo singularnim vrijednostima.

Poznato je da koristeći SVD dekompoziciju možemo efikasno odrediti rang matrice kao i prostor koji razapinje sliku, odnosno jezgru matrice. Naime, prema [8] vrijedi sljedeći teorem.

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. $\text{rang}(A) = r$,

(2) $\mathcal{R}(A) = \text{span}(u_1, \ldots, u_r)$, odnosno, stupci matrice U, $u_1, \ldots, u_r$ razapinju sliku[9] od A

(3) $\mathcal{N}(A) = \text{span}(v_{r+1}, \ldots, v_n)$, odnosno, stupci matrice V, $v_{r+1}, \ldots, v_n$ razapinju jezgru[10] od A

Prema [14, 2, 6] 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, pri čemu se pogreška aproksimacije mjeri koristeći spektralnu normu [2].

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 $r = \operatorname{rang}(A) \leq p = \min(m,n)$ i definiramo*

$$A_k = \sum_{i=1}^{k} u_i \sigma_i v_i^T, \tag{2}$$

gdje je $k < r$. Tada je

$$\min_{\substack{B\in\mathbb{R}^{m\times n}\\ \operatorname{rang}(B)\leq k}} \|A-B\|_2^2 = \|A-A_k\|_2^2 = \sigma_{k+1}^2. \tag{3}$$

Matrica A_k dobivena je korištenjem trojki $(u_i, \sigma_i, v_i^T), i = 1, \ldots, k$ koje pripadaju najvećim singularnim vrijednostima. Ona daje aproksimaciju matrice A matricom ranga k. Štoviše, to je najbolja aproksimacija nižeg ranga u skupu matrica ranga manjeg ili jednakog k, za bilo koju unitarnu invarijantnu normu[11]. Specijalno, ako pogrešku mjerimo u Frobeniusovoj normi ($\|\cdot\|_F$), vrijedi da je:

$$\min_{\substack{B\in\mathbb{R}^{m\times n}\\ \operatorname{rang}(B)=k}} \|A-B\|_F^2 = \|A-A_k\|_F^2 = \sigma_{k+1}^2 + \ldots + \sigma_p^2.$$

Ovaj teorem je od velike važnosti za CF sustave jer omogućuje znatno smanjenje količine podataka koje CF sustav treba obraditi. Umjesto s velikom matricom A reda $m \times n$ sustav može raditi s matricom manjeg ranga A_k koja se u faktoriziranom obliku može efikasno zapisati sve dok je $k \ll r$. Točnije, ovaj teorem daje najbolji mogući način kako se matrica može aproksimirati matricom ranga koji je najviše k.

Matrica S tada se očito aproksimira matricom S_k tako da se na dijagonali zadrži najvećih k singularnih vrijednosti, dok se matrice U i V aproksimiraju matricama U_k i V_k. Matrica U_k nastaje zadržavanjem prvih k stupaca iz matrice U, dok zadržavanjem prvih k stupaca matrice V nastaje matrica V_k i tada je $A_k = U_k S_k V_k^T$.

Napomena. U praksi se često javljaju matrice koje se mogu efikasno aproksimirati matricom nižeg ranga. Naime, zbog velikog broja podataka podaci iz primjene često sadrže veliki broj redundantnosti, što znači da će matrice kojima se modeliraju takvi problemi imati kolinearne stupce, odnosno retke. Jedan takav primjer pojavljuje se kod kompresije slike pri čemu se slika zapisuje koristeći matrični zapis. Naime, u tom slučaju su stupci "bliski" kolinearnosti (jer susjedni stupci sadrže bliske elemente) pa je matrica koja modelira te podatke bliska matrici malog ranga. Stoga se efikasno može primijeniti Eckart–Young–Mirskyjev teorem dan u Teoremu 3 te se prilikom aproksimacije i za vrlo male dimenzije k ne gubi značajno na kvaliteti slike. Više detalja vezano za aproksimaciju, ali i spomenuti primjer s kompresijom slike može se pronaći u [2].

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 [4] osnovni tipovi su: *metrike obzirom na predikciju* (engl. *predictive accurancy metrics*), *metrike obzirom na rang* (engl. *rank accurancy metrics*) i *normalizirana metrika obzirom na udaljenost* (engl. *normalized distance-based performance metrics*).

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.

Računamo li točnost preporuke za proizvod j, MAE možemo računati po slijedećoj formuli:

$$MAE_j = \frac{\sum_{i=1}^{m} |prediction_{ij} - r_{ij}|}{m}, \text{ gdje je } j = 1, \ldots, n \quad (4)$$

Pri tome je m je broj korisnika, sa R označavamo matricu korisnosti sustava, a $prediction_{ij}$ je procjena za element r_{ij} matrice sustava R, odnosno za procjena za i-tog korisnika i j-ti proizvod. 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 preferencijama. 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 baziran na SVD dekompoziciji s redukcijom dimenzije

Standardni algoritmi 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 [3], 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 p_j označava j-ti proizvod u sustavu tako da je $i \in \{1, \ldots, m\}$ i $j \in \{1, \ldots, n\}$.
- $R \in \mathbb{R}^{m \times n}$ označava matricu m korisnika i n proizvoda sa elementima r_{ij} koji predstavljaju numeričku oznaku koliko se korisniku u_i sviđa proizvod p_j. Ukoliko korisnik nije rangirao neki proizvod, tada se na to mjesto u matrici R dogovorno stavlja 0.

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

Algoritam 1 (prema [3])

(1) Konstruirati matricu korisnosti sustava, odnosno matricu $R \in \mathbb{R}^{m \times n}$ koja predstavlja m korisnika i n proizvoda čije elemente označavamo s r_{ij} koji predstavljaju numeričku oznaku koliko se korisniku u_i sviđa proizvod p_j.

(2) Eliminirati nepoznata polja, točnije vrijednosti koje su jednake nuli, u matrici R na sljedeći način:

- Izračunati prosjeke redaka koji označavaju korisnike, r_i za

$i = 1, 2, \ldots, m$, koje ćemo označavati s $\overline{r_i}$ i prosjeke svih stupaca (proizvoda) s_j za $j = 1, 2, \ldots, n$, koje ćemo označavati sa $\overline{s_j}$ matrice R tako da je $r_{ij} \neq 0$.

$$\overline{r_i} = \frac{\sum_{j=1}^{n} r_{ij}}{\sum_{\substack{j=1 \\ r_{ij} \neq 0}}^{n} 1}, i = 1, 2, \ldots m,$$

te ako je $r_{ij} = 0, \forall j = 1, 2, \ldots, n$, onda je $\overline{r_i} = 0$,

$$\overline{s_j} = \frac{\sum_{1=1}^{m} r_{ij}}{\sum_{\substack{i=1 \\ r_{ij} \neq 0}}^{m} 1}, j = 1, 2, \ldots n,$$

te ako je $r_{ij} = 0, \forall i = 1, 2, \ldots, m$, onda je $\overline{s_j} = 0$.

- Nepoznate vrijednosti u svakom stupcu s_j (one koje su jednake 0) zamijeniti sa prethodno izračunatim prosjekom toga stupca, $\overline{s_j}$. Rezultat ovog koraka je popunjena matrica koju ćemo označiti sa $R_{filled-in}$.
  Za $R = (r_{ij})$ je $(R_{filled-in})_{ij} = r_{ij}$ za $r_{ij} \neq 0$ i $(R_{filled-in})_{ij} = \overline{s_j}$ za $r_{ij} = 0$.
- Svakom elementu matrice $R_{filled-in}$ oduzeti prosjek $\overline{r_i}$ pripadnog retka r_i matrice R i rezultat ovog koraka je normalizirana matrica koju ćemo označiti s R_{norm}, odnosno $(R_{norm})_{ij} = r_{filled-in_{ij}} - \overline{r_i}$ za $i = 1, 2, \ldots, m$ i $j = 1, 2, \ldots, n$.

(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 matricu R_{norm} tako da se zadrži samo k najvećih singularnih vrijednosti, $k \ll \operatorname{rang}(R_{norm})$ te je tada aproksimacija matrice R_{norm} dana matricom $U_k S_k V_k^T$ sa elementima $\hat{r}_{ij}$, gdje su $U_k \in \mathbb{R}^{m \times k}$, $S_k \in \mathbb{R}^{k \times k}$ i $V_k \in \mathbb{R}^{n \times k}$

(5) Izračunati matrični korijen od S_k, $\sqrt{S_k}$ te $U_k \sqrt{S_k}^T \in \mathbb{R}^{m \times k}$ koja određuje podatke od m korisnika i $\sqrt{S_k} V_k^T \in \mathbb{R}^{k \times n}$ koja određuje podatke od n proizvoda u k-dimenzionalnom prostoru. Matrica $W := \sqrt{S_k} V_k^T$ dimenzije $k \times n$, k pseudo korisnika[12] i n proizvoda je od velikog značaja jer njezini elementi w_{ij} predstavljaju numeričku oznaku koliko se pseudo korisniku u_i sviđa proizvod p_j

(6) Za promatrani proizvod p_j kreiramo njegovo susjedstvo na sljedeći način:

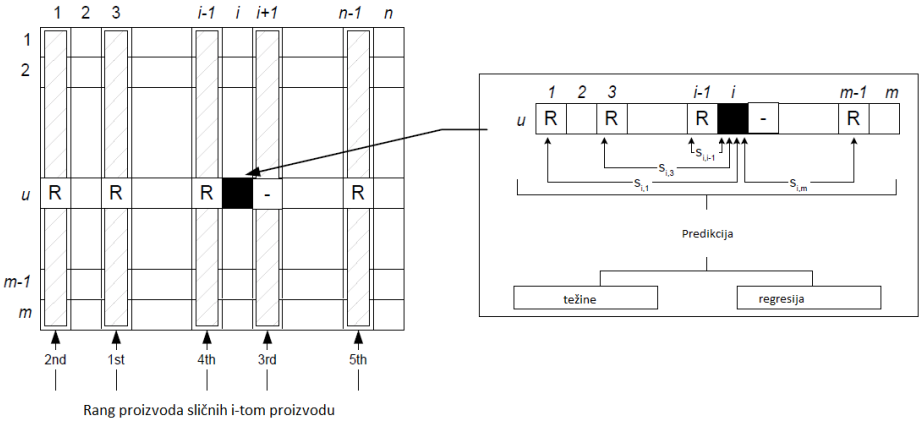
- Izračunati sličnost između proizvoda p_j i p_f pomoću funkcije sličnosti, prilagođene kosinus sličnosti[13] (engl. *adjusted cosine similarity*) za $f = 1, 2, \ldots, n$:

$$sim_{jf} = adjcosr_{jf} = \frac{\sum_{i=1}^{k} w_{ij} \cdot w_{if}}{\sqrt{\sum_{i=1}^{k} w_{ij}^2 \sum_{i=1}^{k} w_{if}^2}}. \quad (5)$$

- Obzirom na rezultate prilagođene kosinus sličnosti sim_{jf} za $f = 1, 2, \ldots, n$ odabrati l proizvoda najsličnijih promatranom proizvodu p_j, odnosno njih l za koje je pripadna vrijednost sim_{jf} najveća. Dobivenih l proizvoda nazivamo l-susjedstvo promatranog proizvoda p_j, a dani postupak nazivamo kreiranje l-susjedstva za dani proizvod (Slika 3).

(7) Izračunati predikciju za korisnika u_i i proizvod p_j prema sljedećoj formuli:

$$prediction_{ij} = \frac{\sum_{k=1}^{l} sim_{jk} \cdot (\hat{r}_{ik} + \overline{r_i})}{\sum_{k=1}^{l} |sim_{jk}|}. \qquad (6)$$



Slika 3: *Ilustracija izračunavanja* 5-susjedstva za proizvod i

Uočimo da se u 1. i 2. koraku algoritma matrica sustava 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 2 i Teorema 3, koristeći SVD dekompoziciju matrice, u prethodnim koracima smo dobili dobru aproksimaciju početne matrice. 6. korak algoritma koristeći prilagođenu kosinus 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_i i proizvod p_j računa uzimajući u obzir proizvode iz skupa l-susjeda za proizvod p_j, a svakom pripadnom pseudo korisniku $\hat{r}_{ik}$, za $k = 1, \ldots, l$ mora se dodati prosjek retka za pripadnog korisnika, tj. $\overline{r_i}$, koji je bio oduzet u 2. koraku pri normalizaciji.

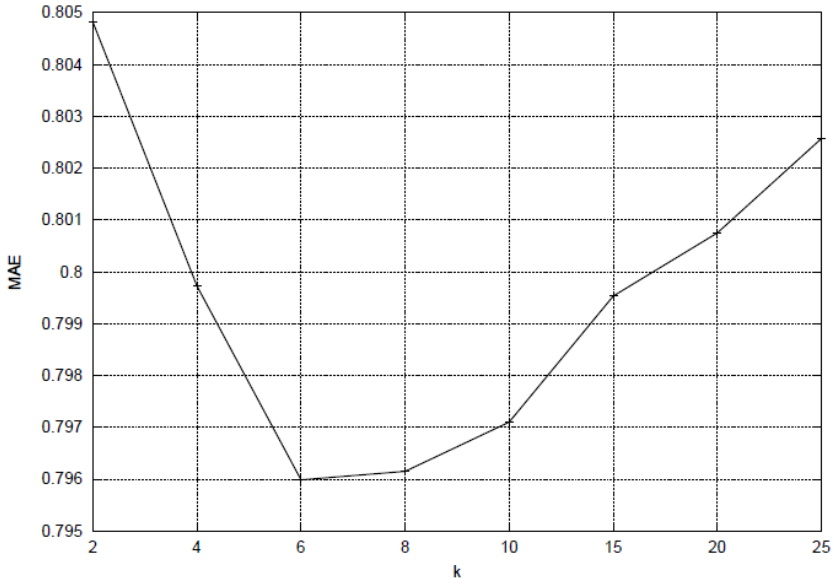
3.3.1 Svojstva algoritma

Prema [3] složenost standardnog algoritma bez koraka redukcije je $\mathcal{O}(mn^2)$, a s redukcijom dimenzije na rang $k \ll m$, 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.

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.

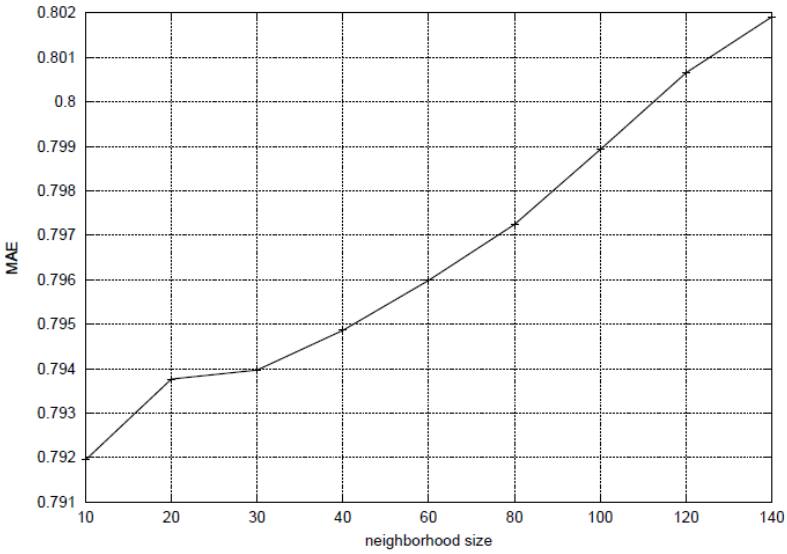
Kako je prethodno navedeno, parametar k određuje broj singularnih vrijednosti matrice S koje će se zadržati u matrici S_k, odnosno određuje rang matrice R_k i broj pseudo korisnika u reduciranom sustavu. Parametar k bira se tako da bude što manji kako bi utjecao na poboljšanje brzine i efikasnosti te ujedno mora biti takav da reducirani sustav zadrži što više svojstava početnog sustava, ali da ne bude preblizak početnom sustavu (engl. *overfitting*). Fiksiranjem parametra l može se eksperimentom za različite k odrediti onaj koji prema MAE daje najbolju preporuku za dani sustav. Slika 4 prikazuje rezultate eksperimenta koji je u [3] 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 $k = 6$. Svakako, odabir najboljeg parametra k ovisi o veličini danog sustava i popunjenosti matrice korisnosti. Što je k veći, podatci u reduciranoj matrici sustava biti će bliži podatcima početnog sustava, odnosno biti će raspršeniji pa će i preporuka biti lošija.



Slika 4: *Eksperiment za različite vrijednosti parametra* k. Prikaz prosječne apsolutne pogreške (MAE) za sustavu od 943 korisnika i 1682 proizvoda za različite vrijednosti parametra k, $k = 2, 4, 6, 8, 10, 15, 20, 25$ i fiksan parametar $l = 60$.Najbolja kvaliteta postiže se za $k = 6$.

Veličina susjedstva, l, koje se promatra za određeni proizvod također ima utjecaj na kvalitetu 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 5, a to znači da kvaliteta predikcije opada za veliko susjedstvo. Eksperiment sa slike proveden je u [3] 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. Očito je da veliko susjedstvo negativno utječe na kvalitetu preporuke jer su podatci tada raspršeniji, također optimalne vrijednosti za oba parametra, k i l, ovisit će o skupu podataka nad kojima se određuju.



Slika 5: *Eksperiment za različite vrijednosti parametra* l. Prikaz prosječne apsolutne pogreške (MAE) za sustav od 943 korisnika i 1682 proizvoda za različite vrijednosti parametra l, $l = \{10, 20, 30, 40, 60, 80, 100, 120, 140\}$ i fiksan parametar $k = 6$. Najbolja kvaliteta postiže se za $l = 10$.

Nedostatak CF algoritama baziranih na SVD dekompoziciji jest da se vrijeme računanja 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 sa osvježavanjem sustava bez ponovnog računanja SVD dekompozicije

3.4.1 Osvježavanje sustava metodom *folding-in*

Glavni problem prethodnog algoritma jest kako dodati korisnika u sustav bez ponovnog računanja modela cijelog sustava, odnosno bez ponovnog računanja SVD dekompozicije cijele matrice korisnosti. U matričnom jeziku, za dani novi vektor r_{new} i već poznatu matricu R potrebno je napraviti SVD dekompoziciju nove matrice $\begin{bmatrix} R \\ r_{new} \end{bmatrix}$ koristeći već postojeću SVD dekompoziciju matrice R. U [13] je dan postupak kako je moguće u sustav dodati novi vektor korisnika $r_{new} \in \mathbb{R}^{1\times n}$ bez ponovnog računanja SVD dekompozicije matrice sustava.

Algoritam 2

(1) Novi korisnik možda nije ocijenio sve proizvode te je odgovarajuća, nepopunjena mjesta vektora r_{new} prvo potrebno popuniti i to prosjecima pripadnih stupaca matrice R koji su ažurirani. Srednje vrijednosti stupaca dobivenih u 2. koraku Algoritma 1 potrebno je ažurirati, a to se radi za svaki $j = 1, 2, \ldots, n$ po sljedećoj formuli:

$$\overline{s'_j} = \frac{m \cdot \overline{s_j} + \sum_{i=m+1, r_{ij}\neq 0}^{(m+p)} r_{ij}}{m + \sum_{i=m+1, r_{ij}\neq 0}^{m+p} 1} \qquad (7)$$

gdje je $\overline{s'_j}$ prosjek j-tog stupca matrice sustava nakon dodavanja novih p korisnika u sustava, p je broj novih redaka (korisnika) koji se dodaju u sustav, a $\overline{s_j}$ je prosjek j-tog stupca početne matrice sustava R. Time smo ažurirali matricu $R_{filled-in}$.
Još je potrebno elemente u svakom novom retku normalizirati s prosjekom tog retka. Za svaki $i = (m+1), \ldots (m+p)$ izračunati $\overline{r'_i}$ kao u Algoritmu 1 te ažurirati matricu R_{norma} tako da je $(R_{norma})_{ij} = (R_{filled-in})_{ij} - \overline{r'_i}$ za $i = (m+1), \ldots (m+p)$ i $j = 1, 2, \ldots, n$.

(2) Popunjen i normaliziran 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) Aproksimaciju 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 $\hat{U_k} = \begin{bmatrix} U_k \\ r_{new_k} \end{bmatrix} \in \mathbb{R}^{(m+1)\times n}$ dok matrice V_k i S_k ostaju nepromijenjene.

Nakon *folding-in* metode ponovno se izvršavaju koraci 5., 6. i 7. iz Algoritma 1, samo umjesto matrice U_k koristimo matricu $\hat{U_k}$.

Postupak je moguće ponavljati više puta pri dodavanju jednog ili više korisnika odjednom. 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. Algoritam 2 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.

3.4.2 Osvježavanje sustava metodom *update*

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 poznajemo sve $\overline{s_j}$ prosjeke stupaca matrice R. Zadaća metode *update* je osvježiti matrice U_k, S_k, V_k novim podatcima i prosjeke svih stupaca $\overline{s_j}$ te dodati nove podatke u sustav. Prema [9],[13] i [11] algoritam za metodu *update* možemo iskazati na sljedeći način:

Algoritam 3

(1) Nove podatke potrebno je dopuniti i normalizirati. Postupak je identičan koraku 1. iz Algoritma 2.

(2) Izračunati projekciju r_{new_k} popunjenog i normaliziranog vektor $r_{new} r_{new_k} = (I_n - V_k V_k^T) r_{new}^T$ te SVD dekompoziciju matrice $\hat{S_k}$ dane sa:

$$\hat{S_k} = \begin{bmatrix} S_k & 0 \\ r_{new} V_k & \|r_{new_k}\| \end{bmatrix} = U'_k S'_k V'^T_k$$

(3) Izračunati $\hat{U}_k = \begin{bmatrix} U_k & 0 \\ 0 & 1 \end{bmatrix} U_k'$

(4) Izračunati $\hat{V}_k = \begin{bmatrix} V_k & \frac{r_{new_k}}{\|r_{new_k}\|} \end{bmatrix} V_k'$

Nakon postupka *update* metode ponovno se izvršavaju koraci 5., 6. i 7. iz Algoritma 1, ali koristeći nove osvježene matrice $\hat{S}_k$, $\hat{U}_k$ i $\hat{V}_k$ i s osvježenom matricom sustava.

Napomena. Algoritam 3 nalaže računanje SVD dekompozicije matrice $\hat{S}_k$ 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.

Prema [11], ukoliko se u jednoj iteraciji algoritma želi u sustav dodati više od jednog korisnika, točnije p korisnika, možemo to učiniti na sljedeći način.

Algoritam 4

(1) Pretpostavimo da u sustav dodajemo p novih korisnika te u tom slučaju matricu sustava treba proširiti matricom $T \in \mathbb{R}^{p \times n}$ koja predstavlja matricu korisnosti za p novih korisnika i n proizvoda. Nove podatke potrebno je dopuniti i normalizirati kao što je opisano u 1. koraku Algoritma 2, a rezultat je normalizirana matrica sustava $R_{norm} = \begin{bmatrix} R_{norm} \\ T_{norm} \end{bmatrix} \in \mathbb{R}^{(m+p) \times n}$. Uočimo da $T_{norm} \in \mathbb{R}^{p \times n}$ popunjena i normalizirana matrica novih p korisnika.

(2) Za matricu T_{norm} izračunati $T_k = (I_n - V_k V_k^T) T_{norm}^T$ i QR faktorizaciju[14] te matrice $T_k = Q_T S_T$.

(3) Izračunati SVD dekompoziciju matrice

$$\hat{S}_k = \begin{bmatrix} S_k & 0 \\ T_{norm} V_k & S_T^T \end{bmatrix} = U_k' S_k' V_k'^T$$

(4) Izračunati $\hat{U}_k = \begin{bmatrix} U_k & 0 \\ 0 & I_p \end{bmatrix} U_k'$

(5) Izračunati $\hat{V}_k = \begin{bmatrix} V_k & Q_T \end{bmatrix} V_k'$

Također, kao kod Algoritma 3, nakon što se provede postupka *update* metode ponovno se izvršavaju koraci 5., 6. i 7. iz Algoritma 1, ali koristeći nove osvježene matrice $\hat{S}_k$, $\hat{U}_k$ i $\hat{V}_k$ i s osvježenom matricom sustava.

Lako se može zaključiti da i ova metoda osvježavanja ima nedostatke, pogotovo ukoliko se odjednom u sustav dodaje mnogo korisnika. Prema [13] 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 hibridna metoda *folding-up* koja je kombinacija metoda *folding-in* i *update*. Ideja metode je da u početku koristi metodu *folding-in* do unaprijed određenog udjela novih korisnika u sustavu. Nakon toga da se pokreće metoda *update* nad novim korisnicima kako bi se sustav vratio u ravnotežu i osvježio prostor proizvoda, taj proces se iterativno nastavlja kombiniranjem te dvije metode.

4 Testiranje opisanih algoritama

4.1 Podatci za testiranje

Osim teorijskih iskaza SVD algoritama za CF sustave, u ovome radu provedeni su eksperimenti za dane algoritme. Uz pomoć programskog paketa *Matlab*, implementirani je algoritam zasnovan na SVD dekompoziciji s redukcijom dimenzije te algoritmi za osvježavanje sustava *folding-in* i *update* metodom.

Prije same implementacije, potrebno je prikupiti i pripremiti podatke za testiranje. Testiranje algoritama provedeno je na manjem primjeru, odnosno na bazi od 11 korisnika i 11 proizvoda, a podatci za matricu korisnosti R, prikazanu Tablicom 2, prikupljeni su pomoću *Google ankete*[15]. Ispitani korisnici 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.

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_i i film p_j i svojstva predikcije za slične proizvode i korisnike te je na kraju algoritam testiran za različite parametre k i l kako bi se odredili najbolji.

Kako je navedeno u algoritmu, prvi korak kod generiranja preporuke zasnovane na SVD dekompoziciji s redukcijom dimenzije jest popunjavanje i normalizacija matrice korisnosti. Prije svega normalizirali smo početnu matricu sustava R i dobivena matrica R_{norma} dana je sa:

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

Prema algoritmu sljedeći korak je SVD dekompozicija matrice R_{norm} i redukciju dimenzije na prethodno zadanu dimenziju obzirom na parametar k. Koristeći SVD dekompoziciju i Teorem 3 izračunali smo k-dimenzionalnu aproksimaciju matrice R_{norm} i rezultat je matrica $R_k = U_k S_k V_k^T$. Za $k = 4$ i prethodno dobivenu matricu R_{norm}, matrica R_k je dana s:

$$R_k = U_k S_k V_k^T = \begin{bmatrix} -0.2851 & -0.1262 & -0.1178 & 0.4282 \\ -0.3844 & -0.1205 & 0.1729 & -0.0128 \\ -0.2486 & -0.1736 & -0.0588 & 0.0629 \\ -0.3128 & 0.2237 & 0.5132 & -0.4167 \\ -0.3501 & -0.2353 & 0.3780 & 0.3606 \\ -0.1226 & 0.1704 & 0.3441 & -0.0788 \\ -0.3328 & -0.1106 & -0.1221 & 0.1566 \\ -0.3713 & 0.2016 & -0.2069 & -0.4905 \\ -0.1765 & -0.5131 & 0.0201 & -0.1395 \\ -0.3548 & -0.0353 & -0.5996 & -0.2072 \\ -0.2607 & 0.6983 & -0.1022 & 0.4185 \end{bmatrix} \begin{bmatrix} 3.8664 & 0 & 0 & 0 \\ 0 & 3.6269 & 0 & 0 \\ 0 & 0 & 3.2647 & 0 \\ 0 & 0 & 0 & 2.4602 \end{bmatrix} \times \tag{8}$$

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

i to je model testnog sustava.

Primjer 4.

a)

Promatra li se sličnost proizvoda za svaki par proizvoda u danom sustavu možemo konstruirati matricu $(sim_{jf}) \in \mathbb{R}^{n \times n}$ koju nazivamo matrica sličnosti proizvoda, a elementi matrice dobiju se iz formule (5) za svaki $j, f = 1, \ldots, n$. Općenito, matrica sličnosti proizvoda simetrična je matrica s jedinicama na dijagonali. Iz takve matrice danog sustava uočavamo da su najsličniji proizvodi, odnosno filmovi, redom:

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

što možemo potvrditi promatrajući Tablicu 2 iz koje se vidi da su baš filmovi $F3$ i $F7$, $F1$ i $F9$, $F8$ i $F11$ najsličniji.

b)

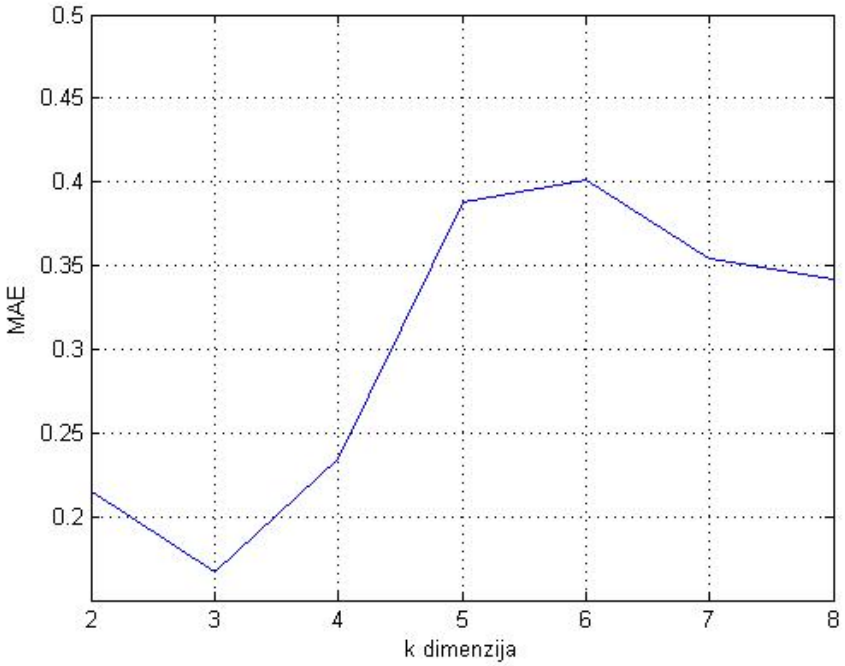
Nakon što se izračuna sličnost svih filmova, može se odrediti l-susjedstvo za svaki film. Točnije, za svaki pojedini film odabrati njemu l najsličnijih filmova iz matrice sličnosti i kažemo da oni tada čine l-susjedstvo tog filma. Primjerice, za fiksni $k = 4$ dobije se da 3-susjedstvo filma $F2$ čine filmovi $F5$, $F1$ i $F6$ te se film $F2$ nalazi u 3-susjedstvu filmova $F1$ i $F6$. Navedeno se može potvrditi i pažljivijim promatranjem podataka iz Tablice 2.

c)

Napokon je moguće izračunati predikciju za određenog korisnika u_i i filma p_j. U ovom slučaju promatramo korisnika $K7$ i film $F1$, The Martian(2015). Predikcija je izračunata koristeći parametre $k = 2$ i $l = 2$ prema formuli (6) za rezultat daje $prediction_{7,1} = 4.0177$. Točnije, algoritam predviđa da će korisnik $K7$ rangirati sa 4 film The Martian(2015). Ukoliko se susjedstvo poveća 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 što potvrđuje da preporuka uistinu ovisi o parametrima l i k.

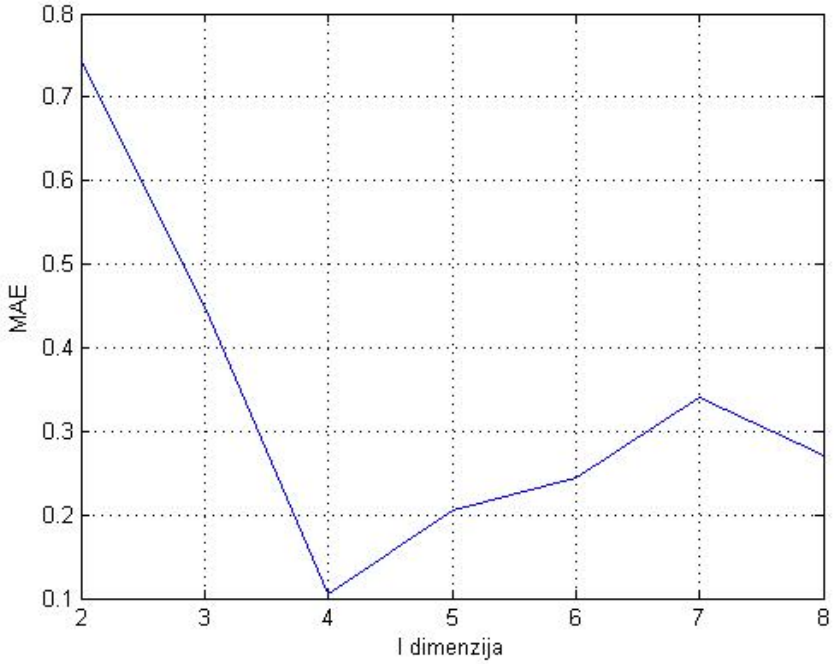
Iz Primjera 4.c očito je da parametri k i l mogu bitno utjecati na rezultat algoritma pa je važno provesti analizu za oba parametra i pronaći one koji najbolje odgovaraju sustavu.

Analiza parametra k:
Testiranje za parametar k provedeno je za $k = 2, \ldots, 8$ i fiksan $l = 3$, a rezultat je prikazan na Slici 6. Sukladno analizi, za najprikladniji k može se uzeti $k = 3$, odnosno najbolja aproksimacije matrice R je matrica ranga $k = 3$.



Slika 6: *Za sustav dan Tablicom 2 provedena je analiza parametar k.* Za fiksan $l = 3$ testirani su $k \in \{2, \ldots, 8\}$ i dobiveno je da je za parametar k najbolje uzeti $k = 3$.

Analiza parametra l: Testiranje za parametar l provedeno je za $l = 2, \ldots, 8$ i fiksan $k = 3$, a rezultat je prikazan na Slici 7. Analizom 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 7: *Za sustav dan Tablicom 2 provedena je analiza parametar l.* Za fiksan $k = 3$ testirani su $l \in \{2, \ldots, 8\}$ i dobiveno je da je za parametar l najbolje uzeti $l = 4$.

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 5. *U matricu sustava prikazanu Tablicom 2 dodajemo novog korisnik takvog da je on identičan nekom već postojećem korisniku, npr. $r_{new} = K5$. Pretpostavka je da će tada preporuka za korisnika r_{new} biti ista kao i preporuka za postojećeg korisnika $K5$. Predikcija za korisnika $K5$ i film $F5$, tj. Ice age(2002) iznosi 4.048, a predikcija za novog korisnika r_{new} i film $F5$ iznosi 4.1867. Možemo reći da je rezultat ovog primjera potvrdio pretpostavku, no zbog male dimenzije sustava, odnosno veće osjetljivosti podataka, vidljivo je manje odstupanje u preporuci za postojećeg i novog korisnika.*

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 razlika 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 6. *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 u sustav tako da se ponovno računa SVD dekompozicija cijelog sustava, 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. Zbog toga je neophodno koristiti i metode koje pravilno osvježavaju cijeli sustav kao što je metoda update ili još bolje metoda folding-up.*

4.4 *Update* metoda osvježavanja sustava

U ovoj cjelini na testnom sustavu od 11 korisnika i 11 proizvoda ilustrirana je metoda $update$, implementirana kako je opisano u Algoritmima 3 i 4.

Primjer 7. *Za usporedbu update i folding-in metode može se pogledati rezultat update metode na Primjeru 5. U zadanu matricu sustava prikazanu u Tablici 2 dodaje se opet novi korisnik* r_{new} *takav da je identičan postojećem korisniku* $K5$*, pretpostavka je da će tada preporuke za korisnika* r_{new} *biti ista kao i preporuka za postojećeg korisnika* $K5$*.*

Predikcija za korisnika $K5$ *i film* $F5$*, Ice age(2002) iznosi* 4.048*. Predikcija za novog korisnika* r_{new} *i film* $F5$ *pomoću Folding-in metode iznosi* 4.1867*, a pomoću* $update$ *metode iznosi* 4.1301*. Uočimo da je već na ovom malom primjeru vidljivo da je kvaliteta predikcije bolja kod metode update (MAE je manja).*

Kod *folding-in* metode će dodavanjem sve većeg broja korisnika opadati kvaliteta predikcija, no u ovom slučaju kvaliteta se neće mijenjati bez obzira na broj novih korisnika. Primjerice, ako se dodaju tri korisnika kao korisnik $K5$, preporuka za $K12$ i film $F5$ iznosi 4.1697.

Kao prethodno, može se promotriti razliku u predikciji ukoliko se novi korisnik doda metodom *update* ili se doda u početni sustav ponovno računajući cijelu SVD dekompoziciju, što je prikazano u sljedećem primjeru.

Primjer 8. *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* $F4$*, kao u Primjeru 6. Za korisnika dodanog metodom update predikcija za dani film iznosi* 3.5490*, dok za korisnika dodanog standardnom metodom u sustav predikcija iznosi* 3.4851*. Ovaj rezultat sličan je onom iz Primjera 6, no ovdje dodavanje više korisnika neće narušiti kvalitetu preporuke.*

U zadnjem primjeru promotrimo kakva će biti početna preporuka, ako se u sustav doda korisnik koji nije pogledao niti jedan film.

Primjer 9. *U slučaju kada se sustavu dodaje potpuno novi korisnik koji još nije pogledao niti jedan film, zadan s* $u_{12} = [0,0,0,0,0,0,0,0,0,0]$ *može se uočiti da će preporuka za neki film biti u skladu s prosječnom ocjenom za taj film. Točnije za korisnika* $K12$ *i film* $F1$ *predikcija iznosi* 4.1511*, odnosno* 4 *što je i prosječna ocjena tog filma kod prethodnih korisnika.*

Bibliografija

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

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

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

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

[5] Mirna Marković, *Kolaborativno filtriranje*, Odjel za matematiku, Sveučilište Josipa Jurja Strossmayera u Osijeku, diplomski rad (2016.)

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

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

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

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

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

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

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

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

[14] 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.)

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

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

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

---

[1]Prema [7] *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 os 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 "repa".
[2]https://www.amazon.com Prema [16] Amazon je u 2015. godini imao 488 milijuna proizvoda samo na američkom tržištu.
[3] https://www.netflix.com
[4]Ubrzanje za 10% odnosilo sa na poboljšanje pogreške algoritma za 10% na uzorku iz Netflix baze, za više vidi [9].
[5]Binarna skala je ona koja prikazuje je li korisnik nešto označio da mu se sviđa ili da mu se ne sviđa, ternarna skala uvodi i dodatnu vrijednost ukoliko je korisnik pogledao neki proizvod i nije ga rangirao. Za razliku od binarne i ternarne, unarna skala se ne oslanja na korisnikovo rangiranje nego na njegove akcije i u tom slučaji bilježi se samo je li se korisniku nešto svidjelo (je li nešto pogledao ili kupio), ali ne bilježi se ako mu se nešto nije svidjelo (nije pogledao ili nije kupio).
[6]Sustav neće pronaći proizvode koji nisu slični proizvodima za koje je korisnik do sada pokazao interes, nedostatak faktora iznenađenja u preporuci (engl. *serendipity*, [10])
[7]Sposobnost sustava da se dobro ponaša pri povećanju korisnika i sve većem broju zahtijeva
[8]http://www.ebay.com
[9]Slika operatora $A \in L(V,W)$ je potprostor definiran kao $\mathcal{R}(A) = \{Av : v \in V\} \leq W$, a još se naziva područje vrijednosti operatora A.
[10]Jezgra operatora $A \in L(V,W)$ je potprostor definiran kao $\mathcal{N}(A) = \{v \in V : Av = 0\}$, a još se naziva nulprostor operatora A.
[11]Za normu $\|\cdot\|$ kažemo da je unitarno invarijantna ako za sve unitarne matrice U i V vrijedi da je $\|UAV\| = \|A\|$.
[12]Nakon što se početna matrica sustava aproksimira matricom $U_k S_k V_k^T$, matricom $W := \sqrt{S_k} V_k^T$ aproksimiramo sustav od m korisnika i n proizvoda sa sustavom k korisnika i n proizvoda, gdje navedenih k korisnika predstavljaju aproksimaciju početnih m korisnika i zovemo ih pseudo korisnici.
[13]Standardna funkcija kosinus 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 stupca j matrice R.
[14]Prema [8], vrijedi da za bilo koju matricu $A \in \mathbb{R}^{m\times n}$, takvu da je $m \geq n$, postoje matrice Q i R takve da $A = QR$, pri čemu je $Q \in \mathbb{R}^{m\times m}$ ortogonalna, a $R \in \mathbb{R}^{m\times n}$ je gornje trokutasta.
[15]Anketa: https://goo.gl/forms/QPu36O4PK7FJznEt1,
Rezultati ankete:
https://docs.google.com/spreadsheets/d/1CL0j43vxyF6RVU0cp4pDvfJM
GdME1fHtdm52QORUs94/edit?usp=sharing

ISSN 1334-6083