# Sustav preporuke filmova zasnovan na suradničkom filtriranju

**Đurđević, Alan**

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1514

# SUSTAV PREPORUKE FILMOVA ZASNOVAN NA SURADNIČKOM FILTRIRANJU

Alan Đurđević

Zagreb, lipanj 2024.

SVEUČILIŠTE U ZAGREBU
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

ZAVRŠNI RAD br. 1514

# SUSTAV PREPORUKE FILMOVA ZASNOVAN NA SURADNIČKOM FILTRIRANJU

Alan Đurđević

Zagreb, lipanj 2024.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**

Zagreb, 4. ožujka 2024.

# ZAVRŠNI ZADATAK br. 1514

| | |
|---|---|
| Pristupnik: | **Alan Đurđević (0036542944)** |
| Studij: | Elektrotehnika i informacijska tehnologija i Računarstvo |
| Modul: | Računarstvo |
| Mentor: | izv. prof. dr. sc. Goran Delač |

Zadatak: **Sustav preporuke filmova zasnovan na suradničkom filtriranju**

Opis zadatka:

Proučiti i opisati postupke preporuke zasnovane na suradničkom filtriranju. Opisati obilježja algoritama i objasniti njihov princip rada nad pokaznim primjerima. Proučiti postojeće metrike za vrednovanje uspješnosti sustava preporučivanja. Prikupiti prikladan skup podataka za preporuku filmova korisnicima. Opisati programsko ostvarenje sustava preporuke te provesti vrednovanje nad primjerenim skupom podataka.

Rok za predaju rada: 14. lipnja 2024.

# Sadržaj

# 1. Uvod

U današnjem svijetu prepunom informacija izazovno je odabrati one najrelevantnije. Zbog velike rasprostranjenosti interneta i raznih digitalnih tehnologija pristup raznim informacijama postaje sve lakši, ali pitanje je kako doći do onih najrelevantnijih. Upravljanje informacijama danas je jedan od najbitnijih segmenata računarstva, stoga se razvijaju brojni sustavi koji nam u tome pomažu. Najpoznatiji su primjer razne svakodnevno posjećivane web tražilice od kojih očekujemo pomoć u pronalasku informacija koje nas zanimaju. Iste nude sadržaj koji se smatra relevantnim s obzirom na naše preference. Sustavi koji u tome pomažu nazivaju se sustavi preporuka. Sustavi preporuka zbog svoje velike primjene postaju popularniji te se sve više i više razvijaju. O samoj popularnosti i koristi sustava preporuka govori informacija da je platforma za gledanje filmova i serija, Netflix, 2006. godine ponudila nagradu od milijun dolara onome tko osmisli algoritam preporuke koji za barem 10 % nadmašuje njihov algoritam "CineMatch" u predviđanju ocjena za filmove koje daju korisnici. Pobijedio je tim "BellKor's Pragmatic Chaos", koji se sastojao od 7 istraživača, čiji je algoritam bio za 10.06% bolji od Netfiixovoga [1].

U ovome radu opisani su sustavi preporuke te potrebni algoritmi i tehnike za rad samoga sustava, pokazana je njihova podjela te programsko ostvarenje jednostavnoga sustava za preporučivanje filmova napisanoga u programskome jeziku Java. Korištena je javno dostupna baza podataka MovieLens [2] u vlasništvu istraživačke grupe GroupLens. Zaključno, predstavljeno je vrednovanje ostvarenog sustava izračunavanjem standardnih matematičkih mjera RMSE i MAE.

# 2. Sustavi preporuke

Sustavi preporuke tipovi su aplikacija koji predviđaju korisnikove odgovore/odabire s obzirom na razna svojstva. Oni u sebi sadrže mnoštvo algoritama koji se koriste kako bi se poboljšale preporuke te kako bi krajnji korisnici bili što zadovoljniji onime što sustav preporučuje. Sustavi preporuke danas su dio svake ozbiljnije web aplikacije koja u sebi ima bilo kakav doticaj s korisnicima te su općenito neizostavan dio današnjega virtualnoga svijeta. Najpopularniji su oni sustavi koji preporučuju filmove, serije, proizvode u internet trgovinama, ljude na raznim društvenim mrežama, vijesti na internet portalima i drugi.

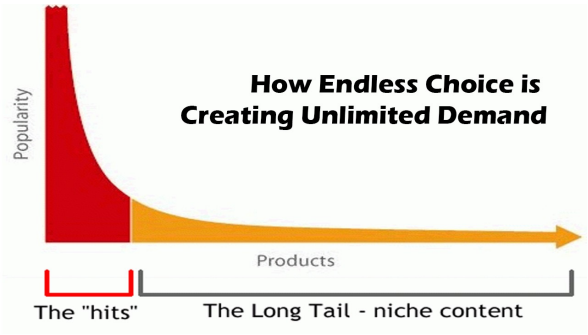
Primjeri web sjedišta koja koriste sustave preporuka:

- netflix.com, voyo.rtl.hr, primevideo.com - platforme za gledanje filmova i serija
- amazon.com, ebay.com, alibaba.com - internet trgovine
- youtube.com, vimeo.com, metacafe.com - platforme za pregled videozapisa
- facebook.com, linkedin.com, twitter.com - društvene mreže
- booking.com, trivago.com, airbnb.com - smještajne agencije

## 2.1. Fenomen dugačkoga repa

Fizičke poslovnice zbog svoga ograničenoga prostora ne mogu svakom korisniku prilagoditi svoju ponudu, odnosno zbog nedostatka resursa one se moraju prilagoditi široj masi što dovodi do toga da poslovnice najčešće izlažu samo najtraženije proizvode. S druge strane, internet trgovine svojim kupcima mogu prikazati sve što u njima postoji. Fizičke novine ograničene su brojem odnosno količinom papira na kojem se tiskaju, dok

internet portali za vijesti mogu korisnicima prikazati tisuće članaka dnevno. Ova razlika između fizičkoga i virtualnoga svijeta naziva se fenomen dugačkoga repa (slika 2.1. , eng. long tail phenomen).




**Slika 2.1.** Fenomen dugačkoga repa [3]

Ordinata predstavlja popularnost proizvoda (broj puta koliko je proizvod izabran), dok apscisa predstavlja poredane proizvode po njihovoj popularnosti (od većeg prema manjoj). Fizički entiteti u stanju su prikazati samo najpopularnije proizvode (crveni dio) dok su virtualni entiteti u stanju prikazati cijeli raspon proizvoda, od najpopularnijih pa sve do onih manje popularnih [4].

## 2.2. Podjela tipova preporuka

Tipove preporuka dijelimo u dvije osnovne skupine, personalizirane i nepersonalizirane [4]. Nepersonalizirane preporuke ne uzimaju u obzir svakoga korisnika već preporučuju na jednostavne načine poput:

- najpopularniji entiteti
- najnoviji entiteti
- najbolje ocjenjeni entiteti

Prednost nepersonaliziranih sustava preporuka je ta što su vrlo jednostavni te ih nije teško programski ostvariti i vrlo je jednostavno prikupiti podatke s pomoću kojih se stvaraju preporuke.

Personalizirane preporuke uzimaju u obzir svakoga korisnika kojemu preporučuju te na osnovu njegovih karakteristika stvaraju preporuke. Zbog toga su personalizirane

preporuke najčešće bolje i relevantnije nego nepersonalizirane. Mane personaliziranih preporuka su [5]:

- Previše rijetko uzorkovanje (eng. sparsity) - osnovni problem sustava preporuka koji se događa kada imamo puno korisnika i proizvoda, ali mali broj je u međudjelovanju. Naprimjer, u sustavu preporuke za filmove to bi značilo da su neki korisnici ocijenili mali broj filmova te da neki filmovi uopće nisu ocijenjeni.

- Premali broj početnih podataka (eng. cold start) - nastaje zbog toga što ne postoje podaci za nove korisnike ili proizvode. Naprimjer, prilikom registracije novoga korisnika ne postoji njegova povijest s pomoću koje bi sustav mogao kreirati preporuke.

- Problem skalabilnosti (eng. scalability) - nastaje zbog prevelikog broja podataka te je sustavu teško kreirati nove preporuke u stvarnome vremenu zbog velikoga broja podataka.

- Prenaučenost (eng. overfitting) - osnovni problem u strojnome učenju, kao i u sustavima preporuke. On nastaje zato što model nije u stanju generalizirati već uči previše iz danih podataka.

- Mala raznolikost (eng. diversity) - dolazi zbog toga što sustav nije u stanju ponuditi korisniku nove proizvode već prednost imaju oni popularniji zbog većega broja ocjena.

- Narušavanje privatnosti (eng. privacy) - nastaje zbog toga što sustav mora pratiti pojedine korisnike, a time dolazi do zabilježavanja podataka o samim korisnicima te se time narušava njihova privatnost.

# 3. Personalizirani sustavi

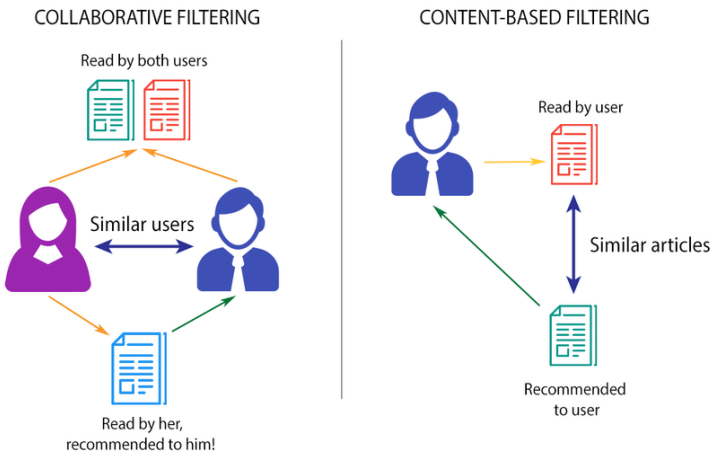
Personalizirani sustavi preporuke dijele se u dvije osnovne arhitekture:

- Sustavi preporuke zasnovani na sadržaju (eng. content-based)
- Sustavi preporuke zasnovani na suradničkom filtriranju (eng. collaborative)

Moderni komercijalni sustavi najčešće kombiniraju oba principa preporučivanja te na osnovi toga grade sustav.

## 3.1. Sadržajno filtriranje

Sustavi preporuke zasnovani na sadržajnome filtriranju temelje se na ideji predviđanja tako što gledaju proizvode koje je korisnik u prošlosti gledao ili kupio te na osnovi njih predviđaju nove proizvode koji bi se korisniku mogli svidjeti [6].




**Slika 3.1.** Razlika između dvaju arhitektura [7]

Slika 3.1. prikazuje osnovne razlike između dvije arhitekture. Lijevi dio prikazuje sustav zasnivan na suradničkome filtriranju, dok desni dio prikazuje filtriranje prema sadržaju.

# 4. Suradničko filtriranje

Sustavi preporuke zasnovani na suradničkome filtriranju dijele se u dvije osnovne skupine:

- sustavi zasnovani na memoriji (eng. memory-based)
- sustavi zasnovani na modelu (eng. model-based)

Sustavi zasnovani na memoriji koriste dostupne podatke bez pretprocesiranja te na osnovu toga daju preporuke. Osnovna je ideja ovakvoga sustava pretpostavka da će korisnici sa sličnim preferencama iz prošlosti nastaviti pokazivati slično ponašanje u budućnosti. Postoje razne izvedbe te razni algoritmi, a neki od njih opisani su u poglavlju 4.4.

Sustavi zasnovani na modelu ne koriste ranije predefinirana pravila za filtriranje već na osnovu treniranoga modela stvaraju preporuke. Ovakav način rada specifičan je za područje strojnoga učenja. Također, postoje razni algoritmi za izvedbu ovih sustava, a neki od njih su:

- Bayesove mreže (eng. Bayesian networks)
- grupiranje podataka (eng. clustering models)
- neuronske mreže (eng. neural networks)
- stabla odluke (eng. decision tree)
- matrična faktorizacija (eng. matrix factorization)

U ovome radu naglasak će biti na sustavima zasnovanima na memoriji.

## 4.1. Sustavi zasnovani na memoriji

Sustav preporuka zasnovan na memoriji fokusira se na sličnosti između korisnika te na osnovu toga filtrira podatke. Da bi se mogao objasniti takav princip rada prvo je potrebno konstruirati korisničke profile koji su zapravo vektori. Svaki korisnik bit će modeliran kao jedan redak u matrici, a stupci pripadajuće matrice bit će ocjene proizvoda koje je korisnik dao. Ako korisnik nije ocijenio proizvod, matrica će na toj poziciji ostati prazna. Ocjene korisnika za proizvode su u intervalu od 0.5 do 5, s inkrementom od 0.5. Korisnici su slični ako su blizu prema nekoj mjeri sličnosti. Mjera sličnosti ima mnogo, ali one najvažnije objašnjene su u poglavlju 4.2. Preporuka za nekog korisnika dana je na osnovi najsličnijih korisnika tomu korisniku te se preporučuju proizvodi koje slični korisnici preferiraju.

Tablica 4.1. Matrica korisnika i proizvoda

| | I1 | I2 | I3 | I4 | I5 | I6 | I7 |
|---|---|---|---|---|---|---|---|
| U1 | 4 | | | 5 | 1 | | |
| U2 | 5 | 5 | 4 | | | | |
| U3 | | | | 2 | 4 | 5 | |
| U4 | | 3 | | | | | 3 |

U ovome primjeru matrica je dimenzija 4 x 7 (4 retka i 7 stupaca), odnosno sastoji se od 4 različita korisnika koji mogu ocijeniti 7 različitih proizvoda. Korisnici ne moraju imati jednak broj ocijenjenih proizvoda, naprimjer korisnik U2 je ocijenio proizvode I1, I2, I3 redom ocjenama 5, 5, 4 dok je korisnik U4 ocijenio samo proizvode I2, I7 s ocjenom 3. Također se vidi da je većina matrice zapravo prazna odnosno najčešće će biti više proizvoda koje korisnici nisu ocijenili nego onih koje jesu.

## 4.2. Mjere sličnosti

Prvo je pitanje, kada razmišljamo o sličnosti korisnika s obzirom na proizvode, kako na osnovi redaka i stupaca u matrici odrediti sličnost. U tome nam pomažu razne matematičke mjere sličnosti dvaju vektora:

- kosinusova sličnost (eng. cosine similarity)
- Pearsonov koeficijent korelacije (eng. Person correlation coefficient)

- Spearmanov koeficijent korelacije (eng. Spearman correlation coefficient)
- Jaccardova sličnost (eng. Jaccard similarity)
- Euklidska sličnost (eng. Euclidean similarity)
- L1 sličnost (eng. Manhattan similarity)

U idućim primjerima $u$ i $v$ predstavljaju korisnike, $I$ predstavlja skup proizvoda koje su korisnici ocijenili, dok $r$ predstavlja poziciju u matrici.

### 4.2.1. Kosinusova sličnost

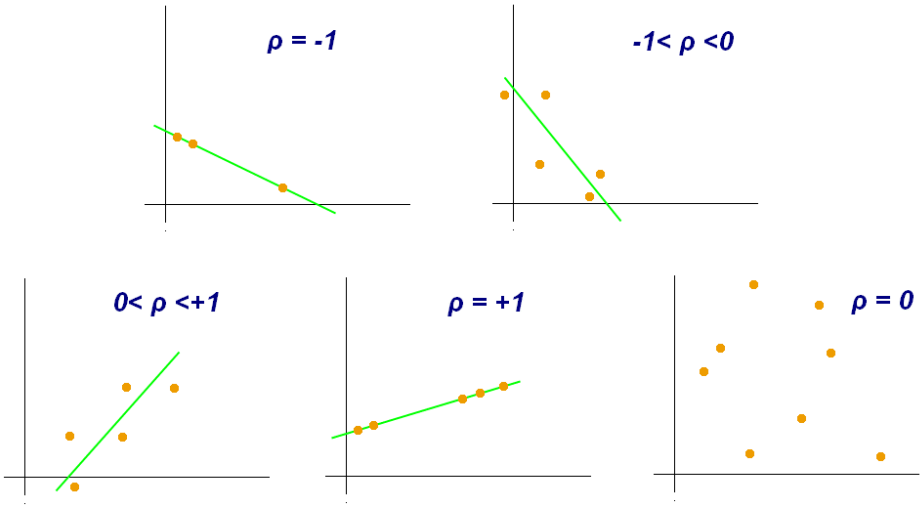
Kosinusova sličnost zasniva se na kosinusu kuta između dvaju vektora. Vrijednosti koje u ovome slučaju može poprimiti nalaze se u intervalu [0,1] zbog toga što su ocjene uvijek pozitivan broj [0.5-5]. Vrijednost 1 predstavlja potpunu sličnost, odnosno kut između vektora iznosi 0°dok vrijednost 0 predstavlja potpunu razliku, to jest kut je između vektora pravi (90°).

$$Cosine(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2} \sqrt{\sum_{i \in I_v} r_{vi}^2}} \tag{4.1}$$

### 4.2.2. Pearsonov koeficijent korelacije

Ovu mjeru za mjerenje linearne veze predložio je Karl Pearson te se ona danas učestalo koristi u statistici. Vrijednosti koje poprima koeficijent kreću se u intervalu [-1,1], 1 predstavlja strogu linearne korelaciju dok -1 predstavlja strogu negativnu korelaciju, 0 nam govori da korelacija između dvaju vektora ne postoji.

$$PCC(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})(r_{vi} - \overline{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \overline{r_u})^2} \sqrt{\sum_{i \in I_{ij}} (r_{vi} - \overline{r_v})^2}} \tag{4.2}$$




**Slika 4.1.** Pearsonov koeficijent korelacije [8]

### 4.2.3. Spearmanov koeficijent korelacije

Spearmanov koeficijent korelacije neparametarska je inačica Pearsonova koeficijenta koja se zasniva na rangovima[1] između varijabli. Vrijednosti koje poprima koeficijent također se kreću u intervalu [-1,1], 1 predstavlja strogu linearne korelaciju dok -1 predstavlja strogu negativnu korelaciju, 0 nam govori da korelacija između dvaju vektora ne postoji.

$$SCC(u,v) = 1 - \frac{6 * \sum_{i \in I_{uv}} (Rank(r_{ui}) - Rank(r_{vi}))^2}{n(n^2-1)} \tag{4.3}$$

### 4.2.4. Jaccardova sličnost

Jaccardova sličnost ne uzima u obzir ocjene koje su dali korisnici već vrijednost sličnosti mjeri na osnovi presjeka i ukupnog broja proizvoda koje su korisnici ocijenili. U brojniku se nalazi broj proizvoda koje su oba korisnika ocijenila dok se u nazivniku na- lazi ukupan broj proizvoda. Vrijednosti koje poprima kreću se u intervalu [0,1], što je vrijednost veća to je sličnost veća.

$$J(u,v) = \frac{|u \cap v|}{|u \cup v|} \tag{4.4}$$

[1] U statistici rang predstavlja podjelu podataka od najmanjih pa sve do najvećih vrijednosti. Svaki podatak se u takvome poretku numerira te je to njegov rang.

### 4.2.5. Euklidska sličnost

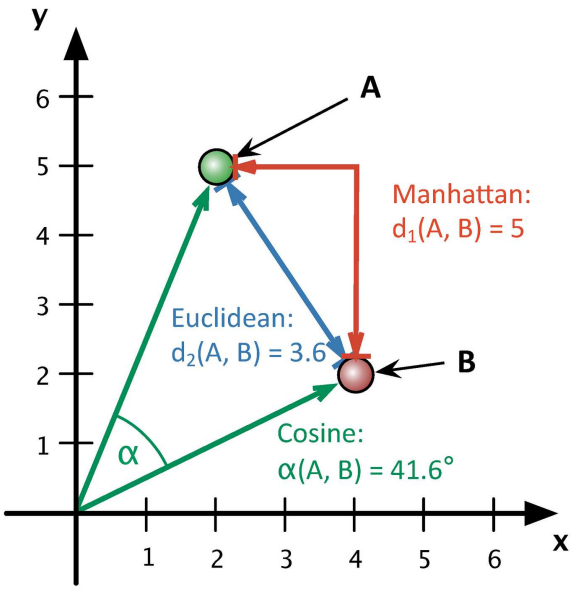
Euklidska sličnost mjera je koja mjeri udaljenost dvaju vektora na osnovi točaka u n-dimenzionalnom prostoru. Ova se mjera često naziva Pitagorina udaljenost.

$$d(u,v) = \sqrt{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2} \tag{4.5}$$

### 4.2.6. L1 sličnost

L1 ili Manhattan sličnost predstavlja apsolutnu udaljenost između dvije točke u n-dimenzionalnom prostoru. Ova mjera ima čestu uporabu prilikom određivanja heurističke funkcije u strojnome učenju.

$$l1(u,v) = \sum_{i \in I_{uv}} |r_{ui} - r_{vi}| \tag{4.6}$$




**Slika 4.2.** Prikaz različitih sličnosti [9]

## 4.3. Normalizacija ocjena

Prilikom određivanja sličnih korisnika te prije provođenja algoritma predviđanja poželjno je obaviti normalizaciju ocjena korisnika za filmove. Normalizacija je proces u kojemu ćemo od svake ocjene korisnika za pojedini film oduzeti aritmetičku sredinu svih

ocjena kojima je pojedini korisnik ocijenio filmove. Proces normalizacije nam omogućuje bolji izračun sličnosti zato što će prilagoditi ocjene po srednjoj vrijednosti, odnosno neutralizirati one korisnike koji daju samo polarne ocjene (0.5 i 5) te će s time dati bolje procjene i izračune. Također, normalizacija nam prilikom izračuna kosinusove sličnosti daje bolje rezultate zbog toga što će korisnici s različitim gledištima biti udaljeni puno više nego li u slučaju ako ne normaliziramo.

Primjer

Tablica 4.2. Vektor korisnika U1 prije normalizacije

| | I1 | I2 | I3 | I4 | I5 | I6 | I7 |
|---|---|---|---|---|---|---|---|
| U1 | 4 | | | 5 | 1 | | |

Tablica 4.3. Vektor korisnika U1 nakon normalizacije

| | I1 | I2 | I3 | I4 | I5 | I6 | I7 |
|---|---|---|---|---|---|---|---|
| U1 | 2/3 | | | 5/3 | -7/3 | | |

Tablica 4.2. prikazuje ocjene korisnika U1 prije proces normalizacije, dok tablica 4.3. prikazuje ocjene korisnika U1 nakon normalizacije (svakoj ocjeni je oduzeta aritmetička sredina svih ocjena toga korisnika koja u primjeru iznosi $\frac{4+5+1}{3} = \frac{10}{3}$).

Preporuka je svakako prije računanja sličnosti normalizirati ocjene svih korisnika, proces nije vremenski niti algoritamski zahtjevan, a doprinos rezultatima je značajan.

## 4.4. Algoritam zasnovan na memoriji

Osnovni je algoritam predviđanja ocjena za korisnika $U$ i proizvod $I$ taj da se pronađe $n$ (unaprijed određeni broj) sličnih korisnika koji su ocijenili proizvod $I$. Kako bi se dobili točniji rezultati prije provođenja procesa prikupljanja sličnih korisnika potrebno je provesti normalizaciju ocjena svih korisnika. Nakon što se pronađu slični korisnici potrebno je izračunati aritmetičku sredinu normaliziranih ocjena tih korisnika za traženi proizvod $I$ te tu vrijednost pridodati aritmetičkoj sredini svih ocjena od korisnika $U$.

Primjer.

Nakon normalizacije ocjena pojedinoga korisnika te nakon pronalaska sličnih korisnika korisniku C prema nekoj mjeri sličnosti koji su ocijenili traženi proizvod (tablica

Tablica 4.4. Normalizirana matrica sličnih korisnika korisniku C

| | I1 | I2 | I3 |
|---|---|---|---|
| A | 5/3 | -1/3 | -4/3 |
| B | -1/2 | 1/2 | |
| C | | -3/2 | 3/2 |

4.4.), potrebno je predvidjeti ocjenu za proizvod I1. Ocjena za proizvod I1 jednaka je sumi aritmetičke sredini ocjena korisnika sličnih korisniku C za proizvod I1 te dodanoj aritmetičkoj sredini ocjena korisnika C (3.5). $\frac{5/3-1/2}{2} + 3.5 = 4.08$.

# 5. Programsko ostvarenje

Programsko ostvarenje sustava preporuke zasnovanog na suradničkome filtriranju napisano je u programskome jeziku Java koristeći Eclipse kao razvojno okruženje. Skup podataka korišten za rad sustava te za kasnije ispitivanje javno je dostupna baza podataka filmova te ocjena MovieLens.

Za određivanje sličnosti između korisnika korištena je kosinusna sličnost (poglavlje 4.2.1.) s normaliziranim ocjenama za korisnike (poglavlje 4.3.). Algoritam korišten za procjenu ocjena osnovni je algoritam iz poglavlja 4.4.

## 5.1. Programski jezik Java

Programski jezik Java (slika 5.1.) jedan je od najraširenijih i najkorištenijih programskih jezika današnjice. Prema TIOBE indeksu[1] on zauzima visoko 4. mjesto u popularnosti [10]. Javu je originalno razvio James Gosling 1991. godine u kompaniji Sun Microsystems, ali 2010. godine kompanija Oracle preuzima kompaniju Sun Microsystem te s time preuzima odgovornost upravljanja Javom [11]. Jedna je od najvećih prednosti Jave što se stalno mijenja te prati trendove u programiranju dok još uvijek ostaje kompatibilna sa starijim verzijama programskoga jezika. Također, najveća prednost programskoga jezika Java jest ideja napiši-jednom-pokreni-bilo-gdje. Centralna tema u razvoju bila je potpora za višeplatformnost, odnosno ideja je da se programeru ponudi apstraktni pogled na računalo koji ne uključuje detalje platforme na kojima se program izvodi. Java definira apstraktni računski stroj kod kojeg je sve propisano i opisano specifikacijom, što je moguće jer se prilikom programiranja u Javi programi ne pišu specifično za neki tip računala već za Javin virtualni stroj koji je svjestan na kojoj se platformi izvodi te kako

[1] TIOBE indeks je pokazatelj popularnosti programskoga jezika. Indeks se ažurira jednom mjesečno, a ocjene su bazirane na broju programera koji koriste programski jezik, broju tečajeva i drugih značajki. https://www.tiobe.com/tiobe-index/

se na toj platformi obavljaju pojedini zadaci [12]. Nove verzije Jave izlaze svakih pola godine te je u ovome trenutku najnovija Java 22. Programsko ostvarenje ovoga sustava napisano je u Javi 20.


**Slika 5.1.** Logotip programskoga jezika Java [13]

## 5.2. Razvojno okruženje Eclipse

Eclipse (slika 5.2.) jedno je od najpopularnijih programskih okruženja za pisanje programskoga koda. Pretežito se koristi za pisanje koda u programskome jeziku Java o čemu govori podatak da je do 2016. godine bio najpopularnije okruženje za Javu, a danas zauzima drugo mjesto [14], no Eclipse također pruža potporu i za druge programske jezike poput C++, C#, Python, Rusta i drugih. Najveća prednost Eclipsea je što se lako modificira te se u njega jednostavno ubacuju razni pluginovi[2] ovisno o tome za što nam je potreban. Eclipse je 2001. godine razvila Eclipse Foundation koja je i dalje vlasnik. Eclipse se stalno mijenja te nova verzija izlazi svaka 3 mjeseca, a trenutno je najnoviji Eclipse verzije 2024-03. Programsko ostvarenje ovog sustava napisano je uz pomoć najnovije verzije.


**Slika 5.2.** Logotip razvojnoga okruženja Eclipse [16]

[2]Pluginovi su komponente programske potpore koje omogućuju specifične funkcionalnosti za već postojeću aplikaciju ili program. Oni omogućuju nadogradnju, personalizaciju i optimizaciju bez mijenjanja postojećega koda.[15]

## 5.3. MovieLens baza podataka

"GroupLens" istraživački je laboratorij odjela za računarsku znanost i inženjerstvo na Sveučilištu u Minnesoti. Specijalizirani su za sustave preporuka, internetske zajednice, mobilne i sveprisutne tehnologije, digitalne knjižnice te lokalne geografske informacijske sustave. Istraživačka skupina prikupila je i omogućila korištenje raznih baza podataka poput MovieLensa, WikiLensa, Book-Crossinga i drugih. MovieLens dolazi u više oblika ovisno o broju prikupljenih podataka. Postoji baza od 25 milijuna, 20 milijuna, jedan milijun, sto tisuća te također postoji baza od jedne milijarde simuliranih zapisa koja je zapravo proširenje one od 20 milijuna zapisa [17].

U ovome radu korištena je baza koja se sastoji od 25000095 zapisa u kojoj su korisnici ocijenili ukupno 62423 filma. Baza se sastoji od 162541 korisnika koji su nasumično odabrani. Svaki je korisnik ocijenio minimalno 20 filmova te je predstavljen samo svojim identifikacijskim brojem. Ocjene kojima su korisnici ocjenjivali filmove su od 0.5 do 5.0. Datoteke koje su korištene u radu su ratings.csv te movies.csv napisane u formatu CSV[3](slika 5.3.).

```
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
6,Heat (1995),Action|Crime|Thriller
```

**Slika 5.3.** Primjer CSV zapisa

[3]CSV (eng. comma-separated values) je format u kojemu su vrijednosti zapisa odvojene zarezima dok su sami zapisi odvojeni sa znakom novoga retka. Unutar same datoteke prvi redak predstavlja zaglavlje, odnosno imena stupaca dok su ostali redci sami zapisi. Ovakav način zapisivanja se koristi u raznim područjima poput baza podataka, strojnoga učenja i drugih.

Datoteka ratings.csv sastoji se od 4 stupca:

- userId - jedinstveni identifikator korisnika
- movieId - jedinstveni identifikator filma
- rating - korisnikova ocjena za film
- timestamp - vremenski trenutak ocjenjivanja filma

Tablica 5.1. Dio tablice ratings.csv

| userId | movieId | rating | timestamp |
|---|---|---|---|
| 1 | 296 | 5.0 | 1147880044 |
| 1 | 306 | 3.5 | 1147868817 |
| 1 | 307 | 5.0 | 1147868828 |
| 1 | 665 | 5.0 | 1147878820 |
| 1 | 889 | 3.5 | 1147868510 |

U prikazu (tablica 5.1.) djelomično je prikazana datoteka ratings.csv koja zapravo predstavlja tablicu ocjena. Na primjer, prvi zapis u tablici predstavlja da je korisnik s userId=1 ocijenio film s movieId=296 ocjenom 5.0 u zadanome trenutku.

Datoteka movies.csv sastoji se od 3 stupca:

- movieId - jedinstveni identifikator filma
- title - puni naziv filma te godina izlaska
- genres - popis žanrova kojima film pripada odvojenih znakom "|"

Tablica 5.2. Dio tablice movies.csv

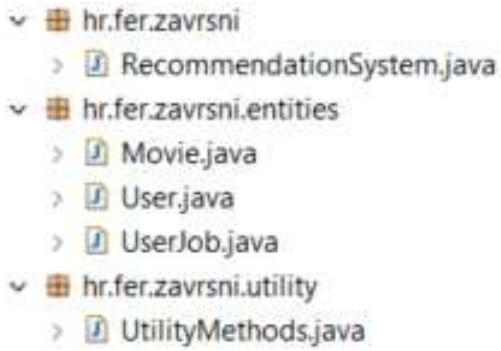
| movieId | title | genres |
|---|---|---|
| 1 | Toy Story (1995) | Adventure\|Animation\|Children\|Comedy\|Fantasy |
| 2 | Jumanji (1995) | Adventure\|Children\|Fantasy |
| 3 | Grumpier Old Men (1995) | Comedy\|Romance |
| 4 | Waiting to Exhale (1995) | Comedy\|Drama\|Romance |
| 5 | Father of the Bride Part II (1995) | Comedy |

U prikazu (tablica 5.2.) djelomično je prikazana datoteka movies.csv koja zapravo predstavlja tablicu filmova. Naprimjer, prvi zapis u tablici predstavlja da se film s movieId=1 zove "Toy Story" te da pripada navedenim žanrovima u 3. stupcu.

## 5.4. Programski kod i rad sustava

### 5.4.1. Paketi i razredi

Prilikom pisanja programskoga koda u Javi zbog bolje preglednosti i lakše nadogradnje poželjno je kod dijeliti u pakete[4]. U ovome programskom ostvarenju postoje vršni paket *hr.fer.zavrsni* te dva potpaketa *hr.fer.zavrsni.entities* i *hr.fer.zavrsni.utility* (slika 5.4.).



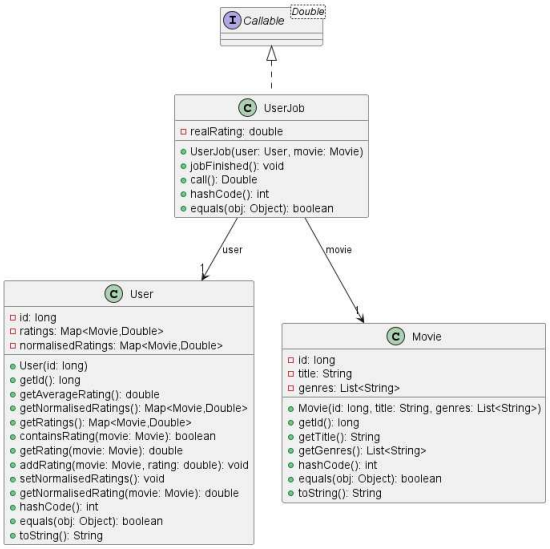

**Slika 5.4.** Prikaz paketa i pripadajućih klasa

Paket *hr.fer.zavrsni* (slika 5.6.) sadrži razred *RecommendationSystem* u kojemu se nalazi *main* metoda čijim pokretanjem započinje rad sustava. Unutar tog razreda nalaze se metode koje omogućavaju sam rad sustava.

Paket *hr.fer.zavrsni.entities* (slika 5.5.) sadrži razrede zadužene za enkapsulaciju entiteta:

- *Movie* - enkapsulira objekte koji predstavljaju pojedini film
- *User* - enkapsulira objekte koji predstavljaju pojedinog korisnika
- *UserJob* - enkapsulira posao koji je potreban za evaluaciju sustava

[4]U Javi paketi predstavljaju nakupinu sličnih klasa. Njihova svrha je omogućiti lakše grupiranje, pretragu te pristup programskome kodu.




**Slika 5.5.** UML prikaz paketa hr.fer.zavrsni.entities

Unutar paketa *hr.fer.zavrsni.utility* (slika 5.7.) nalazi se razred *UtilityMethods* koji sadrži metode osnovne matematičke funkcionalnosti potrebne za rad sustava. Unutar samoga razreda nalaze se 3 statičke metode:

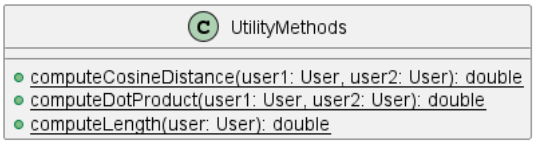
- *double computeCosineDistance(User user1, User user2)* - metoda koja računa kosinusnu udaljenost dvaju korisnika (njihovih vektora) (formula 4.1)
- *double computeDotProduct(User user1, User user2)* - metoda koja računa skalarni umnožak dvaju korisnika (njihovih vektora)(formula 5.1)
- *double computeLength(User user)* - metoda koja računa duljinu korisnika (njegovog vektora)(forumla 5.2)

$$DP(u,v) = \sum_{i=1}^{M} u_i v_i \tag{5.1}$$

$$L(u) = \sqrt{\sum_{i=1}^{M} u_i^2} \tag{5.2}$$

Unutar formula 5.1 i 5.2 $u$ i $v$ predstavljaju pojedine korisnike dok $u_i$ te $v_i$ predstavljaju ocjenu korisnika za i-ti film. $M$ predstavlja skup svih filmova.


**Slika 5.6.** UML prikaz paketa hr.fer.zavrsni




**Slika 5.7.** UML prikaz paketa hr.fer.zavrsni.utility

Slike 5.5., 5.6. i 5.7. predstavlja UML prikaz pojedinih razreda unutar paketa. Alat korišten za izradu prikaza je PlantUML (https://plantuml.com/).

"UML (engl. Unified Modeling Language) je normirani vizualni jezik koji se koristi za modeliranje programske potpore. Pomoću jezika UML mogu se izraditi UML-dijagrami – grafički prikazi koji se koriste za modeliranje programske potpore. Oni pružaju normirani i vizualni način za prikazivanje arhitekture, oblikovanja i ponašanja složenih programskih sustava, što olakšava komunikaciju, suradnju i razumijevanje među razvojnim inženjerima. UML-dijagrami pomažu u održavanju dosljednosti i jasnoće u dokumentaciji projekata te za identificiranje problematičnih područja prije implementacije. UML-dijagrame je u 1990-ima predložila grupa razvojnih inženjera predvodenih Grady Boochom, Jamesom Rumbaughom i Ivarom Jacobsonom. Oni su objedinili u jedinstvenu normu različite načine bilježenja modela programske potpore koji su se dotad koristile pri razvoju, a koje dolaze iz ranije razvijenog objektno orijentiranog pristupa analizi i dizajnu (engl. Object-Oriented Analysis and Design, OOAD), tehnike za objektno modeliranje (engl. Object Modeling Technique, OMT) i objektno orijentiranog programskog inženjerstva (engl. Object-Oriented Software Engineering, OOSE). Glavni cilj je bio pružiti normirani način bilježenja (notaciju) za modeliranje programskih sustava, zasnivan na najboljim praksama i tehnikama koje koriste iskusni razvojni inženjeri. Od tada pa sve do danas, UML se kontinuirano razvija te se danas smatra najcešće korištenom grafickom notacijom za modeliranje programske potpore. Njegova popularnost posljedica je činjenice da ga podržava velik broj programskih alata i radnih okvira, što olakšava razvojnim inženjerima izradu, izmjenu i dijeljenje UML-dijagrama. Zbog svoje opće prihvaćenosti, UML-dijagrami postali su nezaobilazan alat u razvoju programske potpore. [18]"

### 5.4.2. Rad sustava

Nakon što se pokrene sustav te se učitaju potrebni podaci u memoriju, od korisnika se očekuje da unese broj koji predstavlja jednu od pet mogućnosti koje mu sustav nudi (slika 5.8.):

- preporuka filmova za postojećeg korisnika - 1
- preporuka filmova za nepostojećeg korisnika - 2
- predviđanje pojedinačnih pozicija u matrici za postojećeg korisnika - 3
- izračun RMSE (eng. Root Mean Square Error) i MAE (eng. Mean Absolute Error) - 4
- izlazak iz sustava - 5

```
Enter number:
Top movies for existing user-1
Top movies for you (not existing user)-2
Estimating ratings-3
Error-4
Exit-5
```

**Slika 5.8.** Prikaz svih opcija sustava

#### Preporuka filmova za postojećeg korisnika

Prilikom preporuke filmova za postojećeg korisnika *U*, sustav traži 50 najsličnijih korisnika te na osnovu filmova koje su oni ocijenili, a korisnik *U* nije, predviđaju ocjene za dane filmove. Filmovi se nakon toga poredaju po predviđenim ocjenama u silaznome redoslijedu te se prikažu korisniku (slika 5.9.).

U primjeru na slici 5.9. traži se preporuka najboljih filmova za korisnika s jedinstvenim identifikatorom 528 te se za njega preporučuje 10 filmova koje taj korisnik nije pogledao te one za koje sustav smatra da su najbolji za njega.

```
private static List<Movie> findNewTopMovies(User user, int n) {
    //finding similar users and movies to estimate rating
    List<User> similarUsers = similarUsers(user, 50);
    Set<Movie> movies = new HashSet<>();
```

```
Please enter user id.
528
Top movies:
Movie [id=610, title=Heavy Metal (1981)]
Movie [id=373, title=Red Rock West (1992)]
Movie [id=783, title="Hunchback of Notre Dame, The (1996)"]
Movie [id=534, title=Shadowlands (1993)]
Movie [id=535, title=Short Cuts (1993)]
Movie [id=800, title=Lone Star (1996)]
Movie [id=594, title=Snow White and the Seven Dwarfs (1937)]
Movie [id=348, title=Bullets Over Broadway (1994)]
Movie [id=116, title=Anne Frank Remembered (1995)]
Movie [id=121, title="Boys of St. Vincent, The (1992)"]
```

**Slika 5.9.** Preporuka filmova za postojećeg korisnika

```
    for (User similarUser : similarUsers) {
       similarUser.getRatings().entrySet().stream().filter(e ->
           !user.containsRating(e.getKey()))
       .forEach(e -> movies.add(e.getKey()));
    }
    //estimating ratings for movies
    Map<Movie, Double> map = new HashMap<Movie, Double>();
    for (Movie m : movies) {
       double rating = estimateRating(user, similarUsers, m);
       map.put(m, rating);
    }
    //sorting and return list of top movies
    return map.entrySet().stream().sorted(new Comparator<Entry<Movie,
        Double>>() {
       @Override
       public int compare(Entry<Movie, Double> o1, Entry<Movie, Double> o2)
           {
          return o2.getValue().compareTo(o1.getValue());
       }
    }).limit(n).map((e) -> e.getKey()).toList();
}
```

Statička metoda *findNewTopMovies* osnovna je metoda za rad ove funkcionalnosti. Argumenti koje metoda dobiva su objekt razreda *User* koji predstavlja traženog korisnika *U* te integer *n* koji predstavlja broj filmova koje je potrebno pronaći. Ona u prvome dijelu

poziva metodu *similarUsers* koja pronalazi 50 najsličnijih korisnika te nakon toga obavlja predviđanje ocjena. U posljednjem dijelu obavlja se sortiranje silaznim redoslijedom te se korisniku vraća *n* "najboljih" filmova za njega.

### Preporuka filmova za nepostojećeg korisnika

Prilikom preporuke filmova za nepostojećeg korisnika, zbog toga što korisnik ne postoji, prvo je potrebno unijeti podatke kako bi sustav mogao dati preporuke. Podaci koje je potrebno unijeti parovi su movieId-ocjena (slika 5.10.).

```
Enter movie:
98929
Enter rating:
5
Enter movie:
231
Enter rating:
3,5
Enter movie:
1999
Enter rating:
2
```

**Slika 5.10.** Unos filmova i ocjena

Prilikom unosa podataka potrebno je unijeti jedinstveni identifikator filma te pripadnu ocjenu. Za određivanje "najboljih" filmova koristi se metoda *findNewTopMovies* isto kao i za određivanje filma za postojećega korisnika.

### Predviđanje pojedinačnih pozicija u matrici za postojećeg korisnika

Prilikom predviđanja pozicija u matrici za postojećeg korisnika potrebno je unijeti jedinstveni identifikator korisnika. Nakon što se unese identifikator sustav će prikazati popis filmova koje je korisnik ocijenio te je među tim filmovima potrebno odabrati onaj za koji se želi ispitati sustav. Nakon što se unese jedinstveni identifikator filma sustav će prikazati pravu ocjenu korisnika te ocjenu koju je sustav predvidio.

```
User id: 9000
Movie [id=2278, title=Ronin (1998)]
Movie [id=1258, title="Shining, The (1980)"]
Movie [id=2028, title=Saving Private Ryan (1998)]
Movie [id=1261, title=Evil Dead II (Dead by Dawn) (1987)]
Movie [id=2542, title="Lock, Stock & Two Smoking Barrels (1998)"]
Movie [id=1263, title="Deer Hunter, The (1978)"]
```

**Slika 5.11.** Unos identifikator korisnika

```
Movie [id=733, title="Rock, The (1996)"]
Movie [id=1247, title="Graduate, The (1967)"]
Movie [id=3039, title=Trading Places (1983)]
Movie [id=480, title=Jurassic Park (1993)]
Movie id:  733
Movie [id=733, title="Rock, The (1996)"]
Real rating: 4,000000, Estimated rating: 3,500000
```

**Slika 5.12.** Djelomičan prikaz filmova te prikaz rezultata

Slika 5.11. prikazuje unos jedinstvenoga identifikator korisnika koji je u ovome slučaju 9000. Nakon što se unese identifikator, korisniku se ispišu ponuđeni filmovi. Slika 5.12. prikazuje unos jedinstvenoga identifikatora filma te u zadnjemu retku rezultate obrade. U ovome primjeru zatraženo je predviđanje ocjene za film s identifikatorom 733 koju je dao korisnik 9000. Prava ocjena je 4.0 dok je predviđena 3.5.

### Ostale metode

Unutar sustava postoji mnoštvo metoda, no ovdje su izdvojene one najvažnije za rad samoga sustava:

- *List<User> estimateRating(User u, int number)*
- *List<User> similarUsers(User u, int number)*
- *List<User> similarUsersBasedOnMovie(User u, int number, Movie movie)*

```
public static double estimateRating(User user, List<User> similarUsers,
    Movie movie) {
   double averageMovieRating = getNormalisedAverageRating(similarUsers,
      movie);
   double averageMoviesRatingOfUser = user.getAverageRating();
   double estimatedRating = averageMovieRating +
      averageMoviesRatingOfUser;
   return round(estimatedRating);
}
```

Metoda *estimateRating* služi za procjenu ocjena. Argumenti koje ona dobiva su objekt razreda *User*, koji predstavlja korisnika za kojeg je potrebno predvidjeti ocjenu, lista slič-

nih korisnika te objekt razreda *Movie*, koji predstavlja film za koji je potrebno procijeniti ocjenu. Metoda radi na principu osnovnog algoritma koji je opisan u poglavlju 4.4. Na kraju je još potrebno s pomoću metode *round* zaokružiti vrijednosti na polovične. Na primjer broj 4.20 će se zaokružiti na 4 dok će se 4.30 zaokružiti na 4.5.

```
private static List<User> similarUsers(User u, int number) {
   Map<User, Double> cosineDistances = new HashMap<>();
   //find cosine distance between desired user and all users
   for (Entry<Long, User> userEntry : users.entrySet()) {
      User user = userEntry.getValue();
      if (!u.equals(user)) {
         double cosineDistance = UtilityMethods.computeCosineDistance(u,
             user);
         cosineDistances.put(user, cosineDistance);
      }
   }
   //sort users based on cosine distance
   List<User> similarUsers =
       cosineDistances.entrySet().stream().sorted(new
       Comparator<Entry<User, Double>>() {
      @Override
      public int compare(Entry<User, Double> o1, Entry<User, Double> o2)
          {
         return o2.getValue().compareTo(o1.getValue());
      }
   }).filter((e) -> !e.getValue().isNaN()).limit(number).map((e) ->
       e.getKey()).toList();
   return similarUsers;
}
```

Metoda *similarUsers* pronalazi najsličnije korisnike za nekog zadanog korisnika. Argumenti koje ona dobiva su objekt razreda *User* koji predstavlja korisnika za kojega je potrebno pronaći slične korisnike te integer *number* koji predstavlja broj korisnika koje je potrebno pronaći. Ona u početku računa kosinusnu udaljenost (poglavlje 4.2.1.) između

svih korisnika i zadanoga korisnika. Na kraju se odvija silazno sortiranje te metoda vraća *n* najsličnijih korisnika.

```
public static List<User> similarUsersBasedOnMovie(User u, int number,
    Movie movie) {
   Map<User, Double> cosineDistances = new HashMap<>();
   for (Entry<Long, User> userEntry : users.entrySet()) {
      User user = userEntry.getValue();
      if (!u.equals(user) && user.containsRating(movie)) {
         double cosineDistance = UtilityMethods.computeCosineDistance(u,
             user);
         cosineDistances.put(user, cosineDistance);
      }
   }
   List<User> similarUsers =
       cosineDistances.entrySet().stream().sorted(new
       Comparator<Entry<User, Double>>() {
      @Override
      public int compare(Entry<User, Double> o1, Entry<User, Double> o2) {
         return o2.getValue().compareTo(o1.getValue());
      }
   }).filter((e) -> !e.getValue().isNaN()).limit(number).map((e) ->
       e.getKey()).toList();
   return similarUsers;
}
```

Također vrlo bitna metoda je metoda *similarUsersBasedOnMovie* koja radi na sličnome principu rada kao i metoda similarUsers, jedino što ona kao dodatan kriterij ima taj da pronalazi najsličnije korisnike koji su pogledali zadani film. Zbog toga je ovoj metodi kao argument potrebno dodati objekt razreda *Movie* koji predstavlja zadani film.

# 6. Vrednovanje sustava

Najpopularnija je mjera za vrednovanje sustava preporuke korijen srednje kvadratne pogreške (eng. Root Mean Square Error).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{6.1}$$

Ona u obzir uzima kvadrate razlike između procjenjene vrijednosti i stvarne vrijednosti ocjene za pojedini proizvod.

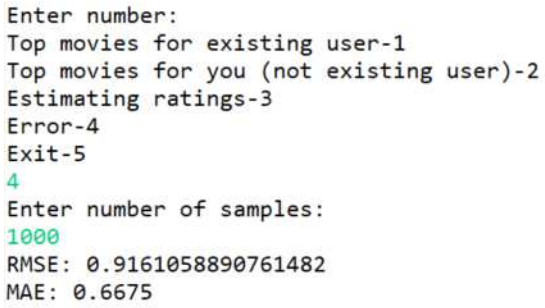
Još jedna bitna mjera za vrednovanje sustava preporuke je mjera srednje apsolutne pogreške (eng. Mean Absolute Error).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{6.2}$$

Ona u obzir uzima apsolutne vrijednosti razlike između procjenjene vrijednosti i stvarne vrijednosti ocjene za pojedini proizvod.

Prilikom pokretanja sustava, kako bi započela evaluacija, potrebno je odabrati opciju broj 4. Nakon odabira opcije, od korisnika će se zatražiti da unese broj koji predstavlja broj ocjena pomoću kojih je potrebno izračunati RMSE i MAE te samo izračunavanje nakon toga započinje.

Na slici 6.1. prikazan je izračun RMSE i MAE za 1000 uzoraka. RMSE iznosi približno 0.9161, dok MAE iznosi 0.6675. Vrijednosti su između 0.5 i 1 što znači da sustav vrlo dobro predviđa ocjene.




**Slika 6.1.** Prikaz rada izračunavanja RMSE i MAE

Najvažnija metoda za izračun pogreške jest *error*. Unutar te metode odvija se odabir *n* nasumičnih uzoraka (parova korisnik-film) koji će biti izbačeni iz početnoga skupa podataka i čije će se vrijednosti predviđati i pomoću kojih će se izračunati RMSE i MAE.

# 7. Zaključak

Sustavi preporuke kompleksni su sustavi koji nam u današnjem digitalnom dobu s mnoštvom raznih informacija omogućuju njihovo iskorištavanje davanjem preporuka. U ovome je radu naglasak stavljen na sustave zasnovane na suradničkom filtriranju te su opisani poneki algoritmi koji omogućuju rad samoga sustava. Također, navedene su i objašnjene najvažnije mjere kojima određujemo sličnost između različitih entiteta.

Osim toga, objašnjene su tehnologije koje mogu poslužiti za izradu samoga sustava te je prikazan rad programskoga ostvarenja jednostavnog sustava preporuke filmova napisanog u programskom jeziku Java koji koristi javno dostupnu bazu podataka MovieLens. Na kraju je rada prikazano vrednovanje takvoga sustava. Izračunat RMSE iznosi približno 0.9161, dok MAE iznosi 0.6675.

Sustavi preporuke konstantno se razvijaju i predstavljaju područje velikoga interesa. Upravo zbog tog velikog interesa, u područje strojnoga učenja, jačanja dubokog učenja i razvoja drugih naprednih algoritama dolazi do pojačanog razvoja sustava. Dakle, bit će zanimljivo vidjeti što donosi budućnost, kako će se sami sustavi nastaviti razvijati te u kojem će smjeru napredovati.

# Literatura


[1] Mariana Maroto, https://marianamarotob.medium.com/collaborative-filtering-and-some-history-on-the-netflix-prize-63d63c2af22b, [mrežno; stranica posjećena: svibanj 2024.].

[2] MovieLens, https://grouplens.org/datasets/movielens/, [mrežno; stranica posjećena: svibanj 2024.].

[3] Marc Dimmick, https://www.linkedin.com/pulse/long-tail-phenomenon-marc-dimmick-churchill-fellow-mmgmt/, [mrežno; stranica posjećena: svibanj 2024.].

[4] Stanford, http://infolab.stanford.edu/~ullman/mmds/ch9.pdf, [mrežno; stranica posjećena: svibanj 2024.].

[5] Turing.com, https://www.turing.com/kb/content-based-filtering-in-recommender-systems#disadvantages, [mrežno; stranica posjećena: svibanj 2024.].

[6] Prateek Gaurav, https://medium.com/@prateekgaurav/step-by-step-content-based-recommendation-system-823bbfd0541c#:~:text=Content%2Dbased%20recommendation%20systems%20are,their%20viewing%20and%20purchasing%20history., [mrežno; stranica posjećena: svibanj 2024.].

[7] Lionel Tondji, https://www.researchgate.net/figure/Content-based-filtering-vs-Collaborative-filtering-Source_fig5_323726564, [mrežno; stranica posjećena: svibanj 2024.].

[8] Wikipedia, https://en.wikipedia.org/wiki/Pearson_correlation_coefficient, [mrežno; stranica posjećena: svibanj 2024.].

[9] DH2016, https://dh2016.adho.org/abstracts/253, [mrežno; stranica posjećena: svibanj 2024.].

[10] Tiobe, https://www.tiobe.com/tiobe-index/, [mrežno; stranica posjećena: svibanj 2024.].

[11] IBM, https://www.ibm.com/topics/java, [mrežno; stranica posjećena: svibanj 2024.].

[12] M. Čupić, *Programiranje u Javi*, 2015.

[13] Oracle, https://en.wikipedia.org/wiki/Java_%28programming_language%29, [mrežno; stranica posjećena: svibanj 2024.].

[14] Curtis Johnson, https://www.jrebel.com/blog/best-java-ide, [mrežno; stranica posjećena: svibanj 2024.].

[15] Chiradeep BasuMallick, https://www.spiceworks.com/tech/tech-general/articles/what-are-plugins/, [mrežno; stranica posjećena: svibanj 2024.].

[16] Eclipse Foundation, https://www.eclipse.org/org/artwork/#eclipse-ide-logos, [mrežno; stranica posjećena: svibanj 2024.].

[17] GroupLens, https://grouplens.org/datasets/movielens/, [mrežno; stranica posjećena: svibanj 2024.].

[18] N. Frid i A. Jović, *Modeliranje programske potpore UML-dijagramima*, 2023.

# Popis slika

# Popis tablica

# Sažetak


## Sustav preporuke filmova zasnovan na suradničkom filtriranju

Alan Đurđević

U ovom završnom radu, analizirani su sustavi preporuka zasnovani na suradničkom filtriranju. Opisane su osnovne mjere određivanja sličnosti među korisnicima i osnovni algoritmi predviđanja ocjena. Prikazano je programsko ostvarenje sustava napisanog u programskom jeziku Java koje koristi javno dostupnu bazu podataka MovieLens te je provedeno vrednovanje rada sustava.



**Ključne riječi:** sustavi preporuka; suradničko filtriranje; programski jezik Java

# Abstract


## Movie recommendation system based on collaborative filtering

Alan Đurđević

This thesis analyzes recommendation systems based on collaborative filtering. The measures for detecting similarity between users and the basic algorithms for predicting ratings are described. The implementation of the system written in Java programming language, which uses the publicly available database MovieLens is shown and finally, the evaluation of the system is performed.



**Keywords:** recommendation systems; collaborative filtering; Java programming language