Sustavi za preporuke u e-trgovinama

Mušica, Marko

SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Marko Mušica

SUSTAVI ZA PREPORUKE U E-TRGOVINAMA

DIPLOMSKI RAD

Varaždin, 2024.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Marko Mušica

Matični broj: 0016142667

Studij: Informacijsko i programsko inženjerstvo

SUSTAVI ZA PREPORUKE U E-TRGOVINAMA

DIPLOMSKI RAD

Mentor:

doc. dr. sc. Bogdan Okreša Đurić

Varaždin, rujan 2024.

Marko Mušica

Izjava o izvornosti

Izjavljujem da je ovaj diplomski rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Rad se temelji na opisu i implementaciji sustava za preporuke na primjeru e-trgovina. Teorijski dio rada obuhvaća obradu odabranih koncepata vezanih uz algoritme strojnog učenja i umjetnu inteligenciju, pregled skupa podataka i opis primjene sustava za preporuke u stvarnom životu. Teorijski dio obuhvaća i opis korištenog modela strojnog učenja te opis korištenih alata i tehnologija za njegovu izradu i izradu web-stranice. Praktični dio sadrži prikaz i implementaciju web-stranice i sustava za preporuku kojeg će ona koristiti. Osim same web-stranice, koristi se mikroservisna arhitektura u kojoj više različitih modula svaki sa svojom specifičnom odgovornošću prenosi i obrađuje podatke.

Ključne riječi: Quarkus, Java, Preporuke, Sustavi za preporuke, React, Strojno učenje, Mikroservisi

Sadržaj

1. Uvod

E-Trgovina, najjednostavnije rečeno, predstavlja kupovinu preko interneta. Ovaj pojam obuhvaća različite kategorije proizvoda, uključujući knjige, odjeću, filmove, elektroniku, videoigre i mnoge druge. Zajednička taktika prodaje svim ovim kategorijama je sustav preporuka proizvoda.

Cilj sustava preporuka je predložiti proizvod koji bi se korisniku svidio, odnosno proizvod koji bi korisnik htio kupiti. Ova taktika prodaje postala je toliko uobičajena da se nalazi na gotovo svakoj popularnoj web-stranici.

Posebno je značajna na web-stranicama za e-trgovinu kao što su Amazon, Netflix, Temu, eBay i AliExpress. Osim usluge ovih stranica, koje korisnici plaćaju bilo pretplatom ili izravno, postoje i besplatne stranice koje koriste sustave preporuka za oglašavanje i općenito za preporuku sadržaja, poput Reddita, YouTubea i HackerOnea.

Za njihov uspjeh i profit uvelike su zaslužni sustavi preporuka koji koriste različite tehnike strojnog učenja, ali i jednostavne, ali efektivne, statističke metode za predviđanje ocjena, odnosno bodovanja proizvoda prilagođenih kupcu.

U kasnijim poglavljima obrađuju se i jednostavnije statističke metode analize podataka te nešto složenije metode strojnog učenja. Također, objašnjava se jednostavniji prikaz arhitekture koja se koristi za sustave preporuka. Na kraju je prikazan i jednostavniji primjer web-stranice na kojoj je vidljiv rad sustava za preporuku.

Izradi praktičnog dijela rada prethodi odabir skupa podataka, koji su također detaljno objašnjeni. Analizirani su različiti skupovi podataka i izazovi povezani s radom na velikim skupovima podataka.

Motivacija za ovaj rad proizlazi iz povećane, ponekad i štetne, upotrebe e-trgovina, posebno tijekom i nakon pandemije COVID-19. Cilj rada je izrada i detaljan opis praktičnog dijela u sklopu pojednostavljene mikroservisne arhitekture, koja je česta u današnjim aplikacijama i web-stranicama koje koriste sustave preporuka. Teorijski dio rada obuhvaća opis postojećih sustava preporuka i analizu koncepata potrebnih za razumijevanje rada tih sustava.

2. Metode i tehnike rada

Praktični dio ovog rada sadržava više različitih tehnologija koje sudjeluju zajedno kao jedna velika aplikacija. Ovakav način rada aplikacije naziva se mikroservisnom arhitekturom.

Poslužiteljski (*engl. Backend*) dio koristi programski jezik Java, s programskim okvirom (*engl. frameworkom*) Quarkus [1]. Osim Quarkusa, PostgreSQL [2] baza smještena je u tehnogiji Docker [3]. Korišten je i Python [4] s nekim svojim bibliotekama poput: Numpy [5], Scikit-kit learn [6], Flask [7], Keras [8]. Za sučelje (*engl. frontend*) korištena je popularna bliblioteka React [9]. Još je vrijedno napomenuti da je ovaj rad izrađen u programskom jeziku LaTeX uz pomoć alata Overleaf [10].

Istraživanja o sustavima preporuke i njihovom korištenju bila su početna točka ovoga rada, a nakon toga slijedi izrada praktičnog dijela. Nakon ostvarenja praktičnog rada sastavljen je ovaj dokument.

[1] https://quarkus.io/
[2] https://www.postgresql.org/
[3] https://www.docker.com/
[4] https://www.python.org/
[5] https://numpy.org/
[6] https://scikit-learn.org/stable/
[7] https://flask.palletsprojects.com/en/3.0.x/
[8] https://keras.io/
[9] https://react.dev/
[10] https://www.overleaf.com/

3. Umjetna inteligencija

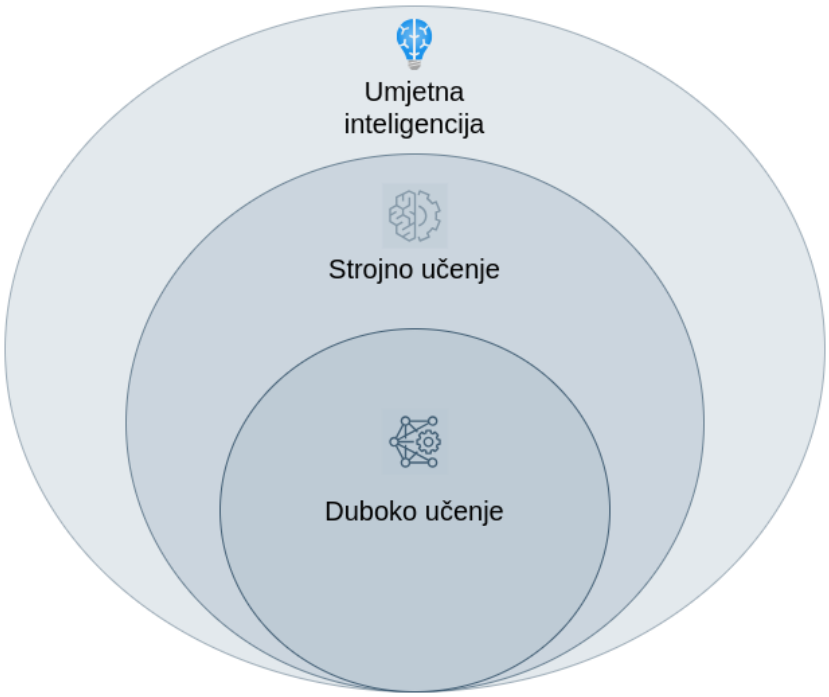
„Umjetna inteligencija, ili AI, je tehnologija koja omogućuje računalima i strojevima da simuliraju ljudsku inteligenciju i sposobnosti rješavanja problema. ” [1]

Kao polje u računalnoj znanosti, AI se često spominje zajedno sa strojnim učenjem i dubokim učenjem. Ove discipline omogućuju algoritmima da "uče" iz dostupnih podataka i s vremenom postanu sve precizniji u svojim predviđanjima. [1].

Postoje različiti načini uporabe umjetne inteligencije, a neki od njih prema [1] su:

- Prepoznavanje govora - koristi se metoda obrade prirodnog jezika (NLP) kako bi se ljudski govor pretvorio u pisani format. Mnogi mobilni uređaju koriste ovakav sustav za prepoznavanje govora, jedna od najpoznatijih implementacija je Siri.
- Korisnička podrška - odnosi se na virtualne agente i razgovorne agente koji odgovaraju na učestalo postavljena pitanja i pružaju personalizirane usluge korisnicima.
- Računalni vid - ova tehnologija omogućuje računalnim sustavima izvlačenje smislenih značenja iz slika, videa i drugih grafičkih prikaza.

Bitno je razlikovati strojno učenje, duboko učenje i umjetnu inteligenciju:



Slika 1: Razlike strojnog, dubokog učenja i umjetne inteligencije; vlastita izrada prema [2]

Na slici 1 može se vidjeti da su strojno i duboko učenje podskupovi umjetne inteligencije. Osim toga, duboko učenje je podskup strojnog učenja, a može se definirati kao: „Tehnika strojnog učenja koja koristi višeslojne neuronske mreže za obradu podataka na način koji oponaša

rad ljudskog mozga" [2]. Strojno učenje odnosi se na obuku sustava da uče i prilagođavaju se na temelju podataka (iskustva), bez potrebe za eksplicitnim programiranjem. [2]

Ukratko se može i spomenuti povijest umjetne inteligencije prema [1]:

- **1950**: Alan Turing objavljuje rad u kojem postavlja pitanje *"Mogu li strojevi misliti?"* i predstavlja poznati Turingov test za razlikovanje ljudskih i strojnih odgovora.
- **1956**: John McCarthy prvi put koristi termin *"umjetna inteligencija"* na prvoj AI konferenciji na Dartmouth Collegeu. Također, ove godine je stvoren prvi AI softverski program, *Logic Theorist*.
- **1967**: Frank Rosenblatt razvija *Mark 1 Perceptron*, prvo računalo bazirano na neuronskoj mreži koje uči na svojim greškama.
- **1980-e**: Neuronske mreže koje koriste algoritam propagacije unatrag (backpropagation) postaju široko korištene u AI aplikacijama.
- **1995**: Stuart Russell i Peter Norvig objavljuju *Artificial Intelligence: A Modern Approach*, vodeću literaturu o umjetnog inteligenciji.
- **1997**: IBM-ov *Deep Blue* pobjeđuje šahovskog prvaka Garryja Kasparova.
- **2004**: John McCarthy objavljuje rad pod nazivom *What Is Artificial Intelligence?*, koji je jedan od najviše referenciranih radova u polju umjetne intelgiencije.
- **2011**: IBM-ov *Watson* pobjeđuje prvake Kena Jenningsa i Brada Ruttera u kvizu *Jeopardy!*.
- **2015**: Baidu *Minwa* napravio je superračunalo koje koristi konvolucijsku neuronsku mrežu za prepoznavanje i kategorizaciju slika s većom točnošću od prosječnog čovjeka.
- **2016**: DeepMindov *AlphaGo*, temeljen na dubokim neuronskim mrežama, pobjeđuje svjetskog prvaka u igri Go.
- **2023**: Uspon velikih jezičnih modela (LLM-ova) poput *ChatGPT*-a označava značajan napredak u performansama umjetne inteligencije, posebno u generativnoj umjetnog inteligenciji, gdje se modeli dubokog učenja unaprijed treniraju na velikim količinama podataka.

4. Skup podataka

U sklopu ovog rada korišten je MovieLens 20M dataset [1], jedan od najpoznatijih javno dostupnih skupova podataka pogodan za sustave preporuke. Ovaj skup podataka sadrži 20 milijuna ocjena koje su korisnici dali za više od 27 tisuća filmova. Podaci su prikupljeni i objavljeni od strane GroupLens istraživačke grupe sa Sveučilišta Minnesota, a široko su korišteni u istraživanjima vezanim uz algoritme preporučivanja zbog svoje veličine i kvalitete. Skup podataka sadrži informacije poput identifikacijskog broja (ID-a) korisnika, identifikacijskog broja filma, ocjene (na ljestvici od 0.5 do 5), te vremena kada je ocjena unesena.

Inicijalno je planirano korištenje Amazonovog skupa podataka zbog njegove opsežnosti i raznolikosti (koji sadrži podatke o milijunima proizvoda i recenzija), ali zbog njegovih velikih dimenzija i kompleksnosti (s obzirom na količinu stupaca i veličinu podataka), treniranje modela i obrada podataka postali su prezahtjevni. Stoga je odlučeno da je bolje raditi s MovieLens 20M podacima, koji su prikladniji za analizu s obzirom na resurse dostupne za ovaj rad. MovieLens 20M skup podataka ima 205MB dok Amazonov "Amazon Reviews'23" [2] uključuje više skupova podataka grupiranih po kategorijama, od kojih svaki ima po nekoliko GB. Ova razlika u veličini dodatno otežava posao obrade podataka.

Kao izvor podataka korištena je javno dostupna baza podataka sa stranice Kaggle [3]. Stavke MovieLens skupa podataka su:

Ocjene (*engl. Ratings*):

- userId - jedinstveni identifikacijski broj korisnika
- movieId - jedinstveni identifikacijski broj filma
- rating - ocjena koju je korisnik dao filmu
- timestamp - vrijeme koje prikazuje trenutak kada je film ocijenjen

Filmovi (*engl. Movies*):

- movieId - jedinstveni identifikacijski broj filma
- title - naziv filma, uključujući godinu izlaska
- genres - žanrovi filma

Oznake (*engl. Tags*):

- userId - jedinstveni identifikacijski broj korisnika
- movieId - jedinstveni identifikacijski broj filma

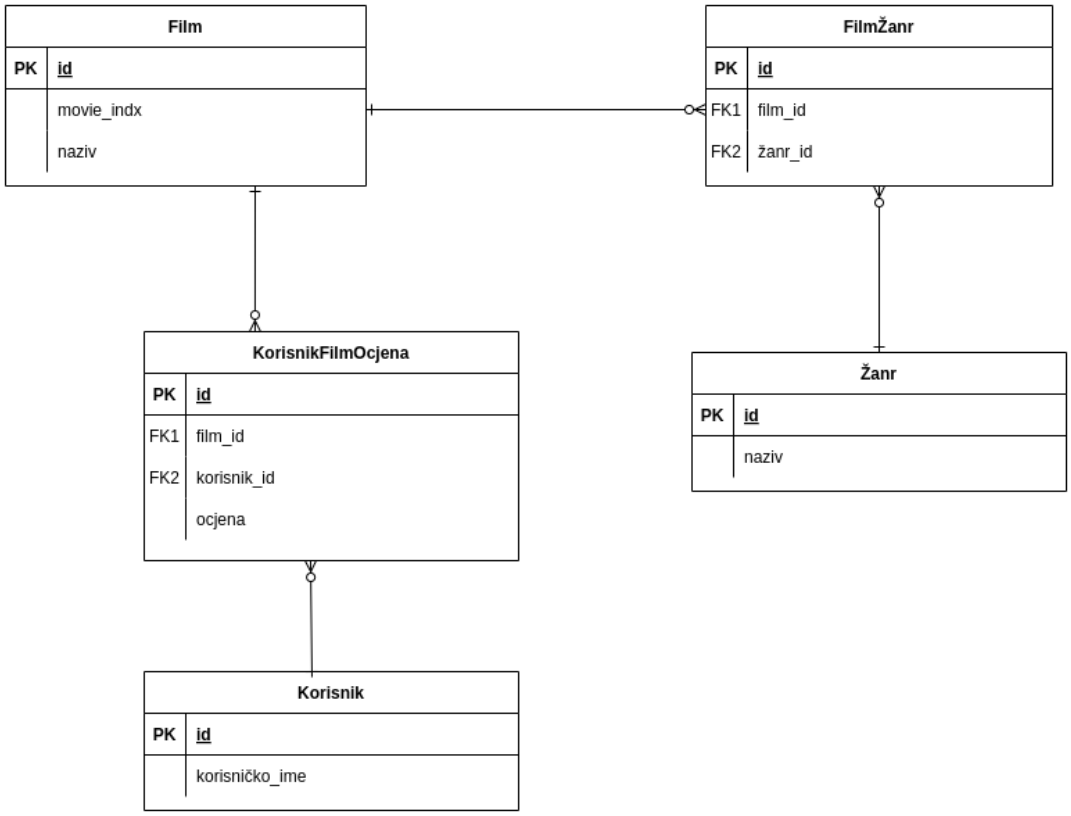
[1] https://www.kaggle.com/datasets/grouplens/movielens-20m-dataset/data
[2] https://amazon-reviews-2023.github.io/index.html
[3] https://www.kaggle.com/

- tag - oznaka koju je korisnik dao filmu
- timestamp - vrijeme koje prikazuje trenutak kada je filmu dana oznaka

Osim ovih, postoje i dodatne informacije temeljene na zajedničkim značajkama filmova, poput "Genome Scores" i "Genome Tags", ali one se nisu koristile u ovoj analizi. Najvažniji skupovi podataka korišteni u radu su Ocjene i Filmovi, koji su detaljnije obrađeni kako bi se modeli mogli učinkovito trenirati.

4.1. Priprema skupa podataka

Za pripremu skupa podataka korištena je ranije spomenuta biblioteka Pandas. Detalji implementacije nisu prikazani, no važno je napomenuti da je skup podataka smanjen i pretvoren u Python objekte, poput listi i rječnika (*engl. dictionaries*). Izabrano je tisuću korisnika s najvećim brojem ocijenjenih filmova, kao i dvije tisuće filmova s najviše ocjena. Dodatno, uklonjeni su neki nepotrebni stupci, poput vremenske oznake (*engl. timestamp*). Ovi prerađeni podaci, zajedno s imenima filmova, žanrovima i informacijama o korisnicima, pohranjeni su u PostgreSQL bazu podataka unutar Docker okruženja radi lakšeg i bržeg pristupa. Za bolje razumijevanje podatkovnog modela izrađen je ERA dijagram:



Slika 2: ERA dijagram

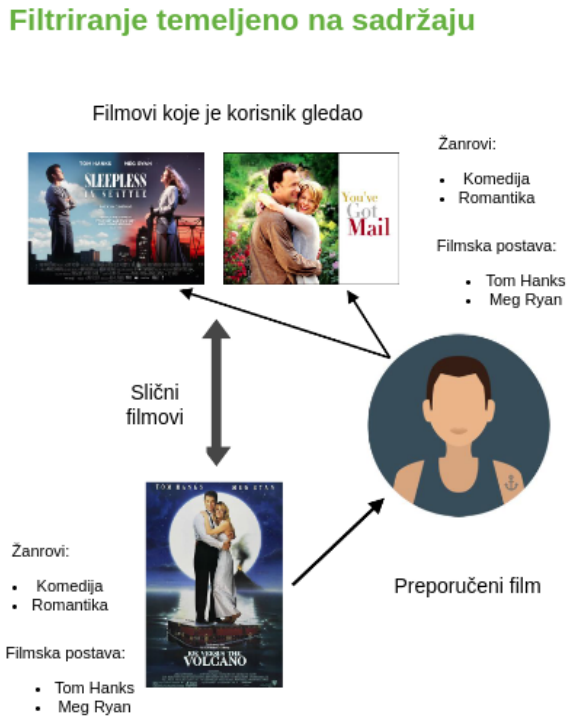
5. Sustavi za preporuke

Sustavi preporuka ključna su tehnologija modernih digitalnih platformi koji pomažu korisnicima da otkriju sadržaj ili proizvode prilagođene njihovim željama. Ovi sustavi analiziraju obrasce ponašanja korisnika i koriste te informacije za stvaranje personaliziranih preporuka, čime se povećava zadovoljstvo korisnika i angažman na platformi.

5.1. Općenito o sustavima za preporuke

Postoje tri glavne vrste preporuka sustava: sustavi temeljeni na filtriranju sadržaja, sustavi temeljeni na kolaborativnom filtriranju i hibridni sustavi. [3]

Filtriranje sadržaja koristi sličnosti između stavki koje korisnik već voli i novih stavki kako bi se preporučili novi proizvodi. Ovi sustavi analiziraju atribute stavki, poput žanra filma ili teme knjige, kako bi generirali preporuke [4]. U slici 3, sustav koristi žanrove filmova i glumce kako bi pronašao dobru preporuku filma.



Slika 3: Filtriranje prema sadržaju; prema [3]

S druge strane, kolaborativno filtriranje temelji se na povijesti interakcija korisnika sa stavkama i koristi sličnosti između korisnika ili stavki kako bi se napravile preporuke. Ovaj pristup posebno je učinkovit jer može otkriti nove stavke koje slični korisnici vole, ali koje korisnik još nije otkrio. Prikaz ovakve logike vidljiv je u slici 4, gdje korisnica i korisnik imaju slične interese prema kojima se dobiva predviđeni film. [3]

Slika 4: Kolaborativno filtriranje; prema [3]

Hibridni sustavi preporuke kombiniraju prednosti u oba pristupa kako bi postigli bolja predviđanja i smanjili nedostatke konvencionalnih sustava za preporučivanje. [3]

Uz svoju korisnost, sustavi preporuka suočavaju se s nekoliko izazova [5]. Jedan od ključnih problema je "hladni start", gdje je teško preporučiti nove stavke ili korisnike koji nemaju dovoljno podataka. Drugi izazov je osiguranje privatnosti korisnika, posebno u sustavima koji zahtijevaju velike količine podataka za personalizaciju.

S porastom popularnosti neuronskih mreža, njihova primjena proširila se i na sustave preporuke. Korištenjem umjetnih neuronskih mreža, platforme su u stanju obrađivati ogromne količine podataka o korisnicima koje se često prikupljaju i pohranjuju. Ti podaci zatim služe za treniranje modela neuronskih mreža, koji mogu ponuditi personalizirane preporuke. Neuronske mreže omogućuju veću fleksibilnost i učinkovitije procesiranje velike količine podataka, što rezultira preciznijim i relevantnijim preporukama za korisnike. [6]

U konačnici, sustavi preporuka ključni su alat za personalizaciju digitalnih iskustava, koji korisnicima omogućuje otkrivanje relevantnih sadržaja na način koji je prilagođen njihovim jedinstvenim ukusima.

5.2. Sustavi za preporuke u e-trgovinama i popularnim web-stranicama

Primjene sustava za preporuke su raznolike i uključuju internetske trgovine poput Amazona, gdje se preporučuju proizvodi temeljeni na prethodnim kupovinama, ocjenama korisnika,

pregledanim prozvodima i pretraživanim pojmovima. Također, sustavi za preporuke koriste se i na streaming platformama kao što su Netflix i Spotify, na kojima se korisnicima nude personalizirani prijedlozi filmova, serija i glazbe.

Amazon koristi mnoge od ranije spomenutih metoda [7]: kolaborativno filtriranje bazirano na korisnicima i na stavkama, filtriranje temeljno na sadržaju, obrada prirodnog jezika (NLP), matričnu faktorizaciju, duboko učenje i hibridne pristupe.

Jedna od najčešće korištenih metoda na Amazonu je kolaborativno filtriranje temeljeno na stavkama. Ovaj algoritam preporučivanja analizira nedavne kupnje korisnika i stvara popise povezanih proizvoda [8]. Takav pristup često dovodi do preciznijih i relevantnijih prijedloga u usporedbi s algoritmima kolaborativnog filtriranja temeljenima na korisnicima.

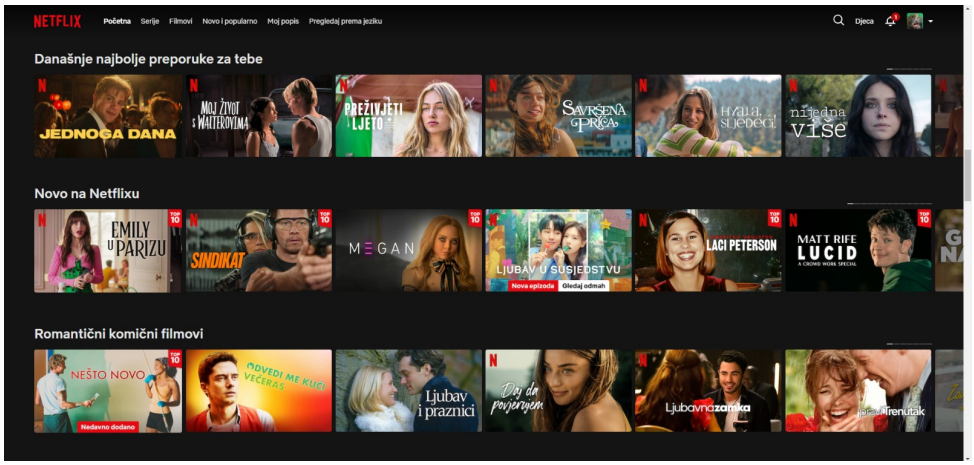
Osim Amazona, zanimljive načine preporučivanja koristi Netlfix. Netflix vjeruje da bi bez svog sustava preporuke gubio milijardu dolara svake godine. [9]

> „Netflixova uspješnost leži u vrlo inteligentnom algoritmu preporuka, poznatom kao Netflixov sustav preporuke (*engl. Netflix Recommendation Algorithm*) (NRE). NRE sastoji se od više algoritama koji filtriraju sadržaj na temelju korisničkog profila. Sustav filtrira preko 3.000 naslova koristeći 1.300 preporučnih klastera, sve u skladu s preferencijama pojedinog korisnika. Ovaj pristup se pokazao vrlo učinkovitim jer 80% aktivnosti gledatelja na Netflixu dolazi iz osobnih preporuka, što Netflixu godišnje štedi milijarde dolara jer zadržava pretplatnike od otkazivanja usluge.” [9]

Netflix izbjegava problem "hladnog starta" korištenjem tehnike "jump start", koja pri prvoj prijavi od korisnika traži informacije o omiljenim žanrovima serija i filmova. Osim toga, prikuplja se velik broj podataka kako bi se poboljšale preporuke. Neki od tih podataka, prema [9], uključuju:

- Koliko dugo je video gledan
- Povijest gledanja
- Ocjene filmova i serija
- Podaci o drugim korisnicima sa sličnim ukusima
- Doba dana u kojem je sadržaj gledan
- Koliko puta je sadržaj pauziran, premotan ili ubrzan
- Kada je serija ili film isključen

Izgled Netlfixovog sustava za preporuku može se vidjeti u slici 5:



Slika 5: Netflixove preporuke - slika zaslona

Osim Netflixovih filmova i Amazonove web-kupovine, postoji i sustav preporuka glazbe na Spotifyju [10]. Sustav preporuka na Spotifyju koristi složene algoritme kako bi personalizirao glazbena iskustva svojih korisnika. Ključni dio ovog sustava je analiza korisničkih navika, uključujući pjesme koje slušaju, popise pjesama koje kreiraju i njihove glazbene žanrove. Spotify primjenjuje različite tehnike za preporučivanje glazbe, među kojima se izdvaja filtriranje temeljeno na sadržaju, koje analizira karakteristike pjesama kao što su tempo i ton, te preporučuje slične pjesme. Također, koristi filtriranje temeljeno na korisniku, koje predlaže pjesme slične onima koje su slušali korisnici sa sličnim ukusom. Kolaborativno filtriranje dodatno unapređuje preporuke koristeći podatke o ponašanju drugih korisnika sličnih ukusa.

Skupa s kolaborativnim filtriranjem i filtriranjem temeljenim na sadržaju, koristi se i obrada prirodnog jezika (NLP) za izdvajanje semantičkih informacija iz teksta pjesama, koje se također koriste za preporuke. Sustav je dizajniran da stalno uči i prilagođava se na osnovu novih korisničkih podataka, što omogućava kontinuirano poboljšanje točnosti preporuka. [10]

6. Metode sustava preporuke

6.1. Kolaborativno filtriranje

> „Kolaborativno filtriranje je metoda preporuke koja koristi interakcije korisnika s različitim stavkama kako bi predložila stavke drugim korisnicima koji imaju slične preference i ponašanje. Drugim riječima, algoritmi kolaborativnog filtriranja grupiraju korisnike na temelju njihovog ponašanja i koriste zajedničke karakteristike grupe kako bi preporučili stavke korisnicima.” [11]

Ova metoda temelji se na pretpostavci da će korisnici koji su pokazali slične interese i ukuse u prošlosti imati slične interese i u budućnosti. Na primjer, ako dva korisnika imaju slične ocjene za određene filmove, postoji velika vjerojatnost da će obojica biti zainteresirani za filmove koje je jedan od njih ocijenio visoko, a drugi još nije otkrio.

Kako navodi [12] postoje dvije vrste kolaborativnog filtiranja:

- Filtriranje temeljeno na memoriji
- Filtriranje temeljeno na modelu

„Sustavi preporuka temeljeni na memoriji ili sustavi temeljeni na susjedima, su proširenje klasifikatora k-najbližih susjeda jer nastoje predvidjeti ponašanje korisnika prema određenoj stavci na temelju sličnih korisnika ili skupa stavki.” [12]

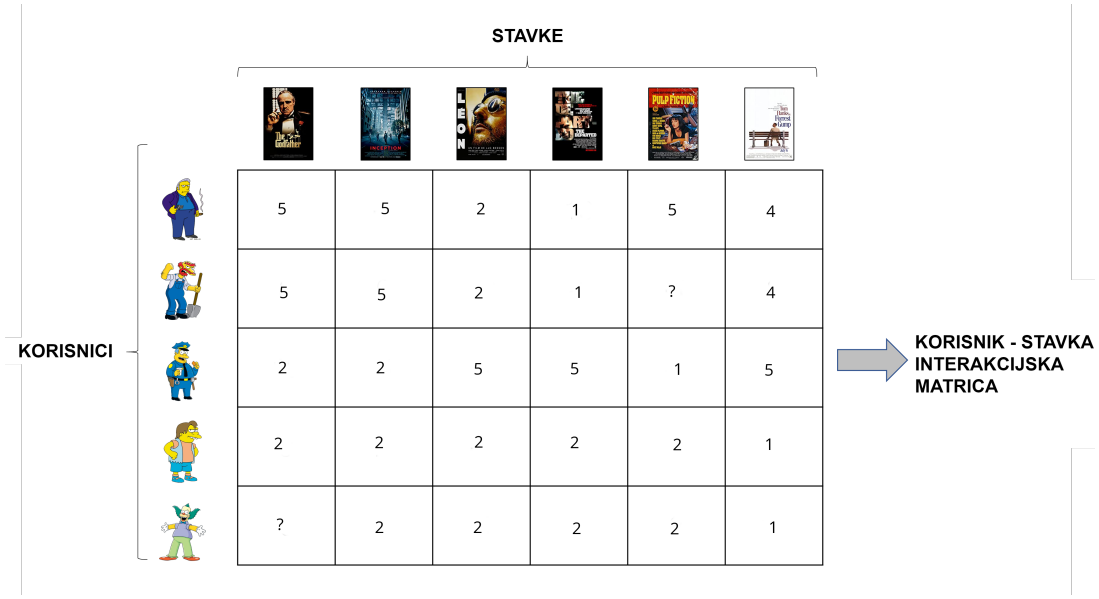
Prema [13] postoje dvije glavne podjele kolaborativnog filtriranja temeljenog na memoriji:

- Filtriranje temeljeno na korisnicima
- Filtriranje temeljeno na stavkama

6.1.1. Filtriranje temeljeno na korisnicima

Korisničko kolaborativno filtriranje bavi se sličnošću među korisnicima temeljenim na njihovom ocjenjivanju neke stavke. Jedan od pristupa unutar ove metode izračunava predviđenu ocjenu pojedinog korisnika za svaku stavku tako što uzme ponderirani prosjek ocjena neke stavke ocijenjene od strane sličnih korisnika gdje ponder (težinu) predstavlja sličnost korisnika. [14]

Ovakvu metodu najbolje je objasniti grafičkim putem:



Slika 6: Matrica kolaborativnog filtriranja temeljnog na korisnicima [15]

Na slici 6 može se vidjeti nekoliko korisnika i filmova, uz nekoliko praznih mjesta označenih upitnicima. Na tim mjestima, koristeći prethodne podatke, treba donijeti zaključak o potencijalnoj ocjeni. Pretpostavlja se da se slika čita s gornje strane prema donjoj i s lijeva na desno, te se pokušava predvidjeti ocjena za petu stavku drugog korisnika.

Uspoređivanjem ocjena prvog i drugog korisnika, može se primijetiti da su njihove ocjene identične za sve stavke koje su ocijenili. Na temelju ovog, može se zaključiti da će predviđena ocjena za upitnik biti 5, budući da je to ocjena koju je dao korisnik najviše sličan korisniku za kojeg se predviđa ocjena. Isto tako, uspoređujući podatke s posljednjim korisnikom, može se predvidjeti da će njegova ocjena biti 2. Iako je ovo vrlo pojednostavljen primjer, upravo na ovom principu funkcionira korisničko kolaborativno filtriranje.

Pravi način na koji se može predvidjeti ocjenu korisnika uključuje par formula:

Prosjek ocjena [14]:

$$s(i,j) = \frac{\sum_{i' \in \Omega_j} r_{i'j}}{|\Omega_j|}$$

$s(i,j)$ - suma svih ocjena za stavku j podijeljena sa brojem ukupnih ocjena
$r_{i'j}$ - ocjena koju je korisnik i' dao stavki j
Ωj - skup svih korisnika i' koji su ocijenili stavku j (broj ukupnih ocjena)

Jednostavno rečeno, prosjek ocjena predstavlja sumu svih ocjena podijeljenu s brojem ukupnih ocjena. Ova metoda ima dva velika nedostatka. Prvi je pristranost korisnika, što znači da ocjena 3 korisnika A nije jednaka ocjeni 3 korisnika B. Nekome ocjena 3 znači da je film bio relativno dobar, dok za nekoga drugoga ocjena 3 znači da je film bio užasan. Drugi nedostatak je manjak pondera, odnosno ne uvažavamo sličnosti različitih korisnika. Svačije

mišljenje smatra se jednako važnim, što je netočno jer su neki korisnici sličniji promatranom korisniku od drugih.

Rješenje prvog problema je korištenje devijacija [16]. Ideja je takva da se ne mari o apsolutnim ocjenama nego o tome koliko je odstupanje od prosjeka.

$$dev(i,j) = r(i,j) - \overline{r_i}$$

$r(i,j)$ - ocjena koju je korisnik i dao stavki j
$\overline{r_i}$ - prosječna ocjena korisnika i

Da bi se dobio prosjek odstupanja svih korisnika mora se izračunati suma odstupanja svih korisnika i podijeliti s brojem ocjena.

$$\hat{dev}(i,j) = \frac{\sum_{i' \in \Omega_j} r(i',j) - \overline{r'_i}}{|\Omega_j|}$$

S ovakvom generalnom idejom ocjena se može predvidjeti kao prosječna ocjena zbrojena s prosječnim odstupanjem svih ocjena.

$$s(i,j) = \overline{r_i} + \hat{dev}(i,j)$$

Naravno, formula još nije potpuna jer se ne uračunavaju sličnosti s drugim korisnicima.

$$s(i,j) = \overline{r_i} + \frac{\sum_{i' \in \Omega_j} W_{ii'}\{r_{i'j} - \overline{r_{i'}}\}}{\sum_{i' \in \Omega_j} |W_{ii'}|}$$

Ovakva formula sada uvažava sličnost korisnika [17]. Pitanje je kako izračunati ovaj ponder W? Koristi se Pearsonov koeficijent korelacije [18]:

$$pcc(u,u') = \frac{\sum_{i \in I}(r_{u,i} - \overline{r}_u)(r_{u',i} - \overline{r}_{u'})}{\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r}_u)^2}\sqrt{\sum_{i \in I}(r_{u,i} - \overline{r}_{u'})^2}}$$

Prema [18] ova formula se može iskoristiti s trenutnim vrijednostima:

$$W_{ii'} = \frac{\sum_{j \in \eta_{ii'}}(r_{ij} - \overline{r_i})(r_{i'j} - \overline{r'_i})}{\sqrt{\sum_{j \in \eta_{ii'}}(r_{ij} - \overline{r}_i)^2}\sqrt{\sum_{j \in \eta_{ii'}}(r_{i'j} - \overline{r}_{i'})^2}}$$

Gdje su:

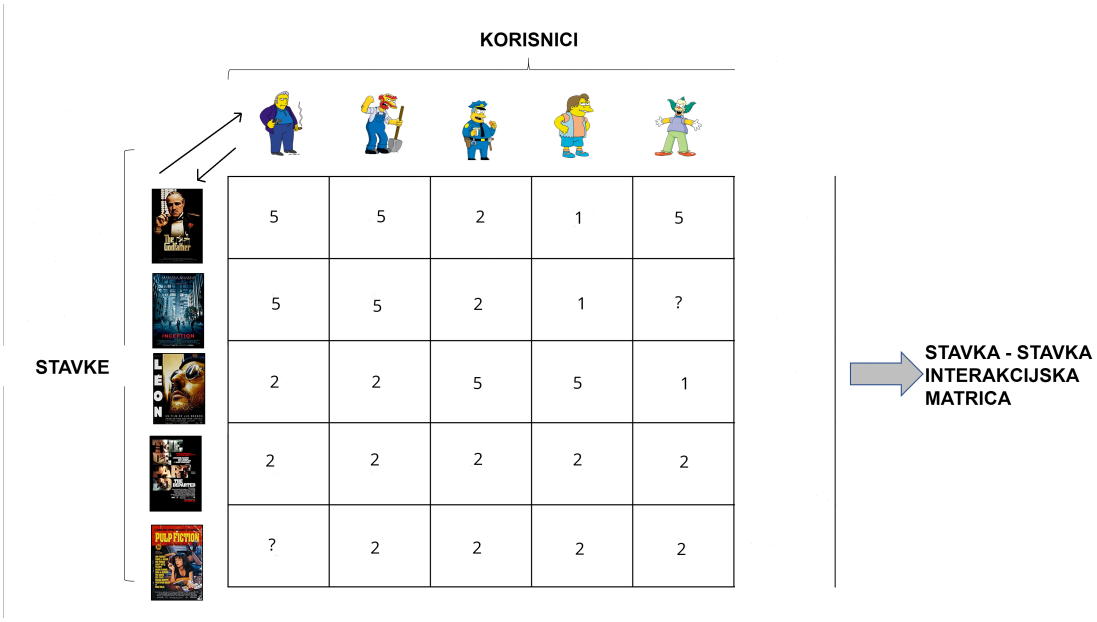
$\eta_{ii'}$ - skup filmova koji su korisnik i i i' ocijenili odnosno presjek filmova koje je ocijenio korisnik i' i filmova koje je ocijenio korisnik i

Sada se konačno dobiva finalna formula koja se može koristiti za predviđanja. Problem koji se javlja s ovom formulom je količina korisnika. Naime, nepotrebno je usporediti baš svakog korisnika koji je ocijenio neki pojedinačni film, ovakav pristup će predugo trajati. Najbolje je uzeti *K najbližih susjeda*. Uzima se *K* broj susjeda s najvećim težinama, odnosno korisnici koji su najsličniji.

6.1.2. Filtriranje temeljeno na stavkama

„Umjesto uparivanja korisnika sa sličnim kupcima, kolaborativno filtriranje temeljeno na stavkama povezuje svaku korisnikovu kupljenu i ocijenjenu stavku sa sličnim stavkama, zatim kombinira te slične stavke u listu preporuka." [19]

Drugim riječima, ako se filtriranje temeljeno na korisnicima definira kao mjerenje koliko se neka stavka sviđa određenom korisniku, onda se filtriranje temeljeno na stavkama može definirati kao mjerenje koliko se neki korisnik sviđa određenoj stavci.



Slika 7: Matrica kolaborativnog filtriranja temeljnog na stavkama [15]

Na slici 7 prikazana je inverzija prethodne matrice. Sada se analiziraju sličnosti između dvije stavke. Vidi se da su prvi i drugi film vrlo slični, što ukazuje da bi njihove ocjene trebale biti slične. Na temelju ove logike, može se zaključiti da upitnik u drugom redu odgovara ocjeni 5, dok upitnik u zadnjem redu odgovara ocjeni 2. Ako slični korisnici ocjenjuju prvi film na sličan način kao i drugi film, može se pretpostaviti da će filmovi dobiti slične ocjene od korisnika odnosno da će korisnici dobiti slične ocjene od filmova.

Formula izgleda isto, ali prilagođena prema stavkama [18]:

$$W_{jj'} = \frac{\sum_{i \in \eta_{jj'}} (r_{ij} - \overline{r_j})(r_{i'j} - \overline{r'_j})}{\sqrt{\sum_{i \in \eta_{jj'}} (r_{ij} - \overline{r}_j)^2} \sqrt{\sum_{i \in \eta_{jj'}} (r_{i'j} - \overline{r}_{j'})^2}}$$

Gdje su:

$\eta_{jj'}$ - skup korisnika koji su filmovi j i j' ocijenili odnosno presjek korisnika koje je ocijenio film j' i korisnika koje je ocijenio film j

$\overline{r_j}$ - prosječna ocjena svih korisnika (filmovi ocjenjuju korisnike)

6.2. Matrična faktorizacija

Matrična faktorizacija jedan je od najvažnijih modela u sustavima za preporuke. Model je odličan jer omogućuje razradu velikih podataka pomoću dekompozicije na manje matrice koje se koriste za izradu preporuke.

Ovom tehnikom originalna matrica može se dekomponirati na više manjih matrica. Originalna matrica R u ovom slučaju predstavlja ocjene za filmove, svaki redak predstavlja korisnika, a svaki stupac predstavlja film. Ako korisnik u ocjenjuje film i ocjenom r_{ui}, tada je cilj predvidjeti takvu ocjenu.

Metoda koja se koristi zove se ALS - Metoda najmanjih kvadrata (*engl. Alternating Least Squares*) i detaljnije je objašnjena nakon prikaza formule.

Generalna formula koja se najviše spominje u literaturi [20]:

$$\hat{r}_{ui} = q_i^T p_u.$$

> „Model matrične faktorizacije modelira korisnike i artikle u faktor prostor dimenzionalnosti f, takav da su interakcije između korisnika i artikla modelirane kao produkt u tom prostoru. Svaki artikl i povezan je s vektorom $q_i \in \mathbb{R}^f$, a svaki korisnik u povezan s vektorom $p_i \in \mathbb{R}^f$. Za određeni artikl i, elementi vektora q_i označavaju u kojoj mjeri artikl posjeduje željene faktore, pozitivne ili negativne. Za određenog korisnika u_i, elementi vektora p_u mjere stupanj interesa korisnika za artikle koji odgovaraju željenim faktorima. Produkt $q_i^T p_u$ predstavlja interakciju između korisnika u i artikla i. Ovakav produkt je na kraju označen kao r_{ui}, procjena interesa korisnika za karakteristiku artikla.” [20]

Ideja je dobiti matricu predviđenih ocjena $\hat{R}$ množenjem dviju manjih matrica W i U^T. Ovo se može jednostavnije izraziti kao:

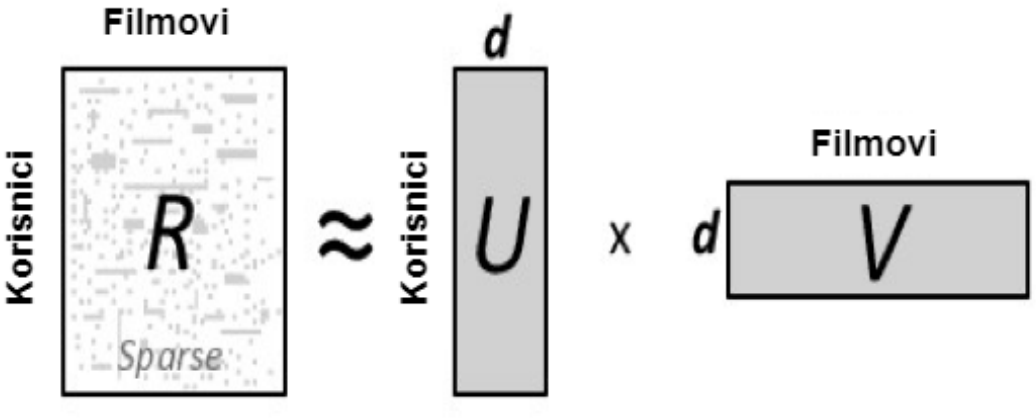
$$\hat{R} = WU^T$$

Gdje:
$\hat{R}$ - matrica procjena
W - matrica korisnika

U^T - transponirana matrica filmova

Nedostatak ove formule je množenje cijelih matrica. Ove matrice imaju previše podataka za pohraniti u memoriju računala efikasno. S ovime na umu, koristi se formula spomenuta u [20] kako bi se dobila procjena samo jednog filma za određenog korisnika.

Prikaz opisanog koncepta kao slike:



Slika 8: Matrična faktorizacija - preuzeto s [21]

Nakon slike 8 i ovih osnovnih formula može se interpretirati model. Matrice W i U^T vektori su veličine $W = N \times K$ i $U^T = M \times K$. Gdje je K - gore spomenuti faktor u matrici. Faktori zapravo predstavljaju svojstva filmova ili korisnika.

Pretpostavlja se da je $K = 3$ gdje su faktori:

- Akcija
- Komedija
- Animacija

$w_i(1)$ predstavlja koliko korisnik i voli akciju. $u_i(1)$ predstavlja koliko film j sadržava akcije. $w_i(3)$ predstavlja koliko korisnik i voli animaciju. $u_i(3)$ predstavlja koliko film j sadržava animacije.

Ako se uzme skalarni produkt ovih vektora, dobiva se rezultat koji kaže koliko korisniku i odgovaraju svojstva filma j.

Objašnjeno na primjeru matrica, dolje slijedi prikaz matrice w_i

$$\begin{bmatrix} 1 & -1 & 0.8 \end{bmatrix}$$

Ovakva matrica znači da korisnik i obožava akciju $w_i(1) = 1$, mrzi komedije $w_i(2) = -1$ i jako mu se sviđaju animacije $w_i(3) = 0.8$.

Matrica za film u_i izgleda:

$$\begin{bmatrix} 1 \\ -0.2 \\ 1 \end{bmatrix}$$

Ovo znači da film u sadržava puno akcije $u_i(1) = 1$, ne sadržava puno komedije $u_i(2) = -0.2$ i da je film animacija $u_i(3) = 1$.

Ako se uzme skalarni produkt ove dvije matrice:

$$rezultat = 1 \times 1 + -1 \times -0.2 + 0.8 \times 1 = 2$$

Dobiven je pozitivan rezultat što znači da se korisniku uvelike sviđa ovakav film što i ima smisla jer se njegova svojstva preklapaju s ukusima korisnika. U slučaju da je rezultat negativan, ovo bi značilo da se korisniku film ne sviđa. Važno je shvatiti da ne postoji utjecaj na ove faktore, ovo su latentni (skriveni) faktori. Tek kroz interpretaciju rezultata može se pokušati razumjeti i identificirati ove faktore.

Ako su ovi faktori poznati, može se koristiti nadzirano strojno učenje. Međutim, budući da ne postoji znanje o ovim faktorima, matrična faktorizacija ih samostalno pronalazi koristeći samo ocjene filmova.

U poglavlju o kolaborativnom filtriranju već su se obradili detaljni izvodi i objašnjenja formula. Kako se ovaj rad primarno fokusira na programiranje i implementaciju sustava za preporuke, izostavljaju se detalji matematičkog izvoda. Umjesto toga, prikazane su konačne formule koje se koriste.

Prvo se mora dokazati da kreirani model ima dobra predviđanja - za ovo se koristi suma kvadratnih razlika (pogrešaka) [22]:

$$\sum_{i,j \in \Omega} (r_{ij} - \hat{r_{ij}})^2 = \sum_{i,j \in \Omega} (r_{ij} - w_i^T u_j)^2$$

Gdje su:
w - korisnici
u - filmovi
r_{ij} - ocjena korisnika i za film j

Derivacijom se može izvući w_i i u_j [20]:

$$w_i = (\sum_{j \in \Psi_i} u_j u_j^T)^{-1} \sum_{j \in \Psi_i} r_{ij} u_j$$

$$u_j = (\sum_{i \in \Omega_j} w_i w_i^T)^{-1} \sum_{i \in \Omega_j} r_{ij} w_i$$

Kako bi model bio precizniji, mora se uvažiti i pristranost [20]:

$$\hat{r_{ij}} = w_i^T u_j + b_i + c_j + \mu$$

Gdje:
b_i - pristranost korisnika
c_j - pristranost filmova
μ - globalni prosjek

Ako se uvrsti novi r_{ij} u formulu sa sumom kvadratnih pogrešaka dobiva se [20]-[21]:

$$w_i = (\sum_{j \in \Psi_i} u_j u_j^T)^{-1} \sum_{j \in \Psi_i} (r_{ij} - b_i - c_j - \mu) u_j$$

$$u_j = (\sum_{i \in \Omega_j} w_i w_i^T)^{-1} \sum_{i \in \Omega_j} (r_{ij} - b_i - c_j - \mu) w_i$$

Na ove dvije formule može se vidjeti da w_i ovisi o u_i i obrnuto. U kodu započinje se s iniijalizacijom w_i i u_i nasumičnim vrijednostima. Nakon ovoga u svakoj iteraciji popunjavaju se sve w_i i onda sve u_i tako da postoji najmanja suma kvadratnih pogrešaka. Iz ovog alternirajućeg popunjavanja vrijednosti s ciljem najmanje sume kvadratnih pogrešaka dolazi naziv metode najmanjih kvadrata (*engl. Alternating least squares*) [21]-[23]

Sada se još izvode b_i i c_j [20]-[21]:

$$b_i = \frac{1}{|\Psi_i|} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - c_j - \mu) u_j$$

$$c_j = \frac{1}{|\Omega_j|} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - b_i - \mu) w_i$$

Model je trenutno suma matrice korisničkih faktora i matrice filmskih faktora te zbroj sa pristranošću korisnika, filmova i globalnim prosjekom.

Zadnji korak je regularizacija koja je potrebna jer ALS metoda ima problem s prekomjernim prilagođavanjem (*engl. overfitting*) na podatake što može rezultirati gorim predviđanjima na novim, nepoznatim podacima. Metoda regularizacije koju ćemo koristiti naziva se L2 penalizacija, točnije zatvorena forma L2 penalizacije. [20]-[21]-[23]-[24]

$$J = \sum_{i=1}^{N} (y_i - \hat{y_i})^2 + \lambda ||w||_2^2$$

Kao što je ranije spomenuto, detalji ovakvih izvoda su izostavljeni jer primarni fokus rada nije na matematičkim izvodima i derivacijama. Za detaljna objašnjenja i izvode preporuča se

pregledati literaturu [20]-[23]-[21].

Konačne formule za w_i, u_j, b_i, c_j su:

$$w_i = (\sum_{j \in \Psi_i} u_j u_j^T + \lambda I)^{-1} \sum_{j \in \Psi_i} (r_{ij} - b_i - c_j - \mu) u_j$$

$$u_j = (\sum_{i \in \Omega_j} w_i w_i^T + \lambda I)^{-1} \sum_{i \in \Omega_j} (r_{ij} - b_i - c_j - \mu) w_i$$

$$b_i = \frac{1}{|\Psi_i| + \lambda} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - c_j - \mu)$$

$$c_j = \frac{1}{|\Omega_j| + \lambda} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - b_i - \mu)$$

Kao i do sada, sve prijašnje formule uzete su od [20]-[21]. Sa svim formulama može se početi s implementacijom koja je prikzana u kasnijem poglavlju. Zašto su ove formule korisne?

Zaključno, ove formule izvedene su tako da se svaku iteraciju dobivaju bolja rješenja za matrice w_i i u_j čime s većim brojem iteracija smanjuje srednja kvadratnu pogrešku (MSE) koristeći metodu alternirajućih najmanjih kvadrata (ALS). Manjom srednjom kvadratnom greškom se zatim postižu bolji rezultati u predviđanjima.

6.3. Duboko učenje - umjetne neuronske mreže

Duboko učenje podskup je strojnog učenja koje koristi višeslojne neuronske mreže, koje se nazivaju dubokim neuronskim mrežama, za simulaciju donošenja odluka ljudskog mozga. Većinu aplikacija umjetne inteligencije u današnjici pokreće neki oblik dubokog učenja. [25]

> „Dok modeli nadziranog učenja zahtijevaju strukturirane, označene ulazne podatke za točne rezultate, modeli dubokog učenja mogu koristiti učenje bez nadzora. S nenadziranim učenjem, modeli dubinskog učenja mogu izdvojiti karakteristike, značajke i odnose koji su im potrebni za točne rezultate iz sirovih, nestrukturiranih podataka. Osim toga, ovi modeli mogu čak procijeniti i poboljšati svoje rezultate za veću preciznost.” [25]

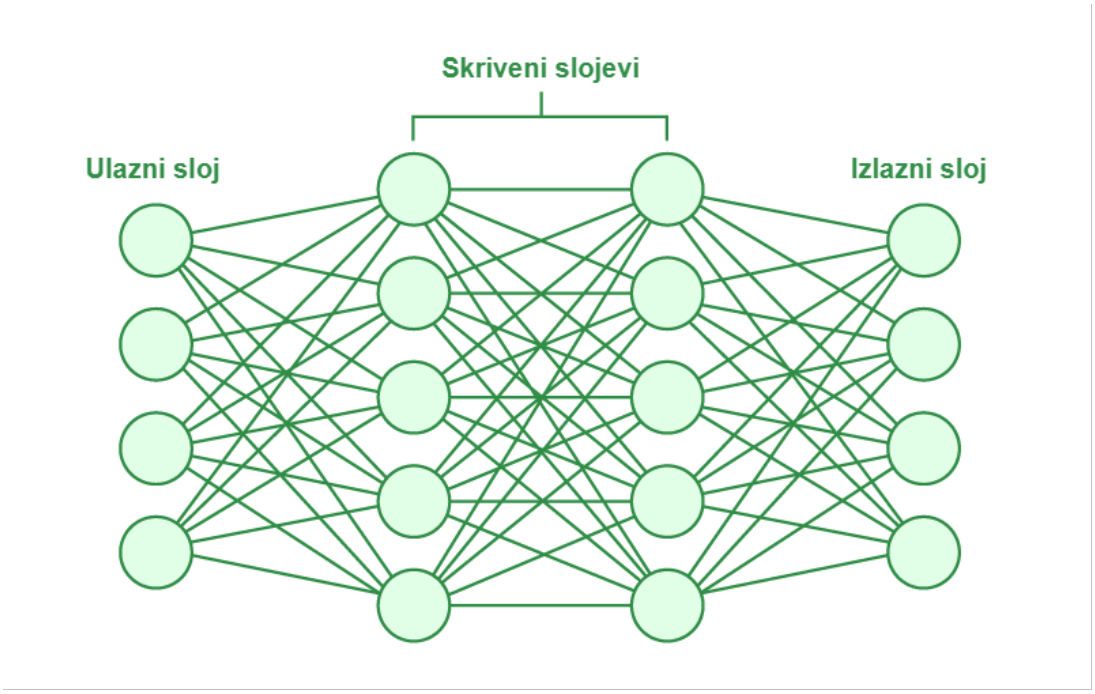
Nenadzirano učenje odlično se uklapa u slučaj gdje se svakom korisniku sviđa neki film, a nije poznato koja su svojstva korisnika ili filma. Ovakva skrivena svojstva ranije su spomenuta kao latentni faktori. Model dubokog učenja koji se koristi za predviđanja ocjene filmovi naziva se umjetna neuronska mreža.

Umjetne neuronske mreže (*engl. Atificial Neural Network*) prema [26] su računalni modeli inspirirani strukturom i funkcijama ljudskog mozga. One se koriste za prepoznavanje obrazaca, klasifikaciju i predviđanje podataka. Osnovna jedinica umjetnih neuronskih mreža je čvor

(neuron), koji prima ulaze, obrađuje ih kroz aktivacijsku funkciju, i proizvodi izlaz.

> „Umjetna neuronska mreža sastoji se od velikog broja procesnih elemenata s njihovim vezama, a ima tri karakteristična sloja, naime ulazni, skriveni i izlazni sloj. Ti slojevi nazivaju se osnovnim elementima arhitekture i poznati su kao čvorovi/neuron. Povezivanje čvorova ostvaruje se preko sinapsi, a svaki čvor ima faktor težine. U dizajnu umjetne neuronske mreže, signali se prenose kroz neurone, a istovremeno se mijenjaju težine i funkcije prijenosa, a ovaj proces se ponavlja sve dok se ne postigne zadovoljavajući izlaz. Broj neurona u svakom sloju određuje se ovisno o strukturi problema.” [26]

Primjer ovakvog modela dubokog učenja može se vidjeti u slici 9:



Slika 9: Duboke neuronske mreže - preuzeto s [27]

Na slici je prikazan jedan ulazni sloj, dva skrivena sloja i jedan izlazni sloj. Svaki čvor ima više veza, a svaka od ovih veza ima neku pridruženu težinu. Svaka od ovih težina predstavlja utjecaj jednog čvora na drugi. Kako neuronska mreža iterira kroz epohe (sveukupan prolazak kroz dostupne podatke) tako postaje bolja u predviđanjima i prilagođuje težine svake veze pomoću različitih algoritama. Na kraju se nalazi par izlaznih čvoreva koji predstavljaju rezultate predviđanja neuronske mreže.[26].

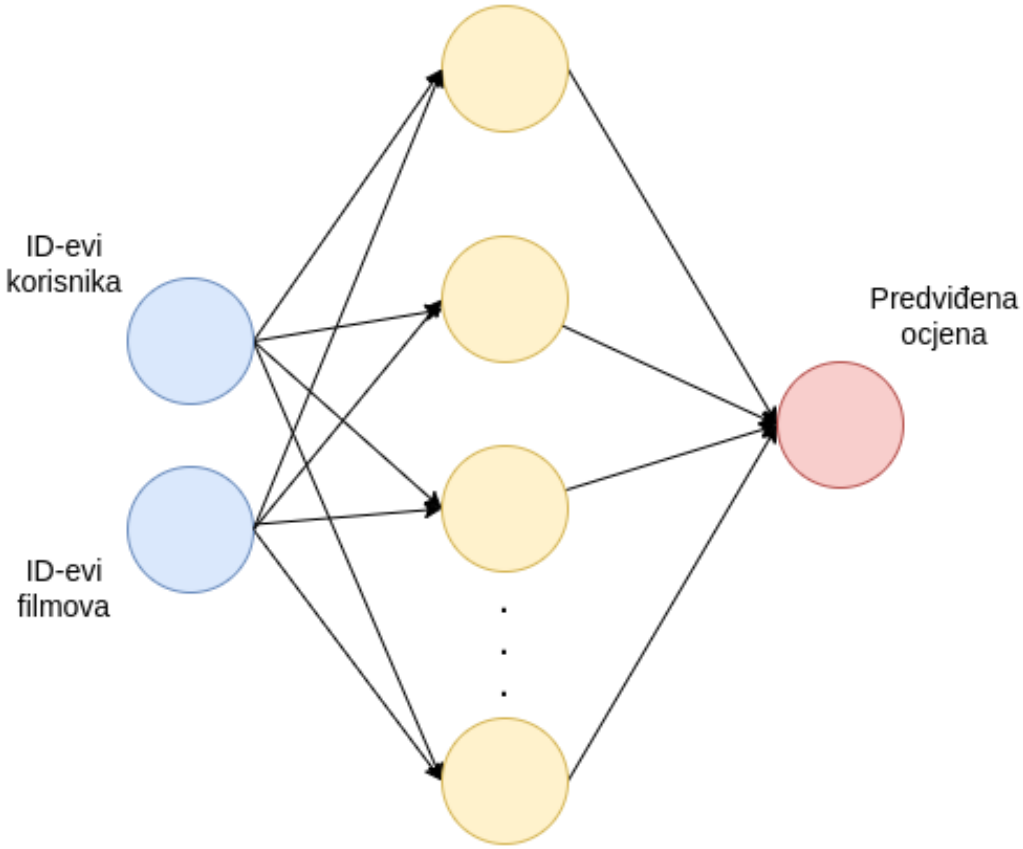
Algoritam koji neuronske mreže koriste kako bi smanjile pogreške kroz epohe nazivaju se algoritmima gradijentnog spusta.

> „Algoritam gradijentnog spusta je algoritam koji numerički procjenjuje gdje funkcija postiže svoje najniže vrijednosti. To znači da pronalazi lokalne minimume, ali ne

tako da postavi $\nabla f = 0$. Umjesto da pronalazi minimume manipuliranjem simbolima, gradijentni spust približava rješenje pomoću brojeva.” [28]

Algoritam koji se koristi kako bi se pronašao minimum grešaka je *Adam*. *Adam* je vrsta optimizacijskog algoritma koja služi za prilagođavanje težina veza neuronske mreže. [29]

Prikaz neuronske mreže koja je bliže onome što se koristi u implementaciji:



Slika 10: Neuronska mreža - vlastita izrada prema uzoru na [27]

Za ulazni sloj uzeti su identifikacijski brojevi korisnika i filmova. Ovo su kategoričke varijable koje se moraju dodatno obrađivati prije nego što se proslijede u model. Nakon prosljeđivanja, postoji skriveni sloj i izlazni sloj. U izlaznom sloju je samo jedan čvor - rezultat predviđanja. O detaljima implementacije priča se detaljnije u poglavlju implementacije modela neuronskih mreža.

Vrijedno je spomenuti razlike između strojnog učenja i dubokog učenja. Razlike su prikazane u tablici 1:

Tablica 1: Prikaz razlika između strojnog i dubokog učenja; prema [2]

Strojno učenje	Duboko učenje
Podskup umjetne inteligencije	Podskup strojnog učenja
Može biti treniran na manjem podskupu podataka	Zahtjeva veći broj podataka
Kraće treniranje, ali manja preciznost	Dulje treniranje, ali bolja preciznost
Jednostavne, linearne korelacije	Nelinearne složene korelacije

Još jedna razlika koja je spomenuta u [2] tvrdi da duboko učenje "zahtjeva" jedinicu za grafičku obradu (GPU). GPU uvelike ubrzava procese dubokog učenja, ali nije potreban kako bi model dubokog učenja bio treniran. Iako je model koji se kasnije u ovom radu trenira dosta jednostavan, još uvijek je model neuronskih mreža i treniran je s centralnom procesorskom jedinicom (CPU).

Postoji nekoliko vrsta neuronskih mreža, neke od njih su:

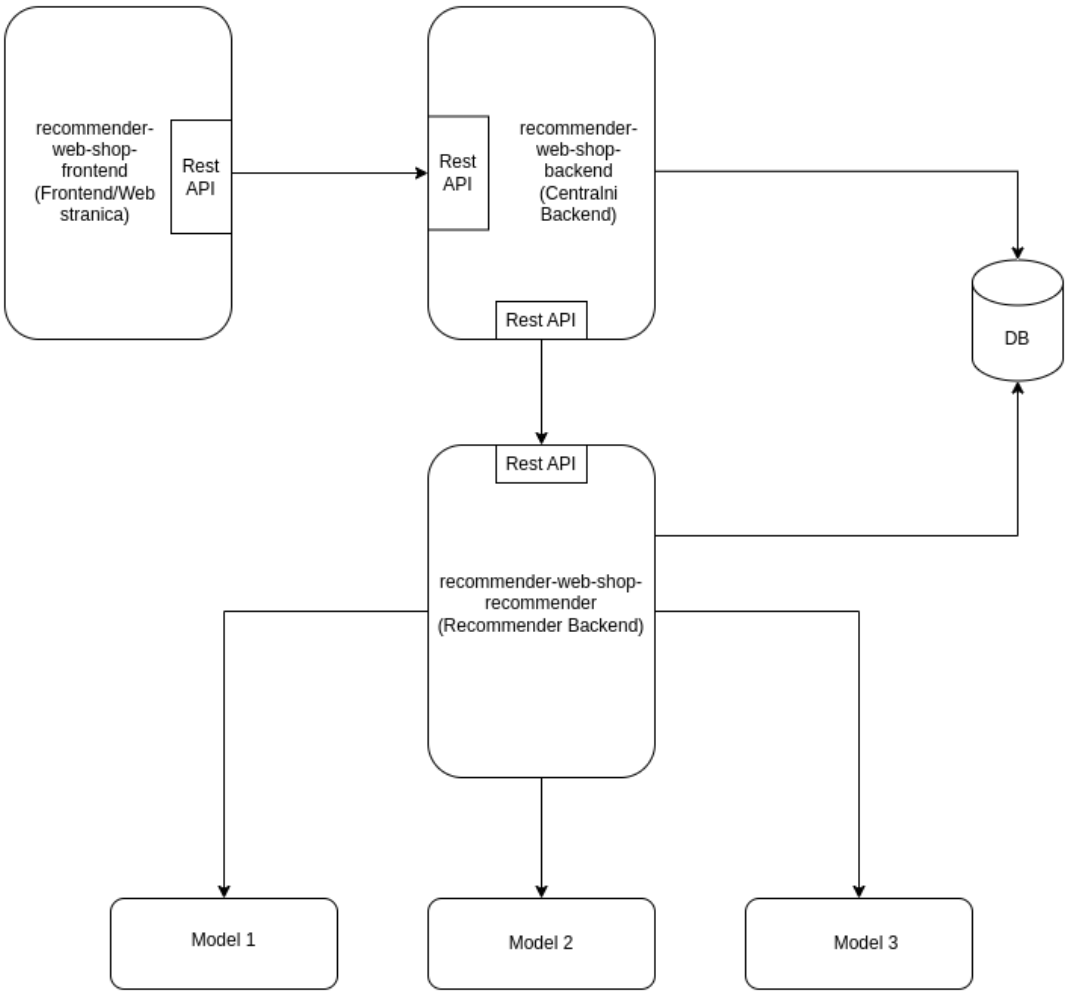
- Perceptron - jednostavne, plitke mreže s ulaznim slojem i izlaznim slojem. [30]
- Konvolucijska neuronska mreža (CNN) - koriste trodimenzionalne podatke za klasifikaciju slika i zadatke prepoznavanja objekata. [31]
- Transformer neuronske mreže - neuronska mreža koja uči kontekst, a time i značenje praćenjem odnosa u sekvencijalnim podacima poput riječi u rečenicama. [32]
- Feedforward neuronska mreža (FNN) - dopuštaju prosljeđivanje informacija samo unaprijed [30]

7. Implementacija sustava za preporuke

Za implementaciju sustava za preporuke postoje tri modula, a za njihovu povezanost koristi se mikroservisna arhitektura.

> „Mikroservisna arhitektura predstavlja arhitektonski stil izrade aplikacija. Mikroservisi omogućuju velikim aplikacijama da se podijele na manje, samostalne dijelove, od kojih svaki ima svoju specifičnu odgovornost. Kako bi se odgovorilo na jedan korisnički zahtjev, mikroservisna aplikacija koristi više internih mikroservisa koji zajedno sastavljaju odgovor." [33]

7.1. Mikroservisna arhitektura



Slika 11: Prikaz mikroservisne arhitekture aplikacije

Vidljiva su tri modula, svaki sa svojom jedinstvenom ulogom. Korisničko sučelje - šalje i prima podatke od poslužiteljskog sustava te vraća odgovore. Ti odgovori prikazuju se korisniku na smislen i prezentabilan način.

Centralni poslužiteljski sustav - prima zahtjeve od korisničkog sučelja te ih, u slučaju potrebe za preporukom, prosljeđuje sustavu za preporuke. Zadužen je za vraćanje podataka iz baze koji se prikazuju na sučelju, kao i za obradu podataka i njihovo zapisivanje u bazu.

Poslužiteljski sustav za preporuke - za svaki primljeni zahtjev obraća se treniranom modelu i predviđa rezultate. Predviđene rezultate zatim putem HTTP zahtjeva vraća centralnom poslužiteljskom sustavu. Ovaj modul je također odgovoran za treniranje modela.

Na slici 11 prikazan je tok komunikacije između tri modula. Korisničko sučelje putem HTTP zahtjeva šalje i prima podatke od centralnog poslužiteljskog sustava. Centralni poslužiteljski sustav odgovara na zahtjeve sučelja, odnosno korisnika, tako što ili odmah obrađuje podatke i vraća ih, ili prosljeđuje zahtjev poslužiteljskom sustavu za preporuke. Poslužiteljski sustav za preporuke, koristeći svoje trenirane modele, izvršava potrebna predviđanja te putem HTTP zahtjeva i JSON payloada vraća tražene podatke centralnom poslužiteljskom sustavu, koji zatim predviđanja prosljeđuje korisničkom sučelju spremne za prikaz korisniku.

Za primjer rada sustava može se predstaviti sljedeći scenarij. Korisnik A otvara web-stranicu, na web-stranici vidi popis filmova koji bi mu se mogli sviđati svi označeni nekim redoslijedom. Na jedan klik dogodilo se više zahtjeva više modula ove aplikacije:

- Korisničko sučelje šalje HTTP GET zahtjev centralnom poslužiteljskom sustavu za listu predviđenih filmova sa parametrima korisnika i brojem filmova koji se prikazuju korisniku
- Centralni poslužiteljski sustav šalje HTTP GET zahtjev poslužiteljskom sustavu za preporuke
- Sustav za preporuke vraća listu traženih preporuka centralnom poslužiteljskom sustavu
- Centralni poslužiteljski sustav dobiva listu, obrađuje ju i vraća je korisničkom sučelju
- Korisničko sučelje prikazuje obrađene podatke korisniku

7.2. Centralni poslužiteljski sustav

Kako je napomenuto ranije u radu, centralni poslužiteljski sustav napravljen je u programskom okviru Quarkus i u programskom jeziku Java. Koristi se popularan uzorak dizajna Kontroler-Servis-Repozitorij (*engl. Controller-Service-Repository*).

Ovaj kontroler u isječku koda 1 služi za komunikaciju između korisničkog sučelja i centralnog poslužiteljskog sustava, točnije, ovo je krajnja točka koja služi za dobivanje preporuka movie-movie načina kolaborativnog filtriranja koji je objašnjen malo kasnije. Kontroler vraća odgovor koji mu vrati servis u kojemu je većina logike.

```java
@RequestScoped
@Path("recommender")
public class RecommendationController {

    @Inject
    RecommenderService recommenderService;

    @GET
    @Path("/collaborative_filtering/movie-movie")
    public List<MovieNameRatingDto> getTopNMoviesForUser(@QueryParam("userId") Long
    ↪  userId, @QueryParam("count") Integer count) {
        return recommenderService.getTopNMoviesForUser(userId, count);
    }
}
```

Isječak koda 1: Primjer Controllera

```java
@ApplicationScoped
public class RecommenderService {

    @RestClient
    RecommenderClient recommenderClient;

    @Inject
    MovieRepository movieRepository;

    public List<MovieNameRatingDto> getTopNMoviesForUser(Long userId, Integer count)
    ↪  {
        List<MovieRatingDto> movieTopN = recommenderClient.getMovieTopN(userId,
        ↪  count);
        return movieTopN.stream().map(movieRatingDto ->
                MovieNameRatingDto.builder()
                        .movieId(movieRatingDto.getMovieId())
                        .rating(movieRatingDto.getRating())

                        ↪  .name(movieRepository.findById(movieRatingDto.getMovieId()).getName())
                        .build()
        ).toList();
    }
}
```

Isječak koda 2: Primjer Servisa

Servis u isječku koda 2 radi dvije bitne stvari. Prva je slanje HTTP zahtjeva poslužiteljskom sustavu za preporuke koje radi `recommenderClient`. Predviđanje filmova koji se sviđaju korisniku je odgovornost sustava za preporuke, nakon predviđanja vraća objekt `MovieNameRatingDto` iz isječka koda 3:

```java
@Data
@AllArgsConstructor
@NoArgsConstructor
public class MovieRatingDto {
    Long movieId;
    Float rating;
}
```

Isječak koda 3: Klasa *MovieRatingDto*

Druga važna stvar je oblikovanje podataka u korisniku razumljiv oblik. Korisniku identifikacijski broj i ocjena filma ne znače mnogo, ali centralni sustav pretražuje bazu podataka kako bi pronašao ime filma i tako oblikuje podatke koje vraća korisniku. Važno je također prikazati kako izgleda poziv poslužiteljskom sustavu za preporuke.

U isječku koda 4 predstavljen je jednostavan poziv čiju implementaciju uvelike olakšava programski okvir Quarkus. U pozadini se generira HTTP zahtjev koji se šalje na odgovarajuću krajnju točku (*engl. endpoint*). Ova krajnja točka zatim vraća željene podatke, odnosno preporuke.

```java
@RegisterRestClient(configKey = "recommender-api")
public interface RecommenderClient {

    @GET
    @Path("/collaborative_filtering/movie-movie")
    List<MovieRatingDto> getMovieTopN(@QueryParam("userId") Long userId,
    ↪  @QueryParam("count") Integer count);
}
```

Isječak koda 4: RestClient - Klasa *RecommenderClient*

Da bi ovakva komunikacija mogla funkcionirati mora se također imati prikaz modela baze podataka u kodu. Svaka klasa u modelu predstavlja tablicu u bazi podataka, atributi klase predstavljaju stupce, a posebne anotacije predstavljaju veze među tablicama. Bitne klase su Žanr (*Genre*), Film (*Movie*), FilmŽanr (*MovieGenre*), Korisnik (*User*), KorisnikFilmOcjena (*UserMovieRating*). Ove klase mogu se pronaći u isječcima koda 5, 6, 7, 8, 9:

```java
@Entity
@Table(name = "genre")
public class Genre {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @Column(name = "name")
    private String name;

    @OneToMany(mappedBy = "genre", cascade = CascadeType.ALL, orphanRemoval = true)
    private Set<MovieGenre> movieGenres;

}
```

Isječak koda 5: Klasa Žanr (*Genre*)

```java
@Entity
@Table(name = "movie")
public class Movie {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(name = "movie_indx")
    private Long movieIndx;

    @Column(name = "name")
    private String name;

    @OneToMany(mappedBy = "movie")
    private Set<MovieGenre> movieGenres;

    @OneToMany(mappedBy = "movie")
    private List<UserMovieRating> userMovieRatings;
}
```

Isječak koda 6: Klasa Film (*Movie*)

```java
@Entity
@Table(name = "movie_genre")
public class MovieGenre {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", nullable = false)
    private Long id;

    @ManyToOne
    @JoinColumn(name = "genre_id")
    private Genre genre;

    @ManyToOne(cascade = CascadeType.PERSIST)
    @JoinColumn(name = "movie_id")
    private Movie movie;
}
```

Isječak koda 7: Klasa FilmŽanr (*MovieGenre*)

```java
@Entity
@Table(name = "user", schema = "public")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "username", nullable = false, unique = true)
    private String username;

    @OneToMany(mappedBy = "user")
    List<UserMovieRating> userMovieRatings;
}
```

Isječak koda 8: Klasa Korisnik (*User*)

```java
@Entity
@Table(name = "user_movie_rating")
public class UserMovieRating {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @ManyToOne
    @JoinColumn(name = "movie_id")
    private Movie movie;

    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;

    @Column(name = "rating")
    private Float rating;
}
```

Isječak koda 9: Klasa KorisnikFilmOcjena (*UserMovieRating*)

Par ključnih anotacija su više naprema jedan (*@ManyToOne*) i jedan naprema više (*@OneToMany*) veze koje su detaljnije prikazane u ERA dijagramu u poglavlju *Skup podataka*. Osim ovih stavki, postoji anotacija *@Id* koja predstavlja primarni ključ tablice zajedno sa strategijom generacije. Na kraju postoje anotacije *@Column* i *@Table* koje predstavljaju stupac i ime tablice.

7.3. Implementacija poslužiteljskog sustava za preporuke

Najbitniji dio ove aplikacije, koji obrađene podatke pretvara u predviđanja, je poslužiteljski sustav za preporuke. Osim što generira predviđanja, ovaj sustav također šalje predviđene podatke centralnom poslužiteljskom sustavu.

7.3.1. Kolaborativno filtriranje

Općenito osim implementacija formula definiranih u ranijim poglavljima, koristi se *Flask* biblioteka, *Psycopg* biblioteka i *Pickle* biblioteka.

- *Flask* - služi za slanje HTTP zahtjeva i komunikacije sa centralnim poslužiteljskim sustavom
- *Psycopg* - pomaže kod izvršavanju upita nad bazom podataka
- *Pickle* - pomaže kod serijalizacije objekata u .bin datoteke

```
def get_rated_movies_excluding_user(self, user_id):
        con = self.__get_connection()
        cur = con.cursor()

        cur.execute("""SELECT m.id AS movie_id
        FROM movie m
        LEFT JOIN user_movie_rating umr ON m.id = umr.movie_id AND umr.user_id = %s
        WHERE umr.movie_id IS NULL""", (user_id,))
        all_movies = cur.fetchall()
        self.__close_connection(con, cur)
        return [movie_id[0] for movie_id in all_movies]
```

Isječak koda 10: Funkcija za dobivanje svih filmova koje korisnik nije ocijenio

U isječku koda 10 koristi se *Psycopg*. Ovakvim SQL upitima dobivaju se potrebni podaci za treniranje i predviđanje u modelima.

```
import flask
from api.service.collaborative_filtering_service import
↪  CollaborativeFilteringService

app = flask.Flask(__name__)

cfs = CollaborativeFilteringService()


@app.route("/collaborative_filtering/movie-movie")
def get_movie_movie_top_n():
    user_id = flask.request.args.get('userId')
    count = flask.request.args.get('count')
    return flask.jsonify(cfs.get_movie_movie_top_n(user_id, count))
```

Isječak koda 11: Krajnja točka za predviđanje temeljeno na stavkama

Isječak koda 11 prikazuje kranju točku (*engl. Endpoint*) koja omogućuje komunikaciju između dva poslužiteljska sustava.

7.3.1.1. Filtriranje temeljeno na korisnicima

```
def user_user_based(self, K, limit):
        self.neighbours = {}
        self.averages = {}
        self.deviations = {}

        user_count = np.max(list(self.user2movie.keys())) + 1
        count = 0
        for i in self.user2movie.keys():
            movies_i = self.user2movie[i]
            movies_i_set = set(movies_i)

            dev_i = {}
            self.user_user_calculate(movies_i, dev_i, i)
            sigma_i = self.__calculate_sigma(dev_i)

            sorted_list = SortedList()

            for j in self.user2movie.keys():
                if j == i:
                    continue

                movies_j = self.user2movie[j]
                movies_j_set = set(movies_j)
                common_movies = movies_i_set.intersection(movies_j_set)

                if len(common_movies) > limit:
                    dev_j = {}
                    self.user_user_calculate(movies_j, dev_j, j)
                    sigma_j = self.__calculate_sigma(dev_j)

                    numerator = sum(dev_i[m] * dev_j[m] for m in common_movies)
                    w_ij = numerator / (sigma_i * sigma_j)

                    sorted_list.add((-w_ij, j))

                    if len(sorted_list) > K:
                        del sorted_list[-1]

            self.neighbours[i] = sorted_list
            print(f"On user: {count}/{user_count}")
            count += 1

        self.__dump("/neighbours_user.bin", "/averages_user.bin",
        ↪  "/deviations_user.bin")
```

Isječak koda 12: Kolaborativno filtriranje temeljeno na korisnicima

Isječak koda 12 zapravo je implementacija formula spomenutih u ranijim poglavljima. Ovu metoda može se razdvojiti na dvije petlje koje iteriraju kroz hash tablice `user2movie` i

izračunate vrijednosti spremaju u hash tablice `ratings, averages, deviations`. Ključ u `user2movie` je id korisnika, a vrijednost je lista svih filmova koje je korisnik ocijenio. Prvo postoji funkcija `self.__calculate_sigma` koja izvuče i izračuna vrijednosti koje su potrebne za daljnju obradu.

Podaci koji se dobivaju su sve ocjene korisnika za svaki film (`ratings`), prosjek ocjena za pojedinog korisnika (`averages`), odstupanja od ocjene korisnika (`deviations`). S ovakvim vrijednostima može se izračunati sigma koja zapravo predstavlja $\sqrt{\sum_{j \in \eta_{ii'}} (r_{ij} - \overline{r}_i)^2}$. Nakon što je ovo napravljeno za korisnika i, radi se isto za korisnika j. U liniji 19, zatim se gleda presjek svih filmova korisnika i i korisnika j.

Onda postoji uvjet koji provjerava minimalan broj zajedničkih filmova da bi se uopće razmatrala daljnja obrada. Ako je broj zajedničkih filmova veći od granice (`limit`) onda se može nastaviti. Od linije 20 ponavljaju se isti koraci za računanje vrijednosti, ali ovaj put se računa brojnik (`nominator`) koji je zapravo $\sum_{j \in \eta_{ii'}} (r_{ij} - \overline{r_i})(r_{i'j} - \overline{r_i'})$. Nakon toga, računa se težina prema ranije navedenoj formuli. Negirana težina se zatim doda u sortiranu listu (sorted_list objekt) koja automatski sortira vrijednosti prema rastućim vrijednostima (zato se i negira težina).

Ako lista postane veća od dozvoljenih `K najbližih susjeda` briše se zadnji element polja. Odnosno, briše se element polja koji ima najmanju povezanost. U Pearsonovom koeficijentu korelacije, broj bliže 1 znači veća povezanost. Na kraju se u hash tablicu `neighbours` stavlja sortirana lista *tuplea* `((-w_ij, j))`. Konačno, objekti `neighbours, averages, deviations` spremaju se u .bin datoteke koje se koriste za predviđanja. Vrijedno je napomenuti da *K* iznosi 25.

```
1  def predict(self, entity_to_predict, entity_predicting_with, type):
2          numerator = 0
3          denominator = 0
4          for neg_w, other_entity in self.neighbours[entity_to_predict]:
5              try:
6                  numerator += (
7                      -neg_w * self.deviations[other_entity][entity_predicting_with]
8                  )
9                  denominator += abs(neg_w)
10             except KeyError:
11                 pass

12         if denominator == 0:
13             prediction = self.averages[entity_to_predict]
14         else:
15             prediction = self.averages[entity_to_predict] + numerator / denominator

16         prediction = min(5, prediction)
17         prediction = max(0.5, prediction)  # min rating is 0.5
18         return prediction
```

Isječak koda 13: Predviđanje ocjene

Isječak koda 13 prikazuje metodu za predviđanje. Prima 3 argumenata. To su: entity_to_predict, entity_predicting_with, type. Type služi za učitavanje ispravnih .bin datoteka. Entity_to_predict, entity_predicting_with su za korisničko filtriranje korisnički id pa id filma, a za filtriranje temeljno na filmu vrijedi obrnuto.

Linije 6-15 prate formulu [17]:

$$s(i,j) = \overline{r_i} + \frac{\sum_{i' \in \Omega_j} W_{ii'} \{r_{i'j} - \overline{r_{i'}}\}}{\sum_{i' \in \Omega_j} |W_{ii'}|}$$

Ako je nazivnik (*denominator*) jednak 0 onda se samo dodaje prosječna ocjena korisnika/filma. Na kraju se pobrine da previđeni rezultat ne može ići iznad maksimuma od 5 ili ispod minimuma od 0.5.

7.3.1.2. Filtriranje temeljeno na stavkama

```
1  def movie_movie_based(self, K, limit):
2          self.neighbours = {}
3          self.averages = {}
4          self.deviations = {}
5          count = 0
6          for i in self.movie2user.keys():
7              users_i = self.movie2user[i]
8              users_i_set = set(users_i)
9              dev_i = {}
10             self.movie_calculate_values(users_i, dev_i, i)
11             sigma_i = self.__calculate_sigma(dev_i)
12             sorted_list = SortedList()
13             for j in self.movie2user.keys():
14                 if i == j:
15                     continue
16                 users_j = self.movie2user[j]
17                 users_j_set = set(users_j)
18                 users_in_common = users_i_set.intersection(users_j_set)
19                 if len(users_in_common) > limit:
20                     dev_j = {}
21                     self.movie_calculate_values(users_j, dev_j, j)
22                     sigma_j = self.__calculate_sigma(dev_j)
23                     numerator = sum(dev_i[u] * dev_j[u] for u in users_in_common)
24                     w_ij = numerator / (sigma_i * sigma_j)
25                     sorted_list.add((-w_ij, j))
26                     if len(sorted_list) > K:
27                         del sorted_list[-1]

28             self.neighbours[i] = sorted_list
```

Isječak koda 14: Kolaborativno filtriranje temeljeno na stavkama

Implementaciju filtriranja temeljenog na stavkama u isječku koda 14 nepotrebno je do-

datno objašnjavati jer je gotovo identična prethodnoj implementaciji za korisnike. Jedina razlika je u tome što se u ovoj funkciji iterira kroz ID-eve filmova, čime se analiziraju ocjene koje su korisnici dobili od filmova. Također, gledaju se zajednički korisnici svakog filma, a računanje i ostale formule funkcioniraju na isti način kao u ranije spomenutim poglavljima. Važno je napomenuti da je *K* u ovoj funkciji jednak 20.

7.4. Matrična faktorizacija

U prošlim poglavljima ispričano je sve o paketima koji se koriste tako da se odmah može uroniti u kod za matričnu faktorizaciju.

```
K = 10
W = np.random.randn(N, K)
b = np.zeros(N)
U = np.random.randn(M, K)
c = np.zeros(M)
mu = np.mean(list(usermovie2rating.values()))
```

Isječak koda 15: Matrična faktorizacija koja je preuzeta i modificirana od [34] - postavljanje parametara

U isječku koda 15 prikazano je postavljanje inicijalnih varijabli:

K - broj latentnih faktora
W - matrica korisnika $N \times K$ veličine, N predstavlja broj korisnika
U - matrica filmova $M \times K$ veličine, M predstavlja broj fimova
b - vektor pristranosti korisnika
c - vektor pristranosti filmova
mu - vrijednost penalizacije koja iznosi prosjek ocjena zapravo μ

Ispod, u isječku koda 16 slijedi funkcija koja računa srednju kvadratnu pogrešku tako da uzme *engl. dictionary* objekt koji sadržava identifikacijske brojeve korisnika i filma kao ključeve, te ocjenu kao vrijednost. Za svaku ocjenu, funkcija postavlja i, j kao identifikacijske brojeve korisnika i filma, pristupa ranije spomenutim matricama i računa skalarni produkt latentnih faktora korisnika i filma, čime se dobije predviđena ocjena. Razlika između predviđene i stvarne ocjene se zatim kvadrira kako bi se dobila suma kvadratnih pogrešaka (SSE). Na kraju, ova kvadratna greška se podijeli s brojem ocjena, čime se dobiva srednja kvadratna pogreška (MSE).

```
def get_loss(d):
    # d: (user_id, movie_id) -> rating
    N = float(len(d))
    sse = 0
    for k, r in d.items():
        i, j = k
        p = W[i].dot(U[j]) + b[i] + c[j] + mu
        sse += (p - r)*(p - r)
    return sse / N
```

Isječak koda 16: Matrična faktorizacija koja je preuzeta i modificirana od [34] - funckija izračuna pogreške

Ostatak matrične faktorizacije u kodu prati formule koje su ranije detaljno objašnjene.

```
for i in user2movie.keys():
matrix = np.eye(K) * reg
vector = np.zeros(K)
bi = 0
    for j in user2movie[i]:
        r = usermovie2rating[(i, j)]
        matrix += np.outer(U[j], U[j])
        vector += (r - b[i] - c[j] - mu) * U[j]
        bi += (r - W[i].dot(U[j]) - c[j] - mu)

W[i] = np.linalg.solve(matrix, vector)
b[i] = bi / (len(user2movie[i]) + reg)
```

Isječak koda 17: Matrična faktorizacija koja je preuzeta i modificirana od [34] - izračun w_i, b_i

U isječku koda 17, treba se raspraviti o `matrix i vector` varijablama. Varijabla `matrix` predstavlja λI i inicijalizirana je jediničnom matricom. `Vector` započinje jednodimenzionalom matricom inicijaliziranom s nulom za stupce kojih ima K. Nakon ovoga kod prati formulu:

$$w_i = (\sum_{j \in \Psi_i} u_j u_j^T + \lambda I)^{-1} \sum_{j \in \Psi_i} (r_{ij} - b_i - c_j - \mu) u_j$$

$$b_i = \frac{1}{|\Psi_i| + \lambda} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - c_j - \mu)$$

Vanjska petlja prolazi svakog korisnika i kojemu se računa vrijednost $W[i]$ u unutarnjoj petlji koja iterira svaki film j koji je korisnik ocijenio i prema formulama računa bitne vrijednosti. Osim $W[i]$ računa se i $b[i]$ koji predstavlja pristranost korisnika.

Isti princip samo s fokusom na filmove odvija se u drugom dijelu funkcije, u isječku koda 18:

```
for j in movie2user.keys():
        matrix = np.eye(K) * reg
        vector = np.zeros(K)

        cj = 0
        try:
            for i in movie2user[j]:
                r = usermovie2rating[(i, j)]
                matrix += np.outer(W[i], W[i])
                vector += (r - b[i] - c[j] - mu)*W[i]
                cj += (r - W[i].dot(U[j]) - b[i] - mu)

            # set the updates
            U[j] = np.linalg.solve(matrix, vector)
            c[j] = cj / (len(movie2user[j]) + reg)
```

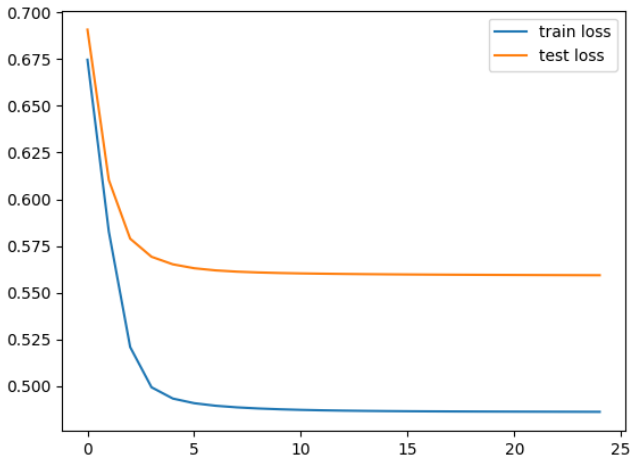
Isječak koda 18: Matrična faktorizacija koja je preuzeta i modificirana od [34] - izračun u_j, c_j

Ovaj dio funkcije predstavlja formule:

$$u_j = (\sum_{i \in \Omega_j} w_i w_i^T + \lambda I)^{-1} \sum_{i \in \Omega_j} (r_{ij} - b_i - c_j - \mu) w_i$$

$$c_j = \frac{1}{|\Omega_j| + \lambda} \sum_{j \in \Psi_i} (r_{ij} - w_i^T u_j - b_i - \mu)$$

Postoji grafički prikaz treniranja modela, koji je prikazan na slici 12:



Slika 12: Matplot graf matrične faktorizacije - vlastita izrada

Proces se odvija kroz 25 epoha, što znači da se cijeli skup podataka prolazi 25 puta, pri čemu se svaki put poboljšava preciznost predviđanja. Graf na slici prikazuje kako se srednja kvadratna pogreška (MSE) mijenja tijekom epoha. Na apscisi (x-osi) prikazan je broj epoha,

dok ordinata (y-os) predstavlja MSE. Većim brojem prolazaka kroz skup podataka smanjuje se vrijednost MSE, što znači da se model poboljšava. Plava linija, koja predstavlja MSE na skupu podataka za treniranje, vrlo je slična liniji za testni skup, što ukazuje da kod modela nema pretreniranja. Pretreniranje se događa kada model postane previše prilagođen podacima za treniranje, što rezultira slabijim performansama na nepoznatim podacima iz testnog skupa.

7.5. Duboko učenje - umjetne neuronske mreže

Za razliku od prethodnih implementacija koje su koristile matematičke formule, sada postoji lakši način koristeći biblioteku. Biblioteka koja je korištena zove se Keras [1]. Kako bi se započelo s izradom neuronske mreže, prvo je potrebno učitati podatke:

```
train_user_ids = np.array([user for (user, movie) in usermovie2rating.keys()])
train_movie_ids = np.array([movie for (user, movie) in usermovie2rating.keys()])
train_ratings = np.array(list(usermovie2rating.values())) - mu

test_user_ids = np.array([user for (user, movie) in usermovie2rating_test.keys()])
test_movie_ids = np.array([movie for (user, movie) in usermovie2rating_test.keys()])
test_ratings = np.array(list(usermovie2rating_test.values())) - mu

N = np.max(list(user2movie.keys())) + 1
m1 = np.max(list(movie2user.keys()))
m2 = np.max([m for (u, m), r in usermovie2rating_test.items()])
M = max(m1, m2) + 1

K = 10
epochs = 15
```

Isječak koda 19: Neuronska mreža koja je preuzeta i modificirana od [35] - priprema podataka

Isječak koda 19 prikazuje varijable `train_user_ids` i `train_movie_ids` koje sadrže ID-eve korisnika i filmova, te se, kako bi postale ulazni slojevi, moraju pohraniti u formatu `numpy` polja. `train_ratings` također se pretvara u `numpy` polje, pri čemu se oduzima prosječna ocjena (`mu`) iz skupa podataka. Slična transformacija se primjenjuje na testne varijable.

Nakon ovoga, određuju se N, M, K, i epochs, gdje N predstavlja ukupni broj korisnika (maksimalni korisnički ID + 1), M predstavlja ukupni broj filmova (maksimalni filmski ID + 1), K označava broj latentnih faktora, a epochs broj iteracija kroz cijeli skup podataka.

U isječku koda 20 prikazano je kreiranje jednostavne neuronske mreže koja se koristi za modeliranje korisničkih i filmskih latentnih faktora te za predviđanje ocjena.

[1] https://keras.io/

```
u = Input(shape=(1,))
m = Input(shape=(1,))
u_embedding = Embedding(N, K)(u)
m_embedding = Embedding(M, K)(m)
u_embedding = Flatten()(u_embedding)
m_embedding = Flatten()(m_embedding)
x = Concatenate()([u_embedding, m_embedding])

x = Dense(400)(x)
x = Activation('relu')(x)
x = Dense(1)(x)
```

Isječak koda 20: Neuronska mreža koja je preuzeta i modificirana od [35] - kreiranje slojeva neuronske mreže

Ulazni sloj predstavljaju varijable `u` koje su id-evi korisnika, `m` koji su id-evi filmova gdje `Input(shape=(1,))` definira ulazni vektor dimenzije 1 jer su ulazi samo id-evi. Ovi id-evi kategoričke su varijable koje neuronske mreže ne mogu razumijeti. Iz ovog razloga koriste se ugrađivački (*engl. embedding*) slojevi kako bi se vratili matrični prikazi kategoričke varijable [36]. Ovi slojevi ključni su za pretvaranje korisničkih i filmskih id-eva u latentne faktore.

`u_embedding = Embedding(N, K)(u)` vraćaju matrice dimenzija $(N, 1, K)$ i $(M, 1, K)$ gdje 1 označava da postoji jedan ID po korisniku/filmu, a K označava broj latentnih faktora.

`Flatten()` funkcija uklanja dimenziju 1 i vraća matricu dimenzija (N, K) ili (M, K). Nakon toga, funkcija `Concatenate([u_embedding, m_embedding])` spaja vektore latentnih faktora u jedan vektor dimenzija `(batch_size, K)`.

Dodaje se gusti sloj s 400 neurona `x = Dense(400)(x)` s aktivacijskim algoritmom `x = Activation('relu')(x)`.

Konačno `x = Dense(1)(x)` predstavlja izlazni sloj s jednim neuronom koji predviđa ocjenu koju će korisnik dati filmu.

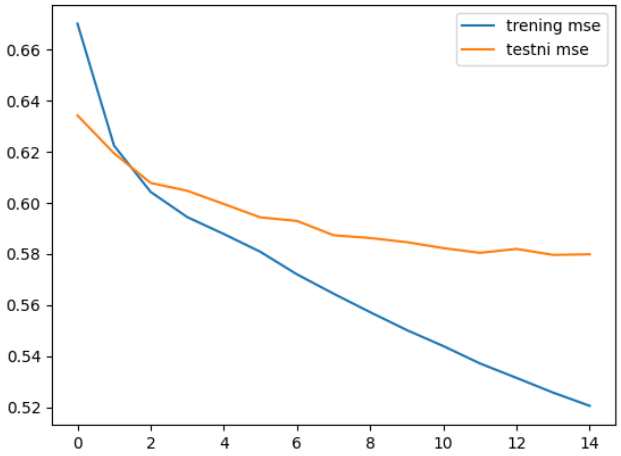
Na kraju se ovakav model izgradi i trenira, a to je prikazano u isječku koda 21. Model se kreira koristeći ulazne slojeve `u` i `m` te izlazni sloj `x`. Izgradi se s optimizacijskim algoritmom `'adam'` i funkcijom pogreške `'mse'`, koja se koristi za minimiziranje pogreške predviđanja. Također, metrika `'mse'` koristi se za praćenje performansi modela tijekom treniranja.

```
model = Model(inputs=[u, m], outputs=x)
model.compile(
    loss='mse',
    optimizer='adam',
    metrics=['mse'],
)

r = model.fit(
    x=[train_user_ids, train_movie_ids],
    y=train_ratings,
    epochs=epochs,
    batch_size=128,
    validation_data=(
        [test_user_ids, test_movie_ids],
        test_ratings
    ),
    verbose=2,
)
```

Isječak koda 21: Neuronska mreža koja je preuzeta i modificirana od [35] - izgradnja i treniranje modela

Model se trenira koristeći skup podataka za obuku (`train_user_ids`, `train_movie_ids`, i `train_ratings`) s određenim brojem epoha i veličinom batch-a. Također, koristi se skup za validaciju (`test_user_ids`, `test_movie_ids`, i `test_ratings`) kako bi se pratila sposobnost modela da generalizira na nove podatke.

Kao i u prošlom slučaju s matričnom faktorizacijom postoji graf koji prikazuje treniranje modela kroz epohe:



Slika 13: Matplot graf neuronske mreže - vlastita izrada

Interpretacija ovog grafa slična je onoj za matričnu faktorizaciju, stoga nema potrebe

za ponovnim detaljnim objašnjenjem. Iz grafa se može vidjeti da su rezultati dobri i da model nema problema s pretreniranjem.

Model se koristi za predviđanje na sljedeći način:

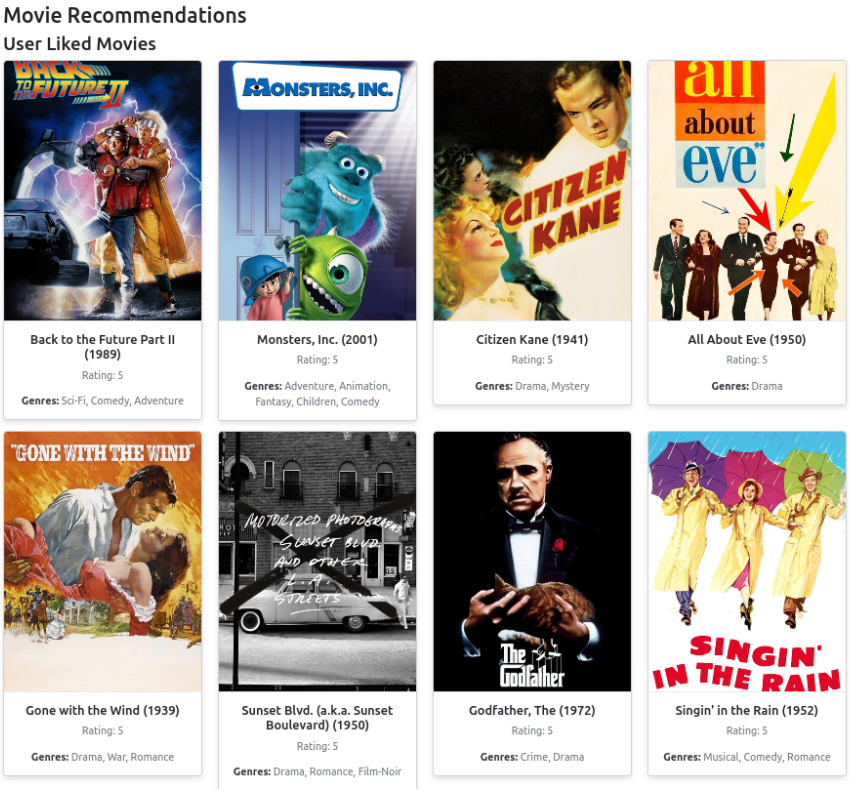
```
def predict_batch(self, user_id, movie_batch):
      user_id_array = np.array([user_id] * len(movie_batch)).reshape(-1, 1)
      movie_id_array = np.array(movie_batch).reshape(-1, 1)

      predicted_ratings = self.model.predict([user_id_array,
      ↪  movie_id_array]).flatten()
      predicted_ratings += self.mu
      return predicted_ratings
```

Isječak koda 22: Predviđanja neuronske mreže

Za predviđanje u isječku koda 22 koristi se ID korisnika i lista ID-eva filmova. Ovi podaci prvo se pretvaraju u odgovarajući oblik koristeći `reshape`. To omogućava modelu da pravilno obradi ulaze. Nakon što model izvrši predviđanja, dodaje se prosjek svih ocjena (`self.mu`) kako bi se dobile originalne ocjene. Ovaj korak je potreban jer su se prethodno oduzeli prosjeci kako bi se poboljšala učinkovitost modela. Na ovaj način, model vraća predviđene ocjene u izvornom rasponu.

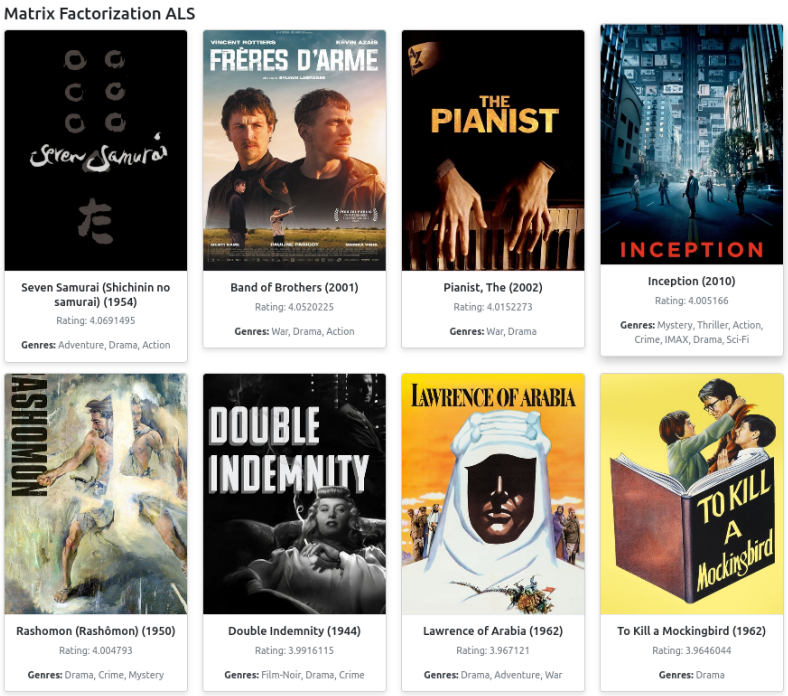
7.6. Rad web-stranice

Svi izrađeni modeli koriste se na web-stranici nakon prijave korisnika. Web-stranica relativno je jednostavna i prikazuje osam najboljih preporuka za svaki od spomenutih sustava preporuka.



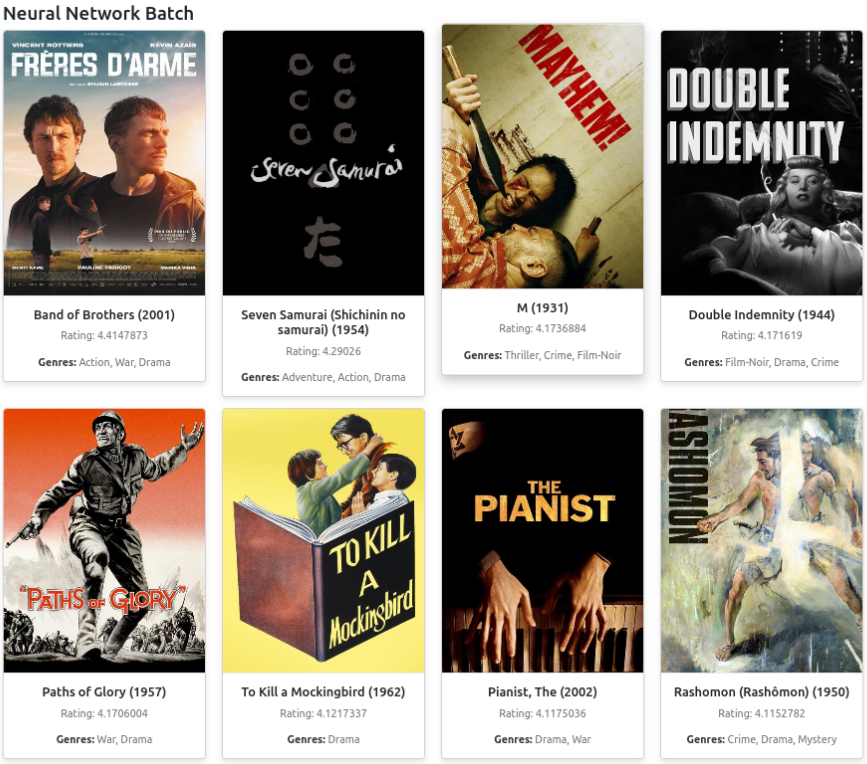
Slika 14: Prikaz filmova koji se sviđaju korisniku

Na slici 14 prikazano je osam najdražih filmova korisnika. Analizom žanrova može se primijetiti da korisniku najviše odgovaraju dramski, avanturistički i ratni filmovi. Slika 15 prikazuje preporuke modela matrične faktorizacije:



Slika 15: Prijedlozi filmova od modela matrične faktorizacije

Vidi se da model preporučuje filmove koji odgovaraju korisnikovim ukusima. Većina preporučenih filmova su akcijski filmovi s puno drame, a neki su čak kriminalni i ratni filmovi. Ovi žanrovi su u skladu s onime što korisnik voli. Slične preporuke od neuronskih mreža mogu se vidjeti na slici 16:



Slika 16: Prijedlozi filmova od modela neuronskih mreža

Slične preporuke vraća kolaborativno filtriranje prema korisniku i prema sadržaju, stoga ih nije potrebno sve prikazati. Rad web-stranice objašnjen je u ranijim poglavljima, ali može se ukratko ponoviti. Web-stranica (frontend) šalje zahtjeve modulu recommender-web-shop-backend, koji zatim upućuje zahtjeve modelima. Ovi modeli generiraju predviđanja koja se vraćaju recommender-web-shop-backendu, koji zatim parsira podatke i vraća ih web-stranici. Važno je napomenuti da slike filmova nisu pohranjene u bazi podataka, već se dohvaćaju putem The Movie Database API-a [2]

[2]https://developer.themoviedb.org/reference/intro/getting-started

8. Zaključak

Ovaj rad predstavlja sveobuhvatan pristup razvoju sustava preporuka koji se temelji na kolaborativnom filtriranju, matričnoj faktorizaciji i neuronskim mrežama, implementiranim unutar arhitekture mikroservisa. Kroz integraciju ovih različitih komponenti, sustav uspješno preporučuje filmove korisnicima na temelju njihovih prethodnih interakcija i ukusa.

Kolaborativno filtriranje, koje se temelji na sakupljenim podacima o ocjenama, pokazalo se moćnim alatom za prepoznavanje odličnih filmova za preporuke. Međutim, pati od skalabilnosti i "cold start" problema. Za prevladavanje ovih izazova korištena je matrična faktorizacija koja omogućuje učinkovito otkrivanje latentnih faktora koji opisuju odnose između korisnika i filmova. Metoda alternirajućih najmanjih kvadrata (ALS), korištena u ovom radu, nudi ravnotežu točnosti i računalne učinkovitosti.

Osim toga, uključivanjem neuronskih mreža u postupak preporuke, uzimaju se u obzir složeniji obrasci u podacima. Neuronske mreže, kao primjer dubokog učenja, omogućuju modelu učenje apstraktnih, višeslojnih reprezentacija podataka čime se poboljšava kvaliteta preporuka. Razlika između strojnog učenja i dubokog učenja leži upravo u složenosti i dubini modela. Strojnim učenjem vladaju relativno jednostavniji algoritmi i pristupi, a duboko učenje omogućuje rad sa složenijim strukturama podataka i uključuje više slojeva transformacija što rezultira većom sposobnošću generalizacije na neviđenim podacima.

Implementacija sustava unutar arhitekture mikroservisa omogućila je skalabilnost, fleksibilnost i modularnost. Podjela sustava na tri modula: web-stranice (frontend), centralnog poslužiteljskog sustava (recommender-web-shop-backend) i komponente sustava za preporuku osigurava bolju održivost koda, jednostavnost pri dodavanju novih funkcionalnosti i otpornost na pogreške u pojedinim dijelovima sustava. Ovo je osobito važno u kontekstu modernih web-aplikacija, gdje su brzina i pouzdanost ključni za zadovoljstvo korisnika.

Ovakvom kombinacijom različitih tehnologija i metodologija može se dovesti do stvaranja sustava preporuka koji ne samo da ispunjava svoje osnovne funkcije, već je i prilagodljiv, skalabilan i sposoban nositi se s budućim izazovima. Budući rad u ovom području mogao bi uključivati eksperimentiranje s različitim konfiguracijama neuronskih mreža, istraživanje novih metoda regulacije i optimiziranje arhitekture mikroservisa za još brže i pouzdanije preporuke. Brz razvoj područja umjetne inteligencije i strojnog učenja osigurava da će sustavi poput ove implementacije igrati sve važniju ulogu u personalizaciji korisničkog iskustva u nadolazećim godinama.

Popis literature

[1] „What is ai?" IBM. (), adresa: https://www.ibm.com/topics/artificial-intelligence (pogledano 17. 8. 2024.).

[2] C. Staff. „Deep learning vs. machine learning: A beginner's guide," Coursera. (), adresa: https://www.coursera.org/articles/ai-vs-deep-learning-vs-machine-learning-beginners-guide (pogledano 16. 8. 2024.).

[3] „Recommendation system," Nvidia. (), adresa: https://www.nvidia.com/en-us/glossary/recommendation-system/ (pogledano 16. 8. 2024.).

[4] „Content-based filtering," Google. (), adresa: https://developers.google.com/machine-learning/recommendation/content-based/basics (pogledano 17. 8. 2024.).

[5] „7 critical challenges of recommendation engines," Appier. (), adresa: https://www.appier.com/en/blog/7-critical-challenges-of-recommendation-engines (pogledano 17. 8. 2024.).

[6] B. Schifferer. „Using neural networks for your recommender system," Nvidia. (), adresa: https://developer.nvidia.com/blog/using-neural-networks-for-your-recommender-system/ (pogledano 17. 8. 2024.).

[7] S. Bouguezzi. „How does the amazon recommendation system work?" Baeldung-CS. (), adresa: https://www.baeldung.com/cs/amazon-recommendation-system (pogledano 17. 8. 2024.).

[8] L. Hardesty. „The history of amazon's recommendation algorithm," Amazon. (), adresa: https://www.amazon.science/the-history-of-amazons-recommendation-algorithm (pogledano 17. 8. 2024.).

[9] „Behind the scenes of the netflix recommendation algorithm," invisibly. (), adresa: https://www.invisibly.com/learn-blog/netflix-recommendation-algorithm/ (pogledano 17. 8. 2024.).

[10] D. Pastukhov. „Inside spotify's recommender system: A complete guide to spotify recommendation algorithms," Music tomorrow. (), adresa: https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022 (pogledano 17. 8. 2024.).

[11] M. Sarwat i M. F. Mokbel, „Collaborative Filtering," *Encyclopedia of Database Systems, Second Edition*, L. Liu i M. T. Özsu, ur., Springer, 2018. DOI: 10.1007/978-1-4614-8265-9_80733.

[12] E. K. Jacob Murel Ph.D. „What is collaborative filtering?” IBM. (), adresa: https://www.ibm.com/topics/collaborative-filtering#:~:text=Collaborative%20filtering%20is%20an%20information,have%20interacted%20with%20that%20item. (pogledano 20. 7. 2024.).

[13] W. Yue, Z. Wang, W. Liu, B. Tian, S. Lauria i X. Liu, „An optimally weighted user- and item-based collaborative filtering approach to predicting baseline data for Friedreich's Ataxia patients,” *Neurocomputing*, sv. 419, str. 287–294, 2021., ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.08.031.

[14] F. Isinkaye, Y. Folajimi i B. Ojokoh, „Recommendation systems: Principles, methods and evaluation,” *Egyptian Informatics Journal*, sv. 16, br. 3, str. 261–273, 2015., ISSN: 1110-8665. DOI: https://doi.org/10.1016/j.eij.2015.06.005.

[15] B. Kim. „Introduction to matrix factorization - collaborative filtering with python 12,” Codinfox. (), adresa: https://buomsoo-kim.github.io/recommender%20systems/2020/09/25/Recommender-systems-collab-filtering-12.md/ (pogledano 20. 7. 2024.).

[16] D. E. Caughlin. „Chapter 21 centering standardizing variables,” Kent State University. (), adresa: https://rforhr.com/center.html (pogledano 20. 7. 2024.).

[17] C. Saluja. „Collaborative filtering based recommendation systems exemplified..,” Medium. (), adresa: https://towardsdatascience.com/collaborative-filtering-based-recommendation-systems-exemplified-ecbffe1c20b1 (pogledano 20. 7. 2024.).

[18] L. S. S. H. Alizadeh. „A note on pearson correlation coefficient as a metric of similarity in recommender system,” Kent State University. (), adresa: https://ieeexplore.ieee.org/document/7270736 (pogledano 22. 7. 2024.).

[19] G. Linden, B. Smith i J. York, „Amazon.com recommendations: item-to-item collaborative filtering,” *IEEE Internet Computing*, sv. 7, br. 1, str. 76–80, 2003. DOI: 10.1109/MIC.2003.1167344.

[20] Y. Koren, R. Bell i C. Volinsky, „Matrix Factorization Techniques for Recommender Systems,” *Computer*, sv. 42, br. 8, str. 30–37, 2009. DOI: 10.1109/MC.2009.263.

[21] Lazy programmer. „Tutorial on collaborative filtering and matrix factorization in python.” (), adresa: https://lazyprogrammer.me/tutorial-on-collaborative-filtering-and-matrix-factorization-in-python/ (pogledano 7. 8. 2024.).

[22] „Error sum of squares (sse),” Stanford. (), adresa: https://hlab.stanford.edu/brian/error_sum_of_squares.html (pogledano 7. 8. 2024.).

[23] Y. Zhou, D. Wilkinson, R. Schreiber i R. Pan, „Large-Scale Parallel Collaborative Filtering for the Netflix Prize,” *Algorithmic Aspects in Information and Management*, 6. 2008., str. 337–348, ISBN: 978-3-540-68865-5. DOI: 10.1007/978-3-540-68880-8_32.

[24] aunnnn. „Linear regression with regularization.” (), adresa: https://aunnnn.github.io/ml-tutorial/html/blog_content/linear_regression/linear_regression_regularized.html (pogledano 7. 8. 2024.).

[25] Jim Holdsworth, Mark Scapicchio. „What is deep learning?" IBM. (), adresa: https://www.ibm.com/topics/deep-learning (pogledano 16. 8. 2024.).

[26] E. Goz, M. Yuceer i E. Karadurmus, „Total Organic Carbon Prediction with Artificial Intelligence Techniques," *29th European Symposium on Computer Aided Process Engineering*, serija Computer Aided Chemical Engineering, A. A. Kiss, E. Zondervan, R. Lakerveld i L. Özkan, ur., sv. 46, Elsevier, 2019., str. 889–894. DOI: https://doi.org/10.1016/B978-0-12-818634-3.50149-1.

[27] „Artificial neural networks and its applications," Geeks for geeks. (), adresa: https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/ (pogledano 16. 8. 2024.).

[28] „Gradient descent," Khan Academy. (), adresa: https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent#:~:text=Gradient%20descent%20is%20an%20algorithm,approximates%20the%20solution%20with%20numbers. (pogledano 16. 8. 2024.).

[29] D. P. Kingma i J. Ba, *Adam: A Method for Stochastic Optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[30] „What is a neural network?" Cloudfare. (), adresa: https://www.cloudflare.com/learning/ai/what-is-neural-network/ (pogledano 16. 8. 2024.).

[31] „What are convolutional neural networks?" IBM. (), adresa: https://www.ibm.com/topics/convolutional-neural-networks (pogledano 16. 8. 2024.).

[32] R. Merritt. „What is a transformer model?" Nvidia. (), adresa: https://blogs.nvidia.com/blog/what-is-a-transformer-model/ (pogledano 16. 8. 2024.).

[33] „What is microservices architecture?" Google. (), adresa: https://cloud.google.com/learn/what-is-microservices-architecture (pogledano 19. 7. 2024.).

[34] L. programmer. „Recommenders-mf." (), adresa: https://github.com/lazyprogrammer/machine_learning_examples/blob/master/recommenders/mf.py (pogledano 16. 8. 2024.).

[35] L. programmer. „Recommenders-mf_keras_deep." (), adresa: https://github.com/lazyprogrammer/machine_learning_examples/blob/master/recommenders/mf_keras_deep.py (pogledano 16. 8. 2024.).

[36] soham Medewar. „Embedding layers in keras." (), adresa: https://www.naukri.com/code360/library/embedding-layers-in-keras (pogledano 17. 8. 2024.).

Popis slika

Popis tablica

Popis isječaka koda

9. Prilog

Programski kod korišten u radu može se pronaći na repozitoriju: https://github.com/mmusica/recommender-web-shop