UNIVERZITET U BEOGRADU
MATEMATIČKI FAKULTET

Jovana D. Pejkić

METODA UZAJAMNOG FILTRIRANJA I PRIMENE U ELEKTRONSKOJ TRGOVINI

master rad

Beograd, 2023.

Mentor:

dr Aleksandar KARTELJ, docent
Univerzitet u Beogradu, Matematički fakultet

Članovi komisije:

dr Nenad MITIĆ, redovni profesor
Univerzitet u Beogradu, Matematički fakultet

dr Jelena GRAOVAC, docent
Univerzitet u Beogradu, Matematički fakultet

Datum odbrane: _____

Posvećeno Jovanu i porodici

Naslov master rada: Metoda uzajamnog filtriranja i primene u elektronskoj trgovini

Rezime:

Elektronska trgovina postaje sve zastupljeniji vid kupovine proizvoda i usluga. Preplavljenost informacijama predstavlja izazovan i veoma zastupljen problem u komercijalnom svetu. Rukovanje ogromnom količinom podataka uz korišćenje postojećih alata postalo je nemoguće, te se javlja potreba za naprednijim pristupima u pretraživanju i filtriranju informacija. Mnoge veb stranice za e-trgovinu već uveliko koriste alate kao što su sistemi za preporuke, koji pomažu kupcima da u moru proizvoda pronađu one koje žele. Glavni ishod dobrog sistema za preporuke je povećanje lojalnosti kupaca, a time i povećanje zarade.

U ovom radu opisano je više različitih implementacija ovih sistema. Korišćeni skup podataka sadrži informacije o proizvodima koji se prodaju na Amazonu. Najpre je izvršena priprema i analiza odabranog skupa podataka, a zatim odvajanje bitnih atributa i njihovo kodiranje. Nakon toga, predstavljene su implementacije modela, njihovo testiranje i tumačenje rezultata. Cilj razvijanih pristupa je predviđanje ocene korisnika za određeni proizvod i davanje preporuka. Korišćene tehnike su prethodno teorijski opisane. Podaci na osnovu kojih modeli vrše predviđanja preuzeti su iz javno dostupnih baza podataka.

Ključne reči: e-trgovina, sistem za davanje preporuke, uzajamno filtriranje, predviđanje

Sadržaj

1 O radu **1**
1.1 Opis problema 1
1.2 Motivacija 2

2 Uvod u sisteme za preporuke **4**
2.1 Uvod 4
2.2 Istorijski pregled 6

3 Osnovni pojmovi sistema za preporuke **8**
3.1 Vrste sistema za preporuke 8
3.2 Sistemi za preporuke zasnovani na dubokom učenju 20
3.3 Mere sličnosti 22
3.4 Funkcije predviđanja 26
3.5 Evaluacija sistema za preporuke 28

4 Predloženi metodi **31**
4.1 Sistem za preporuke koji koristi uzajamno filtriranje i KNN 32
4.2 Sistemi za preporuke koji koriste tehnike KNN i SVD 35
4.3 Sistem za preporuke zasnovan na dubokom učenju 42
4.4 Hibridni sistemi za preporuke 42

5 Eksperimentalni rezultati **46**
5.1 O skupu podataka 46
5.2 Sistem za preporuke koji koristi uzajamno filtriranje i KNN 47
5.3 Sistemi za preporuke koji koriste tehnike KNN i SVD 51
5.4 Sistem za preporuke zasnovan na dubokom učenju 53
5.5 Hibridni sistemi za preporuke 54

6 Zaključak **57**

Bibliografija **59**

Glava 1

O radu

1.1 Opis problema

Sve do pojave veba, pretraživanje robe za kupovinu se zasnivalo na sopstvenom ukusu ili na preporuci osobe od poverenja. Uglavnom je proces kupovine tekao tako što se pregleda sva roba u radnji ili tržnom centru i stvari klasifikuju na: „ovo mi se sviđa", „ovo mi se ne sviđa". Na kraju se izlazi sa jednim ili više proizvoda koje vredi kupiti [6].

Tokom ranih godina veba, dok se kupovina onlajn kretala sporim tempom, bilo je manje onlajn prodavnica i njihov izbor proizvoda nije bio veliki. Stvari su se značajno promenile sa početkom Veb 2.0 oko 2004. godine. Tada su se i prodavnice i kupci podjednako „preselili" na veb, a radikalne promene grafičkog korisničkog interfejsa veb stranica, posebno onih koje su koristile AJAKS tehnologiju, omogućile su mnogo bogatije i impresivnije iskustvo pregledanja. Od tada pa nadalje, protok informacija je počeo značajno da se uvećava, da je čak i fizički sloj mreže u početku imao problema sa obezbeđivanjem tolike količine sadržaja [6].

Ubrzo, ograničenje brzine koje su nametnuli rani modemi ustupilo je mesto ADSL-u. Od tog trenutka pa nadalje, veb stranice su se nadmetale jedna sa drugom u prikazivanju bogatijeg grafičkog sadržaja, sa više slika i teksta. Informaciono doba je rođeno, a preopterećenost informacijama postala je uobičajena pojava [6].

Iako e-trgovina nije nužno omogućila preduzećima da proizvode više proizvoda, omogućila im je da kupcima pruže veći izbor. Tako, umesto 10000 knjiga u knjižari, kupci mogu da biraju među milionima knjiga u nekoj onlajn prodavnici. Ograničenje u broju potencijalnih stavki koje se mogu pogledati na vebu praktično ne postoji. Uvećanjem izbora, povećana je količina informacija koju korisnici moraju da obrade

pre nego što budu u mogućnosti da odaberu koji proizvod zadovoljava njihove potrebe. Iako se mogućnost da se potrošačima pruži veći izbor smatra prednošću, to za sobom povlači nove probleme. Na primer, javlja se sledeće pitanje: kako bi onlajn kupac trebalo da filtrira ogromnu količinu opcija koje se sada nude? Zadatak je postao sličan pronalaženju igle u plastu sena. Korišćenje metode verovanja sopstvenom ukusu ili ukusu bliskog prijatelja više ne pomaže. Vidljivost informacija je znatno smanjena. Postavlja se pitanje kako otkriti nove proizvode i kako među milionima takvih proizvoda pronaći onaj koji se traži. Pretražiti pojam i za rezultat očekivati razumnu količinu proizvoda više nije realno očekivanje. Pretraživači su dobri u vraćanju ključnih reči, ali rezultati nisu prilagođeni korisniku. Proces obrade datih informacija i donošenje odluke šta kupiti postao je mukotrpan [6].

1.2 Motivacija

Onlajn prodavnice su se rano suočile sa problemom ogromne količine informacija, shvatajući da će ih vremenom biti sve više i da će kupcima biti potrebna pomoć da pronađu proizvode koje žele. Da bi rešile ovaj problem, e-prodavnice primenjuju različite principe, tehnike i alate kako bi potrošačima pružile što bolje korisničko iskustvo i preporuku o tome šta bi mogli sledeće da pregledaju [1].

Osnovna ideja koja stoji iza ovih algoritama je jednostavna: analiza korisničkih podataka, njegovih kupovina, ocena i odnosa sa drugim korisnicima može poslužiti kao osnov prema kome se korisniku mogu ponuditi drugi proizvodi koji će mu se potencijalno dopasti. Takođe, korisniku se nude i informacije koje mogu da mu pomognu da odluči koje proizvode da kupi [1].

Alati koji se bave pomenutom problematikom nazivaju se **sistemi za preporuke** (eng. **recommender systems, RS**). Sistemi za preporuke su za kratko vreme prešli put od istraživačkih projekata do potpune komercijalizacije. Cilj sistema za preporuke je da nauči preferencije ciljnog korisnika, sa namerom da mu preporuči relevantne proizvode [1].

Upotreba sistema za preporuke je postala značajan i neizostavan deo e-industrije. Onlajn prodavnice koje poseduju ove sisteme su posećenije u odnosu na one tradicionalne. Sistemi za preporuke u prodavnicama za elektronsku trgovinu predstavljaju poslovni alat koji dovodi do povećanja zarade. Istraživači, kao i vlasnici prodavnica za e-trgovinu traže načine kako da unaprede performanse preporučivanja. Dobar sistem za preporuke u tome zasigurno pomaže, jer korisnici mogu da naprave izbor

bez iscrpne pretrage. Danas je teško pronaći onlajn prodavnicu koja ne koristi ove sisteme u pozadini, učeći preferencije korisnika i povezivajući ih sa listom dostupnih proizvoda. To je razlog zašto sistemi za preporuke imaju značajnu ulogu na platformama kao što su Amazon, Facebook, YouTube i mnoge druge [1, 4].

Glava 2

Uvod u sisteme za preporuke

2.1 Uvod

Sistem za preporuke predstavlja softverski alat koji koristi skup tehnika i algoritama za davanje preporuka [1]. Sistemi za preporuke su sastavni deo raznih aplikacija, gde pokušavaju da korisnicima pruže tačnu i pouzdanu preporuku tako da zadovolje njihove potrebe i doprinesu razvoju poslovanja kompanija. Sistemi za preporuke imaju značajnu ulogu u procesima koji se tiču donošenja odluka (eng. decision-making processes), kao što su: koje proizvode kupiti, koju muziku slušati, ili koju knjigu pročitati [1].

Stavka (eng. item) je opšti pojam koji se koristi da označi šta sistem preporučuje korisnicima. Jedan sistem za preporuke se obično specijalizuje za određene stavke (na primer isključivo za vesti ili muziku), i koristi tehnike za davanje preporuka koje su prilagođene tako da pružaju korisne i efektivne sugestije samo za tu specifičnu vrstu stavki [1].

Sistemi za preporuke su primarno namenjeni pojedincima kojima nedostaje adekvatno znanje ili iskustvo da bi ocenili potencijalno ogroman broj alternativa koje veb stranica može da nudi. Osim što uz sistem za preporuke kupci mogu lakše da pronađu proizvode koji im se sviđaju, sistem kupcima predstavlja i proizvode o kojima nisu ni razmišljali, ali koji zapravo odgovaraju njihovim potrebama. Zbog ove prednosti, sistemi za preporuke mogu biti i značajniji od standardne pretrage [1].

Preporučivanje proizvoda može biti zasnovano na sveukupno najprodavanijim proizvodima u onlajn prodavnici, na demografiji potrošača ili na analizi kupovine potrošača u prošlosti kako bi se predvidelo njegovo ponašanje u budućnosti [1, 4].

Na primer, na veb stranici Amazon.com, sistem za preporuke prilagođava onlajn

prodavnicu svakom kupcu posebno. Naravno, u virtuelnom svetu, sve što se menja jeste odabir proizvoda koji se prikazuju kupcu, a ne i fizička radnja koja stoji iza toga. Ovakve preporuke zovu se personalizovane (eng. personalized) [1].

Kod njih, sistemi za preporuke prikupljaju informacije od korisnika vezano za njegove preferencije i u skladu sa tim, oni nude personalizovane preporuke koje su najčešće predstavljene u vidu rang listi (eng. ranked lists) stavki. Personalizovani sistemi za preporuke se sastoje iz više delova koji međusobno interaguju, a to su: metode za obradu podataka, modeli korisnika, tehnike filtriranja i razne metrike. Osim ovih, postoje i ne-personalizovane preporuke (eng. non-personalized), koje se mnogo lakše dobijaju. Tipični primeri uključuju „deset najboljih knjiga", „tri najgledanija filma" i slično [1].

Preferencije korisnika mogu biti izražene eksplicitno ili implicitno. Najmerodavnija je eksplicitna povratna informacija (eng. feedback), gde korisnici direktno iskazuju interesovanje za neki proizvod. Na primer, Netflix prikuplja ocene u vidu zvezdica (eng. star ratings) za filmove, dok TiVo korisnici iskazuju svoje preferencije za TV emisiju tako što biraju *thumbs-up* ili *thumbs-down* dugme. S obzirom na to da eksplicitna povratna informacija nije uvek dostupna, neki sistemi za preporuke donose zaključke o preferencijama korisnika preko obilnije implicitne povratne informacije i to kroz posmatranje ponašanja korisnika. Tipovi implicitne povratne informacije uključuju: istoriju kupovine, istoriju pregledanja, šablone pretraživanja (eng. search patterns), ili čak kretanje kursora po ekranu. Na primer, iz prethodnih kupovina korisnika, u kojima je on kupio mnogo knjiga od istog autora, može se zaključiti da on verovatno voli tog autora i da bi kupio još njegovih knjiga ukoliko mu se preporuče. U ovom radu fokus je na modelima prilagođenim za rad sa eksplicitnim povratnim informacijama. Ipak, time se ne umanjuje značaj implicitnih povratnih informacija, koje su od posebne važnosti onda kada korisnik ne pruža dovoljno eksplicitnih informacija [1].

Generalno, algoritmi razvijeni za sisteme za preporuke se oslanjaju na kupovine i preglede stranica koje su se prethodno desile. Na primer, ukoliko je korisnik pogledao neku kameru na veb stranici, sistem može da nauči (odnosno zaključi) da je korisnik zainteresovan za kamere i onda mu preporuči još sličnih proizvoda. Međutim, to ne mora da se poklapa sa pravom namerom korisnika. Moć sistema za preporuke je veća ukoliko on ima više informacija od korisnika – informacije tokom jedne sesije i informacije tokom više sesija. Da bi se rešio ovaj prolem, koriste se sistemi za preporuke svesni sesije (eng. session-aware recommender systems). Ova-

kvi sistemi mogu da razumeju kratkoročni cilj (eng. short-term goal) korisnika i njegove dugoročne preferencije (eng. long-term preference) u cilju da se prema tome preporuče odgovarajući proizvodi [1].

Skorašnja istraživanja primenjuju nove metode za razvijanje sistema za preporuke zasnovane na dubokom učenju. Osim mašinskog učenja i statistike, razvoj ovih sistema uključuje i tehnike obrade prirodnih jezika (eng. natural language processing, NLP) [5].

Uspeh sistema za preporuke je danas evidentan. To se može videti i kroz jedno od Guglovih najbitnijih poboljšanja njihovog pretraživača (eng. search engine) – praćenje istorije pregledanja (eng. browsing history tracking). Naime, naučene navike korisnika pri pretraživanju su uspešno iskorišćene za preporučivanje reklama [1].

Mnoga istraživanja, kako akademska tako i u industriji, postignuta su na polju razvoja sistema za preporuke. I dok je oblast sada stara već 20 godina, razvijene tehnike ostaju jednostavne. Pored raznovrsnih algoritama koji su razvijeni, ističu se dve opšte kategorije. Jedna od njih se odnosi na preferencije korisnika i poznata je kao uzajamno filtriranje (eng. collaborative filtering, CF), a druga se odnosi na specifikacije stavki i poznata je kao filtriranje zasnovano na sadržaju (eng. content-based) [1, 6]. Obe kategorije su detaljno obrađene u nastavku rada.

2.2 Istorijski pregled

Jedan od prvih istrazivačkih projekata o sistemima za preporuke bio je GroupLens [22] iz 1994. godine za filtriranje vesti. Ovo je bio prvi sistem za preporuke koji je koristio uzajamno filtriranje. Njegova platforma je pružala personalizovano predviđanje ocena za UseNet [22] vesti, koristeći susede ciljnog korisnika kao strategiju. Ovo je kombinovano sa Pirsonovom korelacijom kao funkcijom sličnosti, kako bi na ocene suseda i srednju ocenu suseda bile dodate težine i time bilo izračunato konačno predviđanje [6].

Nakon GroupLens, nastao je Ringo [19] sistem za preporuke za muziku iz 1995. i Bellcore [21] sistem koji preporučuje video snimke iz iste godine. Ringo je koristio sličnu strategiju kao GroupLens, ali je umesto Pirsonove korelacije (korišćene kao funkcija sličnosti) koja koristi prosečne vrednosti, upotrebljena ograničena Pirsonova korelacija koja za računanje koeficijenata sličnosti koristi medijanu. Takođe, korišćene su tehnike klasterovanja kako bi bili identifikovani muzički žanrovi a potom i iskorišćeni u algoritmima predviđanja [6].

Ista istraživačka grupa koja je proizvela GroupLens, proizvela je i MovieLens [20] 1997. godine. Cilj ovog sistema za preporuke bio je da korisnicima preporuči odgovarajuće filmove na osnovu preferencija drugih korisnika (ljubitelja filmova) [6].

Prva komparativna analiza algoritama zasnovanih na susedstvu koji koriste uzajamno filtriranje izvršena je 1998. godine. U tom radu, autori su suprotstavili rezultate dobijene korišćenjem Pirsonove korelacije i kosinusne sličnosti, konstatujući da je prvi rezultat bolji od drugog [6].

Veliki pomak kod sistema za preporuke desio se kada je ova tehnologija usvojena od strane Amazona 1990-ih. Njihova implementacija se zasnivala na sličnostima proizvoda pre nego na sličnostima korisnika. Ovo je omogućilo kompaniji da plasira tvrdnje kao što su: „Korisnici koji su kupili ovaj proizvod takođe su kupili i ove proizvode." [6].

Još jedan značajan projekat poznat kao The Music Genome Project [18] pokrenut od strane Pandora.com [18] 2000. godine, sastojao se od sistema za preporuke koji je učio sličnosti između muzičkih žanrova umetnika i pesama i prema tome korisniku preporučivao melodiju (muziku) koja se poklapa sa njegovim ukusom i preferencijama. Iste godine nastao je Netflix.com [17] koji je pružao iznajmljivanje filmova onlajn i nudio preporuke filmova sa svojim sistemom za preporuke zvanim Cinematch [17]. Nekoliko godina kasnije, 2006. godine, Netflix je obećao nagradu od milion dolara za unapređenje postojećeg sistema za preporuke za 10% ili više, gde je za merenje korišćen koren srednje kvadratne greške. Nagrađen je bio tim BellKor's Pragmatic Chaos [16] tri godina kasnije. Važan doprinos takmičenja bilo je uvođenje pristupa zasnovanog na modelu (eng. model-based approaches) kod korišćenja tehnika uzajamnog filtriranja [6].

Glava 3

Osnovni pojmovi sistema za preporuke

3.1 Vrste sistema za preporuke

Postoji više načina filtriranja ulaznih podataka, pa samim time i više vrsta sistema za preporuke. Neke od najzastupljenijih tehnika koje ovi sistemi implementiraju, a koje će u nastavku detaljno biti opisane, su:

- Filtriranje zasnovano na sadržaju (eng. content-based filtering): Preporuke su zasnovane na prethodnim odabirima korisnika. Na primer, korisniku bi bila preporučena nova knjiga iz programiranja ukoliko je on prošle godine kupio mnogo knjiga na ovu temu.
- Uzajamno filtriranje (eng. collaborative filtering): Preporuke za svakog korisnika (ciljnog korisnika) se izračunavaju uzimajući u obzir preferencije drugih korisnika koji su ocenili proizvode slično kao ciljni korisnik.
- Filtriranje zasnovano na demografiji (eng. demographic filtering): Preporuke se daju na osnovu demografskih karakteristika korisnika. Demografski parametri uključuju godine, pol, nacionalnost i slično.
- Hibridno filtriranje (eng. hybrid filtering): Preporuke se formiraju kombinacijom prethodnih tipova filtriranja. Na primer, upotrebom filtriranja zasnovanog na sadržaju ili uzajamnog filtriranja i demografskog filtriranja.

3.1.1 Preporuke zasnovane na sadržaju

Sistem za preporuke koji koristi pristup zasnovan na sadržaju, korisniku preporučuje proizvode na osnovu njegovih prethodnih ocena i karakteristika proizvoda. Proizvodi koji imaju karakteristike slične proizvodima kojima je korisnik davao visoke ocene su oni koji bivaju preporučeni. Sadržaj ovde referiše na karakteristike ili atribute proizvoda. Ideja kod ovog pristupa je da uz svaki proizvod stoji i njegov opis ili ključne reči. Opis doprinosi tome da se razume kakve proizvode ciljni korisnik voli, ali i da se pronađu drugačiji proizvodi sa sličnim karakteristikama [1, 6].

Pristup zasnovan na sadržaju zahteva određenu količinu informacija u vezi karakteristika proizvoda, jer je to ono na šta se oslanja prilikom davanja preporuka (nije zasnovan na interakcijama korisnika). Na primer, za sistem za preporuke koji preporučuje filmove, za jednu stavku (film) relevantne informacije (karakteristike) bi bile: žanr, godina, direktor, glumci i slično. Prema [5], ceo proces preporučivanja zasnovanog na sadržaju se može podeliti na tri osnovna koncepta:

1. **Pretprocesiranje (eng. preprocessing) i izvlačenje karakteristika (eng. feature extraction)**: Cilj je transformisati skup podataka koji predstavlja opise stavki u vektorski prostor zasnovan na ključnim rečima (eng. keyword-based vector space) – terminima, a potom odrediti njihovu važnost. Ovo se postiže upotrebom modela „vreća reči" (eng. bag of words model), gde je svaki *dokument*[1] predstavljen kao skup reči, dakle, redosled reči nije značajan. Pre ovog koraka potrebno je očistiti podatke – eliminisati reči (poput zamenica, predloga i slično) koje se često javljaju u dokumentu (eng. stop-word removing). Postoje dva pristupa za smanjenje raznolikosti termina izvučenih iz nekog teksta (kroz svođenje termina na njihov osnovni/koreni oblik), kao i za izvlačenje fraza, i to su: odsecanje sufiksa (eng. stemming) i lematizacija[2] (eng. lemmatization). Konačno, da bi se odredila sličnost između opisa stavki, koristi se neka od mera sličnosti koje će biti opisane u sekciji 3.3. Koncepti koji se koriste da odrede značaj, odnosno važnost ključnih reči (termina) u dokumentu su:

    - Učestalost termina (eng. term frequency, TF): Predstavlja broj pojavljivanja datog termina t u razmatranom dokumentu. Ideja je da što se

[1] Dokument je skup termina.

[2] Lematizacijom se reč pretvara u lingvistički validnu lemu razmatranjem konteksta.

termin češće pojavljuje u dokumentu, to je značajniji za taj dokument. Formula za računanje TF data je u nastavku.

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}},$$

gde $f_{t,d}$ predstavlja broj pojavljivanja termina t u dokumentu d, a delilac predstavlja ukupan broj termina u dokumentu d.

- Inverzna učestalost dokumenta (eng. inverse document frequency, IDF): Koristi se da odredi relativnu važnost dokumenta. Ideja je dodeliti veće težine neuobičajenim terminima tj. onima koji nisu toliko prisutni u korpusu[3]. IDF se određuje na osnovu korpusa i opisuje korpus kao celinu, a ne pojedinačne dokumente. Formula za računanje IDF data je u nastavku.

$$IDF(t,D) = \log\left(\frac{N}{|\{d \in D : t \in d\}|}\right),$$

gde je D korpus, N ukupan broj dokumenata u korpusu, a delilac predstavlja broj dokumenata koji sadrže termin t. Ukoliko termin nije u korpusu, dolazi do deljenja nulom. Da bi se ovo sprečilo, delilac se uvećava za 1. Tada formula postaje:

$$IDF(t,D) = \log\left(\frac{N}{|\{d \in D : t \in d\}| + 1}\right).$$

- $TF - IDF$: Svaka reč ili termin ima svoju TF i IDF vrednost. Proizvod TF i IDF vrednosti termina predstavlja $TF - IDF$ težinu termina. Ideja je vrednovati one termine koji nisu uobičajeni u korpusu (imaju visok IDF), a pri tome imaju nezanemarljiv broj pojavljivanja u datom dokumentu (imaju visok TF). Što je veća $TF - IDF$ vrednost, termin je ređi i značajniji, i obrnuto. Formula za računanje $TF - IDF$ vrednosti data je u nastavku.

$$TF - IDF(t,d,D) = TF(t,d) \cdot IDF(t,D)$$

2. **Učenje profila korisnika zasnovano na sadržaju**: U ovom koraku koriste se povratne informacije korisnika (eksplicitne ili implicitne) u kombinaciji sa

[3]Korpus je skup dokumenata.

informacijama o opisima stavki, da bi se konstruisao skup podataka za obučavanje. Nad ovim podacima za obučavanje, konstruiše se profil korisnika koji predstavlja preferencije svakog korisnika.

3. **Filtriranje i preporučivanje**: Naučeni model iz prethodnog koraka prima ulazne podatke i daje listu preporuka za svakog korisnika.

3.1.1.1 Prednosti i mane filtriranja zasnovanog na sadržaju

Filtriranje zasnovano na sadržaju, za razliku od uzajamnog filtriranja, nije efikasno za velike skupove podataka. Međutim, za razliku od uzajamnog filtriranja, ovaj metod prevazilazi problem „nove stavke" jer i nove stavke sadrže opis. Takođe, sistem zasnovan na sadržaju je nezavisan od korisnika (eng. user independent), jer ne koristi ocene drugih korisnika [1, 6].

3.1.2 Preporuke zasnovane na uzajamnom filtriranju

Sistemi zasnovani na uzajamnom filtriranju prikupljaju informacije o interakcijama korisnik-stavka, kao što je to implicitno ili eksplicitno ocenjivanje, i to čuvaju u formi matrice ocena. Na osnovu tih ocena, ovi sistemi nastoje da predvide korisnost pojedinog proizvoda za određenog korisnika. To postižu na osnovu vrednovanja tog proizvoda od strane ostalih korisnika sistema [1, 6].

Mnoge veb stranice, a naročito stranice za elektronsku trgovinu, koriste tehnike uzajamnog filtriranja u svojim sistemima za preporuke da personalizuju korisničko iskustvo pregledanja. Tako je Amazon, koristeći ove tehnike, povećao prodaju za 29%, Netflix je povećao iznajmljivanje filmova za 60% i Google vesti su povećali broj klikova (eng. click-through rates) za 30.9% [10].

Kod sistema zasnovanih na uzajamnom filtriranju, ocene koje su korisnici dodelili proizvodima koriste se kao relativna prezentacija njihovih interesa i potreba. Za razliku od preporuka na osnovu sadržaja, ovi modeli ne sadrže podatke o proizvodima već se ocene dodeljene od strane ciljnog korisnika upoređuju sa ocenama koje su dodelili drugi korisnici sistema, pa se na osnovu toga određuje skup proizvoda od interesa. Glavna podela algoritama za uzajamno filtriranje je na [1, 6]:

- Algoritme zasnovane na memoriji (eng. memory-based algorithms),
- Algoritme zasnovane na modelu (eng. model-based algorithms).

Kod algoritama zasnovanih na memoriji ciljnom korisniku će među proizvodima koje još nije ocenio biti preporučeni oni koje su visoko ocenili njemu slični korisnici. Osim donošenja odluka razmatranjem sličnih korisnika, moguće je razmatrati i slične stavke. Za određivanje sličnosti između korisnika, odnosno stavki, mogu se koristiti neke od mera sličnosti iz sekcije 3.3. Za predviđanje ocena mogu se koristiti funkcije za predviđanje koje će biti opisane u sekciji 3.4. Algoritmi zasnovani na memoriji biće opisani u nastavku ove sekcije.

Kod algoritama zasnovanih na modelu koriste se razni algoritmi mašinskog učenja kako bi se predvidele ocene korisnika za neocenjene stavke. Algoritmi zasnovani na modelu biće opisani u nastavku.

3.1.2.1 Algoritmi zasnovani na memoriji

Sistemi koji koriste algoritme zasnovane na memoriji primenjuju statističke tehnike kako bi pronašli skup korisnika, poznatih kao susedi, koji imaju slične obrasce ponašanja kao ciljni korisnik (na primer, različite proizvode ocenjuju slično ili kupuju uglavnom isti ili sličan skup proizvoda). Takođe, ove tehnike se mogu iskoristiti i za nalaženje skupa sličnih stavki (na primer, proizvoda koji su slični kao proizvodi koje je korisnik već ocenio). Stoga, pod susedima se mogu podrazumevati susedi zasnovani na korisnicima ili susedi zasnovani na stavkama [1, 6]. Prema ovome se dele i varijante algoritma uzajamnog filtriranja zasnovanog na memoriji. Oba tipa, uzajamno filtriranje zasnovano na korisnicima i uzajamno filtriranje zasnovano na stavkama, biće opisana u nastavku ove sekcije.

Algoritmi zasnovani na memoriji rade sa $m \times n$ korisnik-stavka matricama ocena, gde je m broj korisnika i n je broj stavki. Ova matrica ocena će nadalje biti označena sa R. Svaka ćelija r_{ui} matrice R predstavlja ocenu za stavku i datu od strane korisnika u. Skup stavki ocenjen od strane korisnika u označava se kao I_u, a skup stavki ocenjen od strane oba korisnika u i v kao $I_u \cap I_v$ (ili kraće I_{uv}). Skup korisnika koji su ocenili stavku i označava se kao U_i, a skup korisnika koji su ocenili stavke i i j označava se kao $U_i \cap U_j$ (ili kraće U_{ij}). U praksi je matrica ocena R veoma retka, sa mnogo nepoznatih ulaza, s obzirom na to da je prosečan korisnik ocenio samo mali deo svih postojećih stavki [1, 6].

Najčešća aplikacija pristupa zasnovanog na memoriji je algoritam k najbližih suseda (eng. k-nearest neighbors, KNN), koji može biti zasnovan na korisnicima ili stavkama. Ovaj algoritam se može ugrubo opisati u dva koraka:

1. Pronalaženje k najsličnijih korisnika/stavki. Esencijalni deo ovog koraka je izbor odgovarajuće funkcije sličnosti koja meri rastojanje između svaka dva korisnika ili svake dve stavke. Mere sličnosti biće diskutovane u sekciji 3.3.

2. Predviđanje ocena koje bi ciljni korisnik dao svim neocenjenim stavkama koristeći ocene k korisnika pronađenih u prvom koraku. Varijante funkcija za predviđanje biće diskutovane u sekciji 3.4.

Uzajamno filtriranje zasnovano na korisnicima Osnovna ideja ovog metoda je predviđanje ocena na osnovu preferencija suseda ciljnog korisnika. Na primeru sistema koji preporučuje knjige, da bi knjiga bila preporučena nekom korisniku, sistem koji koristi uzajamno filtriranje zasnovano na korisnicima najpre pokušava da pronađe druge korisnike koji imaju slične preferencije (ocenili su istu knjigu slično). Zatim, samo knjige koje se sviđaju k najsličnijim korisnicima bivaju preporučene [1, 6]. Da bi slični korisnici bili pronađeni koriste se funkcije (mere) sličnosti koje su oblika $s(u, v)$ i koje računaju sličnost između vektora ocena korisnika u i v, odnosno između redova matrice ocena R. U praksi se koriste različite funkcije sličnosti, a odabir odgovarajuće je veoma bitan, s obzirom na to da različite funkcije sličnosti daju različite rezultate. Onda kada su izračunate sličnosti između korisnika i kada je definisana grupa k najsličnijih, mogu se predvideti ocene za bilo koji par korisnik-stavka. Ovo se postiže koristeći neku od varijanti funkcije za predviđanje (zasnovane na korisnicima).

Uzajamno filtriranje zasnovano na stavkama Pristup zasnovan na stavkama pruža predviđanja zasnovana na ocenama koje je dao ciljni korisnik stavkama sličnim ciljnim stavkama. Na primeru sistema koji preporučuje knjige, da bi se predvidela ocena ciljnog korisnika A za neku knjigu B, pronalazi se najsličnija knjiga (na primer S) knjizi B iz skupa već ocenjenih knjiga od strane korisnika A. Zatim, prosečna ocena (sa težinama) za knjigu S se koristi da se predvidi ocena za knjigu B [1, 6]. U ovom slučaju, funkcije sličnosti se računaju između kolona matrice ocena R, kako bi se pronašle slične stavke. S obzirom na to da su algoritmi zasnovani na stavkama slični algoritmima zasnovanim na korisnicima, slične varijante funkcije sličnosti se mogu razmatrati u ovom pristupu. Jedina razlika je ta što se funkcija sličnosti računa za dve stavke umesto za dva korisnika i računanje se vrše nad skupom korisnika koji su ocenili stavke i i j (što se označava kao $Ui \cap Uj$, ili U_{ij}). Jednom kada je sličnost

među stavkama izračunata, može se preći na predviđanje ocena. Za predviđanje se može iskoristiti neka od funkcija za predviđanje, zasnovana na stavkama.

Prednosti i mane tehnika zasnovanih na memoriji Glavna prednost tehnika zasnovanih na memoriji je to što su lake za implementaciju, dok pružaju relativno visok kvalitet predviđanja. Takođe, nezavisne su od sadržaja (eng. context independent). Međutim, ovi sistemi nailaze i na probleme kao što je mali broj ocena na mnogo stavki, odnosno retkost podataka i problem iniciranja početka (eng. cold start problem) koji se odnosi na uvođenje novog proizvoda i uvođenje novog korisnika. U oba slučaja nedostaju podaci, odnosno interakcije korisnik-stavka, a to je nešto na šta su sistemi na uzajamnom filtriranju zasnovani [1, 6].

3.1.2.2 Algoritmi zasnovani na modelu

Ovaj pristup pre svega koristi tehnike istraživanja podataka i mašinskog učenja da nauči model za predviđanje, koristeći skup podataka za obučavanje. Taj model treba da karakteriše ocenjivanje krajnjih korisnika. Da naprave predviđanja, metode zasnovane na modelu koriste formiran model, umesto da direktno koriste celu matricu korisnik-stavka [1, 8].

Tipični primeri metoda za filtriranje zasnovanih na modelu uključuju drveta odlučivanja (eng. decision trees), pravila pridruživanja (eng. association rules), Bajesove mreže (eng. Bayesian networks), latentne semantičke modele (eng. latent semantic models) i modele klasterovanja (eng. clustering model). U nastavku ove sekcije će, bez umanjenja značaja ostalih modela, biti opisana SVD (eng. singular value decomposition) tehnika, koja pripada latentnim semantičkim modelima. O ostalim nabrojanim modelima se može pročitati u radu [8].

SVD Dekompozicija singularnih vrednosti je metod linearne algrebre koji se u mašinskom učenju koristi kao tehnika za smanjenje dimenzionalnosti. SVD je tehnika faktorizacije matrice koja uzima matricu A dimenzija $m \times n$ i dekomponuje je na sledeći način [2, 11]:

$$SVD(A) = USV^T$$

gde je U leva ortogonalna singularna matrica dimenzija $m \times r$ koja predstavlja vezu između korisnika i latentnih faktora (karakteristika), S je $r \times r$ diagonalna matrica koja opisuje jačinu (značaj) latentnih faktora (vrednosti na dijagonali su

ne-negativni realni brojevi) i V je $r \times n$ dijagonalna desno singularna matrica, koja predstavlja sličnost između stavki i latentnih faktora [2, 11].

Latentni faktori su zapravo karakteristike stavki, na primer muzički žanr. SVD redukuje dimenziju matrice A izvlačenjem njenih značajnih informacija (latentnih faktora) i preslikavanjem svakog korisnika i svake stavke u r-dimenzionalni latentni prostor. Ovime je postignuta jasna reprezentacija veze između korisnika i stavki [2, 11].

U kontekstu sistema za preporuke, SVD se koristi kod uzajamnog filtriranja i primenjuje se nad matricom ocena (gde svaki red predstavlja korisnika, svaka kolona predstavlja stavku, a elementi ove matrice su ocene koje su date stavkama od strane tih korisnika). Najpre se pronalaze latentni faktori matrica iz procesa faktorizacije početne matrice korisnik-stavka-ocena. Zatim se data matrica dekomponuje, odnosno razlaže na manje matrice. Na primer, matrica koja ima 2000 korisika i 1000 proizvoda imaće čak 2 miliona ulaza (ocena). Umesto da se čuva ovakva matrica, ona se može razložiti na dve matrice – jednu od 2000 korisnika i 100 atributa i drugu od 1000 stavki i 100 atributa. Ove matrice sada imaju 200 000 i 100 000 ulaza.

Ukoliko je potrebno pristupiti određenoj oceni, odnosno polju velike matrice, dovoljno je izračunati skalarni proizvod odgovarajućih vektora (vektora koji predstavlja tog korinika i vektora koji predstavlja tu stavku) i dobiti odgovarajuću ocenu [2, 11].

Jedna od najčešćih primena SVD-a jeste redukcija dimenzionalnosti skupa podataka. Prednosti modela zasnovanih na faktorizaciji matrice su efikasnost, tačnost i skalabilnost. Međutim, ako su podaci strogo ne-linearni, ispostavlja se da ponašanje nije uvek dobro [2, 11].

Prednosti i mane algoritama zasnovanih na modelu Algoritmi zasnovani na modelu su brži nego algoritmi zasnovani na memoriji, jer vreme potrebno da se od modela zatraži da nešto predvidi je uglavnom mnogo manje od vremena potrebnog da se pretraži ceo skup podataka. Ipak, njihova mana je ta što su mnogi modeli složeni i nekada, iako u teoriji deluju dobro, ne mogu se prilagoditi stvarnim podacima. Takođe, mnogo je vremena potrebno za implementaciju i formiranje ovakvih modela kao i za njihovo ažuriranje, što ih čini nefleksibilnim [1, 8].

3.1.2.3 Ograničenja sistema koji koriste uzajamno filtriranje

Budući da je broj korisnika i stavki konstantno u porastu, prikupljanje ulaznih podataka i formiranje sistema za preporuke koji koristi uzajamno filtriranje, a koji tačno predviđa je veliki izazov [1, 6, 8]. Dva glavna problema kod uzajamnog filtriranja opisana su u nastavku.

Oskudnost podataka (eng. data sparsity) Uzajamno filtriranje zasnovano na korisnicima zavisi od eksplicitne povratne informacije, kao što su ocene date proizvodu od strane korisnika. Ulazna matrica podataka korisnik-stavka može imati samo nekoliko ocena u odnosu na ukupan broj stavki, iako su korisnici veoma aktivni. Dodatno, s obzirom na to da korisnici uglavnom ne ocenjuju stavke aktivno, računanje sličnosti nad ovakvim skupom stavki može biti izazovno. Ovi problemi dovode do loših preformansi, odnosno nedovoljno tačnih predviđanja sistema za preporuke. Čak je i problem iniciranja početka uzrokovan nedostatkom podataka. Kako uzajamno filtriranje daje preporuke na osnovu prethodnog ponašanja korisnika, to znači da predviđanje nije moguće sve dok novi korisnici ne ocene određeni broj stavki. Mnoge metode za prevazilaženje problema oskudnosti podataka uključuju pristupe zasnovane na sadržaju budući da se oni ne oslanjaju u potpunosti samo na ocene korisnika. Ovi pristupi koriste eksterne informacije o sadržaju da bi izvršili predviđanje za nove korisnike i stavke [1, 6, 8].

Skalabilnost podataka (eng. data scalability) S obzirom na to da su sistemi za preporuke dizajnirani da vode korisnike kroz ogromne kolekcije stavki, jedan od najvažnijih ciljeva takvog sistema je da može da se skalira kada su u pitanju pravi skupovi podataka. Kada broj postojećih korisnika i stavki dovoljno poraste, tradicionalni algoritmi zasnovani na uzajamnom filtriranju „pate" od problema skalabilnosti. Ovi algoritmi računaju sličnost između svaka dva korisnika/svake dve stavke, što dovodi do uskog grla (eng. bottleneck) u takvim sistemima. Sistemi za preporuke koji koriste algoritme uzajamnog filtriranja zasnovane na memoriji donekle rešavaju ovaj problem upotrebom KNN algoritma i predviđanjem uzimajući u obzir samo susede. Algoritmi uzajamnog filtriranja zasnovani na modelu, kao što su algoritmi klasterovanja, pristupaju problemu skalabilnosti tako što particionišu korisnike na male, a slične klastere, koristeći susedstvo, odnosno susedne particije za predviđanje. Tehnike za smanjenje dimenzionalnosti, kao što je SVD tehnika,

mogu se izboriti sa problemom skalabilnosti i brzo proizvesti kvalitetne preporuke. Međutim, ove tehnike podrazumevaju skupe operacije faktorizacije matrice [1, 6, 8].

3.1.3 Hibridni sistemi

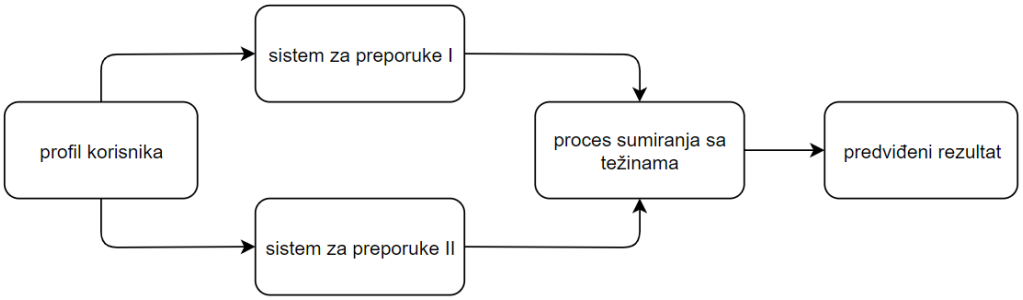
Svi do sada pomenuti algoritmi imaju različite prednosti i mane i svaki od tih algoritama je efikasan u različitim slučajevima. Na primer, sistemi koji koriste uzajamno filtriranje zavise od ocena korisnika, algoritmi zasnovani na sadržaju se oslanjaju na tekstualni opis stavki, a metode zasnovane na znanju (eng. knowledge-based methods) se oslanjaju na interakcije korisnika. Hibridni sistemi kombinuju različite tehnike, pokušavajući da iskoriste prednosti jedne i time poprave, odnosno zaobiđu, nedostatke druge tehnike. Na primer, metode za uzajamno filtriranje „pate" od problema „nova stavka" (eng. new item problem), te ne mogu da preporuče stavku koja nema ni jednu ocenu. Međutim, ovo nije nikakvo ograničenje za pristup zasnovan na sadržaju, budući da se predviđanje ocene za novu stavku bazira na njenom opisu (karakteristikama) koje su uglavnom lako dostupne [7, 8].

Postoje različite strategije hibridizacije koje se mogu primeniti, zavisno od načina kombinovanja različitih tehnika za davanje preporuka. Najčešće korišćen hibridni sistem je onaj koji kombinuje uzajamno filtriranje i metode zasnovane na sadržaju, gde se u obzir uzimaju karakteristike sadržaja i vrednovanje sadržaja od strane korisnika [8]. Neki od tipova hibridnih sistema koji se najčešće koriste opisani su u nastavku.

3.1.3.1 Hibridni sistem sa težinama

Ukoliko je potrebno zajedničko prikazivanje preporuka dobijenih primenom različitih tehnika, pribegava se korišćenju hibridnog sistema sa težinama (eng. weighted hybrid system). Izlaz, odnosno ocene dobijene od strane pojedinačnih sistema za preporuke, su kombinovane upotrebom težina. Ovakav hibridni pristup karakteriše dizajn ansambla, jer podrazumeva kombinovanje zasebnih metoda za preporučivanje. Na primer, inicijalno se mogu dodeliti jednake težine preporukama dobijenim koristeći uzajamno filtriranje i onim zasnovanim na sadržaju. A zatim se težine posebno prilagođavaju u zavisnosti od toga koja predviđanja ocena korisnika su tačne, odnosno netačne. Prednost ove vrste hibridnog sistema je što je u njemu objedinjeno nekoliko različitih modela koji učestvuju u procesu preporučivanja, dok je ceo

proces linearan [2, 7, 8]. Arhitektura sistema koji implementira hibridni pristup sa težinama je prikazana na slici 3.1.

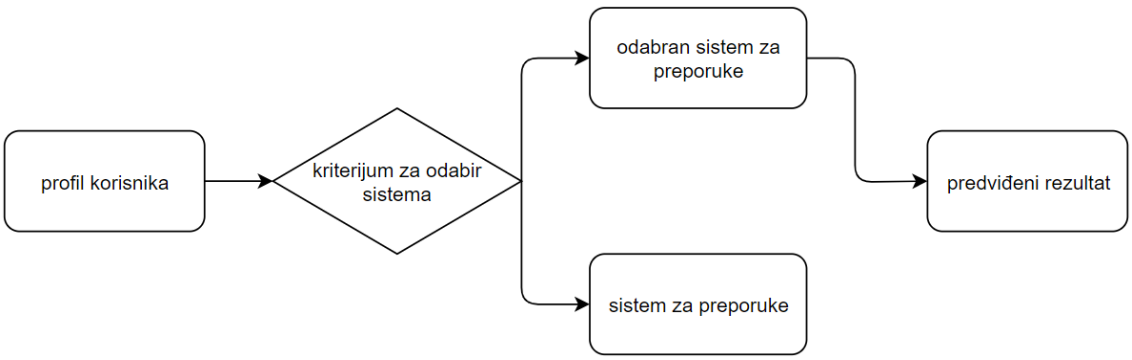


Slika 3.1: Hibridni sistem sa težinama

3.1.3.2 Hibridni sistem sa smenjivanjem

Smenjivanje modela podrazumeva da se na osnovu određenog kriterijuma vrši izmena korišćenih tehnika u nekoliko faza i da se primenom jedne tehnike dobija model koji služi kao ulaz za narednu tehniku. Hibridni sistem sa smenjivanjem (eng. switching hybrid system) bira tačno jedan sistem za preporuke u određenoj situaciji. Uglavnom se postavlja kriterijum na osnovu kojeg se bira sistem za preporuke prema profilu korisnika ili na osnovu nekih drugih karakteristika. U svakom slučaju, izabran je onaj sistem za preporuke koji je u datom trenutku „bolji" od drugog na osnovu neke mere za ocenjivanje kvaliteta preporuka. Na primer, ukoliko se kombinuju sistemi za preporuke zasnovani na sadržaju i oni koji koriste uzajamno filtriranje, prvo može biti odabran sistem za preporuke zasnovan na sadržaju a onda u nekom trenutku da se promeni i odabere sistem zasnovan na uzajamnom filtriranju. Kod ovog konkretnog hibridnog modela i dalje ostaje problem dodavanja novog korisnika, jer oba sistema, i sistem zasnovan na filtriranju i sistem zasnovan na sadržaju ne rešavaju ovaj problem [2, 7].

Korist hibridnog sistema sa smenjivanjem je to što se mogu iskoristiti prednosti svakog od sistema. Međutim, ovaj sistem sa sobom donosi i nivo složenosti u proces preporučivanja, budući da postoji dodatni kriterijum koji se mora odrediti, što uključuje još jedan nivo parametrizacije [2, 7]. Dizajn hibridnog sistema sa smenjivanjem prikazan je na slici 3.2.



Slika 3.2: Hibridni sistem sa smenjivanjem

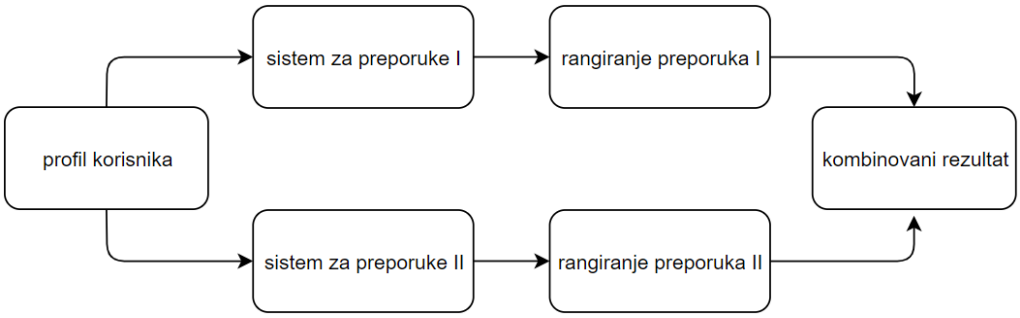
3.1.3.3 Mešoviti hibridni sistem

Kada je poželjno imati veći broj preporuka istovremeno, moguće je iskoristiti mešoviti hibridni sistem (eng. mixed hybrid system). Ovaj sistem podrazumeva da se preporuke dobijene primenom više različitih tehnika prikazuju zajedno.

PTV sistem[4] koristi ovaj pristup da sastavi preporučeni program za gledanje televizije. Korišćene su tehnike zasnovane na sadržaju koje koriste tekstualne opise TV emisija i tehnike zasnovane na uzajamnom filtriranju koje koriste informacije o preferencijama korisnika. Preporuke dobijene ovim tehnikama su kombinovane zajedno u konačni program koji se predlaže. Ovakav hibridni sistem uspešno prevazilazi problem „nova stavka" jer se može osloniti na sistem zasnovan na sadržaju, koji će preporučiti nove emisije prema njihovim opisima, iako one još nisu ocenjene. Međutim, ne zaobilazi se problem „novi korisnik" (eng. new user problem), budući da oba sistema, i sistem zasnovan na sadržaju i sistem zasnovan na uzajamnom filtriranju zahtevaju neke podatke o preferencijama korisnika kako bi otpočeli proces preporučivanja [2, 7, 9].

Nekada se kod ovih sistema više preporuka prikazuju jedna do druge. Međutim, nekada je potrebno rangirati preporuke (preporučene stavke) ili čak odabrati i prikazati samo jednu, najbolju preporuku, u kom slučaju se primenjuje neka tehnika za kombinaciju [2, 7]. Sistem za preporuke koji implementira arhitekturu mešovitog hibrida, prikazan je na slici 3.3.

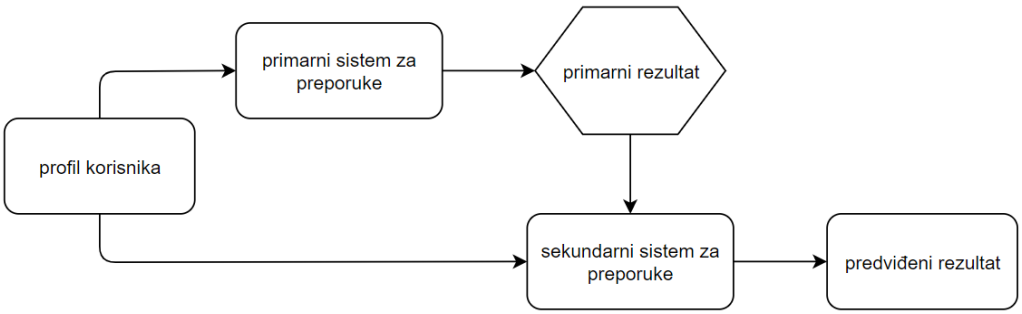
[4]Cotter Smyth's PTV (2000) je primer hibridnog sistema za preporuke koji za učenje preferencija korisnika koristi rezonovanje zasnovano na slučaju (eng. case-based reasoning) i uzajamno filtriranje [9].



Slika 3.3: Mešoviti hibrid

3.1.3.4 Kaskadni hibridni sistem

Hibridni sistem koji definiše strogu hijerarhijsku strukturu sistema za preporuke nazvan je kaskadni hibridni sistem (eng. cascade hybrid system). Arhitektura ovog sistema podrazumeva da glavni sistem za preporuke daje primarni rezultat. Nakon njega se koristi sekundarni model kako bi rešio neke manje probleme primarnog rezultata, kao što je to „razbijanje" izjednačavanja u ocenjivanju. U praksi, većina skupova podataka je oskudna, pa sekundarni model sistema za preporuke može biti efikasan po pitanju problema jednakog ocenjivanja (eng. equal scoring issue) ili problema sa nedostajućim podacima (eng. missing data issue) [2, 7]. Arhitektura kaskadnog hibridnog sistema je prikazana na slici 3.4.



Slika 3.4: Kaskadni hibridni sistem

3.2 Sistemi za preporuke zasnovani na dubokom učenju

Nedavno, duboko učenje je pokrenulo revoluciju u razvoju sistema za preporuke i značajno poboljšalo njihove performanse. Poslednja otkrića na polju sistema

za preporuke zasnovanim na dubokom učenju prevazilaze prepreke tradicionalnih modela, a njihove preporuke dostižu visok kvalitet.

Duboko učenje je podoblast mašinskog učenja koja se bazira na veštačkim neuronskim mrežama. Ove mreže imaju sposobnost da uče karakteristike (engl. feature learning) podataka, odnosno da izdvoje određene karakteristike koje su od značaja. To znači da se vrši transformacija sirovih ulaznih podataka u reprezentacije koje mogu efikasno biti iskorišćene u zadacima mašinskog učenja. Ovaj proces naziva se izvlačenje atributa i smatra se da je to jedan od najbitnijih razloga za delotvornost neuronskih mreža. Ovi modeli izuzetno su fleksibilni i imaju široku primenu. Koriste se za prepoznavanje govora, prevođenje, prepoznavanje oblika na slikama, uspostavljanje dijagnoza u medicini, igranje igara, upravljanje vozilima, itd. Tehnikama dubokog učenja je moguće efikasno obuhvatiti nelinearne i netrivijalne odnose korisnika i stavke, koristeći dostupne izvore podataka kao što su kontekstualne, tekstualne i vizuelne informacije.

Neuronske mreže se sastoje od velikog broja skrivenih slojeva (odakle i potiče naziv duboko učenje), koji se sastoje iz perceptrona[5] i njihovih međusobnih veza. Karakteriše ih sposobnost obučavanja nad velikim skupovima podataka kao i to što im se mogu predati neobrađeni podaci. Ovime se rešavaju dva velika problema klasičnih algoritama mašinskog učenja. Perceptroni uvode nelinearnost u sistem pomoću odgovarajuće aktivacione funkcije, dok težinski koeficijenti u vezama između perceptrona kombinuju rezultate više perceptrona. Težinski koeficijenti predstavljaju parametre koje neuronska mreža uči i u njima se sadrži „znanje“ modela. Perceptron predstavlja osnovnu gradivnu komponentu neuronske mreže. Veštački neuroni su inspirisani neuronima koji se nalaze u ljudskom mozgu. Perceptron dobija ulazni podatak, transformiše ga i prosleđuje ga drugim perceptronima koji ponavljaju isti postupak. Na ovaj način se transformacija vrši u više slojeva tokom čega se postiže željena nelinearnost između ulaznog i izlaznog vektora. Način na koji će transformacija biti obavljena zavisi od izbora aktivacione funkcije perceptrona. Kako bi se postigla nelinearnost sistema, aktivaciona funkcija perceptrona mora biti nelinearna. Postoji nekoliko često korišćenih funkcija, a to su: ReLu funkcija (engl. rectifying linear unit, ReLU), propustljivi ReLU (eng. leaky ReLU), sigmoid i tangens hiperbolički. U izlaznom sloju vrlo često korišćena funkcija je softmax. Ova funkcija, koja se još naziva i normalizovana eksponencijalna funkcija, predstavlja generalizaciju logističke funkcije na više dimenzija. Kod višeklasne klasifikacije, če-

[5]Perceptron je veštački neuron.

sto se koristi u izlaznom sloju neuronske mreže jer se njen rezultat može posmatrati kao verovatnoća tačnosti određene klasifikacije.

Postoji više poznatih arhitektura neuronskih mreža. Arhitektura koja je iskorišćena u ovom radu biće opisana u poglavlju 4. Više o neuronskim mrežama i poznatim arhitekturama može se pročitati u radovima [14] i [15].

3.3 Mere sličnosti

U srcu algoritma uzajamnog filtriranja leži stroga pretpostavka da se „sličnim korisnicima sviđaju slične stavke". Iako se čini da je ova pretpostavka u velikoj meri istinita, ključno pitanje postaje: kako definisati i identifikovati slične korisnike, da bi se na osnovu toga pronašle slične stavke koje mogu biti preporučene? U obzir treba uzeti podatke koji su na raspolaganju – matricu korisnik-stavka, koja je uglavnom retka, jer je većina korisnika ocenila samo mali deo mnogobrojnih stavki (na primer, proizvoda u prodavnici).

U GroupLens sistemu za preporuke, za odgovarajuću meru za računanje sličnosti korisnika predložena je Pirsonova korelacija (eng. Pearson correlation, PC). Osim ove, postoje i druge mere sličnosti. Neke od njih su Euklidsko rastojanje (eng. Euclidean distance, ED) i kosinusno rastojanje (eng. cosine distance). O tome koja je najbolja mera vodi se otvorena rasprava, gde neki autori ističu da je Pirsonova korelacija najpogodnija za pristup zasnovan na sličnostima korisnika, dok je kosinusno rastojanje najbolji izbor za pristup zasnovan na sličnostima stavki [1, 2, 6]. U nastavku su detaljno opisane funkcije sličnosti i odgovarajuće formule. Korisnici i stavke su predstavljeni kao vektori atributa.

3.3.1 Euklidsko rastojanje

Euklidsko rastojanje meri sličnost dva vektora tako što računa rastojanje između njih. Euklidsko rastojanje između dva korisnika se računa po formuli:

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

gde $ED(u,v)$ predstavlja sličnost korisnika u i v, za sve stavke koje su zajednički ocenjene, a r_{ui} i r_{vi} predstavljaju ocene dodeljene stavki i od strane korisnika u i v, respektivno. Naredna formula predstavlja Euklidsko rastojanje između dve stavke:

$$ED(i,j) = \sqrt{\sum_{u \in U_{ij}} (r_{iu} - r_{ju})^2}$$

gde $ED(i,j)$ predstavlja sličnost stavki i i j, za sve korisnike koji su ocenili te stavke, a r_{iu} i r_{ju} predstavljaju ocene koje je korisnik u dodelio stavkama i i j, respektivno. U nekim implementacijama sistema za preporuke, Euklidska sličnost je definisana kao:

$$\hat{ED}(a,b) = \frac{1}{1 + ED(a,b)}$$

gde $\hat{ED}(a,b)$ predstavlja meru Euklidske sličnosti, a $ED(a,b)$ predstavlja euklidsko rastojanje. Efekat ove formule je ograničenje ocene na interval (0..1] [1, 2, 6].

3.3.2 Pirsonova korelacija

Ovu meru je predložio Karl Pirson [23] za merenje linearne korelacije između dve (neprekidne) promenljive i ona je postala veoma široko korišćena u domenu statistike. Što se tiče sistema za preporuke, mera je prvi put uvedena od strane projekta GroupLens 1994. godine, i od toda se svuda koristi kao osnov za poređenje sa drugim merama [6].

Pirsonov koeficijent korelacije predstavlja linearnu korelaciju između korisnika (odnosno stavki), a izračunava se kao količnik kovarijanse dva korisnika i standardne devijacije između njih. Vrednosti koeficijenta korelacije kreću se u rasponu od -1 do +1. Predznak pokazuje da li je korelacija pozitivna (obe promenljive zajedno i opadaju i rastu) ili negativna (jedna promenljiva opada kada druga raste i obrnuto). Apsolutna vrednost tog koeficijenta pokazuje jačinu veze. Korelacija koja iznosi +1 ili -1, pokazuje da se vrednost jedne promenljive može tačno utvrditi ukoliko se zna vrednost druge (+1 predstavlja strogo pozitivnu korelaciju, -1 predstavlja strogo negativnu korelaciju). S druge strane, korelacija jednaka nuli pokazuje da između te dve promenljive ne postoji nikakva veza, te poznavanje vrednosti jedne promenljive nimalo ne pomaže u predviđanju vrednosti druge. Jednačina Pirsonove korelacije za računanje sličnosti između dva korisnika definiše se kao:

$$PC(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r_v})^2}}$$

gde $PC(u, v)$ predstavlja sličnost korisnika u i v, I_{uv} predstavlja skup stavki ocenjenih od strane oba korisnika u i v, a $\bar{r_u}$ i $\bar{r_v}$ predstavljaju srednju ocenu korisnika u odnosno v, u odnosu na sve ocenjene stavke. U slučaju sistema za preporuke zasnovanih na stavkama, formula je sledeća:

$$PC(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \bar{r_i})(r_{uj} - \bar{r_j})}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \bar{r_i})^2} \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \bar{r_j})^2}}$$

gde $PC(i, j)$ predstavlja sličnost stavke i i j, U_{ij} predstavlja skup zajedničkih korisnika koji su ocenili stavke i i j, a $\bar{r_i}$ i $\bar{r_j}$ koji predstavljaju srednju ocenu stavke i odnosno j, u odnosu na sve korisnike koji su ocenili te stavke [1, 2, 6].

3.3.3 Kosinusna sličnost

Upotrebom kosinusne sličnosti (rastojanja) određuje se sličnost između dva vektora (dve stavke/dva korisnika), i to tako što se računa kosinus ugla između njih. Kosinusna sličnost se često koristi da izmeri sličnost dokumenata prilikom analize teksta. Osim toga, ima primenu i kod sistema za preporuke. Skala vrednosti koja se koristi kod kosinusne sličnosti ista je kao ona koja se koristi kod Pirsonove korelacije, gde +1 i -1 predstavljaju visoku (direktnu i inverznu) korelaciju. Naime, +1 označava da su vektori 100% slični, dok 0 predstavlja nekorelisanost. U formi vektora, kosinusna sličnost je definisana kao:

$$\cos(a, b) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}|^2 |\vec{b}|^2}$$

gde $cos(a, b)$ predstavlja sličnost vektora a i b, a a i b su dati vektori. Jednačina kosinusne sličnosti koja računa sličnost između dva korisnika definiše se kao:

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

gde $CD(u, v)$ predstavlja sličnost korisnika u i v, preko svih zajednički ocenjenih stavki, I_u predstavlja skup stavki ocenjen od strane korisnika u, I_v predstavlja skup stavki ocenjen od strane korisnika v. Kosinusna sličnost između dve stavke računa se po formuli:

$$CD(i,j) = \frac{\sum_{u \in U_{ij}} r_{iu} r_{ju}}{\sqrt{\sum_{u \in U_i} r_{iu}^2 \sum_{u \in U_j} r_{ju}^2}}$$

gde $CD(i, j)$ predstavlja sličnost stavki i i j, preko svih korisnika koji su ocenili takve stavke, U_i predstavlja skup zajedničkih korisnika koji su ocenili stavku i, U_j predstavlja skup zajedničkih korisnika koji su ocenili stavku j [1, 2, 6].

3.3.4 Prilagođena kosinusna mera sličnosti

Prilagođena kosinusna mera sličnosti (eng. adjusted cosine similarity measure, AC) je modifikovani oblik vektorske (kosinusne) sličnosti gde se u obzir uzima i činjenica da različiti korisnici imaju različite šeme ocenjivanja. Drugim rečima, neki korisnici mogu generalno visoko ocenjivati stavke, dok drugi mogu dati niže ocene iako im se proizvod sviđa. Da bi se uklonio ovaj nedostatak iz sličnosti zasnovane na vektorima, oduzima se prosečna ocena od svake pojedinačne ocene, odnosno od svake ocene korisnika za svaku stavku. U ovom slučaju, formula kod pristupa zasnovanog na korisnicima postaje:

$$AC(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_i})(r_{vi} - \bar{r_i})}{\sqrt{\sum_{i \in I_u} (r_{ui} - \bar{r_i})^2 \sum_{i \in I_v} (r_{vi} - \bar{r_i})^2}}$$

gde $AC(u, v)$ predstavlja sličnost korisnika u i v za sve stavke koje su zajednički ocenjene. U slučaju sistema za preporuke zasnovanih na stavkama, formula postaje:

$$AC(i, j) = \frac{\sum_{u \in U_{ij}} (r_{iu} - \bar{r_u})(r_{ju} - \bar{r_u})}{\sqrt{\sum_{u \in U_i} (r_{iu} - \bar{r_u})^2 \sum_{u \in U_j} (r_{ju} - \bar{r_u})^2}}$$

gde $AC(i, j)$ predstavlja sličnost stavki i i j za sve korisnike koji su ocenili te stavke [1, 2, 6].

3.3.5 Žakardova sličnost

Žakardov indeks (eng. Jaccard index, J) određuje sličnost, odnosno različitost dva skupa. Žakardov indeks između dva konačna skupa je definisan kao količnik kardinalnosti preseka i kardinalnosti unije. Odnosno, ova sličnost meri udeo broja elemenata koji su zajednički za dva skupa u odnosu na ukupan broj elemenata u oba skupa. Vrednost Žakardovog indeksa je u opsegu između 0 i 1. Što je vrednost bliža 1, data dva vektora su sličnija. Razultat je 0 ukoliko dva vekora nemaju nikakvih sličnosti.

Žakardov indeks ignoriše vrednosti ocena, proveravajući samo da li je korisnik izrazio preferenciju ili ne. Ova mera računa sličnost između dva korisnika tako što

računa ukupan broj jedinstvenih stavki za koje je bar jedan od dva korisnika zainteresovan i to deli sa brojem stavki koje su zajednički ocenjene. Dakle, dva korisnika će biti sličnija kada imaju više stavki koje su zajedno ocenili [1, 2, 6]. Jednačina koja računa Žakardov indeks (koeficijent) sličnosti između korisnika (vektora) u i v, data je u nastavku:

$$J(u,v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}.$$

3.3.6 Srednje kvadratno rastojanje

Sličnost preko srednje kvadratnog rastojanja (eng. mean square distance, MSD) se definiše kao:

$$MSD(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2}{|I_{uv}|}$$

gde $MSD(u,v)$ predstavlja sličnost korisnika u i v za sve stavke koje su zajednički ocenjene. U slučaju sistema za preporuke zasnovanog na stavkama, formula je sledeća:

$$MSD(i,j) = \frac{\sum_{u \in U_{ij}} (r_{iu} - r_{ju})^2}{|U_{ij}|}$$

gde $MSD(i,j)$ predstavlja sličnost stavki i i j za sve korisnike koji su ocenili ove stavke. U nekim radovima [1] se preporučuje upotreba inverznog srednje kvadratnog rastojanja:

$$\hat{MSD}(a,b) = \frac{1}{MSD(a,b)}$$

gde $\hat{MSD}(a,b)$ predstavlja srednje kvadratnu sličnost, a $MSD(a,b)$ predstavlja srednje kvadratno rastojanje [1, 2, 6].

3.4 Funkcije predviđanja

Nakon odabira suseda, poslednji korak u procesu preporučivanja je predviđanje ocena. Funkcije za predviđanje podrazumevaju različite načine za procenu nepoznatih ocena za stavke koje su ocenjene od strane suseda ciljnog korisnika, ali nisu ocenjene od samog ciljnog korisnika. Sa listom procenjenih ocena za neocenjene stavke, sistem za preporuke jednostavno sortira listu u opadajući poredak po vrednosti

ocena i prvih n stavki preporučuje ciljnom korisniku. Tri najčešće korišćene varijante funkcije za predviđanje, koje će biti opisane u nastavku, prema [6] su:

- Težinska suma (eng. weighted sum),
- Pristup zasnovan na centriranju proseka (eng. mean centering approach),
- Pristup zasnovan na z-rezultatu (eng. z-score approach).

3.4.1 Težinska suma

Kod ovog pristupa, predviđanje se računa kao prosek svih dostupnih ocena, sa težinama. Za težinu se uzima vrednost korelacije (sličnosti) koja se izračunava pomoću funkcije sličnosti. Formula za računanje procene ocene u sistemu zasnovanom na korisnicima data je sa:

$$\hat{r}_{ui} = \frac{\sum_{v \in K_i(u)} s(u,v) r_{vi}}{\sum_{v \in K_i(u)} |s(u,v)|}$$

gde $K_i(u)$ predstavlja broj suseda koji imaju stavku i zajedničku sa korisnikom u, od kojih je v jedan takav sused, r_{vi} je ocena korisnika v za stavku i, a $s(u,v)$ predstavlja sličnost između korisnika u i jednog od njegovih suseda v. Za sisteme zasnovane na stavkama, formula je sledeća:

$$\hat{r}_{ui} = \frac{\sum_{j \in K_u(i)} s(i,j) r_{ju}}{\sum_{j \in K_u(i)} |s(i,j)|}$$

gde $K_u(i)$ predstavlja broj susednih stavki za korisnika u, od kojih je j jedan takav sused, r_{ju} je ocena korisnika u za stavku j, a $s(i,j)$ predstavlja sličnost stavke i i jedne od njenih suseda – stavke j [6].

3.4.2 Pristup zasnovan na centriranju proseka

Upotrebom pristupa zasnovanog na centriranju proseka, formula za sisteme koji su zasnovani na korisnicima je sledeća:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in K_i(u)} s(u,v)(r_{vi} - \bar{r}_v)}{\sum_{v \in K_i(u)} |s(u,v)|}$$

gde $\hat{r}_{ui}$ predstavlja ocenu korisnika u za stavku i, a $\hat{r}_v$ je prosečna ocena korisnika v. Za sisteme zasnovane na stavkama, fromula je sledeća:

$$\hat{r}_{ui} = \bar{r}_i + \frac{\sum_{j \in K_u(i)} s(i,j)(r_{ju} - \bar{r}_j)}{\sum_{j \in K_u(i)} |s(i,j)|}$$

gde $\hat{r}_{ui}$ predstavlja ocenu korisnika u za stavku i, a $\hat{r}_j$ je prosečna ocena za stavku j [6].

3.4.3 Pristup zasnovan na z-rezultatu

Formula koja predstavlja varijaciju prethodne funkije, u kojoj se koristi z-rezultat, za sisteme zasnovane na korisnicima, definiše se na sledeći način:

$$\hat{r}_{ui} = \bar{r}_u + \sigma u \frac{\sum_{v \in K_i(u)} s(u,v)(r_{vi} - \bar{r}_v)/\sigma u}{\sum_{v \in K_i(u)} |s(u,v)|}$$

gde $\hat{r}_{ui}$ predstavlja ocenu korisnika u za stavku i, a σu je varijansa ocena korisnika u. Za sisteme zasnovane na stavkama, fromula je sledeća:

$$\hat{r}_{ui} = \bar{r}_i + \sigma i \frac{\sum_{j \in K_u(i)} s(i,j)(r_{ju} - \bar{r}_j)/\sigma i}{\sum_{j \in K_u(i)} |s(i,j)|}$$

gde $\hat{r}_{ui}$ predstavlja ocenu korisnika u za stavku i, a σu je varijansa ocena stavke i. Ovaj pristup je zasnovan na srednjoj vrednosti i varijansi (robusnim statistikama), što dovodi do umanjenja značaja vrednosti koje su van očekivanog opsega (eng. outliers) [6].

3.5 Evaluacija sistema za preporuke

Ukoliko je dato više različitih implementacija sistema za preporuke, potrebno je nekako izmeriti koji sistem je bolji u smislu efikasnosti, tačnosti predviđanja i drugih performansi. Evaluacija (eng. evaluation) sistema za preporuke podrazumeva upotrebu različitih mera za merenje performansi sistema u različitim aspektima. Zbog raznovrsnosti sistema za preporuke, ne postoji jedinstvena mera kojom se mogu oceniti svi aspekti sistema za preporuke. Naime, različite mere mogu biti kombinovane kako bi se izvršila celokupna evaluacija sistema [2, 3].

Sistemi za preporuke dele nekoliko konceptualnih sličnosti sa problemom klasifikacije i problemom regresije. Kod modela za klasifikaciju i regresiju treba predvideti klasu odnosno ciljnu promenljivu, koristeći preostale vrednosti atributa. U sistemima za preporuke, data je matrica ocena u kojoj mnoga polja (ocene) nedostaju i treba

ih predvideti, uzimajući u obzir ostatak polja (ocena) iz matrice. U ovom smislu, sistemi za preporuke mogu biti posmatrani kao generalizacija problema klasifikacije. Stoga, mnoge mere koje se koriste za evaluaciju modela za klasifikaciju, uz određene modifikacije, mogu biti iskorišćene i kod sistema za preporuke. U ovoj sekciji, fokus je na merenju tačnosti, koja se smatra za jedan od najvažnijih kriterijuma. Ipak, najpre će biti objašnjeno prema čemu se sve može vršiti evaluacija [2].

Sistemi za preporuke mogu biti evaluirani u pogledu tačnosti predviđenih ocena ili u pogledu tačnosti rangiranja stavki. Što se tiče tačnosti predviđenih ocena, koriste se mere kao što su srednje apsolutna greška (eng. mean absolute error, MAE) i koren srednje kvadratne greške (eng. root mean squared error, RMSE). Ove mere će detaljno biti opisane u sekciji 3.5.1. Evaluacija rangiranja se može izvršiti korišćenjem različitih metoda, kao što su proračuni zasnovani na korisnosti (eng. utility-based computations), koeficijenti korelacije ranga (eng. rank correlation coefficients) i ROC krive (eng. receiver operating characteristic curve). Više o ovim merama može se pročitati u radu [2], u poglavljima 7.5.2–7.5.4.

3.5.1 Merenje tačnosti predviđenih ocena

U ovom radu je fokus na merenju kvaliteta odnosno tačnosti predviđenih ocena i to onih koje su dobijene oslanjajući se na eksplicitne povratne informacije, gde korisnik direktno izražava svoje preferencije koristeći ocene. Za ovu evaluaciju najčešće se koriste već pomenute mere: srednje apsolutna greška i koren srednje kvadratne greške. Ove mere su zasnovane na razlici između ocene koja je predviđena za neku stavku i ocene koju je korisnik zaista dao toj stavci [2, 3]. Obe mere će biti detaljno opisane u nastavku.

Za ove mere važe sledeća tvrđenja. Tačnost se meri na skupu za testiranje. Neka je S skup svih razmatranih podataka, a $E \subset S$ skup podataka korišćenih za testiranje. Svaki podatak skupa E je uređeni par korisnik-stavka oblika (u, j), što odgovara poziciji u matrici ocena. Neka je r_{uj} vrednost prave ocene na poziciji $(u, j) \in E$ i neka je $\hat{r}_{uj}$ ocena na poziciji (u, j) koja je predviđena, korišćenjem modela prethodno obučavanog na skupu za obučavanje. Razlika ove dve ocene označena je sa e_{uj}, gde je $e_{uj} = \hat{r}_{uj} - r_{uj}$ [2].

3.5.1.1 MAE

Srednje apsolutna greška predstavlja prosečnu apsolutnu grešku između predviđene i stvarne ocene date od strane korisnika. Prednosti ove mere su to što je jednostavna za razumevanje i laka za implementaciju. Formula za računanje MAE data je u nastavku [2]:

$$MAE = \frac{\sum_{(u,j)\in E} |e_{uj}|}{|E|}.$$

3.5.1.2 RMSE

Koren srednje kvadratne greške predstavlja koren prosečne kvadratne greške između predviđene i stvarne ocene date od strane korisnika. Jedna od karakteristika ove mere, a ujedno i razlika u odnosu na srednje apsolutnu grešku, jeste to što RMSE ima tendenciju da nesrazmerno kažnjava velike greške zbog kvadratnog člana unutar sume. Formula za računanje RMSE data je u nastavku [2]:

$$RMSE = \sqrt{\frac{\sum_{(u,j)\in E} e_{uj}^2}{|E|}}.$$

Glava 4

Predloženi metodi

U sklopu ovog rada razvijeno je nekoliko različitih modela sistema za preporuke. Za njihovu implementaciju korišćen je programski jezik Python, platforma Jupyter Notebook i biblioteke Surprise, Keras, scikit learn, Pandas, Numpy i druge. Opisi razvijanih modela, zajedno sa brojevima sekcija u kojima su diskutovani, nabrojani su u nastavku:

- Sistem za preporuke zasnovan na memoriji, koji koristi uzajamno filtriranje i k najbližih suseda, opisan je u sekciji 4.1. Ovaj sistem implementira pristup zasnovan na korisnicima i pristup zasnovan na stavkama.
- Sistem za preporuke zasnovan na memoriji, koji koristi uzajamno filtriranje i k najbližih suseda, implementiran upotrebom biblioteke Surprise, opisan je u sekciji 4.2.
- Sistem za preporuke zasnovan na modelu, koji koristi uzajamno filtriranje, implementiran upotrebom biblioteke Surprise, opisan je u sekciji 4.2.
- Sistem za preporuke zasnovan na dubokom učenju, implementiran upotrebom biblioteke Keras, opisan je u sekciji 4.3.
- Hibridni sistemi za preporuke sa težinama i sa smanjivanjem, opisani su u sekciji 4.4.

4.1 Sistem za preporuke koji koristi uzajamno filtriranje i KNN

U ovoj sekciji biće predložena implementacija sistema za preporuke koji predviđa ocene korisnika koristeći pristup zasnovan na korisnicima ili pristup zasnovan na stavkama. Takođe, prilikom predviđanja u obzir se uzimaju ocene samo k najsličnijih korisnika, odnosno stavki. Glavni metod koji vrši ova predviđanja je metod *selection*. Za njegovu implementaciju je iskorišćeno nekoliko pomoćnih metoda koji će najpre biti opisani u nastavku. Fokus ovog dela rada je na razumevanju ova dva pristupa i implementaciji korak po korak, bez korišćenja postojećih metoda neke od biblioteka.

Osnovni metodi koji predviđaju ocenu korisnika u za stavku i na osnovu matrice sličnosti korisnika, odnosno matrice sličnosti stavki su user_based_ratings_prediction i item_based_ratings_prediction. Njihove implementacije date su u kodovima na listinzima 4.1 i 4.2. Parametri koji se prosleđuju ovim metodima su: id korisnika, id stavke, matrica sličnosti korisnika (odnosno stavki), matrica ocena i broj suseda. Ova dva metoda su dalje iskoriščena za implementaciju metoda get_prediction.

```
1
2  def user_based_ratings_prediction(u, i, users_similarity, ratings, k = 5):
3      neighbors = []
4
5      similarities = list(zip(users_similarity[u][:], range(users_similarity.
       shape[0])))
6
7      similarities_sorted = sort_descending(similarities)
8
9      for i in range(1, k + 1):
10         neighbors.append(similarities_sorted[i][1])
11
12     rated_by_u = ratings[u].nonzero()[1]
13
14     user_u_mean = 0
15     arr = ratings[u, :].toarray()[0]
16     user_u_mean = np.sum(arr)
17
18     if len(rated_by_u) != 0:
19         user_u_mean = user_u_mean / len(rated_by_u)
20
21     numerator, denominator = 0.0, 0.0
22
23     for v in neighbors:
24         rated_by_v = ratings[v].nonzero()[1]
25         user_v_mean = 0
26
```

```
        if len(rated_by_v) != 0:
            for i in rated_by_v:
                user_v_mean += ratings[v, i]

            user_v_mean = user_v_mean / len(rated_by_v)

        r_vi = ratings[v,i]
        numerator += users_similarity[u][v]*(r_vi - user_v_mean)
        denominator += users_similarity[u][v]

    return user_u_mean + numerator/denominator
```

Kôd 4.1: Predviđanje zasnovano na korisnicima

```
def item_based_ratings_prediction(u, i, items_similarity, ratings, k = 5):
    neighbors = []

    similarities = list(zip(items_similarity[i][:],range(items_similarity.
    shape[0])))

    similarities_sorted = sort_descending(similarities)

    for i in range(1,k+1):
        neighbors.append(similarities_sorted[i][1])

    rated_i = ratings[:, i].nonzero()[0]

    item_i_mean = 0
    item_i_mean = np.sum(ratings[:, i].toarray()[0])

    if len(rated_i) != 0:
        item_i_mean = item_i_mean / len(rated_i)

    numerator, denominator = 0.0, 0.0

    for j in neighbors:
        rated_j = ratings[:, j].nonzero()[0]
        item_j_mean = 0

        if len(rated_j) != 0:
            arr2 = ratings[:, j].toarray()[0]
            item_j_mean = np.sum(arr2)
            item_j_mean = item_j_mean / len(rated_j)

        r_uj = ratings[u,j]
        numerator += items_similarity[i][j]*(r_uj - item_j_mean)
        denominator += items_similarity[i][j]

    return item_i_mean + numerator/denominator
```

Kôd 4.2: Predviđanje zasnovano na stavkama

Za predviđanje ocena svih korisnika za sve stavke implementiran je metod get_prediction. Parametri koji se prosleđuju ovom metodu su: matrica ocena, matrica sličnosti, tip koji označava da li će biti korišćen pristup zasnovan na korisnicima ili pristup zasnovan na stavkama i broj suseda. Kao povratnu vrednost ovaj metod vraća predviđene ocene. Implementacija ovog metoda data je u kodu na listingu 4.3.

```
1
2  def get_prediction(ratings, similarity_matrix, cf_type, k = 5):
3      predictions = np.zeros(ratings.shape)
4      if cf_type == 'user':
5          for u in range(ratings.shape[0]):
6              for i in range(ratings.shape[1]):
7                  predictions[u,i] = user_based_ratings_prediction(u, i,
       similarity_matrix , ratings, k)
8      elif cf_type == 'item':
9          for u in range(ratings.shape[0]):
10             for i in range(ratings.shape[1]):
11                 predictions[u,i] = item_based_ratings_prediction(u, i,
       similarity_matrix, ratings, k)
12     else:
13         print("Greska! Tip mora biti user ili item.")
14         return
15
16     return predictions
```

Kôd 4.3: Metod *get_prediction*

Implementacija metoda selection, koji predviđa ocena za sve korisnike i sve stavke za različite vrednosti parametra k (različit broj suseda), dat je u kodu na listingu 4.4. Ovaj metod poziva prethodno opisani metod get_prediction za različite vrednosti parametra k. Ulazni parametri metoda selection su: skup podataka, lista sa brojevima suseda i tip koji označava da li će biti korišćen pristup zasnovan na korisnicima ili pristup zasnovan na stavkama. Povratne vrednosti ovog metoda su predviđanja (ocene) i greške nastale prilikom predviđanja.

```
1
2  def selection(train_and_validation, k_values, cf_type):
3      train, validation = split_train_test(train_and_validation, 20)
4      errors = np.array([])
5      predictions_arr = np.array([])
6
7      if cf_type == 'user':
8          similarity = cosine_similarity(train) + EPS
9          for k in k_values:
10             predictions = get_prediction(train, similarity, 'user', k)
11             error = np.sqrt(mean_squared_error(validation.toarray()[
       validation.nonzero()],predictions[validation.nonzero()]))
```

```
12              errors = np.append(errors, error)
13              predictions_arr = np.append(predictions_arr, predictions)
14      else:
15          similarity = cosine_similarity(train.T) + EPS
16          for k in k_values:
17              predictions = get_prediction(train, similarity, 'item', k)
18              error = np.sqrt(mean_squared_error(validation.toarray()[
    validation.nonzero()],predictions[validation.nonzero()]))
19              errors = np.append(errors, error)
20              predictions_arr = np.append(predictions_arr, predictions)
21
22      k_optimal = k_values[np.argmin(errors)]
23
24      if cf_type == 'user':
25          similarity = cosine_similarity(train_and_validation) + EPS
26          predictions = get_prediction(train_and_validation, similarity, '
    user', k_optimal)
27          return errors, predictions
28      else:
29          similarity = cosine_similarity(train_and_validation.T) + EPS
30          predictions = get_prediction(train_and_validation, similarity, '
    item', k_optimal)
31          return errors, predictions
```

Kôd 4.4: Metod *selection*

4.2 Sistemi za preporuke koji koriste tehnike KNN i SVD

U ovoj sekciji biće prikazana dva različita pristupa: pristup zasnovan na memoriji i pristup zasnovan na modelu. Za implementaciju ovih pristupa korišćena je biblioteka Surprise. Kod pristupa zasnovanog na memoriji biće isprobane varijacije algoritma KNN, dok će samo onaj algoritam sa najboljim performansama biti na kraju iskorišćen za davanje preporuka u metodu generate_recommendationsKNN. Kod pristupa zasnovanog na modelu biće korišćena SVD tehnika. Metod koji daje preporuke koristeći ovu tehniku nazvan je generate_recommendationsSVD. U nastavku će biti opisana oba pristupa, kao i implementacije pomenutih metoda za davanje preporuka.

4.2.1 KNN – pristup zasnovan na memoriji

Ovaj pristup za formiranje preporuka uzima u obzir do k najsličnijih korisnika (kod uzajamnog filtriranja zasnovanog na korisnicima) ili do k najsličnijih stavki (kod uzajamnog filtriranja zasnovanog na stavkama). Podrazumevano, algoritam je zasnovan na korisnicima i parametar k ima vrednost 40. To znači da se razmatraju 40 najsličnijih korisnika da bi neka stavka bila preporučena korisniku. Varijante osnovnog KNNBasic algoritma (koji će biti opisan u podsekciji 4.2.1.1) uključuju WithMeans, WithZScore i Baseline (koji će biti opisani u podsekcijama 4.2.1.2, 4.2.1.3 i 4.2.1.4, respektivno), gde se radi davanja preporuka dodatno razmatraju i prosečna ocena korisnika, normalizovani z-rezultati ocena ili početna ocena. Svakom od ovih algoritama se prosleđuje broj suseda koji ulazi u razmatranje. Takođe, uključeni su samo oni susedi za koje mere sličnosti daju pozitivne rezultate, s obzirom na to da nema smisla prikupljati ocene od korisnika (za stavke) koji su negativno korelisani. Za dato predviđanje, stvarni broj suseda se može pročitati iz polja *actual_k* iz rečnika detalja predviđanja.

4.2.1.1 Osnovni KNN metod

Osnovni KNN metod, nazvan u implementaciji KNNBasic, predstavlja osnovni algoritam zasnovan na susedstvu koji koristi uzajamno filtriranje za predviđanje ocena. Parametri koje prima metod KNNBasic su:

- k (int): Maksimalan broj suseda koji se uzima u razmatranje. Podrazumevana vrednost ovog parametra je 40.
- *min_k* (int): Minimalan broj suseda koji se uzima u razmatranje. Ukoliko nema dovoljno suseda, za predviđanje se uzima globalna prosečna vrednost svih ocena. Podrazumevana vrednost ovog parametra je 1.
- *sim_options* (dict): Rečnik opcija vezano za meru sličnosti koja se koristi.
- *verbose* (bool): Vrednost ovog parametra govori o tome da li će se štampati prateće poruke vezano za procenu bias-a, sličnosti, itd. Podrazumevana vrednost je *True*.

Predviđena ocena $\hat{r}_{ui}$ se računa na sledeći način:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} s(u,v) r_{vi}}{\sum_{v \in N_i^k(u)} s(u,v)}$$

ili

$$\hat{r}_{ui} = \frac{\sum_{j \in N_u^k(i)} s(i,j) r_{uj}}{\sum_{j \in N_u^k(i)} s(i,j)},$$

zavisno od vrednosti polja *user_ based* unutar parametra *sim_ options* [12].

4.2.1.2 KNN metod sa sredinama

KNN metod sa sredinama, u implementaciji nazvan KNNWithMeans, je osnovni algoritam zasnovan na susedima koji koristi uzajamno filtriranje i koji uzima u obzir i prosek ocena za svakog korisnika. Parametri koje prima metod KNNWithMeans su:

- k (int): Maksimalan broj suseda koji se uzima u razmatranje. Podrazumevana vrednost ovog parametra je 40.
- *min_ k* (int): Minimalan broj suseda koji se uzima u razmatranje. Ukoliko nema dovoljno suseda, predviđanje postaje jednako prosečnoj vrednosti μ_u ili μ_i. Podrazumevana vrednost ovog parametra je 1.
- *sim_ options* (dict): Rečnik opcija vezano za meru sličnosti koja se koristi.
- *verbose* (bool): Vrednost ovog parametra govori o tome da li će se štampati prateće poruke vezano za procenu bias-a, sličnosti, itd. Podrazumevana vrednost je *True*.

Predviđena ocena $\hat{r}_{ui}$ se računa na sledeći način:

$$\hat{r}_{ui} = \mu_u + \frac{\sum_{v \in N_i^k(u)} s(u,v)(r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} s(u,v)}$$

ili

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} s(i,j)(r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} s(i,j)},$$

zavisno od vrednosti polja *user_ based* unutar parametra *sim_ options* [12].

4.2.1.3 KNN metod sa z-rezultatom

KNN metod sa z-rezultatom, u implementaciji nazvan KNNWithZScore, predstavlja osnovni algoritam zasnovan na susedstvu koji koristi uzajamno filtriranje i koji uzima u obzir z-rezultat normalizaciju (eng. z-score normalisation) svakog korsinika. Parametri koje prima metod KNNWithZScore su:

- k (int): Maksimalan broj suseda koji se uzima u razmatranje. Podrazumevana vrednost ovog parametra je 40.
- *min_ k* (int): Minimalan broj suseda koji se uzima u razmatranje. Ukoliko nema dovoljno suseda, predviđanje postaje jednako prosečnoj vrednosti μ_u ili μ_i. Podrazumevana vrednost ovog parametra je 1.
- *sim_ options* (dict): Rečnik opcija vezano za meru sličnosti koja se koristi.
- *verbose* (bool): Vrednost ovog parametra govori o tome da li će se štampati prateće poruke vezano za procenu bias-a, sličnosti, itd. Podrazumevana vrednost je *True*.

Predviđena ocena $\hat{r}_{ui}$ se računa na sledeći način:

$$\hat{r}_{ui} = \mu_u + \sigma_u * \frac{\sum_{v \in N_i^k(u)} s(u,v)(r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} s(u,v)}$$

ili

$$\hat{r}_{ui} = \mu_i + \sigma_i * \frac{\sum_{j \in N_u^k(i)} s(i,j)(r_{uj} - \mu_j)/\sigma_j}{\sum_{j \in N_u^k(i)} s(i,j)},$$

zavisno od vrednosti polja *user_ based* unutar parametra *sim_ options* [12].

4.2.1.4 KNN metod sa početnom ocenom

KNN metod sa početnom ocenom, u implementaciji nazvan KNNBaseline, je osnovni algoritam zasnovan na susedstvu koji koristi uzajamno filtriranje i koji uzima u obzir početnu ocenu (eng. baseline rating). Parametri koje prima metod KNNBaseline su:

- k (int): Maksimalan broj suseda koji se uzima u razmatranje. Podrazumevana vrednost ovog parametra je 40.

- *min_k* (int): Minimalan broj suseda koji se uzima u razmatranje. Ukoliko nema dovoljno suseda, predviđanje postaje jednako početnoj oceni (eng. baseline rating). Podrazumevana vrednost ovog parametra je 1.
- *sim_options* (dict): Rečnik opcija vezano za meru sličnosti koja se koristi. Preporuka je da se koristi *pearson_baseline* mera sličnosti.
- *bsl_options* (dict): Rečnik opcija na osnovu kojih se računa odnosno procenjuje početna ocena (eng. baseline).
- *verbose* (bool): Vrednost ovog parametra govori o tome da li će se štampati prateće poruke vezano za procenu bias-a, sličnosti, itd. Podrazumevana vrednost je *True*.

Predviđena ocena $\hat{r}_{ui}$ se računa na sledeći način:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} s(u,v)(r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} s(u,v)}$$

ili

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} s(i,j)(r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} s(i,j)},$$

zavisno od vrednosti polja *user_based* unutar parametra *sim_options* [12].

4.2.2 SVD – pristup zasnovan na modelu

Ovaj algoritam koristi pristup faktorizacije matrica. Matrica ocena korisnik-stavka je faktorisana na matrice korisnika i stavki, manjih dimenzija, koje se sastoje od latentnih faktora (skrivenih karakteristika). Podrazumevano, broj skrivenih karakteristika je 100. Ovi latentni faktori mogu da obuhvate poznate ocene i učestvuju u procesu predviđanja ocene za sve parove korisnik-stavka gde korisnik još nije ocenio stavku [13]. Parametri koje prima metod SVD su:

- *n_factors* (int): Broj latentnih faktora, odnosno skrivenih karakteristika (eng. hidden characteristics) za matricu faktorizacije (redukcije) dimenzionalnosti. Podrazumevano, ova vrednost je 100.
- *n_epochs* (int): Broj iteracija algoritma stohastičkog gradijentnog spusta (eng. stochastic gradient descent, SGD) koji se koristi za učenje parametara i minimizaciju greške.

- *lr_all* (float): Stopa učenja (eng. learning rate) koja predstavlja veličinu koraka u prethodno pomenutom algoritmu SGD. Podrazumevano, ova vrednost je 0.005.

- *reg_all* (float): Stopa regularizacije (eng. regularization rate) koja služi da spreči preprilagođavanje, tako da model generalizuje dobro nad podacima koje još nije video. Podrazumevano, ova vrednost je 0.02.

4.2.3 Metodi za davanje preporuka

Za davanje preporuka implementirana su dva metoda: generate_recommendationsKNN, koji koristi pristup zasnovan na memoriji i generate_recommendationsSVD, koji koristi pristup zasnovan na modelu. Za implementaciju prvog metoda koristi se algoritam KNNBasic i filtriranje zasnovano na stavkama dato u kodu na listingu 4.5. Ovom metodu je potrebno proslediti id korisnika, broj koji označava koliko ocena korisnika sa datim id-em se uzima u razmatranje i broj preporuka koje treba dati.

```
1
2  def generate_recommendationsKNN(userID='AVH1YOFM8WG9W', like_recommend=5,
       get_recommend =10):
3
4      sim_options       = {'name':'msd','min_support':3,'user_based':False}
5      similarity_matrix = KNNBasic(sim_options=sim_options).fit(trainset).\
6                          compute_similarities()
7
8      userID      = trainset.to_inner_uid(userID)
9      userRatings = trainset.ur[userID]
10
11     temp_df = pd.DataFrame(userRatings).sort_values(by=1, ascending=False)
       .\
12                head(like_recommend)
13     userRatings = temp_df.to_records(index=False)
14
15     recommendations   = {}
16
17     for user_top_item, user_top_item_rating  in userRatings:
18
19         all_item_indices          =   list(pd.DataFrame(similarity_matrix)[
       user_top_item].index)
20         all_item_weighted_rating  =   list(pd.DataFrame(similarity_matrix)[
       user_top_item].values*\
21                                           user_top_item_rating)
22         all_item_weights          =   list(pd.DataFrame(similarity_matrix)[
       user_top_item].values)
23
24         for index in range(len(all_item_indices)):
```

```
            if index in recommendations:
                recommendations[index] += all_item_weighted_rating[index]
            else:
                recommendations[index]  = all_item_weighted_rating[index]

    for index in range(len(all_item_indices)):
            if all_item_weights[index] != 0:
                recommendations[index] = recommendations[index]/\
                                         (all_item_weights[index]*
    like_recommend)

    temp_df = pd.Series(recommendations).reset_index().sort_values(by=0,
    ascending=False)
    recommendations = list(temp_df.to_records(index=False))

    final_recommendations = []
    count = 0

    for item, score in recommendations:
        flag = True
        for userItem, userRating in trainset.ur[userID]:
            if item == userItem:
                flag = False
                break
        if flag == True:
            final_recommendations.append(trainset.to_raw_iid(item))
            count +=1

        if count > get_recommend:
            break

    return(final_recommendations)
```

Kôd 4.5: Metod *generate_recommendationsKNN*

Implementacija drugog metoda za davanje preporuka, koji koristi algoritam SVD, data je u kodu na listingu 4.6. Ovom metodu je potrebno proslediti id korisnika (kome se daju preporuke) i broj preporuka koje treba dati.

```


def generate_recommendationsSVD(userID='AVH1YOFM8WG9W', get_recommend =10):

    model = SVD(n_factors=30, n_epochs=5, lr_all=0.002, reg_all= 0.5)
    model.fit(trainset)

    testset = trainset.build_anti_testset()
    predictions = model.test(testset)
    predictions_df = pd.DataFrame(predictions)

    predictions_userID = predictions_df[predictions_df['uid'] == userID].\
```

```
13                          sort_values(by="est", ascending = False).head(
    get_recommend)
14
15      recommendations = []
16      recommendations.append(list(predictions_userID['iid']))
17      recommendations = recommendations[0]
18
19      return(recommendations)
```

Kôd 4.6: Metod *generate_recommendationsSVD*

4.3 Sistem za preporuke zasnovan na dubokom učenju

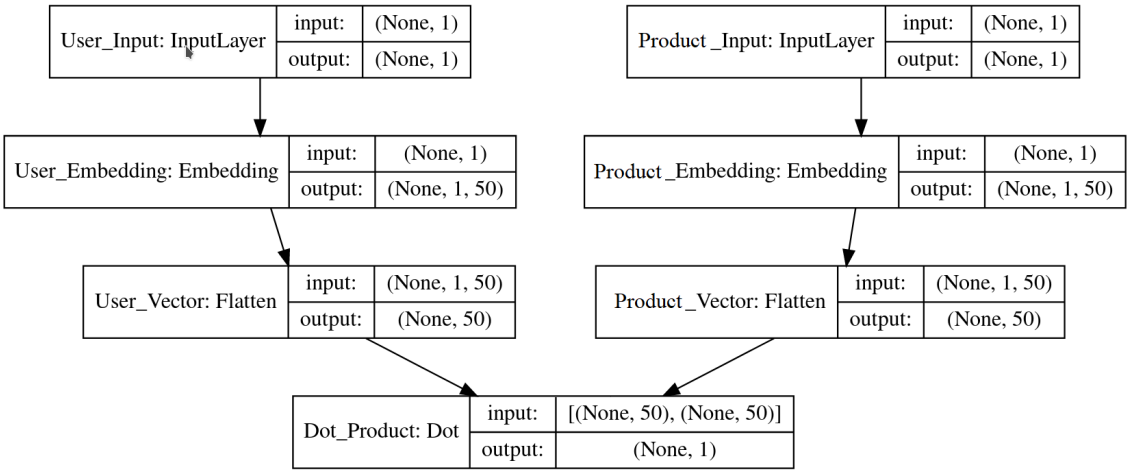
U ovoj sekciji biće predstavljen model zasnovan na dubokom učenju koji koristi latentne faktore, nalik na SVD tehniku. Za implementaciju je korišćena biblioteka Keras. Arhitekturu modela čine tri sloja i to:

- Ulazni sloj (eng. input layer) za korisnika i za proizvod – ovaj sloj binarizuje vektore za identifikaciju korisnika i proizvoda, gde za Item(i) vrednost 1 označava da je korisnik u interagovao sa stavkom Item(i), dok User(u) identifikuje korisnika.
- Ugnježdeni sloj (eng. embedding layer) – ovo je potpuno povezani sloj koji preslikava retku reprezentaciju u gust vektor. Dobijeni gusti vektori za korisnika i stavku se mogu tumačiti kao vektori sa latentnim karakteristikama.
- Ravnajuci sloj (eng. flatten layer) – ovaj sloj konvertuje multi-dimenzionalni niz u jedno-dimenzioni niz.

Na kraju je izračunat skalarni proizvod i time izračunata predviđena ocena. Arhitektura modela je prikazana na slici: 4.1.

4.4 Hibridni sistemi za preporuke

U ovoj sekciji biće predložene dve implementacije hibridnih sistema: hibrid zasnovan na težinama i hibrid sa smenjivanjem. Oba sistema koriste prethodno implementirane metode zasnovane na sadržaju i uzajamnom filtriranju. Fokus ovog dela poglavlja je na razumevanju kako dva metoda mogu biti kombinovana i iskorišćena



Slika 4.1: Arhitektura mreže

u predviđanju ocena. Ova rešenja, s obzirom na to da koriste filtriranje zasnovano na sadržaju, mogu (uz male modifikacije) biti iskorišćena za rešavanje problema iniciranja početka u situacijama kada se dodaje novi korisnik. Naime, ovaj metod radi sa ostalim atributima vezanim za proizvod, dok interakcije korisnika ovde nisu od presudnog značaja.

4.4.1 Implementacija metode koja koristi filtriranje zasnovano na sadržaju

Implementacija metode koja koristi filtriranje zasnovano na sadržaju prikazana je u kodu na listingu 4.7. Ovaj metod za predviđanje ocena koristi klasifikator KNeighborsClassifier biblioteke sklearn. Metod prima četiri parametara: tabelu ocena, tabelu stavki, id korisnika i id stavke. Povratna vrednost metoda je predviđena ocena, razlika između predviđene i stvarne ocene i stvarna ocena.

```
1  def cb_predict(ratings, items, user_id, item_id):
2      user_ratings = ratings[ratings['userID'] == user_id].join(items.
       set_index('id'), on='itemID', how='left', lsuffix='_left', rsuffix='
       _right')
3
4      is_target = (user_ratings['itemID_left'] == item_id)
5
6      features = pd.get_dummies(user_ratings.drop(columns=['overall']))
7      train_features = features[~is_target]
8      target_features = features[is_target]
9
10     encoder = LabelEncoder()
```

```
    train_labels = encoder.fit_transform(user_ratings[~is_target]['overall'])
    target_label = user_ratings[is_target]['overall'].iloc[0]

    clf = KNeighborsClassifier(n_neighbors=1)
    clf.fit(train_features, train_labels)
    prediction = encoder.inverse_transform(clf.predict(target_features))[0]

    return prediction, prediction - target_label, target_label
```

Kôd 4.7: Metod za predviđanje ocene zasnovan na sadržaju

4.4.2 Implementacija metode koja koristi uzajamno filtriranje

Implementacija metode koja koristi uzajamno filtriranje je prikazana u kodu na listingu 4.8. Ovaj metod za predviđanje ocena koristi KNNBasic algoritam biblioteke Surprise. Metod prima tri parametara: tabelu ocena, id korisnika i id stavke. Povratna vrednost metoda je predviđena ocena, razlika između predviđene i stvarne ocene i stvarna ocena.

```
def cf_predict(ratings, user_id, item_id):
    is_target = (ratings['userID']==user_id)&(ratings['itemID']==item_id)
    target = ratings[is_target].iloc[0]

    train_set = sp.Dataset.load_from_df(ratings[~is_target][['userID', 'itemID', 'overall']], sp.Reader(rating_scale=(1, 5))).build_full_trainset()

    algo = sp.KNNBasic(verbose=False)
    algo.fit(train_set)

    prediction = algo.predict(target['userID'], target['itemID'], verbose=False)

    return prediction.est, prediction.est - target['overall'], target['overall']
```

Kôd 4.8: Metod za predviđanje ocene zasnovan na uzajamnom filtriranju

4.4.3 Implementacija hibridnog sistema koji koristi težine

Implementacija hibridnog sistema koji koristi težine, prikazana je u kodu na listingu 4.9. Pristup zasnovan na težinama dodaje vrednosti (značaj/težine) pred-

viđenim ocenama od strane modela zasnovanog na uzajamnom filtriranju, odnosno modela zasnovanog na sadržaju.

```
def predict_weighted(ratings, items, user_id, item_id):
    prediction_cf, _, true_rating = cf_predict(ratings, user_id, item_id)
    prediction_cb, _, true_rating = cb_predict(ratings, items, user_id,
    item_id)

    prediction = 0.5 * prediction_cf + 0.5 * prediction_cb
    error = prediction - true_rating

    return prediction, error, true_rating
```

Kôd 4.9: Hibridni sistem zasnovan na težinama

4.4.4 Implementacija hibridnog sistema sa smenjivanjem

Implementacija hibridnog sistema sa smenjivanjem prikazana je u kodu na listingu 4.10. Pristup sa smenjivanjem koristi model zasnovan na uzajamnom filtriranju samo ukoliko postoji dovoljan broj ocena (više od 3) za datu stavku (čiji se id prosleđuje metodu). Inače, koristi model zasnovan na sadržaju.

```
def predict_switching(ratings, items, user_id, item_id):
    num_ratings = len(ratings[ratings['itemID'] == item_id])
    if num_ratings > 3:
        print('Koriscenjem sistema zasnovanog na uzajamnom filtriranju')
        return predict_cf(ratings, user_id, item_id)
    else:
        print('Koriscenjem sistema zasnovanog na sadrzaju')
        return predict_cn(ratings, items, user_id, item_id)
```

Kôd 4.10: Hibridni sistem sa smenjivanjem

Glava 5

Eksperimentalni rezultati

U ovom poglavlju biće prikazano evaluiranje predloženih modela. Najpre će biti opisan skup podataka (sekcija 5.1) koji se koristi u implementacijama ovih rešenja, a zatim rezultati primene metoda nad tim podacima, kao i uporedna analiza rezultata sa diskusijom.

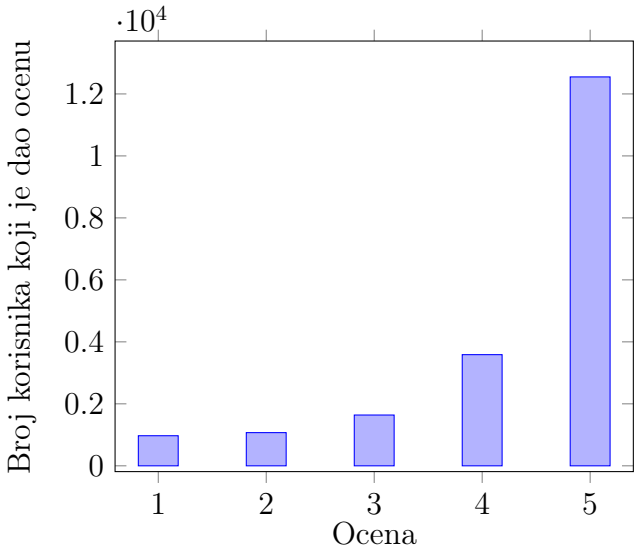
Evaluacija sistema za preporuke zasnovanog na memoriji koji koristi uzajamno filtriranje i k najbližih suseda (pristup zasnovan na korisnicima i pristup zasnovan na stavkama) biće data u sekciji 5.2. Sistem za preporuke zasnovan na memoriji koji koristi uzajamno filtriranje i k najbližih suseda, a implementiran je upotrebom biblioteke Surprise biće evaluiran u sekciji 5.3. Evaluacija sistema za preporuke zasnovanog na modelu koji koristi uzajamno filtriranje a implementiran je upotrebom biblioteke Surprise biće data u sekciji 5.3, a sistema za preporuke zasnovanog na dubokom učenju implementiranog upotrebom biblioteke Keras u sekciji 5.4. Hibridni sistemi za preporuke sa težinama i sa smenjivanjem biće evaluirani u sekciji 5.5.

Određivanje hiperparametara pomenutih modela biće izvršeno *GridSearchCV* metodom biblioteke sklearn. Skup podataka će biti podeljen na skup za obučavanje i skup za testiranje. Ocenjivanje modela vršiće se u odnosu na test skup.

5.1 O skupu podataka

Za obučavanje i testiranje modela korišćeni su podaci o proizvodima koji se prodaju na Amazonu, preuzeti sa javno dostupnog linka: https://cseweb.ucsd.edu/ jmcauley/datasets/amazon_v2/. Skup podataka se sastoji iz dve tabele. Jedna tabela sadrži samo informacije o proizvodima, kao što su: naziv proizvoda, brend, dimenzije, težina i ostale propratne informacije vezane za proizvode. Ova tabela se sastoji iz

19999 redova i 449 kolona. Druga tabela sadrži informacije o interakcijama korisnik-stavka, odnosno sadrži atribute o korisniku, oceni, proizvodu, vremenu pregledanja proizvoda, utisku i ostale prateće informacije. Ova tabela se sastoji iz 19999 redova i 29 kolona. S obzirom na to da skup podataka sadrži i atribute koji za sprovođenje pomenutih algoritama nisu od značaja, takvi atributi su eliminisani u procesu pretprocesiranja. Vrednosti atributa koji predstavljaju ocenu su iz intervala [1, 5] i mogu biti samo celi brojevi. Vrednost 1 predstavlja najnižu ocenu, a vrednost 5 najvišu ocenu. Na grafiku 5.1 je, nakon što su izbačeni duplikati iz skupa podataka, prikazano koliko korisnika je dalo koju ocenu.



Slika 5.1: Ocenjivanje proizvoda

5.2 Sistem za preporuke koji koristi uzajamno filtriranje i KNN

U ovoj sekciji biće evaluiran sistem za preporuke koji koristi uzajamno filtriranje i k najbližih suseda. Skup podataka koji je korišćen za potrebe ove implementacije sastoji se samo iz tabele sa ocenama proizvoda koje su im korisnici dodelili. Razlog korišćenja drugačijeg skupa podataka u odnosu na ostatak rada je što se za ovaj skup podataka dobijaju reprezentativniji rezultati.

5.2.1 Pretprocesiranje podataka

Nakon učitavanja skupa podataka (tabela 5.1), formirana je matrica ocena (tabela 5.2) i sve nedefinisane vrednosti su zamenjene nulom. S obzirom na to da je veliki broj elemenata matrice jednak nuli, dobijena matrica je retka.

Indeks	Id korisnika	Id stavke	Ocena
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0
...	...	...	...
100831	610	166534	4.0
100832	610	168248	5.0
100833	610	168250	5.0
100834	610	168252	5.0
100835	610	170875	3.0

Tabela 5.1: Skup podataka – tabela ocena

Id stavke / Id korisnika	1	2	3	...	193585	193587	193609
1	4.0	0.0	4.0	...	0.0	0.0	0.0
2	0.0	0.0	0.0	...	0.0	0.0	0.0
3	0.0	0.0	0.0	...	0.0	0.0	0.0
4	0.0	0.0	0.0	...	0.0	0.0	0.0
5	4.0	0.0	0.0	...	0.0	0.0	0.0

Tabela 5.2: Matrica ocena

Iz matrice ocena u obzir su uzeti samo redovi sa više od 700 ocena, kako bi modelu bilo pruženo što više poznatih ocena. Zbog velikog vremena izvršavanja, urađena je i redukcija dimenzija ove matrice. Nakon toga izvršena je podela skupa podataka na skupove za obučavanje (i validaciju) i testiranje. Implementacija metoda za filtriranje, koji prima dva parametra – skup podataka i broj ne-nula elemenata označen sa m, data je na listingu 5.1. Ovaj metod filtrira matricu ocena tako da ostanu samo redovi koji imaju više od m ne-nula elemenata. Implementacija metoda za podelu skupa podataka data je na listingu 5.2. Ovaj metod vrši podelu datog skupa podataka na skup za obučavanje i skup za testiranje tako što se iz datog skupa „uzima" procenat ocena koje se upisuju u test skup, a uklanjaju iz skupa za obučavanje.

Parametri koje prima ovaj metod su retka matrica ocena (skup podataka koji treba podeliti na skup za obučavanje i skup za testiranje) i broj koji predstavlja procenat ocena koje ulaze u test skup.

```
def filter_users(train_and_validation, m):
    xy = train_and_validation.toarray()
    xy_filtered_matrix = []
    for y in xy:
        sum = 0
        nz = np.count_nonzero(y)
        if (nz > m):
            xy_filtered_matrix.append(y)

    arr_shape = np.vstack(xy_filtered_matrix).shape
    filtered_arr = np.vstack(xy_filtered_matrix)

    return sps.csr_matrix(filtered_arr)
```

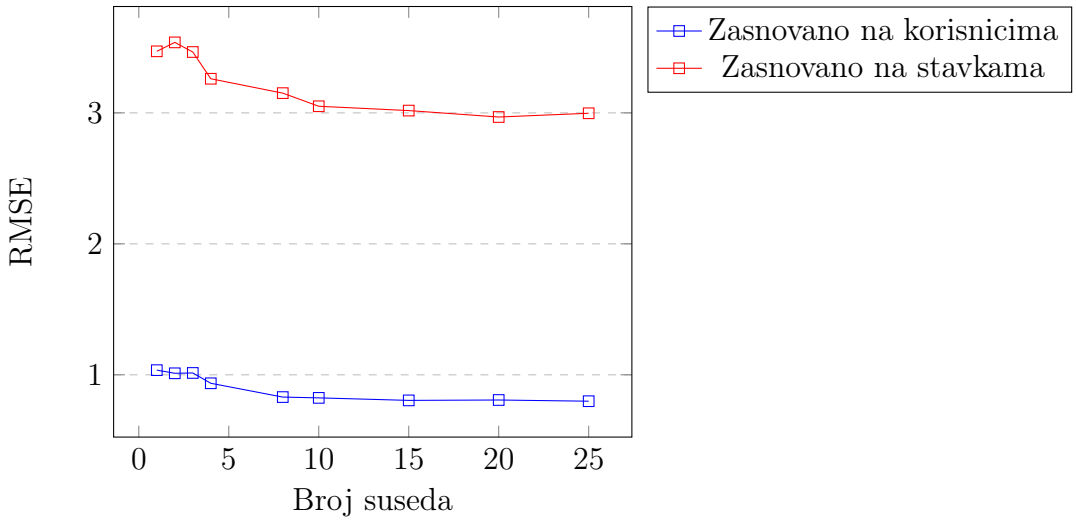
Kôd 5.1: Filtriranje korisnika

```
def split_train_test(ratings_csr_matrix, percentage):
    test_ratings_number = percentage / 100
    print("Odnos ocena u skupu za testiranje: ", percentage, "%")
    print("Odnos ocena u skupu za obucavanje: ", 100-percentage, "%")

    total_ratings = ratings_csr_matrix.toarray()
    dimensions_of_total_ratings = total_ratings.shape
    print("Ukupan broj korisnika: ", dimensions_of_total_ratings[0])
    print("Ukupan broj stavki: ", dimensions_of_total_ratings[1])

    test = np.zeros(dimensions_of_total_ratings)
    train = total_ratings.copy()

    nonzero_ratings_per_row = (total_ratings != 0).sum(1)
    print("Ukupan broj ne-nula ocena u svim redovima: \n",
    nonzero_ratings_per_row)

    for user in range(dimensions_of_total_ratings[0]):
        nonzero_test_ratings_per_user = int(np.ceil(test_ratings_number*
    nonzero_ratings_per_row[user]))
        test_ratings = np.random.choice(total_ratings[user, :].nonzero()
    [0], size = nonzero_test_ratings_per_user, replace = False)
        train[user, test_ratings] = 0
        test[user, test_ratings] = total_ratings[user, test_ratings]

    if (not(np.all((train * test) == 0))):
        print("Greska!")
    else:
        return sps.csr_matrix(train),sps.csr_matrix(test)
```

Kôd 5.2: Podela skupa podataka

5.2.2 Evaluacija modela

Mera za merenje tačnosti predviđenih ocena je RMSE. Sistem je evaluiran za različite vrednosti parametra k (broja suseda), za pristup zasnovan na korisnicima i pristup zasnovan na stavkama. Rezultati su prikazani na slici 5.2 i u tabeli 5.3. Prema datim rezultatima može se zaključiti da pristup zasnovan na korisnicima daje bolja predviđanja.



Slika 5.2: Grafički prikaz greške RMSE za pristup zasnovan na stavkama i pristup zasnovan na korisnicima

Broj suseda k	1	2	3	4	8	10	15	20	25
Zasnovano na korisnicima	1.036	1.012	1.014	0.935	0.830	0.824	0.805	0.808	0.799
Zasnovano na stavkama	3.470	3.538	3.465	3.261	3.151	3.051	3.018	2.968	2.997

Tabela 5.3: Tabelarni prikaz greške RMSE za pristup zasnovan na stavkama i pristup zasnovan na korisnicima

5.3 Sistemi za preporuke koji koriste tehnike KNN i SVD

U ovoj sekciji biće evaluirani sistemi za preporuke koji koriste tehnike KNN i SVD. U nastavku će najpre biti opisano pretprocesiranje skupa podataka, a zatim će biti diskutovane performanse modela. Oni modeli koji budu imali najbolje performanse biće iskorišćeni za davanje preporuka.

5.3.1 Pretprocesiranje podataka

Nakon učitavanja skupa podataka, eliminisani su atributi koji nisu značajni za dalju implementaciju. Dve tabele, iz kojih se sastoji skup podataka (opisane u sekciji 5.1), su spojene kako bi se vrednosti atributa koji predstavlja id proizvoda zamenile nazivom proizvoda (radi čitljivosti). Novodobijena tabela sadrži atribute: id korisnika, naziv proizvoda i ocena. Takođe, iz datog skupa podataka eliminisani su duplikati.

5.3.2 Evaluacija modela

Svi modeli (sa podrazumevanim vrednostima parametara) opisani u sekciji 4.2 su obučavani korišćenjem 5-slojne unakrsne provere[1] (eng. 5 fold cross validation). Za evaluaciju su korišćene mere MAE i RMSE. Performanse modela su prikazane u tabeli 5.4.

Među isprobanim algoritmima, najmanju grešku RMSE ima SVD. Što se tiče varijacija KNN algoritama, najmanju grešku RMSE ima KNNBasic. Algoritam KNNWithMeans ima najveću vrednost parametra *fit_ time*, dok algoritam SVD ima najmanju vrednost parametra *test_ time*.

Skup podataka je podeljen slučajnim uzorkom na skup podataka za obučavanje (70%) i skup podataka za testiranje (30%). Za modele KNNBasic i SVD izvršeno je podešavanje parametara pri čemu je opet korišćena 5-slojna unakrsna provera.

Ispostavilo se da se najbolje performanse modela KNNBasic postižu sa sledećim vrednostima parametara: MSD kao mera sličnosti, vrednost 3 za parametar

[1]k-slojna unakrsna provera podrazumeva podelu podataka D na k približno jednakih podskupova, takozvanih slojeva (eng. folds) S1, ..., Sk. Zatim se za $i = 1, ..., k$ obučava model nad podacima D, gde $S_i \not\subset D$. Potom se izvrše predviđanja dobijenim modelom na sloju Si. Na kraju se izračuna ocena kvaliteta na osnovu svih predviđanja na celom skupu D.

Metod	MAE	RMSE	Parametar *fit_ time*	Parametar *test_ time*
SVD	0.846150	1.066627	0.142742	0.002011
KNNBasic	0.852506	1.076491	0.120428	0.004301
KNNBaseline	0.850317	1.082411	0.091381	0.003928
KNNWithMeans	0.859457	1.097607	0.296933	0.005781
KNNWithZScore	0.865696	1.101871	0.243386	0.005185

Tabela 5.4: Performanse modela

min_ support i vrednost *False* za parametar *user_ based*. Tada MAE iznosi 0.87106, a RMSE 1.08917.

Za model SVD, da bi se postigla najveća vrednost mere MAE, potrebno je odabrati sledeće vrednosti parametara: vrednost 50 za parametar *n_ factors*, vrednost 5 za parametar *n_ epochs*, vrednost 0.005 za parametar *lr_ all* i vrednost 0.2 za parametar *reg_ all*. Tada MAE iznosi 0.85317. Da bi se postigla najveća vrednost mere RMSE, potrebno je odabrati sledeće vrednosti parametara: vrednost 90 za parametar *n_ factors*, vrednost 5 za parametar *n_ epochs*, vrednost 0.002 za parametar *lr_ all* i vrednost 0.5 za parametar *reg_ all*. Tada RMSE iznosi 1.079925.

Nakon što su pronađene odgovarajuće vrednosti parametara i prosleđene ovim modelima koji su obučavani na skupu podataka za obučavanje, izvršeno je testiranje na skupu podataka za testiranje i evaluacija korišćenjem mera MAE i RMSE. Dobijene vrednosti ovih mera za model KNNBasic iznose: 0.9197 za MAE i 1.1572 za RMSE, što je uporedivo sa vrednostima dobijenim koristeći unakrsnu proveru na skupu za obučavanje, pa se može zaključiti da model dobro generalizuje. Vrednosti mera za model SVD, pri testiranju, su: MAE iznosi 0.8760, a RMSE 1.0996, što je uporedivo sa vrednostima dobijenim koristeći unakrsnu proveru na skupu za obučavanje, što znači da i ovaj model dobro generalizuje. Ovi modeli, čije su performanse zadovoljavajuće, su iskorišćeni za davanje preporuka u metodama *generate_ recommendationsKNN* i *generate_ recommendationsSVD*. Za meru sličnosti u algoritmu *generate_ recommendationsKNN* korišćena je srednje kvadratna distanca.

Za kraj je ostalo analizirati rezultate (dobijene preporuke) ovih metoda. Najpre se može primetiti da su preporuke dobijene koristeći algoritme KNNBasic i SVD iste, samo je redosled, odnosno prioritet, drugačiji. Ovo je i očekivano s obzirom na to da su u pitanju različiti algoritmi.

Uspešno je implementiran pristup zasnovan na memoriji kao i pristup zasnovan na modelu. Međutim, s obzirom na to da modeli uzajamnog filtriranja nisu dobra

opcija u slučaju novog korisnika ili nove stavke, kada je malo toga poznato u vezi preferencija i ocena, ovaj model se može unaprediti implementiranjem hibridnog pristupa, gde bi se kod predviđanja ocena u slučaju novog korisnika ili stavke iskoristio model zasnovan na sadržaju, koji je pogodniji u tom slučaju.

5.4 Sistem za preporuke zasnovan na dubokom učenju

U ovoj sekciji će biti predstavljeno treniranje i evaluacija sistema za preporuke zasnovanog na dubokom učenju. U nastavku će najpre biti opisano pretprocesiranje skupa podataka. Zatim će, model sa arhitekturom opisanom u poglavlju 4 u sekciji 4.3, biti formiran nad skupom podataka za obučavanje. Pri tome će biti korišćen različit broj epoha i različite vrednosti parametra *batch_size*. Za evaluaciju modela biće korišćena srednje kvadratna greška kao funkcija gubitka, prilikom obučavanja i prilikom testiranja modela.

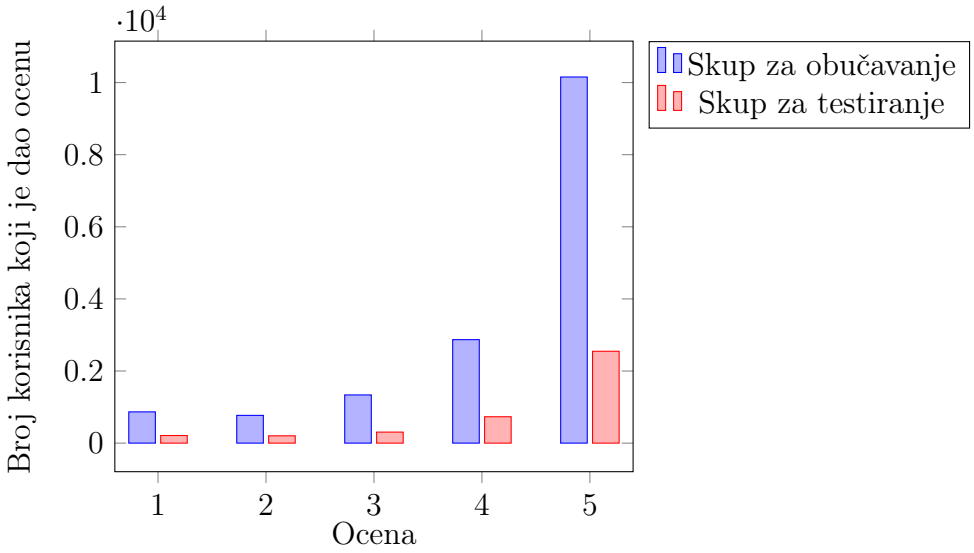
5.4.1 Pretprocesiranje podataka

Nakon učitavanja skupa podataka, uklonjeni su atributi koji nisu značajni za implementaciju tako što su eliminisane odgovarajuće kolone tabele. Atributi koji predstavljaju (identifikuju) korisnika i proizvod su kodirani pomoću metoda *encode_user_item* koja modifikuje prosleđeni skup podataka tako što dodaje još dva atributa – *User*, koji predstavlja indeks korisnika i *Item* koji predstavlja indeks proizvoda. Vrednosti novododatih atributa su izračunate pomoću metoda *LabelEncoder* biblioteke sklearn, koji kodira ne-numeričke labele (id korisnika i id proizvoda) u numeričke labele.

Skup podataka je podeljen na skupove za obučavanje i testiranje u razmeri 70:30. Prikaz broja korisnika koji su dali određenu ocenu, nakon ove podele, dat je na slici 5.3.

5.4.2 Evaluacija

Obučavanje modela je izvršeno za različite vrednosti parametara *epochs*, *batch_size* i *n_latent_factor* (broj latentnih faktora/karakteristika). Korišćen je *Adam* optimizator sa stopom učenja 0.0005. Za funkciju greške (eng. loss function) prilikom



Slika 5.3: Vizuelni prikaz podele skupa podataka

obučavanja i testiranja korišćena je srednje kvadratna greška. U nastavku su prikazane vrednosti funkcije gubitka tokom obučavanja i tokom testiranja modela. Ovo je urađeno za različite vrednosti prethodno nabrojanih parametara. Rezultati su prikazani u tabeli 5.5.

Parametar *epochs*	Parametar *batch_size*	Obučavanje	Testiranje
10	64	0.02155	17.58496
10	128	0.06493	17.59478
20	64	0.02150	17.59012
20	128	0.52304	17.63714
30	64	0.01992	17.57510
30	128	0.00501	17.56958
100	64	0.01391	17.54213
100	128	0.00964	17.55469

Tabela 5.5: Vrednosti funkcije gubitka za različite vrednosti parametara *epochs* i *batch_size*

5.5 Hibridni sistemi za preporuke

U ovoj sekciji biće evaluirani hibridni sistemi za preporuke opisani u sekciji 4.4. Ovi sistemi su evaluirani tako što su izračunate razlike između predviđene i stvarne ocene. Što je razlika manja, predviđanje je bolje. U nastavku će najpre biti opi-

san proces pretprocesiranja podataka, a onda će biti evaluiran hibrid zasnovan na težinama i hibrid sa smenjivanjem.

5.5.1 Pretprocesiranje podataka

Nakon učitavanja inicijalnog skupa podataka (dve tabele), izdvajanja atributa od značaja, uklanjanja redova koji sadrže nedefinisane vrednosti i uklanjanja duplikata, podaci su grupisani u tri tabele: *users_ all* (sa atributima: id korisnika, ime korisnika, standardni identifikacioni broj Amazona (eng. Amazon standard identification number), komentar korisnika, opis proizvoda, ocena, id korisnika), *items_ all* (sa atributima: standardni identifikacioni broj Amazona, naziv proizvoda, brend proizvoda, dimenzije proizvoda, težina proizvoda, id proizvoda) i *ratings_ all* (sa atributima: standardni identifikacioni broj Amazona, id korisnika, id proizvoda, ocena).

5.5.2 Hibridni sistem sa težinama

Apsolutna vrednost greške koju prilikom predviđanja ocene za ciljnog korisnika i stavku pravi hibridni sistem sa težinama je 0.4375. Rezultati korišćenja hibridnog sistema koji koristi težine, odnosno predviđena ocena i greška nastala pri tom predviđanju, dati su u tabeli 5.6.

Predviđena ocena	Greška
4.56250	0.43750

Tabela 5.6: Hibridni sistem zasnovan na težinama – rezultati

5.5.3 Hibridni sistem sa smenjivanjem

Kod hibridnog sistema sa smenjivanjem razlikuju se greške koje se dobijaju prilikom korišćenja uzajamnog filtriranja i prilikom korišćenja pristupa zasnovanog na sadržaju. U prvom slučaju, apsolutna vrednost greške je 0.875, a u drugom 3. Rezultati oba hibridna sistema data su u tabeli 5.7. S obzirom na to da je skup podataka relativno mali, nije bilo prostora za eksperimentisanje sa različitim vrednostima parametra koji predstavlja granicu koja određuje kada se koristi model zasnovan na uzajamnom filtriranju, a kada model zasnovan na sadržaju.

Pristup	Predviđena ocena	Greška
Uzajamno filtriranje	4.12500	0.87500
Zasnovan na sadržaju	2.00000	3.00000

Tabela 5.7: Hibridni sistem sa smenjivanjem – rezultati

Glava 6

Zaključak

Kao što je pomenuto na početku rada, sistemi za preporuke su široko zastupljeni u elektronskoj trgovini. Precizno predviđanje i davanje preporuka može da dovede do poboljšanja poslovanja, a samim time i do povećanja profita. Prilikom implementacija ovih sistema nailazi se na različite probleme koji uključuju oskudnost podataka, nove korisnike, preporučivanje novih proizvoda i drugo. Pored toga što sistem za preporuke treba da prevaziđe sve ove probleme, on treba da radi brzo, efikasno i tačno nad realnim skupovima podataka ogromne veličine.

U ovom radu su opisani različiti pristupi za implementaciju sistema za preporuke koji koriste uzajamno filtriranje, ali i filtriranje zasnovano na sadržaju i duboko učenje. Ovi sistemi rade sa skupom proizvoda koji se prodaju na Amazonu. S obzirom na to da sistemi za preporuke generalno donose bolje zaključke što im je više ulaznih podataka prosleđeno, a da je u ovom radu korišćeno relativno malo podataka, preptpostavka je da bi implementirani pristupi postizali još bolje rezultate nad većim skupom podataka. Implementacija pomenutih pristupa je izvršena u Python programskom jeziku uz odgovarajuću vizuelizaciju podataka. Neki pristupi daju odlične rezultate, dok su rezultati dobijeni drugim pristupima nezadovoljavajući. Za implementaciju su korišćene biblioteke Surprise, scikit learn, Keras i druge biblioteke koje se koriste za učitavanje, pretprocesiranje i vizuelizaciju podataka.

Postoji prostor za usavršavanje razvijenih sistema za preporuke. Rad sa većim skupom podataka, odnosno obogaćivanje baze podataka može doprineti dobijanju boljih rezultata. Dodavanjem novih atributa ili korišćenjem drugačijih atributa, posebno za učenje modela zasnovanog na sadržaju, može dovesti do tačnijih predviđanja ocena. Takođe nešto što nije razmatrano, jeste efekat promene preferencije korisnika. U „živom" sistemu za preporuke, korisnici konstantno dodaju nove i me-

njaju date ocene. Onda kada se određeni broj ovih podataka promeni, treba izvršiti preračunavanje. Sam prag na osnovu kojeg se određuje kada se vrši preračunavanje, treba biti pažljivo odabran. Moguće je isprobati i druge varijacije predloženih modela i eksperimentisati sa različitim vrednostima njihovih parametara. Kombinovanjem više različitih modela u okviru hibridnog sistema, moguće je doći do unapređenog sistema za davanje preporuka.

Bibliografija

[1] F. Ricci, L. Rokach and B. Shapira. Introduction to recommender systems handbook. In Recommender systems handbook (pp. 1-35). Springer, Boston, MA, 2010.

[2] Charu C. Aggarwal. Recommender Systems: The Textbook. Springer International Publishing Switzerland, 2016.

[3] Xiaoyuan Su and Taghi M. Khoshgoftaar. A Survey of Collaborative Filtering Techniques. Department of Computer Science and Engineering, Florida Atlantic University, 777 Glades Road, Boca Raton, FL 33431, USA, 2009, https://www.researchgate.net/publication/220173171_A_Survey_of_Collaborative_Filtering_Techniques.

[4] J. Ben Schafer, Joseph Konstan and John Riedl. Recommender Systems in E-Commerce. Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455, 1-612-625-4002, 1999, https://www.researchgate.net/publication/2507550_Recommender_Systems-_in_E-Commerce.

[5] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci and Giovanni Semeraro. Semantics-aware Content-based Recommender Systems. Department of Computer Science, University of Bari „Aldo Moro", Italy, 2015, https://www.researchgate.net/publication/294285452_Semantics-Aware_Content-Based_Recommender_Systems.

[6] Claudio Adrian Levinas. An Analysis of Memory Based Collaborative Filtering Recommender Systems with Improvement Proposals. UPC, URV, UB, 2014, https://upcommons.upc.edu/bitstream/handle/2099.1/22602/102384.pdf?sequence=1isAllowed=y.

[7] Robin Burke. Hybrid Recommender Systems: Survey and Experiments. California State University, Fullerton, Department of Information Systems and Decision Sciences, 2002, https://www.researchgate.net/publication/263377228_Hybrid_Recommender-_Systems_Survey_and_Experiments.

[8] Bc. Guzel Samigullina. Algorithms for collaborative filtering in Point-of-Interest Recommendation Systems. Faculty of Information Technology, CTU in Prague, Department of Software Engineering, 2019, https://dspace.cvut.cz/bitstream/handle/10467/82612/F8-DP-2019-Samigullina-Guzel-thesis.pdf?sequence=-1isAllowed=y.

[9] John Zimmerman, Kaushal Kauapati, Anna L. Buczak and Dave Schaffer. TV Personalization System. Carnegie Mellon University, IBM Corporation, Lockheed Martin Corporation, Philips Research, MIT, 2004, https://www.researchgate.net/publication/227074394_TV_Personalization-_System.

[10] Yunkyoung Lee. Recommendation system using collaborative filtering. The Faculty of the Department of Computer Science, San Jose State University, 2015, https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?article=1444context=etd-_projects.

[11] András Péter Kolbert. A Scalable Recommender System Using Latent Topics and Alternating Least Squares Techniques. NOVA Information Management School, Instituto Superior de Estatística e Gestão de Informação, Universidade Nova de Lisboa, 2017, https://run.unl.pt/bitstream/10362/34383/1/TAA0013.pdf.

[12] Dokumentacija biblioteke Surprise (KNN), onlajn na: https://surprise.readthedocs.io/en/stable/knn_inspired.html.

[13] Dokumentacija biblioteke Surprise (SVD), onlajn na: https://surprise.readthedocs.io/en/stable/matrix_factorization.html.

[14] SHUAI ZHANG, LINA YAO, AIXIN SUN and YI TAY. Deep Learning based Recommender System: A Survey and New Perspective. University of New South Wales Nanyang Technological University, 2019, https://dr.ntu.edu.sg/bitstream/10356/142804/2/Deep%20learning%20based%

20recommender%20system%20-%20A%20survey%20and%20new%20perspectives.pdf.

[15] Ruihui Mu. A Survey of Recommender Systems Based on Deep Learning. College of Computer and Information, Hohai University, Nanjing 210098, China College of Computer and Information Engineering, Xinxiang University, Xinxiang 453003, China, 2018, https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8529185.

[16] Andreas Töscher and Michael Jahrer. The BigChaos Solution to the Netflix Grand Prize. Neuer Weg 23, A-8580 Koflach, Austria, 2009, https://www.asc.ohio-state.edu/statistics/statgen/joul_aut2009/BigChaos.pdf.

[17] CARLOS A. GOMEZ-URIBE and NEIL HUNT. The Netflix Recommender System: Algorithms, Business Value, and Innovation. ACM Transactions on Management Information Systems, 2015, https://dl.acm.org/doi/pdf/10.1145/2843948.

[18] Matthew Prockup, Andreas F. Ehmann, Fabien Gouyon, Erik M. Schmidt, Oscar Celma and Youngmoo E. Kim. Modeling genre with the music genome project: comparing human-labeled attributes and audio features. Electrical and Computer Engineering, Drexel University and Pandora Media Inc., 2015, https://www.ismir2015.uma.es/articles/276_Paper.pdf.

[19] Upendra Shardanand and Pattie Maes. Social Information Filtering: Algorithms for Automating „Word of Mouth". 20 Ames Street Rm. 305, Cambridge, MA 02139, 1995, https://www.stat.cmu.edu/ cshalizi/dm/19/readings/shardanand-and-maes-1995.pdf.

[20] Verónika Peralta. Extraction and Integration of MovieLens and IMDb Data. Laboratoire PRiSM, Université de Versailles, 45, avenue des Etats-Unis, 78035 Versailles Cedex, France, 2007, https://www.researchgate.net/publication/228429288_Extraction_and_Integration_of_MovieLens_and_IMDb_Data.

[21] Will Hill, Larry Stead, Mark Rosenstein and George Furnas. Recommending and evaluating choices in a virtual community of

use. Bellcore, 445 South Street, Morristown, NJ 07962-1910, 1995, https://dl.acm.org/doi/pdf/10.1145/223904.223929.

[22] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. GroupLens: An Open Architecture for Collaborative Filtering of Netnews. ACM, 1994, http://ccs.mit.edu/papers/CCSWP165.html.

[23] Bernard J. Norton. Karl Pearson and Statistics: The Social Origins of Scientific Innovation. Sage Publications, Ltd., 1978, https://faculty.fiu.edu/ blissl/Pearson1.pdf.

Biografija autora

Jovana Pejkić *(Negotin, 6. decembar 1996.)* je završila O.Š. „Vuk Karadžić" u Negotinu 2011. godine, Negotinsku gimnaziju (prirodno-matematički smer) 2019. godine i Matematički fakultet (smer Informatika) na Univerzitetu u Beogradu 2020. godine. Nakon toga je upisala master studije na istom fakultetu. Od 2020. godine radi kao Softver inženjer, baveći se veb programiranjem i testiranjem softvera.