Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Karmen Gostiša

Razvoj priporočilnega sistema za personalizacijo ponudbe trgovine s tekstilnimi izdelki

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Matjaž Kukar

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Raziščite področje priporočilnih sistemov in se seznanite z različnimi paradigmami (priporočanje na podlagi vsebine, priporočanje na podlagi sodelovanja). Preučite različne pristope k zbiranju podatkov (eksplicitno in implicitno ocenjevanje) in implementaciji metod. Nad realnimi podatki večje slovenske trgovine implementirajte in ovrednotite več različic sistemov ter podajte oceno uporabnosti.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ALS	alternating least squares	izmenični najmanjši kvadrati
DCG	discounted cumulative gain	diskontirani kumulativni prispevek
MAE	mean absolute error	srednja absolutna napaka
MAP	mean average precision	srednja povprečna preciznost
MRR	mean reciprocal rank	srednji recipročni rang
NDCG	normalized discounted cumulative gain	normalizirani diskontirani kumulativni prispevek
POS	point of sale	točka prodaje
RMSE	root mean squared error	koren povprečne kvadrirane napake
ROC	receiver operating characteristics	operativne značilnosti sprejemnika
SGD	stochastic gradient descent	stohastični gradientni spust
SOM	self-organizing map	samoorganizirajoče mape
SVD	singular value decomposition	singularni razcep

Povzetek

Naslov: Razvoj priporočilnega sistema za personalizacijo ponudbe trgovine s tekstilnimi izdelki

Avtor: Karmen Gostiša

V diplomskem delu se posvetimo problemu razvoja priporočilnega sistema za trgovino s tekstilnimi artikli na podlagi podatkov o nakupih. V prvem delu pregledamo teoretično ozadje priporočilnih sistemov in povezovalnih pravil. V nadaljevanju opišemo podatke in kvantitativno ter kvalitativno predstavimo njihove osnovne značilnosti. Podrobneje opišemo metode, s katerimi smo se lotili razvoja priporočilnega sistema in sicer, metodi najbližjih sosedov ter matrični razcep. Rezultate metod primerjamo z naivno metodo priporočanja najbolj priljubljenih artiklov in pri vseh dosežemo bistveno boljše rezultate. Najbolje se je izkazal matrični razcep, ki bi ga lahko uporabili v produkcijski aplikaciji.

Ključne besede: priporočilni sistem, priporočanje na podlagi vsebine, priporočanje na podlagi sodelovanja, strojno učenje, matrični razcep, povezovalna pravila.

Abstract

Title: Recommender system for personalized assortment in a clothing store

Author: Karmen Gostiša

In the diploma thesis we are dealing with the problem of developing a recommender system for a clothing store based on transaction data. We start with theoretical basics about recommenders and association rules. Afterwards we describe data and represent its quantitative and qualitative aspects. We continue with the detailed explanation of implemented methods, namely, nearest neighbors and matrix factorization. In the end we compare the results of our methods with naive method of recommending most popular products, achieving much better results. Matrix factorization produced the best results and we would use it in production.

Keywords: recommender system, content-based filtering, collaborative filtering, machine learning, matrix factorization, association rules.

Poglavje 1

Uvod

Dandanes se ljudje v nakupovalnem procesu soočamo s preveliko ponudbo, kar oteži izbiro artikla, ki ga iščemo v danem trenutku. V ta namen so se razvili priporočilni sistemi, s pomočjo katerih lahko podjetje kupcu priporoči prilagojen nabor artiklov in mu s tem prihrani čas ter izboljša nakupovalno izkušnjo. Na drugi strani učinkoviti priporočilni sistemi izboljšajo poslovanje celotnega podjetja, saj pospešujejo prodajo in v nekaterih primerih znižujejo stroške oglaševanja.

Brskanje po spletu nam sicer omogoča izločiti artikle, ki ne ustrezajo našim zahtevam, vendar brskalniki pri razvrščanju zadetkov ne upoštevajo naših interesov in preferenc. Poleg tega to velja le za artikle spletnih trgovin, ki lahko iz uporabnikove zgodovine brskanja izvedo, kateri artikli so uporabnika pritegnili, kako dolgo se je zadržal pri posameznem artiklu, kakšne artikle išče in podobno. Vsi ti podatki so spletnim trgovinam zelo v pomoč pri razvoju učinkovitega priporočilnega sistema. Vendar dandanes še vedno obstaja veliko trgovin, na primer trgovine s tekstilnimi izdelki ali obutvijo, ki ne podpirajo spletnega nakupovanja in tovrstnih podatkov o uporabnikih ne morejo pridobiti. Edina pridobljena informacija je nakup artikla, zato je tem trgovinam veliko težje zgraditi uporabniški profil, ki predstavlja posameznega kupca. Spletne trgovine prikažejo nabor priporočil na svoji spletni strani, medtem ko prve ponavadi pošljejo priporočila preko elektronske ali

navadne pošte.

Razvoja priporočilnih sistemov se lahko lotimo z različnimi pristopi. Pri priporočanju na podlagi vsebine se uporabniku priporočijo artikli, ki so podobni tistim, ki jih je v preteklosti že kupil, gledal ali visoko ocenil. Priporočanje na podlagi sodelovanja pa uporabniku priporoči artikel na podlagi ocen, ki jih je prejel s strani njemu podobnih uporabnikov. Prednost takšnega pristopa je v tem, da lahko sistem na podlagi podobnih uporabnikov priporoča tudi artikle, ki so drugačni od tistih, ki jih je uporabnik že ocenil. Vsak pristop prinaša nekatere slabosti, zato so se razvile hibridne metode, ki kombinirajo oba pristopa in s tem zvišajo uspešnost priporočilnega sistema [1].

V diplomski nalogi se posvetimo razvoju priporočilnega sistema na podatkih trgovine s tekstilnimi izdelki z več podružnicami v Sloveniji, ki na lastno željo ostaja neimenovana. V poglavju 2 predstavimo teoretičen pregled področja priporočilnih sistemov in povezovalnih pravil. V poglavju 3 opišemo podatke in njihovo statistično analizo ter pripravo. Predstavimo tudi rezultate algoritma za iskanje povezovalnih pravil. Sledi poglavje 4 z opisom implementacij metod in poglavje 5 z rezultati, analizo in primerjavo metod. V poglavju 6 je zaključek, kjer ovrednotimo rezultate ter predlagamo možne izboljšave.

Poglavje 2

Priporočilni sistemi

Priporočilni sistemi so v zadnjih letih postali izjemno priljubljeni na različnih področjih. Zelo pomembno vlogo imajo pri podjetjih, ki se ukvarjajo s prodajo. Velika izbira artiklov lahko hitro zmede uporabnika, da ne najde iskanega. Lahko se zgodi tudi, da ne vidi tistega, kar bi ga lahko zanimalo. Trgovine tako uporabljajo sisteme, ki uporabniku priporočijo prilagojen nabor artiklov. S tem dosežejo večjo lojalnost in zadovoljstvo kupcev, sebi pa povečajo prodajo. Priporočilni sistemi so že široko razširjeni v spletnih trgovinah, na primer eBay in Amazon, in na področjih e-uprave, e-izobraževanja ter e-storitev [2]. Počasi prodirajo tudi na področje trgovin, ki želijo ponoviti uspeh spletnih trgovin in povečati prodajo [3].

2.1 Problem priporočanja

Za delovanje priporočilnega sistema so potrebne ocene artiklov. Te so lahko pridobljene eksplicitno ali implicitno. Eksplicitno pridobivanje ocen zahteva interakcijo z uporabnikom. Najpogostejša oblika eksplicitne ocene je celo število z nekega vnaprej določenega intervala, na primer med 1 in 5, kjer višina ocene določa, kako všeč je artikel uporabniku. Takšno ocenjevanje uporabljajo spletne trgovine kot so eBay, Amazon in Google Play. Pogosto je tudi binarno ocenjevanje, pri katerem lahko artikel ocenimo bodisi z 0

bodisi z 1, kjer 0 pomeni, da nam artikel ni všeč in 1, da nam je. Pri sistemih, ki ne dopuščajo ocenjevanja artiklov na številski lestvici, so ocene pridobljene implicitno iz uporabnikove zgodovine nakupov ali drugih vzorcev iskanja informacij o artiklih. Takšen način pridobivanja ocen je značilen za trgovine, ki podatke o uporabniških preferencah pridobivajo preko nakupov preko terminala POS, uporabnika pa identificirajo s kartico zvestobe.

V splošnem je problem priporočanja definiran kot problem napovedovanja ocen artiklov, ki jih uporabnik še ni ocenil. Napovedovanje ocen se lahko izvede s pomočjo:

- ocen, ki jih je uporabnik podelil drugim artiklom,
- ocen, ki so jih artiklu dodelili drugi uporabniki, in
- preostalih informacij o artiklu in uporabnikih.

Naj bo U množica vseh uporabnikov in A množica vseh artiklov, ki jih sistem lahko priporoča. Obe množici sta lahko zelo veliki in vsebujeta svoje značilnosti, na primer vsak uporabnik $u \in U$ je predstavljen s profilom, ki vsebuje starost, spol, prihodek, zakonski stan in podobno. Na podoben način so lahko predstavljeni tudi artikli iz množice A, na primer z barvo in velikostjo.

Naj bo f funkcija koristnosti (angl. *utility function*), ki meri stopnjo koristnosti artikla a uporabniku u. Koristnost artikla je ponavadi predstavljena s številsko oceno (angl. *rating*) $R \in [1, 5]$, kjer višina ocene določa, kako všeč je ta artikel uporabniku. Za funkcijo koristnosti velja:

$$f : U \times A \to R, \tag{2.1}$$

kjer je R popolnoma urejena množica, na primer cela ali realna števila v določenem intervalu. Glavni problem priporočilnih sistemov je v tem, da f ni definirana za celotni prostor $U \times A$, ampak samo za artikle, ki so jih predhodno ocenili uporabniki. Manjkajoče vrednosti se ponavadi izračuna s pomočjo

ocenitve funkcije koristnosti, ki optimizira določena merila uspešnosti, na primer srednja absolutna napaka (angl. *Mean Absolute Error*, s kratico MAE). To pomeni, da za manjkajoče ocene artiklov nastavi takšne vrednosti, ki pri učenju prinesejo najmanjšo napako. Ta pristop se uporablja pri priporočanju z matričnim razcepom, podrobneje opisanim v razdelku 4.3. Ko v prostoru $U \times A$ ni več neznanih vrednosti, se lahko uporabniku priporoči seznam N artiklov, ki imajo napovedano najvišjo oceno in najbolje maksimizirajo f. Tak izhod je znan pod imenom vrhnjih-N (angl. *top-N*) priporočil, uporabljen za implementacijo metode najbližjih sosedov [4].

2.2 Vrste priporočilnih sistemov

Priporočilne sisteme delimo glede na pristop priporočanja na naslednje skupine [4]:

1. Priporočanje na podlagi vsebine (angl. *content-based filtering*): Priporočijo se artikli, ki so podobni tistim, ki jih je uporabnik v preteklosti visoko ocenil.

2. Priporočanje na podlagi sodelovanja (angl. *collaborative filtering*): Uporabniku se priporočijo artikli, ki so jih visoko ocenili njemu podobni uporabniki.

3. Priporočanje s hibridnimi metodami (angl. *hybrid filtering*): Kombinira zgornja pristopa. Sistem izkoristi prednosti enega pristopa, da odpravi slabosti drugega.

2.2.1 Priporočanje na podlagi vsebine

Sodobni informacijski sistemi stremijo k čim bolj učinkovitem načinu interakcije z uporabnikom. To dosežejo s spremljanjem in analizo njegovih akcij, kar privede v izgradnjo uporabniškega profila. Pri takem načinu modeliranja uporabnikov gre za učenje na podlagi vsebine (angl. *content-based learning*),

ki temelji na predpostavki, da se uporabnikovo obnašanje čez čas ne spreminja [4].

Uporabniški profili so lahko zgrajeni neposredno - na podlagi odgovorov, ki jih uporabnik poda v posebej za ta namen izdelan vprašalnik. Rezultati priporočilnih sistemov, ki uporabljajo takšen pristop, so dobri, vendar uporabnika ne moremo prisiliti v reševanje vprašalnika. Pogosto se torej uporabniški profili izdelajo posredno, to je na podlagi uporabnikove zgodovine interakcij s priporočilnim sistemom. Na primer, uporabnik je kupil prvo knjigo v neki knjižni seriji in priporočilni sistem mu priporoči nadaljevanje; ali mu priporoči nov glasbeni album skupine, od katere je uporabnik v preteklosti že kupil album [5].

Vsebino uporabnikovih preteklih dejanj tako lahko uporabimo za napovedovanje akcij v prihodnosti. Pri priporočanju na podlagi vsebine se torej ocena $f(u, a)$ artikla a za uporabnika u izračuna na podlagi ocen, ki jih je uporabnik u podelil artiklom, ki so podobni artiklu a. Za določanje podobnosti med artikli moramo iz vsakega artikla izluščiti njegove značilke. Artikel a je tako predstavljen z vektorjem značilk (angl. *feature vector*)

$$F(a) = [značilka_1(a), značilka_2(a), značilka_3(a), ... \; značilka_n(a)]. \quad (2.2)$$

Podobnost v kontekstu priporočanja predstavlja razdaljo, kjer nižja vrednost pomeni večjo podobnost. Najpogosteje se uporablja evklidsko, Pearsonovo ali kosinusno razdaljo. Izbor mere podobnosti je odvisen od podatkov in metode priporočilnega sistema [4].

2.2.2 Priporočanje na podlagi sodelovanja

Pristop priporočanja na podlagi sodelovanja temelji na predpostavki, da imajo podobni uporabniki podoben okus za artikle. Tipično lahko potek dela takšnega pristopa razdelimo na naslednje korake [6]:

1. Uporabnik izrazi svoje mnenje o artiklih, tako da jim podeli ocene. Te ocene izražajo uporabnikov interes v domeno artiklov.

2. Sistem primerja uporabnikove ocene z ocenami drugih uporabnikov in poišče podobne uporabnike. To so uporabniki, ki imajo podoben okus.

3. Sistem uporabniku priporoča artikle, ki jih uporabnik še ni ocenil in so jih podobni uporabniki ocenili visoko.

Glavna prednost takšnega priporočanja je v vsebinski neodvisnosti. Uporabniku se namreč lahko priporočijo tudi artikli, ki niso podobni artiklom, ki jih je v preteklosti že kupil, ampak izhajajo iz popolnoma druge domene.

Algoritmi, ki uporabljajo ta pristop, torej priporočajo artikle, ki imajo napovedano najvišjo oceno s strani podobnih uporabnikov. Delimo jih v dve skupini [7]:

- algoritmi na podlagi pomnjenja (angl. *memory-based*) in
- algoritmi na podlagi modela (angl. *model-based*).

Algoritmi na podlagi pomnjenja

Metode na podlagi pomnjenja hranijo informacije uporabnikov o preferencah artiklov v računalniškem spominu in do njih dostopajo, ko algoritem to zahteva. Algoritem pri generiranju priporočil za uporabnika takrat poišče njemu podobne uporabnike in nato priporoči artikle, ki so jih podobni uporabniki visoko ocenili. Takšen pristop uporablja algoritem k-najbližjih sosedov, ki je zaradi učinkovitosti in enostavnosti implementacije uporabljen v številnih komercialnih sistemih.

Naj $r_{u,a}$ predstavlja vrednost neznane ocene, ki jo uporabnik u dodeli artiklu a. u' predstavlja podobnega uporabnika iz množice N najbolj podobnih uporabnikov U. Funkcija podobnost(u, u') meri podobnost med uporabnikoma u in u'. Oceno $r_{u,a}$ izračunamo kot agregatno funkcijo, ki na svoj vhod dobi ocene podobnih uporabnikov:

$$r_{u,a} = \text{agr}_{u' \in U} r_{u',a}, u' \neq u. \tag{2.3}$$

Primeri agregatnih funkcij so lahko:

$$r_{u,a} = \frac{1}{N} \sum_{u' \in U} r_{u',a}, \tag{2.4}$$

$$r_{u,a} = c \sum_{u' \in U} \text{podobnost}(u, u') r_{u',a} \text{ in} \tag{2.5}$$

$$r_{u,a} = \bar{r_u} + c \sum_{u' \in U} \text{podobnost}(u, u')(r_{u',a} - \bar{r'_u}), \tag{2.6}$$

kjer je $\bar{r_u}$ povprečna ocena uporabnika u za vse ocenjene artikle in c normalizacijski faktor, izračunan po formuli:

$$c = 1/ \sum_{u' \in U} |\,\text{podobnost}(u, u')|\,. \tag{2.7}$$

Podobnost dveh uporabnikov a in b lahko izračunamo z uteženim povprečjem vseh ocen. Tak pristop uporabljata Pearsonov koeficient korelacije (2.8) in kosinusna podobnost (2.9), ki prideta v poštev predvsem za priporočilne sisteme, ki na vhod prejmejo preferenčne ocene. Množica I_{ab} vsebuje artikle, ki sta jih ocenila oba uporabnika. Ocena $\bar{r_a}$ je povprečna ocena uporabnika a za artikle iz množice I_{ab}.

$$\text{podobnost}_{Pearson}(a, b) = \frac{\sum_{i \in I_{ab}} (r_{a,i} - \bar{r_a})(r_{b,i} - \bar{r_b})}{\sqrt{\sum_{i \in I_{ab}} (r_{a,i} - \bar{r_a})^2 (r_{b,i} - \bar{r_b})^2}} \tag{2.8}$$

$$\text{podobnost}_{cos}(a, b) = \cos\left(\vec{a}, \vec{b}\right) = \frac{\vec{a} \cdot \vec{b}}{||\vec{a}||\ ||\vec{b}||} = \frac{\sum_{i \in I_{ab}} r_{a,i}\ r_{b,i}}{\sqrt{\sum_{i \in I_a} r_{a,i}^2}\ \sqrt{\sum_{i \in I_b} r_{b,i}^2}} \tag{2.9}$$

Obe meri podobnosti sta definirani na intervalu med -1 in 1. Pri priporočanju se želimo izogniti negativnih vrednosti, saj jih nekatere metode

ne podpirajo, na primer nenegativni matrični razcep. Nenegativne vrednosti zagotavljata razdalji 2.10 in 2.11.

$$\mathrm{d}_{Pearson}(a,b) = 1 - \text{ podobnost}_{Pearson} \tag{2.10}$$

$$\mathrm{d}_{cos}(a,b) = 1 - \text{ podobnost}_{cos} \tag{2.11}$$

V primeru, da razpolagamo z implicitno pridobljenimi podatki in vemo le, katere artikle je uporabnik kupil, lahko oceno podobnosti med uporabnikoma izračunamo z Jaccardovim koeficientom podobnosti [8]:

$$\text{podobnost}_{Jaccard}(a,b) = \frac{||P_a \cap P_b||}{||P_a \cup P_b||}, \tag{2.12}$$

kjer $||P_a \cap P_b||$ pomeni število artiklov, ki sta jih kupila tako uporabnik a kot uporabnik b, in $||P_a \cup P_b||$ število različnih artiklov iz nakupov obeh uporabnikov.

Z Jaccardovo mero lahko izračunamo tudi podobnost med artikloma a in b, kjer $||P_a \cap P_b||$ pomeni število uporabnikov, ki so kupili tako artikel a kot artikel b, in $||P_a \cup P_b||$ število uporabnikov, ki so kupili artikel a ali artikel b.

Jaccardov koeficient podobnosti je definiran na intervalu med 0 in 1. Razdaljo izračunamo na sledeč način:

$$\mathrm{d}_{Jaccard} = 1 - \text{ podobnost}_{Jaccard}. \tag{2.13}$$

Algoritmi na podlagi modela

Algoritmi na podlagi modela s pomočjo strojnega učenja zgradijo model, ki se nato uporablja pri napovedovanju ocen artiklov. Primeri takšnih algoritmov so singularni razcep (angl. *Singular Value Decomposition*, s kratico SVD), dopolnjevanje matrik (angl. *matrix completion technique*), verjetnostno iskanje skritih semantik (angl. *probabilistic latent semantic analysis*), regresija (angl. *regression*) in gručenje (angl. *clustering*). Vsi uporabljajo

vnaprej izračunan model in so po priporočilih podobni tistim, ki temeljijo na iskanju najbližjih sosedov.

Poleg naštetih metod vse bolj pomembni postajajo tudi algoritmi strojnega učenja, saj v procesu priporočanja ni pomembna samo ponudba, ampak tudi kdaj je bila ta ponudba dana. Ponudniki artiklov namreč želijo priporočiti artikle ob tistem času, ko je največja verjetnost, da bo kupec artikel kupil.

Najbolj pogosti algoritmi strojnega učenja, ki se uporabljajo za priporočanje, so [4]:

- **Povezovalno pravilo** (angl. *association rule*): Algoritmi skušajo izluščiti pravila, ki napovedujejo verjetnost pojava nekega artikla glede na ostale artikle v transakciji T. Na primer, pravilo $\{čebula, paradižnik\} \Rightarrow \{hamburger\}$ najdeno v prodajnih podatkih trgovine z živili, pomeni, da je zelo verjetno, da stranka, ki kupi čebulo in paradižnik skupaj, kupi tudi hamburger. S takšnimi pravili si lahko trgovina pomaga pri odločanju o postavitvi artiklov, promocijah in gradnji priporočilnih sistemov. Povezovalna pravila predstavimo podrobneje v razdelku 2.5.

- **Odločitveno drevo** (angl. *decision tree*): Temelji na strukturi drevesnega grafa, zgrajenega z množico učnih primerov, za katere je razred znan. Takšno drevo se nato uporabi za klasifikacijo novih primerov. V primeru, da testna množica vsebuje dovolj kvalitetnih podatkov, je točnost priporočil lahko zelo visoka.

- **Gručenje** (angl. *clustering*): Algoritmi za gručenje želijo med podatki odkriti povezave in jih na podlagi podobnosti razvrstiti v skupine. Rezultati dobre metode gručenja so skupine, znotraj katerih je zelo velika podobnost, navzven pa so skupine med seboj zelo različne. Na ta način lahko poiščemo podobne uporabnike in uporabniku priporočimo artikle, ki so jih ti podobni uporabniki dobro ocenili. Najbolj znani metodi gručenja sta metoda K-tih voditeljev (angl. *K-means*) in samoorganizirajoče mape (angl. *Self-Organizing Map*, s kratico SOM).

V uporabi so tudi drugi algoritmi kot so analiza povezav (angl. *link analysis*), regresija in Bayesov klasifikator [7].

2.2.3 Priporočanje s hibridnimi metodami

Hibridne metode združujejo dve ali več priporočilnih tehnik, da bi dosegle boljšo učinkovitost in se znebile slabosti, ki jih prinese posamezna tehnika. Ideja je torej, da kombinacija algoritmov prinese boljša priporočila, saj lahko slabost enega algoritma odpravi drugi algoritem, kar potrjujejo primeri naslednjih metod [9]:

- **Utežena metoda** (angl. *weighted method*): Najbolj enostavni hibridni sistem dobimo z linearno kombinacijo izhodov vseh sistemov. Primer takšnega sistema je P-Tango [1], ki na začetku enakovredno uteži tako sistem priporočanja na podlagi vsebine kot sistem priporočanja na podlagi sodelovanja, nato pa postopoma prilagaja uteži glede na uspeh napovedi (ali se je uporabnik pri ocenjevanju približal napovedani številski oceni).

- **Zamenjalna metoda** (angl. *switched method*): Hibridni sistem izbere priporočilo tistega sistema, ki ima po vnaprej določeni oceni uspešnosti kvalitete boljši rezultat. Primer uporabe te metode je The Daily Learner [10], storitev za prilagojeno prikazovanje novic, ki najprej preveri izhod priporočilnega sistema na podlagi vsebine. V primeru, da izhod ne doseže željene stopnje zaupanja, upošteva priporočilo, ki ga generira sistem na podlagi sodelovanja.

- **Kaskadna metoda** (angl. *cascade method*): Ta metoda uporablja proces postopnega izboljševanja priporočil. Najprej prvi sistem izda neka groba priporočila, ki so nato izboljšana z uporabo drugega sistema. Takšna metoda je zelo učinkovita in odporna na šum v podatkih. Primer kaskadnega hibridnega sistema je EntreeC [9], priporočilni sistem za restavracije. Najprej sistem priporočanja na podlagi vsebine

razporedi restravracije v skupine glede na oceno od najbolj ustrezne do najmanj, nato pa sistem priporočanja na podlagi sodelovanja oceni še vsako restravracijo znotraj skupine.

- **Metoda združevanja značilnosti** (angl. *feature combination*): Sistem priporočanja na podlagi vsebine lahko uporabi dodatne informacije iz izhoda sistema priporočanja na podlagi sodelovanja. V [11] je opisan sistem, ki pri generiranju priporočil za filme poleg vsebinskih informacij o filmih (žanr) in uporabnikih (starost, spol) upošteva še ocene filmov, pridobljene s sistemom za priporočanje na podlagi sodelovanja. Tak sistem se je izkazal za bolj učinkovitega in fleksibilnega kot čisti sistem priporočanja na podlagi vsebine.

- **Metoda obogatitve z značilnostmi** (angl. *feature augmentation*): Izhod prvega sistema je uporabljen pri vhodu v drugega. Na primer, Amazon ima za priporočila knjig razvit sistem na podlagi sodelovanja, ki uporabniku priporoča sorodne avtorje in knjige. Sistem Libra [12] uporabi informaciji o sorodnih avtorjih in knjigah pri svojem sistemu priporočanja na podlagi vsebine in s tem močno izboljša ustreznost priporočil. Od kaskadne metode se razlikuje v tem, da je izhod prvega sistema uporabljen kot vhod v drugega. Izhod največkrat predstavljajo kar ocene artiklov.

2.3 Vrednotenje priporočilnih sistemov

Za ocenjevanje kvalitete priporočilnih sistemov se uporabljajo mere za statistično točnost (angl. *statistical accuracy metrics*) in mere za točnost podpore odločanja (angl. *decision support accuracy metrics*). Izbor primerne mere je odvisen od značilnosti podatkovne zbirke in tipov nalog, ki jih priporočilni sistem opravlja [4].

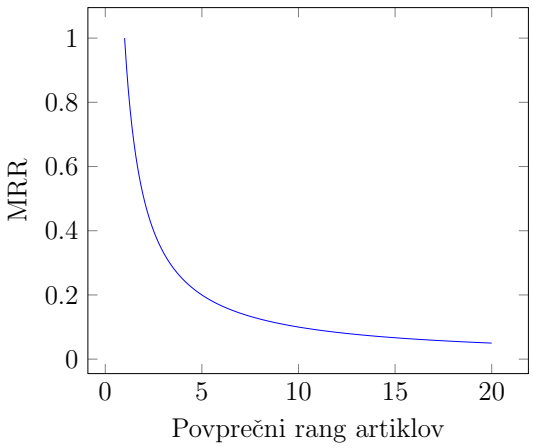
Mere za statistično točnost ovrednotijo točnost priporočilnega sistema tako, da primerjajo napovedano oceno z oceno, ki jo je podal uporabnik.

Najpogostejše takšne mere so srednja absolutna napaka, koren povprečne kvadrirane napake (angl. *Root Mean Squared Error*, s kratico RMSE) in korelacija (angl. *correlation)* [13].

Za vrednotenje priporočilnih sistemov, ki vrnejo seznam rangiranih priporočil, lahko uporabimo tudi mere iz področja odkrivanja informacij (angl. *information retrieval*) kot so srednji recipročni rang (angl. *Mean Reciprocal Rank*, s kratico MRR), srednja povprečna preciznost (angl. *Mean Average Precision*, s kratico MAP), diskontirani kumulativni prispevek (angl. *Discounted Cumulative Gain*, s kratico DCG) in normalizirani diskontirani kumulativni prispevek (angl. *Normalized Discounted Cumulative Gain*, s kratico NDCG). Srednji recipročni rang je povprečje recipročnih rangov, izračunano po naslednji formuli:

$$MRR = \frac{1}{k}\sum_{i=1}^{k}\frac{1}{rang_i}, \tag{2.14}$$

kjer je $rang_i$ mesto, na katerem se je v iteraciji i pojavil artikel t. MRR lahko zavzame vrednosti med 0 in 1, kjer 1 pomeni idealno razvrstitev (vse artikle t je sistem uvrstil na 1. mesto seznama priporočil) [14]. Tako na primer $MRR = 0.2$ pomeni, da je sistem v povprečju artikle t napovedoval na 5. mesto, saj je $\frac{1}{0.2} = 5$. Slika 2.1 prikazuje graf MRR v odvisnosti od povprečnega ranga artiklov.



Slika 2.1: Graf MRR v odvisnosti od povprečnega ranga artiklov.

Mere za točnost podpore odločanja ocenijo, kako dobro priporočilni sistem pomaga uporabniku najti pravo odločitev v smislu pravilne izbire priporočenega artikla. Najpogostejše so stopnja preobrata (angl. *reversal rate*), senzitivnost (angl. *sensitivity*) operativnih značilnosti sprejemnika (angl. *Receiver Operating Characteristics*, s kratico ROC), preciznost (angl. *precision*) in priklic (angl. *recall*) [13].

2.4 Omejitve in izzivi priporočilnih sistemov

V tem razdelku predstavimo glavne omejitve in izzive, ki nam jih prinašajo priporočilni sistemi.

Problem hladnega zagona V splošnem lahko problem hladnega zagona razdelimo na problem novih uporabnikov in problem novih artiklov. Priporočilni sistem mora za generiranje dobrih priporočil dobro poznati uporabnika. V kolikor uporabnik ni ocenil nobenega artikla, opravil nobenega nakupa ali na kakšen drug način podal mnenja o artiklih, priporočilni sistem ne more predlagati ustreznih priporočil. Ta problem se lahko reši z uporabo hibridnih metod ali tehnike pridobivanja znanja o novem uporabniku. Slednja novemu uporabniku ponudi ocenjevanje vzorčnih artiklov, določenih z upoštevanjem priljubljenosti in raznolikosti artiklov, s pomočjo katere čim bolje opišemo uporabniški profil [15]. Do problema novih artiklov pride pri sistemih na podlagi sodelovanja, saj novih artiklov sistem ne more priporočiti, dokler jih ne oceni zadostno število uporabnikov. Tudi ta problem lahko rešimo z uporabo hibridnih metod.

Redkost podatkov Uporabnik običajno oceni zelo majhen delež artiklov, ki so na voljo. Posledično to poslabša možnost natančnega izbora podobnih uporabnikov, kar vodi do slabih priporočil [6]. Za ta problem je bilo predlaganih več metod, ki temeljijo na hibridnih metodah ali tehnikah zmanjševanja števila dimenzij, kot je SVD [16].

Skalabilnost Priporočilni sistemi se soočajo s stalno rastjo novih uporabnikov in artiklov, zato je pomembno, da so skalabilni. To pomeni, da ob rasti količine podatkov še vedno zagotavljajo priporočila v sprejemljivem času z zadostno uspešnostjo [4]. Za reševanje tega problema se uporabljajo različne tehnike kot so gručenje, zmanjševanje števila dimenzij, Bayesove mreže in uporaba distribuiranih algoritmov, ki tečejo na več strežnikih [17].

Problem sivih ovc Pojavi se pri priporočanju na podlagi sodelovanja, kjer določeni uporabniki, imenovani *sive ovce*, niso podobni nobeni skupini uporabnikov, kar oteži celoten proces generiranja priporočil. Za reševanje tega problema se uporabljajo hibridne metode z gručenjem [18].

2.5 Povezovalna pravila

V tem razdelku predstavimo definicijo povezovalnih pravil in mere ocenjevanja moči pravil. Nadaljujemo s postopkom iskanja pravil in predstavitvijo algoritma Apriori, ki ga lahko uporabimo tudi v priporočilnem sistemu.

2.5.1 Definicija

V raziskavi iskanja pravil v velikih podatkovnih zbirkah so avtorji Agrawal, Imielinski in Swami problem povezovalnega pravila (angl. *association rule*) definirali sledeče [19]:

Naj bo $A = a_1, a_2, ..., a_n$ množica n binarnih atributov (artiklov) in $D = t_1, t_2, ..., t_m$ podatkovna zbirka z vsemi transakcijami. Vsaka transakcija iz D ima svoj unikaten identifikator in vsebuje podmnožico artiklov iz A. Povezovalno pravilo je definirano kot implikacija $X \Rightarrow Y$, kjer velja $X, Y \subseteq A$.

2.5.2 Mere ocenjevanja moči pravil

Za ocenjevanje moči povezovalnega pravila se najpogosteje uporabljata meri podpora (angl. *support*) in zaupanje (angl. *confidence*) [20].

Naj bo X množica artiklov, Y artikel, $X \Rightarrow Y$ povezovalno pravilo in T množica transakcij iz dane podatkovne zbirke.

Podpora X je definirana kot delež transakcij t, ki vsebujejo X v T:

$$\text{podpora}(X) = \frac{|t \in T; X \subseteq t|}{|T|}. \tag{2.15}$$

Na primer, v neki podatkovni zbirki ima množica artiklov $X = \{kruh, mleko\}$ podporo enako 0,2. To pomeni, da se pojavi v 20% vseh transakcij.

Zaupanje povezovalnega pravila $X \Rightarrow Y$ je pogojna verjetnost oziroma razmerje transakcij, ki vsebujejo X in Y proti transakcijam, ki vsebujejo le X:

$$\text{zaupanje}(X \Rightarrow Y) = \frac{\text{podpora}(X \cup Y)}{\text{podpora}(X)} = P(Y|X). \tag{2.16}$$

Kot primer vzemimo povezovalno pravilo $\{maslo, kruh\} \Rightarrow \{mleko\}$ z zaupanjem $\frac{0,2}{0,25} = 0,8$. To pomeni, da kupec v 80% primerih ob nakupu masla in kruha kupi še mleko.

2.5.3 Postopek iskanja pravil

Uporabnik običajno vnaprej določi minimalno podporo in zaupanja, ki ju mora imeti povezovalno pravilo. Postopek iskanja pravil nato razdelimo v naslednja koraka [21]:

1. Poiščemo vse podmnožice artiklov, ki imajo podporo večjo ali enako minimalni podpori.

2. Nad dobljenimi podmnožicami artiklov izračunamo še zaupanje in zavržemo tiste podmnožice, ki imajo zaupanje manjše od minimalnega zaupanja.

Prvi korak je računsko zahteven, saj terja iskanje vseh možnih kombinacij artiklov. Število teh je enako moči množice I brez prazne množice, to je $2^n - 1$. Čeprav velikost množice raste eksponentno s številom artiklov, je možno izvesti učinkovito iskanje. Če je neka množica artiklov nepogosta, potem so nepogoste tudi vse njene podmnožice. Na primer, artikel kvas je nepogost. Vse množice, ki vsebujejo kvas, so tudi nepogoste, saj je kvas vsebovan v premajhnem številu transakcij, da bi bile pogoste. Podpora ima namreč lastnost antimonotonosti in to v prid uporabljata algoritma Apriori in Eclat [22].

Za iskanje pogostih množic algoritem Apriori najprej poišče pogoste množice prvega nivoja, torej množice, ki vsebujejo le en artikel (imenovane *pogoste 1-množice*) in imajo podporo večjo ali enako minimalni podpori. Nato generira *pogoste k-množice*, kjer k iterativno povečuje za ena. Ustavi se, ko na k-tem nivoju ni več pogostih množic, saj to pomeni, da jih tudi na višjih nivojih ni [21].

Rezultate iskanja pogostih množic in povezovalnih pravil lahko uporabimo pri priporočanju artiklov uporabnikom, o katerih vemo zelo malo. To so uporabniki, ki so ocenili premalo artiklov, da bi uporabili metode na podlagi sodelovanja, kjer bi iskali podobne uporabnike ali metode na podlagi vsebine, kjer bi iskali podobne artikle.

Poglavje 3

Opis in priprava podatkov

V tem poglavju opišemo podatke, ki smo jih prejeli od velikega trgovca s tekstilnimi izdelki. Nadalje predstavimo statistično analizo podatkov in postopek obdelave ter transformacije podatkov, ki smo jih uporabili za vhod v priporočilni sistem. Na koncu poglavja predstavimo rezultate algoritma za iskanje povezovalnih pravil med artikli, s katerimi vidimo, kateri artikli se pogosto kupujejo skupaj.

3.1 Opis podatkov

Podatke za vhod v priporočilni sistem smo pridobili v obliki tekstovne datoteke, kjer vsaka vrstica opisuje artikel trgovine, ki ga je kupil včlanjeni kupec v obdobju let od 2014 do vključno 2016. Gre torej za implicitno pridobljene podatke o nakupih preko terminala POS. Primer izmišljene vrstice z atributi in njihovimi vrednostmi:

- **identifikator računa**: 429041467,
- **datum nakupa artikla**: 15.6.2015 0:00:00,
- **artikel** (identifikator in ime artikla): 8941 - ž. majica,
- **dobavna šifra artikla** (identifikator in dobaviteljevo ime artikla): 56463 - ž. majica,
- **barva artikla**: midnight navy,

- **velikost artikla**: S,
- **sezona - leto artikla**: pomlad/poletje 2015,
- **blagovna skupina (nivo 1, 2 in 3)**: KŽ majica, K majica Ž, zgornji deli,
- **blagovna znamka (nivo 1, 2 in 3)**: blagovna znamka 1, blagovna znamka 2, blagovna znamka 3,
- **številka kartice** (identifikator kupca artikla): C43245,
- **vrednost** (vrednost, po kateri je bil prodan artikel): 15.23,
- **popust artikla**: 10.0,
- **količina** (število, ki predstavlja količino kupljenih artiklov, vračilo ali menjavo): 1.

3.2 Osnovna statistična analiza podatkov

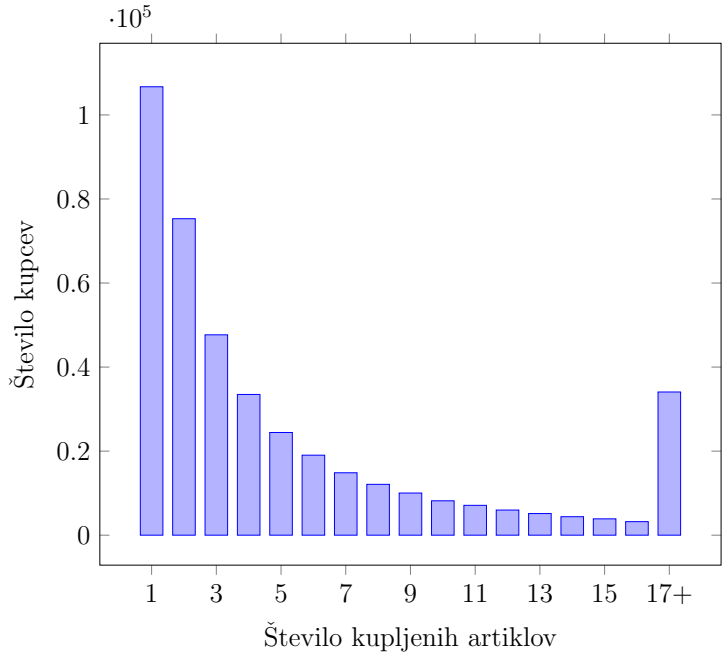
Za statistično analizo podatkov smo uporabili orodje Microsoft Power BI [23]. Osnovne lastnosti podatkovne zbirke so prikazane v tabelah 3.1 in 3.2. Podatka pri številu kupcev in številu različnih artiklov skupaj pomenita število različnih kupcev oz. artiklov skozi vsa tri leta. Podatek o povprečnem številu artiklov v vseh nakupih kupca iz tabele 3.2 nam pove, da se bomo pri razvoju priporočilnega sistema soočili z izzivom redkosti podatkov. Povprečje kupljenih artiklov v vseh treh letih na kupca je namreč nekaj več kot 5, kar je zelo malo v primerjavi s celotnim številom artiklov, ki je 574.327. Graf števila kupcev v odvisnosti od števila kupljenih artiklov si lahko ogledamo na sliki 3.1.

Lastnost	2014	2015	2016	Skupaj	Povprečje
Št. nakupov	427.189	424.652	405.905	1.257.746	419.248,6
Št. prodanih artiklov	764.238	765.784	1.132.815	2.662.837	1.331.418,5
Št. kupcev	182.993	177.908	258.881	415.620	309.891
Št. različnih artiklov	263.113	266.048	351.465	574.327	293.542

Tabela 3.1: Lastnosti podatkovne zbirke.

Lastnost	Povprečje	Modus	Mediana
Št. artiklov v posameznem nakupu	1,803	1	1
Št. artiklov v vseh nakupih kupca	5,094	1	2
Št. nakupov posameznega artikla	3,373	1	2

Tabela 3.2: Povprečje, modus in mediana števila artiklov v posameznem nakupu in vseh nakupih kupca ter števila nakupov posameznega artikla med letoma 2014 in vključno 2016.



Slika 3.1: Graf števila kupcev v odvisnosti od števila kupljenih artiklov.

Podatki vsebujejo naslednje blagovne skupine in podskupine:

- **zgornji deli**: kratka majica ž, kratka majica m, dolga majica ž, dolga majica m, pulover ž, pulover m, srajca ž, srajca m, trenirka ž, trenirka m,
- **spodnji deli**: kratke hlače ž, kratke hlače m, dolge hlače ž, dolge hlače m, jeans ž, jeans m, krilo ž,
- **površnik**: brezrokavnik ž, brezrokavnik m, plašč ž, plašč m, suknjič ž,

suknjič m, jakna ž, jakna m,

- **perilo in kopalke**: perilo ž, perilo m, kopalke ž, kopalke m,
- **obleka**: obleka ž, obleka m,
- **dodatki**: tekstilni dodatki ž, tekstilni dodatki m, torbice ž, torbice m, mali usnjeni dodatki ž, mali usnjeni dodatki m, ure/nakit/očala ž, dežnik/ure/nakit/očala m, šal, rokavice, parfumi/kozmetika, ostali dodatki,
- **obutev**: obutev ž, obutev m, obutev otroška,
- **nogavice**: nogavice ž, nogavice m,
- **otroško**,
- **nakit**,
- **nedoločeno** in
- **ostalo**.

Kategorija nedoločeno zajema artikle, ki nikoli niso bili razvrščeni. Kategorija ostalo vsebuje artikle, ki ne spadajo v nobeno izmed naštetih blagovnih skupin.

3.3 Iskanje povezovalnih pravil

Zanimalo nas je, ali v naši podatkovni zbirki obstajajo pravila, ki prikazujejo, kateri artikli so pogosto zastopani znotraj istega nakupa. Zaradi prevelikega števila različnih artiklov in premajhnega števila nakupov pravil znotraj nakupov iste stranke nismo odkrili. Poizkusili smo še na blagovnih skupinah in podskupinah. Podatke smo razdelili na vrstice, kjer je posamezna predstavljala nakup z blagovnimi podskupinami kupljenih artiklov, iz katerih smo za generalizacijo izločili oznako spola. Izločili smo tudi blagovni skupini nedoločeno in ostalo.

Za implementacijo smo uporabili algoritem Apriori iz knjižnice *arules* za programski jezik R [24]. Branje podatkov in generiranje povezovalnih pravil smo izvedli z naslednjima ukazoma:

```
transactions <- read.transactions("data.csv", sep=",",
                rm.duplicates=TRUE, format="basket")
rules = apriori(transactions, parameter = list(supp = 0.011,
        conf = 0.5, maxtime = 120, target = "rules"))
```

Vrednosti parametrov minimalna podpora in zaupanje smo določili eksperimentalno. V tabeli 3.3 so prikazani rezultati, iz katerih lahko sklepamo, da je kratka majica artikel, ki se največkrat pojavi v kombinaciji z drugimi blagovnimi skupinami. Pri gradnji priporočilnega sistema si lahko s temi rezultati pomagamo pri priporočanju kupcem z malo nakupi in sicer na seznam priporočil umestimo najbolj priljubljene kratke majice.

Povezovalno pravilo	Podpora	Zaupanje
{kratke hlače} ⇒ {kratka majica}	0,027	0,552
{dolga majica, pulover} ⇒ {kratka majica}	0,014	0,529
{jakna, pulover} ⇒ {kratka majica}	0,013	0,559
{srajca, pulover} ⇒ {kratka majica}	0,011	0,591
{jeans, pulover} ⇒ {kratka majica}	0,011	0,592
{dolge hlače, pulover} ⇒ {kratka majica}	0,012	0,594

Tabela 3.3: Povezovalna pravila in njihova podpora ter zaupanje.

3.4 Priprava podatkov

Preden smo podatke podali kot vhod v priporočilni sistem, jih je bilo potrebno transformirati v ustrezno obliko. To smo storili v programskem jeziku C# in na koncu podatke zapisali v datoteko CSV pomočjo knjižnice CSV helper [25].

Korak 1 Atribut *artikel*, ki vsebuje enoličen niz numeričnih znakov in ime artikla, smo ločili na *identifikator artikla* in *ime artikla*.

Korak 2 Iz podatkov smo izločili atribut *barva*, saj smo po podrobnem

pregledu možnih vrednosti barv ugotovili, da so neenotne, pomanjkljive in nesmiselne - nizi numeričnih znakov, ki enolično ne predstavljajo barve. Izločili smo tudi atribut *velikost*, saj njegove vrednosti niso bile enotne med različnimi blagovnimi znamkami in skupinami.

Korak 3 Atribut *blagovna skupina (nivo 2)* vsebuje ime blagovne skupine in oznako spola (m ali ž), za katerega je namenjen artikel. Dodali smo nov atribut *spol artikla* s pomočjo metode za delo z nizi *substring*, ki vrne podniz podanega niza.

Korak 4 Pri atributih *blagovna skupina (nivo 1)* in *blagovna skupina (nivo 2)* smo opazili ponavljanje vrednosti, zato smo ta dva atributa združili v enega - *blagovna podskupina*.

Korak 5 Iz atributov *blagovna znamka (nivo 1), blagovna znamka (nivo 2)* in *blagovna znamka (nivo 3)* smo izdelali atributa *identifikator blagovne znamke* in *ime blagovne znamke*.

Korak 6 Želeli smo pridobiti še nekaj vsebinskih podatkov o uporabniku. Za *spol uporabnika* smo privzeli prevladujočo vrednost atributa *spol artikla* v uporabnikovih nakupih. Iz podatkov o cenah artiklov, ki jih je kupil uporabnik, smo izračunali še povprečno ceno uporabnikovih nakupov in vrednost shranili v atribut *povprečna vrednost nakupov*.

Korak 7 V zadnjem koraku smo izločili 5.000 kupljenih artiklov različnih kupcev in jih v fazi testiranja uporabili za neodvisno testno množico. Izločili smo tudi 5.000 artiklov za validacijsko množico, ki smo jo uporabili pri iskanju optimalnih parametrov za implementirane metode. Učni množici smo dodelili 851.749 artiklov.

Pri izračunu števila različnih artiklov glede na enolični identifikator artikla za vsa leta smo dobili 574.327 (tabela 3.1), kar se nam je zdelo zelo veliko število. Pri pregledu podatkov smo ugotovili, da atribut *identifikator artikla* ni skupen za artikle, ki so sicer enaki, le drugačne velikosti ali barve, vendar imajo skupnih prvih šest numeričnih znakov, na podlagi katerih smo ločili različne artikle. Število različnih artiklov se nam je še vedno zdelo veliko in odkrili smo, da nekateri artikli z različnim identifikatorjem artikla in enako dobavno šifro artikla predstavljajo isti artikel in obratno (tabela 3.4).

Identifikator artikla	Dobavna šifra dobavitelja	Artikel
46	20	kratka majica
97	20	kratka majica
112	20	kratka majica
5	32	dolge hlače
5	34	dolge hlače
5	78	dolge hlače

Tabela 3.4: Isti artikli z različnimi identifikatorji artiklov v prvem primeru in različnimi dobavnimi šiframi dobavitelja v drugem primeru.

Za rešitev zgornjega problema smo napisali algoritem, ki uporablja dva slovarja, enega za hranjenje atributa *identifikator artikla* in enega za atribut *dobavna šifra dobavitelja*. Pri branju vrstic podatkov smo preverili, ali slovarja že vsebujeta takšen vnos. V tem primeru smo prebranemu artiklu dodelili vrednosti, ki jo je vseboval slovar pod ustreznim ključem, sicer smo v slovar dodali nov vnos. Število različnih artiklov nam je uspelo skrčiti na 88.844. Primer izmišljene vrstice z atributi in njihovimi vrednostmi po koncu postopka:

- **identifikator računa**: 378042431,
- **datum nakupa artikla**: 9.4.2014 0:00:00,
- **identifikator artikla**: 84613,
- **ime artikla**: m. majica,

- **sezona artikla**: pomlad/poletje,
- **leto artikla**: 2014,
- **blagovna skupina artikla**: zgornji deli,
- **blagovna podskupina artikla**: kratka majica m,
- **identifikator blagovne znamke artikla**: 12654,
- **ime blagovne znamke artikla**: blagovna znamka,
- **spol artikla** (spol, za katerega je namenjen artikel): m,
- **vrednost artikla** (cenovna vrednost artikla, po kateri je bil prodan): 12.99,
- **popust artikla**: 5.0,
- **količina** (število, ki predstavlja količino kupljenih artiklov, vračilo ali menjavo): 1,
- **identifikator kupca**: C794823,
- **spol kupca**: m,
- **povprečna vrednost nakupov kupca**: 15.32.

Naša podatkovna zbirka nakupov vsebuje le podatek, ali je uporabnik artikel kupil, ne pa tudi ocene artikla na številski ocenjevalni lestvici (npr. med 1 in 5). Posledično ne vemo nič o artiklih, ki uporabniku niso všeč. Na podlagi majhnega števila nakupov je težko sklepati o uporabnikovem profilu in poiskati njemu podobne uporabnike, zato smo izločili vse uporabnike, ki imajo manj kot 5 kupljenih artiklov. To število smo določili po premisleku, koliko artiklov je najmanj potrebno za dobro priporočilo in z ozirom na povprečno število kupljenih artiklov na kupca, ki je znašalo nekaj več kot 5. Hkrati nismo želeli odrezati prevelikega števila kupcev. V končni produkcijski aplikaciji bi takšnim kupcem priporočali najbolj priljubljene artikle. Po tem koraku je ostalo 152.449 kupcev. Podobno smo izločili tudi artikle, ki so bili kupljeni manj kot desetkrat. Število artiklov smo tako zmanjšali na 16.153. Po tem filtriranju je ostalo 851.749 vrstic kupljenih artiklov v obdobju treh let.

Spodnja programska koda naloži podatke in jih filtrira po prej opisanem postopku.

```
podatki = pd.read_csv("./data.csv", sep=",",
          names=["uporabnik", "artikel"])
st_nakupov_art = podatki["artikel"].value_counts()
st_nakupov_up = podatki["uporabnik"].value_counts()
podatki = podatki[podatki["artikel"].
          isin(st_nakupov_art[st_nakupov_art > 5].index)]
podatki = podatki[podatki["uporabnik"].
          isin(st_nakupov_up[st_nakupov_up > 10].index)]
```

V naslednjem koraku smo identifikatorje uporabnikov in artiklov prevedli na števila z intervala med 1 in N oziroma 1 in M, kjer N pomeni število vseh uporabnikov in M število vseh artiklov. S tem smo bistveno pohitrili vse nadaljnje operacije, saj jim ni bilo treba primerjati nizov, kar je zelo časovno in prostorsko zahtevno, vendar samo nekaj bajtov za število.

```
def ustvariSlovar(vrednosti):
    i = range(0, len(vrednosti))
    v2i = pd.Series(indeks, index=vrednosti)
    i2v = pd.Series(vrednosti, index=i)
    return v2i, i2v

v2i_art, i2v_art = ustvariSlovar(podatki["artikel"].unique())
v2i_up, i2v_up = ustvariSlovar(podatki["uporabnik"].unique())
```

Za implementacijo metod, opisanih v naslednjem poglavju, smo zgradili matriko nakupov (tabela 3.5), kjer vrstica predstavlja uporabnika in stolpec artikel. Posamezno število v matriki predstavlja število nakupov. Za učinkovito delovanje s pomnilnikom smo uporabili podatkovno strukturo redka matrika (angl. *sparse matrix*), saj je celotna matrika vsebovala kar 99.84% ničel.

	artikel_1	artikel_2	artikel_3	...	artikel_m
uporabnik_1	0	1	0	...	1
uporabnik_2	0	0	1	...	1
uporabnik_3	1	1	2	...	1
...	...	...	...	...	...
uporabnik_n	2	0	0	...	1

Tabela 3.5: Matrika nakupov.

Postopek izdelave redke matrike prikazuje spodnja programska koda:

```
N = v2i_up.shape[0]
M = v2i_art.shape[0]

v = podatki["uporabnik"].map(v2i_up)
s = podatki["artikel"].map(v2i_art)

nakupi = sparse.coo_matrix((np.ones(len(podatki)), (v, s)),
         shape=(M, N), dtype="float32").tocsr()
```

Poglavje 4

Implementacija metod

V tem poglavju bodo opisane metode, s katerimi smo se lotili razvoja priporočilnega sistema za podatke, opisane v poglavju 3.

Metode smo implementirali v programskem jeziku Python s pomočjo knjižnic NumPy, SciPy in pandas.

4.1 Priporočanje najbolj priljubljenih artiklov

Najprej smo preizkusili priporočanje najbolj priljubljenih artiklov glede na število nakupov. Ta metoda je naivna, saj vrne seznam priporočil, ki je enak za vse uporabnike. Na drugi strani je idealna za priporočanje uporabnikom, ki nimajo zgodovine nakupov ali je njihovo število kupljenih artiklov zelo majhno.

4.2 Priporočanje z metodo k-najbližjih sosedov

V razdelku 2.2.2 smo opisali algoritme na podlagi pomnjenja, kamor spada tudi metoda k-najbližjih sosedov. Jaccardovo razdaljo (2.13) smo implementirali s sledečo funkcijo:

```
def jaccard(x, y):
    x[x > 0] = 1
    y[y > 0] = 1
    a = x + y
    return 1 - numpy.sum(a == 2) / numpy.sum(a > 0)
```

kjer vhodna parametra x in y predstavljata tabeli kupljenih artiklov s strani dveh uporabnikov. Na mestih, kjer je uporabnik kupil artikel je 1, sicer 0.

4.2.1 *k*-najbližjih uporabnikov

Implementacijo metode opisujejo naslednji koraki:

1. Izračun podobnosti uporabnikov Za uporabnika, za katerega smo želeli pridobiti priporočila, smo izračunali podobnost z ostalimi uporabniki. Za mero podobnosti med uporabnikoma a in b smo uporabili Jaccardov koeficient podobnosti (2.12) in funkcijo podobnosti, ki smo jo napisali po lastnem premisleku glede na razpoložljive podatke ter tako dodali nekaj predznanja o samem problemu. Definirali smo jo sledeče:

$$\text{podobnost}_{po\ meri}(a, b) = \frac{1}{2} d_{spol} + \frac{1}{2} d_{vrednost}, \tag{4.1}$$

kjer velja:

- $d_{spol} = 1$, če sta uporabnika istega spola, in 0 sicer,
- $d_{vrednost} = 1 - \left| \frac{d_{\overline{vrednost_a}} - d_{\overline{vrednost_b}}}{d_{\overline{vrednost_a}} + d_{\overline{vrednost_b}}} \right|$, kjer je $d_{\overline{vrednost_a}}$ povprečna vrednost nakupov uporabnika a in $d_{\overline{vrednost_b}}$ povprečna vrednost nakupov uporabnika b.

V kodi smo funkcijo 4.1 implementirali na sledeč način:

```
def podobnostUporabnikovPoMeri(a, b):
    d_spol = (a[0,0] == b[0,0])
    d_vrednost = 1 - abs((a[0,1] - b[0,1]) / (a[0,1] + b[0,1]))
    return 1/2 * d_spol + 1/2 * d_vrednost
```

kjer sta a in b tabeli uporabnikov, v kateri prvi element predstavlja spol uporabnika in drugi element povprečno vrednost nakupov uporabnika.

V prvi različici metode smo za končno podobnost dveh uporabnikov vzeli le Jaccardov koeficient, v drugi pa smo podobnost izračunali uteženo na sledeč način:

$$\text{podobnost}(a,b) = \frac{3}{4}\,\text{podobnost}_{Jaccard}(a,b) + \frac{1}{4}\,\text{podobnost}_{po\ meri} \qquad (4.2)$$

oziroma v kodi:

```
def podobnostUporabnikov(a, b):
    return  3/4 * jaccard(a[0,2:], b[0,2:]) +
            + 1/4 * podobnostUporabnikovPoMeri(a, b)
```

Razdaljo med dvema uporabnikoma smo izračunali tako, da smo od 1 odšteli vrednost, ki jo je vrnila zgornja funkcija.

2. Iskanje podobnih uporabnikov Za obravnavanega uporabnika u smo poiskali k najbolj podobnih uporabnikov in jih shranili v matriko. Za k smo privzeli različne vrednosti z intervala med 25 in 500. Parameter *metric* predstavlja izbrano razdaljo med uporabnikoma.

```
knn = NearestNeighbors(metric=jaccard, algorithm="brute")
knn.fit(nakupi)
sosedje = knn.kneighbors(u, n_neighbors=k,return_distance=False)
nakupi_sosedov = nakupi[sosedje[0]]
```

3. Izračun napovedi ocen artiklov in izgradnja priporočil Za vsak posamezni artikel smo izračunali napoved ocene artikla uporabnika u, torej verjetnost, da bo artikel kupil. Verjetnost smo dobili tako, da smo prešteli, koliko podobnih uporabnikov je artikel kupilo in vsoto delili s k. Napovedi ocen smo nato razporedili padajoče. V produkcijski aplikaciji bi uporabniku u priporočili najvišje uvrščenih N artiklov, ki jih še ni kupil.

```
ocene_artiklov = numpy.mean(nakupi_sosedov, axis=0)
ocene_artiklov[u.toarray().astype("bool")] = 0;
priporocila = numpy.argsort(ocene_artiklov)[::-1]
```

4.2.2 *k*-najbližjih artiklov

Matriko nakupov iz tabele 3.5 smo transponirali, tako da so vrstice predstavljale posamezne artikle in stolpci uporabnike. Nadaljnji postopek smo razdelili na naslednje korake:

1. Izračun podobnosti artiklov Za mero podobnosti med artikloma a in b smo uporabili Jaccardov koeficient podobnosti (2.12) in svojo funkcijo podobnosti, ki smo jo napisali glede na razpoložljive podatke o artiklih ter jo definirali kot:

$$\text{podobnost}_{po\ meri}(a,b) = \frac{3}{4}d_{spol} + \frac{1}{12}d_{blag.} + \frac{1}{12}d_{sez.} + \frac{1}{12}d_{cena}, \tag{4.3}$$

kjer velja:

- $d_{spol} = 1$, če sta artikla namenjena istemu spolu in 0 v nasprotnem primeru,
- $d_{blag.} = 1$, če sta artikla iste blagovne podskupine in 0 v nasprotnem primeru,
- $d_{sez.} = 1$, če sta artikla iz iste sezone in 0,5 sicer,
- $d_{cena} = \frac{c_{min}}{c_{max}}$, kjer je c_{min} cena artikla z nižjo ceno in c_{max} cena artikla z višjo ceno.

V programu smo funkcijo 4.3 implementirali tako:

```
def podobnostArtiklovPoMeri(a, b):
    d_spol = (a[0,1] == b[0,1])
    d_blag = (a[0,0] == b[0,0])
    d_sez = 1 if (a[0,2] == b[0,2]) else 0.5
```

```
    d_cena = min(a[0,3], b[0,3]) / max(a[0,3], b[0,3])
    return 3/4 * d_spol + 1/12 * d_blag +
           + 1/12 * d_sez + 1/12 * d_cena
```

kjer vhodna parametra a in b predstavljata tabeli artiklov z elementi blagovna skupina, spol, sezona in cena.

Tako kot v prejšnjem razdelku, smo tudi tu preizkusili delovanje metode z dvema različnima načinoma izračuna podobnosti. Za prvega smo vzeli mero po Jaccardu, za drugega pa podobnost, izračunano na naslednji način:

$$\text{podobnost}(a,b) = \frac{3}{4}\,\text{podobnost}_{Jaccard}(a,b) + \frac{1}{4}\,\text{podobnost}_{po\ meri} \qquad (4.4)$$

in v kodi:

```
def podobnostArtiklov(a, b):
    return  3/4 * jaccard(a[0,4:], b[0,4:]) +
            + 1/4 * podobnostArtiklovPoMeri(a, b)
```

Razdaljo med dvema artikloma smo izračunali tako, da smo od 1 odšteli vrednost, ki jo je vrnila zgornja funkcija.

2. Iskanje podobnih artiklov Za vsak artikel, ki ga obravnavani uporabnik še ni kupil, smo poiskali k najbolj podobnih artiklov. Poskusili smo z vrednostmi k z intervala med 25 in 500.

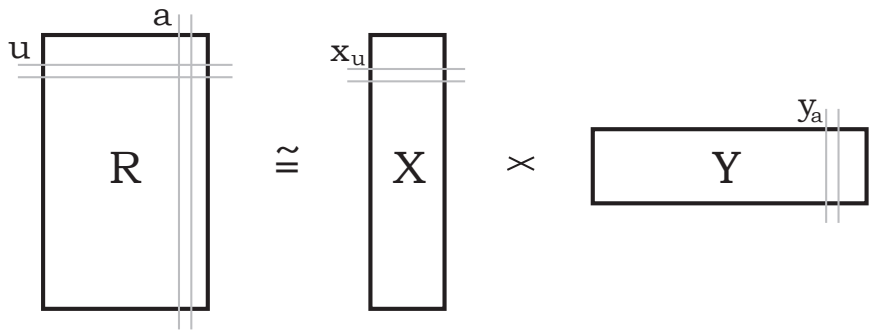
3. Izračun napovedi ocen artiklov in izgradnja priporočil Iz preostalih kupljenih artiklov uporabnika smo prešteli, koliko se jih pojavi na seznamu k najbolj podobnih artiklov in vrednost delili s k. Dobljeno število je predstavljalo napoved ocene artikla uporabnika. To smo storili za vsak artikel, ki ga obravnavani uporabnik še ni kupil in na koncu seznam razvrstili padajoče. Tako kot pri prejšnji metodi bi tudi tukaj v produkcijski aplikaciji uporabniku s seznama priporočali zgornjih N artiklov.

4.3 Priporočanje z matričnim razcepom

Poleg metod, ki za priporočanje na podlagi sodelovanja uporabljajo algoritme na podlagi pomnjenja, obstaja še pristop, ki uporablja vnaprej zgrajen model. Takšen pristop je značilen za metodo priporočanja z matričnim razcepom, osnovano na prikritih faktorjih (angl. *latent factor model*).

Na primer, priporočanja filmov nekemu uporabniku se lahko lotimo tako, da na podlagi ocen filma poiščemo njemu podobne uporabnike in mu priporočimo filme, ki so jih visoko ocenili in jih uporabnik še ni ocenil. Takšne metode temeljijo na izgradnji soseske za danega uporabnika in ne upoštevajo drugih faktorjev, na primer količino akcije v filmih, naravnanost h komedijam ali dramam in podobno. Modeli, osnovani na prikritih faktorjih, skušajo takšne faktorje odkriti in filme opisati v prikritem prostoru skupin uporabnikov, ki pa jih tudi še morajo odkriti na podoben način. Takšno predstavitev podatkov v latentnem prostoru skupin stvari in skupin uporabnikov lahko pridobimo z enotnim postopkom, imenovanim matrični razcep [26].

V našem primeru smo matriko nakupov R (tabela 3.5) predstavili z matrikama $X_{M\times K}$ in $Y_{K\times N}$ tako, da velja $R \approx XY$ in da je $K << M, N$. Konstanta K predstavlja število latentnih dimenzij ali stopnjo razcepa. M je število vseh uporabnikov in N je število vseh artiklov. Konstanta K je veliko manjša od dimenzij M in N, kar pomeni, da s produktom XY prav gotovo ne bomo vedno mogli predstaviti matrike R. Matrika X je matrika uporabniških profilov v latentnem prostoru artiklov, torej prostoru tipičnih vrst artiklov, ki jih pridobimo z razcepom matrike. Latentni profil uporabnika u označimo z vektorjem $\vec{x_u}$. Matrika Y je matrika profilov artiklov v prostoru latentnih uporabnikov, vektor $\vec{y_a}$ pa predstavlja profil artikla a v tem prostoru. Skalarni produkt latentnih profilov je približek ocene artikla a uporabnika u (slika 4.1): $\hat{r}_{ua} = \vec{x_u}\vec{y_a} = \sum_{k=1}^{K} x_{uk}y_{ka}$. S tem izračunom lahko napovemo, kako verjetno je, da bo uporabniku všeč artikel, ki ga še ni kupil.



Slika 4.1: Skalarni produkt latentnega profila uporabnika $\vec{x_u}$ in latentnega profila artikla $\vec{y_a}$ predstavlja približek ocene artikla a uporabnika u, torej $\hat{r}_{ua}$.

Glavni problem faktorizacije matrike nakupov je redkost matrike. Zgodnje metode so problem reševale tako, da so zapolnile prazne vrednosti in nato izvedle razcep. Ta postopek se je izkazal za počasnega in požrešnega, saj zahteva shranjevanje kar nekaj novih podatkov, ki lahko ob neprimerni metodi izračuna celo pokvarijo izvirne podatke. Novejše raziskave [27, 28, 29] predlagajo modeliranje na izvirnih podatkih in uporabo regularizacije za preprečitev prekomernega prilagajanja [26].

Naša podatkovna zbirka vsebuje implicitno pridobljene podatke o nakupih in zato pri učenju modela ne moremo uporabiti negativnih primerov oziroma artiklov, ki uporabniku niso všeč. V [30] je opisan postopek uteženega matričnega razcepa, s katerim smo se lotili reševanja tega problema. Uvedli smo množico binarnih atributov p_{ua}, kjer posamezni predstavlja prednost (angl. *preference*) artikla a uporabnika u. Vrednosti p_{ua} so pridobljene iz matrike nakupov (tabela 3.5), kjer r_{ua} predstavlja število nakupov artikla a uporabnika u:

$$p_{ua} = \begin{cases} 1, & r_{ua} > 0 \\ 0, & r_{ua} = 0. \end{cases}$$

Z drugimi besedami, če je uporabnik kupil artikel a ($r_{ua} > 0$), potem sklepamo, da je uporabniku u všeč artikel a ($p_{ua} = 1$). Na drugi strani, če u ni

nikoli kupil a, označimo p_{ua} z 0. V tem primeru ne moremo sklepati, da artikel uporabniku ni všeč. Lahko da ga ni videl ali pa ni kupil zaradi previsoke cene oziroma neustrezne velikosti artikla. V splošnem lahko rečemo, da večji kot je r_{ua}, bolj všeč je artikel uporabniku. Uvedimo množico spremenljivk c_{ua}, ki predstavljajo zaupanje (angl. *confidence*), izračunano sledeče [30]:

$$c_{ua} = 1 + \alpha r_{ua}. \tag{4.5}$$

Na ta način pridobimo nekaj minimalnega zaupanja za vsak par uporabnik-artikel. Za tiste, ki jih je uporabnik dejansko kupil, se vrednost ustrezno poveča. Faktor povečanja zaupanja je določen s parametrom α, v našem primeru se je za najbolj ugodno nastavitev izkazala $\alpha = 50$. To vrednost smo dobili tako, da smo na podatkih iz validacijske množice poskušali različne vrednosti in opazovali vrednost ocene uspešnosti MRR.

Za približek ocene artikla (slika 4.1) želimo, da je čim boljši, oziroma da je napaka, ki jo s tem približkom naredimo, čim manjša. Pri učenju faktorjev torej minimiziramo regulariziran kvadrat napake [30]:

$$\min_{\vec{x_u},\vec{y_a}} \sum_{u,a} c_{ua}(p_{ua} - x_u^T y_a)^2 + \lambda(\sum_u ||x_u||^2 + \sum_a ||y_a||^2). \tag{4.6}$$

Konstanta λ je potrebna za regularizacijo modela, da ne pride do prekomernega prilagajanja. Tako kot pri α smo tudi tu v okviru testiranja na validacijski množici preizkusili več različnih vrednosti in dobili najbolj ugodno $\lambda = 256$.

Obstajata dve metodi za reševanje enačbe 4.6. Prva je stohastični gradientni spust (angl. *Stochastic Gradient Descent*, s kratico SGD), druga pa izmenični najmanjši kvadrati (angl. *Alternating Least Squares*, s kratico ALS). V splošnem je prva enostavnejša in hitrejša, vendar obstajata vsaj dva primera, ko je ALS primernejša metoda. Prvi je, ko lahko sistem izkoristi vzporedno obdelavo procesorskih enot (angl. *paralel processing*), saj se pri ALS latentni profil artikla izračuna neodvisno od drugih faktorjev artiklov in latentni profil uporabnika neodvisno od ostalih faktorjev uporabnikov [26].

Drugi primer pa je za sisteme, ki temeljijo na implicitnih ocenah. Pri takšnih sistemih neznanih ocen artiklov pri reševanju enačbe 4.6 ne izpustimo, saj ima prav vsak par uporabnik-artikel nekaj minimalnega zaupanja (enačba 4.5). SGD v osnovi iterira čez vsak učni primer, kar se pri velikem številu uporabnikov in artiklov lahko izkaže za časovno zelo zahtevno operacijo [30].

V enačbi 4.6 sta tako x_u kot y_a neznani vrednosti, kar pomeni, da enačba ni konveksna. Če označimo eno izmed neznanih vrednosti za konstanto, postane problem kvadratne časovne zahtevnosti in je lahko rešen optimalno. Metoda ALS izmenično fiksira enega izmed latentnih vektorjev in rešuje drugega z uporabo metode najmanjših kvadratov. To počne toliko časa, dokler ne doseže konvergence [31].

Metodo smo implementirali v programskem jeziku C# s pomočjo knjižnice MyMediaLite [32]. Spodnji kos kode prikazuje izgradnjo in učenje modela ter ovrednotenje rezultatov.

```
using MyMediaLite;

var recommender = new WRMF
{
    Alpha = 50,
    Regularization = 256,
    NumFactors = 270,
    Feedback = trainingData
};

recommender.Train();

var results = recommender.Evaluate(testData, trainingData);
```

Poglavje 5

Rezultati in analiza implementiranih metod

V tem poglavju predstavimo način ovrednotenja rezultatov in analiziramo ter primerjamo rezultate implementiranih metod.

Za ovrednotenje rezultatov implementiranih metod smo uporabili 10-kratno testiranje na neodvisni testni množici, ki je v posamezni iteraciji zajemala 500 artiklov. Učni in testni primeri se med različnimi metodami niso razlikovali. V vsaki ponovitvi metode smo naredili naslednje:

1. Iz podatkovne zbirke nakupov smo naključno uniformno izbrali 500 vrstic in jih premaknili v testno množico. Pri tem smo pazili, da smo izbrali 500 artiklov, ki pripadajo različnim kupcem. Preostale vrstice smo dodelili učni množici.

2. Z izbrano metodo smo model naučili na učni množici in za uporabnike, katerih artikel smo premaknili v testno množico, napovedali ocene vseh artiklov.

3. Ocene artiklov smo razvrstili padajoče in pogledali, na katerem mestu se je pojavil testni primer za posameznega uporabnika.

Nato smo izračunali vrednost naslednjih metrik:

- **priklic vrhnjih N priporočil** (z oznako rec@N): delež testnih primerov, ki so se pojavili na prvih N mestih,
- **srednji recipročni rang** (z oznako MRR): povprečje recipročnih rangov testnih primerov (2.14).

Vsa testiranja smo izvajali na računalniku s procesorjem Intel Xeon E5-2560 v3 z desetimi jedri in frekvenco 2,30 GHz ter 16 GB delovnega pomnilnika. Podatki so bili za hitrejši dostop shranjeni na pogonu SSD.

5.1 Priporočanje najbolj priljubljenih artiklov

Rezultati metode so naslednji:

- rec@5: 0,52%
- rec@10: 0,60%
- MRR: 0,0232

MRR je znašal 0,0232, kar pomeni, da je metoda v povprečju napovedovala skriti testni primer na 43. mesto. Rezultati niso vzpodbudni, kar je razumljivo, saj je metoda naivno vsem uporabnikom priporočala enak seznam artiklov, zgrajen na podlagi števila nakupov posameznih artiklov.

5.2 Priporočanje z metodo k-najbližjih sosedov

V tem razdelku bomo predstavili in analizirali rezultate metod k-najbližjih uporabnikov in artiklov z različnimi vrednostmi parametra k in dvema načinoma izračuna podobnosti sosedov.

5.2.1 k-najbližjih uporabnikov

Glede na način izračuna podobnosti uporabnikov smo ločili dve metodi:

- **metoda 1**: Jaccardov koeficient podobnosti,
- **metoda 2**: utežen izračun podobnosti z Jaccardovim koeficientom in funkcijo podobnosti po meri.

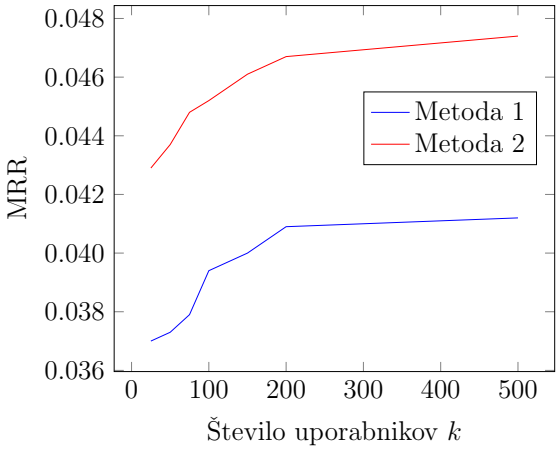
Za število najbližjih uporabnikov k smo izbrali vrednosti z intervala med 25 in 500. Rezultati metode 1 so prikazani v tabeli 5.1, metode 2 v tabeli 5.2, slika 5.1 pa prikazuje graf MRR v odvisnosti od števila najbližjih uporabnikov k pri obeh metodah.

št. uporabnikov	rec@5 (%)	rec@10 (%)	MRR
$k = 25$	0,61	1,72	0,0370
$k = 50$	0,69	1,78	0,0373
$k = 75$	0,77	1,88	0,0379
$k = 100$	0,86	1,94	0,0394
$k = 150$	0,95	2,04	0,0400
$k = 200$	1,00	2,18	0,0409
$k = 500$	1,28	2,25	0,0412

Tabela 5.1: Rezultati metode k-najbližjih uporabnikov glede na k z uporabo Jaccardove mere podobnosti.

št. uporabnikov	rec@5 (%)	rec@10 (%)	MRR
$k = 25$	1,12	2,20	0,0429
$k = 50$	1,23	2,24	0,0437
$k = 75$	1,34	2,35	0,0448
$k = 100$	1,48	2,41	0,0452
$k = 150$	1,53	2,48	0,0461
$k = 200$	1,68	2,55	0,0467
$k = 500$	1,76	2,61	0,0474

Tabela 5.2: Rezultati metode k-najbližjih uporabnikov glede na k z uporabo Jaccardovega koeficienta in funkcije podobnosti po meri.



Slika 5.1: Graf MRR v odvisnosti od števila k-najbližjih uporabnikov pri metodah 1 in 2.

Iz slike 5.1 je vidno, da se MRR izboljšuje s povečevanjem števila uporabnikov. Metoda 2, okrepljena z vsebinskimi podatki o uporabnikih, je rahlo boljša in v povprečju testne primere napove na 21. mesto, medtem ko jih metoda 1 napove na 24. mesto. Rezultat pri $k = 500$ pokaže, da se MRR izboljšuje s povečevanjem števila uporabnikov.

5.2.2 k-najbližjih artiklov

Podobno kot v prejšnjem razdelku smo tudi tu ločili dve metodi glede na način izračuna podobnosti artiklov:

- **metoda 1**: Jaccardov koeficient podobnosti,
- **metoda 2**: utežen izračun podobnosti z Jaccardovim koeficientom in funkcijo podobnosti po meri.

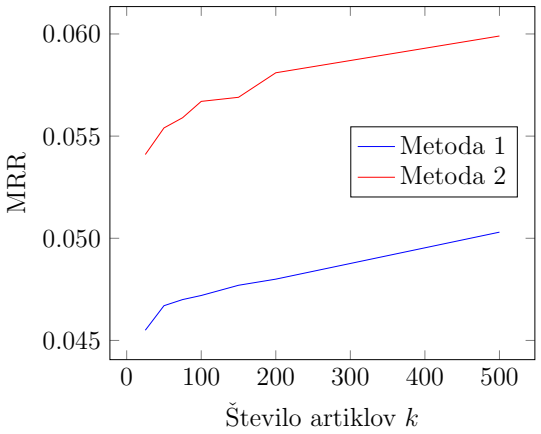
Za število najbližjih artiklov k smo izbrali različne vrednosti z intervala med 25 in 500. Rezultati metode 1 so prikazani v tabeli 5.3, metode 2 v tabeli 5.4, slika 5.2 pa prikazuje graf MRR v odvisnosti od števila najbližjih artiklov k pri obeh metodah.

št. artiklov	rec@5 (%)	rec@10 (%)	MRR
$k = 25$	2,02	3,12	0,0455
$k = 50$	2,11	3,14	0,0467
$k = 75$	2,25	3,24	0,0470
$k = 100$	2,37	3,39	0,0472
$k = 150$	2,49	3,41	0,0477
$k = 200$	2,58	3,65	0,0480
$k = 500$	2,69	3,76	0,0503

Tabela 5.3: Rezultati metode k-najbližjih artiklov glede na k z uporabo Jaccardove mere podobnosti.

št. artiklov	rec@5 (%)	rec@10 (%)	MRR
$k = 25$	2,71	3,60	0,0541
$k = 50$	2,82	3,69	0,0554
$k = 75$	2,93	3,77	0,0559
$k = 100$	2,95	3,86	0,0567
$k = 150$	2,99	3,96	0,0569
$k = 200$	3,12	4,14	0,0581
$k = 500$	3,21	4,28	0,0599

Tabela 5.4: Rezultati metode k-najbližjih artiklov glede na k z uporabo funkcije podobnosti po meri.



Slika 5.2: Graf MRR v odvisnosti od števila k-najbližjih artiklov pri metodah 1 in 2.

Pričakovano se tudi pri tej metodi MRR izboljšuje s povečevanjem števila najbližjih artiklov k. Metoda 2 je dala boljše rezultate kot metoda 1 in v povprečju testne primere napovedovala na 17. mesto, kar je za tri mesta bolje od metode 1. Tako kot pri metodi k-najbližjih uporabnikov tudi tu rezultat pri $k = 500$ pokaže trend izboljševanja.

5.3 Priporočanje z matričnim razcepom

Ogledali si bomo vpliv treh najpomembnejših parametrov matričnega razcepa na rezultate ocen uspešnosti. To so regularizacija λ, faktor povečanja zaupanja α in število faktorjev f. Za testiranje na validacijski množici podatkov smo uporabili naslednje vrednosti parametrov:

- $\alpha = 10, 30, 50, 70, 90,$
- $\lambda = 1, 4, 16, 64, 256, 1024, 2048$ in
- f = 10, 30, 50, 70, 90, 110, 130.

Rezultati najboljših 10 kombinacij zgornjih parametrov so prikazani v tabeli 5.5.

α	λ	f	rec@5 (%)	rec@10 (%)	MRR
50	256	110	2,61	6,91	0,0653
30	256	130	2,55	6,81	0,0650
70	256	110	2,54	6,73	0,0641
30	256	110	2,52	6,58	0,0633
90	256	110	2,52	6,49	0,0628
50	256	90	2,49	5,41	0,0619
30	64	130	2,48	5,29	0,0602
30	256	90	2,48	5,09	0,0592
50	256	70	2,46	4,99	0,0586

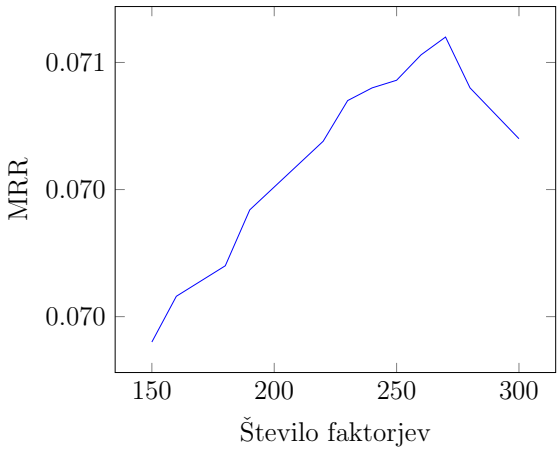
Tabela 5.5: Rezultati najboljših 10 kombinacij parametrov matričnega razcepa.

Najboljši rezultat matričnega razcepa nam je dala kombinacija $\alpha = 50, \lambda = 256$ in $f = 110$ s priklicem prvih 5 priporočil 2,61% in prvih 10 priporočil 6,91%. Metoda je v povprečju napovedovala skriti testni primer na 15. mesto. V tabeli 5.5 opazimo, da se je vrednost MRR povečevala s številom faktorjev f in da je najboljše rezultate dosegala pri $\lambda = 256$. Parameter α ni imel tako visokega vpliva, saj lahko opazimo, da njegove vrednosti izrazito skačejo.

Test smo ponovili še z večjimi vrednostmi parametra f, da bi potrdili trend povečevanja vrednosti MRR. Parametra α in λ smo fiksirali na vrednosti $\alpha = 50$ in $\lambda = 256$. Rezultati so prikazani v tabeli 5.6 in na sliki 5.3.

f	rec@5 (%)	rec@10 (%)	MRR
300	6,72	9,33	0,0707
270	6,98	9,52	0,0711
240	6,81	9,42	0,0709
210	6,72	9,37	0,0706
180	6,59	9,29	0,0702
150	6,48	9,20	0,0699

Tabela 5.6: Rezultati matričnega razcepa glede na število faktorjev.



Slika 5.3: Graf MRR v odvisnosti od števila faktorjev pri matričnem razcepu.

Na sliki 5.3 vidimo, da MRR s številom faktorjev narašča do 270 faktorjev. Vrednosti naprej pa že kažejo znake prekomernega prilagajanja podatkom (angl. *overfitting*). Idealno število faktorjev je tako okoli 270. Vrednost MRR izračunana pri $f = 270$ znaša 0,0711, kar pomeni, da metoda v povprečju testne primere napove na 14. mesto. Postopek postane pri velikem številu faktorjev zelo časovno zahteven. V produkcijski aplikaciji bi za parameter f vzeli vrednost 150, saj računanje s to vrednostjo še ni tako časovno zahtevno in prinaša dobre rezultate, saj tudi ta vrednost v povprečju testne primere napove na 14. mesto.

5.4 Primerjava rezultatov implementiranih metod

Metoda	rec@5 (%)	rec@10 (%)	MRR
Najbolj priljubljeni	0,52	0,60	0,0232
k-najbližjih uporabnikov	1,76	2,61	0,0474
k-najbližjih artiklov	3,21	4,28	0,0599
Matrični razcep	6,98	9,52	0,0711

Tabela 5.7: Rezultati implementiranih metod.

Najboljši rezultati posameznih metod so prikazani v tabeli 5.7. Pričakovano je najslabše rezultate prinesla naivna metoda priporočanja najbolj priljubljenih artiklov.

Priporočanje z metodo k-najbližjih uporabnikov nam je ponudilo nekoliko slabše rezultate kot metoda k-najbližjih artiklov. Kot smo že opisali v razdelku 3.4, smo iz podatkov izločili uporabnike, ki imajo manj kot pet nakupov in artikle, ki so bili kupljeni manj kot desetkrat. Na podlagi tega lahko sklepamo, da so podatki o uporabnikih bolj redki kot podatki o artiklih in metoda pri iskanju ne najde tako *dobrih* sosedov kot pri artiklih.

Matrični razcep ima najboljše rezultate. Priporočanje na podlagi skritih faktorjev je bolj robustno kot priporočanje na podlagi podobnih uporabnikov

ali artiklov. Faktorji lahko nosijo informacijo, ki se je ne da pridobiti s podobnostjo, na primer široke hlače ali pulover iz mehkega elastičnega bombaža. Sklepamo, da je do tega prišlo tudi v naši podatkovni zbirki, saj metoda napoveduje skrite testne primere kar za tri mesta bolje od drugouvrščene metode k-najbližjih artiklov.

V tabeli 5.8 so prikazani časi, ki so ga metode porabile za učenje modela in priporočanje.

Metoda	Čas učenja	Čas priporočanja za enega uporabnika
Najbolj priljubljeni	/	1 ms
k-najbližjih uporabnikov	/	30 s
k-najbližjih artiklov	/	30 s
Matrični razcep	30 min	30 ms

Tabela 5.8: Približni časi učenja in priporočanja metod.

V razdelku 2.2.2 smo omenili, da algoritem k-najbližjih sosedov spada med algoritme na podlagi pomnjenja. To pomeni, da si vse informacije o uporabnikih in artiklih zapomni in do njih dostopa, ko je to potrebno. Posledično pri njem ne moremo govoriti o času, ki ga porabi za učenje. Na drugi strani pa je čas, potreben za priporočanje, toliko večji, saj mora algoritem poiskati najbližje sosede v celotni podatkovni zbirki. To operacijo lahko nekoliko pohitrimo z uporabo podatkovnih struktur kot sta kroglasto drevo (angl. *ball tree*) in k-d drevo (angl. *k-d tree*), vendar na račun aproksimacije podatkovni strukturi ne zagotavljata optimalnega rezultata [33]. Uporaba takšnih podatkovnih struktur je zelo pomembna v produkciji. V primeru stalnega prihajanja novih podatkov so algoritmi na podlagi pomnjenja primerna izbira, saj jih je enostavno posodabljati. Nasprotno matrični razcep uporablja model, ki ga je ob prihodu novih podatkov potrebno znova naučiti popolnoma od začetka, kar je lahko hitro časovno zelo potratno. Vendar je potem čas za priporočanje toliko manjši, tako da velja premisliti, kaj je v dani aplikaciji pomembno in predvsem dopustno.

Same rezultate smo poskušali ovrednotiti tudi kvalitativno. V tabeli 5.9 sta prikazana seznama kupljenih in priporočenih artiklov dveh kupcev, pri katerih se je napovedani testni artikel nahajal na prvem mestu. Seznam priporočenih artiklov zajema prvih pet priporočil. Blagovne znamke pri posameznih artiklih so zaradi zasebnosti podatkov izpuščene.

ID kupca	Kupljeni artikli	Priporočeni artikli
C894871	kratke hlače m 2x kratka majica ž 2x dolge hlače ž 2x dolga majica ž perilo ž kopalke ž nakit	kratka majica ž dolga majica ž kratke hlače m kratka majica ž dolge hlače ž
C179385	jeans m perilo m 3x nogavice m obleka m kratka majica m dolga majica m trenirka m	perilo m perilo m kratka majica m kratka majica m jakna m

Tabela 5.9: Seznam kupljenih in priporočenih artiklov dveh kupcev.

Kot vidimo bi za dobro kvalitativno vrednotenje morali poznati izgled artiklov, saj so imena artiklov preveč generična. Še bolje pa bi bilo pridobiti mnenje o priporočilih s strani teh dveh kupcev.

Poglavje 6

Zaključek

V diplomskem delu smo implementirali najbolj znane metode za izgradnjo priporočilnega sistema trgovine s tekstilnimi izdelki z uporabo podatkov iz realnega sveta. V prvem delu smo predstavili teoretično ozadje priporočilnih sistemov, kjer smo opisali problem priporočanja, vrste priporočilnih sistemov, metode za izgradnjo priporočilnih sistemov, vrednotenje in glavne omejitve ter izzive, ki jih prinašajo takšni sistemi. Predstavili smo tudi povezovalna pravila in algoritem Apriori, ki se uporablja za iskanje takšnih pravil. V nadaljevanju smo prešli na praktični del diplome, kjer smo opisali statistično analizo podatkov in njihovo transformacijo v ustrezno obliko za priporočilni sistem. Sledil je opis implementiranih metod in rezultati skupaj z analizo.

V diplomskem delu smo ugotovili, da je izgradnja priporočilnega sistema, ki na vhod prejme implicitne podatke, težak problem, ki v akademskem svetu še ni tako raziskan kot problem priporočanja na podlagi eksplicitnih ocen. Implementacijo smo tako prilagajali na vseh korakih, tako pri meri za podobnost uporabnikov in artiklov kot tudi pri načinu vrednotenja in izboru ocen uspešnosti.

Možnih izboljšav je kar nekaj. V prvi vrsti bi priporočanje močno izboljšali s poznavanjem uporabnikovih interesov in preferenc. Te informacije bi lahko pridobili z vprašalniki, ki zajemajo ocenjevanje vzorčnih artiklov. Tako bi za posameznega uporabnika izvedeli za primere artiklov, ki mu niso

všeč, kar bi bila zelo dobra informacija pri učenju modela. Seveda trgovini predstavlja možno rešitev tudi postavitev spletne trgovine, ki omogoča lažje sledenje uporabniku preko njegove zgodovine brskanja artiklov. Priporočanje na podlagi vsebine bi zagotovo izboljšali tudi z dodatnimi atributi o artiklih, na primer s podatkom o barvi in materialu artikla.

Naši rezultati so vzpodbudni, saj smo z metodami k-najbližjih sosedov in matričnim razcepom dosegli precej boljša priporočila kot pri naivni metodi priporočanja najbolj priljubljenih artiklov. S funkcijama podobnosti uporabnikov in artiklov, ki smo ju napisali glede na razpoložljive podatke, smo izboljšali delovanje metod k-najbližjih sosedov, ki za izračun podobnosti uporabljata le Jaccardovo razdaljo. Metodo matričnega razcepa bi zaradi kratkega časa učenja modela, hitrega časa priporočanja in vzpodbudnih rezultatov lahko uporabili v produkcijski aplikaciji.

Literatura

[1] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *Proceedings of ACM SIGIR workshop on recommender systems: algorithms and evaluation*, (California, USA), 1999.

[2] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, "Recommender system application developments: A survey," *Decision Support Systems*, vol. 74, pp. 12–32, 2015.

[3] F. E. Walter, S. Battiston, M. Yildirim, and F. Schweitzer, "Moving recommender systems from on-line commerce to retail stores," *Information Systems and e-Business Management*, vol. 10, no. 3, p. 367–393, 2012.

[4] A. S. Lampropoulos and G. A. Tsihrintzis, *Machine Learning Paradigms: Applications in Recommender Systems*. Springer, 2015.

[5] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," *The Adaptive Web*, pp. 325–341, 2007.

[6] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in Artificial Intelligence*, vol. 4, 2009.

[7] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, "Recommendation systems: Principles, methods and evaluation," *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261–273, 2015.

[8] G. Koutrika, B. Bercovitz, and H. Garcia-Molina, "Flexrecs: expressing and combining flexible recommendations," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, (Rhode Island, USA), pp. 745–758, 2009.

[9] R. Burke, "Hybrid web recommender systems," *The Adaptive Web*, pp. 377–408, 2007.

[10] M. J. Pazzani and D. Billsus, "User modeling for adaptive news access," *User Modeling and User-Adapted Interaction*, vol. 10, no. 2-3, pp. 147–180, 2000.

[11] C. Basu, H. Hirsh, and W. Cohen, "Recommendation as classification: using social and content-based information in recommendation," in *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, (Wisconsin, USA), pp. 714–720, 1998.

[12] C. C. Aggarwal, *Recommender Systems: The Textbook*. Springer, 2016.

[13] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, "Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system," in *Proceedings of the 1998 ACM Conference on Computer Supported Cooperative Work*, (New York, USA), pp. 345–354, ACM, 1998.

[14] N. Craswell, "Mean Reciprocal Rank," in *Encyclopedia of Database Systems*, pp. 1703–1703, Springer, 2009.

[15] A. M. Rashid, I. Albert, D. Cosley, S. K. Lam, S. M. McNee, J. A. Konstan, and J. Riedl, "Getting to know you: learning new user preferences in recommender systems," in *Proceedings of the 7th international conference on Intelligent user interfaces*, (California, USA), pp. 127–134, 2002.

[16] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. T. Riedl, "Application of dimensionality reduction in recommender system—a case study," *ACM WebKDD Workshop*, 2000.

[17] S. Khusro, Z. Ali, and I. Ullah, "Recommender systems: Issues, challenges, and research opportunities," in *Information Science and Applications (ICISA) 2016*, pp. 1179–1189, Springer Singapore, 2016.

[18] M. A. Ghazanfar and A. Prügel-Bennet, "Leveraging clustering approaches to solve the gray-sheep users problem in recommender systems," *Expert Systems with Applications: An International Journal*, vol. 41, no. 7, pp. 3261–3275, 2014.

[19] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, (Washington, USA), pp. 207–216, 1993.

[20] R. J. Bayardo, "Efficiently mining long patterns from databases," in *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, (Washington, USA), pp. 85–93, 1998.

[21] J. Gu, B. Wang, F. Zhang, W. Wang, and M. Gao, "An improved Apriori algorithm," in *Communications in Computer and Information Science*, pp. 127–133, Springer Berlin Heidelberg, 2011.

[22] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Boston, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[23] "Microsoft Power BI." Available: `https://powerbi.microsoft.com/en-us/`. Accessed: 31 August 2017.

[24] "arules: Mining association rules and frequent itemsets." Available: `https://cran.r-project.org/web/packages/arules/index.html`. Accessed: 31 August 2017.

[25] "CSV helper." Available: https://github.com/JoshClose/CsvHelper. Accessed: 31 August 2017.

[26] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[27] S. Funk, "Netflix update: Try this at home." Available: http://sifter.org/simon/journal/20061211.html. Accessed: 31 August 2017.

[28] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, (Nevada, USA), pp. 426–434, 2008.

[29] G. Takács, I. Pilászy, N. B., and D. Tikk, "Matrix factorization techniques for recommender systems," *SIGKDD Explorations*, vol. 9, pp. 80–84, 2007.

[30] Y. Hu, Y. Koren, and C. Volinsky, "Collaborative filtering for implicit feedback datasets," in *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, (Washington, USA), pp. 263–272, 2008.

[31] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, (Washington, USA), pp. 43–52, 2007.

[32] "MyMediaLite recommender system library." Available: http://www.mymedialite.net/. Accessed: 31 August 2017.

[33] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," in *Proceedings of the 17th International Conference on Neural Information Processing Systems*, NIPS'04, (Cambridge, USA), pp. 513–520, MIT Press, 2004.