VILNIAUS UNIVERSITETAS

AURIMAS RAPEČKA

REKOMENDACINIŲ SISTEMŲ SOCIALINIUOSE TINKLUOSE EFEKTYVUMO DIDINIMAS

Daktaro disertacija

Fiziniai mokslai, informatika (09 P)

Vilnius, 2015

Disertacija rengta 2010–2014 metais Vilniaus universiteto Matematikos ir informatikos institute.

Mokslinis vadovas:
prof. habil. dr. Gintautas Dzemyda (Vilniaus universitetas, fiziniai mokslai, informatika – 09 P).

Padėka

Nuoširdžiai dėkoju darbo vadovui prof. habil. dr. Gintautui Dzemydai už vertingas mokslines konsultacijas, nuoseklų vadovavimą, pagalbą ir neišsenkančią kantrybę rengiant šią disertaciją.

Esu dėkingas disertacijos recenzentams prof. habil. dr. Antanui Žilinskui ir dr. Viktorui Medvedevui, pateikusiems vertingų pastabų ir patarimų bei padejusiems pagerinti šio darbo kokybę.

Dėkoju kolegoms doktorantams Mykolui Bilinskui ir Martynui Sabaliauskui bei kitiems Vilniaus universiteto Matematikos ir informatikos instituto Sistemų analizės bei Atpažinimo procesų skyrių kolektyvų nariams už bendradarbiavimą, pagalbą ir palaikymą, asociacijai INFOBALT už finansinę paramą doktorantūros studijų metu.

Nuoširdžiai dėkoju šeimos nariams, artimiesiems ir draugams už supratingumą, paramą ir kantrybę.

Taip pat dėkoju redaktorėms Jorūnei Rimeisytei ir Janinai Kazlauskaitei bei visiems kitiems, kurie tiesiogiai ar netiesiogiai prisidėjo prie šio darbo rengimo.

Aurimas Rapečka

Santrauka

Šiais laikais vis daugiau produktų ir paslaugų perkeliama į virtualią aplinką. Interneto socialiniai tinklai ir įvairios elektroninės parduotuvės šiuo metu išgyvena didelį pakilimą. Įsigydamas prekę ar pasinaudodamas paslauga internetu klientas gali taupyti savo laiką. Tačiau čia kyla teisingo siūlymo ir pasirinkimo problemos. Šioms problemoms spręsti plačiai taikomos rekomendacinės sistemos.

Disertacijos tikslas – sukurti naują rekomendacijų kūrimo metodą, įvertinantį vartotojų grupių specifiką, kai vartotojų įverčių produktams matrica yra tankiai užpildyta. Dažniausiai šiuose duomenų rinkiniuose yra didelis vartotojų ir santykinai mažas produktų skaičius. Tokio tipo duomenų rinkiniai paplitę specializuotose elektroninėse parduotuvėse, taip pat įvairiuose specifiniuose interneto kataloguose.

Atlikta analitinė rekomendacinių sistemų veikimo principų apžvalga; susistemintos žinios apie rekomendacijoms kurti taikomus metodus ir jų efektyvumą; atliktas eksperimentinis rekomendacinėse sistemose dažniausiai taikomų metodų efektyvumo tyrimas; sukurtas naujas rekomendavimo metodas, tinkamas duomenų rinkiniams, kai vartotojų įverčių produktams matrica yra tankiai užpildyta, ir įvertinantis vartotojų grupių specifiką kuriant rekomendacijas.

Disertaciją sudaro 5 skyriai, literatūros sąrašas ir priedas. Visa disertacijos apimtis yra 113 puslapių su priedais; 31 paveikslas ir 10 lentelių.

Tyrimų rezultatai publikuoti 5 periodiniuose recenzuojamuose moksliniuose ir 3 kituose leidiniuose.

Tyrimų rezultatai pristatyti ir aptarti 9 mokslinėse konferencijose.

Paveikslų sąrašas

Lentelių sąrašas

Žymėjimai

Simboliai

V	Vartotojų įverčių produktams matrica
V_{ij}	i-tojo vartotojo įvertis j-tajam produktui
$\check{V}_{ij}$	Prognozuojamas i-tojo vartotojo įvertis j-tajam produktui
N_{a_i}	Produktų, tinkamiausių vartotojui a_i, sąrašas
Z_t	Reali paklausa laiko momentu t
Z'_t	Prognozuota paklausa laiko momentu t
X_l	l-tojo klasterio centras
a_N	Naujas sistemos vartotojas
P_{lu}^{j}	Tikimybė, kad klasterio C_l vartotojai suteiks produktui b_j įvertį u
$\bar{V}_l^{\,j}$	Vidutinis įvertis, produktui b_j suteiktas visų C_l klasterio vartotojų

Santrumpos

AUC	Įvertis, nurodantis eilės nustatymo kokybę įvertinant plotą po *ROC* kreive.
BF	Bendrasis filtravimas
DB	Duomenų bazė
MAP	Vidutinių preciziškumų vidurkis
NDCG	Normalizuotas diskontuotas suminis naudingumas.
prec@5, *prec@10*	Rekomendacinės sistemos specifiškumas, kai įvertinami tik pirmieji 5 arba 10 rekomendacinės sistemos rekomenduojamų elementų.
recall@5, *recall@10*	Rekomendacinės sistemos jautrumas, kai įvertinami tik pirmieji 5 arba 10 rekomendacinės sistemos rekomenduojamų elementų.
ROC	Kreivė, iliustruojanti klasifikatoriaus, atskiriančio dvi klases, jautrumo ir $(1 - specifiškumo)$ santykį.
RS	Rekomendacinė sistema
MRR	Vidutinis atvirkštinis rangas
MSE	Vidutinė kvadratinė paklaida
NMAE	Vidutinė absoliutinė paklaida
MD	Vidutinis nuokrypis

Turinys

1

Įvadas

1.1 Tyrimų sritis

Tobulėjant technologijoms ir daugėjant aktyvių interneto vartotojų vis daugiau produktų ir paslaugų perkeliama į virtualią aplinką. Neišeidamas iš namų klientas gali internete įsigyti prekę ar pasinaudoti paslauga, taip taupydamas savo laiką. Tačiau čia atsiranda naujų problemų.

Pirmiausia – kaip išsirinkti prekę tada, kai visi siūlomi produktai panašūs, o patirties mažai; ir antra – kaip surasti reikiamą produktą tarp daugelio kitų, visai nereikalingų. Šioms problemoms spręsti plačiai taikomos rekomendacinės sistemos (RS). Didžiajai daliai metodų, taikomų rekomendacijoms kurti, reikalingos vartotojų, produktų ir vartotojų vertinimų produktams aibės [22]. Šios aibės plačiai kaupiamos elektroninėse parduotuvėse ir socialiniuose tinkluose, kur žmonės gali bendrauti tarpusavyje, išsakyti savo nuomonę ir tokiu būdu tiesiogiai ar netiesiogiai vertinti produktus ir paslaugas. Todėl manoma, kad tiek elektroninės parduotuvės, tiek socialiniai tinklai yra tinkamiausios terpės taikyti RS.

1.2 Darbo aktualumas

Yra sukurta daug metodų, taikomų rekomenduoti prekėms ar paslaugoms. Kiekvienas iš jų turi privalumų ir trūkumų. Pavyzdžiui, plačiai paplitusiais universaliais metodais (pavyzdžiui, *k* artimiausių kaimynų) galima pasiekti gerų rezultatų su didžiąja dalimi duomenų rinkinių, tačiau dėl savo veikimo principų (koreliacijos koeficiento apskaičiavimo tarp konkretaus ir kiekvieno kito vartotojo įverčių viso duomenų rinkinio produktams) šie metodai reikalauja didelių skaičiavimo resursų [21]. Jei skaičiavimo resursų ištekliai arba skaičiavimų trukmė ribota, šių metodų rezultatyviai pritaikyti nepavyksta [50]. Ši problema ypač aktuali realaus laiko rekomendacinėms sistemoms, veikiančioms internete.

Praktikoje išskiriami du pagrindiniai rekomendacinių sistemų tipai: rekomenduojančios pagal skaitinius įverčius ir rekomenduojančios pagal „naudojo ar nenaudojo“, tai yra dvejetainį įverčių formatą (0 ir 1). Pastarasis tipas yra populiaresnis, nes duomenims surinkti nereikia tiesioginių vartotojų atsiliepimų (nereikalaujama įvertinti žiūrėtą filmą, suvartotą produktą ir pan.) [64]. Pasaulyje labiausiai paplitę *k* artimiausių kaimynų tipo metodai, kuriais pasiekiami rezultatai esti pakankamai geri su dauguma duomenų rinkinių [47]. Greitesni ir dideliems duomenų rinkiniams galimi taikyti metodai nėra labai universalūs ir geriausius rezultatus pasiekia taikomi konkretaus tipo, užpildymo ar dydžio duomenų rinkiniams. Tai patvirtino ir disertacijos 3 skyriuje pateikti atliktų eksperimentų rezultatai.

Siekiant sumažinti *k* artimiausių kaimynų metodų skaičiavimų apimtis, panašius vartotojus galima suskirstyti į grupes klasterizavimo metodais [24] [48]. Tačiau svarbu nustatyti optimalų klasterių skaičių, kuris gali smarkiai skirtis įvairiuose duomenų rinkiniuose. Kita problema – naujo vartotojo priskyrimas prie tinkamo klasterio.

Vartotojų klasterizavimu ir minėtų problemų sprendimu paremtas ir disertacijos ketvirtojoje dalyje pristatomas autoriaus sukurtas metodas, per

rekomendacijų kūrimo procesą įvertinantis ir vartotojų grupių specifiką, kai vartotojų įverčių produktams matrica yra tankiai užpildyta.

1.3 Darbo tikslas ir uždaviniai

Disertacijos tikslas – sukurti naują rekomendacijų kūrimo metodą, įvertinantį vartotojų grupių specifiką, kai vartotojų įverčių produktams matrica yra tankiai užpildyta.

Šiam tikslui pasiekti sprendžiami tokie uždaviniai:

1. Atlikti analitinę rekomendacinių sistemų veikimo principų apžvalgą.
2. Susisteminti žinias apie metodus rekomendacijoms kurti ir jų efektyvumą.
3. Atlikti eksperimentinį rekomendacinėse sistemose dažniausiai taikomų metodų efektyvumo tyrimą.
4. Sukurti naują rekomendavimo metodą, tinkamą taikyti duomenų rinkiniams, kai vartotojų įverčių produktams matrica yra tankiai užpildyta, ir įvertinantį vartotojų grupių specifiką kuriant rekomendacijas.

1.4 Mokslinis naujumas

Šiame darbe pateiktas ir ištirtas naujas rekomendavimo metodas, įvertinantis vartotojų grupių specifiką, kai vartotojų įverčių produktams matrica yra tankiai užpildyta.

Šis metodas tinkamas taikyti tokiems duomenų rinkiniams, kuriuose yra didelis vartotojų ir santykinai mažas produktų skaičius. Tokio tipo duomenų rinkiniai paplitę specializuotose elektroninėse parduotuvėse, taip pat įvairiuose specifiniuose interneto kataloguose.

1.5 Ginamieji teiginiai

1. Pateiktas rekomendavimo metodas įvertina vartotojų grupių elgsenos specifiką.
2. Vartotojų klasterizavimas pagreitina rekomendacijų generavimo procesą.

3. Nėra universaliai gerų rekomendavimo metodų – jų rezultatus lemia duomenų rinkinio specifika.

4. Pateikto metodo generuojamų rekomendacijų efektyvumą lemia vartotojų įverčių produktams matricos tankis.

1.6 Darbo rezultatų aprobavimas

Tyrimų rezultatai publikuoti 5 periodiniuose recenzuojamuose ir 3 kituose moksliniuose leidiniuose:

Straipsniai recenzuojamuose periodiniuose moksliniuose leidiniuose:

1. **Rapečka, Aurimas**; Marcinkevičius, Virginijus. Knygų paklausos prognozavimo elektroniniame knygyne galimybės. *Informacijos mokslai*, ISSN 1392-1487. Priimta.

2. **Rapečka, Aurimas**; Dzemyda, Gintautas. A new recommendation model for the user clustering-based recommendation system. *Information Technology and Control*, 2015, Vol. 44, No. 1, ISSN 1392-124X, p. 54–63. (Impact Factor 2014: 0,623)

3. Kurasova, Olga; Marcinkevičius, Virginijus; Medvedev, Viktor; **Rapečka, Aurimas**; Stefanovič, Pavel. Strategies for big data clustering. *ICTAI 2014: 2014 IEEE 26th International Conference on Tools with Artificial Intelligence* [Proceedings], 10–12 November, 2014, Limassol, Cyprus., IEEE Computer Society, 2014. ISSN 1082-3409, p. 740–747.

4. **Rapečka, Aurimas**; Marcinkevičius, Virginijus; Dzemyda, Gintautas. Rekomendacinės sistemos algoritmų veikimo elektroninio knygyno duomenų bazėje analizė. *Informacijos mokslai*, 2013, t. 65, ISSN 1392-0561, p. 45–55.

5. Kurasova, Olga; Marcinkevičius, Virginijus; Medvedev, Viktor; **Rapečka, Aurimas**. Duomenų tyrybos sistemos, pagrįstos saityno paslaugomis. *Informacijos mokslai*, 2013, t. 65, ISSN 1392-0561, p. 66–74.

Straipsniai kituose tarptautinių ir respublikinių mokslinių konferencijų leidiniuose:

6. **Rapečka, Aurimas**; Dzemyda, Gintautas. A new recommendation method for the user clustering-based recommendation system. *EMC2015:*

Engineering Management and Competitiveness: 5th International Symposium [Proceedings], June 19–20, 2015, Zrenjanin, Serbia, ISBN 9788676722563. p. 328–332.

7. Kligienė, Stanislava Nerutė; **Rapečka, Aurimas**. Challenges of digital era: potential and pitfalls of social media. ethics and trust in collaborative cross-domains. *COLLA 2011*: *The First International Conference on Advanced Collaborative Networks, Systems and Applications* [Proceedings], June 19–24, 2011, Luxembourg, ISBN 9781612081434. p. 34–39.

8. **Rapečka, Aurimas**; Dzemyda, Gintautas. Rekomendacinių sistemų ir jose naudojamų rekomendavimo algoritmų apžvalga. *XV Kompiuterininkų konferencijos mokslo darbai*. Vilnius, 2011, ISBN 9789986342618. p. 175–185.

Tyrimų rezultatai buvo pristatyti ir aptarti 9 nacionalinėse ir tarptautinėse mokslinėse konferencijose:

1. 15-oji tarptautinė mokslinė kompiuterininkų konferencija. Klaipėda, Lietuva. 2011 m. rugsėjo 22–24 d.

2. 4-oji Lietuvos jaunųjų mokslininkų konferencija „Operacijų tyrimai versle, inžinerijoje ir informacinėse technologijose“. Kaunas, Lietuva. 2011 m. rugsėjo 30 d.

3. 3-oji LMA Jaunųjų mokslininkų konferencija „Fizinių ir technologijos mokslų tarpdalykiniai tyrimai“. Vilnius, Lietuva. 2013 m. vasario 12 d.

4. 16-oji tarptautinė mokslinė kompiuterininkų konferencija. Šiauliai, Lietuva. 2013 m. rugsėjo 19–21 d.

5. 5-asis tarptautinis seminaras „Duomenų analizės metodai programų sistemoms“. Druskininkai, Lietuva. 2013 m. gruodžio 5–7 d.

6. 55-oji Lietuvos matematikų draugijos konferencija. Vilnius, Lietuva. 2014 m. birželio 26–27 d.

7. 6-oji Lietuvos jaunųjų mokslininkų konferencija „Operacijų tyrimas ir taikymas“. Vilnius, Lietuva. 2014 m. rugsėjo 22 d.

8. 6-oji tarptautinė konferencija „Duomenų analizės metodai programų sistemoms“. Druskininkai, Lietuva. 2014 m. gruodžio 4–6 d.

9. 5-oji tarptautinė konferencija „Engineering Management and Competitiveness“. Zrenjanin, Serbija. 2015 m. birželio 19–20 d.

Tyrimų rezultatų pritaikymas projektinėje veikloje:

1. Naujų knygų paklausos prognozavimo trumpuoju laikotarpiu modelio sukūrimas. Taikomųjų mokslinių tyrimų ir eksperimentinės plėtros darbų sutartis Nr. MTS-580000-2707 su UAB „Barzda“ pagal Inovacinį čekį Nr. 43V-347. (2014-11-25 – 2015-02-27).

2. Knygyno Manoknyga.lt prekių rekomendacinės sistemos metodikos sukūrimas. Atlygintinų paslaugų sutartis Nr. APS-580000-61 su UAB „Barzda“ pagal Inovacinį čekį Nr. 31V-247 (2013-01-15 – 2013-04-10).

3. Europos socialinio fondo finansuojamas projektas „Paslaugų interneto technologijų kūrimo ir panaudojimo našių skaičiavimų platformose teoriniai ir inžineriniai aspektai“ (Nr. VP1-3.1-ŠMM-08-K-01-010).

Darbų pripažinimas:

1. Geriausias III LMA Jaunųjų mokslininkų konferencijos „Fizinių ir technologijos mokslų tarpdalykiniai tyrimai“ pranešimas. 2013 m.

2. Asociacijos „INFOBALT” stipendija už mokslo tiriamąjį darbą *Vartotojų klasterizavimo taikymo rekomendacijų kūrimo procese galimybės*. 2013 m.

1.7 Disertacijos struktūra

Disertaciją sudaro 5 skyriai ir literatūros sąrašas. Disertacijos skyriai: Įvadas, Rekomendacinių sistemų ir jose taikomų metodų apžvalga, Populiariausių rekomendavimo metodų eksperimentinis įvertinimas, Siūlomas rekomendavimo metodas ir jo taikymo galimybių tyrimai, Rezultatų apibendrinimas. Papildomai disertacijoje pateikta: paveikslų, lentelių, naudotų žymėjimų ir santrumpų sąrašas bei priedai. Visa disertacijos apimtis yra 113 puslapių, juose pateiktas 31 paveikslas ir 10 lentelių. Disertacijoje remtasi 74 literatūros šaltiniais.

2

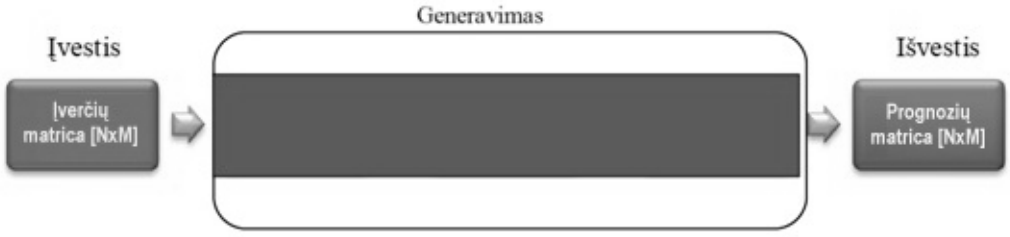
Rekomendacinių sistemų ir jose taikomų metodų apžvalga

Šiame skyriuje atlikta rekomendacinių sistemų, jose taikomų metodų, šių metodų efektyvumą įvertinančių matų, rekomendacinių sistemų efektyvumo tyrimams imamų duomenų rinkinių ir rekomendacinių sistemų realizacijų programinėje įrangoje apžvalga. Pagrindiniai skyriaus rezultatai paskelbti autoriaus mokslinių straipsnių apžvalginėse dalyse.

2.1 Rekomendacinių sistemų struktūra

Kiekviena rekomendacinė sistema (RS) turi du subjektus – vartotoją ir produktą [65]. RS vartotoju laikomas tas subjektas, kuris naudoja šią sistemą, pateikia jai savo nuomonę (nebūtinai sąmoningai) apie įvairius produktus ir gauna naujų produktų rekomendacijų.

RS turi tris pakopas: įvesties, filtravimo (generavimo) ir išvesties (1 paveikslas) [65]. Duomenų į RS įvesties mechanizmas priklauso nuo RS įdiegto informacijos filtravimo mechanizmo ir glaudžiai siejamas su įvairiais duomenų tyrybos metodais. Tinkamų ir reprezentatyvių duomenų išskyrimas iš didelės apimties duomenų aibės yra vienas iš esminių efektyvaus RS veikimo (tikslesnių rekomendacijų) aspektų.



1 paveikslas. *Standartinės RS pakopos*

Nors RS įvesties duomenys gali būti labai įvairūs, tačiau dažniausios yra šios 3 pagrindinės įvesties duomenų kategorijos [45]:

1. Vertinimai, išreiškiantys vartotojo nuomonę apie produktą. Šis duomenų tipas dažniausiai turi skaitinę išraišką, naudojamas ir dvejetainis formatas. Vartotojas šiuos vertinimus palieka sąmoningai (įvertindamas produktus) arba nesąmoningai (palikdamas žymes pirkinių istorijoje ar tinklalapių naršymo istorijoje).

2. Demografiniai duomenys – informacija apie vartotojo amžių, lytį, išsilavinimą ir kt.

3. Turinio duomenys, gaunami iš tekstinių vartotojo dokumentų (profilių) analizės [65].

Informacijos filtravimo pakopoje RS kuria naujų produktų siūlymus arba tikrina šių produktų tinkamumą vartotojui. RS disponuoja vertinimų matrica, o ji pildoma žinomais vartotojų įverčiais produktams. i-tojo vartotojo j-tojo produkto įvertis žymimas V_{ij} ir iš šių įverčių formuojama $m \times n$ dydžio vartotojų įverčių produktams matrica. Paprastai dalis šios matricos yra neužpildyta. Lyginant vartotojų vertinimų produktams skaičių su visu galimu vertinimų skaičiumi gaunama procentinė išraiška, vadinama duomenų rinkinio tankiu (angl. *Density*). Tankis beveik niekada nebūna didelis. Literatūroje aprašomuose moksliniuose tyrimuose tik išskirtiniais atvejais įvertinimų tankis siekia 50 % ir daugiau (*Jester*)[1], o dažniausiai įvertinimų tankis – nuo 1 % iki 5 % ar dar mažesnis [26] [33].

[1] Jester jokes evaluation database (http://www.ieor.berkeley.edu/~goldberg/jester-data/)

Rekomendacijoms kurti taikomos personalizuotos RS skirstomos į dvi dideles grupes [5]: bendrojo filtravimo metodu veikiančios ir turiniu (atributais) paremtos. Savo ruožtu bendrojo filtravimo metodu veikiančios RS skirstomos į dar dvi kategorijas [12]: rekomendacijas kuriančias atminties pagrindu ir modeliais pagrįstas. Specifiniams uždaviniams spręsti taikomos hibridinės RS, sujungiančios bendrojo filtravimo metodu veikiančias ir turiniu paremtas RS [3]. Vartotojo duomenų privatumo atžvilgiu RS skirstomos į personalizuotas ir nepersonalizuotas [57]. Rekomendacijų kūrimo šaltinio atžvilgiu RS praktikoje skirstomos į du didelius tipus [49]: produktų koreliacijos ir vartotojų koreliacijos.

2.1.1 Nepersonalizuotos rekomendacinės sistemos

Nepersonalizuotos rekomendacinės sistemos rekomenduoja produktus vartotojui atsižvelgdamos į bendrą kitų vartotojų nuomonę. Rekomendacijos nepriklauso nuo vartotojo, taigi kiekvienas vartotojas gauna tas pačias rekomendacijas. Šios RS yra automatinės, nereikalauja papildomo vartotojo indėlio, ir trumpalaikės, nes rekomendacijos pateikiamos tik per vieną sesiją ir nekaupiama vartotojo veiksmų istorija. Šios rekomendacinės sistemos yra pakankamai dažnos mažesnėse elektroninėse parduotuvėse, kadangi reikalauja mažiausiai technologinių išteklių [49].

2.1.2 Personalizuotos rekomendacinės sistemos

Personalizuotos RS nuo nepersonalizuotų skiriasi vienu esminiu dalyku – konkretaus vartotojo veiksmų istorijos fiksavimu. Kaip minėta 2.1.1 poskyryje, nepersonalizuotose RS rekomendacijos pateikiamos tik per vieną sesiją ir nekaupiama vartotojo veiksmų istorija. O štai personalizuotos RS turi „atmintį“ – jos duomenų bazėje saugo ir unikalius kiekvieno vartotojo identifikavimo raktus (nuo paprasto slapuko (angl. *Cookie*) iki unikalaus registruoto vartotojo identifikacinio numerio (ID) sistemoje). Tokiu atveju kiekvieną kartą į sistemą sugrįžęs konkretus vartotojas yra identifikuojamas ir rekomendacijos jam pateikiamos atsižvelgiant ne tik į kitų vartotojų vertinimų

vidurkius, bet ir į visą konkretaus vartotojo veiksmų istoriją. Personalizuotos RS taikomos daugelyje aukštesnio lygmens elektroninių parduotuvių ir kitų aplinkų, kuriose vartotojas gali rinktis jam tinkamus produktus.

2.1.3 Produktų koreliacija paremtos rekomendacinės sistemos

Produktų koreliacija paremtos RS rekomenduoja vadinamuosius „suderinamus“ produktus atsižvelgdamos į produktų rinkinius, kuriuos įsigijo konkretūs vartotojai. Pavyzdžiui, jei vartotojas įsidėjo kelis produktus į vartotojo krepšelį, sistema gali rekomenduoti daugiau „suderinamų“ produktų [46].

Ši RS gali būti automatinė, mat įsikišti nereikia nei vartotojui, nei administratoriui, tačiau kartais reikalingas ir neautomatinis grupavimas susijusių produktų deriniams atrinkti. Pastebėtina, kad tokio tipo RS gali būti ir personalizuota, ir nepersonalizuota, kadangi generuojant rekomendacijas nėra būtina žinoti ankstesnę vartotojo veiksmų istoriją. Vis dėlto jei istorija žinoma, rekomendacija gali būti tikslesnė – tas pats prekių rinkinys gali būti nupirktas ir per kelias vartotojo sesijas.

2.1.4 Vartotojų koreliacija paremtos rekomendacinės sistemos

Vartotojų koreliacija paremtos RS rekomenduoja produktus atsižvelgdamas į konkretaus vartotojo elgsenos panašumą su kitais, dalį tų pačių produktų jau pirkusiais vartotojais. Tai dažnai vadinama bendruoju filtravimu ir yra plačiai taikoma rekomendacinėse sistemose [74].

Tokio tipo RS remiasi jau minėta vartotojų įverčių produktams matrica ir veikia vartotojų vertinimų produktams istorijoje ieškodamos panašumų tarp vartotojų pagal jų elgseną. Panašumo tarp vartotojų įverčiai apskaičiuojami pagal Pearsono ar analogišką koreliacijos koeficientą [43]. Plačiau tokių RS metodų veikimas aprašomas 2.4.5 poskyryje.

2.1.5 Turinio specifiką įvertinančios rekomendacinės sistemos

Produkto turinio specifiką (atributus) įvertinančios rekomendacinės sistemos rekomenduoja produktus atsižvelgdamos į specifines produktų ypatybes. Pavyzdžiui, elektroninio knygyno atveju šie atributai galėtų būti: knygų kategorijos, knygų autoriai, knygų leidyklos ir pan. Natūralu, kad RS, užfiksavusi, jog vartotojas ieško knygos iš istorinių knygų kategorijos, jam rekomenduoja populiariausias šios kategorijos knygas.

Deja, tokios RS reikalauja daug rankinio darbo – kiekvienam produktui atributai dažniausiai priskiriami rankiniu būdu.

Atributus įvertinančios RS gali būti tiek personalizuotos, tiek nepersonalizuotos, atsižvelgiant į tai, ar kaupiama konkretaus vartotojo veiksmų istorija ir ar jis atpažįstamas, kai sugrįžta į sistemą.

Personalizuotos RS remiasi produktų, kurie gali būti rekomenduojami, charakteristikomis ir kita informacija, tai yra RS bando rekomenduoti tuos produktus, kurie yra panašūs į vartotojo mėgstamus produktus – kandidatai yra lyginami su anksčiau vartotojo įvertintais pagal ypatybių aibes. Rekomenduojami panašiausi į vartotojo mėgstamus produktai.

Dažniausiai šios RS sukuria produktų profilius, aprašančius produkto ypatybes konkrečios sistemos atžvilgiu. Taip pat sistema sukuria vartotojų profilius, paremtus palankiausiai šių vartotojų vertinamomis produktų ypatybėmis ir šių ypatybių svoriais. Svorių nustatymas yra labai svarbus žingsnis, nes sietema gali tinkamai rekomenduoti produktus. Tam taikomi ne tik paprasčiausi metodai, skaičiuojantys ypatybių vidurkius, bet ir sudėtingi metodai, kaip Bajeso klasifikatoriai, klasterių analizė, sprendimų medžiai ar neuroniniai tinklai, nustatantys tikimybę, kad vartotojas mėgs konkrečius produktus [40].

2.1.6 Hibridinės rekomendacinės sistemos

Pastaruoju metu vis plačiau plinta hibridinės rekomendacinės sistemos, sujungiančios bendrojo filtravimo principus ir turinio ypatybes.

Hibridiniai sprendimai gali būti sukurti keliais būdais:

- prognozuojant tiek bendruoju filtravimu, tiek turiniu paremtais metodais, o rezultatus sujungiant;
- importuojant atributais paremto rekomendavimo galimybes į tradicinius BF metodus;
- sujungiant abu metodus į vieną hibridą.

2006 spalio mėnesį *Netflix*[2] paskelbė didelį rekomendacinių sistemų bandymams tinkamą DVD nuomos ir filmų transliavimo internetu duomenų rinkinį ir inicijavo rekomendacinių sistemų efektyvumo varžybas. Vien pirmaisiais metais šios varžybos sutraukė apie 20000 dalyvių iš 167 valstybių [60]. Konkursas baigėsi 2009 metais. Konkurso laimėtojo sukurta sistema yra puikus hibridinės rekomendacinės sistemos pavyzdys. Ši sistema rekomenduoja lygindama vartotojų matytus filmus ir jų paieškos įpročius (BF metodas) ir rekomenduoja filmus, kurių charakteristikos atitinka geriausiai vartotojo įvertintas (atributais paremtas metodas) [25].

Disertacijoje plačiau hibridinių RS metodų nenagrinėsime, dėmesį sutelksime ties bendrojo filtravimo metodu paremtomis rekomendacinėmis sistemomis.

2.2 Rekomendacinių sistemų pateikiamų rezultatų tipai

Apskritai galima teigti, kad yra du labiausiai paplitę RS scenarijai – generuojantys prognozes ir generuojantys rekomendacijas [65]:

1. Prognozė išreiškiama skaitiniu pavidalu $\check{V}_{ij}$, kuris reiškia prognozuojamą vartotojo a_i įvertinimą produktui b_j. Prognozės dažniausiai

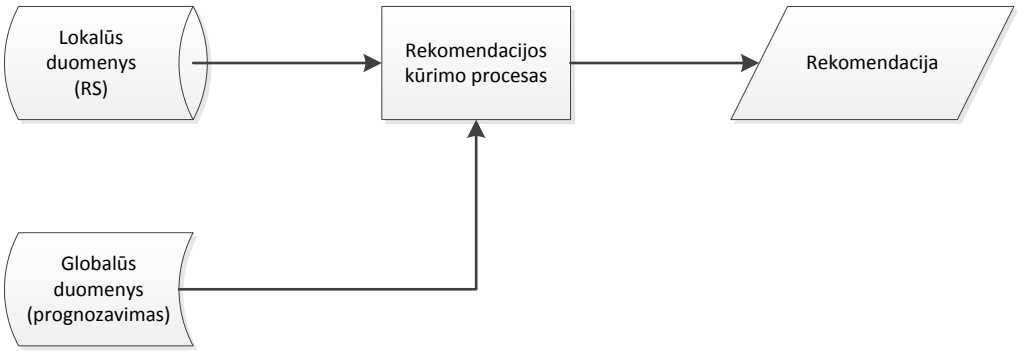
[2] http://www.netflixprize.com

generuojamos tada, kai turimos vartotojų įverčių produktams aibės. Tai – vertinimai, išreiškiantys vartotojo nuomonę apie produktą. Šis duomenų tipas turi skaitinę išraišką ir dažniausiai vartotojas šiuos vertinimus sistemoje palieka sąmoningai, tai yra įvertindamas produktus rankiniu būdu.

2. Rekomendacija išreiškiama kaip produktų, kurie vartotojui a_i turėtų būti tinkamiausi, sąrašas N_{a_i}. Rekomendacijos dažniausiai generuojamos tada, kai turime dvejetainę vartotojo įverčių produktams aibę (0 – nepatinka, 1 – patinka). Dažniausiai šiuos įverčius vartotojas sistemoje palieka nesąmoningai, pavyzdžiui, tiesiog pirkdamas prekę.

2.3 Paklausos prognozavimo metodų ir rekomendacijų tikslumo sąsajos

Apskritai rekomendacijų tikslumas labai priklauso ir nuo platesnio produkto pardavimo konteksto, todėl rekomendacinės sistemos glaudžiai siejamos ir su globaliais produkto paklausos prognozavimo metodais [69]. RS ir globalios paklausos prognozavimo sąveikos schema pateikiama 2 paveiksle.



2 paveikslas. *Rekomendacinės sistemos sąveikos su produktų paklausos prognozavimo metodais schema*

Taigi įvairūs paklausos prognozavimo metodai atlieka pagalbinę funkciją – didina rekomendacijų tikslumą (efektyvumą), todėl toliau yra apžvelgti ir pagrindiniai globalaus produktų paklausos prognozavimo metodai, ir jų santykis su RS.

2.3.1 Prognozavimo metodų tipai

Didžioji dalis įmonių naudoja produktų paklausos prognozavimo būdus, tačiau kiekviena įmonė turi ir savitų metodų, labai priklausančių nuo įmonės konteksto, todėl iš esmės neįmanoma rasti įmonių, kurios naudotų universalius, tinkamus ir kitoms įmonėms, paklausos prognozavimo būdus. Pastaruoju metu sukurta daug inžinerinių sprendimų, leidžiančių prognozuoti paklausą. Daugelis iš jų išbandyta ir praktikoje, nors, tiesa, gauti gana skirtingi bandymų rezultatai [72].

Prognozavimo metodai ne visada tiesiogiai susiję su matematiniais informatikos metodais. Galimi ir ekonominiai ar socialiniai-psichologiniai tyrimai. Kai kuriais atvejais su informatika susiję aspektai (įvairūs kompiuteriniai skaičiavimai) tėra antraeiliai, pavyzdžiui, kai reikia modeliuoti produkto gyvavimo ciklą arba pritaikyti statistinius modelius surinktiems apklausų duomenims.

Šiame poskyryje aptariami tiesiogiai kompiuterine analize paremti paklausos prognozavimo metodai, susiję su laiko eilučių (angl. *Time Series*) taikymu.

2.3.1.1 Laiko eilutės

Laiko eilučių analize paremti metodai tiria paklausos istorinius duomenis remdamiesi iš esmės vien tik laikiniu paklausos pasiskirstymu, dažnai eliminuodami net geografinius veiksnius [62]. Kitaip tariant, jei galima susidaryti ankstesnio laikotarpio paklausos kitimo modelį, vadinasi, tokį modelį dažnai galima pritaikyti ir būsimai paklausai prognozuoti.

Apskritai laiko eilutės yra proceso stebėjimų tam tikru laikotarpiu rinkinys [16]. Laiko eilutės sąvoka atspindi tai, kad duomenys yra interpretuojami kaip kryptinga seka $Z_t, t = 0,1,2, \ldots$, kurioje t indeksas yra diskretaus laiko kintamasis – laiko momentas. Laiko eilutėmis paremti metodai prognozuoja paklausą Z'_t laiko momentu t pagal ankstesniu laikotarpiu sudarytą modelį. Prognozavimo paklaida e_t apskaičiuojama įvertinus skirtumą tarp realios paklausos Z_t momentu t ir prognozuotos paklausos Z'_t :

$$e_t = Z_t - Z'_t. \quad (2.1)$$

Teigiamas skirtumas rodo, kad prognozė buvo per maža, o neigiamas – kad prognozė buvo per didelė.

Pačia paprasčiausia laiko eilute gali būti laikomi stacionarūs duomenys. Stacionarūs duomenys – duomenys, išlaikantys mažai kintamą lygmenį, kurio kitimas priklauso tik nuo aplinkos triukšmo. Matematiškai tai gali būti išreikšta (2.2) formule:

$$Z_t = L + n_t, t = 0,1,2, \ldots, \quad (2.2)$$

čia: Z_t – duomenys, L – nekintamas duomenų lygmuo, n_t – triukšmo lygmuo.

Deja, realiame gyvenime taip palankiai kintamų arba visai nekintamų duomenų pasitaiko labai retai. Dažniausiai duomenys varijuoja didėjimo ir mažėjimo kryptimis, taip pat nuolat kinta triukšmo lygmuo, todėl prognozuoti paklausą nėra labai paprasta. Sudėtingiausias mokslinėje literatūroje aptariamas modelis vadinamas ciklu [56]. Ciklas – eilučių tendencija, kuri kinta ne tik trumpesniu laikotarpiu (pavyzdžiui, per mėnesį), bet ir kartojasi ilgesniais laiko tarpais (pavyzdžiui, kiekvienais metais).

Plačiausiai paplitę modeliai apima lokalias tendencijas, sezoniškumą ir triukšmą. Įprasti laiko eilučių analizės metodai stengiasi išskirti šiuos komponentus ir kiekvieno poveikį įvertinti atskirai.

2.3.1.2 Prognozės tikslumas ir klaidų įvertinimas

Įprastai manoma, kad prognozė yra tiksli tuomet, kai prognozavimo klaida yra santykinai maža. Nustatant prognozavimo tikslumą svarbu suprasti skirtumą ir tarp prognozavimo paklaidos ir triukšmo. Triukšmas dažniausiai yra visiškai nekontroliuojamas ir gali daryti didelę įtaką prognozių paklaidai.

Žinant Z_t ir Z'_t galime apskaičiuoti (2.1) formule išreikštą paklaidą, kuri tinka įvertinti konkrečios prognozės tikslumui. Iš esmės visi paklaidos įverčiai yra vienoks ar kitoks (2.1) formule apskaičiuotų paklaidų vidurkis. Kai žinoma daugiau (n) eilutės stebėjimų ir tų stebėjimų prognozių, dažniausiai taikomi šie paklaidos įverčiai:

1. Vidutinis nuokrypis (angl. *Mean Deviation*) [14]. Tai – paprastas aritmetinis visų prognozavimo paklaidų vidurkis (2.4). Pastebėtina, kad taikant šį įvertį didelės teigiamos ir didelės neigiamos paklaidos gali eliminuoti vienos kitas:

$$MD = \frac{1}{n}\sum e_t. \tag{2.4}$$

2. Absoliutinis vidutinis nuokrypis (angl. *Mean Absolute Deviation*) [36]. Šis įvertis rodo vidutinį paklaidos dydį, neatsižvelgiant į tai, ar paklaida buvo teigiama ar neigiama:

$$MAD = \frac{1}{n}\sum|e_t|. \tag{2.5}$$

3. Vidutinė kvadratinė paklaida (angl. *Mean Squared Error*) (2.6) [66]. Šis įvertis taip pat eliminuoja teigiamų ir neigiamų paklaidų išsilyginimo problemą. Tai – statistiškai teisingas prognozavimo paklaidų įvertis. Bandant įvairius metodus, dažniausiai ieškomas tas, kurį taikant rezultatų vidutinė kvadratinė paklaida gaunama mažiausia.

$$MSE = \frac{1}{n}\sum e_t^2. \tag{2.6}$$

4. Vidutinės kvadratinės paklaidos šaknis (angl. *Root Mean Squared Error*):

$$RMSE = \sqrt{MSE}. \tag{2.7}$$

5. Vidutinė paklaida (angl. *Mean Error*) [9]:

$$MAE = \frac{1}{n}\sum\frac{e_t}{Z_t}. \tag{2.8}$$

6. Vidutinė absoliutinė paklaida (angl. *Mean Absolute Error*) (2.9) [9]. Absoliutinė paklaida yra dažniausiai prognozavimo tikslumui vertinti taikomas įvertis.

$$NMAE = \frac{1}{n}\sum\frac{|e_t|}{Z_t}. \tag{2.9}$$

Aukščiau aptarti prognozės tikslumo ir klaidų įvertinimo matai taikomi ir produktų įverčius prognozuojančių rekomendacijų sistemų tikslumui įvertinti, kai skaičiuojamos paklaidos tarp rekomendacinės sistemos prognozuojamų ir validavimo aibėje esančių vartotojų įverčių produktams.

2.3.2 Pagrindiniai laiko eilučių analizės metodai

Šiame poskyryje apžvelgiami plačiausiai taikomi laiko eilučių analizės metodai. Rezultatai, gauti pritaikius laiko eilučių metodus globaliai produktų paklausai prognozuoti, gali būti naudojami ir rekomendacijų tikslumui didinti.

2.3.2.1 Bendrojo vidurkio metodas

Bendrojo vidurkio (angl. *Cumulative Mean*) metodas dažniausiai taikomas mažai kintamiems duomenims. Tarkime, kad turime stacionarius duomenis, apibrėžiamus šia formule:

$$Z_t = L + n_t. \tag{2.10}$$

Kadangi prognozuojame neįvertindami triukšmo, prognozės išraiška yra tokia:

$$Z'_{t+1} = L. \tag{2.11}$$

L gauname pasinaudodami turimais istoriniais duomenimis ir išvesdami jų vidurkį stebimu laikotarpiu:

$$Z'_{t+1} = \frac{1}{n}\sum Z_t. \tag{2.12}$$

Šis metodas nėra tikslus, kadangi neįvertina konkrečių paklausos šuolių. Kita vertus, šis metodas laikomas beveik idealiu labai ilgų laikotarpių prognozėms, nes vedant ilgųjų laikotarpių duomenų vidurkį eliminuojamas triukšmas ir prognozavimo tikslumas priklauso iš esmės vien tik nuo paklausos tendencijų taisyklingumo [62].

2.3.2.2 Paprastojo prognozavimo metodas

Kai galima nesunkiai išskirti vienodus paklausos intervalus, taikomas paprastojo prognozavimo (angl. *Naive Forecast*) [68] metodas. Prognozuojant šiuo metodu manoma, kad:

$$Z_t = Z_{t-1} + n_t. \tag{2.13}$$

Tokiu atveju prognozė išreiškiama kaip:

$$Z'_{t+1} = Z_t. \tag{2.14}$$

Taigi šiuo atveju manoma, kad būsimo laikotarpio prognozė bus labai panaši į ankstesnio laikotarpio faktinę situaciją. Šis prognozavimo metodas

kartais taikomas labai trumpų terminų prognozėms ir kai neturima daugiau informacijos apie galimą paklausos kaitą. Be to, tai yra iš esmės geriausias prognozavimo metodas tada, kai paklausos kitimas yra atsitiktinis dydis.

2.3.2.3 Slenkamojo vidurkio metodas

Kartais paklausa esti maždaug vienodo lygmens, tačiau ilgesniu laikotarpiu pastebimas didesnis ar mažesnis paklausos pokytis. Prognozuoti tokiems atvejams taikomas slenkamojo vidurkio (angl. *Simple Moving Average*) metodas [8].

Šis metodas įvertina paklausos vidurkį tam tikru laikotarpiu ir pagal jį apskaičiuoja prognozuojamą paklausą:

$$Z'_{t+1} = \frac{1}{M}\sum_{i=t+M-1}^{t} Z_i. \quad (2.15)$$

Sudėtingiausia čia – parinkti laiko momentų skaičių M, kuris nurodo tam tikrą laikotarpį. Nuo to labai priklauso prognozavimo tikslumas. Praktikoje M dažniausiai parenkamas bandymų būdu ir stebint paklausos prognozavimo paklaidų tendencijas.

2.3.2.4 Laiko eilučių regresija

Vienas iš būdų dirbti su tendencingais paklausos duomenimis, yra šių duomenų aprašymas tiesine regresija (2.16):

$$Z'_t = Tt + I, \quad (2.16)$$

čia: I ir T yra tiesinės regresijos modelio koeficientai. Šie du koeficientai parenkami taip, kad prognozės Z'_t vidutinė kvadratinė paklaida (*MSE*) būtų mažiausia. Prabėgus k laiko momentų, prognozė išreiškiama (2.17):

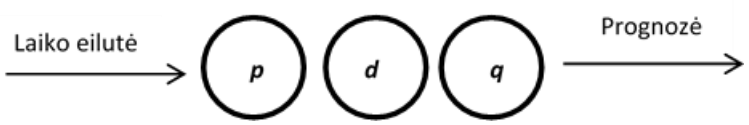
$$Z'_{t+k} = T(t+k) + I. \quad (2.17)$$

Šis metodas gali būti taikomas keliems laikotarpiams; be to, regresija gali būti perskaičiuojama, jei atsiranda naujų duomenų.

2.3.2.5 Autoregresinis integruotas slenkamųjų vidurkių metodas

ARIMA (angl. *AutoRegressive Integrated Moving Average*) – tai autoregresinis integruotas slenkamųjų vidurkių metodas, kuris dažnai taikomas laiko eilučių analizei [6] [67]. Jo esmė yra ta, kad reikia sujungti

autoregresijos, diferencijavimo ir slenkamųjų vidurkių metodo galimybes (3 paveikslas). Visos sudėtinės dalys yra paremtos atsitiktinio triukšmo (nepaaiškinamo išsibarstymo), kuris iškreipia laiko eilutės sisteminę komponentę, koncepcija; be to, metodas turi reakcijos į šį triukšmą aprašymo būdą.



3 paveikslas. *ARIMA modelio veikimo principas*

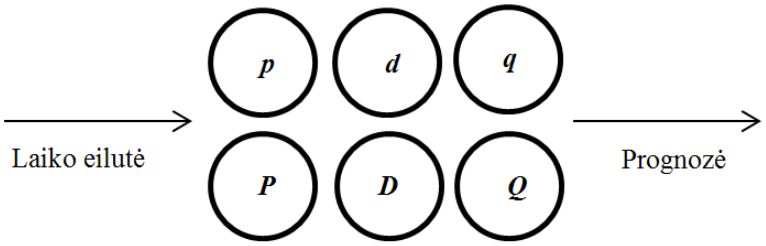
Įprastai, kai apima visas tris dalis, modelis užrašomas $ARIMA(p, d, q)$, kur: p – autoregresijos komponentė, d – diferencijavimo eilė, q – slenkamojo vidurkio narių skaičius.

Remiantis autoregresijos metodu kiekviena laiko eilutės reikšmė yra tiesinė ankstesnės reikšmės ar reikšmių funkcija. Pirmos eilės autoregresinėje lygtyje yra naudojama tik viena ankstesnė reikšmė, antros eilės – dvi ankstesnės reikšmės ir t. t. Laiko eilutės dažnai atspindi, kaip tam tikras procesas nulemia laiko eilutės reikšmių kaitą, bet ne bendrą reikšmių lygmenį. Tokios eilutės vadinamos integruotomis. Žvelgiant iš ilgalaikės perspektyvos jas generuojančio proceso vidurkis gali nesikeisti, tačiau trumpoje atkarpoje eilutės reikšmės gali žymiai nukrypti nuo vidurkio. Integruotos laiko eilutės yra diferencijuojamos, kad būtų išskirti šie informacinę reikšmę turintys pokyčiai ir būtų suvestas eilutę generuojantis procesas į stacionarųjį pavidalą. Pagal slenkamųjų vidurkių metodą kiekviena laiko eilutės reikšmė yra nustatoma išvedant vidurkį iš dabartinės triukšmo reikšmės ir vienos ar kelių ankstesnių triukšmo reikšmių. Slenkamųjų vidurkių metodo eilė nurodo ankstesnių triukšmo reikšmių, pagal kurias yra skaičiuojamas vidurkis, skaičių.

Esant aiškiai išreikštiems sezoniškumo pokyčiams, taikomas išplėstinis *ARIMA* modelis, kuriame įterpiamos ir sezoniškumo komponentės (2.18):

$$ARIMA(p, d, q) \times (P, D, Q), \tag{2.18}$$

čia: P – sezoniškumo autoregresijos komponentė, D – sezoninio diferencijavimo eilė ir Q – sezoninė slenkamojo vidurkio komponentė (4 paveikslas). P, D, Q negali būti didesni už 1, taip pat jų sumos ir santykiai turi kitų apribojimų.



4 paveikslas. *Sezoninio ARIMA metodo veikimo principas*

ARIMA metodas yra gana lankstus ir realizuotas daugelyje duomenų analizės įrankių. Šio metodo sudėtingumo (dėl skaičiavimų resursų ir galimybių) ir efektyvumo santykis yra pakankamai geras ir tai nulemia šio metodo pritaikomumą daugeliui kasdieniškų prognozavimo uždavinių spręsti.

ARIMA metodas gali būti pritaikomas kaip pagalbinis RS rezultatų tikslumą didinantis metodas. Jis gali būti taikomas tuomet, kai duomenų rinkinyje produktų skaičius nėra baigtinis.

2.4 Populiariausi rekomendacinėse sistemose taikomi metodai

2.4.1 Atsitiktinis siūlymas

Atsitiktinis siūlymas (angl. *Random*) – pats paprasčiausias rekomendavimo metodas. Tai atsitiktinis prekės reitingo generavimas. Kiekvienai prekei generuojamas atsitiktinis skaičius $[0;\ 1]$ intervale.

$$Reitingas_{Random} = Atsitiktinis\ skaičius[0;\ 1].$$

Pirmiausia siūlomos tos prekės, kurių sugeneruotas reitingas yra didesnis. Šis metodas taikomas primityviose elektroninės prekybos sistemose ir nėra efektyvus. Tačiau tiriamų duomenų rinkinių rezultatai naudojami kaip atskaitos taškas kitų metodų naudai įrodyti.

2.4.2 Nulinis reitingas

Nulinis reitingas (angl. *Zero*) yra dar vienas atskaitos taškas lyginti metodų efektyvumui su konkrečiais duomenų rinkiniais. Tai – nuolatinio rekomendavimo metodas, visoms prekėms generuojantis nulinį (0) reitingą. Šis reitingas pagrįstas tuo, kad vartotojo įsigytų prekių skaičiaus ir visų siūlomų prekių skaičiaus santykis įprastai artimas nuliui. Todėl ir tikimybė, kad pirkėjas įsigys vieną ar kitą prekę, galima laikyti lygia nuliui.

$$Reitingas_{Zero} = 0.$$

Šiuo atveju tikimybė, kad prekė bus rekomenduota, visoms prekėms yra vienoda.

Šis rekomendavimo metodas gali būti taikomas ir duomenų rinkinių ypatybėms tirti, pavyzdžiui, neatsitiktinių sekų paieškai prekių sąraše.

2.4.3 Populiariausių prekių rekomendavimas

Vienas paprasčiausių rekomendavimo metodų pavyzdžių – populiariausių prekių (angl. *MostPopular*) rekomendavimas visiems pirkėjams [20]. Tai plačiai paplitęs ir dažnai praktikoje taikomas rekomendavimo metodas. Prekės reitingas apskaičiuojamas paprasta formule:

$$Reitingas_{MostPopular} = \frac{Vartotojų, pirkusių\ prekę, skaičius}{Visas\ vartotojų\ skaičius}.$$

Pirmiausia rekomenduojamos tos prekės, kurių reitingas didžiausias.

2.4.4 Populiariausių prekių pagal pasirinktus atributus rekomendavimas

Šiek tiek sudėtingesnis 2.4.3 poskyryje aprašyto populiariausių prekių rekomendavimo metodo atvejis – populiariausių pagal atributus (pavyzdžiui, kategoriją) prekių rekomendavimas (angl. *MostPopularByAttributes*). Šiuo atveju pirmiausia apskaičiuojamas produkto populiarumas:

$$Prekės\ populiarumas = \frac{Vartotojų, pirkusių\ prekę, skaičius}{Visas\ vartotojų\ skaičius}.$$

Tada skaičiuojamas atributo populiarumas vartotojo aplinkoje:

$$Atributo\ populiarumas = \frac{Vartotojo\ pirktų\ prekių\ su\ šiuo\ atributu\ skaičius}{Visų\ vartotojo\ pirktų\ prekių\ skaičius}.$$

Galiausiai apskaičiuojamas produkto reitingas:

$$Reitingas_{MostPopularByAttr} = Prekės\ populiarumas\ \times Atributo\ populiarumas.$$

Šis metodas yra labai konservatyvus ta prasme, kad rekomendacijos neišeina už srities, su kuria vartotojas jau susipažinęs, ribų [34].

2.4.5 k artimiausių kaimynų metodas

k artimiausių kaimynų (angl. *KNN*, *UserKNN*) metodas – įprasčiausias metodas sudėtingesnėse rekomendacinėse sistemose. Reitingai arba prekės prognozuojamos remiantis k panašiausių vartotojų ankstesniais įverčiais. Įprastas *KNN* klasifikatoriaus veikimas paaiškinamas 5 paveiksle.

Vartotojas (skritulys) turi būti priskirtas prie pirmosios klasės (kvadratai) arba prie antrosios klasės (trikampiai). Jei $k = 3$ (ištisinės linijos apskritimas), vartotojas turėtų būti priskirtas prie antrosios klasės, nes į šio vartotojo aplinką patenka 2 trikampiai ir vienas kvadratas. Tačiau tuo atveju, jei $k = 8$ (punktyrinės linijos apskritimas), šis vartotojas priskiriamas prie pirmosios klasės (5 kvadratai ir 3 trikampiai patenka į vartotojo aplinką).

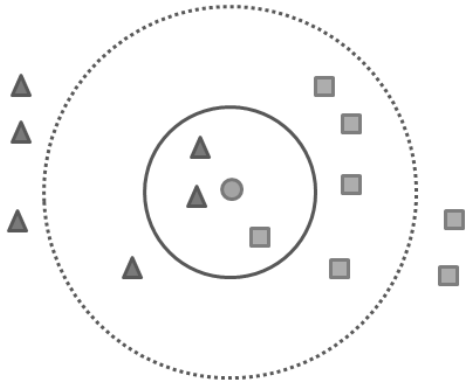
Du esminiai šio metodo, pritaikyto rekomendacinėse sistemose, veikimo aspektai – panašumo koeficiento skaičiavimas ir panašumo slenksčio nustatymas. Populiariausios vartotojo panašumo metrikos yra Pearsono koreliacija (angl. *Pearson Correlation)* ir kosinuso panašumas (angl. *Cosine Similarity*).

Dažniausiai panašumo tarp vartotojų reikšmės apskaičiuojamos pagal Pearsono koreliacijos koeficientą [43]:

$$panašumas(g,r) = \frac{\sum_p (g(p)-\overline{g})(r(p)-\overline{r})}{\sqrt{\sum_p (g(p)-\overline{g})^2 \sum_p (r(p)-\overline{r})^2}}, \quad (2.19)$$

čia g – vartotojas, gaunantis rekomendaciją, r – vartotojas, kurio panašumas tikrinamas. Sumos ženklas formulėje rodo, kad panašumas skaičiuojamas iš visų produktų p, kuriuos abu vartotojai yra įvertinę. $g(p)$ ir $r(p)$ yra vartotojų įverčiai konkrečiam produktui p, o visų vartotojų suteiktų įverčių produktams

vidurkiai – $\bar{g}$ ir $\overline{r}$. Formulės skaitiklis yra vartotojų visų įvertintų produktų įverčių nuokrypio nuo įverčių vidurkio sandaugų suma, o formulės vardiklis normalizuoja gautą rezultatą tam, kad šis priklausytų intervalui $[-1; 1]$. Kuo rezultatas artimesnis 0, tuo vartotojai nepanašesni ir atvirkščiai, kuo arčiau 1, tuo vartotojai panašesni. Jei panašumas lygus -1, turime visiškai skirtingus vartotojus.



5 paveikslas. *KNN klasifikacijos pavyzdys*

Kita panašumo metrika – kosinuso panašumas:

$$panašumas(g,r) = \sum_p \frac{g(p)}{\sqrt{\sum_p g(p)^2}} \times \frac{r(p)}{\sqrt{\sum_p r(p)^2}}. \quad (2.20)$$

Visais tokio tipo metodais skaičiuojamas kiekvieno vartotojo panašumo koeficientas, todėl jei vartotojų ir įverčių skaičius didesnis, reikia didelių skaičiavimo ir laiko resursų.

Kai tyrimuose taikomas BF metodas, išaiškėja keli pagrindiniai jo trūkumai, susiję su generuojamų rekomendacijų tikslumu [65].

1. Tuščių įvertinimų problema. Kaip minėta anksčiau, įverčių tankis vartotojų įverčių produktams matricoje dažniausiai siekia vos kelis procentus, kartais ir dar mažiau. Pagrindinė tai lemianti priežastis – didelis produktų skaičius. Savaime suprantama, kad vienam vartotojui išbandyti ir įvertinti didelį skaičių produktų būtų sudėtinga, daugelis iš jų įvertinę tik kelis, geriausiu atveju – keliolika. Plačiau taikomi du šios problemos sprendimo

būdai: tuščių reikšmių ignoravimas ir šių reikšmių užpildymas (dažniausiai standartiniais statistiniais algoritmais) standartinėmis reikšmėmis [19].

2. Naujų vartotojų problema. Vos užsiregistravęs vartotojas negali suteikti pakankamai informacijos RS, nes jo įvertinimų istorija yra labai trumpa arba iš viso neegzistuoja. Tokiu atveju sunku nustatyti panašius vartotojus ir dėl to rekomendacijos esti labai netikslios. Šios problemos sprendimas dažniausiai paliekamas pačiam vartotojui, registravimosi metu prašoma nurodyti kelis jau išbandytus ir patikusius produktus.

3. Pasitikėjimo rekomendacijomis problema. Didelė dalis vartotojų skeptiškai žiūri į RS teikiamas rekomendacijas, nes nesupranta, kaip jos sukurtos, kokių vartotojų vertinimais pasiremta. Dėl šios priežasties kuriamos sistemos, kurios atrenka keliančius pasitikėjimą vartotojus. Šie metodai grindžiami pasitikėjimu, tačiau dėl savo trūkumų nėra labai populiarūs. Pagal šių metodų koncepciją, vartotojas pats turi pasirinkti kitus vartotojus, kuriais pasitiki ir iš kurių nori gauti rekomendacijas [15]. Kitaip tariant, vos pradėjęs naudoti RS (registruodamasis į elektroninę parduotuvę ar socialinį tinklą), vartotojas turi pažymėti vartotojus, kuriais pasitiki. Tačiau jei tinklas didelis ir turi daug vartotojų, pasitikėjimą keliančių vartotojų dalis – labai nedidelė, todėl rekomendacijos nėra tikslios, o rekomenduojamų produktų apimtis labai maža.

4. Panašių produktų rekomendavimo ir mažos įverčių apimties problema. Ši problema yra keista ir kyla būtent todėl, kad rekomendacijoms generuoti RS naudoja panašių vartotojų informaciją dar ir dėl to, kad vartotojai yra įvertinę per mažai produktų. Pagal panašių vartotojų įvertinimus RS visada rekomenduoja tik panašius produktus tiems, kuriuos vartotojas ar jam panašūs vartotojai yra įvertinę, tokiu būdu mažindama galimą siūlyti asortimentą. Susiduriama su dilema: viena vertus, galima mažinti panašumo tarp vartotojų slenkstį, taip padidinti galimų siūlyti produktų skaičių, tačiau sumažinti rekomendacijų tikslumą. Ir atvirkščiai – jei didinamas panašumo tarp vartotojų slenkstis ir rekomendacijų tikslumas, kardinaliai mažėja rekomenduojamų produktų skaičius. Šiai problemai spręsti dažnai sujungiamos bendrojo

filtravimo metodu paremtos RS su turiniu paremtomis RS, tai yra kuriamos hibridinės RS [3].

2.4.6 Bajeso (standartinis turinio ypatybėmis paremtas) metodas

Šis metodas reikalauja daug skaičiavimo išteklių, tačiau dėl efektyvių skaičiavimo algoritmų taikomas vis plačiau [71]. Pagal Bajeso teoremą, tikimybė, kad produktas d gali priklausyti klasei C_j, apskaičiuojama taip:

$$P(C_j|d) = \frac{P(C_j)P(d|C_j)}{P(d)}, \quad (2.21)$$

čia: $P(C_j|d), P(C_j), P(d|C_j), P(d)$ yra tikimybės, atitinkamai vadinamos: aposteriorine (hipotezės) (angl. *Posterior*); apriorine (angl. *Prior*); įvykio d, esant teisingai hipotezei (angl. *Likelihood*) ir įvykio d (angl. *Evidence*) tikimybėmis. Kadangi produkto ypatybės laikomos nepriklausomomis ir jų yra ne viena $(F_1, \ldots, F_h)$, tai:

$$P(C_j|d) = \frac{P(C_j)\prod_{i=1}^{h} P(F_i|C_j)}{P(F_1,\ldots,F_h)}. \quad (2.22)$$

Antrasis formulės skaitiklio narys gaunamas remiantis produkto ypatybių stebėjimais ir apskaičiuoja suminę kiekvieno produkto ypatybės priklausymo C_j klasei tikimybę. Formulės vardiklis yra visų produkto ypatybių egzistavimo tikimybė. Pagal šią formulę apskaičiuojamos produkto priklausymo kiekvienai klasei tikimybės ir produktas priskiriamas prie tos klasės, kuriai gaunama didžiausia aposteriorinė tikimybė [12]. Tai – standartinis turinio ypatybėmis paremtas metodas [28]. Modifikuotu Bajeso metodu paremti *BPRSLIM* [44] ir *BPRLinear* [9] metodai.

2.4.7 Klasterizavimu paremti rekomendavimo metodai

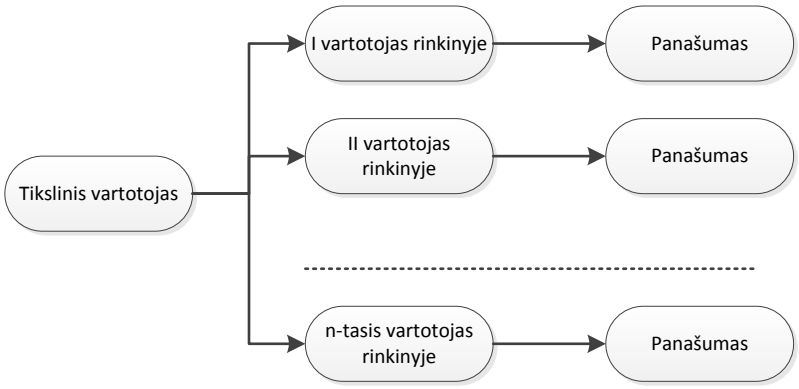
Bendruoju filtravimu paremti rekomendavimo metodai rekomendacijas ir prognozes kuria gana tiksliai. Deja, tokie metodai reikalauja konkretaus vartotojo atliktų įverčių ar veiksmų istorijos lyginimo su kiekvienu iš kitų vartotojų atliktais įverčiais ar jų veiksmų istorijomis. Jei duomenų rinkinys didelis, šie lyginimai reikalauja didelių skaičiavimų resursų ir skaičiavimų

trukmė gali tapti pernelyg ilga. Tai ypač aktualu realaus laiko rekomendacinėms sistemoms, veikiančioms, pavyzdžiui, elektroninėse parduotuvėse ar kitose interneto svetainėse, kuriose RS rezultatus turėtų pateikti per kelias sekundes.

Vienas iš būdų pagreitinti rekomendacijų ar prognozių kūrimą yra duomenų, esančių vartotojų įverčių produktams matricoje $V = \{V_{ij}, i = \overline{1,m}, j = \overline{1,n}\}$, klasterizavimas. Po klasterizavimo panašumų su konkrečiu vartotoju galima ieškoti ne lyginant pastarąjį su visais kitais duomenų rinkinio vartotojais, o lyginant jį su kitų vartotojų grupėmis.

Klasterizavimas – labai populiarus procesas duomenų tyryboje, ypač kai analizuojami didelės apimties duomenys. Pavyzdžiui, jis gali būti taikomas teksto tyryboje. Šiais laikais dideli straipsnių skaičiai saugomi įvairiuose interneto serveriuose, tad susiduriama su didele panašumų tarp dokumentų paieškos problema. Vienas iš šios problemos sprendimo būdų – dokumentų klasterizavimas pagal įvairias jų ypatybes [58].

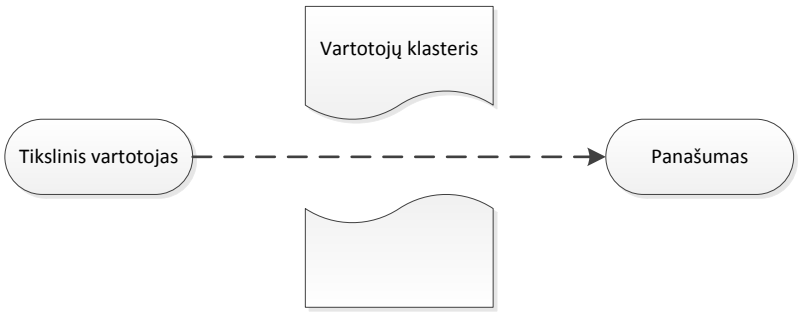
Klasteris – duomenų individų, turinčių panašias ypatybes ar glaudžius ryšius, rinkinys [30]. Klasterizavimas yra tarpinis rekomendacinių sistemų veikimo procesas. Klasterizavimo vaidmuo RS vaizduojamas 6 ir 7 paveiksluose.



6 paveikslas. *RS, netaikančios klasterizavimo, veikimo principas*

Dažniausiai išskiriami keli klasterizavimo rekomendacinėse sistemose būdai. Šis skirstymas nėra visiškai įprastas, tačiau gana tiksliai RS pagal klasterizavimo proceso specifiką yra suskirstytos darbe [70]. Šis skirstymas,

produktų ir vartotojų padalijimas į klasterius įvairiais atvejais, iliustruojamas 8 paveiksle. Punktyrinė linija riboja klasterius, o vartotojų įverčių produktams matricos elementai, padalyti į klasterius, vaizduojami pilka spalva.

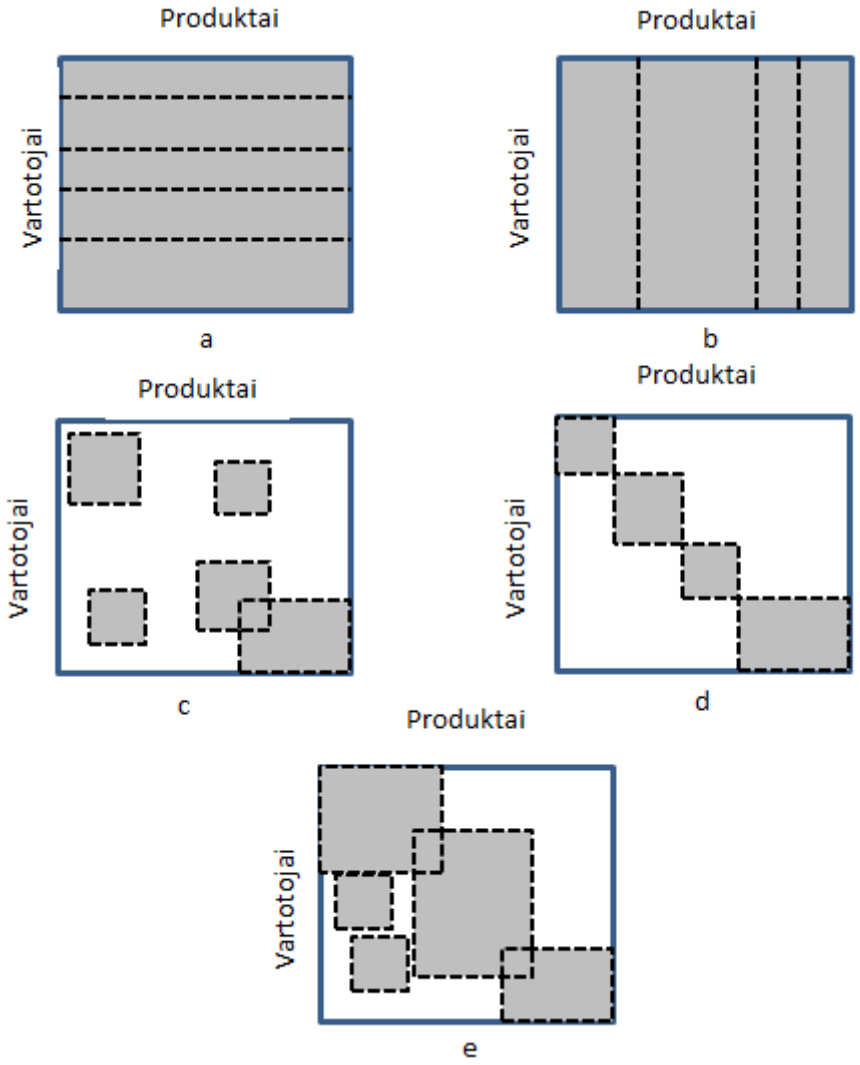


7 paveikslas. *RS, taikančios klasterizavimą, veikimo principas*

Paprasčiausias būdas – suskirstyti vartotojus į klasterius [48]. Kitas kelias – klasterizuoti produktus atsižvelgiant į vartotojų jiems suteiktus įverčius [37]. Trečias būdas – derinti vartotojų ir produktų klasterizavimą [61]. Šie trys metodai apima tik vienokius klasterizavimo būdus, tai yra atskirai klasterizuojami vartotojai arba produktai. Kai kuriuose moksliniuose darbuose taikomas dvipusis klasterizavimas (pavyzdžiui, [7] [18]). Šie metodai vadinami bendruoju klasterizavimu (angl. C*o-Clustering*) paremtu bendruoju filtravimu, mat metodo pagrindas – įprastas bendrasis klasterizavimas. Pagrindinis tikslas – vienu metu išskirti vartotojų ir produktų grupes bendrojo klasterizavimo metodais ir kurti prognozes ar rekomendacijas atsižvelgiant į vidutinius įverčius šiuose bendruose klasteriuose ir į vartotojų specifiką [11]. Šiuo atveju kiekvienas klasteris apima dalį vartotojų ir dalį produktų, visi vartotojai ir produktai priklauso kokiam nors klasteriui ir klasteriai nepersidengia (8 paveikslas, d).

Pagrindinis bendrojo klasterizavimo metodų trūkumas – kiekvienas vartotojas ir produktas gali priklausyti tik vienam klasteriui. Tas pats trūkumas galioja ir paprasto klasterizavimo atveju. Dalis rekomendacinių sistemų galėtų pasiekti geresnių rezultatų, jei būtų galima priskirti vartotojus ir (ar) produktus prie kelių klasterių [1]. Taigi multiklasinis bendrasis klasterizavimas yra

tikslingesnis. Šis būdas leidžia kiekvienam vartotojui ir produktui patekti į kelis klasterius tuo pat metu, tai yra klasteriai gali persidengti (8 paveikslas, e).



8 paveikslas. *Klasterizavimo metodai rekomendacinėse sistemose: a) vartotojų klasterizavimas, b) produktų klasterizavimas, c) dvigubas klasterizavimas, d) bendrasis klasterizavimas e) multiklasinis bendrasis klasterizavimas*

Siauresnis multiklasinio bendrojo klasterizavimo variantas, kai klasteriai niekuomet negali apimti visų vartotojų ir produktų (8 paveikslas, c), išsamiau aptartas [4] ir [29].

2.5 Rekomendacinių sistemų efektyvumą įvertinantys matai

Norint nustatyti, kaip tiksliai rekomendacinė sistema nuspėja, ar produktą vartotojas įsigys (įsigijo), taikomi rekomendacinės sistemos efektyvumo įverčiai. Plačiau nagrinėsime rekomendacines sistemas, paremtas bendro filtravimo (angl. *Collaborative Filtering*) metodu. Taikant bendrojo filtravimo metodą galima rekomenduoti pirkėjui produktą remiantis ankstesnių visų pirkėjų pirkimų istorija. Plačiau šis metodas aprašytas 2.4.5 poskyryje.

Rekomendacinė sistema, taikanti bendrojo filtravimo metodus, veikia kaip klasifikatorius, nurodantis, ar konkreti prekė bus (nebus) pirkėjo pasirinkta. Čia turime dviejų klasių skyrimo problemą (angl. *Binary Classification*). Pirma klasė – pirkėjo įsigyti produktai, o antra klasė – neįsigyti produktai. Klasifikatorių stengiamasi nustatyti taip, kad jis kaip įmanoma geriau (tiksliau) atskirtų pirkėjo įsigytus produktus nuo neįsigytų produktų. Tačiau dažnai klasifikatoriai antros klasės elementus neteisingai priskiria prie pirmos klasės ir atvirkščiai; šias klasifikavimo klaidas stengiamasi įvertinti įvairiais RS efektyvumą nurodančiais įverčiais [54].

1 lentelė. *Klasifikatoriaus vertinimo principas*

		Klasifikatoriaus rezultatai	
		I klasė (rekomenduoti produktai)	II klasė (nerekomenduoti produktai)
Produktų klasės	I klasė (įsigyti produktai)	**TP** (*angl. True Positive*) (teisingai priskirta prie I klasės)	**FN** (*angl. False Negative*) (prie I klasės priskiriami II klasės elementai)
	II klasė (neįsigyti produktai)	**FP** (*angl. False Positive*) (prie II klasės priskiriami I klasės elementai)	**TN** *angl. True Negative*) (teisingai priskirta prie II klasės)

Klasifikatoriaus vertinimo principas pateikiamas 1 lentelėje.

Klasifikatoriui vertinti rekomendacinėse sistemose dažniausiai taikomi įverčiai yra specifiškumas (angl. *Precision*) ir jautrumas (angl. *Sensitivity, Hit Rate Recall*) [63].

2.5.1 Specifiškumas

Specifiškumas yra klasifikatoriaus, atskiriančio dvi klases, klasifikavimo įvertis. Šis įvertis parodo santykį, kiek iš prie pirmos klasės priskirtų elementų iš tiesų priklauso pirmai klasei (kokia dalis iš rekomenduotų produktų buvo įsigyta). Kuo specifiškumas didesnis, tuo daugiau prie pirmos klasės priskirtų elementų iš tikrųjų yra iš pirmos klasės, vadinasi, tuo tiksliau veikia RS.

$$Specifiškumas = \frac{\#(Teisingai\ priskirta\ prie\ pirmos\ klasės)}{\#(\text{iš}\ viso\ priskirta\ prie\ pirmos\ klasės)} = \frac{TP}{TP+FP}. \quad (2.23)$$

$$1 - Specifiškumas = 1 - \frac{TP}{TP+FP} = \frac{FP}{TP+FP}. \quad (2.24)$$

Formulė (2.24) nurodo, kokia rekomenduotų produktų dalis nebuvo įsigyta – rekomenduotų, bet neįsigytų iš visų rekomenduotų produktų dalis.

2.5.2 Jautrumas

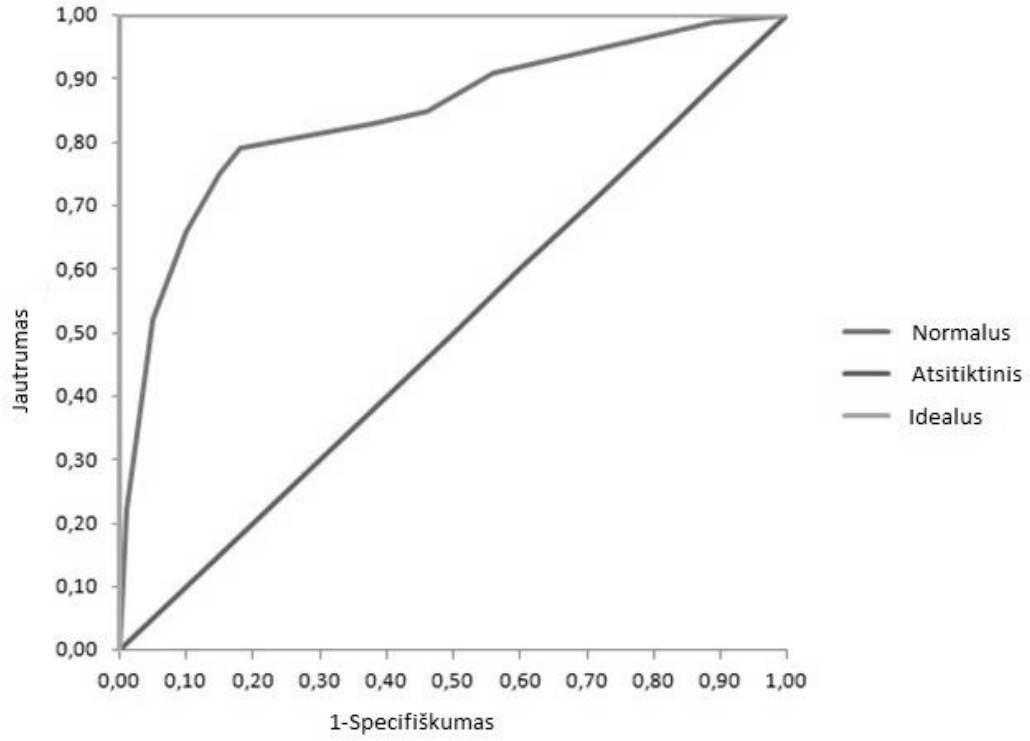
Klasifikatoriaus jautrumas rodo teisingai priskirtų prie pirmos klasės elementų skaičiaus santykį su visu pirmos klasės elementų skaičiumi (kokia dalis produktų buvo rekomenduoti iš visų įsigytų). Mat dažnai prie pirmos klasės priskiriami tie elementai, kuriuos mes norime išskirti, pavyzdžiui, sergantys žmonės. Kuo jautrumas didesnis, tuo daugiau pirmos klasės elementų teisingai priskirta prie pirmos klasės, tuo tiksliau veikia RS.

$$Jautrumas\ = \frac{\#(Teisingai\ priskirta\ prie\ pirmos\ klasės)}{\#(\text{iš}\ viso\ pirmos\ klasės\ elementų)} = \frac{TP}{TP+FN}. \quad (2.25)$$

2.5.3 Plotas po *ROC* kreive (*AUC*)

ROC kreivė – tai grafikas, iliustruojantis klasifikatoriaus, atskiriančio dvi klases, jautrumo ir specifiškumo priklausomybę. *ROC* kreivės pavyzdys pateikiamas 9 paveiksle. Kita *ROC* kreivės interpretacija – tinkamų (rekomenduotų ir įsigytų) produktų dalies priklausomybė nuo netinkamų (rekomenduotų, bet neįsigytų) produktų dalies.

AUC (angl. *Area Under the ROC Curve*) įvertis nurodo rekomendacijos (siūlomo produktų rinkinio) gerumą įvertinant plotą po *ROC* kreive. Didžiausia *AUC* reikšmė yra 1. Geros rekomendacijos atveju $AUC > 0{,}5$, o atsitiktinės rekomendacijos atveju $AUC = 0{,}5$ [17]. *AUC* reikšmė, arba plotas po *ROC* kreive, yra lygus tikimybei, kad klasifikatorius atsitiktinai parinktą pirkėjo nupirktą prekę reitinguos geriau negu atsitiktinai parinktą prekę, kurios neįsigijo.



9 paveikslas. *ROC kreivės pavyzdys*

2.5.4 Specifiškumų vidurkis (*MAP*)

Knygoje [31] vartojama preciziškumo sąvoka, turinti visiškai tokią pačią reikšmę kaip daugelio kitų autorių vartojama specifiškumo sąvoka [52]. Čia įvestas pavadinimas *MAP* (angl. *Mean Average Precission*). Jį vartosime ir toliau.

Tarkime, kad turime M vartotojų. Jei turime j-tojo vartotojo pirmos klasės produktų (įsigytų produktų) aibę $q_j = \{b_1, b_2, \dots, b_{m_j}\}$ ir eilę pirmųjų RS

rekomenduojamų elementų R_{jk}, iki kol įsigytas produktas b_k, tai vidutinis preciziškumas (angl. *Average Precision*) apskaičiuojamas pagal formulę:

$$a_j = \frac{1}{m_j}\sum_{k=1}^{m_j} Specifiškumas(R_{jk}). \quad (2.26)$$

Formulėje (2.26) specifiškumas skaičiuojamas pagal formulę (2.23), kai TP yra j-tajam vartotojui rekomenduotų ir jo įsigytų produktų iš aibės R_{jk} skaičius, FN yra rekomenduotų, bet neįsigytų produktų iš aibės R_{jk} skaičius.

Vidutinis preciziškumas priklauso nuo produkto b_k vietos R_{jk} eilėje.

Vidutinių preciziškumų vidurkis (*MAP*) – klasifikatoriaus jautrumo lygmenų kokybinis įvertis [31]. Jei turime kiekvieno pirkėjo vidutinį preciziškumą, *MAP* galime apskaičiuoti pagal formulę:

$$MAP(Q) = \frac{1}{m}\sum_{j=1}^{m} a_j. \quad (2.27)$$

Čia: Q – vartotojų aibė, m – vartotojų skaičius.

Iš kitų klasifikatoriaus vertinimo įverčių šis išsiskiria stabilumu. Tačiau norint tiksliai įvertinti rekomendacinės sistemos efektyvumą, reikia parinkti didelę ir pakankamai diversifikuotą testavimo aibę.

2.5.5 *recall@k* ir *prec@k* efektyvumo matai

Pirmųjų RS rekomenduojamų produktų sąrašas (tam tikra analogija su R_{jk}) yra naudojamas kurti kitiems rekomendacijų efektyvumo matams [55]. Vienas jų yra *recall@k*. Čia k yra kintamas parametras, nurodantis, kiek produktų yra rekomenduojama vartotojams. Jis susijęs su jautrumo įverčiais (žr. poskyrį 2.5.2). Tarkime, R_i yra rinkinys produktų, rekomenduotų j-tajam vartotojui. $|R_i| = k$. O T_i yra produktų, kuriuos įsigijo šis vartotojas, rinkinys. Tuomet:

$$recall@k = \frac{1}{m}\sum_{i=1}^{m} \frac{|R_i \cap T_i|}{|T_i|}. \quad (2.28)$$

Su specifiškumo įverčiais susijęs matas *prec@k*:

$$prec@k = \frac{1}{m}\sum_{i=1}^{m} \frac{|R_i \cap T_i|}{|R_i|}. \quad (2.29)$$

2.5.6 Normalizuotas diskontuotas suminis naudingumas (*NDCG*)

Normalizuotas diskontuotas suminis naudingumas (*NDCG*, angl. *Normalized Cumulative Discounted Gain*) taikomas paieškos sistemų algoritmų efektyvumui įvertinti ir parodo, kaip naudingai jos vartotojui pateikia tai, ko jis tikisi. Šis metodas taikomas ir rekomendacinėse sistemose [54].

Tarkime, u-tasis vartotojas gauną naudą g_{ui}, kai jam rekomenduojama įsigyti i-tąjį produktą. Vidutinis diskontuotas suminis naudingumas produktų rinkiniui J įvertinamas taip:

$$DCG = \frac{1}{m}\sum_{u=1}^{m}\sum_{j=1}^{J}\frac{g_{ui_j}}{\max(1,\log_b j)}. \quad (2.30)$$

Čia logaritmo b pagrindas paprastai imamas nuo 2 iki 10.

Normalizuotam diskontuotui suminiam naudingumui *NDCG* gauti reikia padalyti diskontuotą sumin naudingumą iš maksimalaus galimo (idealaus) [2]:

$$NDCG = \frac{DCG_p}{DCG^*}. \quad (2.31)$$

2.5.7 Vidutinis atvirkštinis rangas (*MRR*)

Vidutinis atvirkštinis rangas (angl. *Mean Reciprocal Rank, MRR*) skirtas rekomendacijų efektyvumui vertinti, kai žinoma k geriausių (angl. *Top-k*) rekomenduojamų produktų sąrašas, taip pat vartotojo įsigyti produktai. Sąraše produktai išdėstyti pagal svarbą vartotojui (nuo svarbiausio) rekomendacinės sistemos požiūriu.

Produkto rekomendacijos vartotojui atveju galime apskaičiuoti atvirkštinį rangą (angl. *Reciprocal Rank, RR*) vartotojui rekomenduojamų k geriausių produktų sąrašui pagal tai, kelintas sąraše yra produktas, kurį pirmą įsigijo vartotojas, tai yra kuris rekomenduojamų produktų sąrašo produktas yra tinkamiausias vartotojui – mat nebūtinai pirmasis sąrašo produktas yra tinkamiausias. *MRR* yra *RR* vidurkis, skaičiuojamas visiems vartotojams pagal jiems rekomenduojamų produktų individualius sąrašus. *MRR* yra labai svarbus matas vartotojams mažai pateiktų, tačiau reikšmingų rekomendacijų kokybei

vertinti. Tai gali būti draugų rekomendacijos ar rekomendacijos socialiniuose tinkluose, kur tenka apsiriboti 3 ar 5 geriausiais produktais.

Vidutinį atvirkštinį rangą nurodo formulė:

$$MRR = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{rank_i}. \quad (2.33)$$

Čia m – vartotojų skaičius, $rank_i$ – pirmo įsigyto produkto numeris k geriausių rekomenduojamų i-tajam vartotojui produktų sąraše.

Nors visi aprašyti įverčiai skaičiuojami skirtingai, tačiau praktikoje jie yra beveik lygiaverčiai. Todėl siekiant surasti efektyviausiai su konkrečiu duomenų rinkiniu veikiančius rekomendavimo metodus, jų veikimo rezultatus būtina vertinti ne pagal kurį nors atskirą įvertį, o pagal šių įverčių visumą.

2.6 Tyrimams naudojamų duomenų rinkinių apžvalga

Reikia atkreipti dėmesį, kad šia vartotojų įverčių produktams matricos forma sukaupti ir sprendimams priimti naudojami duomenys yra labai vertingi. Todėl didieji socialiniai tinklai ar elektroninės prekybos portalai jų neatskleidžia, tačiau yra ir tyrimams pasiekiamų duomenų rinkinių:

1. *MovieLens* filmų įvertinimų duomenų rinkinys[3]. Pateikiami trys duomenų rinkiniai, kuriuos sudaro:

a) 943 vartotojų įvertinimai 1682 filmams – iš viso daugiau kaip 100000 įvertinimų;

b) 6040 vartotojų įvertinimai 3900 filmų – iš viso daugiau kaip 1000000 įvertinimų;

c) 71567 vartotojų įvertinimai 10681 filmui – iš viso daugiau kaip 10000000 įvertinimų.

Dar pateikiama ir minimali vartotojų demografinė statistika – amžius, lytis, išsilavinimas. *MovieLens* duomenų rinkiniai surinkti per Minesotos universitete vykdytą *GroupLens* mokslinių tyrimų projektą. Šie rinkiniai naudojami [23] [35] ir daugelyje kitų mokslinių tyrimų.

[3] *http://www.grouplens.org/node/73*

2. *Epinions* produktų įvertinimų duomenų rinkinys[4]. Šį duomenų rinkinį sudaro dviejų tipų duomenų rinkiniai:

a) paprastasis, kuriame pateikiami 49290 vartotojų įvertinimai 139738 skirtingiems produktams ir 487181 pasitikėjimo įverčiai;

b) išplėstinis, kuriame pateikiama 132000 vartotojų pateikti 841372 pasitikėjimo įverčiai (717667 pasitikėjimo, 123705 nepasitikėjimo), 1560144 produktai ir 13668319 produktų įvertinimai.

Epinions duomenų rinkiniai surinkti Paolo Massos ir naudoti [27] [32] ir kituose tyrimuose.

3. *Book Crossing* knygų įvertinimų duomenų rinkinys[5]. Šiame duomenų rinkinyje sukaupti 278858 vartotojų įvertinimai 271379 knygoms – iš viso 1149780 įvertinimų. Duomenų rinkinio duomenų tyrimo rezultatai detaliai paskelbti [74], rinkinys taip pat naudotas [73] ir kituose darbuose.

4. *Jester* anekdotų įvertinimų duomenų rinkinys[6]. Pateikiami 73421 vartotojo įvertinimai 100 anekdotų (vertinimų intervalas $[-10; 10]$) – iš viso apie 1000000 įvertinimų. Duomenų rinkinys naudotas tyrime [13] ir kt. Šis duomenų rinkinys gana specifiškas: mažas produktų skaičius (100) lemia, kad įverčiai užpildyti net 50 %. Dėl šios priežasties *Jester* rinkinys tinkamas vertinti naujų metodų efektyvumui, kai eliminuojamos paklaidos, sukeltos tuščių įverčių. Tokio tipo duomenų rinkiniams kurtas ir 4 šios disertacijos dalyje pristatomas rekomendavimo metodas.

Disertacijos rengimo metu dar naudoti ir du anksčiau netirti duomenų rinkiniai:

1. Lietuvoje veikiančio elektroninio knygyno duomenų rinkinys. Nuo el. knygyno atidarymo buvo siūloma įsigyti 19329 skirtingas knygas, iš kurių 12564 bent kartą nupirktos. Pirkimo istorijoje užfiksuoti 14197 vartotojai, iš kurių 9930 įsigijo bent po vieną knygą. Taip pat pateikiama informacija apie 41177 pardavimus (be pasikartojimų) ir jų datas. Tai – gana netinkamas

[4] *http://www.trustlet.org/wiki/Epinions_dataset*
[5] *http://www.informatik.uni-freiburg.de/~cziegler/BX/*
[6] *http://www.ieor.berkeley.edu/~goldberg/jester-data/*

tyrimams duomenų rinkinys, nes duomenų kiekis ir užpildymas santykinai mažas. Šio rinkinio moksliniai tyrimai pritaikyti praktikoje – įvykdyti darbai pagal MTEP sutartį „Knygyno prekių rekomendacinės sistemos metodika“, vieši rezultatai publikuoti [4A].

2. *Sapnai.net* tinklalapio sapnų reikšmių paieškų rinkinys[7]. Rinkinyje pateikiami 1473252 vartotojų paieškų sapnų reikšmių duomenų bazėje rezultatai. Iš viso – 4200 sapnų reikšmių ir 27157698 įrašų. Tai – disertacijos autoriaus 2012–2015 metais rinktas duomenų rinkinys.

Moksliniams tyrimams prieinamų duomenų rinkinių yra ir daugiau. Pastebėtina, kad kiekvienas rinkinys turi savo specifiką, kuri priklauso ne tik nuo rinkinio struktūros, bet ir nuo rinkinius sukūrusių subjektų (vartotojų) fiziologinių ir psichologinių savybių.

2.7 Nemokamos ir atvirojo kodo rekomendacinių sistemų programinės įrangos apžvalga

Jei turimoje programinėje įrangoje gali veikti RS, nėra būtina iš naujo programuoti algoritmų realizacijų. Pasaulyje yra ne viena jau sukurta RS programinė įranga. Kai kurios iš jų yra atvirojo kodo ir laisvai platinamos, todėl galimos taikyti ir komercinėse įmonėse.

Šiame poskyryje pateikiama trumpa plačiausiai paplitusios nemokamos arba atvirojo kodo RS programinės įrangos lyginamosios analizės suvestinė (2 lentelė). Galime pastebėti, kad iš analizuotos programinės įrangos (*MyMediaLite*[8], *Apache Mahout*[9], *GraphLab*[10], *LensKit*[11], *Waffles*[12], *easyrec*[13], *RecLab*[14], *Crab*[15], *recommenderlab*[16], *Jellyfish*[17], *wooflix*[18], *OpenSlopeOne*[19],

[7] http://www.sapnai.net/db
[8] MyMediaLite (http://mymedialite.net/)
[9] Apache Mahout (http://mahout.apache.org/)
[10] GraphLab collaborative filtering library (http://select.cs.cmu.edu/code/graphlab/pmf.html)
[11] LensKit (http://lenskit.grouplens.org/)
[12] Waffles (http://waffles.sourceforge.net/)
[13] Easyrec (http://easyrec.org/)
[14] RecLab (http://code.richrelevance.com/reclab-core/)
[15] Crab (http://muricoca.github.io/crab/)
[16] recommenderlab (http://cran.r-project.org/web/packages/recommenderlab/index.html)

AppRecommender[20]) tiek realizuotų metodų skaičiumi, tiek atnaujinimais ir priežiūra, išsiskiria *MyMediaLite* sistema [10]. Be to, atkreiptinas dėmesys į *Apache Mahout* – ji skirta dideliems duomenims analizuoti.

Metodų efektyvumas su įvairiais duomenų rinkiniais (3 skyrius) testuotas su *MyMediaLite* programine įranga.

2.8 Rekomendacinių sistemų įrankiai MyMediaLite bibliotekoje

2.7 poskyryje vertinant programinę įrangą *MyMediaLite* programinė įranga įvertinta geriausiai. Šioje programinėje įrangoje realizuoti du bendrojo filtravimo scenarijai: reitingų prognozavimas (tai yra rekomendacijų kūrimas pagal turimus įverčius produktams) ir prognozių kūrimas pagal „pirkta ar nepirkta“ duomenų rinkinį (0 ir 1 reikšmės) [10].

Reitingų prognozavimo metodai apskaičiuoja (prognozuoja) nežinomus reitingus, naudodamiesi turimu reitingų duomenų rinkiniu ir kitais duomenimis apie vartotojus ar produktus. Dauguma realiai veikiančių rekomendacinių sistemų negali naudotis reitingų duomenų rinkiniais, kadangi juos sunku gauti ar surinkti iš atsitiktinių vartotojų. Tokiu atveju naudojama teigiamų įvertinimų sistema, kai teigiamą įvertinimą atitinka nuorodos paspaudimas, produkto įsigijimas ir pan.

MyMediaLite programinėje įrangoje realizuoti keli *kNN* modeliai, paprasčiausi atskaitos metodai, matricų faktorizavimo metodai ir kiti (pavyzdžiui, *BPRMF* [44]), tinkantys reitingams prognozuoti[21] ir prognozėms kurti[22].

[17] Jellyfish (http://hazy.cs.wisc.edu/hazy/victor/download/)
[18] Wooflix (http://gustavonarea.net/blog/posts/korens-svd-python-implementation/)
[19] OpenSlopeOne (http://code.google.com/p/openslopeone/)
[20] AppRecommender (https://github.com/tassia/AppRecommender)
[21] http://www.mymedialite.net/documentation/rating_prediction.html
[22] http://www.mymedialite.net/documentation/item_prediction.html

2.9 Rekomendacinių sistemų įrankiai *Apache Mahout* bibliotekoje

Pastaruoju metu didelių duomenų analizei dažnai naudojama *Apache Mahout* įrankių biblioteka [38]. Ši biblioteka gali būti naudojama ir rekomendacijoms ar prognozėms generuoti [51]. Standartinėje *Mahout* bibliotekoje realizuoti du skirtingi rekomendavimo metodų tipai: įprastas – *recommenditembased,* kai skaičiuojami skirtingų vartotojų reitingų panašumai ir randami panašiausi, ir *recommendfactorized,* kai rekomendacijos apskaičiuojamos remiantis įverčių matricos faktorizavimu [53].

Abu tipai turi daug parametrų, kuriuos galima apžvelgti išsamiau. Pavyzdžiui, kviečiant *recommenditembased* komandą, rašoma ši sintaksė:

--input <input> --output <output> --numRecommendations <numRecommendations> --usersFile <usersFile> --itemsFile <itemsFile> --filterFile <filterFile> --booleanData <booleanData> --maxPrefsPerUser <maxPrefsPerUser> --minPrefsPerUser <minPrefsPerUser> --maxSimilaritiesPerItem <maxSimilaritiesPerItem> --maxPrefsInItemSimilarity <maxPrefsInItemSimilarity> --similarityClassname <similarityClassname> --threshold <threshold> --outputPathForSimilarityMatrix <outputPathForSimilarityMatrix> --randomSeed <randomSeed> --sequencefileOutput --help --tempDir <tempDir> --startPhase <startPhase> --endPhase <endPhase>

Reikšmingiausias – *similarityClassname* parametras. Šis parametras nurodo, kokį konkretų algoritmą taikyti skaičiuojant panašumus tarp įverčių. Įprastose rekomendacinėse sistemose dažniausiai taikomi kosinusinio (cosine), Pearsono koreliacijos ir Euklido atstumų panašumų matai [41].

Apache Mahout bibliotekoje pasirinkimo variantai yra keli:

SIMILARITY_COOCCURRENCE,

SIMILARITY_LOGLIKELIHOOD,

SIMILARITY_TANIMOTO_COEFFICIENT,

SIMILARITY_CITY_BLOCK,

SIMILARITY_COSINE,

SIMILARITY_PEARSON_CORRELATION,

SIMILARITY_EUCLIDEAN_DISTANCE

2 lentelė. *Nemokamos ir atvirojo kodo RS programinės įrangos lyginimas (2013 m. gegužės mėnesio duomenimis)*

Rekomendacinė sistema	Paskutinis atnaujinimas	Programavimo kalba	Realizuotų RS metodų skaičius	Galimybė realizuoti savo metodą	Reitingų prognozavimas	Produktų rekomendacijos
MyMediaLite	2013 02	*C#*	>20	Taip	Taip	Taip
Apache Mahout	2012 06	*Java*	3	Ne	Taip	Taip
GraphLab	2012 05	*C++*	15	Taip	Taip	Taip
LensKit	2012	*Java*	?	Taip	Taip	Taip
Waffles	2013 04	*C++*	>5	Ne	Taip	Ne
easyrec	2012 02	*Online*	1	Ne	Taip	Taip
RecLab	2011 02	*Online*	1	Ne	Taip	Ne
Crab	2011	*Python*	>5	Taip	Taip	Taip
recommenderlab	2011 11	*C++*	4	Taip	Taip	Taip
Jellyfish	2012 12	*Python*	1	Ne	Taip	Ne
wooflix	2009 06	*Python*	1	Ne	Taip	Ne
OpenSlopeOne	2010 06	*PHP*	1	Taip	Taip	Ne
AppRecommender	2011	*Python*	>10	Ne	Taip	Taip

Kviečiant *recommendfactorized* komandą, rašoma ši sintaksė:

--input <input> --userFeatures <userFeatures> --itemFeatures <itemFeatures> --numRecommendations <numRecommendations> --maxRating <maxRating> --numThreads <numThreads> --usesLongIDs <usesLongIDs> --userIDIndex <userIDIndex> --itemIDIndex <itemIDIndex> --output <output> --help --tempDir <tempDir> --startPhase <startPhase> --endPhase <endPhase>.

Didžiausią vaidmenį vaidina *userFeatures* arba *itemFeatures* matricos – remiantis jomis apskaičiuojamos tinkamiausios rekomendacijos konkrečiam vartotojui.

Šiais dviem aptartais metodų tipais galima spręsti standartinius rekomendavimo uždavinius – sugeneruoti tinkamiausių prekių sąrašą konkretiems vartotojams. Dėl įvairių būdų skaičiuoti panašumo koeficientą kiekvienam specifiniui duomenų apie vartotojų įverčius produktams rinkiniui galima gauti geriausius rekomendavimo rezultatus.

2.10 Rekomendacinių sistemų naudojamų duomenų etikos problemos

RS naudojimo sritis yra gana jautri etikos požiūriu. Nors šiuo metu šiai problemai skiriama mažai dėmesio, tačiau jau pasirodė pirmieji straipsniai, atkreipiantys dėmesį į tai, kad RS, kurdamos rekomendacijas, renka ir manipuliuoja asmeniniais vartotojo duomenimis [42]. Pasirodė ir siūlymų, kaip kurti rekomendacijas saugant vartotojų duomenis (pagrindinė problema yra ne vartotojo duomenų naudojimas, o kaupimas ir perdavimo trečiosioms šalims grėsmė) [39] [59]. Deja, vyriausybės skiria vis dar mažai dėmesio vartotojų duomenų apsaugai internete, priimtos direktyvos vis dar netaikomos – skaitmeninio pėdsako (angl. *Digital Footprint*) šnipinėjimas ir stebėjimas labai menkai ribojimas. Tačiau atsižvelgiant į tai, kad tyrimai skaitmeninės etikos klausimu jau pradėti (Europos Komisija inicijavo tyrimą 2030 metų skaitmeninės Europos scenarijui numatyti), tikėtina, kad vartotojų privatumo saugojimo politika pasieks ir RS taikymo sritį, ypač taikymo socialiniuose

tinkluose, aplinkoje, kuri šiuo metu kelia daugiausia diskusijų vartotojų privatumo klausimais.

2.11 Antrojo skyriaus apibendrinimas ir išvados

Interneto socialiniai tinklai yra gana naujas reiškinys mūsų gyvenime. Kita vertus, dabar jie vis populiaresni ir vis greičiau plečiasi. Siekiant geriau įvertinti socialinių tinklų vartotojų poreikius ir rekomenduoti jiems tinkamus produktus ar paslaugas, šie vartotojai įvairiopai analizuojami.

Vienas iš socialinių tinklų vartotojų analizės ir jiems tinkančių produktų ar paslaugų parinkimo būdų – rekomendacinių sistemų taikymas. Pagrindinės šių sistemų taikymo sritys yra elektroninėje komercijoje ir socialiniuose tinkluose. Socialiniuose tinkluose susiklostė labai palankios sąlygos rekomendacinėms sistemoms taikyti – aibės vartotojų vertina, komentuoja įvairius produktus. Šiuos vertinimus rekomendacinės sistemos dažniausiai ir naudoja rekomendacijoms teikti.

Kad būtų tiksliau nustatyti vartotojų poreikiai, pateiktos kaip galima tikslesnės rekomendacijos (dažnai randama ir „plika akimi“ nepastebimų ryšių), rekomendacinės sistemos taiko įvairius duomenų gavybos metodus – klasterizavimą, faktorizavimą, neuroninius tinklus ir t. t.

Šiame skyriuje apžvelgta mokslinė literatūra, susijusi su minėtomis problemomis ir uždaviniais – rekomendacinių sistemų veikimu socialiniuose tinkluose, rekomendavimo metodais ir šiuose metoduose taikomais duomenų gavybos metodais. Iš atliktos apžvalgos galime pastebėti, kad duomenų gavybos metodai rekomendacinėse sistemose ne visada akivaizdžiai matomi, tačiau tai, kad šie metodai (ar metodų sintezė) taikomi rekomenduojant ir prognozuojant, – akivaizdu.

Analizė parodė, kad siekiant surasti efektyviausiai su konkrečiu duomenų rinkiniu veikiančius rekomendavimo metodus, jų veikimo rezultatus būtina vertinti ne pagal kurį nors atskirą įvertį, o pagal šių įverčių visumą.

Plačiausiai paplitusių nemokamo arba atvirojo kodo RS programinės

įrangos lyginamoji analizė atskleidė, kad iš analizuotos programinės įrangos (*MyMediaLite, Apache Mahout, GraphLab, LensKit, Waffles, easyrec, RecLab Crab, recommenderlab, Jellyfish, wooflix, OpenSlopeOne, AppRecommender*), tiek realizuotų metodų skaičiumi, tiek atnaujinimais ir priežiūra, *MyMediaLite* sistema yra pati geriausia. Ši sistema galėtų būti atskaitos taškas naujoms sistemoms kurti ir jau esančioms tobulinti.

Matyti, kad yra daug rekomendacinių metodų ir sistemų, tačiau ši įvairovė reiškia ir tai, kad jie neuniversalūs.

3

Populiariausių rekomendavimo metodų eksperimentinis įvertinimas

Šiame skyriuje atliekamas eksperimentinis populiariausių rekomendavimo metodų įvertinimas. Pagrindiniai skyriaus rezultatai yra paskelbti moksliniame straipsnyje [4A] ir pristatyti trijose mokslinėse konferencijose.

3.1 Analizės tikslas

Šiuo metu egzistuoja daug siauriau ar plačiau paplitusių rekomendacinėse sistemose taikomų metodų. Kiekvienas iš jų turi privalumų ir trūkumų, dauguma orientuota į konkretaus tipo duomenų rinkinius. Šios analizės tikslas – eksperimentiškai ištirti populiariausių RS metodų veikimą su skirtingais duomenų rinkiniais, taip įvertinti šių metodų globalų efektyvumą.

3.2 Analizės eiga

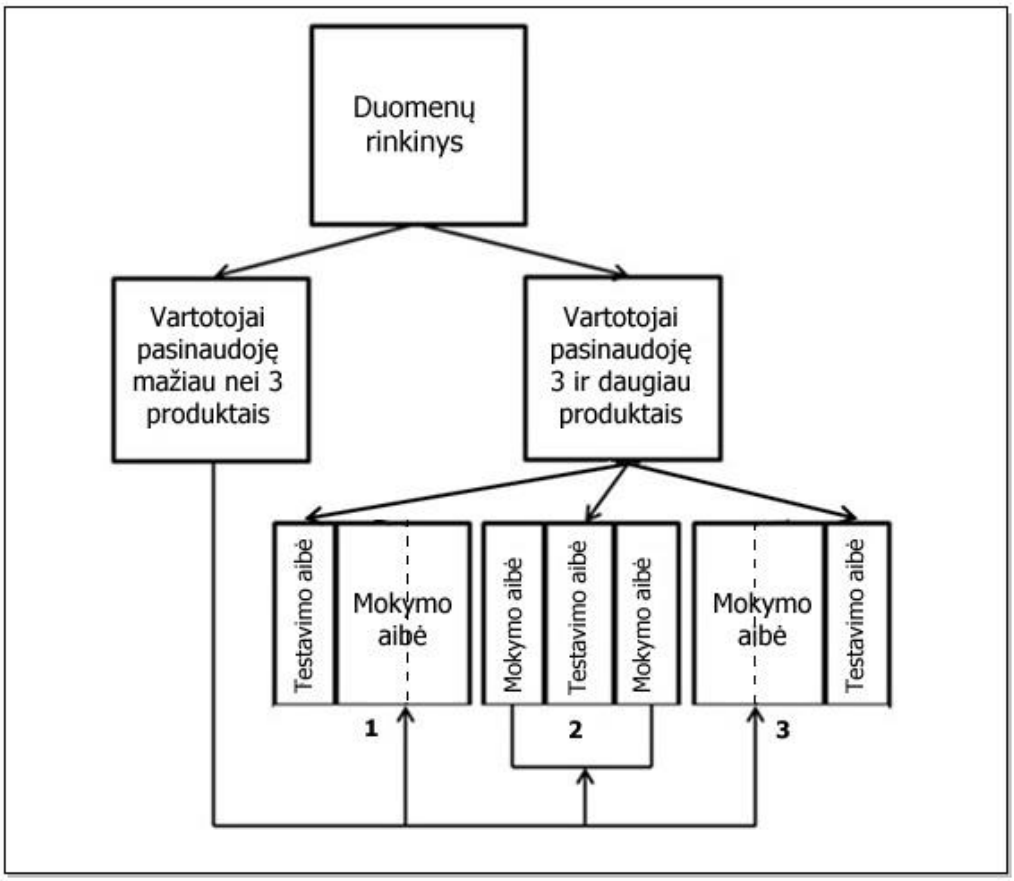
Per rekomendacinių sistemų metodų rezultatų vertinimo eksperimentus taikytas kryžminio patvirtinimo modelis (10 paveikslas). Iš viso duomenų rinkinio į mokymo aibę buvo perkeliami tų vartotojų duomenys, kurie pasinaudojo (įsigijo) mažiau produktų nei K. Kitų duomenų rinkinys

padalijamas į *K* lygių dalių. Tada viena suskirstyto rinkinio dalis atskiriama kaip testavimo aibė, o kitos priskiriamos prie mokymo aibės. Eksperimentai atliekami su visu duomenų rinkiniu, nuosekliai keičiama dalis, laikoma testavimo aibe.

Vykdyti dviejų tipų eksperimentai: reitingų prognozavimo ir produktų rekomendavimo. Kiekvienam tipui parinkta po du skirtingus duomenų rinkinius ir metodus, realizuotus *MyMediaLite* programinės įrangos bibliotekoje. *MyMediaLite* pasirinkimą sąlygojo 2 skyriaus išvados. *C#* ir *MyMediaLite* susiejimo kodas pateiktas A priede.

Prognozuoti reitingams pasirinkti *Jester* ir *MovieLens* duomenų rinkiniai, juose atlikti eksperimentai taikant *Random*, *SlopeOne*, *GlobalAverage*, *ItemAverage*, *MatrixFactorization*, *UserAverage*, *UserItemBaseline*, *CoClustering*, *LatentFeatureLogLinearModel*, *BiasedMatrixFactorization*, *SVDPlusPlus*, *UserKNN*, *ItemKNN* ir *SigmoidCombinedAsymmetricFactorModel* metodus. Eksperimentai atlikti su keliomis *K* reikšmėmis, siekiama pastebėti, ar vartotojų įvertintų produktų skaičius turi didelę įtaką skaičiavimų rezultatams. Metodų efektyvumas įvertinamas pagal *RMSE*, *MAE* ir *NMAE* (žr. 2.3.1.2) įverčius. Gauti rezultatai ranguojami ir tokiu būdu išrenkami geriausi metodai kiekvienam duomenų rinkiniui.

Produktų rekomendavimui tirti pasirinkti labiau specifiniai ir anksčiau kitų autorių netirti vieno Lietuvoje veikiančio el. knygyno ir *Sapnai.net* sapnų reikšmių tinklalapio duomenų rinkiniai. Pirmasis rinkinys – išplėstinis, jam galima taikyti rekomendavimo metodus, kuriems reikalingi atributai. Tad eksperimentai su šiuo rinkiniu atlikti *Random*, *Zero*, *MostPopular*, *MostPopByAttributes*, *BPRMF*, *BPRLinear*, *SoftMarginRankMF*, *WRMF*, *WeightedBPRMF*, *MultiCoreBPRMF*, *CLiMF*, *BPRSLIM*, *UserKNN*, *ItemKNN*, *ItemAttributeKNN*, *BPRLinear* ir *ItemAttributeKNN* metodais.



10 paveikslas. *Kryžminio patvirtinimo modelis, kai duomenų rinkinys dalijamas į* $K = 3$ *dalis*

Eksperimentai su *Sapnai.net* duomenų rinkiniu atlikti taikant *Random*, *Zero*, *MostPopular*, *BPRMF*, *BPRLinear*, *SoftMarginRankingMF*, *WRMF*, *WeightedBPRMF*, *MultiCoreBPRMF*, *BPRSLIM*, *UserKNN* ir *ItemKNN* metodus. Eksperimentai atlikti su keliomis *K* reikšmėmis, siekiama pastebėti, ar vartotojų įvertintų produktų skaičius turi didelę įtaką skaičiavimų rezultatams. Metodų efektyvumas įvertinamas pagal *AUC* (žr. 2.5.3), *prec@5*, *prec@10*, *recall@5*, *recall@10* (žr. 2.5.5), *MAP* (žr. 2.5.4), *NDCG* (žr. 2.5.6) ir *MRR* (žr. 2.5.7) įverčiai. Gauti rezultatai ranguojami ir tokiu būdu išrenkami geriausi metodai kiekvienam atvejui.

Kryžminio patvirtinimo eksperimentų rezultatų dalis pateikiama 3–6 lentelėse ir 13, 16, 19 bei 25 paveiksluose. Lentelėse pateikta eksperimentų rezultatų suvestinė, kai yra konkretus kryžminio patvirtinimo dalių skaičius K. Kiekvieno eksperimento metu metodo efektyvumas vertintas pagal kelis

skirtingus 2.3.1.2 ir 2.5 poskyriuose aprašytus efektyvumą įvertinančius matus. Atsižvelgiant į rezultatus su kiekvienu matu, metodai ranguojami pagal užimamą vietą, metodas vertinamas pagal konkretų efektyvumo matą (1-oji vieta – geriausias metodas ir t. t.). Paskui išvedamas kiekvieno metodo užimamos vietos vidurkis; mažiausią vidurkį turintys metodai laikomi geriausiais. Visi metodai vėl išranguojami pagal šiuos vidurkius ir taip gaunama galutinė vieta metodų sąraše.

Random metodas yra atskaitos taškas įvertinti kitų metodų tikslumui. Taikant *Random* metodą, spėjamas produkto įvertis atsitiktinai parenkamas iš viso galimo įverčių intervalo. Visais intelektualesniais metodais turėtume gauti geresnių rezultatų nei *Random* metodu.

3.3 Eksperimentai su *Jester* duomenų rinkiniu

Tyrimui imami duomenys apie *Jester* anekdotų rinkinio atskirų anekdotų vertinimo istoriją, sukauptą nuo 1999 m. balandžio iki 2003 m. gegužės mėn. Rinkinio formatas pateikiamas 11 paveiksle.

Eksperimentiniams tyrimams naudotame duomenų rinkinyje sukaupti 24984 vartotojų įverčiai 100 anekdotų, iš viso 1810455 skirtingi intervalo $[-10;\ 10]$ įverčiai. Duomenų rinkinio užpildymas – 72,5 %.

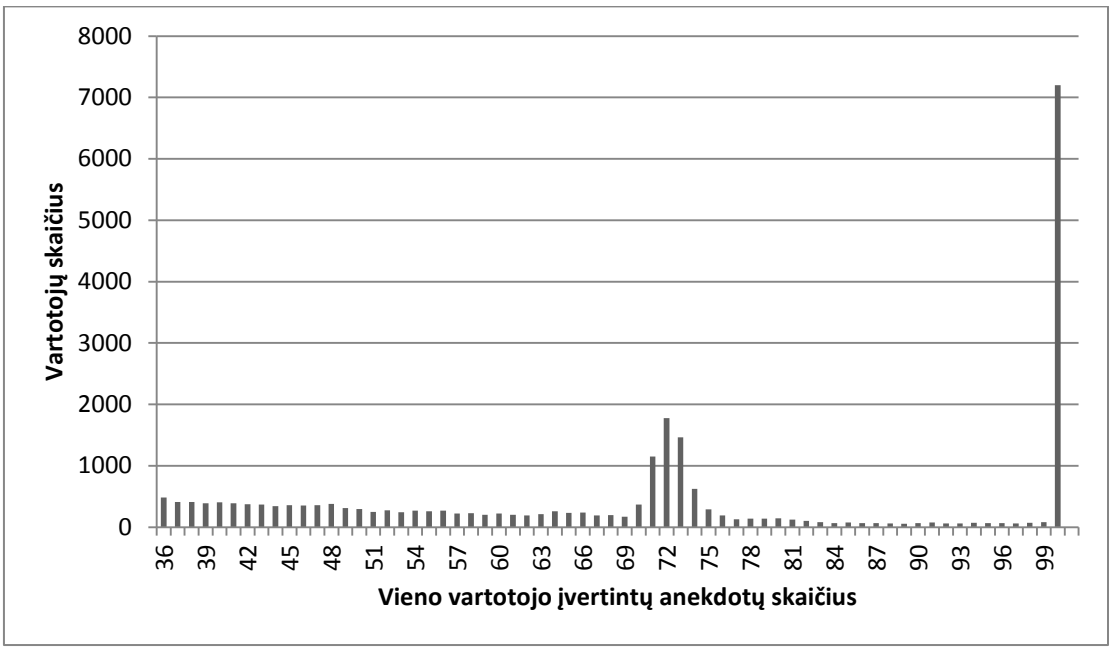
1 vartotojas	Įverčių skaičius	1 anekdoto įvertis	<...>	100-ojo anekdoto įvertis
<…>				
24984-asis	Įverčių skaičius	1 anekdoto įvertis	<...>	100-ojo anekdoto įvertis

11 paveikslas. *Tyrimui naudoto duomenų rinkinio struktūra*

Pakankamai svarbus duomenų rinkinio analizės aspektas yra vartotojų pasiskirstymo pagal įvertintų anekdotų skaičių nustatymas. Šis pasiskirstymas iliustruojamas 12 paveikslo grafike.

Kaip galima pastebėti iš 12 paveikslo, skirstinys nėra tolygus. Rinkinyje turime santykinai didelį skaičių (7200) vartotojų, įvertinusių visus anekdotus, ir kelis mažesnius pikus intervale [71; 74]. Šio rinkinio užpildymas gana didelis, todėl prognozių tikslumas irgi turėtų būti santykinai didelis. Šis

duomenų rinkinys labai tinka disertacijoje siūlomo metodo eksperimentiniam tyrimui. Taip pat šis rinkinys parodo, kad praktikoje iš tikrųjų yra tokių rinkinių, kur vartotojų įverčių produktams matricos užpildymo tankis yra didelis.



12 paveikslas. *Vartotojų pasiskirstymas pagal įvertintų anekdotų skaičių*

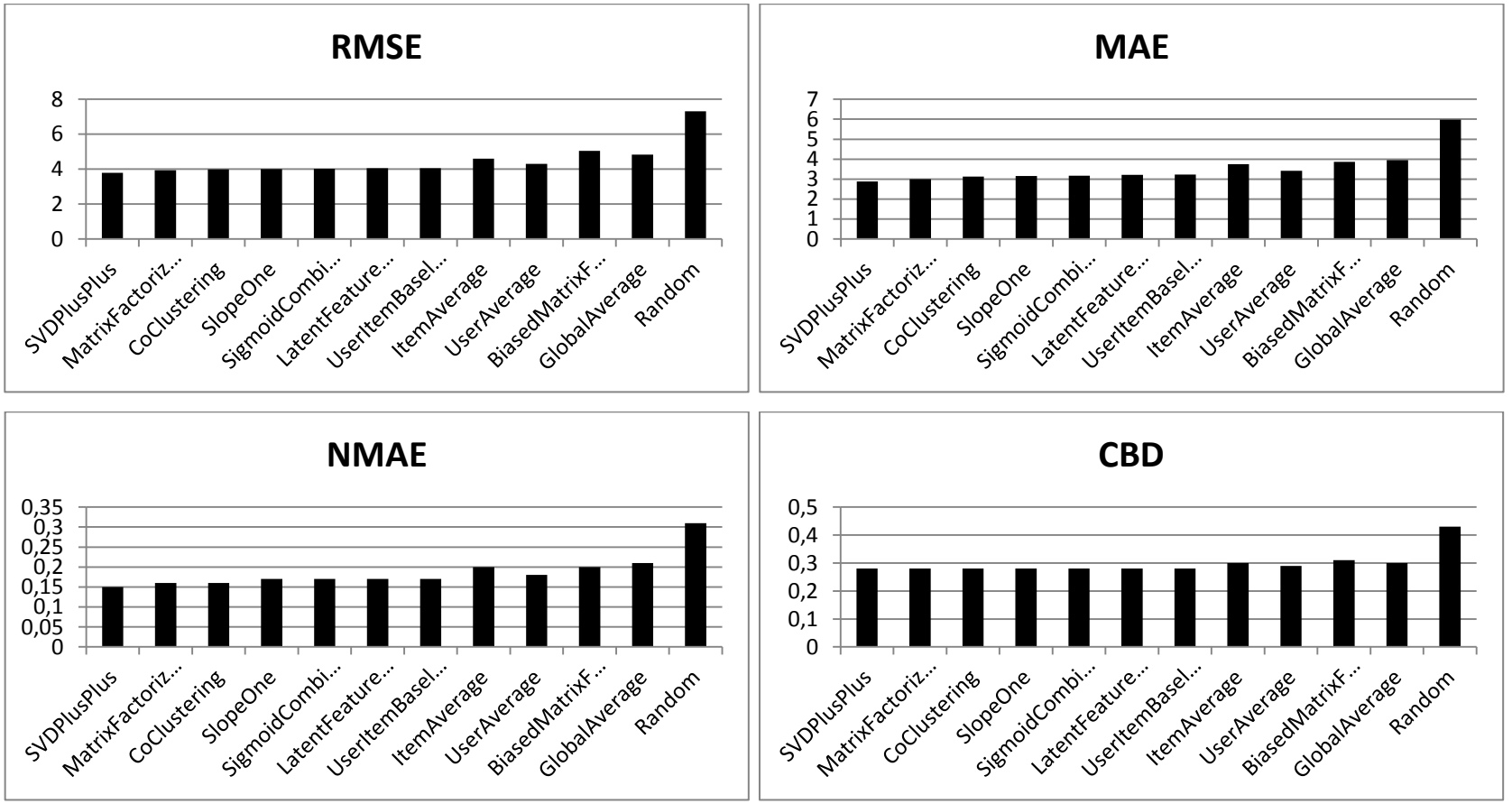
Siekiant įvertinti metodų efektyvumą, kai vartotojo įvertintų anekdotų skaičius kinta, eksperimentai atlikti taikant kryžminio patvirtinimo (angl. C*ross Validation)* modelį. Pastebėta, kad K reikšmių didėjimas turi mažai įtakos eksperimentų rezultatams. *RMSE*, *MAE*, *NMAE* ir *CBD* įverčiai pagal metodus, kai $K = 70$, pateikiami 3 lentelėje ir iliustruojami 13 paveiksle pateiktais grafikais.

Atlikus visų įverčių tipų rangavimą, pagal rezultatų tikslumą metodai išrikiuoti šia tvarka:

1. *SVDPlusPlus*
2. *MatrixFactorization*
3. *CoClustering*
4. *SlopeOne*
5. *SigmoidCombinedAsymmetricFactorModel*

3 lentelė. *Kryžminio patvirtinimo eksperimentų rezultatas ($K = 70$)*

Metodas	RMSE	Vieta	MAE	Vieta	NMAE	Vieta	CBD	Vieta	Vidurkis	Galutinė vieta
Random	7,31	12	5,98	12	0,31	12	0,43	12	12	**12**
SlopeOne	4	4	3,16	4	0,17	4	0,28	1	3,25	**4**
GlobalAverage	4,84	10	3,95	11	0,21	11	0,3	10	10,5	**11**
ItemAverage	4,6	9	3,75	9	0,2	9	0,3	9	9	**9**
MatrixFactorization	3,94	2	3	2	0,16	2	0,28	1	1,75	**2**
UserAverage	4,3	8	3,42	8	0,18	8	0,29	8	8	**9**
UserItemBaseline	4,05	6	3,23	7	0,17	4	0,28	1	4,5	**6**
CoClustering	3,99	3	3,13	3	0,16	2	0,28	1	2,25	**3**
LatentFeatureLog LinearModel	4,06	7	3,21	6	0,17	4	0,28	1	4,5	**6**
BiasedMatrix Factorization	5,05	11	3,87	10	0,2	9	0,31	11	10,25	**10**
SVDPlusPlus	3,79	1	2,88	1	0,15	1	0,28	1	1	**1**
SigmoidCombined AsymmetricFactor Model	4,03	5	3,18	5	0,17	4	0,28	1	3,75	**5**
UserKNN	-	-	-	-	-	-	-	-	-	**-**
ItemKNN	-	-	-	-	-	-	-	-	-	**-**



13 paveikslas. *Skirtingų įverčių reikšmės skirtingiems metodams*

6. *LatentFeatureLogLinearModel*
7. *UserItemBaseline*
8. *ItemAverage*
9. *UserAverage*
10. *BiasedMatrixFactorization*
11. *GlobalAverage*
12. *Random*

Eksperimentų su *UserKNN* ir *ItemKNN* metodais atlikti nepavyko, mat šie metodai, taikomi tokiam duomenų rinkiniui, reikalauja daug skaičiavimo resursų. Kompiuterio 16 GB operatyviosios atminties nepakako.

3.4 Eksperimentai su *MovieLens* duomenų rinkiniu

Tyrimui buvo imami duomenys apie *MovieLens* duomenų bazės filmų rinkinio atskirų filmų vertinimo istoriją, sukauptą nuo 2000 m. balandžio iki 2003 m. vasario mėn.

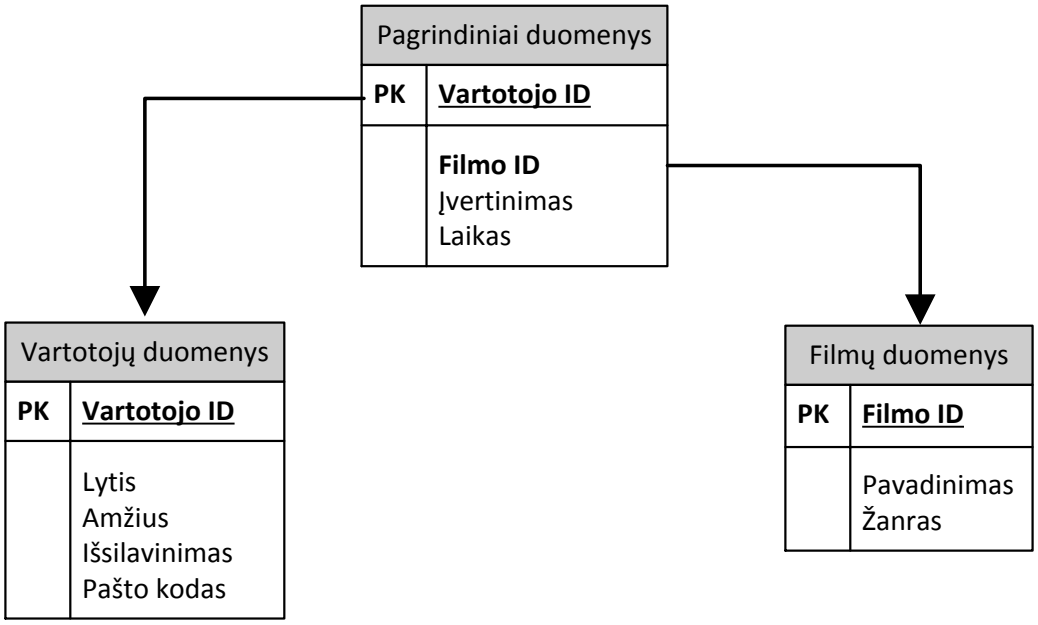
Eksperimentiniams tyrimams naudotame duomenų rinkinyje sukaupti 6040 vartotojų įverčiai 3706 filmams, iš viso 1000209 skirtingi intervalo [1; 5] įverčiai. Duomenų rinkinio užpildymas – 4,5 %. Dar rinkinyje pateikiami kiti vartotojų duomenys (lytis, amžius, išsilavinimas ir gyvenamoji vieta pagal pašto kodą) ir filmų duomenys (pavadinimas ir žanras). Visa duomenų rinkinio struktūra pateikiama 14 paveiksle.

Pakankamai svarbus duomenų rinkinio analizės aspektas yra vartotojų pasiskirstymo pagal įvertintų filmų skaičių nustatymas. Šis pasiskirstymas iliustruojamas 15 paveiksle.

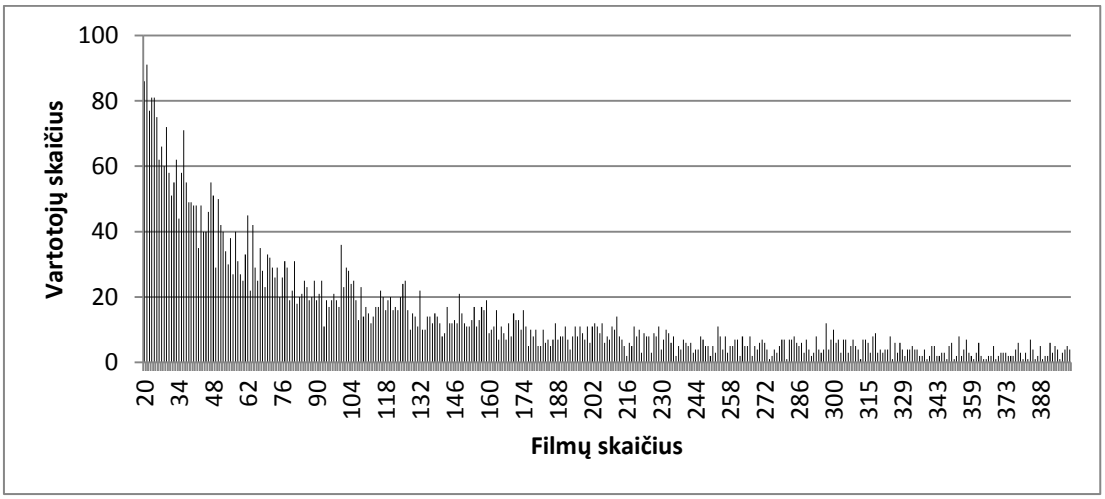
Kaip galima matyti iš 15 paveikslo grafiko, skirstinys yra gana tolygiai mažėjantis su keliais nežymiais pikais. Didesnė dalis vartotojų yra įvertinusi iki 200 filmų.

Siekiant įvertinti metodų efektyvumą, kai vartotojo įvertintų filmų skaičius kinta, eksperimentai atlikti taikant kryžminio patvirtinimo modelį. Šiuo atveju, kaip ir ankstesnio tyrimo, K reikšmės įtaka eksperimentų

rezultatams yra nežymi. *RMSE*, *MAE*, *NMAE* ir *CBD* įverčiai pagal metodus, kai $K = 50$, pateikiami 4 lentelėje ir iliustruojami 16 paveiksle pateiktais grafikais.



14 paveikslas. *Tyrimui naudoto duomenų rinkinio struktūra*



15 paveikslas. *Vartotojų pasiskirstymas pagal įvertintų filmų skaičių*

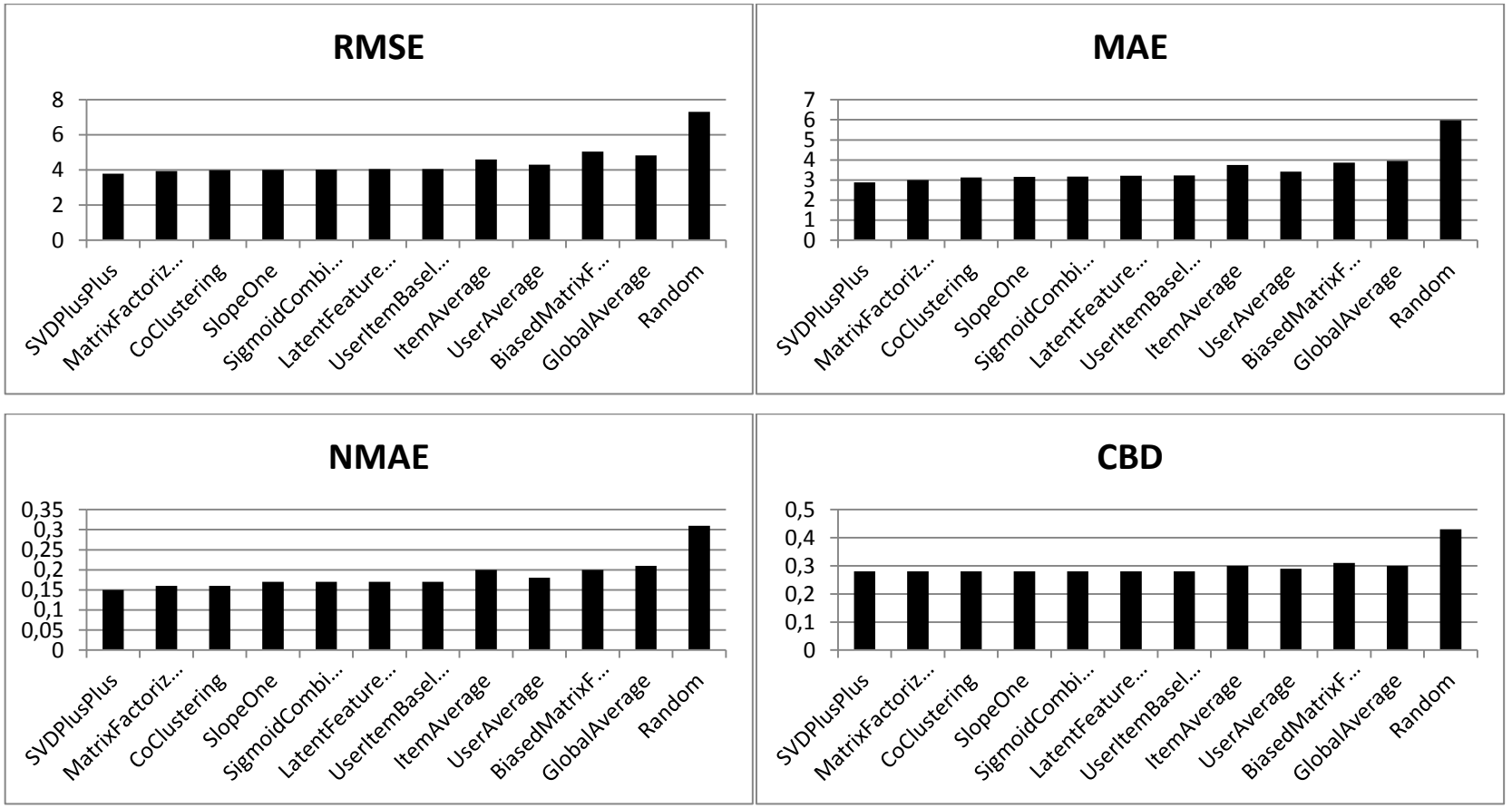
Atlikus visų įverčių tipų rangavimą, pagal rezultatų tikslumą metodai išrikiuoti šia tvarka:

1-2. MatrixFactorization

1-2. BiasedMatrixFactorization

4 lentelė. *Kryžminio patvirtinimo eksperimentų rezultatas ($K = 50$)*

Metodas	RMSE	Vieta	MAE	Vieta	NMAE	Vieta	CBD	Vieta	Vidurkis	Galutinė vieta
Random	1,70	11	1,40	11	0,35	11	0,43	11	11	**11**
SlopeOne	0,90	4	0,71	5	0,18	4	0,26	4	4,25	**5**
GlobalAverage	1,12	10	0,93	10	0,23	10	0,28	10	10	**10**
ItemAverage	0,98	8	0,78	8	0,20	8	0,26	4	7	**8**
MatrixFactorization	0,85	1	0,67	1	0,17	1	0,25	1	1	**1**
UserAverage	1,03	9	0,83	9	0,21	9	0,27	9	9	**9**
UserItemBaseline	0,91	6	0,72	7	0,18	4	0,26	4	5,25	**7**
CoClustering	0,91	6	0,71	5	0,18	4	0,26	4	4,75	**6**
LatentFeatureLog LinearModel	0,90	4	0,70	3	0,18	4	0,26	4	3,75	**4**
BiasedMatrix Factorization	0,85	1	0,67	1	0,17	1	0,25	1	1	**1**
SVDPlusPlus	0,89	3	0,70	3	0,17	1	0,25	1	2	**3**
SigmoidCombined AsymmetricFactor Model	-	-	-	-	-	-	-	-	-	**-**
UserKNN	-	-	-	-	-	-	-	-	-	**-**
ItemKNN	-	-	-	-	-	-	-	-	-	**-**



16 paveikslas. *Skirtingų įverčių reikšmės skirtingiems metodams*

3. *SVDPlusPlus*
4. *LatentFeatureLogLinearModel*
5. *SlopeOne*
6. *CoClustering*
7. *UserItemBaseline*
8. *ItemAverage*
9. *UserAverage*
10. *GlobalAverage*
11. *Random*

Eksperimentų su *UserKNN* ir *ItemKNN* metodais atlikti nepavyko – kaip minėta, šie metodai, taikomi tokiam duomenų rinkiniui, reikalauja daug skaičiavimo resursų; 16 GB operatyviosios atminties nepakanka. Tokio dydžio rinkinio eksperimentų nepavyko atlikti ir taikant *SigmoidCombinedAsymmetricFactorModel* metodą.

3.5 Eksperimentai su *Sapnai.net* duomenų rinkiniu

Tyrimui imami duomenys apie *Sapnai.net* tinklalapio lankytojų sapnų paieškos istoriją nuo 2012 m. vasario iki 2015 m. kovo mėn. Šiame rinkinyje taip pat pateiktas paieškos laikas *UNIX* formatu. Rinkinys paremtas anonimiškumu – informacijos, leidžiančios identifikuoti tinklalapio lankytojus, juos susieti su konkrečiais asmenimis ir taip pažeisti jų privatumą, pateikta nėra.

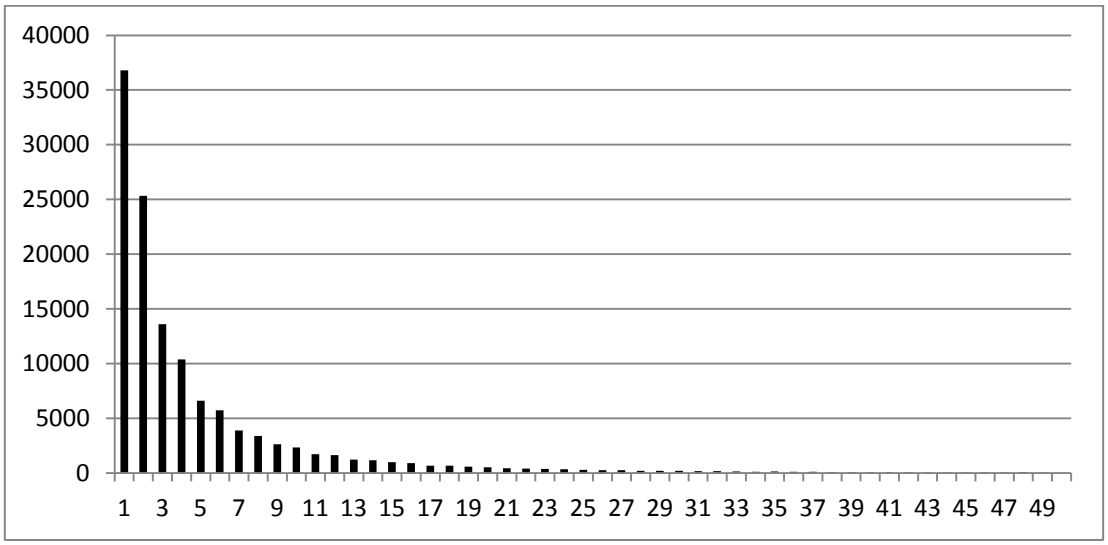
Rinkinyje pateikiami 1473252 vartotojų paieškų sapnų reikšmių duomenų bazėje rezultatai. Iš viso – 4200 sapnų reikšmių ir 27157698 įrašų. Rinkinio užpildymas – 0,43 %. Kaip minėta, šis rinkinys rinktas disertacijos autoriaus 2012–2015 metais. Tyrimai, atlikti su šiuo duomenų rinkiniu, dar nebuvo publikuoti. Duomenų rinkinio formatas pateikiamas 17 paveiksle.

Dėl skaičiavimo resursų trūkumo atlikti eksperimentų su visu duomenų rinkiniu šiuo metu neįmanoma, todėl iš viso rinkinio išrinkta 1000000 atsitiktinių skirtingų įrašų. 18 paveiksle pateiktas šio rinkinio vartotojų

pasiskirstymas pagal ieškotų sapnų reikšmių skaičių. Pateikiami sumažinto rinkinio 127538 vartotojų paieškų sapnų reikšmių duomenų bazėje rezultatai. Iš viso – 4200 sapnų reikšmių ir 1000000 įrašų.

1 įrašas	1 vartotojas	Sapno numeris	Sapno reikšmė	*UNIX* laikas
<...>				
271576898 įrašas	X vartotojas	Sapno numeris	Sapno reikšmė	*UNIX* laikas

17 paveikslas. *Tyrimui naudoto duomenų rinkinio struktūra*



18 paveikslas. *Vartotojų pasiskirstymas pagal sapnų skaičių*

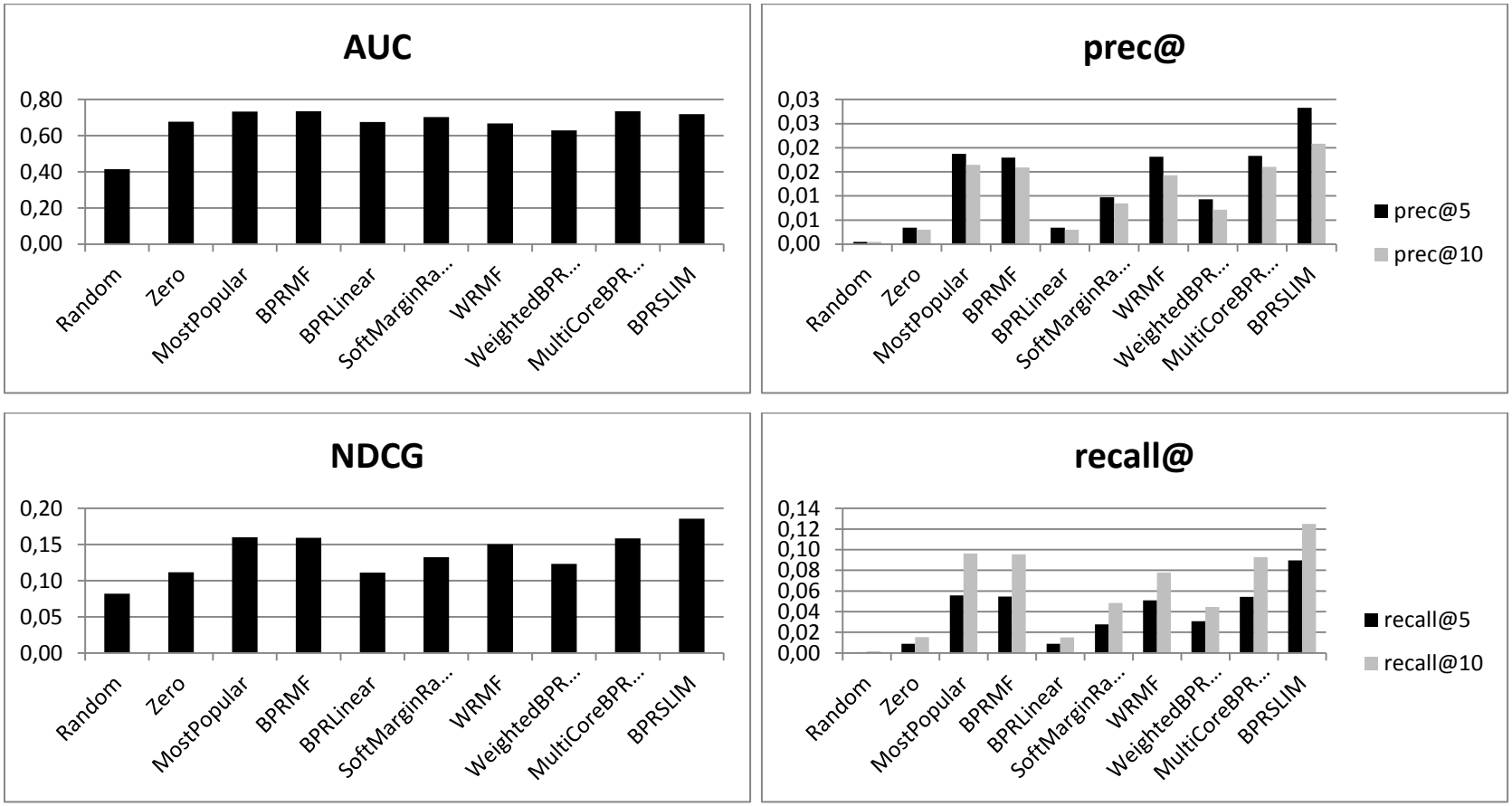
Kaip matyti iš 18 paveiksle pateikto grafiko, 28 % vartotojų ieškojo tik vienos sapno reikšmės, o 67 % sistemos vartotojų yra ieškoję mažiau kaip 5 sapnų reikšmių. Prognozės tokiems vartotojams nebūna labai tikslios, kadangi atsiranda tuščių įvertinimų ir naujų vartotojų problemos, detaliau aprašytos 2.4.5 poskyryje.

Duomenų rinkinyje pastebima ir anomalijų – keli vartotojai yra ieškoję daugiau kaip 1000 sapnų reikšmių. Dažnai šie vartotojai yra paieškos robotai; jų veiklos istorija gali iškreipti rezultatus.

5 lentelė. *Kryžminio patvirtinimo eksperimentų rezultatas ($K = 5$)*

Metodas	AUC	Vieta	prec@5	Vieta	prec @10	Vieta	MAP	Vieta	recall@ 10	Vieta	*	Vidurkis	Galutinė vieta
Random	0,41	10	0,00	8	0,00	8	0,00	10	0,00	10		9,5	**10**
Zero	0,68	6	0,00	8	0,00	8	0,01	8	0,02	8		7,75	**8**
MostPopular	0,73	1	0,02	2	0,02	1	0,04	2	0,10	2		1,75	**2**
BPRMF	0,73	1	0,02	2	0,02	1	0,04	2	0,10	2		1,875	**3**
BPRLinear	0,68	6	0,00	8	0,00	8	0,01	8	0,02	8		7,75	**8**
SoftMargin RankingMF	0,70	5	0,01	6	0,01	5	0,02	7	0,05	6		5,875	**6**
WRMF	0,67	8	0,02	2	0,01	5	0,04	2	0,08	5	.	4	**5**
Weighted BPRMF	0,63	9	0,01	6	0,01	5	0,03	6	0,04	7		6,5	**7**
MultiCore BPRMF	0,73	1	0,02	2	0,02	1	0,04	2	0,09	4		2,125	**4**
BPRSLIM	0,72	4	0,03	1	0,02	1	0,07	1	0,12	1		1,375	**1**
ItemKNN	-	-	-	-	-	-	-	-	-	-		-	**-**
UserKNN	-	-	-	-	-	-	-	-	-	-		-	**-**

* Taip pat įvertinami ir suranguojami *recall@10*, *NDCG* bei *MRR* įverčiai. Apskaičiuojamas rangų vidurkis ir pagal jį išrenkami geriausi metodai.



19 paveikslas. *Skirtingų įverčių reikšmės skirtingiems metodams*

Siekiant įvertinti metodų efektyvumą, kai vartotojo įvertintų anekdotų skaičius kinta, eksperimentai atlikti taikant kryžminio patvirtinimo modelį. Šiuo atveju, kaip ir ankstesnių dviejų tyrimų, K reikšmės įtaka eksperimentų rezultatams yra nežymi. *AUC*, *NDCG*, *recall@5*, *recall@10* ir *prec@5*, *prec@10* įverčiai pagal metodus, kai $K = 5$, pateikiami 5 lentelėje ir iliustruojami 19 paveiksle pateikiamais grafikais. Atlikus visų įverčių tipų rangavimą, pagal rezultatų tikslumą metodai išrikiuoti šia tvarka:

1. *BPRSLIM*
2. *MostPopular*
3. *BPRMF*
4. *MultiCoreBPRMF*
5. *WRMF*
6. *SoftMarginRankingMF*
7. *WeightedBPRMF*

8-9. BPRLinear

8-9. Zero

10. *Random*

Eksperimentų su *UserKNN* ir *ItemKNN* metodais atlikti nepavyko – kaip minėta, šie metodai, taikomi tokiam duomenų rinkiniui, reikalauja daug skaičiavimų resursų; kompiuterio 16 GB operatyviosios atminties nepakanka.

3.6 Eksperimentai su Lietuvoje veikiančio elektroninio knygyno duomenų rinkiniu

Tyrimui imami duomenys apie el. knygyno pirkėjų pirkimų istoriją nuo 2008 m. gegužės iki 2013 m. kovo mėn. Šiame rinkinyje taip pat pateikta informacija apie kiekvienos knygos autorių ir kategoriją, kuriai knyga priskirta. Informacijos, leidžiančios identifikuoti pirkėjus, juos susieti su konkrečiais asmenimis ir taip pažeisti jų privatumą, nėra. Taip pat rinkinyje pateikta informacija apie 41177 pardavimus (be pasikartojimų) ir jų datas. Tačiau dėl pernelyg mažo skaičiaus įvykusių pardavimų, lyginant su knygų ir vartotojų

skaičiumi, į pardavimo datą tolesniuose tyrimuose nebuvo atsižvelgta.

Nuo el. knygyno atidarymo buvo siūloma įsigyti 19329 skirtingas knygas, iš jų 12564 bent kartą nupirktos. Pirkimo istorijoje užfiksuoti 14197 vartotojai, iš jų 9930 įsigijo bent po vieną knygą. Tyrimams atlikti sukonstruojame duomenų rinkinį, kurį sudaro poros $\{(Vartotojo\ ID, Knygos\ ID)\}$. Šį duomenų rinkinį žymime „*Data*“.

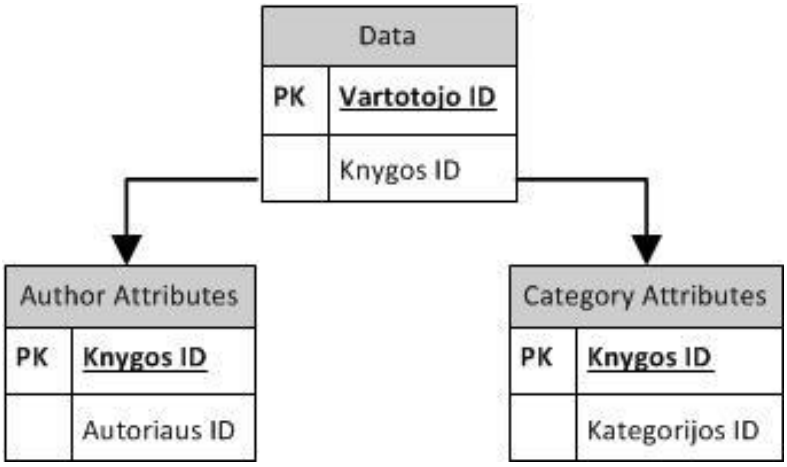
Kitus pateiktus duomenų rinkinius apie kategoriją, kuriai priskirta kiekviena knyga, ir apie autorių atitinkamai žymime „*Category_Attributes*“ ir „*Author_Attributes*“.

Duomenų rinkinių „*Data*“, „*Category_Attributes*“ ir „*Author_Attributes*“ struktūra ir tarpusavio ryšiai pateikiami 20 paveiksle.

Tolesniuose skaičiavimuose „*Data*“ duomenų rinkinys transformuojamas į matricą, turinčią 9930 eilučių ir 12564 stulpelius (tai yra tiek, kiek vartotojų nupirko bent vieną knygą ir kiek nupirkta skirtingų knygų), matrica užpildoma vienetais ir nuliais. Vienetas reiškia, kad i-tasis vartotojas nupirko j-tąją knygą. Šios „*Data*“ matricos užpildymas nenuliniais elementais (tankis) lygus $\frac{41177}{14197*19329} \times 100\ \% = 0{,}015\ \%$, tačiau kai atmetame nė karto nenupirktas knygas ir nė vienos knygos nenusipirkusius vartotojus, gauname $\frac{41177}{9930*12564} \times 100\ \% = 0{,}033\ \%$ užpildymas.

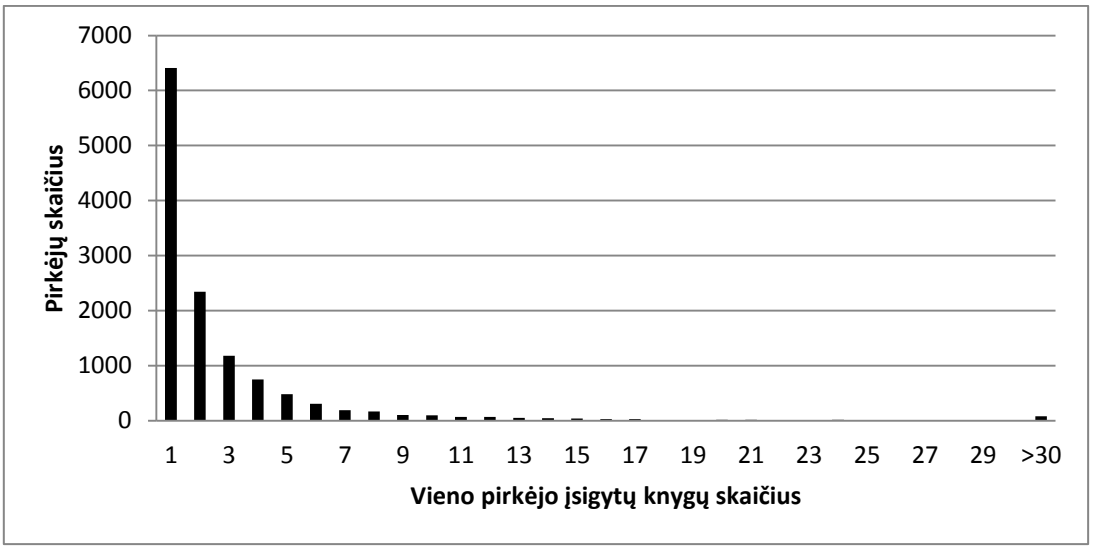
6 lentelė. *Elektroninio knygyno duomenų rinkinio struktūra*

Parametras	**Dydis**
Vartotojų, dėjusių knygas į krepšelius, skaičius	14197
Vartotojų, nupirkusių bent vieną knygą, skaičius	9930
Knygų skaičius sistemoje	19329
Bent kartą nupirktų knygų skaičius	12564
Knygų kategorijų skaičius	233
Knygų autorių skaičius	10058
Unikalių pardavimų skaičius	41177



20 paveikslas. *Tyrimui naudotų duomenų rinkinių struktūra ir jų tarpusavio ryšiai*

Pakankamai svarbus duomenų rinkinio „*Data*“ analizės aspektas yra pirkėjų pasiskirstymo pagal nupirktų knygų skaičių nustatymas. Šis pasiskirstymas iliustruojamas 21 paveiksle vaizduojamame grafike.

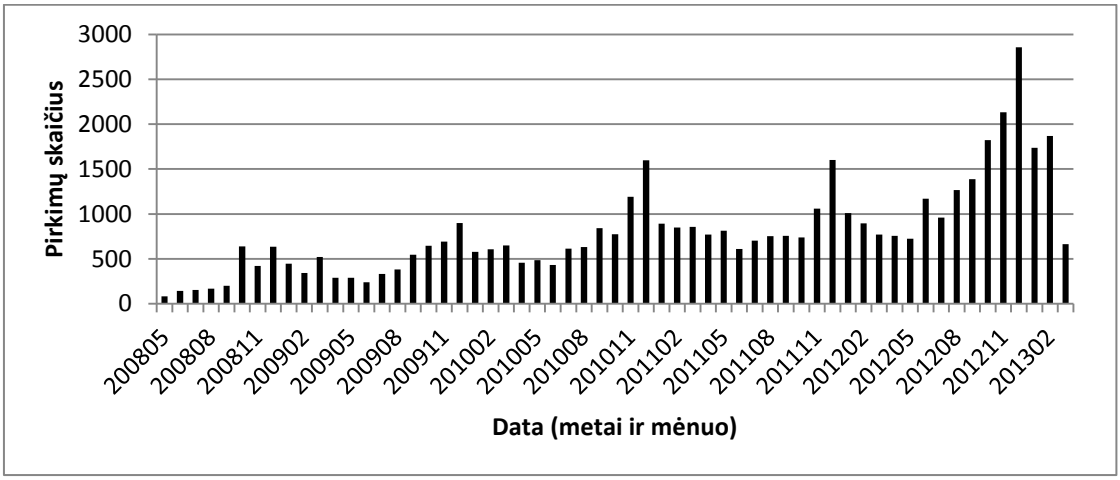


21 paveikslas. *Pirkėjų pasiskirstymas pagal nupirktų knygų skaičių*

Kaip matyti iš 21 paveikslo, 50,68 % pirkėjų įsigijo tik vieną knygą, o 88,29 % sistemos vartotojų yra nusipirkę mažiau kaip 5 knygas. Prognozės tokiems vartotojams nebūna labai tikslios, mat kyla tuščių įvertinimų ir naujų vartotojų problemos, detaliau aprašytos 2.4.5 poskyryje.

Pastebima ir anomalijų – keli vartotojai yra nusipirkę daugiau kaip 500 knygų. Šie vartotojai taip pat gali iškreipti rekomendacijų tikslumą.

Pasiskirstymas laike (angl. *Temporal Distribution*) parodo, kaip kito pirkimų skaičius laikui bėgant (22 paveikslas).

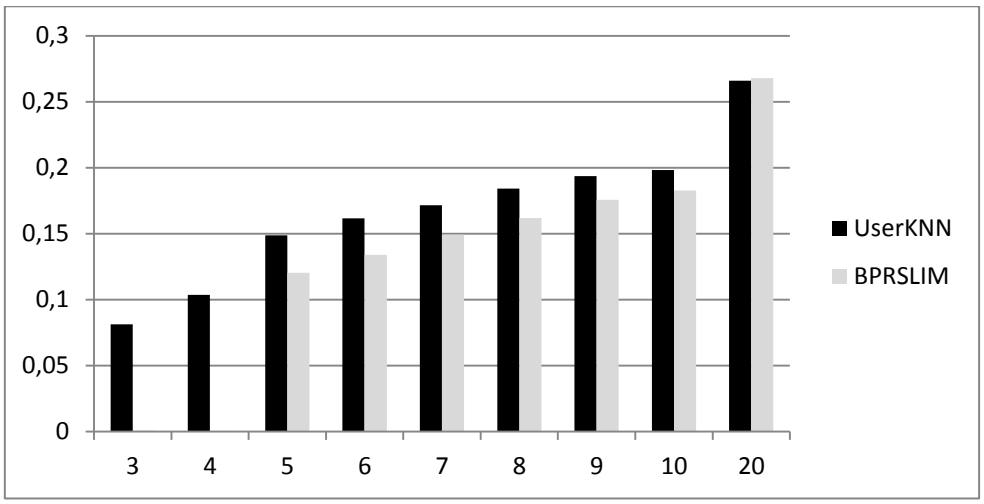


22 paveikslas. *Pirkimų skaičiaus pasiskirstymas laike*

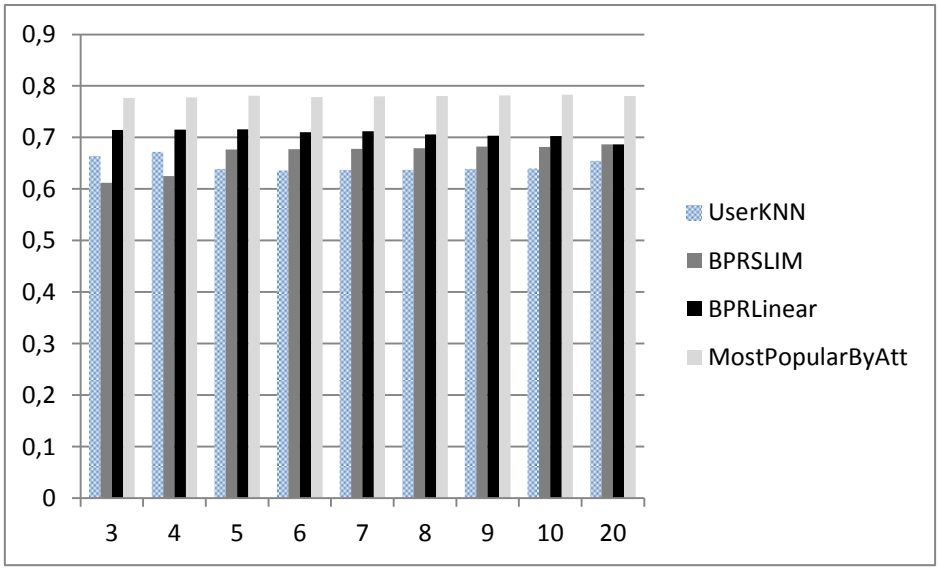
Iš 22 paveikslo galima nustatyti pardavimų didėjimo ir mažėjimo intervalus. Pastebėtina, kad didžiausias pardavimų skaičius fiksuojamas žiemos mėnesiais (ypač gruodį, kas galėtų būti siejama su kalėdinių dovanų pirkimu), o mažiausias pavasarį ir vasaros pradžios mėnesiais. Taip pat pastebėtina, kad nuo 2012 m. birželio fiksuojamas smarkus pardavimų padidėjimas, lyginant su tuo pačiu praėjusių metų laikotarpiu.

Siekiant įvertinti metodų efektyvumą, kai vartotojo įsigytų knygų skaičius kinta, eksperimentai atlikti ir taikant kryžminio patvirtinimo modelį. Kryžminiui patikrinimo modeliui sudaryti turimas duomenų rinkinys atsitiktinai sumaišomas ir padalijamas į K lygių dalių. Tada metodui mokyti nuosekliai imamos $K-1$ dalys, o kita dalis naudojama testuoti. Paskui vertinimas tęsiamas K kartų: imamos kitos $K-1$ dalys mokyti ir viena dalis testuoti (10 paveikslas).

23 paveiksle pateikiamas *UserKNN* ir *BPRSLIM* metodų *MAP* įverčio reikšmės didėjimas priklausomai nuo K reikšmės. Natūralu, kad didėjant vartotojo įsigytų knygų skaičiui, metodų rekomendacijos darosi tikslesnės.



23 paveikslas. *MAP įverčio didėjimas didėjant K reikšmei*

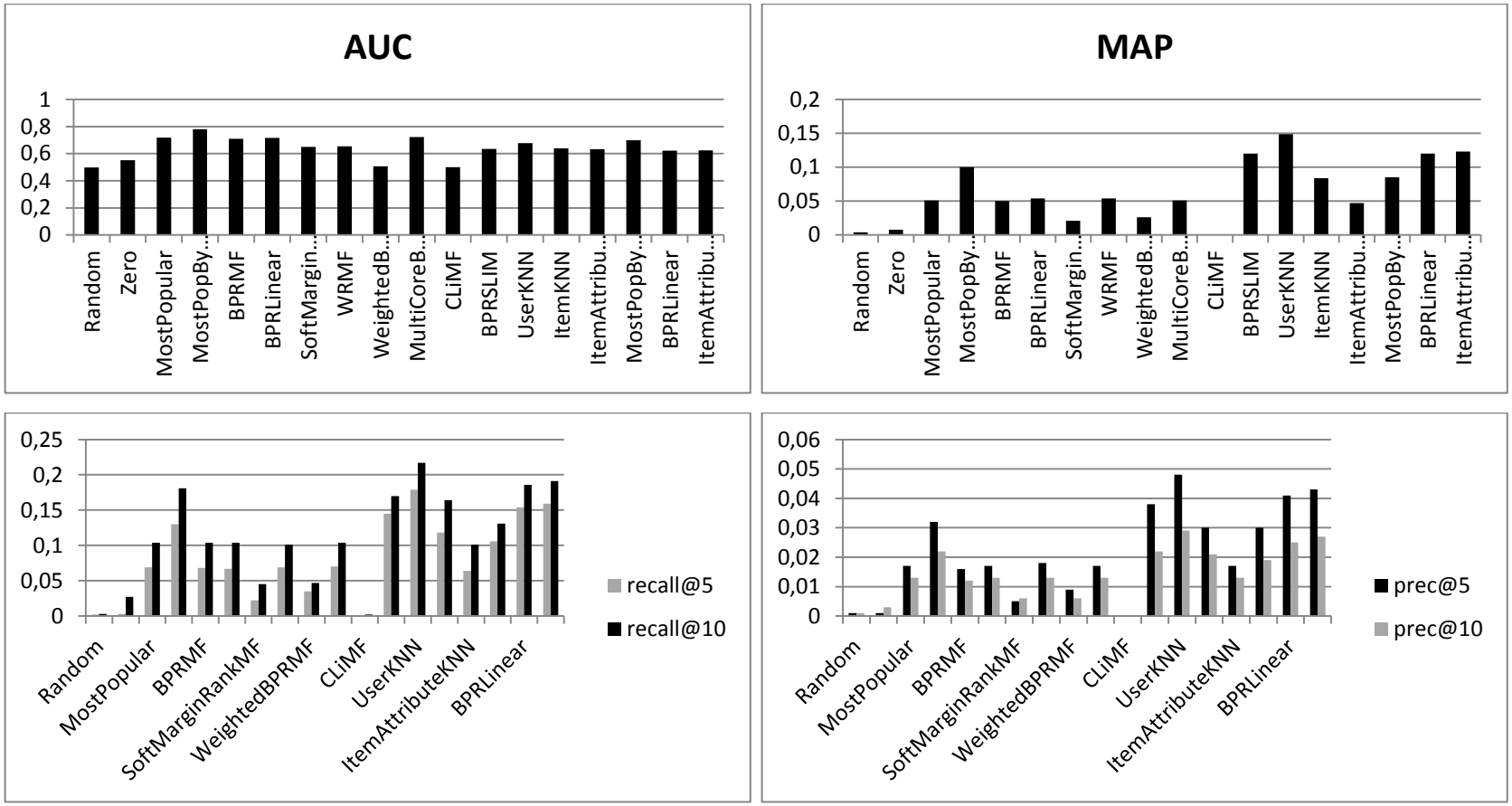


24 paveikslas. *AUC įverčio kaita didėjant K reikšmei*

25 paveiksle pateikiama *UserKNN*, *BPRSLIM*, *BPRLinear* ir *MostPopularByAttributes* metodų *AUC* įverčio reikšmės kaita, priklausanti nuo k reikšmės. Pastebėtina, kad šis RS įvertis dviejų metodų atvejais kinta nežymiai (*MostPopularByAttributes* – paklaidų ribose, *BPRLinear* – fiksuojamas nežymus sumažėjimas $K \in [10; 20]$ intervale), *BPRSLIM* – fiksuojamas žymesnis reikšmės padidėjimas $K \in [3; 5]$ intervale, o *UserKNN* varijuoja $K \in [3; 5]$ intervale, paskui nusistovi ties tam tikra reikšme ir po to vėl padidėja $K \in [10; 20]$ intervale.

7 lentelė. *Kryžminio patvirtinimo eksperimentų rezultatas ($K = 5$)*

Metodas	AUC	Vieta	prec@5	Vieta	prec@10	Vieta	MAP	Vieta	recall@5	Vieta	*	Vidurkis	Galutinė vieta
Random	0,499	18	0,001	17	0,001	17	0,004	17	0,002	17	Kitų įverčių rangai	13,18	**17**
Zero	0,552	15	0,001	16	0,003	16	0,008	16	0,003	16		12,27	**16**
MostPopular	0,719	3	0,017	10	0,013	11	0,051	10	0,069	10		7,18	**9**
MostPopular ByAttributes(C)	0,781	1	0,032	5	0,022	4	0,100	5	0,130	5		5,27	**3**
BPRMF	0,711	5	0,016	13	0,012	13	0,050	12	0,068	11		9,82	**12**
BPRLinear	0,716	4	0,017	11	0,013	10	0,054	8	0,067	12		9,36	**8**
SoftMargin RankingMF	0,650	9	0,005	15	0,006	15	0,021	15	0,022	15		12,18	**14**
WRMF	0,654	8	0,018	8	0,013	9	0,054	9	0,069	9		9,09	**10**
Weighted BPRMF	0,506	16	0,009	14	0,006	14	0,026	14	0,035	14		12,45	**15**
MultiCore BPRMF	0,722	2	0,017	12	0,013	12	0,051	11	0,070	8		8,73	**11**
CLiMF	0,501	17	0,000	18	0,000	18	0,001	18	0,001	18		15,27	**18**
BPRSLIM	0,635	11	0,038	4	0,022	5	0,120	3	0,145	4		7,64	**4**
UserKNN	0,677	7	0,048	1	0,029	1	0,149	1	0,179	1		5,27	**1**
ItemKNN	0,639	10	0,030	7	0,021	6	0,084	7	0,118	6		9,45	**7**
ItemAttribute KNN	0,633	12	0,017	9	0,013	8	0,047	13	0,064	13		13,00	**13**
MostPopular ByAttributes(A)	0,700	6	0,030	6	0,019	7	0,085	6	0,106	7		6,91	**6**
BPRLinear	0,623	14	0,041	3	0,025	3	0,120	4	0,154	3		6,91	**5**
ItemAttribute KNN	0,624	13	0,043	2	0,027	2	0,123	2	0,159	2		7,00	**2**



25 paveikslas. *Skirtingų įverčių reikšmės skirtingiems metodams*

Tai, kad pagal *MAP* ir *AUC* įverčius tais pačiais metodais gaunami skirtingi rezultatai, parodo įverčių rangavimo ir metodo rango vidurkio išvedimo būtinumą – tik apskaičiavus visus galimus įverčius, nustačius metodų rangus ir išvedus vidurkį galima tiksliau nustatyti metodo efektyvumą. *AUC*, *MAP*, *recall@5*, *recall@10* ir *prec@5*, *prec@10* įverčiai pagal metodus, kai $K = 5$, pateikiami **7** lentelėje ir iliustruojami 25 paveiksle pateikiamuose grafikuose.

8 lentelė. *Geriausiai ir blogiausiai veikiantys metodai skirtingų K reikšmių atveju*

K	**3**	**5**	**8**	**10**	**20**
1	*MPopByAtt (c)*	*UserKNN*	*UserKNN*	*UserKNN*	*MPopByAtt (c)*
2	*UserKNN*	*MPopByAtt (c)*	*MPopByAtt (c)*	*MPopByAtt (c)*	*UserKNN*
3	*MostPopular*	*MPopByAtt (a)*	*BPRSLIM*	*BPRSLIM*	*BPRSLIM*
4	*BRPLinear (a)*	*BRPLinear (a)*	*MPopByAtt (a)*	*MPopByAtt (a)*	*MPopByAtt (a)*
5	*MPopByAtt (a)*	*ItemAttributeKNN*	*BRPLinear (a)*	*ItemKNN*	*ItemKNN*
14	*Zero*	*Zero*	*Zero*	*ItemAttributeKNN*	*SoftMarginRankingMF*
15	*ItemAttributeKNN*	*WeightedBPRMF*	*WeightedBPRMF*	*Zero*	*ItemAttributeKNN*
16	*WeightedBPRMF*	*ItemAttributeKNN*	*ItemAttributeKNN*	*SoftMarginRankingMF*	*Random*
17	*Random*	*Random*	*Random*	*Random*	*Zero*
18	*CLiMF*	*CLiMF*	*CLiMF*	*CLiMF*	*CLiMF*

Atlikus visų įverčių tipų rangavimą, 8 lentelėje pateikiami geriausiai veikiantys (pirmi penki) metodai penkiais skirtingais kryžminės patikros atvejais. Blogiausių penkių metodų numeriai – 14–18.

Iš 8 lentelės galima pastebėti, kad *UserKNN* ir *MostPopularByAttributes (category)* metodai pirmauja nepriklausomai nuo vartotojo įsigytų knygų skaičiaus. Jei įsigytų knygų skaičius didesnis, pakankamai efektyvūs yra *BPRSLIM* ir *MostPopularByAttributes*, o naujiems vartotojams, neturintiems

ilgos pirkimų istorijos, efektyvūs yra ir *MostPopular* bei *BPRLinear (author)* metodai.

Taip pat pastebėtinas gana įdomus dalykas – vienas metodas (*CLiMF*) šiame duomenų rinkinyje veikia prasčiau nei atsitiktinių siūlymų metodas (*Random*). Tikėtina, kad šis metodas efektyvus tik su labai specifiniais duomenų rinkiniais. Be to, taikant du metodus (*ItemAttributeKNN* ir *WeightedBPRM)* intervale $K \in [3; 8]$ gaunami prastesni rezultatai nei taikant dar vieną vertinimo atskaitos tašką – *Zero* metodą. Kai $K = 10$, prastesni rezultatai gaunami ir su *SoftMarginRankingMF*. To priežastis – mažas vartotojų įverčių produktams matricos užpildymo tankis. Kai pirktų knygų skaičius didelis ($K \approx 20$), visais tirtais metodais (išskyrus *CLiMF*) gaunami geresni rezultatai nei *Random* ir *Zero* atskaitos taškais.

3.7 Trečiojo skyriaus apibendrinimas ir išvados

Iš šiame skyriuje atliktos analizės matyti, kad nėra universalaus bet kokiam duomenų rinkiniui tinkančio rekomendavimo metodo. Eksperimentiškai tirtuose duomenų rinkiniuose geriausi rezultatai gauti skirtingais metodais. Pavyzdžiui, tiriant *Jester* duomenų rinkinį, tinkamiausias metodas – *MatrixFactorization*; *CoClustering* metodas – tik šeštas. O štai *Movielens* duomenų rinkinio tyrime tinkamiausias metodas buvo *SVDPlusPlus*, tačiau *CoClustering* metodas – trečias.

RS efektyvumui įvertinti taikomi efektyvumo matai nevisiškai atspindi realią padėtį, todėl vertinant RS metodų efektyvumą reikia atsižvelgti į šių matų visumą.

Pastebėta, kad esant mažam duomenų rinkinių užpildymui, geri rezultatai gaunami metodais, paremtais populiariausių prekių išskyrimu (žr. 2.4.3 ir 2.4.4). Tai nėra netikėta, nes tokiuose rinkiniuose tuščių įvertinimų problema daro didelę įtaką kitų metodų rezultatams, o populiariausių prekių išskyrimu paremti metodai šią problemą eliminuoja.

Tyrimai parodė, kad mokslinėje literatūroje universaliais laikomi *UserKNN* ir *ItemKNN* (žr. 2.4.5) metodai yra tinkami tik mažiems duomenų rinkiniams. Eksperimentus šiais metodais pavyko atlikti tik su nedideliu Lietuvoje veikiančio elektroninio knygyno duomenų rinkiniu, o eksperimentams su didesniais duomenų rinkiniais nebeužtekdavo tyrimams naudoto kompiuterio resursų. Tai pagrindžia būtinybę tobulinti šiuos metodus ir mažinti metodų reikalaujamų resursų kiekį. Šis tobulinimas būtų ne efektyvesnė esančių metodų kompiuterinė realizacija, o metodų modifikavimas atsižvelgiant į siauresnes sprendžiamų uždavinių klases, tai yra metodų specializacijos didinimas.

4

Naujas rekomendavimo metodas ir jo taikymo galimybių tyrimai

Šiame skyriuje pristatomas ir eksperimentiškai tiriamas metodas, įvertinantis vartotojų grupių specifiką kuriant rekomendacijas. Pagrindiniai rezultatai paskelbti autoriaus moksliniuose straipsniuose [2A] [6A] ir pristatyti trijose mokslinėse konferencijose.

4.1 Aktualumas ir idėja

Geriau įvertinti interneto vartotojų poreikiams ir rekomenduoti tinkamiems produktams ar paslaugoms kuriami rekomendavimo metodai ir juos taikančios sistemos. Šios sistemos dažniausios elektroninėje komercijoje ir socialiniuose tinkluose, turinčiuose daug vartotojų, vertinančių ir komentuojančių įvairius produktus. Pagal šiuos vertinimus rekomendacinės sistemos dažniausiai ir generuoja naujas rekomendacijas.

1 skyriuje apžvelgta mokslinė literatūra, susijusi su minėtomis problemomis ir uždaviniais, atskleidė, kad rekomendacinių sistemų ir metodų įvairovė yra didelė, tačiau, kita vertus, ši įvairovė rodo ir tai, kad nėra universalaus, t. y. bet kokiam duomenų rinkiniui tinkančio, rekomendavimo metodo. Geriausi rezultatai gauti taikant skirtingus metodus eksperimentiškai

tirtiems duomenų rinkiniams.

Mokslinėje literatūroje universaliais laikomi bendrojo filtravimo tipo metodai *UserKNN* ir *ItemKNN* (žr. 2.4.5). Jie remiasi k artimiausių kaimynų (*KNN*) idėja. 3 skyriaus tyrimai parodė, kad šie metodai reikalauja daug skaičiavimo resursų, ypač kai sukaupta duomenų apie daugybę vartotojų ir produktų. Tad eksperimentus šiais metodais pavyko atlikti tik su nedideliu Lietuvoje veikiančio elektroninio knygyno duomenų rinkiniu, o eksperimentams su didesniais duomenų rinkiniais nebeužtekdavo tyrimams naudoto kompiuterio resursų. Tai pagrindžia būtinybę tobulinti šiuos metodus reikalaujamų resursų kiekio mažinimo linkme. Toks tobulinimas būtų ne efektyvesnė esančių metodų kompiuterinė realizacija, o metodų modifikavimas atsižvelgiant į siauresnes sprendžiamų uždavinių klases, tai yra metodų specializacijos didinimas.

Šiais metodais reitingai arba reikalingos prekės prognozuojami remiantis k panašiausių vartotojų ankstesniais įverčiais. Šių vartotojų radimas yra sudėtingas ir skaičiavimams imlus uždavinys. Naujo metodo tikslas – sugrupuoti panašius vartotojus į tam tikrą skaičių klasterių ir ieškoti klasterio, kurio vartotojai panašiausi į naują vartotoją, kuriam reikia sugeneruoti rekomendaciją. Ieškoma ne k panašiausių vartotojų, o visos jų grupės, kurios dydis iš anksto nėra žinomas.

Šiame skyriuje pateiktas rekomendavimo metodas, įvertinantis vartotojų grupių elgsenos specifiką, kai vartotojų įverčių produktams matrica yra tankiai užpildyta. Dažniausiai šiuose duomenų rinkiniuose yra didelis vartotojų ir santykinai mažas produktų skaičius. Tokio tipo duomenų rinkiniai paplitę specializuotose elektroninėse parduotuvėse, taip pat įvairiuose specifiniuose interneto kataloguose.

4.2 Siūlomas metodas

Tarkime, kad turime m vartotojų $a_1, \dots, a_m$, n produktų $b_1, \dots, b_n$ ir vartotojų įverčių produktams matricą:

$$V = \{V_{ij}, i = \overline{1,m}, j = \overline{1,n}\}. \quad (4.1)$$

Tarkime, vartotojas produktus vertina pagal sveikųjų skaičių skalę $\{u_{min}, \dots, u_{max}\}$, sudarytą iš n_u elementų. Reikšmė V_{ij} rodo i-tojo vartotojo pateiktą įvertį j-tajam produktui. Jeigu i-tasis vartotojas nėra įvertinęs j-tojo produkto, tai $V_{ij} \notin \{u_{min}, \dots, u_{max}\}$.

Manykime, kad matrica V yra visa užpildyta įverčiais, tai yra ją sudaro duomenys apie m vartotojų, įvertinusių visus produktus. Tokią matricą galima gauti, jei iš turimos matricos išrenkami vartotojai, įvertinę 100 % produktų, ir suformuojama etaloninė vartotojų ir jų įverčių produktams aibė.

Taigi šis metodas skirtas generuoti rekomendacijoms tuo atveju, kai įverčių tankis duomenų rinkinyje yra pakankamai didelis. Tokie duomenų rinkiniai esti tada, kai turime didelį vartotojų ir mažą produktų skaičių. Moksliniams tyrimams prieinamų tokių duomenų rinkinių pavyzdžiai – *Jester I*, *Jester II* ir *Jester III*.

Pagal matricą V galima klasterizuoti vartotojus į panašių vartotojų klasterius $C_1, C_2, \dots, C_k$, kuriuos sudaro skirtingas vartotojų skaičius:

$C_1 = \{a_1^1, a_2^1, \dots, a_{m_1}^1\}$ su m_1 vartotojų,

$C_2 = \{a_1^2, a_2^2, \dots, a_{m_2}^2\}$ su m_2 vartotojų,

(...)

$C_k = \{a_1^k, a_2^k, \dots, a_k^k\}$ su m_k vartotojų,

$m = \sum_{l=1}^{k} m_l$.

Čia a_i^l yra l-tojo klasterio i-tasis vartotojas,

$$A = C_1 \cup C_2 \cup \dots \cup C_k;\ C_1 \cap C_2 \cap \dots \cap C_k = \emptyset.$$

Kiekvienas klasteris $C_l, l = \overline{1,k}$ turi centrą:

$$X_l = (x_1^l, \dots, x_n^l),\ x_j^l = \frac{\sum_{a_i \in C_l} V_{ij}}{m_l^j}, \quad (4.2)$$

čia m_l^j – l-tojo klasterio C_l j-tojo produkto b_i vertinimų skaičius. Pastebėsime, kad iš viso j-tąjį produktą vertino

$$m^j = \sum_{l=1}^{k} m_l^j \quad (4.3)$$

vartotojų.

Kai į sistemą užsiregistruoja naujas vartotojas a_N, jam atsitiktinai pateikiami keli (s) produktai $\{b_{N_1}, b_{N_2}, \dots, b_{N_s}\}$ iš $\{b_1, \dots, b_n\}$, kad įvertintų. Čia N_i – produkto eilės numeris tarp 1 ir n visų produktų sąraše; $s < n$, o $V_N = (V_{NN_1}, \dots., V_{NN_s})$ – naujo vartotojo įverčiai.

Kai gaunami vartotojo įverčiai s produktams, kiekviename klasteryje C_l, $l = \overline{1, k_a}$, tik pagal produktus $b_{N_1}, \dots, b_{N_s}$ išskiriami klasterių centrai:

$$X_l^N = \left(x_{N_1}^l, \dots, x_{N_s}^l\right) \tag{4.4}$$

ir apskaičiuojami atstumai

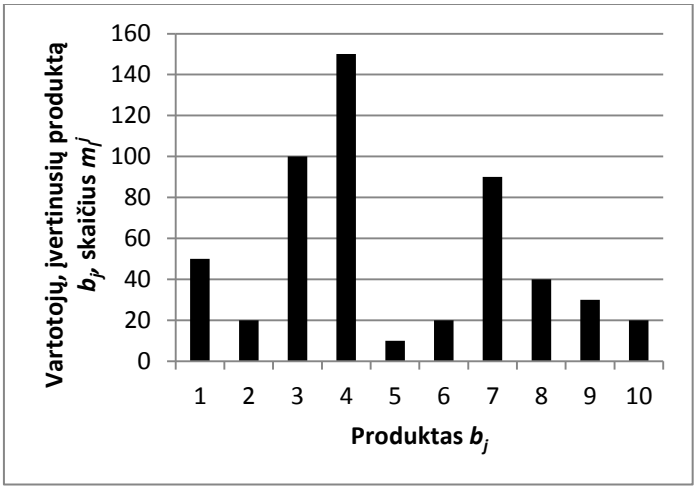
$$\rho(\ \ _N, X_l^N) = \sqrt{\sum_{i=1}^{s}(V_{NN_i} - x_{N_i}^l)^2}, l = \overline{1, k}. \tag{4.5}$$

Vartotojas a_N priklauso prie to klasterio, kur skirtumas tarp vartotojo įverčių ir klasterio centro, apskaičiuoto pagal formulę (4.4), yra mažiausias, tai yra $a_N \in C_{l^*}$, kur

$$l^* = arg \min_{l=\overline{1,k}} \rho(V_N, X_l^N). \tag{4.6}$$

Po šio priskyrimo vartotojui a_N galima siūlyti geriausiai klasterio C_{l*} vartotojų įvertintus produktus.

Kiekvienas vartotojų klasteris C_l turi produktų vartojimo pasiskirstymą, tai yra produktą b_j klasteryje C_l įvertino m_l^j vartotojų (26 paveikslas).



26 paveikslas. *Produktų vartojimo pasiskirstymo klasteryje C_l pavyzdys*

Kiekvienas produktas klasteryje C_l turi įverčių pasiskirstymą, rodantį,

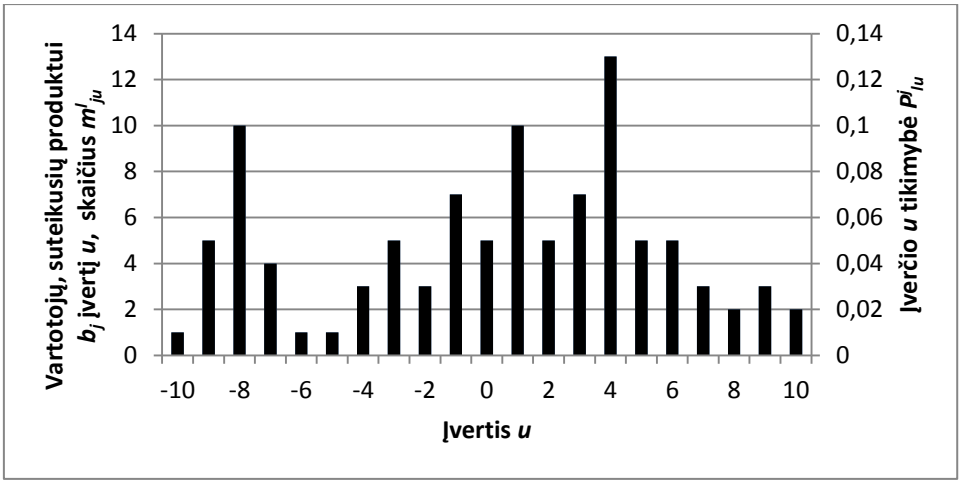
kiek klasterio C_l vartotojų suteikė produktui b_j įvertį $u \in \{u_{min}, \dots, u_{max}\}$ (27 paveikslas). Stulpelio aukštis m_{lu}^j rodo, kiek klasterio C_l vartotojų suteikė produktui b_j įvertį $u \in \{u_{min}, \dots, u_{max}\}$.

Pastebėsime, kad $\sum_{u=u_{min}}^{u_{max}} m_{lu}^j = m_l^j$.

Žinant produkto įverčių pasiskirstymą klasteryje (tokio pasiskirstymo pavyzdys pateikiamas 27 paveiksle), galima apibrėžti įverčių pasiskirstymo tankio funkciją. Skalė dešinėje – tikimybė, kad klasterio C_l vartotojai suteiks produktui b_j įvertį u:

$$P_{lu}^j = P\left(V_{ij} = u, jei\ a_i \in C_l\right) = \frac{m_{lu}^j}{m_l^j}. \tag{4.7}$$

Pastebėsime, kad $\sum_{u=u_{min}}^{u_{max}} P_{lu}^j = 1$.



27 paveikslas. *Pavyzdinis produkto įverčių pasiskirstymas klasteryje* C_l

Pagal (4.8) formulę galime apskaičiuoti vidutinę įverčio reikšmę, tai yra kokį vidutinį įvertį suteiks C_l klasterio vartotojai produktui b_j:

$$\bar{V}_l^{\,j} = \frac{1}{u_{max} - u_{min} + 1} \sum_{u=u_{min}}^{u_{max}} P_{lu}^j\, u. \tag{4.8}$$

Naujam vartotojui pateikiamas tas produktas, kuris klasteryje C_l turi didžiausią vidutinį įvertį. Jei produktas jau buvo pateiktas vartotojui, teikiamas produktas su didžiausiu mažesniu įverčiu ir t. t.

Kai naujas vartotojas įvertina pateiktą produktą, visas jo vertintų

produktų skaičius padidėja, tai yra $s = s + 1$.

Tolesni skaičiavimai kartojami nuo (4.4) formulės, pagal naujo vartotojo įverčius s produktams išskiriami kiekvieno klasterio centrai.

Kai naujas vartotojas baigia darbą (pasinaudoja siūlomais ar pasirinktais produktais), jo vertinimų rezultatais papildoma matrica V, tai yra $m = m + 1$.

4.3 Eksperimentinis siūlomo metodo tyrimas

Pateikiamas rekomendacijų kūrimo metodas grindžiamas klasterizavimu (panašių vartotojų grupių išskyrimu). Norint gauti optimalias rekomendacijas, reikia nustatyti tinkamą klasterių skaičių k. Eksperimentinis tyrimas turėtų parodyti, koks klasterių skaičius yra optimalus. Taip pat šis tyrimas turėtų atskleisti, ar vartotojų grupavimas yra tinkamas būdas kurti rekomendacijoms naujiems vartotojams.

Iš metodo veikimo principų matyti, kad vieno klasterio atveju ($k = 1$) daroma prielaida, jog visi vartotojai – panašūs. Tad tyrimo be vartotojų klasterizavimo rezultatai yra atskaitos taškas vertinti eksperimentų, kuriuose vykdomas panašių vartotojų grupavimas, rezultatams. Siūlomas metodas, kaip ir bendrojo filtravimo, taiko panašių vartotojų atranką, tik šiuo atveju atranka yra klasterizavimas.

4.3.1 Eksperimentinio tyrimo metodika

1. Eksperimentai atlikti su *Jester* I duomenų rinkiniu (*http://goldberg.berkeley.edu/jester-data/jester-data-1.zip*). Šiame duomenų rinkinyje sukaupta 24983 vartotojų, įvertinusių bent 36 produktus, įverčiai produktams. Duomenų rinkinys pateikiamas kaip matrica: pirmasis matricos stulpelis rodo, kiek produktų konkretus vartotojas yra įvertinęs, o kiti stulpeliai – vartotojų vertinimus produktams intervale $[-10; 10]$. Nepateikti įverčiai šiame duomenų rinkinyje užpildyti reikšme 99, todėl gali būti lengvai atskiriami. Šis duomenų rinkinys išskirtinis dėl pakankamai didelio įvertinimų tankio (kiekvienas vartotojas yra įvertinęs santykinai didelį produktų skaičių) –

įvertinimų tankis siekia apie 75 %, tai yra kiekvienas vartotojas yra įvertinęs vidutiniškai 75 produktus iš 100. Kadangi siūlomo metodo specifika lemia, kad jo veikimas turėtų būti efektyviausias tuose duomenų rinkiniuose, kurių vertinimų tankis pakankamai didelis, šis duomenų rinkinys tinka eksperimentams.

2. Siekiant objektyviai patikrinti metodo efektyvumą, duomenų aibėje atrenkami vartotojai, įvertinę visus produktus. Šiame duomenų rinkinyje išskirti 7200 vartotojai, pateikę įverčius visiems 100 produktų.

3. Ši 7200 vartotojų duomenų aibė dalijama į du poaibius:

a) bazinis poaibis, sudarytas iš m vartotojų (šio eksperimento atveju $m = 6200$);

b) validavimo poaibis, sudarytas iš m_v vartotojų (šio eksperimento atveju $m_v = 1000$).

4. Validavimo poaibio vartotojai ankstesnio skyriaus terminais suprantami kaip nauji vartotojai, jiems generuojamos rekomendacijos, o jau pateikti šių vartotojų vertinimai leidžia nustatyti vartotojo klasterį ir siūlomo metodo efektyvumą. Pagal s naujo vartotojo įverčius nustatomas šio vartotojo priskyrimas prie kurio nors klasterio, o pagal rekomenduoto produkto vertinimą – rekomendacijos tikslumas.

5. Siekiant gauti objektyvius rezultatus, skaičiavimai, vykdant s produktų parinkimą vartotojo elgsenai vertinti, atlikti daug kartų, ir iš rezultatų išvestas vidurkis. Tyrimo metu su kiekvienu validavimo aibės vartotoju, esant fiksuotai s reikšmei, atlikta 100 eksperimentų, pateikti vis kiti produktų rinkiniai $\{b_{N_1}, \dots, b_{N_s}\}$. Iš rezultatų išvestas vidurkis. Pažymėkime $\bar{u}_{s+1}$ vidutinį rekomenduojamo $(s+1)$-mojo produkto įvertį, kai žinomi kitų s produktų vertinimo rezultatai.

4.3.2 Eksperimentinio tyrimo rezultatai

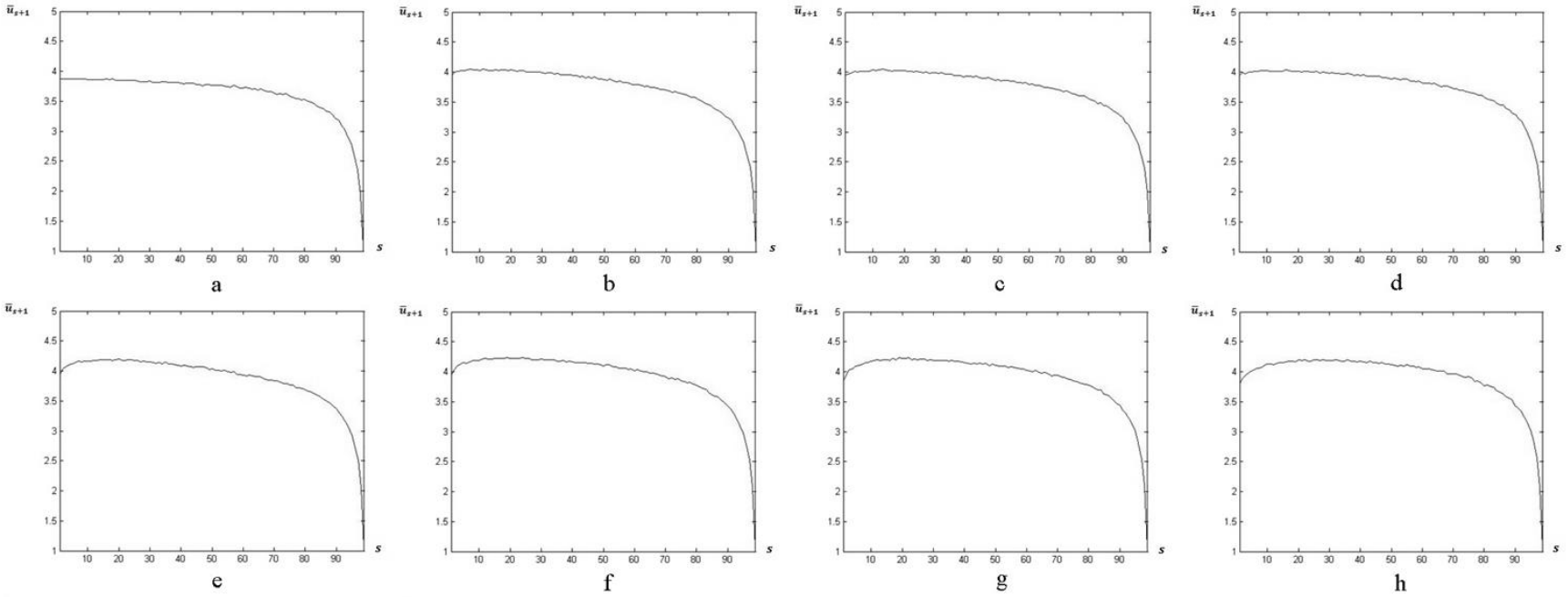
Gauti eksperimento rezultatai pateikiami 28 ir 29 paveiksluose ir 10 lentelėje. 28 paveiksle pateikiama rekomenduojamo $(s+1)$-mojo produkto

įverčio $\bar{u}_{s+1}$ priklausomybė nuo vartotojų klasterių skaičiaus k ir jau įvertintų produktų kiekio s. Iš šių grafikų galima matyti, kad visais atvejais intervale $[1;\ s(\bar{V}_{max})]$ vyksta $\bar{V}$ reikšmės didėjimas. Jei vartotojų klasterių skaičius k didesnis, šis augimas taip pat didesnis, tačiau optimalus rezultatas (didžiausias $\bar{V}_{max}$) gaunamas, kai klasterių skaičius ne pats didžiausias ir ne pats mažiausias. Intervale $\left[s(\bar{V}_{\max});\ 99\right]$ matyti $\bar{V}$ reikšmės mažėjimas, o taške $s = 99$ įverčio $\bar{V}$ reikšmė turėtų sumažėti iki visų duomenų rinkinio vartotojų vertinimų visiems produktams vidurkio. Kai s tampa lygus $n - 1$, įverčio $\bar{u}_{s+1} = \bar{u}_{n-1}$ reikšmė sumažėja iki visų vartotojų vertinimų visiems produktams vidurkio $\bar{V}$. Šio eksperimentams naudoto duomenų rinkinio vertinimų vidurkis $\bar{V} = 1{,}066$.

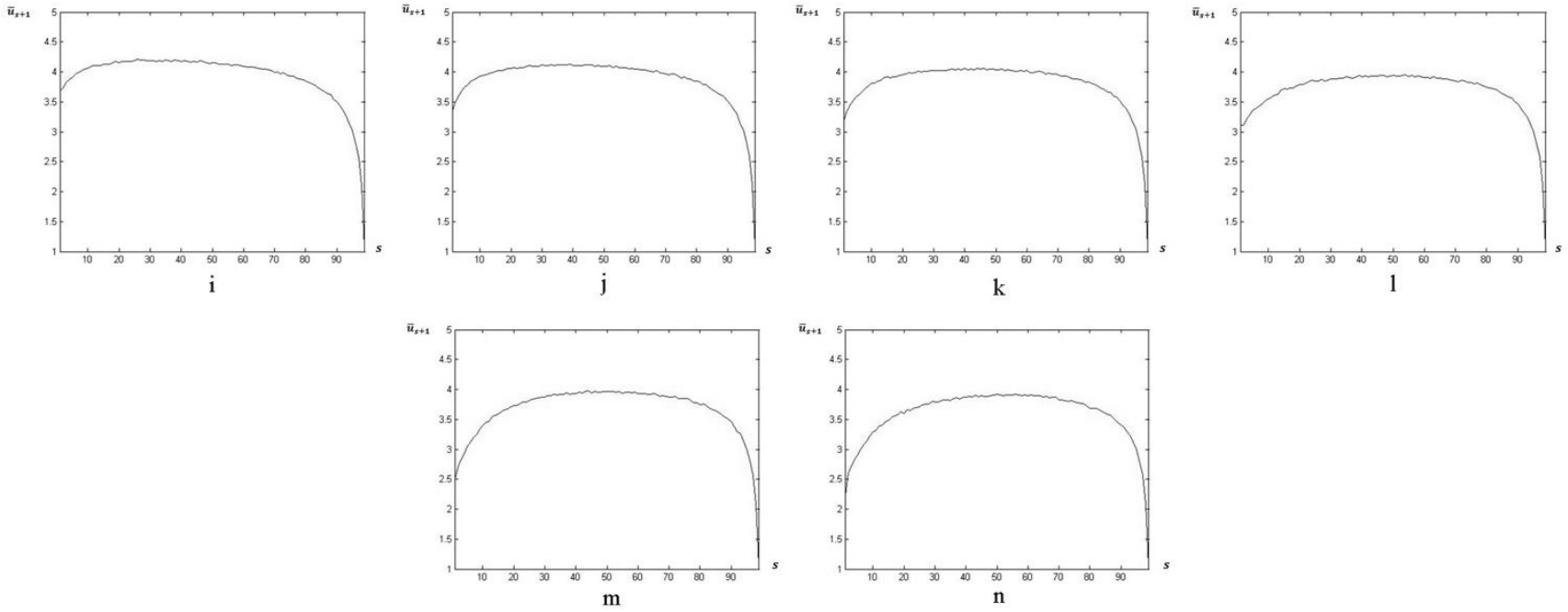
10 lentelėje pateikiami detalūs eksperimentų rezultatai, o .$\bar{u}_2, \bar{u}_{max}, \bar{u}_{n-1}$ ir $\bar{u}_{vid}$ reikšmių priklausomybė nuo klasterių skaičiaus k grafiškai vaizduojama 29 paveiksle.

Iš 10 lentelėje pateiktų rezultatų pastebėtina, kad mažiausia $\bar{u}_2$ reikšmė (tai yra $\bar{u}_{s+1}$ reikšmė, kai $s = 1$) gaunama, kai klasterių skaičius didžiausias (šiuo atveju $k = 200$). Tikėtina, kad jei klasterių skaičius vis didėtų, ši reikšmė mažėtų. Didžiausios $\bar{u}_{s+1}$ (kai $s = 1$) reikšmės gaunamos, kai klasterių skaičius $k \in [2;\ 10]$. Tai aiškiai matyti 10 lentelėje pateikiamoje suvestinėje. Jei klasterių skaičius nedidelis, pakanka mažesnio skaičiaus vartotojo vertinimų, kad vartotoją galima būtų priskirti prie optimalaus klasterio, todėl didėja tikimybė, kad jau po pirmojo naujo vartotojo pateikto produkto įverčio šiam vartotojui bus parinktas tinkamas klasteris.

Žinoma, turint nedidelį naujo vartotojo pateiktų įverčių produktams skaičių, negalima visiškai pasikliauti, kad vartotojo priskyrimas prie tam tikro klasterio yra teisingas, tačiau kai s reikšmė pasiekia tam tikrą dydį, klasteris, prie kurio priskiriamas vartotojas, nebekinta (žr. 9 lentelėje pateikiamą atsitiktinių vartotojų pavyzdį). Tai nereiškia, kad nedidelės s reikšmės yra netinkamos – klasteriuose irgi yra labai panašių įverčių.



28 paveikslas. *Rekomenduojamo* $(s+1)$*-mojo produkto įverčio* $\bar{u}_{s+1}$ *priklausomybė nuo vartotojų klasterių skaičiaus* k *ir jau įvertintų produktų kiekio* s*: a)* $k=1$*; b)* $k=2$*; c)* $k=3$*; d)* $k=5$*; e)* $k=7$*; f)* $k=10$*; g)* $k=15$*; h)* $k=20$



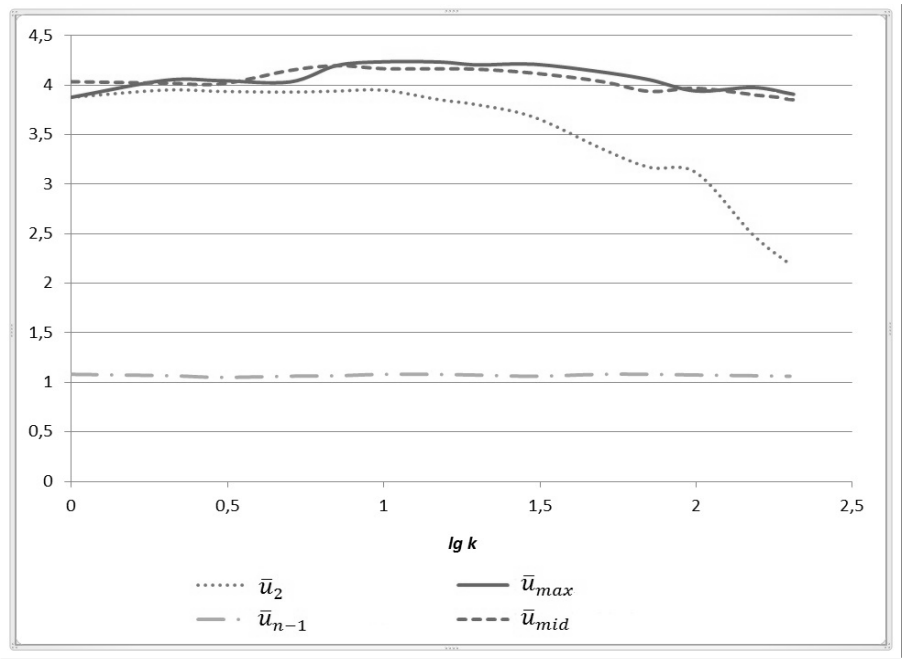
28 paveikslas. *Rekomenduojamo* $(s+1)$*-mojo produkto įverčio* $\bar{u}_{s+1}$ *priklausomybė nuo vartotojų klasterių skaičiaus* k *ir jau įvertintų produktų kiekio* s*: a)* $k=1$*; b)* $k=2$*; c)* $k=3$*; d)* $k=5$*; e)* $k=7$*; f)* $k=10$*; g)* $k=15$*; h)* $k=20$*; i)* $k=30$*; j)* $k=50$*; k)* $k=70$*; l)* $k=100$*; m)* $k=150$*; n)* $k=200$

9 lentelė. *Naujo vartotojo a_N priskyrimo prie klasterio kaita didėjant s reikšmei*

s / a_N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1	1	4	3	1	4	4	4	4	4	1	4	2	4	4	4	4	1	1	2	1	4	4	4	4
2	1	2	4	4	1	4	4	4	4	2	4	4	4	2	1	2	4	4	4	4	4	4	4	4	4
3	1	1	5	5	5	5	2	1	4	5	1	5	5	1	5	5	5	5	5	5	5	5	5	5	5
4	3	5	1	1	1	1	1	1	4	1	4	1	4	4	1	1	1	1	1	1	1	1	1	1	1
5	1	4	3	2	2	4	4	4	1	4	3	1	4	4	4	4	3	4	4	2	2	2	2	2	2

10 lentelė. *Siūlomo metodo eksperimentinio tyrimo rezultatai*

Klasterių skaičius k	**Pradinė reikšmė $\bar{u}_2$**	**Susilyginimo taškas s^***	**Didžiausia reikšmė $\bar{u}_{max}$**	**$\bar{u}_{max}$ reikšmės taškas s_{opt}**	**$\bar{u}_{max} - \bar{u}_2$**	**Galutinė reikšmė $\bar{u}_{n-1}$**	**Intervalo $[1; s^*]$ vidurio reikšmė**
1	3,877	-	3,877	1	0,000	1,081	-
2	3,949	36	4,048	11	0,099	1,068	4,034
3	3,936	38	4,045	13	0,109	1,051	4,020
5	3,929	44	4,032	16	0,102	1,062	4,012
7	3,939	60	4,198	20	0,259	1,067	4,148
10	3,948	68	4,235	24	0,287	1,081	4,196
15	3,851	75	4,232	22	0,387	1,080	4,165
20	3,801	79	4,204	28	0,403	1,073	4,164
30	3,677	86	4,210	26	0,534	1,062	4,160
50	3,355	92	4,131	39	0,796	1,081	4,120
70	3,169	94	4,054	46	0,884	1,081	4,035
100	3,116	94	3,940	50	0,825	1,074	3,936
150	2,495	97	3,976	44	1,481	1,067	3,968
200	2,190	98	3,916	56	1,726	1,062	3,904



29 paveikslas. *Pradinės ($\bar{u}_2$), didžiausios ($\bar{u}_{max}$), galutinės ($\bar{u}_{n-1}$) ir intervalo $[1; s^*]$ vidurio ($\bar{u}_{vid}$) įverčių reikšmės priklausomybė nuo klasterių skaičiaus k*

Iš 28 paveiksle pateikiamų grafikų matyti, kad įverčių $\bar{u}_{s+1}$ reikšmė didėja tol, kol pasiekia didžiausią, o tada reguliariai mažėja ir ties $s = n - 1$ pasiekia visų vartotojų vertinimų visiems produktams vidurkį $\bar{V}$. Pažymėkime s^* tokią įvertintų produktų kiekio reikšmę, kai $\bar{u}_2 = \bar{u}_{s*+1}$. s^* rodo, kada mažėdama produktų įverčio reikšmė susilygina su pirmąja ($\bar{u}_2$). Ši reikšmė yra intervalo, kuriame dėl klasterizavimo gaunami geri rezultatai, pabaiga. Štai šis intervalas: $[1; s^*]$. Iš 10 lentelės galima pastebėti, kad jei klasterių skaičius didėja, šis intervalas plečiasi, o kai klasterių skaičius yra 150 ar daugiau, visiškai priartėja prie maksimalios galimos s reikšmės $(n - 1)$. Nagrinėto pavyzdžio atveju: kai $k = 200$, $s^* = 98$.

Maksimali įverčių $\bar{u}_{s+1}$ reikšmė $\bar{u}_{max}$ taip pat kinta priklausomai nuo klasterių skaičiaus k. Nagrinėjamu atveju, kai $k = 1$, $\bar{u}_{max} = \bar{u}_2 = 3.877$, o kai $k = 2$ (tai yra kai taikomas klasterizavimas esant minimaliam klasterių skaičiui) $\bar{u}_{max} = 4{,}048$. Vadinasi, vos tik vartotojai pradedami skirstyti į klasterius, gaunamas 4,5 % $\bar{u}_{max}$ didėjimas. Didžiausios $\bar{u}_{max}$ reikšmė pasiekiama, kai $k \in [7; 30]$, o kai $k = 10$, ši reikšmė esti didžiausia ($\bar{u}_{max} =$

4,235). Lyginant su $\bar{u}_{max}$, kai $k = 1$, šiuo atveju gaunamas 9,2 % didėjimas.

$\bar{u}_{max}$ reikšmės taškas s_{opt} turi tendenciją didėti, kai didėja klasterių skaičius k. Kai $k = 10$, $\bar{u}_{max}$ yra didžiausia. Čia optimali reikšmė $s_{opt} = 24$. Pastebėtina, kad kai k priklauso intervalui [7; 30], tai yra kai gaunamos didžiausios $\bar{u}_{max}$ reikšmės, s_{opt} reikšmė yra intervale [20; 28]. Vadinasi, šiuo atveju geriausi rezultatai gaunami tada, kai naujo vartotojo vertinimų produktams istorija apima apie 25 % visų duomenų rinkinio produktų.

Maksimalios ($\bar{u}_{max}$) ir pradinės (u_2) reikšmių skirtumas didėja, kai didėja klasterių skaičius k. Kai $k = 1$, $\bar{u}_{max} = \bar{u}_2$, todėl $\bar{u}_{max} - \bar{u}_2 = 0$, o kai $k = 200$, $\bar{u}_{max} - \bar{u}_2 = 1{,}726$. Vadinasi, kai klasterių skaičius didesnis, gaunamas didesnis $\bar{u}_{s+1}$ reikšmės didėjimas lyginant su pradine reikšme u_2, tačiau norint pasiekti $\bar{u}_{max}$ reikšmę reikia turėti daugiau naujo vartotojo įverčių s. Nagrinėjamu atveju, kai $k = 200$, s_{opt}=56, $\bar{u}_2 = 1{,}726$, o $\bar{u}_{max}$=3,916. Kadangi $\bar{u}_{max}(k = 200)$ už $\bar{u}_{max}(k = 1)$ didesnė vos 1 %, galima teigti, kad klasterizavimas su dideliu klasterių skaičiumi nėra efektyvus.

Jei kinta klasterių skaičius, reikšmė $\bar{u}_{n-1}$ beveik nekinta ir priklauso intervalui [1,051; 1,081]. Reikšmės šiame intervale labai artimos visų vartotojų vertinimų visiems produktams duomenų rinkinyje vidurkiui $\tilde{V} = 1{,}066$.

Intervalo [1; s^*] vidurkio reikšmė $\bar{u}_{vid}$ nesutampa su $\bar{u}_{max}$ reikšme, tačiau yra artima.

4.4 Siūlomo metodo ir įprastų rekomendavimo metodų efektyvumo lyginimas

Disertacijoje pateikto metodo efektyvumas palygintas su *Random* metodu (atskaitos tašku), su *MostPopular* metodu (kai netaikomas klasterizavimas ir manoma, kad visi vartotojai priklauso vienai grupei) ir įprastu bendrojo filtravimo metodu. Lyginimo kriterijus – vidutinė rekomenduojamo produkto įverčio reikšmė, o eksperimentai atlikti su *Jester* I duomenų rinkiniu.

Norint objektyviai patikrinti metodo efektyvumą, duomenų aibėje atrinkti 7200 vartotojai, įvertinę visus 100 produktų. Eksperimento tyrimo metodika pateikta 4.3.1 poskyryje: pagal šią metodiką, skaičiavimai, kai vykdomas s produktų parinkimas vartotojo elgsenai vertinti, atlikti daug kartų, ir iš rezultatų išvestas vidurkis. Tyrimo metu su kiekvienu validavimo aibės vartotoju, esant fiksuotai s reikšmei, atlikta 100 eksperimentų, pateikiami vis kiti produktų rinkiniai $\{b_{N_1}, \dots, b_{N_s}\}$. Iš rezultatų išvestas vidurkis. Taip gaunamas vidutinis rekomenduojamo $(s+1)$-mojo produkto įvertis $\bar{u}_{s+1}$, kai žinomi kitų s produktų vertinimo rezultatai.

Tirtų metodų rekomenduojamo $(s+1)$-mojo produkto vidutinio įverčio $\bar{u}_{s+1}$ kaita, atsižvelgiant į naujo vartotojo įvertintų produktų skaičių, pateikiama 30 paveiksle: a – siūlomas metodas dvidešimties klasterių atveju ($k = 20$), b – bendruoju filtravimu (BF) paremtas metodas (*UserKNN*), c – siūlomas metodas vieno klasterio atveju ($k = 1$, *MostPopular* metodo modifikacija), d – atsitiktinio siūlymo (*Random*) metodas.

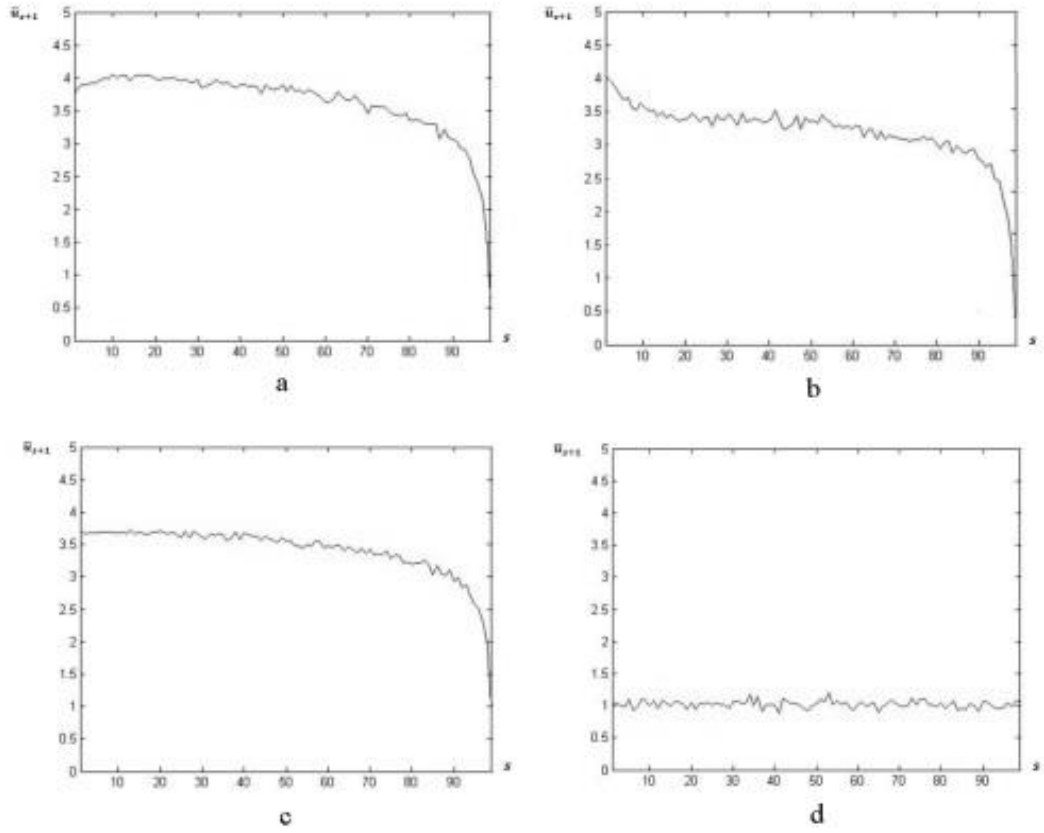
Iš 30 paveikslo pastebėtinos šios tendencijos:

1. Disertacijoje siūlomu metodu rekomenduojamo $(s+1)$-mojo produkto vidutinio įverčio $\bar{u}_{s+1}$ reikšmė didėja iki tam tikros reikšmės, kai didėja s, o po to mažėja, kol sumažėja iki visų vartotojų visų produktų vertinimo vidurkio $\tilde{V}$.

2. Atsitiktinio siūlymo metodu gaunami rezultatai yra artimi visų vartotojų vertinimų visiems produktams vidurkiui $\bar{V}$ nepriklausomai nuo to, kiek produktų (s) vartotojas yra įvertinęs.

3. Bendrojo filtravimo ir *MostPopular* metodų atveju įverčio $\bar{u}_{s+1}$ reikšmė mažėja nuo didžiausios iki $\bar{V}$.

4. Kai $s = 1$, didžiausia $\bar{u}_{s+1}$ reikšmė gaunama bendrojo filtravimo metodu. Vis dėlto pateikiamu naujuoju metodu gaunamos ir didesnės $\bar{u}_{s+1}$ reikšmės nei bendrojo filtravimo metodu, bet tik jei s didesni, tai yra kai naujo vartotojo vertinimų produktams istorija apima daugiau kaip 5 % visų duomenų rinkinio produktų.



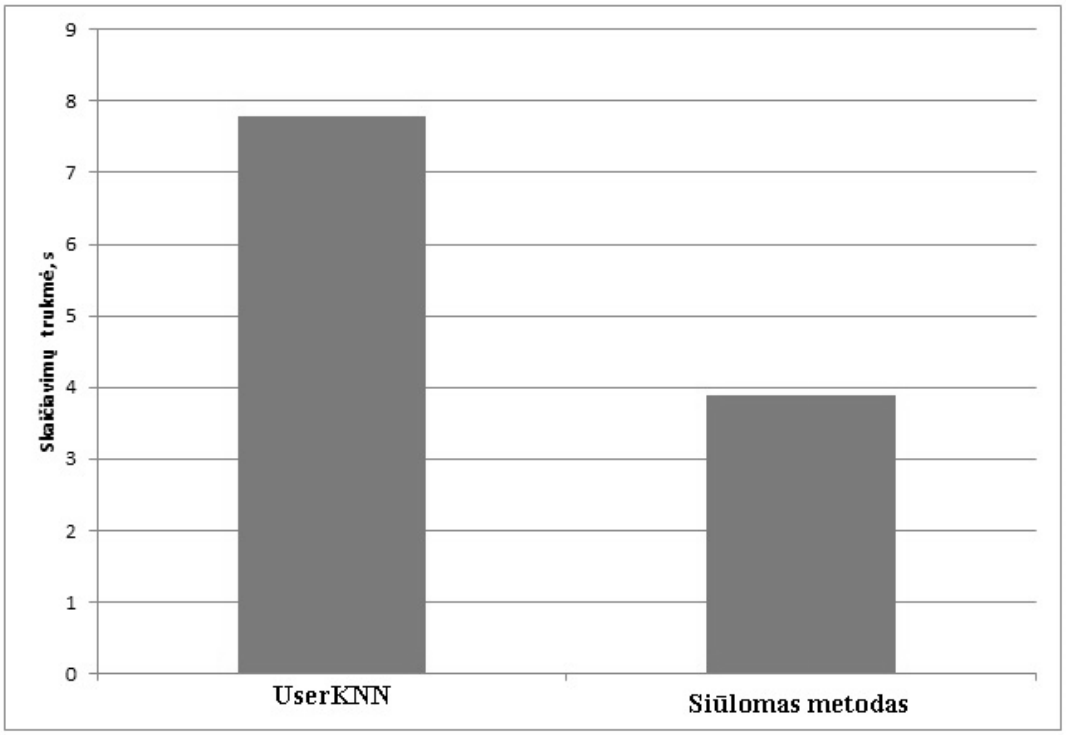
30 paveikslas. *Metodų efektyvumo lyginimas: a) siūlomas metodas, b) bendruoju filtravimu (BF) paremtas metodas (UserKNN), c) siūlomas metodas 1 klasterio atveju (MostPopular metodo modifikacija), d) atsitiktinio siūlymo (Random) metodas*

Naujasis ir bendrojo filtravimo metodai yra efektyvesni vienas už kitą, jei vartotojų vertintų produktų skaičiaus *s* reikšmės skirtingos.

Dar vienas siūlomo naujojo metodo privalumas – skaičiavimai vyksta daug greičiau nei bendrojo filtravimo metodu (žr. 31 paveikslą). Žinoma, santykinai daug laiko užima vartotojų klasterizavimo procesas. Jis 31 paveiksle vaizduojamu atveju truko apie 132 procesoriaus sekundes, vis dėlto šio proceso nereikia vykdyti realiu laiku.

Skaičiavimų trukmė ypač aktuali realaus laiko sistemoms, kai vartotojas negali ilgai laukti rekomendacijos. Šio metodo privalumas – dalies skaičiavimų (vartotojų klasterizavimo) iškėlimas iš realaus laiko sistemos. Nors vartotojų klasterizavimas užima santykinai daug laiko, vartotojų klasterius sukurti ir

atnaujinti galima tuo metu, kai vartotojų srautas yra mažas arba šį procesą netgi galima perkelti į atskirą kompiuterinę sistemą (*grid* sistemą ar superkompiuterį).



31 paveikslas. *Metodų vykdymo trukmės lyginimas (10 klasterių atvejis*), *rekomendacijas generuojant 1000 vartotojų*

4.5 Ketvirtojo skyriaus apibendrinimas ir išvados

Ketvirtajame skyriuje pristatytas ir eksperimentiškai tirtas naujas rekomendacinis metodas, įvertinantis vartotojų grupių specifiką.

Norint gauti geriausias rekomendacijas, reikia nustatyti optimalų vartotojų klasterių skaičių k. Vieno klasterio atveju ($k = 1$) daroma prielaida, kad visi vartotojai yra panašūs, tad tyrimo be vartotojų klasterizavimo (kai $k = 1$) rezultatai yra atskaitos taškas vertinti eksperimentų, kuriuose vykdomas panašių vartotojų grupavimas, rezultatams.

Eksperimentinis tyrimas su *Jester* I duomenų rinkiniu parodė, kad optimalus klasterių skaičius k šio duomenų rinkinio vartotojams priklauso

intervalui $[7; 30]$, o geriausias rezultatas gaunamas, kai $k = 10$. Tada pastebimas 9,2 % vidutinės rekomenduojamo produkto įverčio reikšmės $\bar{u}_{max}$ didėjimas. Iš eksperimentinio tyrimo rezultatų galima pastebėti, kad geriausi rekomendacijų rezultatai gaunami tada, kai turima naujo vartotojo vertinimų produktams istorija apima bent apie 25 % visų duomenų rinkinio produktų.

Vartotojų klasterizavimas su dideliu klasterių skaičiumi nėra efektyvus. Tyrimas parodė, kad didžiausia vidutinė rekomenduojamo produkto įverčio reikšmė $\bar{u}_{max}$, kai yra 200 klasterių, vos 1 % didesnė už $\bar{u}_{max}$ vieno klasterio atveju.

Naujasis ir bendrojo filtravimo metodai yra efektyvesni vienas už kitą, kai vartotojų vertintų produktų skaičiaus s reikšmės skirtingos.

Siūlomu metodu gaunamos geresnės rekomendacijos nei bendrojo filtravimo metodu (atlikto tyrimo atveju gerumo kriterijus nustatytas pagal vidutinę rekomenduojamo produkto įverčio reikšmę), kai naujo vartotojo vertinimų produktams istorija apima daugiau nei 5 % visų duomenų rinkinio produktų.

5

Rezultatų apibendrinimas ir išvados

Interneto socialiniai tinklai ir įvairios elektroninės parduotuvės šiuo metu tampa vis populiaresnės. Siekiama geriau įvertinti vartotojų poreikius ir rekomenduoti jiems tinkamus produktus ar paslaugas, todėl vartotojai ir jų elgsena yra įvairiopai analizuojami. Disertacijoje pateiktas rekomendavimo metodas, įvertinantis vartotojų grupių elgsenos specifiką, kai vartotojų įverčių produktams matrica yra tankiai užpildyta. Tokio tipo duomenų rinkiniai paplitę specializuotose elektroninėse parduotuvėse, taip pat įvairiuose specifiniuose interneto kataloguose. Naujo metodo tikslas – sugrupuoti panašius vartotojus į tam tikrą klasterių skaičių ir ieškoti klasterio, kuriam priklausantys vartotojai yra panašiausi į naują vartotoją, kuriam reikia sugeneruoti rekomendaciją. Čia ieškoma ne panašiausių vartotojų fiksuoto skaičiaus, o visos jų grupės, kurios dydis iš anksto nėra žinomas.

Tyrimai leido padaryti šias išvadas:

1. Pateiktas rekomendavimo metodas įvertina vartotojų grupių elgsenos specifiką. Eksperimentinis tyrimas su *Jester* I duomenų rinkiniu parodė, kad optimalus klasterių skaičius k šio duomenų rinkinio vartotojams priklauso intervalui $[7;\ 30]$, o geriausi rezultatai gaunami, kai $k = 10$. Tada pastebimas 9,2 % vidutinės rekomenduojamo produkto įverčio reikšmės didėjimas lyginant su populiariausių prekių rekomendavimu, kai neatsižvelgiama į pirkėją. Vis dėlto vartotojų klasterizavimas su dideliu klasterių skaičiumi nėra

efektyvus – tyrimas parodė, kad didžiausia vidutinė rekomenduojamo produkto įverčio reikšmė, kai yra 200 klasterių, vos 1 % didesnė už to įverčio reikšmę vieno klasterio atveju.

2. Vartotojų klasterizavimas pagreitina rekomendacijų generavimo procesą. Pateiktu metodu produktų rekomendacijas tūkstančiui vartotojų pavyko sugeneruoti du kartus greičiau nei su įprastu bendrojo filtravimo metodu, realizuotu toje pačioje programinėje aplinkoje. Be to, gaunamos geresnės rekomendacijos nei su bendrojo filtravimo metodu, kai naujo vartotojo vertinimų produktams istorija apima daugiau kaip 5 % visų duomenų rinkinio produktų. Tolesnių tyrimų objektas galėtų būti šių metodų sujungimas – rekomendacijos būtų generuojamos iš pradžių vienu metodu, o po to kitu.

3. Nėra universaliai gerų rekomendavimo metodų – jų rezultatams daro įtaką duomenų rinkinio specifika. Geriausi rezultatai gauti taikant skirtingus metodus eksperimentiškai tirtiems duomenų rinkiniams. Tiriant *Jester* duomenų rinkinį, tinkamiausias metodas buvo *MatrixFactorization*; *CoClustering* metodas buvo tik šeštas. O štai tiriant *MovieLens* duomenų rinkinį tinkamiausias metodas buvo *SVDPlusPlus*, tačiau *CoClustering* metodas buvo trečias. Kadangi naujasis ir bendrojo filtravimo metodai yra efektyvesni vienas už kitą, kai vartotojų vertintų produktų skaičiaus reikšmės skirtingos, nagrinėtinas šių metodų sujungimas – rekomendacijos būtų generuojamos iš pradžių vienu metodu, o po to kitu.

4. Pateikto metodo generuojamų rekomendacijų efektyvumą lemia vartotojų įverčių produktams matricos tankis. Nustatyta, kad geriausios rekomendacijos gaunamos tada, kai naujo vartotojo vertinimų produktams istorija apima bent apie 25 % visų duomenų rinkinio produktų.

Literatūros sąrašas

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6): 734–749, 2005.

[2] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, WSDM '12, p. 143–152, New York, NY, USA, 2012. ACM.

[3] Robin Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4): 331–370, November 2002.

[4] Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, p. 93–103, 2000.

[5] Michelle Keim Condliff, David D. Lewis, and David Madigan. Bayesian mixed-effects models for recommender systems. In *In ACM SIGIR ā€™99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[6] Laima Daukšyte, Ricardas Krikštolaitis, and Algimantas Jonas Bikelis. *Europos šaliu gamintoju kainu indekso prognozavimas*. PhD thesis, Vytauto Didžiojo universitetas, 2011.

[7] Inderjit S Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 269–274. ACM, 2001.

[8] Craig A Ellis and Simon A Parbery. Is smarter better? a comparison of adaptive, and simple moving average trading strategies. *Research in International Business and Finance*, 19(3): 399–411, 2005.

[9] Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. Bayesian personalized ranking for non-uniformly sampled items. *JMLR W&CP*, 18, 2012.

[10] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Mymedialite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*, p. 305–308. ACM, 2011.

[11] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Data Mining, Fifth IEEE International Conference on*, p. 4. IEEE, 2005.

[12] Mustansar Ali Ghazanfar and Adam Prugel-bennett. An improved switching hybrid recommender system using naive bayes classifier and collaborative filtering.

[13] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2): 133–151, 2001.

[14] Stephen Gorard. Revisiting a 90-year-old debate: the advantages of the mean deviation. *British Journal of Educational Studies*, 53(4): 417–430, 2005.

[15] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, p. 403–412. ACM, 2004.

[16] James Douglas Hamilton. *Time series analysis*. Princeton Univ. Press, Princeton, NJ, 1994.

[17] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1): 5–53, 2004.

[18] Thomas Hofmann and Jan Puzicha. Latent class models for collaborative filtering. In *IJCAI*, volume 99, p. 688–693, 1999.

[19] Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1): 116–142, 2004.

[20] Zan Huang, Daniel Zeng, and Hsinchun Chen. A comparative study of recommendation algorithms in e-commerce applications. *IEEE Intelligent Systems*, 22(5): 68–78, 2007.

[21] Michael Jahrer and Andreas Töscher. Collaborative filtering ensemble. In *KDD Cup*, p. 61–74, 2012.

[22] Mohsen Jamali and Martin Ester. Using a trust network to improve top-n recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, p. 181–188, New York, NY, USA, 2009. ACM.

[23] Jason J Jung and Xuan Hau Pham. Attribute selection-based recommendation framework for long-tail user group: An empirical study on movielens dataset. In *Computational Collective Intelligence. Technologies and Applications*, p. 592–601. Springer, 2011.

[24] Kyoung-jae Kim and Hyunchul Ahn. A recommender system using ga k-means clustering in an online shopping market. *Expert systems with applications*, 34(2):1200–1209, 2008.

[25] Yehuda Koren. 1 the bellkor solution to the netflix grand prize, 2009.

[26] Joonseok Lee, Mingxuan Sun, and Guy Lebanon. A comparative study of collaborative filtering algorithms. *CoRR*, abs/1205.3193, 2012.

[27] Haifeng Liu, Ee-Peng Lim, Hady W Lauw, Minh-Tam Le, Aixin Sun, Jaideep Srivastava, and Young Kim. Predicting trusts among users of online communities: an epinions case study. In *Proceedings of the 9th ACM conference on Electronic commerce*, p. 310–319. ACM, 2008.

[28] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, p. 73–105. Springer, 2011.

[29] Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 1(1): 24–45, 2004.

[30] Oded Maimon and Lior Rokach. *Data mining and knowledge discovery handbook*, volume 2. Springer, 2005.

[31] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[32] Paolo Massa and Paolo Avesani. Controversial users demand local trust metrics: An experimental study on epinions. com community. In *Proceedings of the National Conference on artificial Intelligence*, volume 20, p. 121. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[33] Brian McFee, Thierry Bertin-Mahieux, Daniel P.W. Ellis, and Gert R.G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st International Conference Companion on World Wide Web*, WWW '12 Companion, p. 909–916, New York, NY, USA, 2012. ACM.

[34] Brian McFee, Thierry Bertin-Mahieux, Daniel PW Ellis, and Gert RG Lanckriet. The million song dataset challenge. In *Proceedings of the 21st international conference companion on World Wide Web*, p. 909–916. ACM, 2012.

[35] Bradley N Miller, Istvan Albert, Shyong K Lam, Joseph A Konstan, and John Riedl. Movielens unplugged: experiences with an occasionally connected recommender system. In *Proceedings of the 8th international conference on Intelligent user interfaces*, p. 263–266. ACM, 2003.

[36] Richard Oberuc. Dynamic portfolio theory and management. 2003.

[37] Mark Oā€™Connor and Jon Herlocker. Clustering items for collaborative filtering. In *Proceedings of the ACM SIGIR workshop on recommender systems*, volume 128. Citeseer, 1999.

[38] Sean Owen, Robin Anil, Ted Dunning, and Ellen Friedman. *Mahout in action*. Manning, 2011.

[39] Javier Parra-Arnau, David Rebollo-Monedero, and Jordi Forné. A privacy-protecting architecture for recommendation systems via the suppression of ratings. *Int. J. Security, Appl*, 6: 61–80, 2012.

[40] Michael J. Pazzani and Daniel Billsus. The adaptive web. chapter Content-based Recommendation Systems, p. 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.

[41] MichaelJ. Pazzani and Daniel Billsus. Content-based recommendation systems. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, p. 325–341. Springer Berlin Heidelberg, 2007.

[42] Naren Ramakrishnan, Benjamin J Keller, Batul J Mirza, Ananth Y Grama, and George Karypis. Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6): 54–62, 2001.

[43] A. M. Rashid. Clustknn: A highly scalable hybrid model & memory based algorithm, 2006.

[44] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, p. 452–461. AUAI Press, 2009.

[45] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.

[46] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2Nd ACM Conference on Electronic Commerce*, EC ’00, p. 158–167, New York, NY, USA, 2000. ACM.

[47] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW ’01, p. 285–295, New York, NY, USA, 2001. ACM.

[48] Badrul M Sarwar, George Karypis, Joseph Konstan, and John Riedl. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. In *Proceedings of the fifth international conference on computer and information technology*, volume 1. Citeseer, 2002.

[49] J. Ben Schafer, Joseph Konstan, and John Riedl. Recommender systems in e-commerce. In *Proceedings of the 1st ACM Conference on Electronic Commerce*, EC '99, p. 158–166, New York, NY, USA, 1999. ACM.

[50] J Ben Schafer, Joseph A Konstan, and John Riedl. E-commerce recommendation applications. In *Applications of Data Mining to Electronic Commerce*, p. 115–153. Springer, 2001.

[51] Sebastian Schelter and Sean Owen. Collaborative filtering with apache mahout. *Proc. of ACM RecSys Challenge*, 2012.

[52] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems, Chicago, USA*, volume 23, 2011.

[53] Michael Sevilla. Applicability of mahout for large data sets. *Experiences and Lessons Learned, University of California, Santa Cruz Santa Cruz CA USA, unkown*.

[54] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, p. 257–297. Springer, 2011.

[55] Logan Short, Christopher Wong, and David Zeng. Tag recommendations in stackoverflow. 2014.

[56] Michael Small and C.K. Tse. Applying the method of surrogate data to cyclic time series. *Physica D: Nonlinear Phenomena*, 164: 187 – 201, 2002.

[57] Alan F. Smeaton and Jamie Callan. Personalisation and recommender systems in digital libraries. *Int. J. on Digital Libraries*, 5(4): 299–308, 2005.

[58] Pavel Stefanovic and Olga Kurasova. Creation of text document matrices and visualization by self-organizing map. *Information Technology And Control*, 43(1): 37–46, 2014.

[59] Luigi Troiano and Irene Díaz. A model for preserving privacy in recommendation systems. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems*, p. 56–65. Springer, 2014.

[60] Alex Tuzhilin, Yehuda Koren, Jim Bennett, Charles Elkan, and Daniel Lemire. Large-scale recommender systems and the netflix prize competition.

[61] Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, 1998.

[62] Ramnath Vaidyanathan. *Retail demand management: Forecasting, assortment planning and pricing*. PhD thesis, Citeseer, 2011.

[63] Saúl Vargas and Pablo Castells. Rank and relevance in novelty and diversity metrics for recommender systems. In *Proceedings of the fifth ACM conference on Recommender systems*, p. 109–116. ACM, 2011.

[64] João Vinagre, Alpio Mário Jorge, and João Gama. Fast incremental matrix factorization for recommendation with positive-only feedback. In *User Modeling, Adaptation, and Personalization*, p. 459–470. Springer, 2014.

[65] E. Vozalis and K. G. Margaritis. Analysis of recommender systems' algorithms. In *The 6th Hellenic European Conference on Computer Mathematics & its Applications (HERCMA), Athens, Greece*, 2003.

[66] Zhou Wang and Alan C Bovik. Mean squared error: love it or leave it? a new look at signal fidelity measures. *Signal Processing Magazine, IEEE*, 26(1): 98–117, 2009.

[67] William Wu-Shyong Wei. *Time series analysis*. Addison-Wesley publ, 1994.

[68] Thomas R Willemain. The effect of graphical adjustment on forecast accuracy. *International Journal of Forecasting*, 7(2): 151–154, 1991.

[69] Kevin K.F. Wong, Haiyan Song, and Kaye S. Chon. Bayesian models for tourism demand forecasting. *Tourism Management*, 27(5): 773–780, 2006.

[70] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, p. 21–30. ACM, 2012.

[71] Xiwang Yang, Yang Guo, and Yong Liu. Bayesian-inference-based recommendation in online social networks. *Parallel and Distributed Systems, IEEE Transactions on*, 24(4): 642–651, 2013.

[72] Xiaohui Yu, Yang Liu, Xiangji Huang, and Aijun An. A quality-aware model for sales prediction using reviews. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, p. 1217–1218, New York, NY, USA, 2010. ACM.

[73] Zied Zaier, Robert Godin, and Luc Faucher. Evaluating recommender systems. In *Automated solutions for Cross Media Content and Multi-channel Distribution, 2008. AXMEDIS'08. International Conference on*, p. 211–217. IEEE, 2008.

[74] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, p. 22–32, New York, NY, USA, 2005. ACM.

Autoriaus publikacijų sąrašas disertacijos tema

Straipsniai recenzuojamuose periodiniuose moksliniuose leidiniuose

[1A] **Rapečka, Aurimas**; Marcinkevičius, Virginijus. Knygų paklausos prognozavimo elektroniniame knygyne galimybės. *Informacijos mokslai*, ISSN 1392-1487. Priimta.

[2A] **Rapečka, Aurimas**; Dzemyda, Gintautas. A new recommendation model for the user clustering-based recommendation system. *Information Technology and Control*, 2015, Vol. 44, No. 1, ISSN 1392-124X, p. 54–63. (Impact Factor 2014: 0,623)

[3A] Kurasova, Olga; Marcinkevičius, Virginijus; Medvedev, Viktor; **Rapečka, Aurimas**; Stefanovič, Pavel. Strategies for big data clustering. *ICTAI 2014: 2014 IEEE 26th International Conference on Tools with Artificial Intelligence* [Proceedings], 10-12 November, 2014, Limassol, Cyprus., IEEE Computer Society, 2014. ISSN 1082-3409, p. 740–747.

[4A] **Rapečka, Aurimas**; Marcinkevičius, Virginijus; Dzemyda, Gintautas. Rekomendacinės sistemos algoritmų veikimo elektroninio knygyno duomenų bazėje analizė. *Informacijos mokslai*, 2013, t. 65, ISSN 1392-0561, p. 45–55.

[5A] Kurasova, Olga; Marcinkevičius, Virginijus; Medvedev, Viktor; **Rapečka, Aurimas**. Duomenų tyrybos sistemos, pagrįstos saityno paslaugomis. *Informacijos mokslai*, 2013, t. 65, ISSN 1392-0561, p. 66–74.

Straipsniai kitose tarptautinių ir respublikinių konferencijų medžiagose

[6A] **Rapečka, Aurimas**; Dzemyda, Gintautas. A new recommendation method for the user clustering-based recommendation system. *EMC2015: Engineering Management and Competitiveness: 5th International Symposium* [Proceedings], June 19–20, 2015, Zrenjanin, Serbia, ISBN 9788676722563. p. 328–332.

[7A] Kligienė, Stanislava Nerutė; **Rapečka, Aurimas**. Challenges of digital era: potential and pitfalls of social media. ethics and trust in collaborative cross-domains. *COLLA 2011*: *The First International Conference on Advanced Collaborative Networks, Systems and Applications* [Proceedings], June 19–24, 2011, Luxembourg, ISBN 9781612081434. p. 34–39.

[8A] **Rapečka, Aurimas**; Dzemyda, Gintautas. Rekomendacinių sistemų ir jose naudojamų rekomendavimo algoritmų apžvalga. *XV Kompiuterininkų konferencijos mokslo darbai*. Vilnius, 2011, ISBN 9789986342618. p. 175–185.

A PRIEDAS

Pavyzdinis eksperimentų su *MyMediaLite* programine įranga *C#* programinis kodas

```
using System;
using System.IO;
using System.Diagnostics; // nebutina stopwatch naudojimui
using System.Collections.Generic;
using System.Threading.Tasks;
using MyMediaLite;
using MyMediaLite.Data;
using MyMediaLite.DataType;
using MyMediaLite.Eval;
using MyMediaLite.IO;
using MyMediaLite.ItemRecommendation;

namespace BarzdaRecomendation
{
    class Program
    {
        protected string data_dir = "D:\\knyga\\";
        protected string test_users_file =
"data_41177_be_pasikartojimu_pagal_laika.csv";
        protected string items_attribute_file =
"data_24984_book_category_atribute_updated.csv";
        protected string book_author_attribute_file =
"data_20620_book_authors_atribute.csv";
        string data_file;
        string data_attribute_file;
        IMapping item_mapping = new Mapping();
        IMapping user_mapping = new Mapping();
        IBooleanMatrix attribute_data;
        IPosOnlyFeedback training_data;
        IBooleanMatrix attribute_author_data;
        bool show_results = true;

        static void Main(string[] args)
        {
            var program = new Program();
            program.Run(args);
        }

        protected void Init()
        {
            data_file = Path.Combine(this.data_dir, this.test_users_file);
            data_attribute_file = Path.Combine(this.data_dir,
this.items_attribute_file);
            attribute_data = AttributeData.Read(data_attribute_file, item_mapping);
```

```
            attribute_author_data = AttributeData.Read(Path.Combine(this.data_dir,
book_author_attribute_file), item_mapping);
            training_data = ItemData.Read(data_file, user_mapping, item_mapping);
        }

        protected void Run(string[] args)
        {
            Init();
            ExperimentCrossValidation();
        }

        protected void ExperimentCrossValidation()
        {
            string[] recommenders = new string[]  {
"MyMediaLite.ItemRecommendation.Random", "Zero", "MostPopular",
"MostPopularByAttributes", "BPRMF", "BPRLinear", "LeastSquareSLIM",
"ItemAttributeSVM", "SoftMarginRankingMF", "WRMF", "WeightedBPRMF", "MultiCoreBPRMF",
"CLiMF", "BPRSLIM", "UserKNN", "ItemKNN", "ItemAttributeKNN" };
            for (int i = 3; i<=20; i++) {
                Dictionary<string, ItemRecommendationEvaluationResults> results = new
Dictionary<string, ItemRecommendationEvaluationResults>();
                foreach (string str in recommenders)
                {
                    var recommender = str.CreateItemRecommender();
                    if (recommender.GetType().GetProperty("ItemAttributes") != null)
                    {
                        var recommender2 = str.CreateItemRecommender();
                        ((IItemAttributeAwareRecommender)recommender).ItemAttributes =
new SparseBooleanMatrix();
                        results.Add(recommender.ToString(),
TestRecommenderCrossValidation(recommender, i));
                    }
                    else
                    {
                        results.Add(recommender.ToString(),
TestRecommenderCrossValidation(recommender, i));
                    }
                    WriteResultsToFile(results, "rez_crossValidation" + i + "b.txt");
                }
            }
            Console.ReadLine();
        }

        protected void ExperimentRatio() {
            Dictionary<string, ItemRecommendationEvaluationResults> results = new
Dictionary<string, ItemRecommendationEvaluationResults>();
            // Random
            ItemRecommender recommender_r = new
MyMediaLite.ItemRecommendation.Random();
            results.Add(recommender_r.ToString(),
TestRecommenderByRatio(recommender_r));

            // Zero
            Zero recommender_z = new Zero();
            results.Add(recommender_z.ToString(),
TestRecommenderByRatio(recommender_z));

            // MostPopular
            MostPopular recommender_mp = new MostPopular();
            results.Add(recommender_mp.ToString(),
TestRecommenderByRatio(recommender_mp));

            // MostPopularByAttribute
            MostPopularByAttributes recommender_mpba = new MostPopularByAttributes();
            recommender_mpba.ItemAttributes = attribute_data;
            results.Add(recommender_mpba.ToString(),
TestRecommenderByRatio(recommender_mpba));

            // BPRMF
            BPRMF recommender_bprmf = new BPRMF();
            results.Add(recommender_bprmf.ToString(),
TestRecommenderByRatio(recommender_bprmf));
            WriteResultsToFile(results, "rez1.txt");

            // BPRLinear
```

```
            BPRLinear recommender_bprl = new BPRLinear();
            recommender_bprl.ItemAttributes = attribute_data;
            results.Add(recommender_bprl.ToString(),
TestRecommenderByRatio(recommender_bprl));

            // LeastSquareSLIM
            LeastSquareSLIM recommender_lsslim = new LeastSquareSLIM();
            results.Add(recommender_lsslim.ToString(),
TestRecommenderByRatio(recommender_lsslim));

            // ItemAttributeSVM
            ItemAttributeSVM recommender_iasvm = new ItemAttributeSVM();
            recommender_iasvm.ItemAttributes = attribute_data;
            results.Add(recommender_iasvm.ToString(),
TestRecommenderByRatio(recommender_iasvm));

            // SoftMarginRankingMF
            SoftMarginRankingMF recommender_smrmf = new SoftMarginRankingMF();
            results.Add(recommender_smrmf.ToString(),
TestRecommenderByRatio(recommender_smrmf));

            // WRMF
            WRMF recommender_wrmf = new WRMF();
            results.Add(recommender_wrmf.ToString(),
TestRecommenderByRatio(recommender_wrmf));

            // WeightedBPRMF
            WeightedBPRMF recommender_wprmf = new WeightedBPRMF();
            results.Add(recommender_wprmf.ToString(),
TestRecommenderByRatio(recommender_wprmf));

            // MultiCoreBPRMF
            MultiCoreBPRMF recommender_mprmf = new MultiCoreBPRMF();
            results.Add(recommender_mprmf.ToString(),
TestRecommenderByRatio(recommender_mprmf));

            WriteResultsToFile(results, "rez2.txt");

            // CLiMF
            CLiMF recommender_climf = new CLiMF();
            results.Add(recommender_climf.ToString(),
TestRecommenderByRatio(recommender_climf));
            Console.Error.WriteLine("memory {0}", Memory.Usage);

            // BPRSLIM
            BPRSLIM recommender_bprslim = new BPRSLIM();
            results.Add(recommender_bprslim.ToString(),
TestRecommenderByRatio(recommender_bprslim));
            Console.Error.WriteLine("memory {0}", Memory.Usage);

            //UserKNN
            UserKNN recommender_uknn = new UserKNN();
            results.Add(recommender_uknn.ToString(),
TestRecommenderByRatio(recommender_uknn));

            //ItemKNN
            ItemKNN recommender_iknn = new ItemKNN();
            results.Add(recommender_iknn.ToString(),
TestRecommenderByRatio(recommender_iknn));

            //ItemAttributeKNN
            ItemAttributeKNN recommender_iaknn = new ItemAttributeKNN();
            recommender_iaknn.ItemAttributes = attribute_data;
            results.Add(recommender_iaknn.ToString(),
TestRecommenderByRatio(recommender_iaknn));

            WriteResultsToFile(results);
        }

        protected void WriteResultsToFile(Dictionary<string,
ItemRecommendationEvaluationResults> results, string filename = "experiment
results.txt")
        {
            var writer = new StreamWriter(filename);
            foreach ( var key in results) {
```

```
                writer.Write(key.Key);
                foreach (var mesure in key.Value)
                {
                    writer.Write(";{0} = {1}", mesure.Key, mesure.Value);
                }
                writer.Write("\r\n");
            }
            writer.Close();
        }

        protected ItemRecommendationEvaluationResults
TestRecommenderCrossValidation(ItemRecommender recommender, int min_items_number = 3)
        {
            int tests_number = min_items_number;
            IList<int> test_users = null;
            IList<int> candidate_items = null;
            CandidateItems candidate_item_mode = CandidateItems.OVERLAP;
            var avg_results = new ItemRecommendationEvaluationResults();
            bool compute_fit = false;
            IList<string> mesures_to_show = new string[] { "AUC", "MAP", "NDCG",
"prec@5", "prec@10" };
            var split = new
PosOnlyFeedbackCrossValidationSplit<PosOnlyFeedback<SparseBooleanMatrix>>(training_dat
a, (uint)min_items_number);
            long mem = Memory.Usage;
            if (show_results)
                Console.WriteLine("Recommender - {0}", recommender.ToString());
            for (int fold = 0; fold < min_items_number; fold++)
            {
                var split_recommender = (ItemRecommender)recommender.Clone(); // avoid
changes in recommender
                split_recommender.Feedback = split.Train[fold];
                TimeSpan t_train = Wrap.MeasureTime(delegate()
                {
                    split_recommender.Train();
                });
                ItemRecommendationEvaluationResults fold_results = new
ItemRecommendationEvaluationResults();
                TimeSpan t_test = Wrap.MeasureTime(delegate()
                {
                    fold_results = Items.Evaluate(split_recommender, split.Test[fold],
split.Train[fold], test_users, candidate_items, candidate_item_mode);
                });

                fold_results.MeasuresToShow = mesures_to_show;
                if (compute_fit)
                    fold_results["fit"] = (float)split_recommender.ComputeFit();

                lock (avg_results)
                {
                    foreach (var key in fold_results.Keys)
                    {
                        if (avg_results.ContainsKey(key))
                            avg_results[key] += fold_results[key];
                        else
                            avg_results[key] = fold_results[key];
                        string std = string.Concat(key, "_std");
                        if (avg_results.ContainsKey(std))
                            avg_results[string.Concat(key, "_std")] +=
fold_results[key] * fold_results[key];
                        else
                            avg_results[string.Concat(key, "_std")] =
fold_results[key] * fold_results[key];
                    }
                    avg_results["train_time"] = (float)t_train.TotalMilliseconds;
                    avg_results["test_time"] = (float)t_test.TotalMilliseconds;
                }

                if (show_results)
                    Console.Error.WriteLine("Experiment {0} - {1} \n", fold,
fold_results.ToString());
            };

            foreach (var key in Items.Measures)
            {
```

```
                avg_results[key] /= tests_number;
                double std = avg_results[string.Concat(key, "_std")] / tests_number -
avg_results[key] * avg_results[key];
                avg_results[string.Concat(key, "_std")] = (float) Math.Sqrt(std);
            }
            string[] arr = new string[] { "num_users", "num_items", "num_lists"};

            foreach (var key in arr)
            {
                avg_results[key] /= tests_number;
                double std = avg_results[string.Concat(key, "_std")] / tests_number -
avg_results[key] * avg_results[key];
                avg_results[string.Concat(key, "_std")] = (float)Math.Sqrt(std);
            }
            avg_results["train_time"] /= tests_number;
            avg_results["test_time"] /= tests_number;
            avg_results["memory"] = (float)Memory.Usage - mem;
            avg_results.MeasuresToShow = mesures_to_show;

            if (show_results)
            {
                Console.Error.WriteLine("Experiment averages {0} \n", avg_results);
            }
            return avg_results;
        }

        protected ItemRecommendationEvaluationResults
TestRecommenderByRatio(ItemRecommender recommender, double test_ratio = 0.1)
        {
            int tests_number = 10;
            IList<int> test_users = null;
            IList<int> candidate_items = null;
            CandidateItems candidate_item_mode = CandidateItems.OVERLAP;
            var avg_results = new ItemRecommendationEvaluationResults();
            bool compute_fit = false;
            IList<string> mesures_to_show = new string[] { "AUC", "MAP", "NDCG",
"prec@5", "prec@10" };

            long mem = Memory.Usage;

            if (show_results)
                Console.WriteLine("Recommender - {0}", recommender.ToString());

            for (int fold = 0; fold < tests_number; fold++)
            {
                var split = new
PosOnlyFeedbackSimpleSplit<PosOnlyFeedback<SparseBooleanMatrix>>(training_data,
test_ratio);

                var split_recommender = (ItemRecommender)recommender.Clone(); // avoid
changes in recommender
                split_recommender.Feedback = split.Train[0];
                TimeSpan t_train = Wrap.MeasureTime(delegate()
                {
                    split_recommender.Train();
                });
                ItemRecommendationEvaluationResults fold_results = new
ItemRecommendationEvaluationResults();
                TimeSpan t_test = Wrap.MeasureTime(delegate()
                {
                    fold_results = Items.Evaluate(split_recommender, split.Test[0],
split.Train[0], test_users, candidate_items, candidate_item_mode);
                });

                fold_results.MeasuresToShow = mesures_to_show;
                if (compute_fit)
                    fold_results["fit"] = (float)split_recommender.ComputeFit();

                lock (avg_results)
                {
                    foreach (var key in fold_results.Keys)
                        if (avg_results.ContainsKey(key))
                            avg_results[key] += fold_results[key];
                        else
                            avg_results[key] = fold_results[key];
```

```
                    avg_results["train_time"] = (float) t_train.TotalMilliseconds;
                    avg_results["test_time"] = (float) t_test.TotalMilliseconds;
                }

                if (show_results)
                    Console.Error.WriteLine("Experiment {0} - {1} \n", fold,
fold_results.ToString());
            };

            foreach (var key in Items.Measures)
                avg_results[key] /= tests_number;
            avg_results["num_users"] /= tests_number;
            avg_results["num_items"] /= tests_number;
            avg_results["train_time"] /= tests_number;
            avg_results["test_time"] /= tests_number;
            avg_results["memory"] = (float) Memory.Usage - mem;
            avg_results.MeasuresToShow = mesures_to_show;

            if (show_results) {
                Console.Error.WriteLine("Experiment averages {0} \n", avg_results);
            }

            return avg_results;
        }
    }
}
```

AURIMAS RAPEČKA

REKOMENDACINIŲ SISTEMŲ SOCIALINIUOSE TINKLUOSE EFEKTYVUMO DIDINIMAS

Daktaro disertacija

Fiziniai mokslai, informatika (09 P)

Redaktorė Jorūnė Rimeisytė