Rekomendacinės sistemos algoritmų veikimo elektroninio knygyno duomenų bazėje analizė

Aurimas Rapečka

Vilniaus universiteto Matematikos ir informatikos instituto doktorantas
Vilnius University, Institute of Mathematics and Informatics, PhD student
Akademijos g. 4, LT-08663 Vilnius
El. paštas: Aurimas.Rapecka@mii.vu.lt

Virginijus Marcinkevičius

Vilniaus universiteto Matematikos ir informatikos instituto daktaras
Vilnius University, Institute of Mathematics and Informatics, PhD, Senior researcher
Akademijos g. 4, LT-08663 Vilnius
El. paštas: Virginijus.Marcinkevicius@mii.vu.lt

Gintautas Dzemyda

Vilniaus universiteto Matematikos ir informatikos instituto profesorius
Vilnius University, Institute of Mathematics and Informatics, Professor, Principal researcher
Akademijos g. 4, LT-08663 Vilnius
El. paštas: Gintautas.Dzemyda@mii.vu.lt

Straipsnis skiriamas rekomendacinių sistemų algoritmų veikimo konkrečioje elektroninės parduotuvės duomenų bazėje analizei. Analizės tikslas – pagal pasirinktus įverčius rasti rekomendacinių sistemų algoritmus, efektyviausiai veikiančius turimoje duomenų bazėje. Šiame straipsnyje palyginti nemokamos rekomendacinių sistemų programinės įrangos paketai, aprašytas su pasirinkta programine įranga atliktas rekomendavimo algoritmų efektyvumo turimoje duomenų bazėje eksperimentinis tyrimas siekiant nustatyti geriausiai ir prasčiausiai veikiančius algoritmus.

Įvadas

Daugėjant interneto vartotojų vis didesnį mastą įgauna ir prekyba internete. Įvairios elektroninės parduotuvės siūlo platų prekių asortimentą ir standartinis vartotojas čia dažnai susiduria su prekių pasirinkimo keblumais. Skirtingai nuo parduotuvių realiame pasaulyje, kuriose daiktą galima paimti į rankas ar net išbandyti, elektroninės interneto parduotuvės pirkėjas turi mažiau galimybių pats įvertinti siūlomus produktus.

Šiame straipsnyje analizuojamas pavyzdys – elektroninė knygų parduotuvė, turinti tūkstančius knygų, kurių pavartyti fiziškai neįmanoma, todėl neužtenka surūšiuoti pagal žanrą, autorių, kainą ar naujumą. Čia reikalinga rekomendacinė sistema (RS), kurios algoritmai gebėtų greitai padėti potencialiems klientams susigaudyti didelėje esamų ir naujų knygų pasiūloje ir pasiūlyti tik konkrečiam vartotojui tinkamiausias knygas.

Nors egzistuoja daug rekomendacinėse sistemose naudojamų algoritmų (Vozalis, Margaritis, 2003), tačiau mokslinėje literatūroje aprašomuose tyrimuose dažniausiai dėmesys telkiamas į vieną ar du (siūlomą ir palyginimo) algoritmus (Onuma, Tong, Faloutsos, 2009; Gong, 2010). Šiame straipsnyje žvelgiama plačiau – eksperimentiškai vertinamas 18 rekomendavimo algoritmų veikimas konkrečioje duomenų bazėje. Vienas iš rekomendavimo algoritmų efektyvumo palyginimo pavyzdžių galėtų būti 2006–2009 metais internetinėje erdvėje vykęs

Netflix Prize konkursas (jo pagrindinis prizas 1 mln. JAV dolerių), kuriame siekta sukurti efektyviausiai *Netflix* filmų duomenų bazėje veikiantį rekomendavimo algoritmą. Konkursas sulaukė apie 50 000 dalyvių (Netflix, 2009), o nugalėtoju tapo algoritmas, plačiau aptartas šaltinyje (Toscher, Jahrer, Bell, 2009).

Šio straipsnio tikslas – atlikti elektroninės parduotuvės pirkimų įrašų duomenų bazės analizę, pagal iš anksto parinktus kriterijus atlikti atvirojo kodo rekomendacinių sistemų lyginamąją analizę ir eksperimentiškai įvertinti esamų rekomendavimo algoritmų efektyvumą konkrečios duomenų bazės atveju.

Rekomendacinės sistemos ir jų naudojami algoritmai

Nors yra sukurta įvairių rekomendacinių sistemų, tačiau išskiriami du labiausiai paplitę rekomendacinių sistemų tipai: kuriančios prognozes rekomendacinės sistemos ir generuojančios rekomendacijas rekomendacinės sistemos (Ricci, Rokach, Shapira & Kantor, 2010).

Kuriančių prognozes rekomendacinių sistemų pateikiama prognozė išreiškiama skaitiniu pavidalu r_{ij}, kuris reiškia prognozuojamą vartotojo a_i įvertinimą produktui b_j. Kuriant prognozes vartotojams reikalinga įverčių produktų aibė. Dažniausiai šiuos įverčius vartotojai palieka sąmoningai – atitinkamu balu įvertindami patikusius ar nepatikusius produktus.

Rekomendacijas kuriančių RS generuojama rekomendacija išreiškiama kaip vartotojui a_i tinkamiausių produktų sąrašas N_i. Kuriant prognozes užtenka turėti dvejetainę vartotojų įvertintų produktų įverčių aibę. Šiuos įverčius vartotojai palieka nesąmoningai, pavyzdžiui, pirkdami produktus.

Straipsnyje aprašomi eksperimentai atliekami su turima dvejetainių įverčių duomenų baze, todėl aptariami tik su rekomendacijų generavimu susiję aspektai. Šiuo metu yra sukurta apie 20 labiau paplitusių rekomendacijų generavimo algoritmų.

Vienas paprasčiausių rekomendavimo algoritmų – populiariausių prekių (angl. *MostPopular*) rekomendavimas visiems pirkėjams (Huang, Zeng, Chen, 2007). Tai labai paplitęs ir dažnai praktikoje taikomas rekomendavimo algoritmas. Prekės reitingas apskaičiuojamas paprasta formule:

$$Reitingas = \frac{Vartotojų,\ pirkusių\ prekę,\ skaičius}{Vartotojų\ duomenų\ rinkinyje\ skaičius}$$

Pirmiausia rekomenduojami tie produktai, kurių reitingas didžiausias. Šis algoritmas yra labai konservatyvus dėl to, kad rekomendacijos neišeina už erdvės, su kuria vartotojas jau susipažinęs, ribos (McFee et al., 2009).

Kitas populiarus algoritmas – *k* artimiausių kaimynų (angl. *KNN*, *UserKNN*) algoritmas. Tai klasikinis rekomendacinių sistemų algoritmas. Reitingai arba prekės prognozuojamos remiantis *k* panašiausių vartotojų praeities įverčiais.

Du esminiai šio algoritmo, realizuoto rekomendacinėse sistemose, veikimo aspektai – panašumo koeficiento skaičiavimas ir panašumo slenksčio nustatymas. Populiariausios vartotojo panašumo metrikos yra Pearsono koreliacija (angl. *Pearson correlation)* (Rashid, 2006) (1) ir kosinusinis panašumas (*angl. cosine similarity*) (Ziegler, 2013) (2).

$$panašumas(g,r) = \frac{\sum_p \left(g(p)-\bar{g}\right)\left(r(p)-\bar{r}\right)}{\sqrt{\sum_p \left(g(p)-\bar{g}\right)^2 \sum_p \left(r(p)-\bar{r}\right)^2}} \quad (1)$$

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

Formulėse (1) ir (2) g – vartotojas, gaunantis rekomendaciją, r – vartotojas, kurio panašumas tikrinamas. Sumos ženklas formulėje rodo, kad panašumas skaičiuojamas naudojant visus produktus p, kuriuos abu vartotojai yra įvertinę. $g(p)$ ir $r(p)$ yra vartotojų įverčiai konkrečiam produktui p.

Formulėje (1) esantys kintamieji $\bar{g}$ ir $\bar{r}$ – atitinkamai g ir r vartotojų suteiktų įverčių jau vertintiems produktams vidurkiai. Formulės (1) skaitiklis yra vartotojų bendrų įvertintų produk-

tų įverčių nuokrypio nuo įverčių vidurkio sandaugų suma, o formulės vardiklis normalizuoja gautąjį rezultatą, kad jis priklausytų intervalui [-1;1]. Kuo rezultatas artimesnis 0, tuo vartotojai mažiau panašūs.

Savo ruožtu (2) formulės skaitiklis yra vartotojų bendrų įvertintų produktų įverčių sandaugų suma, o formulės vardiklis normalizuoja gautąjį rezultatą tam, kad jis priklausytų intervalui [0;1]. Kuo šis rezultatas artimesnis 0, tuo vartotojai mažiau panašūs, ir atvirkščiai.

KNN tipo algoritmai remiasi kiekvieno vartotojo panašumo koeficientų skaičiavimu, todėl esant didesniam vartotojų ir produktų įverčių skaičiui, šie algoritmai reikalauja didelių skaičiavimo ir laiko išteklių.

Kaip jau minėta, rekomendacinėse sistemose naudojamų algoritmų yra ir daugiau. Eksperimentinėje šio straipsnio dalyje atlikti eksperimentai su *Random* (atsitiktinis), *Zero* (visuomet grąžinantis 0 reitingą), *MostPopular* (populiariausių produktų rekomendavimo), *MostPopularByAttributes* (populiariausių kategorijose produktų rekomendavimo), *BPRMF* (Rendle et al., 2009), *BPRLinear* (Gantner et al., 2011), *LeastSquareSLIM* (Ning, Karypis, 2011), *ItemAttributeSVM* (Min, Han, 2005), *SoftMarginRankingMF* (Rendle, 2011), *WRMF* (Pan et al., 2008), *WeightedBPRMF* (Gantner et al., 2009), *MultiCoreBPRMF* (Rendle et al., 2009), *CLiMF* (Shi et al., 2012), *BPRSLIM* (Ning, Karypis, 2011), *UserKNN*, *ItemKNN* ir *ItemAttributeKNN* (visi *KNN – k* artimiausių kaimynų pagrindu veikiantys).

Rekomendacinių sistemų efektyvumo įverčių matai

Norint įvertinti, kaip tiksliai rekomendacinė sistema nustato, ar vartotojas įsigis (įsigijo) produktą, naudojami rekomendacinės sistemos efektyvumo įverčiai.

Produktus rekomenduojanti rekomendacinė sistema veikia kaip klasifikatorius, kuris nurodo, ar konkreti prekė bus (nebus) pasirinkta pirkėjo (Rajaraman, Leskovec, Ullman, 2013). Čia egzistuoja dviejų klasių atskyrimo problema (angl. *binary classification*). Pirma klasė vadinama susijusių elementų klase, arba šiuo atveju pirkėjo įsigytos prekės. Antrai klasei priklauso prekės (elementai), kurių pirkėjas neįsigijo. Prieš apmokant klasifikatorių turima duomenų aibė padalijama į mokymo ir testavimo aibes. Tuomet klasifikatorius apmokomas turima mokymo aibe, siekiant teisingai nustatyti kiekvienos prekės (elemento) klasę atskiram vartotojui. Paskui jis testuojamas naudojant testavimo aibę. Dažnai klasifikatoriai neteisingai priskiria pirmai klasei antros klasės elementus, ir atvirkščiai, o šias klasifikavimo klaidas stengiamasi įvertinti įvairiais RS efektyvumo įverčiais.

Klasifikatoriaus vertinimo principas pateikiamas 1 lentelėje. Atlikus klasifikatoriaus testavimą, gauti rezultatai surašomi į šią lentelę ir pagal ją apskaičiuojami įvairūs įverčiai. Toliau pateikiami dažniausiai taikomi įverčiai.

1 l e n t e l ė. ***Klasifikatoriaus vertinimo principai***

		Iš tiesų priklauso klasei	
		I klasė (susijusių)	II klasė
Klasifikatoriaus priskirta klasė	I klasė	TP (angl. *true positive*) (teisingai priskirta)	FP (angl. *false positive*) (I tipo klaida, kai I klasė buvo priskirta II)
	II klasė	FN (angl. *false negative*) (II tipo klaida, kai I klasei priskiriami antros klasės elementai)	TN (angl. *true negative*) (teisingai priskirta II klasei)

Preciziškumas

Preciziškumas (angl. *precision*) yra klasifikatoriaus, atskiriančio dvi klases, klasifikavimo įvertis. Šis įvertis parodo santykį, kiek iš pirmai klasei priskirtų elementų iš tiesų priklauso pirmai klasei. Kuo preciziškumas didesnis, tuo daugiau pirmai klasei priskirtų elementų iš tikrųjų yra iš primos klasės, vadinasi, tuo tiksliau veikia RS.

$$Preciziškumas = \frac{Teisingai\ priskirta\ pirmai\ klasei}{Viso\ priskirta\ pirmai\ klasei} = \frac{TP}{TP + FP}$$

Jautrumas

Klasifikatoriaus jautrumas (angl. *sensitivity, hit rate recall*) rodo teisingai priskirtų pirmai klasei pirmos klasės elementų santykį su bendru pirmos klasės elementų skaičiumi. Kuo jautrumas didesnis, tuo daugiau pirmos klasės elementų teisingai priskirta pirmai klasei, tuo tiksliau veikia RS:

$$Jautrumas = \frac{Teisingai\ priskirta\ pirmai\ klasei}{Pirmos\ klasės\ elementų} = \frac{TP}{TP + FN}$$

Plotas po ROC kreive (AUC)

AUC (angl. *area under the ROC curve*) įvertis nusako eilės (eiliškumo) nustatymo (rangavimo) kokybę įvertinant plotą po *ROC* kreive. Didžiausia *AUC* parametro reikšmė yra 1, o jeigu eilė nustatoma neatsitiktinai, tuomet $AUC > 0{,}5$. *AUC* reikšmė, arba plotas po *ROC* kreive, yra lygus tikimybei, kad klasifikatorius atsitiktinai parinktą pirkėjo nupirktą produktą įvertins aukštesniu įverčiu negu atsitiktinai parinktą produktą, kurio jis neįsigijo.

ROC kreivė – tai grafikas, iliustruojantis klasifikatoriaus, atskiriančio dvi klases, jautrumo ir specifiškumo santykį. Specifiškumas rodo antros klasės elementų priskyrimą antrai klasei.

Vidutinių preciziškumų vidurkis (MAP)

Tuo atveju, kai turime pirmos klasės elementų aibę $q_j \in Q, q_j = \{b_1, b_2, ..., b_{mj}\}$ ir eilę pirmųjų RS sugrąžintų elementų R_{jk}, tuomet vidutinis preciziškumas (angl. *average precision*) apskaičiuojamas pagal formulę:

$$ap_j = \frac{1}{m_j} \sum_{k=1}^{m_j} Preciziškumas\left(R_{jk}\right).$$

Vidutinis preciziškumas rodo, kiek tiksliai rekomendacinė sistema nuspėja, kurių produktų pirkėjas įsigis. Vidutinis preciziškumas priklauso nuo elemento vietos R_{jk} eilėje. Vidutinis preciziškumas geometriškai yra interpretuojamas kaip plotas po preciziškumo ir jautrumo kreive.

Imant pirmus 5 arba 10 R_{jk} elementų apskaičiuojami preciziškumo ir jautrumo įverčiai parodo, kiek iš jų yra primos klasės lyginant su R_{jk} ilgiu ir visų pirmos klasės elementų skaičiumi (jie žymimi *prec@5*, *prec@10*, *recall@5*, *recall@10*).

Vidutinių preciziškumų vidurkis (angl. *mean average precision, MAP*) – tai klasifikatoriaus jautrumo lygių kokybinis įvertis (Manning, Raggavan, Schutze, 2009). Turint kiekvieno pirkėjo vidutinį preciziškumą, *MAP* galima apskaičiuoti pagal formulę:

$$MAP(Q) = \frac{1}{|Q|} \sum_{j=1}^{|Q|} ap_j =$$

$$= \frac{1}{|Q|} \sum_{j=1}^{|Q|} \frac{1}{m_j} \sum_{k=1}^{m_j} Preciziškumas\left(R_{jk}\right).$$

Iš kitų klasifikatoriaus vertinimo įverčių šis išsiskiria stabilumu ir geru mokėjimu atskirti rezultatus. Geometriškai *MAP* atitinka vidutinį plotą po preciziškumo kreive. Šis plotas priklauso intervalui [0;1]. Tačiau norit *MAP* įverčiu tiksliai įvertinti rekomendacinės sistemos efektyvumą, reikia parinkti didelę ir pakankamai diversifikuotą testavimo aibę.

Nors visi aprašyti įverčiai skaičiuojami skirtingai ir praktikoje jie yra beveik lygiaverčiai, tačiau įvairiose duomenų bazėse gali parodyti kardinaliai skirtingus rezultatus. Todėl siekiant rasti efektyviausiai konkrečioje duomenų bazėje veikiančius rekomendavimo algoritmus, jų veikimo rezultatus būtina vertinti ne pagal kurį nors atskirą įvertį, bet pagal šių įverčių visumą. Šiame straipsnyje apskaičiuoti kiekvieno klasifikatoriaus įverčiai ranguojami ir galutiniam klasifikatoriaus įvertinimui naudojamas gautų rangų vidurkis.

Nemokamos ir atvirojo kodo rekomendacinės sistemos (programinė įranga)

Šiame poskyryje pateikiama trumpa lyginamosios labiausiai paplitusių nemokamo arba atvirojo kodo RS programinės įrangos analizės suvestinė (2 lentelė).

Svarbiausi programinės įrangos vertinimo aspektai turimos duomenų bazės atžvilgiu – atnaujinimo data, realizuotų RS algoritmų skaičius bei sistemose esančios reitingų prognozavimo ir produktų rekomendavimo galimybės.

2 l e n t e l ė. ***Nemokamos ir atvirojo kodo RS programinės įrangos palyginimas***

RS	Atnaujinta	Progr. kalba	RS algoritmų	Realizuoti savo	Reitingų progn.	Produktų rekomend.
MyMediaLite[1]	2013 04	*C#*	>20	Taip	Taip	Taip
Apache Mahout[2]	2012 06	*Java*	3	Ne	Taip	Taip
GraphLab[3]	2012 05	*C++*	15	Taip	Taip	Taip
LensKit[4]	2012	*Java*	?	Taip	Taip	Taip
Waffles[5]	2013 04	*C++*	>5	Ne	Taip	Ne
easyrec[6]	2012 02	*Online*	1	Ne	Taip	Taip
RecLab[7]	2011 02	*Online*	1	Ne	Taip	Ne
Crab[8]	2011	*Python*	>5	Taip	Taip	Taip
recommenderlab[9]	2011 11	*C++*	4	Taip	Taip	Taip
Jellyfish[10]	2012 12	*Python*	1	Ne	Taip	Ne
OpenSlopeOne[11]	2010 06	*PHP*	1	Taip	Taip	Ne
AppRecommender[12]	2011	*Python*	>10	Ne	Taip	Taip

[1] *MyMediaLite* (http://mymedialite.net/)
[2] *Apache Mahout* (http://mahout.apache.org/)
[3] *GraphLab* collaborative filtering library (http://select.cs.cmu.edu/code/graphlab/pmf.html)
[4] *LensKit* (http://lenskit.grouplens.org/)
[5] *Waffles* (http://waffles.sourceforge.net/)
[6] *Easyrec* (http://easyrec.org/)
[7] *RecLab* (http://code.richrelevance.com/reclab-core/)
[8] *Crab* (http://muricoca.github.io/crab/)
[9] *Recommenderlab* (http://cran.r-project.org/web/packages/recommenderlab/index.html)
[10] *Jellyfish* (http://hazy.cs.wisc.edu/hazy/victor/download/)
[11] *OpenSlopeOne* (http://code.google.com/p/openslopeone/)
[12] *AppRecommender* (https://github.com/tassia/AppRecommender)

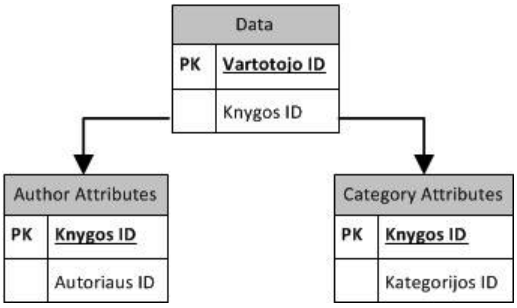
Iš 2 lentelės matyti, kad per paskutinį pusės metų laikotarpį vis dar buvo tobulinamos tik *MyMediaLite*, *Waffles* ir *Jellyfish* sistemos. Daugiausia sistemų sukurta naudojant *C++* ir *Python* programavimo kalbas. Daugiausia algoritmų realizuota *MyMediaLite* sistemoje, nuo jos nedaug atsilieka *GraphLab*. Pažymėtina, kad reitingų prognozavimas galimas visose nagrinėtose sistemose, o produktų rekomendacijos – tik didesnėje pusėje. Tiek algoritmų skaičiumi, tiek atnaujinimais ir kitomis galimybėmis *MyMediaLite* sistema lenkia kitas analizuotas. Todėl ji naudojama algoritmų efektyvumui įvertinti analizuojamoje duomenų aibėje.

Eksperimentinis tyrimas

Siekiant nustatyti, kurie rekomendavimo algoritmai veikia efektyviausiai, atlikti eksperimentai su realiais duomenimis. Paprastai rekomendavimo algoritmo efektyvumui įvertinti naudojami skirtingi įverčiai, nes visų algoritmo savybių ir jo priklausomybės nuo duomenų negalima įvertinti vienu matu. Norit nustatyti efektyviausią algoritmą pagal pasirinktus įverčius, buvo skaičiuojami įverčių rangai bei išvedamas kiekvienam algoritmui vidutinis įverčių rangas, ir kuo jis mažesnis, tuo algoritmo efektyvumas didesnis. Eksperimentiniams tyrimams buvo panaudoti Lietuvoje veikiančios el. knygų parduotuvės duomenys. Nuo el. parduotuvės atidarymo joje buvo siūloma įsigyti 19329 skirtingas knygas, iš kurių 12564 buvo bent kartą nupirktos. Pirkimo istorijoje užfiksuoti 14197 vartotojai, iš kurių 9930 įsigijo bent po vieną produktą. Taip pat buvo pateikta informacija apie 41177 įvykusių pardavimų (be pasikartojimų) ir jų datą. Dėl per mažo įvykusių pardavimų skaičiaus, palyginti su produktų ir vartotojų skaičiumi, atliktuose eksperimentuose pardavimo data nebuvo įvertinama. Tyrimams atlikti konstruojamas duomenų rinkinys, kurį sudaro poros $\{(Vartotojo\ ID, Knygos\ ID)\}$. Šis duomenų rinkinys žymimas *Data*.

Papildomai pateikti duomenų rinkiniai apie kiekvienai knygai priskirtas kategorijas ir autorius žymimi atitinkamai *Category_Attributes* ir *Author_Attributes*.

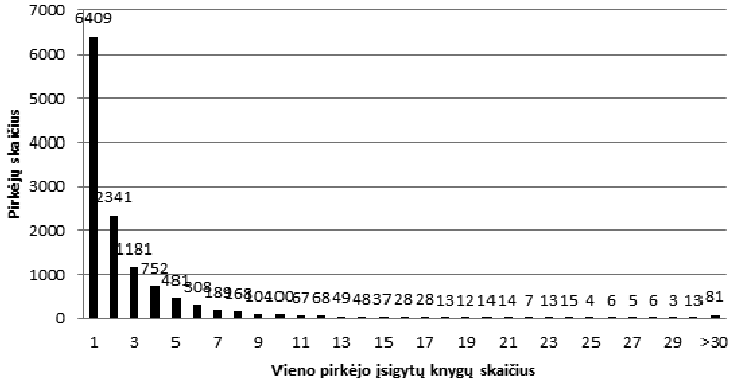
Duomenų rinkinių *Data*, *Category_Attributes* ir *Author_Attributes* struktūra bei tarpusavio ryšiai pateikiami 1 paveiksle.



1 p a v. ***Tyrimui naudotų duomenų rinkinių struktūra ir jų tarpusavio ryšiai***

Tolesniuose skaičiavimuose *Data* duomenų rinkinys transformuojamas į matricą, turinčias 9930 eilučių ir 12564 stulpelius (t. y. tiek, kiek vartotojų nupirko bent vieną knygą ir kiek buvo skirtingų knygų), matrica užpildoma vienetais ir nuliais. Vienetas reiškia, kad vartotojas *i* nupirko knygą *j*. Šios *Data* matricos užpildymas nenuliniais elementais (tankis) lygus $\frac{41177}{14197 \times 19329} = 0{,}015$, tačiau atmetus nė karto nenupirktas knygas bei nė vienos knygos nenupirkusius vartotojus, pasiekiamas $\frac{41177}{9930 \times 12564} = 0{,}033$ užpildymas.

Pakankamai svarbus duomenų rinkinio *Data* analizės aspektas yra pirkėjų pasiskirstymo pagal nupirktų knygų skaičių nustatymas. Toks pasiskirstymas iliustruojamas 2 pav. grafiku.



2 p a v. ***Pirkėjų pasiskirstymas pagal nupirktų knygų skaičių***

Kaip matoma iš pateiktojo grafiko, 50,68 pirkėjų įsigijo tik vieną knygą, o 88,29 sistemos vartotojų yra nupirkę mažiau nei 5 knygas. Esant tokiam vartotojų pasiskirstymui, prognozės nebūna labai tikslios dėl per mažo rekomendacinės sistemos turimo informacijos apie individualų vartotoją kiekio.

Duomenų rinkinyje pastebimos ir anomalijos – čia yra keli vartotojai, nupirkę daugiau nei 500 knygų. Šie vartotojai taip pat gali iškreipti rekomendacijų tikslumą.

Siekiant nustatyti geriausiai su turimais duomenimis veikiančius algoritmus, atlikta 17 rekomendacinėse sistemose dažniausiai taikomų prekių rekomendavimo algoritmų lyginamoji analizė. Visiems šiems algoritmams buvo atlikta po 10 eksperimentų, kai testavimo aibei atsitiktinai parenkami 10 proc. vartotojų, o mokymo aibėje paliekami likusieji. Eksperimentų rezultatų vidurkiai pateikiami 3 lentelėje ir 3 pav. Eksperimentams atlikti buvo naudojamas *DELL Optiplex* 790 kompiuteris.

Trečioje lentelėje *AUC*, *prec@5*, *prec@10*, *MAP*, *recall@5*, *recall@10* ir *NDCG* – rekomendacinių sistemų efektyvumo įverčiai. Mokymo laikas – laikas, kuris buvo reikalingas RS mokymui konkrečiame kompiuteryje, testavimo laikas – laikas, kurio reikėjo RS rezultatų testavimui ir patikrinimui atlikti.

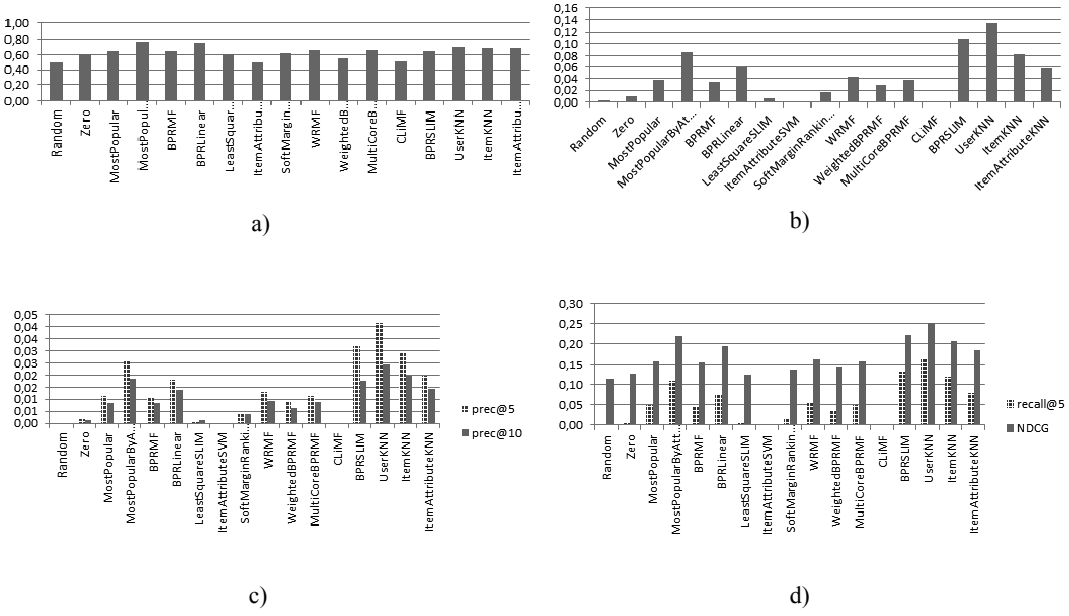
Iš 3 lentelėje ir 3 pav. pateiktų rezultatų galime išskirti turimame duomenų rinkinyje efektyviai ir neefektyviai veikiančius algoritmus. Geriausius rezultatus rodo *UserKNN*, *MostPopularByAttributes*, *BPRSLIM* ir *ItemKNN* algoritmai, tačiau pažymėtina, kad *UserKNN* ir *ItemKNN* algoritmai reikalauja didelių skaičiavimo išteklių.

Savo ruožtu neefektyviausių algoritmų grupei priklauso *LeastSquareSLIM*, *ItemAttributeSVM*, *CliMF*, *WeightedBPRMF* ir *SoftMarginRankingMF* algoritmai, kurie, veikdami šiame duomenų rinkinyje, pateikia prastus rezultatus.

3 l e n t e l ė. ***Rekomendacinės sistemos algoritmų efektyvumo palyginimas***

	RS algoritmų efektyvumo įverčiai						**Mokymo laikas, ms**	**Testavimo laikas, ms**
	AUC	***prec@5***	***prec@10***	***MAP***	***recall@5***	***NDCG***		
Random	0,50	0,00	0,00	0,00	0,00	0,11	0,00	176,91
Zero	0,61	0,00	0,00	0,01	0,01	0,13	0,00	152,21
MostPopular	0,65	0,01	0,01	0,04	0,05	0,16	0,10	173,21
MostPopularByAtt.	**0,77**	0,03	0,02	0,09	0,11	0,22	1,60	238,01
BPRMF	0,65	0,01	0,01	0,03	0,05	0,16	118,91	168,41
BPRLinear	0,76	0,02	0,01	0,06	0,08	0,19	1639,99	201,21
LeastSquareSLIM	0,61	0,00	0,00	0,01	0,00	0,12	244112,00	671,54
ItemAttributeSVM	0,50	0,00	0,00	0,00	0,00	0,00	1032,26	75,00
SoftMarginRankingMF	0,62	0,00	0,00	0,02	0,02	0,14	114,51	184,21
WRMF	0,66	0,01	0,01	0,04	0,05	0,16	524,73	187,81
WeightedBPRMF	0,55	0,01	0,01	0,03	0,04	0,15	108,21	167,91
MultiCoreBPRMF	0,65	0,01	0,01	0,04	0,05	0,16	29,90	209,41
CLiMF	0,50	0,00	0,00	0,00	0,00	0,00	7808,05	29,50
BPRSLIM	0,64	0,03	0,02	0,11	0,13	0,22	4333,45	278,52
UserKNN	0,70	**0,04**	**0,02**	**0,13**	**0,16**	**0,25**	49674,44	2233,73
ItemKNN	0,68	0,03	0,02	0,08	0,12	0,21	125422,70	3787,32
ItemAttributeKNN	0,69	0,02	0,01	0,06	0,08	0,19	129198,60	5252,30

Žymėjimai:	*Random* atskaitos taškas	*Zero* atskaitos taškas	Geras rezultatas	Blogas rezultatas



3 p a v. ***Rekomendacinės sistemos algoritmų efektyvumo palyginimas: a) pagal AUC įvertį; b) pagal MAP įvertį; c) pagal prec@5 ir prec@10 įverčius; d) pagal recall@5 ir NDCG įverčius***

4 l e n t e l ė. ***Kryžminio patvirtinimo eksperimentų rezultatas (k=5)***

Algoritmas	*AUC*	Vieta	*prec@5*	Vieta	*MAP*	Vieta	<…>**	Vidurk.	Galut. vieta
Random	0,499	18	0,001	17	0,004	17		13,18	**17**
Zero	0,552	15	0,001	16	0,008	16		12,27	**16**
MostPopular	0,719	3	0,017	10	0,051	10		7,18	**9**
*MostPopByAtt(C)**	0,781	1	0,032	5	0,100	5		5,27	**3**
BPRMF	0,711	5	0,016	13	0,050	12		9,82	**12**
BPRLinear	0,716	4	0,017	11	0,054	8		9,36	**8**
SoftMarginRankMF	0,650	9	0,005	15	0,021	15		12,18	**14**
WRMF	0,654	8	0,018	8	0,054	9		9,09	**10**
WeightedBPRMF	0,506	16	0,009	14	0,026	14		12,45	**15**
MultiCoreBPRMF	0,722	2	0,017	12	0,051	11		8,73	**11**
CLiMF	0,501	17	0,000	18	0,001	18		15,27	**18**
BPRSLIM	0,635	11	0,038	4	0,120	3		7,64	**4**
UserKNN	0,677	7	0,048	1	0,149	1		5,27	**1**
ItemKNN	0,639	10	0,030	7	0,084	7		9,45	**7**
ItemAttributeKNN	0,633	12	0,017	9	0,047	13		13,00	**13**
*MostPopByAtt(A)**	0,700	6	0,030	6	0,085	6		6,91	**6**
BPRLinear	0,623	14	0,041	3	0,120	4		6,91	**5**
ItemAttributeKNN	0,624	13	0,043	2	0,123	2		7,00	**2**

* Požymiams naudojama kategorija (C) arba autorius (A).

** Taip pat įvertinami ir ranguojami *prec@10*, *recall@5*, *recall@10*, *NDCG* bei *MRR* įverčiai, mokymo ir testavimo laikas bei atminties poreikis. Apskaičiuojamas rangų vidurkis ir pagal jį išrenkami geriausi algoritmai.

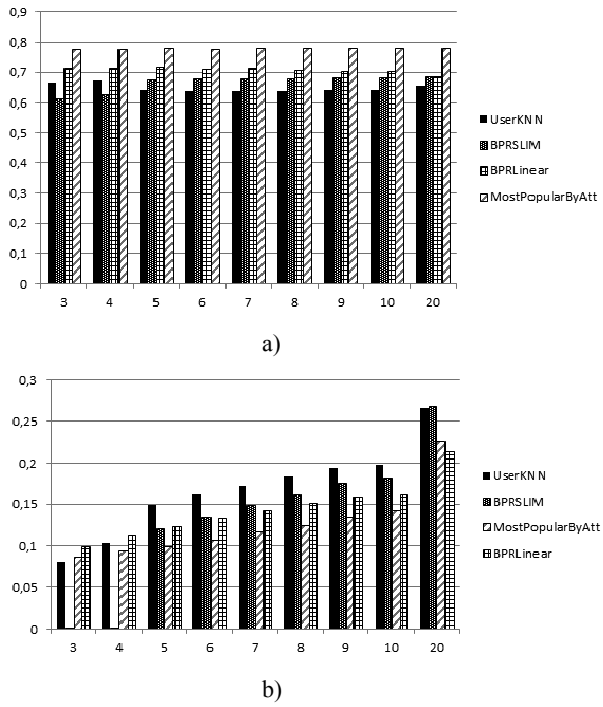
Likusieji algoritmai pateikia vidutiniškus, tačiau geresnius rezultatus nei *Random* ir *Zero* algoritmai, kurie laikomi visų algoritmų vertinimo atskaitos tašku.

Siekiant įvertinti algoritmų efektyvumą kintant vartotojo įsigytų knygų skaičiui, eksperimentai atlikti ir naudojant kryžminio patvirtinimo (angl. *cross_validation*) modelį algoritmui įvertinti. Kryžminis patikrinimo modelis sudaromas turimą duomenų rinkinį atsitiktinai sumaišius ir jį padalijus į k lygių dalių. Tuomet algoritmo mokymui nuosekliai imamos $k-1$, o likusi dalis naudojama testavimui. Toliau vertinimas tęsiamas k kartų imant kitas $k-1$ dalis mokymui ir vieną dalį testavimui.

Rekomendacinių sistemų algoritmo rezultatų vertinimo eksperimentuose buvo taikomas šiek tiek modifikuotas kryžminio patvirtinimo modelis, t. y. iš viso duomenų rinkinio į mokymo aibę buvo perkeliami tų vartotojų duomenys, kurie pirko mažiau knygų nei k reikšmė. Likęs duomenų rinkinys sudalijamas į k lygių dalių. Tuomet viena suskirstyto rinkinio dalis atskiriama kaip testavimo aibė, o likusios priskiriamos mokymo aibei. Eksperimentai atliekami su visomis duomenų rinkinio kopijomis, nuosekliai keičiant dalį, laikomą testavimo aibe.

Mūsų atveju eksperimentai atlikti esant $k = 3; 4; 5; 6; 7; 8; 9; 10; 20$ ir skaičiavimus atliekant su 18 potencialiai tinkamų algoritmų. Gauti rezultatai ranguojami ir tokiu būdu kiekvienam atvejui išrenkami geriausi metodai. Kryžminio patvirtinimo eksperimentų rezultatų dalis, kai $k = 5$, pateikiama 4 lentelėje.

UserKNN, *BPRSLIM*, *BPRLinear* ir *MostPopularBy-Attributes* algoritmų *MAP* įverčio reikšmės augimas, priklausantis nuo k reikšmės, pateikiamas 4 pav. b grafike. Natūralu, kad didėjant vartotojo įsigytų knygų skaičiui algoritmai prognozuoja tiksliau.



4 p a v. ***AUC (a) ir MAP (b) įverčių kaita didėjant k reikšmei***

UserKNN, *BPRSLIM*, *BPRLinear* ir *MostPopularByAttributes* algoritmų *AUC* įverčio reikšmės kaita, priklausanti nuo k reikšmės, pateikiamas 4 pav. a grafike. Pažymėtina, kad šis RS įvertis dviem algoritmams kinta mažai (*MostPopularByAttributes* – paklaidų ribose, *BPRLinear* – fiksuojamas šioks toks sumažėjimas $k \in [10; 20]$ intervale), *BPRSLIM* – fiksuojamas didesnis reikšmės ūgtelėjimas $k \in [3; 5]$ intervale, o *UserKNN* įvairuoja $k \in [3; 5]$ intervale, vėliau nusistovi ties fiksuota reikšme ir vėl paauga $k \in [10; 20]$ intervale.

Tai, kad *MAP* ir *AUC* įverčiai tiems patiems algoritmams rodo skirtingus rezultatus, reiškia būtinumą ranguoti įverčius ir apskaičiuoti vidutinį algoritmą – tik apskaičiavus visus galimus įverčius, nustačius algoritmų rangus ir apskaičiavus vidurkį, galima optimaliai nustatyti algoritmo efektyvumą.

Atlikus visų įverčių tipų, skaičiavimų trukmės ir naudojamų atminties išteklių rangavimą, 5 lentelėje pateikiami geriausiai veikiantys algoritmai penkiais skirtingais kryžminės patikros atvejais.

Iš 5 lentelės matyti, kad *UserKNN* ir *MostPopularByAttributes (category)* algoritmai pirmauja nepriklausomai nuo vartotojo įsigytų knygų skaičiaus. Esant didesniam įsigytų knygų skaičiui, pakankamai efektyvūs yra *BPRSLIM* ir *MostPopularByAttributes*, o naujiems vartotojams, neturintiems ilgos pirkimų istorijos, efektyvūs yra ir *MostPopular* bei *BPRLinear (author)* algoritmai.

Iš analizės rezultatų pastebėtas gana įdomus dalykas: vienas algoritmas (*CLiMF*) šiame duomenų rinkinyje veikia prasčiau nei atsitiktinių pasiūlymų algoritmas (*Random*). Tikėtina, kad šis algoritmas efektyvus tik labai specifiniuose duomenų rinkiniuose. Be to, du algoritmai (*ItemAttributeKNN* ir *WeightedBPRM*) intervale $k \in [3; 8]$ gauna prastesnius rezultatus nei dar vienas vertinimo atskaitos taškas – *Zero* algoritmas. Kai $k = 10$, prastesnius nei *Zero* algoritmas rezultatus demonstruoja ir *SoftMarginRankingMF*. Esant dideliam pirktų knygų skaičiui ($k \approx 20$), visi tirti algoritmai (išskyrus *CLiMF*) demonstruoja geresnius rezultatus už *Random* ir *Zero* atskaitos taškus.

5 l e n t e l ė. ***Geriausiai veikiantys algoritmai 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*

Išvados

Šiame straipsnyje atlikta elektroninės parduotuvės pirkimo įrašų duomenų bazės analizė, pagal iš anksto parinktus kriterijus atlikta atvirojo kodo rekomendacinių sistemų lyginamoji analizė ir eksperimentiškai įvertintas esamų rekomendavimo algoritmų efektyvumas konkrečios duomenų bazės atveju.

Atkreipiamas dėmesys, kad turimo duomenų rinkinio užpildymais nenuliniais elementais yra labai mažas (50,68 pirkėjų įsigijo tik vieną knygą, o 88,29 sistemos vartotojų yra nupirkę mažiau nei 5 knygas), todėl siūlomų algoritmų tikslumas nėra didelis. Kaip rodo eksperimentų suvestinė, algoritmų efektyvumas tiesiogiai priklauso nuo vartotojų įsigytų knygų skaičiaus, vadinasi, šiam skaičiui didėjant RS rekomenduoja tiksliau. Praktikoje šios metodikos veikimą reikėtų įvertinti pasiekus bent 1–3 proc. duomenų rinkinio užpildymą.

Naujiems arba neaktyviems vartotojams (pirkimų istorijoje turintiems tik 1 ar 2 nupirktų knygų įrašus) geriausius rezultatus parodė *MostPopular* ir *MostPopularByAttributes* algoritmai, kai atsižvelgiama ne vien į knygų, bet ir į knygų kategorijų bei autorių populiarumą visoje sistemoje.

Vartotojams, turintiems daugiau nei 3 knygas pirkimų istorijoje, geriausias rekomendacijas generavo *UserKNN* algoritmas, analizuojantis konkretaus vartotojo pirkimų istoriją ir ieškantis panašumų kitų vartotojų pirkimų istorijose. Tačiau reikėtų atkreipti dėmesį, kad šis metodas reikalauja daugiau kompiuterio operatyviosios atminties ir laiko išteklių skaičiavimams, todėl jis negali veikti realiu laiku.

Nustatyta, kad esant didesniam vartotojo pirktų knygų skaičiui (daugiau kaip 8), pakankamai gerus rezultatus demonstravo *BPRSLIM* algoritmas, kurio skaičiavimai vyksta daug greičiau. Šis algoritmas galėtų būti naudojamas kaip alternatyva didelių išteklių reikalaujančiam *UserKNN* algoritmui.

Atlikus lyginamąją rekomendacinių sistemų programinės įrangos analizę pastebėta, kad šiuo metu eksperimentams yra tinkamiausia *MyMediaLite* sistema.

LITERATŪRA

Netflix (2009). Oficialus projekto tinklalapis [interaktyvus] [žiūrėta 2013 m. gegužės 15 d.]. Prieiga per internetą: <http://www.netflixprize.com//leaderboard?showtest=f&limit=1000>.

GANTNER, Z.; DRUMOND, L.; FREUDENTHALER, C.; & SCHMIDT-THIEME, L. (2009). *Bayesian Personalized Ranking for Non-Uniformly Sampled Items*. JMLR: Workshop and Conference Proceedings, no. 18, p. 231–247.

GANTNER, Z.; DRUMOND, L.; FREUDENTHALER, C.; RENDLE, S.; & SCHMIDT-THIEME, L. (2011). Learning Attribute-to-Feature Mappings for Cold-Start Recommendations. In: *ICDM'10 proceedings of the 2010 IEEE International Conference on Data Mining*, p. 176–185.

GONG, S. (2010). A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering. *Journal of Software*, no. 7, p. 745–752.

HUANG, Z.; ZENG, D.; & CHEN, H. (2007). *A Comparative Study of Recommendation Algorithms in ECommerce* [interaktyvus] [žiūrėta 2013 m. gegužės 1 d.]. Prieiga per internetą: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.87.8138&rep=rep1&type=pdf>.

MANNING, C. D.; RAGGAVAN, P.; & SCHUTZE, H. (2009). *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press. 482 p.

McFEE, B.; BERTIN-MAHIEUX, T.; ELLIS, D. P.; & LANCRIET, G. R. (2009). *The million song dataset challenge* [interaktyvus] [žiūrėta 2013 m. gegužės 15 d.]. Prieiga per internetą: <http://www.columbia.edu/~tb2332/Papers/admire12.pdf>.

MIN, S.-H.; & HAN, I. (2005). *Recommender Systems Using Support Vector Machines.* Web Engineering. Springer, p. 387–393.

NING, X., & KARYPIS, G. (2011). *Slim: Sparse linear methods for top-n recommender systems*. ICDM.

ONUMA, K.; TONG, H.; & FALOUSTOS, C. (2009). TANGENT: A Novel, „Surprise-me", Recommendation Algorithm. In: *KDD'09 proceedings*, p. 105–115.

PAN, R.; ZHOU, Y.; CAO, B.; LIU, N. N.; LUKOSE; R. M.; SCHOLZ, M. et al. (2008). *One-class collaborative filtering*. ICDM [interaktyvus] [žiūrėta 2013 m. gegužės 10 d.]. Prieiga per internetą: <http://www.rongpan.net/publications/pan-oneclasscf.pdf>.

RAJARAMAN, A.; LESKOVEC, J.; & ULLMAN, J. D. (2013). *Mining of Massive Datasets.* Cambridge University Press. 326 p.

RASHID, A. M. (2006). ClustKNN: *A highly scalable hybrid model & memory based algorithm* [interaktyvus] [žiūrėta 2013 m. gegužės 10 d.]. Prieiga per internetą: <http://glaros.dtc.umn.edu/gkhome/fetch/papers/clustKNN.pdf>.

RENDLE, S. (2011). Context-Aware Ranking with Factorization Models. New York: Springer. 180 p.

RENDLE, S.; FREUDENTHALER, C.; GANTNER, Z.; & SCHMIDT-THIEME, L. (2009). BPR: Bayesian Personalized Ranking from Implicit Feedback. In: *UAI'09 Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence,* p. 452–461.

RICCI, F.; ROKACH, L. SHAPIRA, B.; & KANTOR, P. B. (2010). *Recommender Systems Handbook.* New York: Springer. 842 p.

SHI, Y.; KARATZOGLOU, A.; BALTRUNAS, L.; LARSON, M.; OLIVIER, N.; & HANJALIC, A. (2012). CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering. In: *RecSys'12 Proceedings of the sixth ACM conference on Recommender systems*, p. 139–146.

TOSCHER, A.; JAHRER, M.; & BELL, R. M. (2009). *The BigChaos Solution to the NetFlix Grand Prize* [interaktyvus] [žiūrėta 2013 m. gegužės 1 d.]. Prieiga per internetą: <http://www.netflixprize.com/assets/GrandPrize2009_BPC_BigChaos.pdf>.

VOZALISs, E.; & MARGARITIS, K. G. (2003). *Analysis of recommender systems algorithms* [interaktyvus] [žiūrėta 2013 m. gegužės 1 d.]. Prieiga per internetą: <http://lsa-svd-application-for-analysis.googlecode.com/svn-history/r72/trunk/LSA/Other/LsaToRead/hercma2003.pdf>.

ZIEGLER, C. N. (2013). *Social Web Artifacts for Boosting Recommenders.* New York: Springer. 187 p.

ANALYSIS OF THE EFFICIENCY OF RECOMMENDATORY SYSTEMS ALGORITHMS IN AN E-BOOKSHOP

Aurimas Rapečka, Virginijus Marcinkevičius, Gintautas Dzemyda

S u m m a r y

In the paper, the efficiency of various recommendatory systems algorithms in a data set of the local e-bookshop is analysed. The key goal of analysis is to determine effective and not effective algorithms in the data set used for analysis. An analytical review of free or open source software of ecommendatory systems is presented. Some comparison criteria are selected. According to the criteria, a comparative analysis of the popular software of ecommendatory systems is made and some experiments with the best evaluated software are done. We have determined here which algorithms are effective in the data set, used for the experiments.