Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Bc. Tomáš Chovaňák

ROZPOZNÁVANIE VZOROV SPRÁVANIA POUŽÍVATEĽOV WEBOVÉHO SÍDLA

Diplomový projekt 2

Študijný program: Informačné systémy
Študijný odbor: 9.2.5
Miesto vypracovania: Ústav informatiky, informačných systémov a softvérového inžinierstva, FIIT STU
Vedúci práce: Ing. Ondrej Kaššák

december 2016

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Rozpoznávanie vzorov správania používateľov webového sídla

Študijný program: Informačné systémy

Autor: Bc. Tomáš Chovaňák

Vedúci diplomovej práce: Ing. Ondrej Kaššák

december 2016

Vzory správania môžeme chápať ako typické a opakujúce sa črty správania používateľov pri návšteve webového sídla. V tejto práci reprezentujeme vzory správania ako frekventované množiny akcií vykonávaných v sedeniach používateľov. Častým zdrojom znalostí o správaní používateľa sú webové logy a v nich zachytené akcie, ktoré vykonali používatelia počas návštevy webového sídla, spájané do používateľských sedení. Celý proces spracovania webových logov, hľadania vzorov a ich analýzy býva označovaný aj ako dolovanie používania Webu (ang. *Web Usage Mining*). Nájdené vzory správania môžu slúžiť napríklad na vytváranie odporúčaní, predikciu zámerov používateľa s využitím na ukladanie stránok do vyrovnávajúcej pamäte, podporu zmeny dizajnu webového sídla, či celkové pochopenie správania používateľov. Väčšina existujúcich riešení tohto procesu hľadá vzory správania zo statických webových logov v množine všetkých používateľov webového sídla.

Táto práca reaguje na súčasný trend personalizácie Webu a zameriavania sa na potreby jednotlivcov a tiež na výzvu dolovania znalostí z rýchlych prúdov dát.Navrhujeme proces, ktorý dokáže spracovávať sedenia používateľov ako prúd dát a objaviť vzory správania, ktoré nehovoria len o správaní globálnej komunity používateľov, ale aj o aktuálnom správaní a zmenách správania menších komunít. Vyhodnocujeme prínos kombinovania skupinových a globálnych vzorov v ich aplikácií na úlohu odporúčania a tiež sledujeme ako dokáže metóda odhaľovať unikátne správanie špecifických skupín používateľov v doméne elektornického výučbového systému a novinového portálu.

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Recognition of Web user's behavioural patterns

Degree Course: Information systems

Author: Bc. Tomáš Chovaňák

Supervisor: Ing. Ondrej Kaššák

2016, December

Behavioural patterns can be understood as typical and repeating features of user's behaviour during their visit of website. In this work we represent behavioural patterns as frequent itemsets of actions frequently taken by user's in user sessions. Frequent source of knowledge about user's behaviour are web logs and actions taken by user's during their visits to website stored there, joined to sessions. Whole process of processing web logs, finding behavioural patterns and their analysis is known as Web Usage Mining. Found behavioural patterns can be used for example to create recommendations, predicting user's intentions (which can be used to cache predicted pages), as support for website design change or complex understanding of website user's behaviour. Most of existing methods of Web Usage Mining search for behavioural patterns common for whole set of web site users in static web logs.

This work responds to actual trend of Web personalization and focusing on needs of individual users. It also responds to challenge to mine knowledge from fast streaming data. We propose solution that is able to process data about user sessions as streaming data and search for behavioral patterns telling us not only about behaviour of global community of users, but also about actual behaviour and changes in behaviour of smaller communities of users. We evaluate contribution of combining global and group behavioural patterns in their application to recommendation task. We also observe way this method is able to detect unique behaviour of specific users groups in domain of e-learning system and news portal.

Obsah

1 Úvod

Snaha získať čo najviac a čo najvýstižnejších znalostí o správaní používateľov webových sídiel je dnes nepochybne veľmi aktuálnou témou. Tak ako to dobrí obchodníci iste vedia, pochopenie zákazníka je nesmierne dôležité. Web je dnes okrem iného vo veľkej miere aj miestom obchodu a reklamy. Aj tu sa ako kedysi na klasickom trhu obchodníci zjednávajú na cene a ponuke produktov so svojimi zákazníkmi, len to robia väčšinou nepriamo a to najmä prostredníctvom analýzy nazbieraných dát.

Samozrejme nie vždy ide iba o obchod. Odhalenie zámerov návštevníka webového sídla a poznanie jeho správania slúži na dobre i jemu samému. Ide o vzájomný profit producentov a konzumentov služieb webového sídla. Napríklad ak ide o elektronický obchod producent chce zarobiť predajom a konzument zas získať produkt. Alebo spravodajský portál, kde producent chce získať čo najväčšiu čítanosť a konzument chce získať informácie.

Správanie človeka nie je deterministické, ale jeho chovanie veľakrát podlieha naučeným vzorom, ktoré si často ani neuvedomuje. Existujú skupiny ľudí, ktorí sa vo viacerých situáciách správajú podobne. Asi však neexistujú dvaja ľudia, ktorí sa správajú celkom rovnako. Príkladom podobného správania skupiny ľudí môže byť spôsob čítania novín. Niektorí ľudia napr. čítajú noviny „odzadu". Nie je to však tak vždy. Pre niekoho je takýto vzor správania silnejší a pre niekoho zas slabší a pre niekoho neplatí vôbec. Vzory správania môžeme chápať napríklad ako často opakujúce sa množiny akcií používateľov. Častým zdrojom znalostí o správaní používateľa sú webové logy a v nich zachytené akcie, ktoré vykonali používatelia počas návštevy webového sídla, spájané do používateľských sedení. V tejto práci vychádzame práve z takto získavanej implicitnej spätnej väzby používateľa. Sedenie používateľa môžeme reprezentovať ako množinu akcií a vzory správania ako frekventované množiny, či sekvencie akcií nájdené v takto upravených logoch.

Cieľom tejto práce je v súlade so súčasným trendom personalizácie Webu a zameriavania sa na potreby jednotlivcov objaviť čo najkvalitnejšie vzory správania, ktoré nebudú hovoriť len o správaní globálnej komunity používateľov, ale aj správaní menších komunít a jednotlivcov k nim patriacich. Dosiahnuť sa to snažíme kombinovaním vzorov správania získaných pre skupiny používateľov s podobným správaním s globálnymi vzormi správania spoločnými pre celú komunitu používateľov webového sídla.

V prvej časti tejto práce sa venujeme najskôr podrobnejšej analýze problematiky dolovania dát na Webe a vo webovom sídle v kapitole 2, kde prehľadne vysvetľujeme samotný pojem dolovania dát na Webe ako súčasť širšieho výskumu a potom opisujeme aj jednotlivé kroky procesu dolovania dát na Webe a tiež možné aplikácie nájdených znalostí. Následne v kapitole 3 opisujeme pojem vzor správania a jeho rôzne reprezentácie v statických datasetoch, priblížujeme niektoré konkrétne spôsoby dolovania vzorov správania v statických dátach reprezentovaných ako frekventované množiny a frekventované sekvencie. V kapitole 4 sa takisto venujeme spôsobom dolovania vzorov správania, ale z prúdu dát. Opisujeme špecifické požiadavky na algoritmy, ktoré spracovávajú prúd dát. Na záver tejto kapitoly ešte približujeme problematiku zhlukovania

v prúde dát a existujúce rámce, ktoré vedia spracovať prúd dát a ponúkajú sadu algoritmov dolovania znalostí z prúdu dát. Tieto znalosti neskôr využívame v kapitole 5, ktorý obsahuje návrh metódy, ktorej hlavným cieľom je hľadať nielen vzory správania spoločne pre celú komunitu používateľov, ale zároveň segmentovať používateľov do skupín podľa ich podobného správania a získavať vzory typické pre identifikované skupiny. Špecifikom navrhovanej metódy je, že reaguje na výzvy súčasného trendu, kedy je potrebné spracovávať neustále rýchlo generované veľké množstva dát tým, že dáta spracováva ako prúd a spĺňa požiadavky, ktoré sú na takýto typ spracovania dát kladené. V kapitole 6 opisujeme implementáciu nášho riešenia. V kapitole 7 opisujeme metodológiu a realizáciu experimentov v ktorých overujeme prínos metódy v aplikácií na odporúčanie a vyhodnocujeme tiež vzťahy medzi globálnymi a skupinovými vzormi. Na záver v kapitole 8 zhŕňame podstatné poznatky, ktoré sme v tejto práci získali a opisujeme možný ďalší smer práce.

2 Dolovanie dát na webe a vo webovom sídle

Dolovanie dát na Webe (ang. Web Mining) znamená získavanie informácií z rozličných aspektov interakcie človeka na Webe. Výskum problematiky dolovania dát na Webe je úzko prepojený aj s ďalšími výskumnými oblasťami ako napr. vyhľadávanie informácií, extrakcia informácií, strojové učenie. Pri interakcií s Webom môžu byť techniky dolovania dát na webe použité na riešenia rôznych problémov ako sa uvádza aj v (Kosala, 2000):

- hľadanie relevantných informácií (špecifickej informácie),
- získavanie nových znalostí z dostupných informácií ,
- personalizácia informácií,
- odhaľovanie záujmov používateľa a zákazníka.

Okrem toho tiež v práci (Kosala, 2000) rozdeľuje problém dolovania dát na Webe do štyroch úloh :

- Hľadanie Webových zdrojov. Teda samotných Webových dokumentov a dát.
- Selekcia špecifických informácií v predspracovaní týchto zdrojov.
- Zovšeobecňovanie. Teda automatizované hľadanie vzorov v individuálnych stránkach, alebo množine prepojených stránok.
- Analýza, validácia a interpretácia nájdených vzorov.

Existujú tri kategórie dolovania Webu podľa časti Webových zdrojov v ktorej sa dolovanie uskutočňuje (Sisodia & Verma, 2012) :

- Dolovanie v obsahu Webu (ang. Web content mining). Predstavuje dolovanie zo samotného obsahu Webových dokumentov (teda vnútornej štruktúry dokumentov).
- Dolovanie štruktúry Webu (ang. Web structure mining). Predstavuje hľadanie modelu štruktúry prepojení medzi stránkami a zvlášť hľadanie zaujímavých vzťahov v tomto modeli.
- Dolovanie dát o používaní Webu (ang. Web usage mining, skr. WUM). Predstavuje dolovanie v dátach generovaných správaním používateľov Webu. Teda dát, ktoré zaznamenávajú interakciu používateľa s Webom. Hľadanie vzorov správania používateľov je teda úlohou vyplývajúcou práve z tejto kategórie dolovania Webu a preto sa jej venujeme podrobnejšie aj v ďalších častiach tejto práce.

WUM je zaujímavou aplikáciou techník dolovania v dátach na získavanie znalostí o správaní používateľov vo webovom sídle. Proces aplikácie WUM na personalizáciu webového sídla je zvyčajne zložený z dvoch fáz a to offline fázy, ktorá sa zaoberá získavaním a prípravou dát, trénovaním modelov a online fázy aplikovania nájdených znalostí. Jednou z výhod použitia personalizácie webového sídla založenej na znalostiach získaných pomocou WUM je to, že vstupom nie sú subjektívne hodnotenia používateľov (explicitná spätná väzba), ale je založený na implicitnej spätnej väzbe, ktorá predstavuje objektívne skutočnosti ako sú akcie používateľa v systéme (Mobasher, Dai, Luo, Sun, & Zhu, 2000).

WUM môže byť považovaný za proces pozostávajúci z 5-tich fáz (Sisodia & Verma, 2012) :

1. Zbieranie dát z Webových zdrojov,

2. Príprava a predspracovanie dát. Bližšie rozoberáme v časti 2.1.
3. Hľadanie vzorov v správaní používateľov. Viacerým zaujímavým metódam hľadania a reprezentovania vzorov sa venujeme v ďalších kapitolách 3 a 4.
4. Analýza a validácia nájdených vzorov a použitie v rôznych aplikáciách. Bližšie rozoberáme v časti 2.2.

2.1 Získavanie a predspracovanie dát

Prvým krokom procesu WUM ako sme si ho uviedli v predchádzajúcej kapitole je získavanie dát o správaní návštevníkov webového sídla. Zdrojmi týchto dát sú najmä logy webových serverov, proxy serverov a prehliadačov, používateľské profily a dáta získané pri registrácií používateľov vo webovom sídle, sedenia používateľov, cookies, dopyty používateľov, kliknutia a akékoľvek iné dáta, ktoré sú výsledkom interakcie používateľa s webovým sídlom (Sisodia & Verma, 2012).

Dátové sady, ktoré používame na overenie navrhovaných metód v tejto práci už prešli viacerými druhmi predspracovania, z toho dôvodu sa otázke predspracovania dát nevenujeme s takým dôrazom na detaily ako v ďalších kapitolách. Skôr sa snažíme poskytnúť všeobecnejší pohľad na rôzne možnosti predspracovania dát.

V tejto práci pracujeme najmä s dátami, ktoré vznikli spracovaním logov webových serverov. Konzorcium W3C poskytuje štandardný odporúčaný formát webových logov[1], ale samozrejme existujú aj iné proprietárne formáty. Použitie štandardného formátu umožňuje využiť niektoré existujúce analytické nástroje, ktoré dokážu tento formát spracovávať[2]. Nazbierané dáta zaznamenané vo webových logoch typicky zdieľajú najmä tieto informácie: dátum, čas, IP klienta, autorizácia klienta, IP servera, port servera, metóda požiadavky, HTTP status kód, referenčná adresa, host, používateľský agent, čas vykonania akcie, počet prenesených byteov a ďalšie.

Webové logy sú typicky veľmi veľké textové súbory (rádovo gigabajty dát). Záznamy v nich sú usporiadané chronologicky. Webové logy môžeme rozdeliť podľa typu informácií, ktoré zachytávajú. Zo špecifickejších logov potom vieme jednoduchšie nachádzať špecifickejšie vzory (Sisodia & Verma, 2012):

- Prístupové logy – zachytávajú detailne informácie o požiadavkách posielaných z prehliadača na webový server.
- Chybové logy – zachytávajú informácie o akýchkoľvek chybách a požiadavkách na webový server, ktoré zlyhali.
- Logy prehliadačov – zachytávajú informácie o prehliadači a operačnom systéme používateľov, ktorý posiela požiadavky na webový server.
- Logy odkazujúcich serverov – zachytávajú URL stránok, ktoré sa odkazujú na skúmanú stránku.

Najmä kvôli obmedzeniam daným špecifikáciou protokolu HTTP sú údaje v bežných webových logov nekompletné a neumožňujú jednoznačnú identifikáciu používateľa a teda ani jeho používateľských sedení. Vo svojej práci Spiliopoulou a kol. (Spiliopoulou, 2003) klasifikuje prístupy k získavaniu dát o používaní Webu do dvoch skupín :

- reaktívne prístupy,
- proaktívne prístupy.

[1] http://www.w3.org/TR/WD-logfile.html

[2] napr. https://www.weblogexpert.com/info/IISLogs.htm

Reaktívny prístup používa existujúce webové logy, ktoré vznikli všeobecne a pravdepodobne bez zámeru použitia pre aplikácie WUM na personalizáciu a teda bez identifikovania používateľov a ich sedení. Reaktívny prístup sa teda snaží transformovať existujúce webové logy, identifikovať používateľov a ich sedenia. Kvôli tomu, že tento prístup nevyžaduje žiaden špecializovaný softvér, ale využíva len všeobecné serverové logy je pomerne rozšírený. Jeho problémom je však, že logy nie sú prispôsobené na takýto typ úlohy (Huntington, Nicholas, & Jamali, 2008).

Proaktívny prístup zbiera dáta spôsobom, ktorý je navrhnutý so zámerom pre konkrétnu aplikáciu. Na najdôležitejšie úlohy z pohľadu prípravy dát WUM teda identifikáciu používateľov a ich sedení využíva explicitné mechanizmy. Na identifikáciu používateľov je možné použiť cookies. Problém s cookies je, že používatelia môžu mať v prehliadači vypnuté ich používanie a v tom prípade je nemožné ich identifikovať. Webový server môže mať tiež implementovaný vlastný mechanizmus identifikácie používateľských sedení (Huntington, Nicholas, & Jamali, 2008). Problémom proaktívneho prístupu je, že používateľ sa musí prihlásiť a nainštalovať softvér, ktorý zabezpečí logovanie. Okrem toho sú tieto metódy niekedy dokonca zakázané ako spyware.

Surové webové logy často obsahujú množstvo irelevantných častí. Dôležitým predpokladom pre zabezpečenie správnosti znalostí získaných pomocou WUM je teda správna príprava dát. V rámci predspracovania dát je často potrebné riešiť problémy uvedené v nasledujúcich častiach.

Očistenie dát

Očistenie dát znamená zbavenie sa nepotrebných záznamov ako napr. neúspešných požiadaviek, či požiadaviek generovaných webovými robotmi, požiadaviek s metódami inými ako GET a POST, suplementárne požiadavky napr. na obrazový obsah stránok alebo rôzne skripty (Srivastava, Garg, & Mishra, 2014).

Identifikácia používateľov

Pri väčšine webových serverov sú používatelia anonymní a problémom je, že nevieme presne určiť priradenie logovaných akcií k týmto používateľom.

Pri reaktívnom prístupe je jednou zo stratégií identifikácie používateľov je aproximácia identity používateľa z IP adries, operačného systému zariadenia z ktorého sa pripája a prehliadača, ktorý používa. Táto stratégia však samozrejme nedokáže presnejšie identifikovať konkrétneho používateľa v prípadoch, keď jedno zariadenie využívajú viacerí používatelia, alebo jeden používateľ používa viacero zariadení. Jednou z ďalších možností spresnenia implicitného identifikovania používateľa, je napr. použitie behaviorálnych profilov používateľa. V tomto prípade však treba získať trénovacie dáta , ktoré majú explicitne uvedené id používateľa. Ďalšou možnosťou identifikácie používateľov je sprístupnenie obsahu webového sídla až po autentifikácií používateľa, čo je však mnoho krát odradzujúcim prvkom. Ďalej je možné identifikovať používateľov pomocou cookies, tu ale existujú scenáre v ktorých takýto prístup nefunguje ako napr. prehliadač, ktorý má zakázané cookies, vymazanie cookies používateľom. (Srivastava, Garg, & Mishra, 2014)

Identifikácia používateľských sedení

Používateľské sedenie predstavuje množinu akcií, ktoré používateľ vykonal v definovanom časovom rozsahu v rámci konkrétneho webového sídla. Jedna z často používaných stratégií identifikácike používateľských sedení je založená na predpoklade, že ďalšie sedenie používateľa začína po prekročení stanoveného časového limitu medzi dvoma vykonanými akciami (často sa využíva limit 30 min). Používané časové limity by mali vychádzať z výstupov štatistických experimentov. (Srivastava, Garg, & Mishra, 2014)

Kompletizácia ciest

Niektoré cesty a prepojenia stránok v identifikovaných používateľských sedeniach môžu byť nekompletné, kvôli tomu, že niektoré prístupy nie sú zaznamenané v logoch. Môže to byť napr. kvôli použitiu tlačidla späť v prehliadači, alebo priamemu zadaniu URL. Preto je niekedy potrebné nájsť existujúce a využité prepojenia medzi jednotlivými stránkami požadovanými v rámci používateľského sedenia. (Srivastava, Garg, & Mishra, 2014)

2.2 Využitie a aplikácie dát o používaní Webu

Posledným krokom procesu WUM je použitie a aplikácia získaných znalostí o správaní používateľov, ktoré ďalej označujeme aj ako vzory správania a v ďalších kapitolách približujeme tento pojem z hľadiska ako vzor správania reprezentujeme v tejto práci. Všeobecným cieľom WUM je zbierať zaujímavé informácie o správaní používateľov a použiť ich na vylepšenie webového portálu z ich hľadiska (Facca & Lanzi, 2003). V tejto podkapitole opisujeme jednotlivé oblasti aplikácie vzorov správania získaných v procese WUM podľa (Facca & Lanzi, 2003).

Personalizácia Webu

Pomocou porovnávania aktuálneho správania s nájdenými vzormi správania je možné využiť vzory správania na predikciu ďalších krokov používateľa, odporúčanie produktov a položiek, či inú adaptáciu webového sídla podľa aktuálnych potrieb používateľa. Existuje viacero prác zaoberajúcich sa aplikáciou vzorov správania na personalizáciu, ktoré sa líšia najmä spôsobom hľadania a reprezentácie vzorov správania. V ohľade na túto aplikáciu je potrebné sa vo WUM procese zaoberať najmä identifikáciou používateľov a ich sedení v rámci fázy predspracovania. V rámci použitia na odporúčanie je zas kľúčové sa sústrediť na spôsob výberu vzorov z ktorých budú generované odporúčania podľa aktuálneho správania používateľa.

Príkladom je metóda WebPUM (Jalali, Mustapha, Nasir Sulaiman, & Mamat, 2010) založená na reprezentácií vzorov správania ako komponentov získaných po aplikácií algoritmu delenia grafu na graf reprezentujúci prepojenia medzi stránkami webového sídla, ktorých váha je odvodená od pohybu používateľov vo webovom sídle v ich sedeniach. Získané vzory správania používajú na generovanie odporúčaní podľa aktuálneho správania používateľov.

Ďalšia metóda na predikciu krokov používateľov je opísaná v práci (Liraki & Harounabadi, 2015). Je to sofistikovaný systém využívajúci Fuzzy C-means zhlukovanie na hľadanie vzorov správania reprezentovaných ako asociačné pravidlá. Vyhodnocujú úspešnosť použitia vzorov správania na predikciu ďalších krokov používateľa.

Ďalšou zaujímavou prácou je (Anandhi & Irfan Ahmed, 2014), kde predstavujú 3 rôzne príspevky k zlepšeniu konkrétnych krokov WUM procesu. Prvým je návrh vhodnejšej heuristiky v rámci fázy predspracovania na identifikáciu sedení používateľov. Druhým príspevkom je použitie zhlukovacieho algoritm DBSCAN na hľadanie vzorov správania. Ten je založený na hustote dátových bodov, čo zvyšuje pravdepodobnosť nájdenia aj menej frekventovaných vzorov, ktoré by napr. pri použití algoritmu hľadania asociačných pravidiel s definovanou hranicou minimálnej podpory boli ignorované. Tretím prínosom je návrh použitia invertovaného indexu pre efektívnejšiu online predikciu správania používateľa.

Ukladanie stránok do dočasnej pamäti a prednačítavanie

To, že sú vzory správania vhodným zdrojom pre metódy predikcie správania používateľa je možné využiť okrem odporúčania položiek aj na zvýšenie výkonnosti a času odozvy webového servera tým, že sa predikované položky na základe aktuálneho správania používateľa, ktoré s vysokou pravdepodobnosťou v ďalších krokoch navštívi prednačítavajú a ukladajú do dočasnej pamäte z ktorej sú potom veľmi rýchlo získané.

V práci (Teng, Chang, & Chen, 2005) opisujú využitie vzorov správania v podobe asociačných pravidiel na predikciu správania používateľa. Navrhujú sofistikovaný algoritmus, ktorý tieto pravidlá efektívne využíva na ukladanie do dočasnej pamäte integrovaný s prednačítavaním stránok na strane klienta. V rámci tohto algoritmu tiež navrhli sofistikovanú funkciu ohodnotenia profitu z prípadného prednačítania konkrétneho objektu uvažujúce mnohé aspekty (napr. veľkosť objektu, počet referencií na objekt, spoľahlivosť použitého asociačného pravidla), vďaka ktorej vedia určiť či daný objekt prednačítať alebo nie.

Podpora pre zlepšovanie dizajnu webového sídla

Ako sme už uviedli dáta o správaní používateľov z webových logov predstavujú implicitnú a teda objektívnu spätnú väzbu, ktorá je veľmi dobrým zdrojom pre účely zlepšovania použiteľnosti webového sídla. Získané znalosti o správaní používateľa môžu byť tak použité na adaptáciu štruktúry a dizajnu webového sídla tak aby sa zrýchlil napr. čas vyhľadania často navštevovaných stránok, položiek a pod. V práci (Perkowitz & Etzioni, 1997) definujú pojem adaptívneho webového sídla a výzvu na automatizovanie procesu reorganizácie štruktúry webového sídla podľa aktuálneho správania používateľov. Práca (Fu, Shih, Creado, & Ju, 2002) je príkladom použitia WUM na adaptáciu dizajnu a štruktúry podľa správania používateľov. Ich cieľom je reorganizovať štruktúru webového sídla tak aby sa dalo používateľom dostať k informáciam pomocou čo najmenšieho počtu kliknutí. Stránky klasifikujú ako indexy a obsahové stránky podľa toho či obsahujú hlavne linky na ďalšie položky resp. skôr textový obsah. Jedným z viacerých príkladov reorganizácie, ktoré opisujú je, že často navštevované stránky klasifikované ako obsahové umiestňujú automaticky vyššie v hierarchickej štruktúre. Malými zmenami štruktúry dosiahnú postupnú adaptáciu štruktúry webového sídla potrebám používateľov.

Porozumenie správaniu používateľa pre podporu biznis rozhodnutí

Pohľad na správanie používateľov webového sídla cez vysokoúrovňové vzory správania môže slúžiť ako základ pre dôležité rozhodnutia v biznis pozadí webového sídla, ako napríklad rozhodnutia o segmentácií trhu. Existujú nástroje podporujúce takéto rozhodovanie od malých lacných nástrojov analyzujúcich webové logy (ako napr. WebLog Expert [3]) po zložitejšie riešenia integrované aj v rámci CRM systémov (napr. WebTrends [4]).

[3] https://www.weblogexpert.com/

[4] https://www.webtrends.com

2.3 Sumarizácia a diskusia

V tejto časti sme opísali, čo znamená dolovanie dát na webe, z akých krokov pozostáva a aké má praktické využitie. Z podkategórií dolovania dát na webe nás ďalej v tejto práci bude zaujímať najmä dolovanie dát o používaní Webu (ang. Web usage mining a skr. WUM). WUM je proces, ktorý je možné automatizovať a pozostáva z troch hlavných krokov: predspracovanie dát, hľadanie vzorov správania a ich následná aplikácia na rôzne účely. V aplikáciach , ktoré sme opísali sú často oddelené kroky predspracovania a hľadania vzorov, ktoré sú vykonávané v rámci offline komponentu, a následná aplikácia nájdených vzorov vykonávaná v rámcií online komponentu. Tu sa vynára výzva na hľadanie takej reprezentácie vzorov správania a metódy ich hľadania aby bolo možné všetky kroky WUM procesu vykonávať v jednom online komponente a dáta o používaní webového sídla spracovávať ako potenciálne nekonečný prúd dát. Práve na túto výzvu sme sa tiež rozhodli v tejto práci reagovať.

Výhodou vzorov správania nájdených pomocou WUM oproti evaluácií webového sídla používateľmi prostredníctvom štúdie, testovania alebo dotazníka je nielen to, že ich hľadanie a analýza môžu byť automatizované, ale hlavne to, že nezáležia od dobrovoľného subjektívneho vstupu používateľov (explicitnej spätnej väzby), ale od objektívnych akcií používateľa vo webovom sidle (Anandhi & Irfan Ahmed, 2014). Sú teda vynikajúcim zdrojom znalostí pre analýzu správania používateľov.

Samotná aplikácia nájdených vzorov môže byť rôzna, uviedli sme niekoľko prác, kde opisujú jednak rôzne metódy hľadania vzorov správania a tiež rôzne spôsoby ich využitia.

3 Existujúce prístupy hľadania a reprezentácie vzorov správania v statických datasetoch

Zatiaľ čo predspracovanie dát je viac-či menej veľmi podobné pri rôznych metódach a aplikáciách WUM, rôzne techniky dolovania dát môžu byť aplikované s cieľom nájsť vzory správania používateľov vo webovom sídle (Sisodia & Verma, 2012).

Vzory správania nie sú nejakým pevne definovaným pojmom a vo webovom sídle si ich môžeme predstavovať ako rôzne skutočnosti napr. sekvencie často po sebe navštívených stránok, typické stránky ktorou začínajú sedenia používateľov, postupnosti navštevovaných sekcií webového sídla, priemerné dĺžky návštev stránok, priemerné dĺžky sedení, a mnoho ďalších identifikovaných pravidelne sa opakujúcich atribútov súvisiacich so správaním používateľa vo webovom sídle.

Odteraz v tejto práci vzory správania používateľov webového sídla rozumieme ako v literatúre bežné uvádzané tzv. frekventované vzory (ang. *frequent pattern*) získané z dát zachytávajúcich správanie používateľov webového sídla (najčastejšie ako klasické webové logy). Ako sa uvádza v práci (Han, Kamber, & Pei, 2012) frekventovaný vzor zachytáva často opakujúce sa vzťahy v dátach. Tieto frekventované vzory môžu mať rôznu podobu – všeobecne to môžu byť často spolu sa vyskytujúce sa neusporiadané podmnožiny z množiny všetkých položiek, usporiadané subsekvencie celých sekvencií akcií, či iné subštruktúry celkovej štruktúry. Frekventované vzory sú dôležitým vstupom pre ďalšie úlohy dolovania v dátach ako hľadanie asociačných pravidiel (ktoré môžu byť tiež chápané ako reprezentácia vzorov správania), klasifikácia, zhlukovanie a ďalšie, čo robí z úlohy ich hľadania veľmi dôležitú oblasť výskumu.

Dôležitým rozdelením frekventovaných vzorov je podľa toho či zachovávajú sekvenčnosť vykonaných akcií alebo nie. Podľa toho možno frekventované vzory, rozdeliť na (Han, Kamber, & Pei, 2012):

- frekventované množiny (ang. *frequent itemsets*), ktoré ignorujú poradie položiek v transakciách. Predstavujú teda často sa spolu vyskytujúce položky v transakciách. Spôsobu ich hľadania sa venujeme v časti 3.1.
- frekventované sekvencie (ang. *frequent sequences*), ktoré zachovávajú sekvenčnosť navštívení položiek v sekvenciách. Spôsobu ich hľadania sa venujeme v časti 3.2.

V časti 3.3 sa v krátkosti venujeme aj iným prístupom reprezentácie vzorov správania používateľov a spôsobu ich získavania napr. zhlukovaním sedení používateľov, rozdeľovaním grafu reprezentujúceho správanie používateľov.

3.1 Dolovanie frekventovaných množín

Nech D je databáza transakcií, Nech $C = \{i_1, i_2, i_3, \dots, i_n\}$ je množina všetkých položiek v D. Položkou i môže byť napríklad záznam o navštívení stránky webového sídla používateľom zachytený vo webových logoch, alebo čokoľvek iné napr. položka v košíku používateľa v internetovom obchode. Transakcia môže predstavovať napríklad sedenie používateľa, alebo košík používateľa v internetovom obchode. Transakcia T je teda množinou položiek. Platí $T \subseteq C$. Frekventovaná množina je akákoľvek množina $A \subset T$ s hodnotou podpory vyššou ako vopred určená hranica minimálnej podpory (ang. *minimum support threshold*). Nech početnosť frekventovanej množiny *A* je *k*. Takúto frekventovanú množinu budeme nazývať *k-množinou*. Podpora množiny je vypočítaná ako absolútna podpora, teda počet transakcií v ktorých sa vyskytuje, alebo ako relatívna podpora, teda pomer počtu výskytov množiny k počtu všetkých transakcií v celej databáze (čo je vlastne pravdepodobnosť výskytu danej množiny v transakcií patriacej uvažovanej databáze).

Dôležitým problémom pri dolovaní frekventovaných množín ako sa uvádza aj v (Han, Kamber, & Pei, 2012) je, že sa generuje často obrovské množstvo množín, ktoré spĺňajú hranicu minimálnej podpory (zvlášť ak je táto nastavená nízko). Je to dôsledok tzv. *apriori* vlastnosti, ktorá hovorí, že akákoľvek podmnožina frekventovanej množiny je tiež frekventovaná.

Ako riešenie tohto problému boli navrhnuté koncepty tzv. uzavretých frekventovaných množín a maximálnych frekventovaných množín ako sa uvádza v (Chi Y. a., 2006):

- Množina *A* je ***maximálna frekventovaná množina*** ak neexistuje množina *B* taká, že $A \subset B$ a zároveň *B* je frekventovaná množina. Nech *M* je množina všetkých maximálnych frekventovaných množín a *F* je množina všetkých frekventovaných množín. Platí, že *M* je oveľa menšie ako *F*. Tiež platí, že vieme z množiny *M* získať *F* ak doplníme do *M* všetky podmnožiny *M* (keďže na základe *apriori* znalosti platí, že všetky tieto podmnožiny budú tiež frekventované). Problémom tohto prístupu je, že dolovaním maximálnych frekventovaných množín sa stráca informácia o podpore jednotlivých dopočítavaných frekventovaných množín.
- Množina *A* je ***uzavretá množina*** ak neexistuje množina *B* taká, že $A \subset B$ a *B* má rovnakú hodnotu podpory ako *A*. Nech *C* je množina všetkých frekventovaných uzavretých množín. Nech *F* je množina všetkých frekventovaných vzorov. Aj pri *C* platí podobne ako pri maximálnych frekventovaných množinách, že veľkosť *C* je stále výrazne menšia ako *F*. Presnejšie platí : $M \subseteq C \subseteq F$ .Tiež platí, že vieme zostrojiť množinu *F* z *C,* keďže každá frekventovaná množina je buď uzavretá, alebo je podmnožinou jednej alebo viacerých frekventovaných uzavretých množín. Takisto vieme dopočítať podporu všetkých frekventovaných množín. Vieme, že podpora frekventovanej množiny bude rovnaká ako maximálna z podpôr uzavretých frekventovaných množín, ktoré ju obsahujú.

Na dolovanie frekventovaných množín bolo od doby kedy bola takáto úloha definovaná navrhnutých viacero algoritmov. Algoritmy hľadania frekventovaných množín v statických datasetoch môžeme klasifikovať do niekoľkých základných skupín podľa toho na akom princípe fungujú (Aggarwal, Bhuiyan, & Hasan, 2014):

- **Algoritmy založené na spájaní množín.** Tieto algoritmy generujú kandidátov na frekventované *(k+1)-množiny* spájaním frekventovaných *k-množín*. Na zmenšenie prehľadávaného priestoru využívajú tzv. apriori princíp, ktorý hovorí že všetky podmnožiny frekventovaných množín sú tiež frekventované množiny. Základným algoritmom tejto skupiny je algoritmus *Apriori* (Agrawal & Srikant, 1994), ktorý má však vážne problémy s výpočtovou náročnosťou. K nemu existuje viacero algoritmov, ktoré sa ho snažia optimalizovať. Algoritmus Apriori a jeho varianty opisujeme podrobnejšie v prílohe A.
- **Algoritmy založené na stromovej štruktúre.** Problém generovania frekventovaných množín je ekvivalentný problému generovania stromovej štruktúry tzv. lexikografického stromu. Koreň stromu je prázdny a uzly stromu reprezentujú lexikograficky usporiadané množiny. Nech $I = \{i_1, \dots i_k\}$ je množina korešpondujúca s uzlom stromu. Potom rodičom uzla je množina $\{i_1, \dots i_{k-1}\}$. Napríklad dve množiny *acdfh* a *acdfg* sú susedné uzly pretože sú potomkovia uzla *acdf*. Ich spojením vznikne kandidátna množina *acdfgh*. V podstate všetky algoritmy založené na apriori princípe možno považovať za algoritmy využívajúce lexikograficky strom. Často sa líšia spôsobom konštrukcie tohto stromu. Algoritmus *Apriori* využíva stratégiu do šírky a lexikografický strom možno uvažovať len implicitne oproti napr. algoritmu *TreeProjection* (Agarwal, Aggarwal, & Prasad, 2001), ktorý existuje vo verzii so stratégiou prehľadávania do šírky aj do hĺbky a explicitne využíva lexikografický strom zobrazujúci proces generovania frekventovaných množín. Niektoré ďalšie algoritmy takisto existujú v modifikáciach podľa toho akú stratégiu prehľadávania stromu používajú (napr. Eclat a modifikácia dEclat (Zaki & Gouda, 2003). Jednou z výhod prístupu hľadania do hĺbky je v prípade dolovania dlhých maximálnych frekventovaných množín ich skoré nachádzanie. Explicitné použitie lexikografického stromu je nápomocné pri vizualizovaní stratégií prehľadávania kandidátnych frekventovaných množín a pochopeniu prínosu rôznych algoritmov.
- **Algoritmy využívajúce vertikálny formát dát**. Nech *tid* označuje identifikátor transakcie, *tidlist* označuje množinu identifikátorov transakcie a *itemset* označuje množinu položiek. Základnou myšlienkou je zefektívnenie počítania podpory pomocou transformácie z horizontálneho formátu dát: <*tid, itemset*> na vertikálny formát dát <*itemset, tidlist*>, kde *tidlist* je zoznam transakcií v ktorých sa nachádza daná množina. Kľúčový princíp je, že je možné vypočítať podporu *k-množiny* vypočítaním prieniku dvoch množín transakcií *tidlist* spájaných (*k-1)-množín.* Príkladom takéhoto algoritmu je *Partition* (Savasere, 1995) a jeho neskorší nasledovník *Eclat* (Zaki M. J., 2000). Podrobnejší opis algoritmu *Eclat*, ktorý patrí do tejto skupiny uvádzame v prílohe A.
- **Algoritmy založené rekurzívnom raste prípon vzorov.** Na rozdiel od väčšiny ostatných prístupov kde sa hľadajú vzory v lexikografickom strome budovaného postupne z predpôn jednotlivých lexikograficky usporiadaných množín, je tento prústup založený na postupnom rozširovaní prípon lexikograficky usporiadaných frekventovaných množín. Prehľadávanie v prístupe s rastom prípon vzorov nie je

principiálne odlišné od klasického prehľadávania s rastom predpôn vzorov (napríklad *TreeProjection* algoritmus) s rozdielom, že položky nie sú usporiadané od tej s najmenšou podporou po najväčšiu ale naopak. Veľmi dôležitým algoritmom patriacim do tejto skupiny je *FP-Growth* (Han, Pei, & Yin, 2000). Jeho základom je vytvorenie vysoko kompaktnej štruktúry s názvom *FP-tree*, reprezentujúcej pôvodné transakčné dáta. Transakčná databáza je rekurzívne rozdeľovaná na menšie projektované časti podľa aktuálnych vzorov. Vzory postupne rastú o lokálne frekventované množiny hľadané v týchto častiach. Podrobne tento algoritmus opisujeme v prílohe A.

- **Algoritmy hľadajúce maximálne a uzavreté frekventované množiny.**
  Veľké množstvo nachádzaných frekventovaných množín býva informačne redundantné. Ako sme už uviedli maximálne a uzavreté frekventované množiny sú kompaktnou reprezentáciou väčšieho množstva frekventovaných množín. Algoritmy z tejto skupiny sa zameriavaju práve na hľadanie týchto kompaktných reprezentácií, čo značne zmenšuje priestor prehľadávania. Príkladom algoritmu na dolovanie maximálnych frekventovaných množín sú *MaxMiner* (Bayardo & Roberto, 1998), *DepthProject* (Agarwal, Aggarwal, & Prasad, 2000), *MAFIA* (Burdick, Calimlim, & Gehrke, 2001). Príkladom algoritmov na dolovanie uzavretých frekventovaných množín je *CLOSET* (Pei, Han, & Mao, 2000) a *CHARM* (Zaki & Hsiao, 2002). V prílohe A bližšie opisujeme algoritmus *CHARM*.

3.2 Dolovanie frekventovaných sekvencií

Úloha dolovania frekventovaných sekvencií bola prvýkrát definovaná v práci (Agrawal & Srikant, 1995). Dôležitým predpokladom je, že spracovávané dáta majú záznamy, ktoré sú sekvenčného charakteru. Príkladom môžu byť transakcie vykonané zákazníkmi elektronického obchodu či sekvencie DNA. Uvádzame definíciu problému dolovania frekventovaných sekvencií tak ako je uvedená v (Agrawal & Srikant, 1995).

Nech *C* je kolekcia všetkých položiek $i \in C$ v dátach. Položkou i môže byť napríklad záznam o navštívení stránky webového sídla používateľom zachytený vo webových logoch. Nech I je neprázdnou množinou položiek, ktoré sa vyskytujú v dátach spoločne (pri sebe v sekvencií). Ak daná množina I obsahuje napríklad položky $a \in C$ a $b \in C$ tak ju môžeme zapísať aj takto: $I = (ab)$. Teda množinu položiek môžeme zapísať ako $I = (i_1, i_2, i_3, \dots, i_m)$. Sekvencia S je zoradeným zoznamom takýchto množín položiek. Teda $S = < s_1, s_2, s_3, \dots, s_n >$ kde s_i označuje v poradí i-tu množinu položiek. $S' = < s'_1, s'_2, s'_3, \dots, s'_n >$ je subsekvenciou $S = < s_1, s_2, s_3, \dots, s_m >$ ak existujú také indexy $1 \leq i_1, i_2, i_3, \dots \leq m$, že $s'_1 \subseteq s_{i_1}, s'_2 \subseteq s_{i_2}, s'_3 \subseteq s_{i_3}, \dots, s'_n \subseteq s_{i_n}$. Nech D je databáza sekvencií. Každá sekvencia $S \in D$ by mala obsahovať minimálne množinu I a tiež identifikátor používateľa, ktorý danú transakciu vykonal a čas uskutočnenia transakcie.

Úlohou je nájsť všetky frekventované sekvencie nachádzajúce sa v databáze *D*. Výpočet podpory frekventovanej sekvencie (vzoru) je vlastne výpočtom frekvencie výskytov danej sekvencie v *D*. Sekvencia je považovaná za frekventovanú (a teda možno ju označiť ako sekvenčný vzor) ak má podporu presahujúcu vopred určenú hranicu minimálnej podpory (ang. *minimum support threshold*).

Sekvenčný vzor s dĺžkou *l* sa nazýva *l-sekvenčný vzor.* Množina frekventovaných *l-sekvencií* nech je F_l. Ak neexistuje žiadna nadsekvencia sekvenčného vzoru $\alpha \in D$ s rovnakou podporou ako *α*, tak *α* je *uzavretý sekvenčný vzor* v databáze *D*. Sekvenčný vzor *α* je *maximálny sekvenčný vzor* ak ho neobsahuje žiadny iný sekvenčný vzor v databáze *D* (Shen, Wang, & Han, 2014).

ID	Sekvencia
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

Tabuľka 1 Príklad databázy sekvencií z (Shen, Wang, & Han, 2014).

Tabuľka 1 je ilustráciou možnej databázy sekvencií *D*. Nech je hodnota minimálnej podpory 2. Keďže sekvencie 1 a 3 obsahujú subsekvenciu $s = < (ab)c >$ tak *s* má hodnotu podpory 2 a je teda sekvenčným vzorom s dĺžkou 3. Sekvencia $\beta = < (ab)dc >$ je zas príkladom uzavretého sekvenčného vzoru keďže neexistuje taká nadsekvencia z *D*, ktorá by mala rovnakú hodnotu podpory (Shen, Wang, & Han, 2014).

Na dolovanie frekventovaných sekvencií bolo od doby kedy bola takáto úloha definovaná navrhnutých viacero algoritmov. Algoritmy hľadania frekventovaných sekvencií v statických datasetoch možno rozdeliť do základných skupín podľa toho na akom princípe fungujú (Shen, Wang, & Han, 2014) :

- **Algoritmy založené na apriori princípe**. Ten hovorí, že všetky podmnožiny frekventovaných sekvencií sú tiež frekventované sekvencie. Ďalej možno algoritmy založené na tomto princípe rozdeliť podľa toho aký formát dát využívajú:
    - **algoritmy využívajúce horizontálny formát dát** (pozri Tabuľka 1). Patria tu napr. algoritmy *AprioriAll*, ktorý bol navrhnutý v práci (Agrawal & Srikant, 1995) ako jedno z prvých riešení problému hľadania frekventovaných sekvencií a je vlastne rozšírením techník používaných v známom algoritme Apriori určenom na hľadanie frekventovaných množín a asociačných pravidiel. Jedno z jeho vylepšení je algoritmus *GSP* (Srikant & Agrawal, 1996). V prílohe B uvádzame podrobnejší opis k algoritmu GSP.
    - **algoritmy využívajúce vertikálny formát dát**:
      $< item: (id\ sekvencie\ S, id\ množiny\ I) >$.
      Patria tu napr. algoritmy *SPADE* (Zaki M. J., 2001), *SPAM* (Ayres, Flannick, Gehrke, & Yiu, 2002). V prílohe B uvádzame podrobnejší opis k algoritmu *SPADE*.
- **Algoritmy založené na raste vzorov**. Pomocou známeho prístupu rozdeľuj a panuj je sekvenčná databáza rekurzívne rozdeľovaná na menšie projektované časti, podľa aktuálnych vzorov. Vzory postupne rastú o lokálne frekventované sekvencie hľadané v týchto častiach. Patria tu napr. algoritmy *FreeSpan* (Han, et al., 2000) a *PrefixSpan* (Pei J. , et al., 2001). V prílohe B uvádzame podrobnejší opis k algoritmu *PrefixSpan*.
- **Rozširujúce metódy.** Existuje viacero ďalších metód, ktoré rozširujú klasické prístupy k dolovaniu sekvencií a snažia sa vylepšiť ich problémy s ktorými sa v praxi stýkajú. V krátkosti opisujeme princípy niektorých z týchto zaujímavých algoritmov v časti 3.2.1.

3.2.1 Rozširujúce prístupy k dolovaniu frekventovaných sekvencií

V tejto časti v krátkosti opisujeme niekoľko ďalších algoritmov a prístupov, ktoré rozširujú základné prístupy k dolovaniu frekventovaných sekvencií a snažia sa vyriešiť niektoré problémy s ktorými sa v praxi stýkajú.

Algoritmy hľadajúce uzavreté frekventované sekvencie

Podobne ako v oblasti dolovania frekventovaných množín (pozri časť 0) sú to algoritmy ako *CHARM, CLOSET* , tak v oblasti dolovania frekventovaných sekvencií boli navrhnuté napr. algoritmy *CloSpan* a *BIDE*. Dolovanie uzavretých frekventovaných sekvencií výrazne zmenšuje oblasť prehľadávania a zvýšuje výkonnosť algoritmu najmä pri nízko nastavenej hladine minimálnej podpory a dolovaní dlhých frekventovaných sekvencií. (Shen, Wang, & Han, 2014)

Algoritmy hľadajúce viacdimenzionálne sekvencie

Štandardné algoritmy hľadajú frekventované sekvencie v jedno alebo dvoj dimenzionálnom priestore. V reálnych situáciach bývajú sekvencie asociované s rôznymi atribútmi, ktoré formujú multidimenzionálny priestor. Napr. sekvencie nákupov zákazníkov sú asociované napr. s regiónom, časom, osobitnou zákazníckou skupinou a iné. Príkladom je algoritmus *Uni-Seq* navrhnutý v práci (Pinto, et al., 2001), kde multidimenzionálnu informáciu vkladajú ako ďalšiu množinu položiek do sekvencie. Touto transformáciou sa stane databáza sekvencií klasickou jednodimenzionálnou sekvenčnou databázou na ktorú v tomto prípade aplikujú algoritmus *PrefixSpan*. Výsledkom môžu byť zaujímavé znalosti ako napr. zistenie, že jedna skupina zákazníkov má celkom odlišné správanie ako iná (Shen, Wang, & Han, 2014).

Aproximatívne metódy

Bežné metódy hľadania sekvenčných vzorov majú problémy s databázami obsahujúcimi mnoho dlhých sekvencií a šumu, kvôli čomu generujú množstvo krátkych a triviálnych vzorov, ale nedokážu nájsť skutočne zaujímavé vzory. V práci (Kum, Pei, Wang, & Duncan, 2003) navrhli pojem *dolovanie aproximatívnych sekvenčných vzorov.* Ide o to, že namiesto hľadania exaktných vzorov sa hľadajú vzory *približne* zdieľané mnohými sekvenciami v databáze. Nimi navrhnutý algoritmus *ApproxMAP* v prvom kroku zhlukuje sekvencie na základe podobnosti (podobnosť určuje editačná vzdialenosť medzi sekvenciami). Pre každý zhluk potom vygenerujú najdlhší aproximatívny sekvenčný vzor, ktorý nazývajú aj vzor zhody (*ang. consensus pattern*). Každej sekvencií v zhluku sú pridané tzv. prázdne položky tak aby mali všetky sekvencie v rámci zhluku rovnakú dĺžku a tak aby vzniklo tzv. globálne optimálne zarovnanie sekvencií (*ang. global sequence alignment*). To znamená, že suma editačných vzdialenosti medzi všetkými dvojicami sekvencií je minimalizovaná. Z takto zarovnaných sekvencií je odvodená váhovaná sekvencia položiek, ktorá sumarizuje počty výskytov jednotlivých položiek v zarovnaných sekvenciách. Tento vzor aj s pridanou informáciou o váhe jednotlivých položiek je kompaktnou reprezentáciou všetkých sekvencií v danom zhluku (Shen, Wang, & Han, 2014).

Hľadanie k-najlepších uzavretých vzorov

Ako sme to už viackrát spomenuli tak dolovanie uzavretých vzorov prináša výrazne zmenšenie prehľadávaného priestoru a pritom sa nestráca žiadna informácia, pretože možno odvodiť všetky ostatné vzory aj s ich hodnotou podpory. Čo sa týka väčšiny algoritmov dolovania frekventovaných vzorov problematické je vhodné nastavenie hladiny minimálnej podpory, pretože ak je príliš vysoká nemusia sa nájsť dôležité vzory a ak je príliš malá môže dôjsť k priveľkému výpočtovému zaťaženiu. Tento problém sa snaží algoritmus *TSP* navrhnutý v práci (Tzvetkov, Yan, & Han, 2005) riešiť nasledovným spôsobom. Namiesto vstupného parametra minimálnej podpory vyžaduje ako vstupný parameter minimálnu dĺžku vzorov a *k* čo je požadovaný počet uzavretých vzorov, ktoré majú byť nájdené. Algoritmus veľmi skoro dokáže nájsť uzavreté vzory s najvyššou podporou a postupne hľadať ďalšie. (Shen, Wang, & Han, 2014)

3.3 Iné reprezentácie vzorov správania a prístupy k ich získavaniu

V tejto časti opíšeme niektoré vybrané ďalšie zaujímavé prístupy k reprezentácií vzorov správania používateľov a prístupov k ich získavaniu.

3.3.1 Algoritmy delenia grafu

V práci (Jalali, Mustapha, Nasir Sulaiman, & Mamat, 2010) sa uvádza zaujímavý prístup hľadania a aplikovania vzorov správania. Z databázy predspracovaných dát zhlukujú používateľské sedenia podľa ich podobných vlastností. Z týchto zhlukov potom vytvára tzv. navigačné vzory, čo je v podstate istá reprezentácia vzorov správania používateľov. Popisovaný proces zhlukovania využíva algoritmus pozostávajúci z týchto základných krokov:

1. Pre každý pár stránok sa počíta stupeň prepojenia na základe vzorca uvažujúceho *frekvencie* spoločného výskytu týchto stránok v používateľských sedeniach a časového rozdielu medzi prístupmi k jednotlivým stránkam v rámci daného sedenia.

$$W_{a,b} = \frac{2 \times TC_{ab} \times FC_{ab}}{TC_{ab} + FC_{ab}}$$

   - $TC_{a,b}$ je stupeň časového prepojenia medzi návštevami každých dvoch stránok webového sídla.
   - $FC_{a,b}$ meria frekvenciu výskytu stránok *a* aj *b* v každom sedení.
   - $W_{a,b}$ stupeň prepojenia medzi stránkami je reprezentovaný ako harmonický priemer časového prepojenia a frekvencie spoločného výskytu stránok v používateľskom sedení.

$$TC_{a,b} = \frac{\sum_{i=1}^{N} \frac{T_i}{T_{ab}} \times \frac{f_a(k)}{f_b(k)}}{\sum_{i=1}^{N} \frac{T_i}{T_{ab}}}$$

   - T_i je časová dĺžka trvania *i-teho* sedenia, ktoré obsahuje návštevy oboch stránok *a* aj *b*.
   - T_{ab} je časový rozdiel medzi časmi navštívenia stránok *a* a *b* v rámci *i*-teho sedenia.
   - Jedna z možných reprezentácií funkcie *f* je $f_a(k) = k$ ak sa stránka *a* nachádza na k-tej pozícií v rámci sedenia. Ak by sme napríklad chceli zvýšiť dôležitosť pozície stránky v rámci sedenia môžeme použiť funkciu *f* kde $f(k) = k^2$.

$$FC_{a,b} = \frac{N_{ab}}{\max\{N_a, N_b\}}$$

   - N_{ab} je počet sedení obsahujúcich stránku *a* aj *b*. N_a a podobne aj N_b je počet sedení obsahujúcich len stránku *a* resp. stránku *b*.

2. Vytvorenie neorientovaného grafu reprezentovaného maticou susedností. Počet hrán je obmedzený danými konštantnými hraničnými hodnotami.
3. Algoritmy delenia grafu sú využité na rozdelenie grafu na K oddelených častí (vzorov správania) obsahujúcich silno prepojené stránky pričom spojenia medzi identifikovanými časťami sú slabé.

Získané vzory správania ďalej používajú na generovanie odporúčaní podľa aktuálneho správania používateľov.

3.3.2 Dolovanie frekventovaných super sekvencií

V práci (Yu & Korkmaz, 2015) skúmajú novú formu hľadania frekventovaných vzoroch v sekvenčných dátach, ktorú nazývajú dolovanie frekventovaných super-sekvencií *(ang. super-sequence frequent pattern mining*). Všetky doposiaľ uvedené algoritmy sa dajú použiť na hľadanie frekventovaných subsekvencií zo sekvencií, teda hľadanie častí týchto sekvencií, ktoré sa vyskytujú v mnohých sekvenciách. Super-sekvencie sú také sekvencie, ktoré môžu obsahovať spoločné časti z viacerých sekvencií, teda spájajú viaceré nájdene frekventované subsekvencie. Super-sekvencie odhaľujú skryté štruktúry ukryté medzi viacerými sekvenciami v databáze. Autori uvádzajú analógiu tohto problému s problémom hľadania najťažšej cesty v grafe. Tento graf si možno predstaviť ako vážený orientovaný graf, kde uzly predstavujú navštívené stránky webového sídla a hrany medzi dvoma uzlami vzniknú ak existuje taká po sebe idúca dvojica navštívených stránok v niektorom zo sedení. Váha danej hrany predstavuje počet takýchto dvojíc. Uvedený problém hľadania najťažšej cesty v grafe je známy ako *NP-zložitý.* Autori v tejto práci navrhli heuristiku, ktorá pomocou techník dynamického programovania pomáha efektívnejšie riešiť tento problém.

3.3.3 Získavanie vzorov správanie pomocou zhlukovania

Táto časť bude obsahovať jeden dva odseky o práci:
(Vellingiri, Kaliraj, Satheeshkumar, & Parthiban, 2015)

3.3.4 SOM

Táto časť bude obsahovať jeden dva odseky o práci:
(Etminani, Delui, Yenehsari, & Rouhani, 2009)

3.4 Sumarizácia a diskusia

V úvode tejto kapitoly sme definovali pojem vzory správania, ktoré v tejto práci chápeme ako v literatúre bežne uvádzané frekventované vzory (*ang. frequent patterns*). Frekventované vzory možno rozdeliť na frekventované sekvencie a frekventované množiny podľa toho či zachovávajú sekvenčnú postupnosť položiek v transakciách alebo nie.

Definovali sme problematiku hľadania frekventovaných množín a uviedli niekoľko existujúcich algoritmov, ktoré túto problematiku riešia. V prílohe A tiež uvádzame podrobnejší opis jednotlivých algoritmov. Riešime tam tiež aké problémy s výpočtovou náročnosťou má prvý navrhnutý algoritmus v tejto problematike s názvom *Apriori*, ktorý má najme pri generovaní kandidátov na frekventované množiny a počítaní ich podpory, čo si vyžaduje viacero prechodov databázou transakcií. Ďalšie prístupy sa snažia tieto problémy vyriešiť ako napr. algoritmus *FP-Growth*, ktorý doluje frekvantované množiny bez potreby generovania kandidátnych frekventovaných množín pomocou kompaktnej dátovej štruktúry s názvom FP-Tree, alebo algoritmus *Eclat*, ktorý zmenšuje nároky na pamäť vďaka rozdeleniu dát na časti, ktoré môžu byť spracované nezávisle. Algoritmus *CHARM*, ktorý sme použili aj ako súčasť riešenia navrhovaného v kapitole 5 ešte výraznejšie zmenšuje problémový priestor vďaka tomu, že hľadá len uzavreté frekventované množiny, ktoré kompaktne reprezentujú väčšie množstvo inak redundantných vzorov.

Ďalej sme definovali sofistikovanejší spôsob reprezentácie a dolovania frekventovaných vzorov ako frekventovaných sekvencií. V prílohe B tiež bližšie opisujeme algoritmy *GSP* a *SPADE*. Opísali sme tiež niektoré ďalšie zaujímavé algoritmy a prístupy, ktoré rozširujú základné prístupy k dolovaniu frekventovaných sekvencií a snažia sa vyriešiť niektoré problémy s ktorými sa v praxi stýkajú. Na konci tejto kapitoly uviedli niekoľko ďalších zaujímavých prístupov k reprezentácií vzorov správania používateľov a prístupov k ich získavaniu napr. pomocou zhlukovania, neurónových sieti či rozdeľovania grafu.

Dôležitou výzvou v oblasti dolovania frekventovaných vzorov je výpočtová náročnosť samotnej úlohy. Už aj pre stredne veľké datasety je samotný priestor prehľadávania obrovský a rastie exponenciálne s dĺžkou transakcií v datasete. (Aggarwal, Bhuiyan, & Hasan, 2014). Opisované algoritmy sa zameriavajú na čo najlepšie riešenie práve čo sa týka zmenšovania prehľadávaného priestoru a zmenšovania výpočtovej náročnosti problému. Tu sa otvárajú aj ďalšie výzvy ako riešiť tento problém pre veľké a rýchle prúdy dát, čomu sa venujeme v ďalšej kapitole tejto práce.

4 Existujúce prístupy hľadania a reprezentácia vzorov správania v prúde dát

V úvode kapitoly 3 sme už uviedli, že v tejto práci budeme pod pojmom vzory správania chápať v literatúre bežne uvádzane tzv. frekventované vzory, čo môžu byť buď frekventované množiny alebo frekventované sekvencie. V tejto časti prenesieme tento problém do oblasti spracovania prúdu dát, kde sa vynárajú nové problémy a výzvy.

V dnešnej dobe v súvislosti s neustálym rastom produkovaných dát pribúdajú aj výzvy na ich efektívne spracovanie. Jedným zo známych vlastností s ktorou sú spájané problémy spracovania veľkých dát je rýchlosť (*ang. Velocity*). Metódy na spracovávanie prúdov dát sa musia vysporiadať s týmto problémom neustáleho rýchleho pribúdania nových inštancií dát. Tie môžu pribúdať teoreticky do nekonečného objemu, ktorý by nebol spracovateľný bežnými metódami určenými pre statické dáta. Získavanie znalostí z prúdov dát je interdisciplinárnou výskumnou oblasťou zaoberajúcou sa metódami a algoritmami získavania znalostí z nestálych dočasných prúdov dát (Krempl, et al., 2014).

Algoritmy spracovania prúdov dát sú stavané na modeloch, ktoré sú inkrementálne aktualizované s prichádzajúcimi inštanciami. Tradičné metódy často vyžadujú viaceré prechody inštanciami v databáze, v prúde dát je každá inštancia spracovávaná práve raz. Počas vývoja dátového prúdu v čase môže nastávať aj k tzv. konceptuálnemu posunu. Konceptuálny posun je významným problémom v dolovaní prúdov dát a vyplýva zo zmien reálneho prostredia v ktorom dáta vznikajú a jeho nárokov. Faktory vyvolávajúce takéto zmeny sú často skryté a neznáme. Konceptuálny posun spôsobuje vážny problém pri modeloch vytvorených zo starých dát pred týmto posunom, ktoré sa samozrejme v takom prípade správajú nesprávne. Konceptuálny posun môžeme klasifikovať podľa rýchlosti zmien ako náhly alebo postupný. Metóda spracovania prúdových dát, by mala takéto zmeny vedieť identifikovať, odlíšiť ich od bežného šumu v dátach a adaptovať sa čo najrýchlejšie (Tsymbal, 2004). Napr. pri dolovaní frekventovaných množín z prúdu dát môže konceptuálny posun celkom zmeniť frekventované množiny aké sa v dátach vyskytujú – k týmto zmenám môže dokonca dochádzať sezónne (napríklad často spolu nakupované položky v internetovom obchode s oblečením budú celkom iné v zime ako v lete).

Čo sa týka samotnej úlohy dolovania frekventovaných vzorov v prúde dát, tak existujú tieto tri problémy (Lee, Jin, & Agrawal, 2014):

1. Exponenciálny nárast počtu prehľadávaných vzorov. Základný algoritmus pre statické dáta *Apriori* spracuje $O(k^2)$ podmnožín aby našiel jeden vzor o dĺžke k.
2. Pamäťová náročnosť, kvôli veľkému priestoru prehľadávania.
3. Potreba balansovania požiadaviek na presnosť a požiadaviek na efektívnosť. Spresňovanie výsledkov znamená viac zabranej pamäte a viac výpočtového času. Používateľovi by teda malo byť umožnené nastaviť tento balans podľa jeho aktuálnych potrieb.

V časti 4.1 sa budeme venovať podrobnejšie problematike dolovania frekventovaných množín z prúdu dát a bližšie si opíšeme algoritmus *IncMine*, ktorý využívame aj v navrhovanej metóde v kapitole 5. Ďalšie zaujímavé algoritmy ako napr. Closet, Moment opisujeme podrobnejšie v prílohe C.

V časti 4.2 trocha odbočujeme a opisujeme niekoľko možností zhlukovania v prúde dát, keďže významnou súčasťou metódy navrhovanej v kapitole 5 bude aj komponent zabezpečujúci práve tento typ úlohy.

V závere tejto kapitoly (4.3) ešte opisujeme existujúce rámce pre spracovanie prúdu dát, z ktorých jeden používame aj na implementáciu našej metódy.

4.1 Dolovanie frekventovaných množín v prúde dát

V časti 3.1 sme už uviedli úlohu dolovania frekventovaných množín v klasickom statickom prostredí. V tejto časti sa budeme venovať algoritmom, ktorú sa musia vysporiadať s novými požiadavkami, ktoré spracovanie prúdu dát vyžaduje. Rýchlosť pribúdania nových dátových inštancií vylučuje možnosť ukladania celej ich histórie do pamäte a opätovné prechádzanie. Ani ukladanie väčších častí prúdu dát do pamäte nie je typicky vhodným riešením (Calders, Dexters, Gillis, & Goethals, 2014).

Na vysporiadanie sa s týmito problémami bolo navrhnutých niekoľko prístupov. Väčšinou sú založené na princípe hľadania frekventovaných množíén v rámci okna obsahujúceho posledných *w* transakcií. Toto okno môže byť (Quadrana, Bifet, & Gavaldà, An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System, 2015):

- **Míľnikové okno** (ang. *landmark window*) $W = < T_1, T_2, \dots, T_t >$, kde T_i predstavuje transakcie v jednej dávke. T_1 je najstaršia časť okna a T_t je najnovšia časť.
- **Posúvajúce sa okno** (*ang. sliding window*) $W = < T_{t-w+1}, \dots, T_t >$, kde *w* je veľkosť posúvajúceho sa okna.
    - S fixovaným časovým intervalom (ang. *time-sensitive*). Počet obsiahnutých transakcií *w* sa rovná počtu transakcií, ktoré prišli v prúde dát za poslednú časovú jednotku t_w.
    - Pevne fixovanej dĺžky (*ang. transaction sensitive*), obsahujúce vždy rovnaký počet transakcií.

Prístupy hľadania frekventovaných množín možno ďalej klasifikovať podľa toho koľko transakcií je vstupom do procedúry aktualizácie aktuálnych frekventovaných množín (Quadrana, Bifet, & Gavaldà, An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System, 2015):

- Prístupy s aktualizáciou v dávkach,
- Prístupy s aktualizáciou v každej transakcií.

A tiež podľa toho či výstupom sú:

- exaktné frekventované množiny (teda také aké by našli aj statické metódy),
- alebo aproximatívne frekventované množiny.

Hľadanie exaktných frekventovaných množín v prúde dát môže byť nezvládnuteľné z hľadiska pamäťovej a výpočtovej náročnosti keďže v takom prípade je nutné urdžiavať v pamäti všetky nefrekventované množiny od začiatku behu (Quadrana, Bifet, & Gavaldà, An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System, 2015).

V tejto kapitole sa bližšie venujeme len algoritmom dolovania uzavretých frekventovaných množín (pozri časť 3.1), keďže sú kompletnou a neredundantnou reprezentáciou všetkých frekventovaných množín a zároveň, ktorá výrazne napomáha k zmenšeniu prehľadávaného priestoru.

4.1.1 Algoritmy dolovania uzavretých frekventovaných množín v prude dát – porovnanie základných charakteristík

Uvádzame niektoré algoritmy dolovania frekventovaných množín v prúde dát. Existuje viacero algoritmov. My sa zameriavame na algoritmy, ktoré sú určené na dolovanie uzavretých frekventovaných množín, ktoré sú kompletnou a neredundantnou reprezentáciou všetkých frekventovaných množín, ktorá zároveň výrazne napomáha k zmenšeniu prehľadávaného priestoru. Takisto sme sa zamerali na algoritmy využívajúce pohybujúce sa okno, ktoré oproti míľnikovému oknu sa dokáže lepšie vysporiadať s konceptuálnym posunom. (Quadrana & Mestre, 2012)

Prvým navrhnutým algoritmom na inkrementálne dolovanie uzavretých frekventovaných množín MOMENT bol navrhnutý v práci (Chi, Wang, Yu, & Muntz, 2004). Tento algoritmus hľadá exaktné frekventované množiny a využíva pohybujúce sa okno a aktualizáciu v každej transakci. Informácie o všetkých o jednotlivých množinách sa ukladajú do špeciálnej stromovej dátovej štruktúry CET (*ang. closed enumeration tree*), ktorá rozdeľuje do rôznych typov uzlov buď nefrekventované množiny, potenciálne frekventované množiny, a uzavreté frekventované množiny. Na ukladanie informácií o všetkých transakciách v aktuálnom okne využíva dátovú štruktúru *FP-tree*, podobne ako v algoritme *FP-Growth* navrhnutý v práci (Han, Pei, & Yin, 2000) na dolovanie frekventovaných množín v statických datasetoch ale bez orezávania nefrekventovaných množín. To, že algoritmus MOMENT musí ukladať všetky doteraz nájdené aj nefrekventované množiny aj napriek uvedeným kompaktným štruktúram spôsobuje značné zaťaženie pamäte. Podrobnejšie ho opisujeme v prílohe C. (Quadrana & Mestre, 2012)

Algoritmus NEWMOMENT reprezentuje transakcie a množiny pomocou bitových sekvencií. Pohyb okna uskutočňuje pomocou bitovej operácie ľavého posunu. Počítanie podpôr množín je vykonávané pomocou bitovej operácie súčinu. Bitová reprezentácia týchto štruktúr zefektívňuje a zlepšuje výkonnosť oproti pôvodnemu algoritmu MOMENT. (Quadrana & Mestre, 2012).

Ďalším algoritmom, ktorý využíva celkom odlišný prístup a iné dátové štruktúry na udržanie informácie o množinách je CLOSTREAM. Podobne ako MOMENT hľadá exaktnú reprezentáciu frekventovaných množín využíva pohybujúce sa okno a aktualizáciu v každej transakci. Podrobnejšie ho opisujeme v prílohe C.

Hľadanie exaktných frekventovaných množín kladie priveľké nároky na algoritmy čo sa týka pamäte. Takéto algoritmy majú tiež problém s adaptáciou v prípade častých výskytov konceptuálnych posunov v prúde dát. Tieto problémy môžu vyriešiť algoritmy, ktoré hľadajú aproximáciu frekventovaných množín a ich podpôr. Odstúpenie od požiadavky hľadať exaktné frekventované množiny má za následok rapídne zlepšenie výkonnosti a zníženie pamäťových nárokov, okrem toho môže byť v mnohých aplikáciach takáto reprezentácia dostatočná a vie sa lepšie vysporiadať aj s konceptuálnym posunom (Quadrana & Mestre, 2012).

IncMine je aproximatívny algoritmus hľadania frekventovaných množín postupne v dávkach aktualizujúci množinu nájdených tzv. uzavretých frekventovaných množín nad pohybujúcim sa oknom v rýchlom prúde dát. Okrem klasickej hladiny minimálnej podpory

definuje aj uvoľnenú hladinu minimálnej podpory, ktorá zabezpečuje zotrvanie potenciálne frekventovaných množín v okne. Tento algoritmus bližšie opisujeme v časti 4.1.2.

V práci navrhujú algoritmus s názvom CLAIM. Je to ďalší algoritmus na dolovanie aproximácie uzavretých frekventovaných množín. Tento algoritmus sa snaží riešiť problém keď časté malé konceptuálne posuny v prúde dát môžu zbytočne spomaľovať algoritmus, redefiníciou pojmu uzavretá frekventovaná množina a určením intervalov hodnôt podpory, ktoré sú považované za rovnaké. Aktualizáciu vykonávajú po každej prichádzajúcej transakcií.

4.1.2 **Algoritmus IncMine**

IncMine je aproximatívny algoritmus hľadania frekventovaných množín postupne v dávkach aktualizujúci množinu nájdených tzv. uzavretých frekventovaných množín nad rýchlymi prúdmi dát. Tento algoritmus bol navrhnutý v práci (Cheng, Ke, & Ng, 2008).

*Uzavreté frekventované množiny (*ang. *Frequent Closed Itemsets, skr. FCI)* sú frekventované množiny, ktoré nemajú nadmnožinu s rovnakou podporou (napr. frekvencia výskytov) . Dôležitou vlastnosťou množiny všetkých FCI je, že je kompletná a neredundantná reprezentácia všetkých frekventovaných množín a často omnoho menšia ako množina všetkých frekventovaných množín, tak ako sme si to uviedli v úvode tejto kapitoly.

Nech D je databáza transakcií. Nech $C = \{i_1, i_2, i_3, \dots, i_n\}$ je množina všetkých položiek v databáze. Transakčný dátový prúd je sekvencia po sebe prichádzajúcich transakcií. Transakcia Y je teda množinou položiek. Platí $Y \subseteq C$. Nech X je množinou $X \subseteq Y$. Časová jednotka nech je označená ako t_i. V časovej jednotke môže prísť v prúde variabilný počet transakcií. Okno W je časovým intervalom – množina po sebe nasledujúcich časových jednotiek $W = < t_i, \dots, t_j >,\ i \leq j$. Posúvajúce sa okno W po prúde je okno s fixným počtom časových jednotiek, ktoré sa posúva dopredu po každej časovej jednotke. Nech t_τ je aktuálna časová jednotka a veľkosť okna je w – potom $W = < t_{\tau-w+1}, \dots, t_\tau >$ je aktuálne okno.

Nech T_w je množina transakcií, ktoré prišli v prúde v rámci aktuálneho okna *w*. Nech *support(X,* T_w *)* označuje počet transakcií Y z T_w v ktorých sa nachádza *X*.
Nech σ ($0 < \sigma \leq 1$) označuje minimálnu hranicu podpory. Ak $support(X, T_w) \geq \sigma \mid T_w \mid$ tak X prehlásime za frekventovanú množinu. Okrem toho X je uzavretá frekventovaná množina (*FCI*) nad T_w ak platí : X je frekventovaná množina nad T_w a

$$\nexists X', \quad X' \subseteq X \wedge \text{support}(X, T_w) = support(X', T_w)$$

Riešime problém hľadania množiny všetkých *FCI* v rámci posúvajúceho sa okna *W*. Problémom je zabezpečiť zotrvanie nefrekventovaných množín s potenciálom stať sa frekventovanými počas behu algoritmu. Predpokladajme, že množina *X*, ktorá nie je frekventovaná sa môže po čase *t* stať frekventovanou. Posúvaním okna sa však podpora množiny *X*, ktorú mala pred časom *t*, stratí. Napr. v práci (Chang & Lee, 2003) tento problém rieši pomocou tzv. *uvoľnenej hranice minimálnej podpory* (ang. *relaxed*

minimum support threshold). Množina *X* zotrvá v okne pokiaľ jej podpora je väčšia ako $\varepsilon, 0 \leq \varepsilon < \sigma$. Čím menšie je ε tým viac sa blíži podpora množiny *X* jej skutočnej podpore avšak tým väčšie je aj množstvo uchovávaných množín. V práci (Cheng, Ke, & Ng, 2008) preto navrhli pojem tzv. *semi-frekventovaných množín*, kde definujú $\varepsilon = r\sigma$, kde r $(0 \leq r \leq 1)$ je tzv. *rýchlosť uvoľňovania* . Definujú progresívne rastúcu funkciu minimálnej podpory *minsup(k)* ako

$$minsup(k) = \sigma|T_k| \times \left(\left(\frac{1-r}{w}\right)(k-1) + r\right)$$

minsup(k) počítame pre všetky k , $1 \leq k \leq w$. Kde $T_k = < t_{\tau-w+1}, \ldots, t_\tau >$. Aproximatívnu hodnotu podpory *X* nad časovou jednotou *t* vypočítame ako (kde T_t je množina transakcií v časovej jednotke *t*):

$$aprox\_\sup(X,t) = \begin{cases} 0 \; if \; support(X,t) < r\sigma|T_t| \\ support(X,t) \;\; inak \end{cases}$$

Aproximatívnu podporu nad časovým intervalom T_k vypočítame ako :

$$approx\_\sup(X,T_k) = \sum_{i=j}^{k} approx\_\sup(X,t_i)$$

X je semi-frekventovaná uzavretá množina (*semi-FCI*) nad oknom *W* ak :

$$\exists k \, , aprox_{sup(X,T_k)} \geq minsup(k) \; Y \, , Y \supset X \wedge aprox\_sup(Y,T_k) = approx\_sup(X,T_k))$$

X je tzv. *k – semi –frekventovaná množina (k-semi-FCI)* nad oknom W ak je to *semi-FCI* a *k* je dané ako najväčšie *k* pre ktoré platí : $approx\_\sup(X,T_k) \geq minsup(k)$.
Uvedená funkcia *minsup* nemusí byť vhodná pre všetky druhy dátových prúdov. Štúdiom charakteristík dátového prúdu môžeme túto funkciu prispôsobiť aby čo najlepšie zachytávala zmeny v dátových prúdoch.

Nech C je množina *semi-FCI* nad aktuálnym oknom, ktorú chceme získať. L je množina *semi-FCI* nad predchádzajúcim oknom. A F je množina *semi-FCI* z aktuálnej časovej jednotky t_τ. Úlohu inkrementálnej aktualizácie množiny *semi-FCIs* nad pohybujúcim sa oknom definujú autori ako získanie množiny C aktualizovaním množiny L pomocou F. Výstupom je nakoniec množina všetkých *k-semi-FCI*, ktoré majú vyššiu podporu ako je minimálna podpora pre okno Wc, teda σ.

Autori ďalej skúmali vlastnosti *semi-FCI* a uvádzajú niekoľko tvrdení vďaka, ktorým ich algoritmus *IncMine* inkrementálne aktualizuje množiny semi-FCIs efektívnejšie ako jeho predchodcovia. Ďalej v opisovanej práci autori uvádzajú, že klasická dátová štruktúra *prefix-tree*, ktorá sa používa pri dolovaní frekventovaných vzorov nie je dostatočne efektívna hlavne pre operáciu hľadania *semi-FCI* a preto navrhujú použitie *invertovaného indexu* ako dátovej štruktúry na ukladanie množín *F*, *C* a *L*. Nakoniec v práci experimentálne potvrdzujú efektivitu a rýchlosť algoritmu ako i porovnateľnú presnosť a pokrytie v porovnaní s inými algoritmami.

4.2 Zhlukovanie nad prúdom dát

Zhlukovanie nad prúdom dát je náročnou úlohou. V tejto podkapitole analyzujeme niekoľko základných princípov a algoritmov zhlukovania nad prúdom dát. Tieto znalosti ďalej využívame v návrhu metódy na zhlukovanie modelov používateľov v kapitole 5. Barbará vo svojej práci (Barbará, 2003) hovorí o všeobecných požiadavkách na algoritmy zhlukovania nad prúdom dát:

- Kompaktnosť reprezentácie zhlukov. Teda dôležité je aby veľkosť reprezentácie nerástla rýchlo zároveň s prichádzajúcimi novými dátovými inštanciami.
- Rýchle a inkrementálne spracovanie nových dátových inštancií.
- Jasné a rýchle identifikovanie odľahlých dátových inštancií.

Odteraz pre zjednodušenie pokiaľ hovoríme o zhlukovaní tak myslíme zhlukovanie nad prúdom dát. Bolo navrhnutých viacero algoritmov napr. Clustream v práci (Aggarwal, a iní, 2003), HPStream (Aggarwal, Han, Wang, & Yu, 2004), DenStream (Cao, Ester, Qian, & Zhou, 2006), Dstream (Chen & Tu, 2007), ClusTree (Kranen, Assent, Baldauf, & Seidl, 2011). Dôležitý rámec pre zhlukovanie v prúdoch dát Clustream, ktorý je základom pre niektoré ďalšie algoritmy opisujeme v časti 4.2.1. Niektoré ďalšie ako DenStream a ClusTree opisujeme v prílohe D.

4.2.1 Rámec Clustream

V práci (Aggarwal, a iní, 2003) navrhli rámec s názvom Clustream na zhlukovanie nad prúdom dát, ktorý je založený na dvoch komponentoch. Online mikro-zhlukovací komponent, zabezpečujúci rýchle spracovanie prichádzajúcich údajov a efektívne ukladanie sumárnych štatistík. A offline komponent používajúci tieto sumárne štatistiky spoločne s ďalším používateľským vstupom, ktorý používateľovi ukazuje informácie o existujúcich zhlukoch na požiadanie.

Online komponent navrhnutého rámca v každom čase vykonávania algoritmu udržuje štatistiky o stave q mikro-zhlukov. Označme tieto mikrozhluky ako $M_1, \dots, M_q$. Počet mikrozhlukov q je určený veľkosťou dostupnej hlavnej pamäte. Typicky je hodnota q značne vyššia ako počet hľadaných zhlukov ale tiež značne menšia ako počet analyzovaných dátových inštancií. Mikrozhluky predstavujú snímku aktuálneho stavu zhlukov nachádzajúcich sa v dátovom prúde. História vývoja mikrozhlukov sa neukladá. Na začiatku algoritmu sa offline vytvoria pomocou klasického k-means zhlukovania z určených počiatočných N dátových bodov iniciálne mikrozhluky. Po tejto inicializácií algoritmus pokračuje v online fáze postupnej aktualizácie mikrozhlukov. Ku aktualizácií dochádza vždy s príchodom novej dátovej inštancie X_i. Určí sa vzdialenosť dátovej inštancie ku všetkým centroidom existujúcich mikrozhlukov. Nech najbližší nájdený centroid je označený ako M_p. Môžu nastať nasledovné situácie:

- Dátová inštancia je pohltená mikrozhlukom.
- Z dátovej inštancie je vytvorený nový mikrozhluk.

Na určenie toho či má byť dátová inštancia pohltená mikrozhlukom je nutné určiť tzv. maximálnu hranicu mikrozhluku, ktorá je definovaná ako stredný štvorcový rozptyl (ang.

Root Mean Square Deviation, skr. RMSD) vzdialeností dátových inštancií patriacich konkrétnemu mikrozhluku. Ak je vzdialenosť dátovej inštancia X_i k centroidu najbližšieho mikrozhluku M_p nižšia ako jeho maximálna hranica tak je jednoducho pohltená mikrozhlukom a prepočíta sa jeho maximálna hranica. Ak sa dátová inštancia nevošla do maximálnej hranice je z nej vytvorený nový mikrozhluk. Iniciálna maximálna hranica mikrozhluku je určená heuristicky. Ak sa však takýmto spôsobom vytvorí nový mikrozhluk tak je potrebné iný mikrozhluk zrušiť alebo dva mikrozhluky spojiť, aby sa zachoval počet q. Prvým krokom je teda hľadanie starého mikrozhluku, ktorý leží príliš mimo ostatných mikrozhlukov (v ang. outlier). Ak je takýto identifikovaný tak je bezpečné ho odstrániť. Autori rámca navrhli identifikovať takéto mikrozhluky pomocou približnej priemernej časovej pečiatky posledných m pridaných dátových inštancií. Ten mikrozhluk, ktorý má túto priemernú časovú počiatku najstaršiu a zároveň staršiu ako je používateľom stanovená hranica môže byť odstránený. V prípade, že žiaden z mikrozhlukov nemôže byť odstránený, musí dôjsť k spojeniu dvojice mikrozhlukov s najmenšiou vzájomnou vzdialenosťou.

Offline komponent zabezpečuje používateľovi možnosť skúmať zhluky v dátovom prúde v rôznych časových momentoch. Vďaka štatistickej reprezentácií dát pomocou mikrozhlukov je offline proces hľadania makrozhlukov značne zrýchlený. Makrozhluky sú získavané pomocou modifikovaného k-means algoritmu. Mikrozhluky sú chápané ako zhlukované dátové inštancie. Vo fáze inicializácie sa nevyberajú počiatočné body náhodne ale mikrozhluky s vyšším počtom bodov sú vybrané pravdepodobnejšie.

4.3 Existujúce rámce na spracovávanie kontinuálneho prúdu dát

V tejto časti analyzujeme niekoľko známych existujúcich rámcov určených pre prácu a dolovanie znalosti z prúdu dát.

4.3.1 RAPID MINER – Streams plugin

RapidMiner (skr. RM) je softvérová platforma určená na rýchly návrh aj zložitých problémov strojového učenia a dolovania znalostí v dátach, ktoré sa modelujú z procesného pohľadu. Na opis procesov používa zreťazené štruktúry operátorov definovaných v XML. RM je implementovaný v Jave a je Open Source. Samotný RM je možné použiť iba v prípade učenia v dávkach. Na spracovanie kontinuálneho prúdu dát slúži doplnková knižnica *streams* integrovateľná do RM ako StreamsPlugin. Knižnica streams poskytuje triedy a rozhrania, ktoré zabezpečujú spracovanie prúdov dát z rôznych zdrojov. Neponúka však implementácie algoritmov strojového učenia pre prúdové dáta. Je však možné integrovať do procesu spracovania algoritmy dostupné napríklad z rámca MOA (Bockermann & Blom, 2012).

4.3.2 MOA – Massive Online Analysis

MOA je softvérový rámec implementovaný v jazyku Java. Obsahuje kolekciu vopred implementovaných a umožňuje vytváranie nových algoritmov pracujúcich s masívnymi vyvíjajúcimi sa prúdmi dát a poskytuje aj možnosti na ich evaluáciu. Je voľne dostupná pod *GNU PL* licenciou a podporuje interakciu s knižnicou *WEKA (Waikato Environment for Knowledge Analysis)* tiež implementovanou v jazyku Java a obsahujúcou širokú škálu implementovaných dávkových metód strojového učenia. (Bifet, Holmes, Kirkby, & Pfahringer, 2010)

Oproti bežnému dávkovému spracovaniu je odlišný aj spôsob evaluácie prúdových dát. Dva známe prístupy, ktoré sú implementované aj v rámci *MOA* sú (Bifet, Holmes, Kirkby, & Pfahringer, 2010):

- Otestovanie modelu na vopred určenej nazbieranej podmnožine dát (v podstate rovnako ako pri rozdelení na trénovaciu a testovaciu množinu pri dávkových metodách strojového učenia)
- Testovanie prepletené s trénovaním. Každá vzorka je použitá na otestovanie modelu predtým než sa použije na jeho trénovanie. Z výsledkov takýchto individuálnych testov sa inkrementálne aktualizujú metriky ohodnocujúce model (napr. presnosť klasifikácie).

4.3.3 SAMOA - Scalable Advanced Massive Online Analysis

SAMOA je platforma pre dolovanie v masívnych dátových prúdoch. Podobne ako *MOA* obsahuje kolekciu state-of-art algoritmov pracujúcich s masívnymi a vyvíjajúcimi sa prúdmi dát a tiež programové abstrakcie umožňujúce implementáciu nových algoritmov. Rozhodujúcou výhodou oproti *MOA* je, že umožňuje tieto algoritmy vykonávať vo viacerých rámcoch umožňujúcich distribuované spracovanie ako *Apache STORM*, *S4* a *SAMZA*. *SAMOA* je implementovaná v Jave a je Open Source (Morales & Bifet, 2015).

4.4 Sumarizácia a diskusia

V úvode tejto kapitoly sme popísali nové požiadavky, ktoré sú kladené na úlohu hľadania frekventovaných vzorov v prúde dát. Hovorili sme o požiadavkach na rýchlosť, pamäťovú efektívnosť a požiadavke na možnosť balansovať efektívnosť výpočtu s presnosťou výsledkov. Prístupy k samotnému dolovaniu frekventovaných množín sú väčšinou založené na princípe hľadania frekventovaných množín v rámci okna obsahujúceho posledných *w* transakcií. Možno ich ďalej rozdeliť podľa toho či je toto okno fixovanej dĺžky alebo predstavuje fixovaný časový interval. Ďalej možno prístupy deliť podľa toho či aktualizujú frekventované množiny v dávkach alebo s každou prichádzajúcou transakciou, či hľadajú exaktné frekventované množiny, alebo len aproximatívne. Bližšie sme analyzovali algoritmy *Moment, Closet* a *IncMine* a uviedli ich krátke porovnanie. Uviedli sme podrobnejší opis fungovania algoritmu *IncMine*. Opisy algoritmov *Moment* a *Closet* uvádzame pre ich rozsiahlosť v prílohe C.

Algoritmus *IncMine* s niektorými úpravami sme sa rozhodli použiť na implementáciu komponentu metódy, ktorú navrhujeme v kapitole 5 zabezpečujúceho dolovanie frekventovaných množín z prúdu dát aj z nasledujúcich dôvodov:

- Jeho implementácia je dostupná v rámci knižnice MOA.
- Je aproximatívny algoritmus s veľmi dobrou presnosťou aproximácie podľa výsledkov experimentov uvedených v práci, čo prispieva k vyššej rýchlosti spracovania oproti exaktným algoritmom ako *MOMENT* a *CLOSTREAM*.
- Hľadá uzavreté frekventované množiny, ktoré sú kompletnou a neredundantnou reprezentáciou všetkých frekventovaných množín, ktorá zároveň výrazne napomáha k zmenšeniu prehľadávaného priestoru.
- využíva pohybujúce sa okno, ktoré oproti míľnikovému oknu sa dokáže lepšie vysporiadať s konceptuálnym posunom.
- Autori sa porovnávajú s algoritmom *MOMENT*. *IncMine* je podľa ich výsledkov značne rýchlejší.
- *IncMine* ako jediný z analyzovaných algoritmov aktualizuje frekventované množiny po dávkach transakcií ako autori pozorujú vo svojich experimentoch zdá sa tento prístup vhodnejší keďže v prípade aktualizácie po každej transakcií je aktualizácie príliš častá a môže viesť k zbytočne častému vyťaženiu výpočtových zdrojov, prínos jednej jedinej transakcie oproti väčšej dávke transakcií spolu je pritom často zanedbateľný.

Opísali sme aj problém zhlukovania v prúde dát a algoritmy *Clustream* a *Clustree* sme bližšie opísali v prílohe D. Tomuto problému sme sa venovali kvôli tomu, že významnou súčasťou metódy navrhovanej v kapitole 5 bude práve komponent zabezpečujúci zhlukovanie modelov používateľov na základe ich správania. Rozhodli sme sa využiť *CluStream* ako základny algoritmus pre zhlukovanie modelov používateľov. Ako sme spomenuli tento algoritmus predstavuje rámec na ktorom sú postavené aj ďalšie algoritmy zhlukovania v prúde dát (napr. *DenStream*). Proces navrhovaný v kapitole 5 sme sa snažili navrhnúť ako rámcový process v ktorom je možné neskôr aj tento komponent *CluStream* zameniť za iný.

5 Navrhovaná metóda reprezentácie, extrakcie a aplikácie vzorov správania nad prúdom dát

V tejto časti opisujeme návrh metódy, ktorá zabezpečuje jednak hľadanie *globálnych vzorov* správania typických pre všetkých používateľov a zároveň segmentáciu používateľov do skupín a hľadanie *skupinových vzorov* správania, teda vzorov typických pre užšie skupiny používateľov. V návrhu tiež opisujeme spôsob aplikácie kombinácie globálnych a skupinových vzorov správania v úlohe odporúčania. Neskôr v kapitole 7 skúmame či vďaka navrhovanej metóde a kombinácií globálnych vzorov so skupinovými vieme dosiahnuť lepšie výsledky v úlohe odporúčania než by bolo možné dosiahnuť len s použitím globálnych vzorov.

Dôležitou charakteristikou metódy je práca s prúdom dát. Ako sme uviedli v úvode kapitoly 4 kladie to na navrhovanú metódu isté požiadavky s ktorými sme pri návrhu museli rátať. Naša metóda je novou aplikáciou v oblasti dolovania dát o používaní Webu, ktorá v porovnaní voči aplikáciam, ktoré sme uviedli v časti 2.2 reaguje na niektoré ďalšie výzvy a to konkrétne personalizáciu Webu (keďže hľadá vzory typické k správaniu používateľa) a spracovanie dát ako rýchleho a potencionálne nekonečného prúdu. Prínos navrhovanej metódy spočíva v schopnosti odhaľovať aktuálne záujmy používateľov webového sídla a zmeny ich záujmov v čase a to nielen na globálnej úrovni ale aj v rámci užších skupín používateľov. Nachádzané vzory správania majú viacero možných aplikácií (pozri časť 0) ako napr. použitie na predikciu, odporúčanie, prednačítavanie stránok do dočasnej pamäte, či celkové porozumenie správania používateľov webového sídla.

Návrh pozostáva z častí venujúcich sa formátu spracovávaných dát (5.1.1), spôsobu reprezentovania vzorov správania (5.1.2) , modelu používateľa (5.1.3), spôsobu aplikácie vzorov správania v úlohe odporúčania (5.1.4), rámcového opisu procesu spracovania transakcií (0). rád pre realizáciu v praxi (5.2.1), návrhu paralelizácie procesu (5.2.2), spôsobu regulovania rýchlosti spracovania transakcií (5.2.3). Na záver ešte sumarizujeme všetky vstupné parametre metódy (0).

5.1 Základné pojmy

Táto časť objasňuje niektoré základné pojmy a princípy navrhovanej metódy.

5.1.1 Zdroje a formát spracovávaných dát

Vstupom do navrhovanej metódy sú dáta z webového sídla. Prúd dát je generovaný ako postupnosť transakcií. Transakcia je vygenerovaná vždy na konci sedenia používateľa. Jedna vstupná transakcia, predstavuje konkrétne, práve ukončené sedenie používateľa. Táto transakcia obsahuje usporiadaný zoznam akcií používateľa v rámci sedenia zakódované do číselných hodnôt. Za akcie sa môžu považovať napríklad návštevy konkrétnych stránok alebo širších kategórií stránok, ale môžu to byť aj iné akcie používateľa v systéme ako napr. pridanie položky do košíka v internetovom obchode,

vymazanie položky z košíka, pridanie komentára, pridanie hodnotenia k položke, kúpa a mnoho ďalších. Okrem toho obsahuje transakcia identifikátor používateľa, ktorý dané akcie vykonal.

5.1.2 Reprezentácia vzoru správania

V navrhovanej metóde chápeme vzor správania *p* ako štruktúru obsahujúcu nasledujúce údaje :

- Skupinový identifikátor, pokiaľ ide o skupinový vzor správania (*ozn. p.gid z ang. pattern group identificator*).
- Uzavretú frekventovanú množinu (pozri 3.1) (*ozn. p.fci z ang. pattern frequent closed itemset*). To je množina identifikátorov akcií používateľa, ktorá spĺňa podmienku minimálnej podpory nad aktuálnym pohybujúcim sa oknom (pozri časť 4.1.2) v rámci prúdu dát.
- Váhu vzoru (*ozn. p.sup z ang. pattern support*). To je hodnota podpory množiny vypočítaná vzhľadom k aktuálnemu pohybujúcemu sa oknu v prúde dát.

5.1.3 Reprezentácia modelov používateľov

Keďže jedným z cieľov navrhovanej metódy je hľadanie skupinových vzorov správania v prúde dát, je dôležitým problémom, ktorý musí metóda riešiť aj reprezentácia modelov používateľov s čo najmenšími nárokmi na pamäť a tiež zabezpečenie aby využitie pamäte na uloženie týchto modelov nerástlo proporcionálne k nárastu počtu spracovaných dát. Model používateľa *u* je štruktúra pozostávajúca z nasledujúcich častí :

- Unikátny identifikátor používateľa (ozn *u.id* z ang. *user identificator*).
- Rad s obmedzenou kapacitou obsahujúci zoznam akcií v rámci posledných sedení používateľa (*ozn. u.aq z ang.user's actions queue*).
- Identifikátor skupiny do ktorej bol používateľ zaradený pri poslednom makrozhlukovaní (*ozn. u.gid z ang. user's group identificator*).
- Počet nových sedení používateľa od poslednej aktualizácie identifikátora skupiny (*ozn. u.nsc z ang. user's new sessions count*).
- Identifikátor posledného makrozhlukovania v ktorom bol používateľovi priradený skupinový identifikátor (*ozn. u.lmid z ang. user's last macroclustering identificator*).

Vo svete webových sídel môže byť počet používateľov nesmierne dynamický. Noví používatelia prichádzajú a starí odchádzajú. Navrhovaná metóda v pravidelných intervaloch aktualizuje zhluky používateľov podľa ich aktuálneho správania. Pre každý model používateľa si pamätá len obmedzený počet akcií aby nedošlo k nadmernému zaťaženiu pamäte a postupnému spomaľovaniu spracovania.

Vstupom do zhlukovacieho algoritmu založeného na rámci *CluStream* (ktorý sme opísali v časti 4.2.1) je nová dátová inštancia vygenerovaná na požiadanie z histórie uskutočnených akcií modelu používateľa v podobe zoznamu frekvencií jednotlivých vykonaných akcií. Každé nové uskutočnené makrozhlukovanie má priradené nové identifikačné číslo a vygeneruje nové zhluky s novými centrami pravidelne po určenom

počte aktualizácií mikrozhlukov (bližšie opisujeme tento proces ďalej v časti 5.2). Priradenie identifikátoru skupiny používateľovi sa deje vždy len pri prvej vyskytnutej požiadavke (teda prvom sedení používateľa od posledného makrozhlukovania) teda iba vtedy ak je identifikátor posledného makrozhlukovania v modeli používateľa *u* (atribút *u.lmid*) iný než identifikátor najnovšieho makrozhlukovania.

Aby sa proporcionálne k času behu metódy nezvyšovalo zaťaženie pamäti tým, že si bude pamätať všetkých používateľov od začiatku behu tak sa pravidelne zoznam modelov používateľov prechádza a odstráni sa každý taký model používateľa *u* pre ktorý platí, že *lmid - u.lmid > tmdiff* , kde *lmid* je identifikátor najnovšieho makrozhlukovania a *tmdiff* je vstupný parameter (sumarizácia parametrov je v časti 5.3).

Dôležitým problémom pri algoritmoch strojového učenia býva tiež dimenzionalita dát. S narastajúcou dimenzionalitou rôznych akcií sa bude spomaľovať aj algoritmus pre zhlukovanie ale aj algoritmus hľadania uzavretých frekventovaných množín, ktoré sú súčasťou navrhovanej metódy. Preto je dôležité pri generovaní dát, ktoré sú vstupom metódy premýšľať aj s ohľadom na toto zaťaženie a pokiaľ sa dá tak generovať dátové inštancie s menšou veľkosťou množiny možných akcií používateľa (dimenzionalitou).

5.1.4 Spôsob aplikácie nájdených vzorov

Výstupom navrhnutej metódy je v každom čase množina objavených globálnych vzorov správania a množina objavených skupinových vzorov správania. Úlohu hľadania vzorov správania môžeme považovať za samostatnú a oddelenú od úlohy aplikácie vzorov správania. Tieto úlohy teda samozrejme môžu byť realizované aj rôznymi fyzickými komponentami architektúry konkrétneho softvéroveho riešenia. Vygenerované vzory správania môžu byť použité na rôzne ciele ako napr. generovanie odporúčaní , analýzu správania používateľov za rôznymi účelmi ako napríklad zlepšenie štruktúry a dizajnu webového sídla, pochopenia používateľa a jeho chovania ako sme to uviedli v časti 2.2.

My sme sa rozhodli skúmať vzťahy medzi skupinovými a globálnymi vzormi správania (okrem iných skúmaných štatistík) najmä prostredníctom aplikácie týchto vzorov v úlohe odporúčania a skúmania prínosu skupinových vzorov v kombinácií s globálnymi práve v tejto úlohe.

Dôležitým problémom pri generovaní odporúčaní je vybranie vzorov správania, ktoré čo najlepšie vystihujú súčasné správanie používateľa. Nech aktuálne sedenie je reprezentované vektorom akcií používateľa $S = < a_1, a_2, \dots a_n >$ a tiež všetky vzory správania $P = < P_1, P_2, \dots P_m >$ sú reprezentované ako vektory položiek. Vstupným parametrom metódy je veľkosť okna vyhodnocovania *ews* (skr. z ang. *evaluation window size*). Platí $ews > 0 \; a \; ews < |S|$. Aktuálne sedenie *S* sa rozdelí na časť, ktorá sa použije na hľadanie vzoru a testovaciu časť, ktorá sa použije na vyhodnotenie odporúčania. Označme časť určenú na hľadanie vzoru ako vyhodnocovacie okno $W_e = < a_1, a_2, .. a_k >$. A označme časť určenú na testovanie ako $T_e = < a_{k+1}, \dots, a_n >$. Nech počet odporúčaných položiek je *rc* (to je vstupný parameter metódy, pozri 5.3). Ak aktuálne sedenie nespĺňa podmienku *n >= (ews + rc)* tak ho pre účely vyhodnocovania odporúčania ignorujeme. V našej metóde používame nasledovnú stratégiu výberu vzorov správania:

1. Pre každý vzor správania P_i vypočítame odhadovanú podporu $(P_i).sup$ normalizovanú na interval <0,1> a tiež nájdeme prienik s aktuálnym vyhodnocovacím oknom W_e sedenia používateľa pomocou algoritmu *LCS* (*Longest common subset*). Označme hodnotu tohto prieniku (teda počet spoločných akcií normalizovaný na hodnotu z intervalu <0,1> podľa dĺžky okna W_e) ako $lcs(P_i, W_e)$.
2. Všetky nájdené vzory sa zoradia.
    a. Prvá úroveň zoradenia je podľa hodnoty prieniku $lcs(P_i, W_e)$ od najväčšieho po najmenšiu hodnotu.
    b. Druhá úroveň zoradenia je podľa hodnoty podpory vzoru správania $(P_i).sup$ od najvyššej po najmenšiu.
3. Postupne sa striedavo prechádzajú globálne a skupinové vzory od najlepšieho k horším a buduje sa mapa kde sa jednotlivým položkám, ktoré sa nachádzajú v týchto vzoroch (a nenachádzajú sa vo vyhodnocovaciom okne W_e) pridávajú „hlasy“. Hodnota položky sa inkrementuje o hodnotu podpory vzoru v ktorom sa nachádza násobenej hodnotou prieniku vzoru s aktuálnym oknom sedenia teda (obidve normalizované do intervalu <0,1>):
    - $hlas = (P_i).sup * lcs(P_i, W_e)$

    Toto sa opakuje pre každý nájdený vzor. Nakoniec sa vyberie *rc* (vstupný parameter metódy udávajúci počet odporúčaných položiek) položiek s najvyššou hodnotou hlasov a tie sa odporúčia používateľovi.

5.2 Proces spracovania transakcií

V tejto časti opisujeme proces spracovania vstupných transakcií prúdiacich do metódy. Ide o návrh platný pre experimentálne prostredie. To znamená že proces je nastavený tak aby zahŕňal aj kroky potrebné pre ohodnotenie úspešnosti metódy.

V časti 5.2.1 opisujeme aké zmeny v tomto procese by bolo vhodné urobiť v prípade zavedenia metódy do prostredia reálnej aplikácie.

Navrhovaný proces predstavuje rámcové riešenie problematiky dolovania skupinových vzorov z prúdu dát keďže niektoré komponenty možno zamieňať (zhlukovací algoritmus, algoritmus dolovania uzavretých frekventovaných množín).

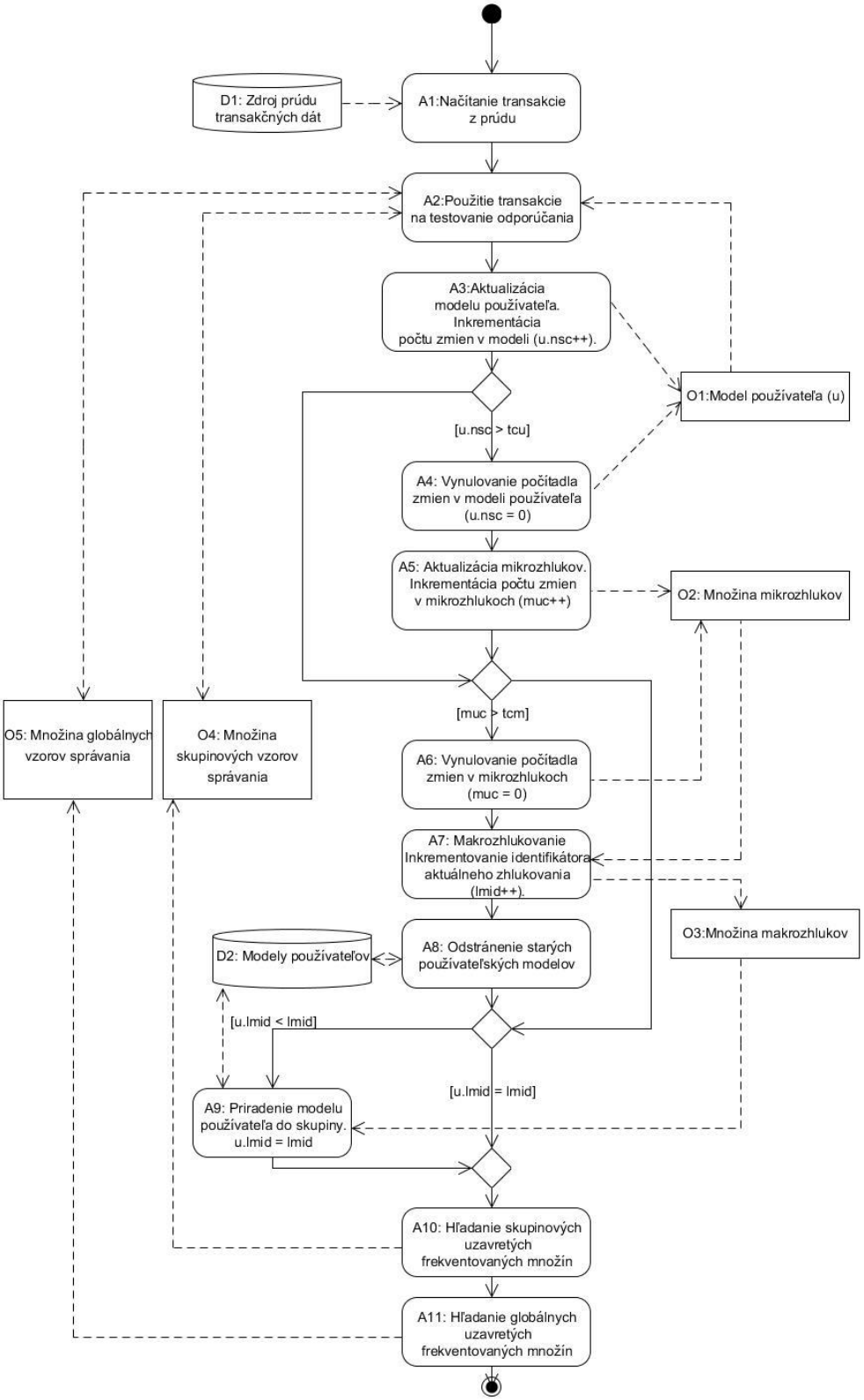
Celý proces spracovania jednej transakcie reprezentujúcej sedenie používateľa je znázornený na diagrame aktivít (Obrázok 1). V tejto časti bližšie popíšeme jednotlivé kroky. V diagrame identifikátory začínajúce písmenom A označujú bloky aktivít, začínajúce písmenom D dátové úložiská (v pamäti alebo externé), začínajúce písmenom O dátové objekty.

- **D1: Zdroj prúdu transakčných dát.** V tejto práci používame kvôli zjednodušeniu vyhodnocovania vopred pripravené dáta v požadovanom formáte. V reálnej aplikácií by boli tieto dáta generované webovým sídlom samotným. Formát dát sme opísali v časti 5.1.1.
- **D2: Modely používateľov.** V pamäti sa udržiava množina všetkých modelov aktívnych používateľov, ktoré sú neskôr vstupom pre zhlukovanie. O použití a obmedzeniach veľkosti tejto množiny a spôsobe reprezentácie modelov používateľov bližšie píšeme v časti 5.1.3.
- **O1: Model používateľa.** Tento objekt reprezentuje konkrétny model používateľa. Reprezentácia modelu používateľa je bližšie opísaná v časti 5.1.3.
- **O2:Množina mikrozhlukov.** Objekt reprezentuje množinu mikrozhlukov udržiavaných v pamäti. Mikrozhluky predstavujú štatistickú snímku aktuálneho stavu zhlukov nachádzajúcich sa v dátovom prúde. Viac o mikrozhlukoch v časti 4.2.1 venujúcej sa rámcu *CluStream* určenému na zhlukovanie nad prúdom dát, ktorý sme sa rozhodli použiť.
- **O3:Množina makrozhlukov:** Objekt reprezentuje množinu makrozhlukov udržiavaných v pamäti. Makrozhluky predstavujú aproximáciu aktuálneho stavu zhlukov v prúde dát, ktorý je získaný pomocou upraveného algoritmu *k-means* z množiny mikrozhlukov. Viac o makrozhlukoch v časti 4.2.1 venujúcej sa rámcu *CluStream* určenom na zhlukovanie nad prúdom dát, ktorý sme sa rozhodli použiť.
- **O4: Množina skupinových vzorov správania:** Objekt reprezentuje množinu aktuálnych objavených skupinových vzorov správania.
- **O5: Množina globálnych vzorov správania:** Objekt reprezentuje množinu aktuálnych objavených globálnych vzorov správania.
- **A1: Načítanie transakcie z prúdu dát.** Prvým krokom procesu spracovania transakcie reprezentujúcej sedenie používateľa je jej načítanie zo zdroja prúdu dát **D1**.
- **A2: Použitie transakcie na testovanie odporúčania.** Prvým krokom, ktorý s transakciou vykonáme je otestovanie na úlohu odporúčania. Spôsob odporúčania

opisujeme v časti 5.1.4. Spôsob samotného vyhodnocovania odporúčania pomocou konkrétnych metrík opisujeme v časti 7.1.1.

- **A3: Aktualizácia modelu používateľa. Inkrementácia počtu zmien v modeli.** Nasleduje aktualizácia modelu používateľa *u* (**O1**) podľa údajov aktuálneho sedenia.
  Do radu s obmedzenou kapacitou (atribút *u.aq*) obsahujúceho zoznam akcií vykonaných v rámci posledných sedení používateľa sa zaradí celé aktuálne sedenie používateľa. Inkrementuje sa tiež počet sedení modelu používateľa od posledného priradeného makrozhlukovania (atribút *u.nsc* sa inkrementuje o 1).
- **A4: Vynulovanie počítadla zmien v modeli používateľa.** Ak počítadlo zmien v modeli používateľa *u* (atribút *u.nsc*) ešte nedosiahlo hodnotu vyššiu ako vopred definovaná hraničná hodnota počtu zmien v modeli používateľa *(skr. tcu z ang. threshold number of changes in usermodel)* tak sa kroky **A4** aj **A5** preskočia.
- **A5: Aktualizácia mikrozhlukov. Inkrementácia počtu zmien v mikrozhlukoch.** V prípade, že došlo k dostatočnému počtu zmien v modeli používateľa *u* (atribút *u.nsc*) sa tieto zmeny prenesú aj do do štatistickej reprezentácie zhlukov – tzv. mikrozhlukov. Na zhlukovanie v prúde dát používame rámec *CluStream*. Ten sme už popísali v časti 4.2.1. Počítadlo zmien v mikrozhlukoch *muc* (*skr. z ang. microclusters updates counter*) je inkrementované s každým vykonaním tohto kroku **A5** o 1.
- **A6: Vynulovanie počítadla zmien v modeli používateľa.** Ak počítadlo zmien v mikrozhlukoch *muc* ešte nedosiahlo hodnotu vyššiu ako vopred definovaný parameter *tcm* (*hraničný počet zmien v mikrozhlukoch skr. z ang. threshold of changes in microclusters*) tak sa krok **A6** aj **A7** preskočia.
- **A7: Makrozhlukovanie.** Ak došlo k dostatočnému počtu zmien v mikrozhlukoch tak sa z tejto štatistickej reprezentácie pomocou klasického zhlukovacieho algoritmu získajú tzv. makrozhluky. Tu možno použiť modifikovaný algoritmus *k-means*, ktorý hľadá vstupným parametrom *gc* (*z ang. groups count*) určený počet makrozhlukov zo vstupných mikrozhlukov. Alebo napr. v prípade použitia algoritmu *DenStream* (opísaný v prílohe D) sa použije ako makrozhlukovací algoritmus *DBSCAN,* ktorý nevyžaduje vopred určiť počet makrozhlukov. Inkrementuje sa tiež identifikátor posledného vykonaného makrozhlukovania *lmid*. Viac o spôsobe zhlukovania v prúde dát v časti 4.2.1.
- **A8: Odstránenie starých modelov používateľov.** Tento krok zabezpečuje, že pri každom novom makrozhlukovaní sa tiež zabezpečí odstránenie starých modelov používateľov reprezentujúcich používateľov, ktorý už dlhšie neboli aktívni. Odstránia sa tie modely, pre ktoré platí, že rozdiel medzi identifikátorom aktuálneho makrozhlukovania *lmid* a identifikátorom posledného markozhlukovania v modeli používateľa *u* (atribút *u.lcid*) je väčší ako hodnota vstupného parametra *tcdiff* (*hraničná hodnota rozdielu identifikátorov zhlukovania skr. tcdiff z ang. threshold of clustering id's difference*). Teda ak platí: $lmid - u.lmid > tdiff$.
- **A9: Aktualizácia priradenia modelu používateľa do skupiny.** Ak sa nerovnajú identifikátory zhlukovania v modeli používateľa *u* a aktuálneho makrozhlukovania ($u.lmid < lmid$) tak sa *u.lmid* modelu používateľa nastaví na rovnakú hodnotu ako *lmid* a modelu používateľa sa priradí identifikátor priradenej skupiny (atribút *u.gid*).

- **A10: Hľadanie skupinových uzavretých frekventovaných množín.** Ak model používateľa *u* už má priradený identifikátor skupiny (atribút *u.gid*) tak je aktuálne sedenie vstupom do algoritmu hľadania uzavretých frekventovaných množín *IncMine*, ktorý sme opísali v časti 4.1.2. Ten je upravený tak, že hľadá separátne uzavreté frekventované množiny pre jednotlivé skupiny používateľov identifikované zhlukovacím algoritmom. Pre každú skupinu je tak akoby simulovaný samostatný prúd len so sedeniami patriacimi danej skupine. A uzavreté frekventované množiny nájdené pre konkrétnu skupinu sú uložené do samostatnej dátovej štruktúry **O4**.
- **A11: Hľadanie globálnych uzavretých frekventovaných množín.** Tento krok sa vykoná s každým jedným sedením bez ohľadu nato či má alebo nemá priradený skupinový identifikátor. Aktuálne sedenie je vstupom do algoritmu hľadania uzavretých frekventovaných množín *IncMine*, ktorý sme opísali v časti 4.1.2. Nájdené uzavreté frekventované množiny sa ukladajú do samostatnej dátovej štruktúry určenej pre globálne uzavreté frekventované množiny **O5**.



Obrázok 1 Proces spracovania transakcie v navrhovanej metóde .

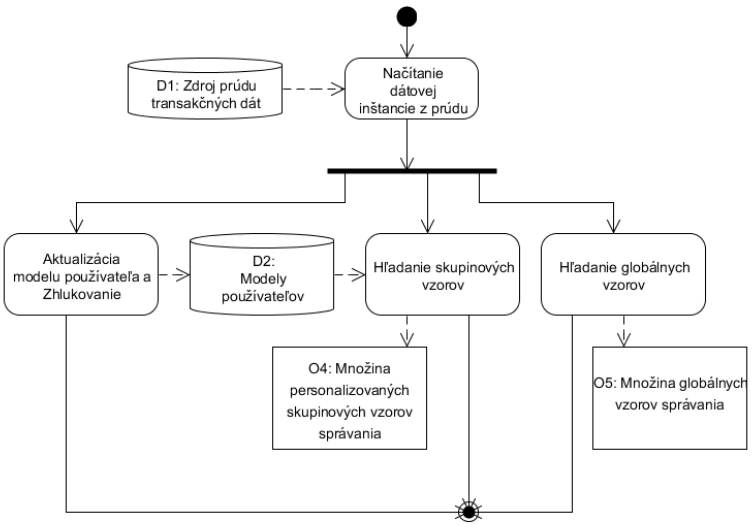
5.2.1 Zmeny v procese v prípade reálnej aplikácie

Proces, ktorý sme opísali je navrhnutý tak ako sme ho použili pre vykonanie experimentov opísaných aj v kapitole 7. V prípade, kedy by sa mala metóda použiť v reálnej aplikácií navrhujeme tieto úpravy:

- Vynechanie kroku **A2.** V prípade reálnej aplikácie by bolo samotné generovanie odporúčaní realizované oddeleným komponentom. Navrhovaná metóda by teda zabezpečila len generovanie vzorov správania. Ich aplikácia by bola záležitosťou ďalších komponentov, ktoré by na požiadanie získali aktuálnu množinu vzorov (dátové objekty **O4** a **O5**). Samotné odporúčanie je v takom prípade oddelené od procesu hľadania vzorov. Ešte počas sedenia používateľa sa posielajú dáta o sedení (množina doteraz navštívených stránok) do časti softvéru, ktorá na požiadanie zabezpečuje odporúčanie pomocou aktuálnej množiny vzorov správania získanej druhou časťou softvéru, ktorá spracúva kompletné dáta o sedeniach používateľov prichádzajúce v prúde. Samozrejme aplikácie objavených vzorov správania môžu byť rôzne (pozri časť 2.2).
- V prípade reálnej aplikácie je možné prispôsobiť časy makrozhlukovania mikrozhlukov tak aby boli vykonávané v časoch nižšej prevádzky. To si ale vyžaduje identifikovať časy zníženej prevádzky a zmeniť konfiguráciu metódy.
- Naša metóda ráta so spracovávaním s oknom fixovanej dĺžky. To je výhodné najmä čo sa týka jednoduchosti vyhodnocovania úspešnosti metódy na prúde dát simulovaného zo statického dataset. Je však pomerne jednoduché zmeniť metódu tak aby uvažovala okno s fixovanou časovou dĺžkou.

5.2.2 Návrh paralelizácie procesu

Navrhovaný proces pozostáva z viacerých komponentov, ktoré môžu fungovať nezávisle od seba. V princípe ide o rozdelenie prúdu dát na viacero vetiev. Každá transakcia z dátového prúdu je kopírovaná do jednotlivých vetiev, kde sa nezávislé od seba vykonávajú úlohy zhlukovania, hľadania globálnych vzorov a hľadania skupinových vzorov správania. Komponent pre zhlukovanie a komponent pre hľadanie skupinových vzorov zdieľajú medzi sebou modely používateľov s informáciou o ich priradení do zhlukov. Návrh je znázornený na obrázku Obrázok 2.



Obrázok 2 Návrh paralelizácie procesu hľadania personalizovaných a globálnych vzorov

5.2.3 Spôsob regulácie rýchlosti spracovania transakcií

Ako sme uviedli na začiatku kapitoly 4 venujúcej sa metódam dolovania vzorov správania z prúdových dát, tak je dôležité umožniť používateľovi nastaviť balans medzi rýchlosťou spracovania a presnosťou výsledkov. My sme sa rozhodli umožniť používateľovi nastaviť dolné obmedzenie požadovanej rýchlosti. Rýchlosť meriame ako počet spracovaných transakcií za jednu sekundu. Ak sa teda používateľ rozhodne, že je preňho spodná hranica rýchlosti spracovania napr. 20 transakcií za sekundu tak jednoducho nastaví toto obmedzenie a metóda sa bude týmto obmedzením riadiť. Čo sa týka výpočtovej náročnosti a vplyvu na toto obmedzenie je kritický krok aktualizácie uzavretých frekventovaných množín v algoritme dolovania frekventovaných množín *IncMine*. Táto aktualizácia nastáva vždy po každom novom prijatom segmente transakcií (viac o tomto algoritme v časti 4.1.2).

Naša metóda si v každom kroku udržiava informáciu o aktuálnej priemernej rýchlosti spracovania. Označme ju *actspeed*. Nech *ctrans* označuje aktuálny počet spracovaných transakcií od konkrétneho bodu začiatku merania. Nech *mts* označuje používateľom zadanú požadovanú minimálnu rýchlosť spracovania v transakciách za sekundu. Potom:

$$actspeed = \frac{ctrans}{mts}$$

Nech *tstart* označuje čas začiatku merania v *ms*. Nech *tsupdate* označuje začiatok aktualizácie množín v *ms* (od začiatku *tstart*). Pred každou aktualizáciou sa vypočíta povolený maximálny čas na aktualizáciu v *ms* takto:

$$tmax[ms] = actspeed - tsupdate + tstart$$

Vnútri samotnej aktualizácie množín:

1. Hľadajú sa uzavreté frekventované množiny v aktuálnom segmente pomocou algoritmu CHARM (opísaný v prílohe A). Použitie tohto algoritmu v rámci implementácie *IncMine* bolo rozhodnutím, ktoré vykonali autori implementácie

algoritmu *IncMine* (Quadrana, Bifet, & Gavaldà, 2015) v rámci *MOA*, ktorú v mierných modifikáciach (opísaných v kapitole 6) používame. Rozhodli sme sa tomuto kroku prideliť časové obmedzenie vo veľkosti $0.5 * tmax$. Získaná množina korešponduje s množinou *F* opísanou v časti 4.1.2.

2. Druhá časť je krok, ktorý korešponduje s úlohou získania množiny *C* aktualizovaním množiny *L* semi-frekventovaných množín nad predchádzajúcim oknom pomocou množiny *F* získanej v prvom kroku (pozri časť 4.1.2). Na tento krok je vyhradený celý zostavajúci čas aktualizácie.

Tento podiel rozdelenia maximálneho času aktualizácie do jednotlivých krokov sme určili len na základe expertného odhadu a teda môže byť predmetom ďalšieho skúmania.

Jedná sa teda o celkom hrubý zásah do výpočtu, kedy niektoré vzory, ktoré by možno boli zaujímavé nebudú nájdené. Jedinou úpravou, ktorú sme urobili je zmena stratégie zoradzovania uzlov v prehľadávanom strome v algoritme CHARM zo vzostupej na zostupnú podľa podpory. Hoci autori algoritmu túto stratégiu označili ako menej efektívnu tak v tomto prípade môže v prípade predčasného ukončenia prehľadávania zabezpečiť že sa skôr nájdu vzory s vyššou podporou. Musí teda dôjsť k určitému kompromisu, kedy znižovanie požiadavky na minimálnu rýchlosť zvyšuje presnosť algoritmu. V kapitole 7 sa venujeme aj vyhodnoteniu tohto kompromisu a zisťujeme či takéto znižovanie presnosti má alebo nemá fatálne dôsledky konkrétne v aplikácií na odporúčanie.

Ďalším zaujímavým riešením tohto problému by bolo použiť implementáciu algoritmu na dolovanie najlepších k-frekventovaných množín, kde nie je potrebné nastavovať hladinu minimálnej podpory a rýchlo sa nájde k najlepších vzorov (podobne ako sme opísali v časti 3.2.1). Implementácia tohto algoritmu a jeho integrácia do algoritmu *IncMine* je však už mimo rozsah tejto práce.

5.3 Sumarizácia vstupných parametrov metódy

Keďže navrhovaná metóda je vlastne kombináciou viacerých metód dolovania v prúde dát, z ktorých každá má svoje vstupné parametre, a samotná metóda má vlastné parametre je celkový počet vstupných parametrov pomerne veľký. Počet parametrov je jedným z kritických častí navrhovanej metódy. V tejto časti v krátkosti opíšeme účel jednotlivých parametrov.

Vstupné parametre sme rozdelili na 4 skupiny podľa ich účelu:

- Parametre algoritmu hľadania uzavretých frekventovaných množín *IncMine*,
- Parametre algoritmu zhlukovania *CluStream*,
- Parametre odporúčania,
- Ostatné všeobecné parametre a obmedzenia metódy.

Parametre algoritmu hľadania frekventovaných množín (IncMine):

- **Minimálna podpora** *(skr. ms z ang. minimal support)*: Hodnota minimálnej podpory. V zmysle výpočtu progresívnej funkcie minimálnej podpory ako sme uviedli v časti 4.1.2 je to vlastne hodnota σ.
- **Miera uvoľnenia** *(skr. rr z ang. relaxation rate)*: Ako sme uviedli v časti 4.1.2 je to vlastne hodnota parametra *r*, teda určuje rýchlosť uvoľnovania potenciálne frekventovaných množín.
- **Dĺžka segmentu** *(skr. sl z ang. segment length)*: Je to počet transakcií, ktoré patria do jedného segmentu. Segment je ekvivalentom časovej t_τ jednotky tak ako sme to uviedli v časti 4.1.2. Kvôli zjednodušeniu testovania používame okno pevne fixovanej dĺžky namiesto okna s fixovaným časovým intervalom.
- **Maximálna dĺžka množiny** (*skr. mil z ang. maximal itemset length*): Obmedzenie kladené na dĺžku hľadaných frekventovaných množín. Obmedzenie môže znížiť počet generovaných množín a tak zrýchliť algoritmus.
- **Veľkosť okna** *(skr. ws z ang. window size)*: Počet posledných segmentov v ktorých sa hľadajú frekventované množiny. Zväčšovanie dĺžky okna spoločne s dĺžkou segmentov znamená pamätanie si dlhšej histórie transakcií a teda zvyšovanie výpočtovej a pamäťovej náročnosti.

Parametre algoritmu zhlukovania (Clustream):

- **Počet zhlukov** *(skr. gc z ang. groups count)*: ekvivalentný počtu hľadaných skupín používateľov.
- **Hraničný počet zmien v modeli používateľa** *(skr. tcu z ang. threshold number of changes in usermodel)*: Počítadlo zmien v modeli používateľa je inkrementované s každým ďalším sedením používateľa. Ak toto počítadlo zmien dosiahne hodnotu vyššiu ako je hodnota tohto parametra tak sa počítadlo vynuluje a zmeny sa prenesú do štatistickej reprezentácie zhlukov v podobe mikrozhlukov. Tento parameter zároveň určuje kapacitu radu posledných sedení v modeli používateľa *u* (atribút *u.niq*, pozri časť 5.1.3).

- **Hraničný počet zmien v mikrozhlukoch** *(skr. tcm z ang. threshold number of changes in microclusters)*: V prípade, že došlo k dostatočnému počtu zmien v modeli používateľa sa tieto zmeny prenesú aj do do štatistickej reprezentácie zhlukov – tzv. mikrozhlukov. Počítadlo zmien v mikrozhlukoch je inkrementované s každým vykonaním tohto kroku o 1. Ak prekročí počet zmien v mikrozlukoch hodnotu danú týmto parametrom tak sa vynuluje a vykoná sa makrozhlukovanie.
- **Maximálny počet mikrozhlukov** *(skr. mmc z ang. maximal microclusters count)*: Viac o mikrozhlukoch v časti 4.2.1.

Parametre odporúčania:

- **Veľkosť okna vyhodnocovania** *(skr. ews z ang. evaluation window size)*: Počet posledných akcií v sedení používateľa, na základe ktorých sa vyberajú vzory správania na odporúčanie.
- **Počet odporúčaných položiek** *(skr. rc z ang. recommendations count)*.

Ostatné všeobecné parametre metódy:

- **Minimálna rýchlosť** *(skr. mts z ang. minimal transactions per second)*: Minimálna požiadavka na rýchlosť (meraná v jednotkách počet transakcií za sekundu). Ak pri niektorej časovo náročnej operácií ako napr. aktualizácia množín dôjde k spomaleniu pod túto hranicu tak sa výpočet preruší (nenájdu sa všetky vzory) a pokračuje sa ďalej.
- **Hraničná hodnota rozdielu identifikátorov zhlukovania** *(skr. tcdiff z ang. threshold of clustering id's difference)*: Hovorí aký je hraničný rozdiel medzi identifikátorom makrozhlukovania *u.lmid* v modeli používateľa *u* a identifikátorom aktuálneho makrozhlukovania *lmid*. Ak teda platí $lmid - u.lmid > tcdiff$ tak je model používateľa označený ako starý a odstránený.

6 Implementácia metódy

V tejto kapitole sa venujeme detailom implementácie metódy pre experimentálne účely. Metódu sme implementovali do rámca *MOA*, ktorý sme opísali v časti 4.3.2. V úvode kapitoly rozoberáme aké externé moduly sme použili na implementáciu a ako sme ich integrovali do podoby navrhnutej v predchádzajúcej kapitole 5. Opisujeme tiež niektoré modifikácie týchto modulov, ktoré sme vykonali a popisujeme spôsob implementácie ďalších opisovaných funkcionalít.

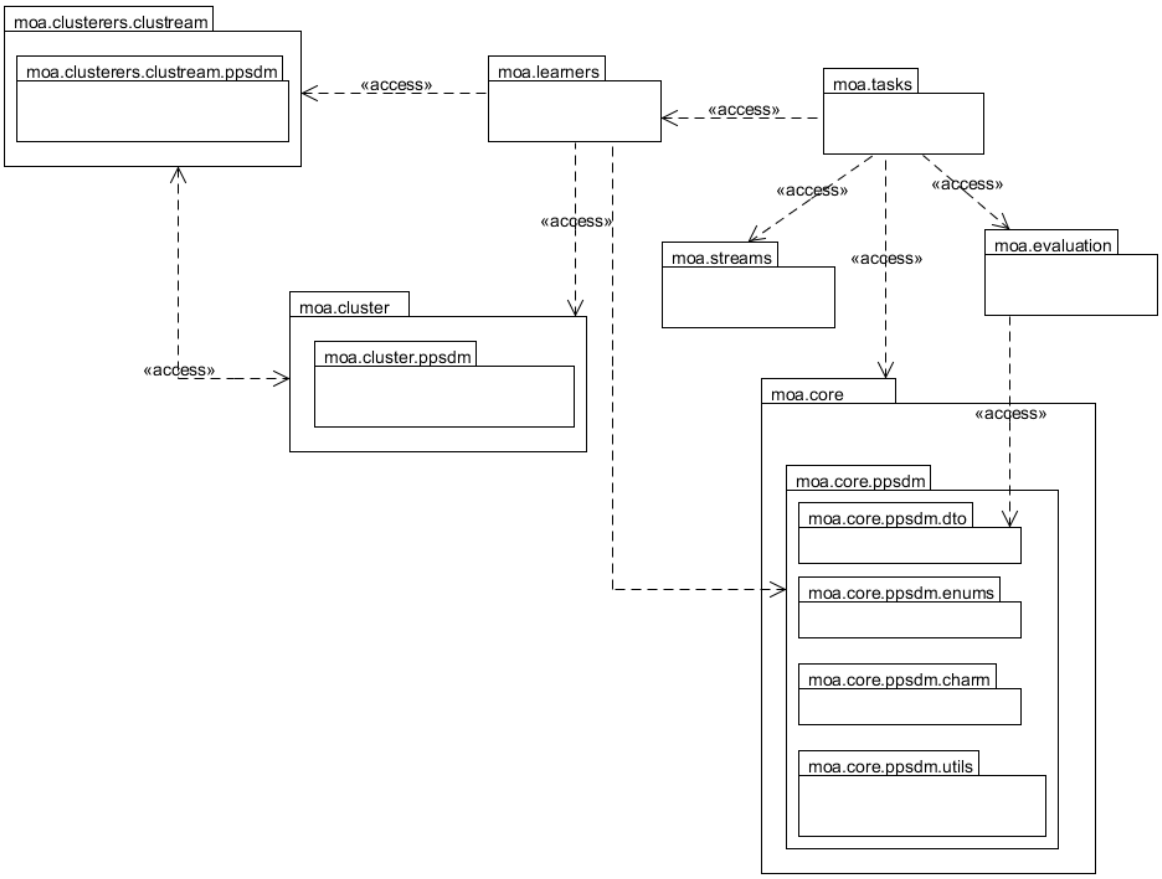
6.1 Architektúra riešenia

Na implementáciu metódy sme sa rozhodli použiť rámec *MOA* (opísaný v časti 4.3.2) aj vďaka dostupnosti viacerých algoritmov dolovania v prúdových dátach. V tomto rámci je ako rozšírenie možné použiť implementáciu algoritmu *IncMine* tak ako ju opisujú v práci (Quadrana, Bifet, & Gavaldà, An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System, 2015). Autori práce implementujú tento algoritmus s niekoľkými rozdielmi oproti pôvodnej práci (Cheng, Ke, & Ng, 2008):

1. Algoritmus implementovali tak, že používa pohyblivé okno fixovanej dĺžky namiesto pohyblivého okna s fixovaným časovým intervalom tak ako v pôvodnej práci.
2. Na dolovanie uzavretých frekventovaných množín, čo je kritický bod čo sa týka celkovej výkonnosti algoritmu použili algoritmus CHARM, konkrétne implementáciu z rámca SFPM (Fournier-Viger, 2016). Je to modifikovaná verzia algoritmu, ktorá používa bitové množiny na reprezentáciu množín identifikátorov transakcií vo vertikálnom formáte dát.

Podobne je v tomto rámci dostupná aj implementácia algoritmu zhlukovania *Clustream* z makrozhlukovaním *k-means* (pozri časť 4.2.1), ktorú používame na zhlukovanie modelov používateľov. Tento algoritmus je súčasťou rámca MOA a objavil sa aj v práci (Hassani & Seidl, 2016).

Na Obrázok 3 možno vidieť diagram balíkov. Balíky, ktoré obsahujú v názve *ppsdm* sú tie, ktoré obsahujú triedy, ktoré sme vytvorili alebo modifikovali tak aby zabezpečovali funkčnosť navrhovanej metódy. Funkcionalitu sme integrovali do balíkov rámca *MOA*. V ďalších častiach sa budeme venovať jednotlivým balíkom, ich účelu a niektorým dôležitým triedam.



Obrázok 3 Diagram balíkov

6.1.1 Balík *moa.core*

Tento balík je súčasťou rámca *MOA* kde obsahuje niekoľko veľmi dôležitých súčastí metódy PPSDM. Všetky tieto súčasti sme vložili do samostatného podbalíka s názvom *moa.core.ppsdm*.

Balík moa.core.ppsdm

V tomto balíku sa nachádzajú aj tieto triedy :

- *ConfigurationPPSDM* : statická konfigurácia metódy.
- *FciTablePPSDM* :

6.1.2 Trieda PersonalizedIncMine

6.1.3 Automatické čistenie tabuľky frekventovaných množín

Štruktúry na ukladanie a prácu so semi frekventovanými množinami v predchádzajúcej implentácií algoritmu IncMine neodstraňovali prázdne polia po uložených frekventovaných množínach v tabuľke ani polia s identifikátormi v invertovanom indexe. Hoci zabezpečili znovupoužitie týchto polí a nedochádzalo tak k neustálemu nárastu dátových štruktúr tak sme sa rozhodli vytvoriť mechanizmus, ktorý v pravidelných intervaloch tieto prázdne miesta odstráni. Tak ak aj dôjde k zmenám v prúde dát keď bolo napríklad nájdených mimoriadne veľa frekventovaných množín a následne sa počet

nachádzaných frekventovaných množín znížil bude dátová štruktúra zmenšená a nebude obsahovať zbytočne veľa prázdnych miest.

6.1.4 Oprava chyby s odstraňovaním položiek v zozname dobrých pozícií v tabuľke frekventovaných množín

Opravili sme chybu v implementácií algoritmu IncMine. Autori použili na zaznamenávanie dobrých pozícií (neprázdnych) v tabuľke frekventovaných množín ďalší zoznam. Položky v tomto zozname mazali nesprávne tak, že namiesto položky s uloženou pozíciou, ktorá sa mala vymazať vymazali položku na tejto pozícií v zozname. Dátovú štruktúru tohto zoznamu sme zmenili na množinu.

6.1.5 Regulovanie rýchlosti spracovania

7 Overenie metódy

Navrhovanú metódu overujeme z viacerých aspektov pričom v každom z nich pomocou pripravených experimentov overujeme vopred stanovenú hypotézu. Overované aspekty riešenia sú tieto:

- úspešnosť aplikácie nájdených vzorov správania na úlohu odporúčania,
- vzťahy medzi skupinovými a globálnymi vzormi,
- rýchlosť spracovania prúdu dát.

7.1 Spôsob overenia jednotlivých aspektov a opis použitých dát

V tejto podkapitole opisujeme spôsob akým sú vyhodnocované jednotlivé aspekty riešenia, metodológiu navrhnutých experimentov a tiež charakteristiky použitých dát.

7.1.1 Spôsob vyhodnotenia úspešnosti aplikácie na úlohu odporúčania

Stanovili sme počiatočnú hypotézu, ktorá je zameraná na aplikáciu vzorov správania v úlohe odporúčania. Jej znenie sme formulovali nasledovne:

Kombináciou skupinových vzorov správania s globálnymi dokážeme zlepšiť presnosť odporúčania v porovnaní s referenčným prístupom v ktorom použijeme len globálne vzory správania.

Odporúčanie sme vybrali ako jednu z možných aplikácií vzorov správania na ktorej vyhodnocujeme či kombinovanie globálnych vzorov so skupinovými, čo je nosnou témou navrhovanej metódy, prináša zlepšenie a v akej miere. Úspešnosť odporúčania hodnotíme pomocou štandardných metrík *presnosť* (ang. precision), a *ndcg* (ang. normalized discounted cumulative gain). Konkrétnejšie sme tieto metriky zisťovali pre počty odporúčaných položiek 1, 2, 3, 4, 5, 10, 15, keďže v zvolených doménach je väčšina sedení pomerne krátka (pozri časť 7.1.5 kde sú opísané použité dáta).

Vyhodnocovanie každej metriky začína až po počte spracovaných transakcií, ktorý je dostatočne veľký na to aby došlo k naplneniu aspoň jedného posúvajúceho sa okna v každej vyhodnocovanej konfigurácií parametrov. Eliminujeme tak rozdiely jednotlivých konfigurácií práve na začiatku spracovania prúdu, kedy konfigurácie s menšou dĺžkou segmentov a menšou dĺžkou okna dokážu odporúčať skôr ako konfigurácie s dlhšími segmentami a väčšou dĺžkou okna. Tento počet sme teda určili ako *max(ws)*max(sl)* teda maximálna dĺžka okna (*ws*) krát maximálna dĺžka segmentu (*sl*) spomedzi všetkých vyhodnocovaných konfigurácií parametrov (podrobnejší opis parametrov je v časti 5.3).

Každá prichádzajúca transakcia T (pod transakciou rozumieme zoznam akcií používateľa v rámci jedného sedenia – pozri časť 5.1.1) je rozdelená na trénovaciu množinu A (vyhodnocovacie okno) a testovaciu množinu B. Ak je dĺžka transakcie menšia ako súčet veľkosti vyhodnocovacieho okna (parameter *ews*) a počtu odporúčaných položiek (parameter *rc*), tak sa pre vyhodnocovanie ignoruje. Nech počet odporúčaných položiek je

N. Nech množina odporúčaných položiek je R. Nech H je bitový vektor dĺžky N, ktorý obsahuje 1 na i-tej pozícií iba ak i-ta odporúčaná položka z R patrí do prieniku $R \cap B$.

Presnosť vypočítame ako:

$$precision = \frac{|R \cap B|}{N} \quad (1)$$

Metrika NDCG, nás bude zaujímať najmä pri väčšom počte odporúčaných položiek, kedy budeme sledovať aj to či správne odporúčané položky boli vybrané medzi prvými.

$$NDCG = \frac{DCG}{IDCG} \quad (2)$$

Kde IDCG je :

$$IDCG = 1 + \sum_{i=2}^{N} \frac{1}{\log_2(i)} \quad (3)$$

DCG je :

$$DCG = H[0] + \sum_{i=2}^{N} \frac{H[i]}{\log_2(i)} \quad (4)$$

V každom behu experimentu sledujeme teda výsledné hodnoty týchto metrík, pre rôzne konfigurácie hodnôt vstupných parametrov. Aby sme mohli overiť stanovenú hypotézu tak pre každý beh experimentu sledujeme hodnoty metrík samostatne pre odporúčanie metódou využívajúcou kombináciu globálnych a skupinových vzorov (teda tak ako je opísaná v časti 5.1.4) a zároveň pre odporúčanie metódou využívajúcou len globálne vzory a len skupinové vzory. Rozdiely medzi výsledkami týchto metód nakoniec vyhodnotíme a potvrdíme ich signifikantnosť štatistickým jednoduchým t-testom.

7.1.2 Spôsob vyhodnotenia vzťahov medzi skupinovými a globálnymi vzormi

Stanovili sme počiatočnú hypotézu, ktorá je zameraná na odhalenie prínosu skupinových vzorov správania v získavaní znalostí o správaní používateľov. Jej znenie sme formulovali nasledovne:

Skupinové vzory správania nájdené navrhovanou metódou sú unikátne oproti globálnym vzorom správania.

V našej práci sa zameriavame na aplikáciu vzorov správania v úlohe odporúčania. Ako sme už uviedli v časti 2.2 existuje viacero možných aplikácií nájdených vzorov. V tejto časti skúmame vzťahy medzi skupinovými a globálnymi vzormi nájdenými prostredníctvom našej metódy.

V každom behu testovania generujeme v čase postupne niekoľko snímok aktuálneho stavu nájdených vzorov. Pre každú skupinu vzorov (globálne a skupinové) sledujeme:

- počty unikátnych vzorov, ktoré sme vedeli získať pomocou našej metódy,
- počet používateľov v skupine,
- pomer priemernej podpory unikátnych vzorov k priemernej podpore všetkých vzorov.

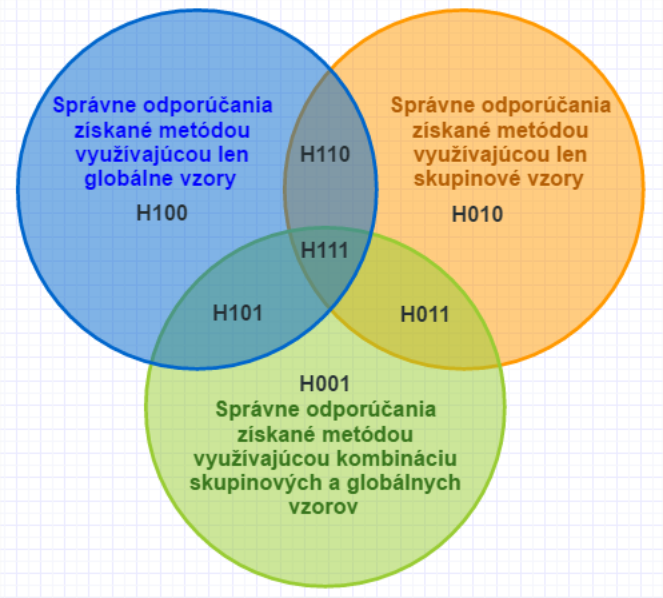
Okrem toho sledujeme počas behu experimentu pre každú transakciu počet správne odporúčaných položiek pre jednotlivé spôsoby odporúčania podľa toho aké vzory používajú. Jedná sa o tri samostatné množiny tak ako sú znázornené na vennovom diagrame (Graf 1). Ide o množiny všetkých správne odporúčaných položiek v sedeniach používateľov získaných pomocou metódy:

- využívajúcej len globálne vzory (1. množina),
- využívajúcej len skupinové vzory (2. množina),
- využívajúcej kombináciu skupinových a globálnych vzorov (3. množina).

Nech napríklad pre počet odporúčaných položiek 5 sú vektory správnosti odporúčania pre jednotlivé metódy odporúčania nasledovné. Hodnota 1 na i-tej pozícií znamená, že i-ta odporúčaná položka bola správna a 0, že odporúčana položka bola nesprávna:

- Vektor úspešnosti odporúčania získaný pomocou metódy odporúčania založenej na globálnych vzoroch: **10001**
- Vektor úspešnosti odporúčania získaný pomocou metódy odporúčania založenej na skupinových vzoroch:**11100**
- Vektor úspešnosti odporúčania získany pomocou metódy odporúčania založenej na kombinovaní skupinových a globálnych vzorov. Jedná sa teda o samostatnú množinu – pozri spôsob odporúčania opísaný v časti 5.1.4): **01001**

Tieto kombinácie označujeme ďalej ako písmeno **H** (od ang. hits) nasledované troma bitmi kde prvý bit označuje použitie globálnych vzorov, druhý bit označuje použitie skupinových vzorov, tretí bit označuje použitie kombinácie globálnych a skupinových vzorov. Teda napr *H101* predstavuje počet *úspešných* odporúčaní získaných pomocou globálnych vzorov a zároveň aj pomocou našej hybridnej metódy. Teda pre uvedený príklad by to bol prienik vektorov 10001 a 11100 čo je 10000. Hodnota *H101* je teda kardinalita tohto vektoru a to je 1. Pre jednoduchšie pochopenie sme znázornili uvedené množiny a ich prieniky pomocou vennovho diagramu (Graf 1).



Graf 1 Vennov diagram znázorňuje označenia jednotlivých množín vektorov úspešnosti odporúčania a ich prienikov.

Pomocou sumarizácie týchto množín a ich prienikov pre všetky sedenia používateľov vieme zistiť nasledujúce fakty, ktoré nás zaujímajú:

- Počet správnych odporúčaní len pomocou skupinových vzorov a nie globálnych: *H011+H010*
- Počet správnych odporúčaní len pomocou globálnych vzorov a nie skupinových: *H101+H100*
- Počet správnych odporúčaní len pomocou kombinácie skupinových a globálnych vzorov a nie iných: *H001*
- Počet správnych odporúčaní len pomocou skupinových vzorov, ktoré sme kombináciou skupinových a globálnych vzorov nevedeli správne odporučiť: *H010*
- Počet správnych odporúčaní pomocou globálnych vzorov, ktoré sme kombináciou skupinových a globálnych vzorov nevedeli správne odporučiť: *H100*
- Počet správnych odporúčaní získaných aj pomocou odporúčaní z globálnych aj skupinových vzorov aj ich kombinácie: *H111+H110*
- Teoreticky najvyšší možný počet správnych odporúčaní, ktoré by mohla metóda kombináciou skupinových a globálnych vzorov dosiahnuť: *H111+H110+H101+H100+H011+ H010*
- Počet správnych odporúčaní pomocou kombinácie skupinových aj globálnych vzorov: *H101+H011+H001+H111*
- Počet správnych odporúčaní pomocou globálnych vzorov: *H111+H110+H101+H100*
- Počet správnych odporúčaní pomocou skupinových vzorov: *H111+H011+H110+H010*

7.1.3 Spôsob vyhodnotenia rýchlosti spracovania transakcií

Pod rýchlosťou spracovania transakcií myslíme počet transakcií za časovú jednotku, ktorú dokáže naše riešenie spracovať. Keďže proces zahŕňa zhlukovanie v prude dát a hľadanie skupinových a globálnych uzavretých frekventovaných množín zároveň to prináša istú réžiu oproti riešeniu, ktoré hľadá len globálne uzavreté frekventované množiny. V tejto časti vyhodnotenia sa zameriavame práve na tento aspekt.

Dôležitým obmedzením v našom testovaní je obmedzenie rýchlosti. Rozhodli sme sa fixovať minimálnu povolenú hodnotu rýchlosti (v transakciách za sekundu) na 15 transakcií/sekunda a príjimame iba take konfigurácie, ktoré túto požiadavku spĺňajú.

Zaujíma nás aj aký je kompromis medzi znížením rýchlosti a zvýšením presnosti, konkrétne v úlohe odporúčania. Ako sme už pri návrhu metódy uviedli tak pokiaľ by išlo o reálnu aplikáciu bol by oddelený samostatný komponent pre zbieranie vzorov a samostatný pre generovanie odporúčaní. Preto testujeme rýchlosť samostne len pre úlohu zbierania vzorov.

Tento kompromis vyhodnocujeme nasledovným spôsobom:

1. Vyberieme najlepšiu konfiguráciu vo vyhodnotení úspešnosti odporúčania.
2. Odstránime komponent odporúčania.
3. N krát spustíme tieto konfigurácie so zapnutým a vypnutým zbieraním skupinových vzorov správania.
4. Vypočítame priemerný rozdiel v rýchlostiach týchto dvoch prípadov a tiež priemerný rozdiel v hodnotách metriky presnosť.

7.1.4 Metodológia hľadania najlepších konfigurácií parametrov

Vzhľadom na to, že navrhnutá metóda kombinuje dve existujúce metódy dolovania znalostí z prúdu dát, ktoré používajú viacero parametrov a sama vyžaduje od používateľa definovať hodnoty niekoľkých parametrov, sme sa rozhodli venovať podstatnú časť vyhodnotenia skúmaniu vplyvov zmien hodnôt týchto parametrov na úspešnosť v úlohe odporúčania.

Testovaním rôznych nastavení parametrov hľadáme ich najlepšiu konfiguráciu z hľadiska presnosti pri dodržaní minimálnej požiadavky na rýchlosť spracovania, ktorú sme stanovili na 15 transakcií za sekundu. Aby sme zmenšili počet rôznych skúmaných kombinácií tak sme parametre rozdelili do 4 skupín podľa ich vzájomného súvisu tak ako sme uviedli v časti 5.3. V rámci každej skupiny parametrov skúmame vplyv zmien konkrétnych parametrov na výsledky odporúčania. Po hľadaní najlepších konfigurácií v rámci jednej skupiny parametrov berieme ako vstupné nastavenie pre ostatné parametre metódy do testovania ďalšej skupiny parametrov jednu z najlepších konfigurácií z hľadiska presnosti odporúčania a rýchlosti. Z každej skupiny parametrov nakoniec vyberáme konfigurácie s najlepšími výsledkami čo sa týka kompromisu presnosti a rýchlosti. V závere nájdeme najlepšie z konfigurácií, ktoré vznikli kombináciami vybraných konfigurácií z jednotlivých skupín parametrov. Iniciálne nastavenie parametrov a voľbu skúšaných hodnôt parametrov v jednotlivých skupinách sme získali iniciálnym odhadom experta. Táto iniciálna konfigurácia parametrov je znázornená v Tabuľka 2 (parametre sú opísané v časti 5.3):

Tabuľka 2 Iniciálne nastavenie parametrov.

Skratka	Názov parametra	Hodnota
ms	minimálna podpora	0.01
rr	miera uvoľnenia	0.5
sl	dĺžka segmentu	100
mil	maximálna dĺžka množiny	10
ws	veľkosť okna	10
gc	počet skupín	4
tcu	hraničný počet zmien v modeli používateľa	10
tcm	hraničný počet zmien v mikrozhlukoch	100
mmc	maximálny počet mikrozhlukov	100
ews	veľkosť vyhodnocovacieho okna.	1
mts	minimálna požadovaná rýchlosť v transakciách za sekundu	15
tcdiff	hraničná hodnota rozdielu identifikátorov zhlukovania	5

7.1.5 Použité dáta

Na vyhodnotenie sme použili 2 datasety z rôznych webových sídel. Prvý dataset sú dáta z webových logov adaptívneho webového výučbového systému (skr. ALEF) (Bieliková, a iní, 2014). Druhý dataset sú dáta z novinového portálu. Predspracovanie dát pozostávalo z:

- Identifikácie sedení používateľov z webových logov pomocou stratégie založenej na predpoklade, že ďalšie sedenie používateľa začína po tom ako prekročil stanovený časový limit 30 minút (viac v časti 2.1).
- Odstránení sedení s dĺžkou 1 pri datasete ALEF, ktoré nevieme využiť na odporúčanie, keďže potrebujeme mať v každej transakcií aspoň jednu akciu na nájdenie vzoru a jednu na testovanie odporúčania. V datasete novinový portál sme odstránili aj sedenia s dĺžkou 2, ktorých bolo priveľké množstvo aby sme dostali menší dataset, ktorý budeme schopní spracovať v rozumnom čase so sedeniami, ktoré majú dostatok akcií aby zachytili zaujímavé vzory správania používateľov a bolo ich zároveň možné použiť na vyhodnocovanie.
- Transformácie do požadovaného dátového formátu, ako sme uviedli v časti 5.1.1.

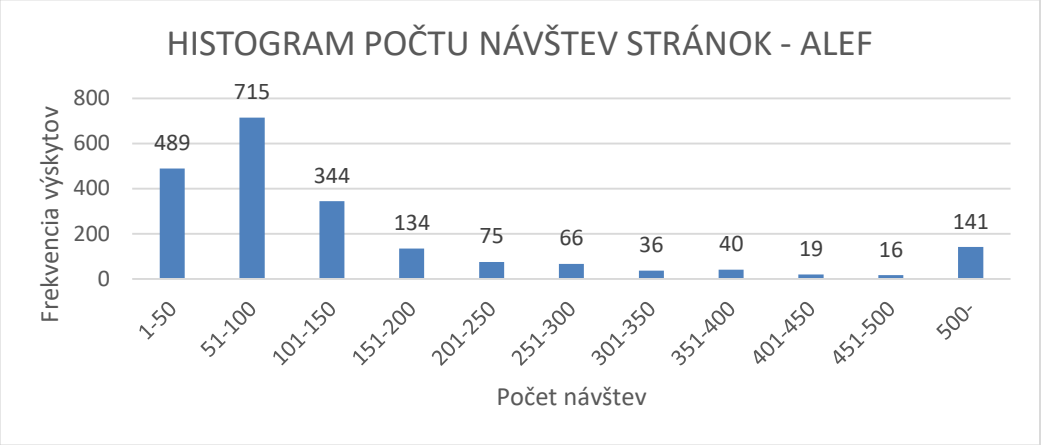
Dataset ALEF

Sedenia sú zoradené v chronologickej časovej následnosti. Výsledný predspracovaný dataset pozostáva z:

- 24 594 sedení,
- 870 používateľov,
- Dáta sú zbierané v období od 26/10/2010 do 30/04/2013. To je 917 dní. Kvôli tomu že ide o systém pre študentov tak aktivita používateľov logicky stúpa najmä v okolí skúškového obdobia a klesá v období prázdnin.
- 2 072 rôznych položiek (stránok),

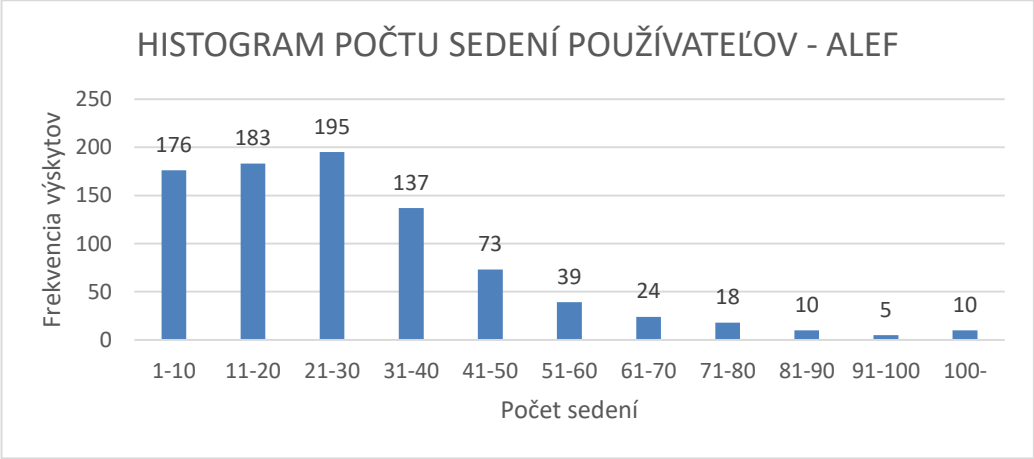
- Priemerná dĺžka sedenia je 15 navštívených stránok.

Na Obrázok 4 je znázornená distribúcia počtu návštev stránok používateľom. Môžeme vidieť, že stránky v datasete sú znovunavštevované pomerne veľakrát pretože sú to študijné materiály, ktoré sú aktuálne dlhší čas (ako by to bolo napríklad pri novinových článkoch).



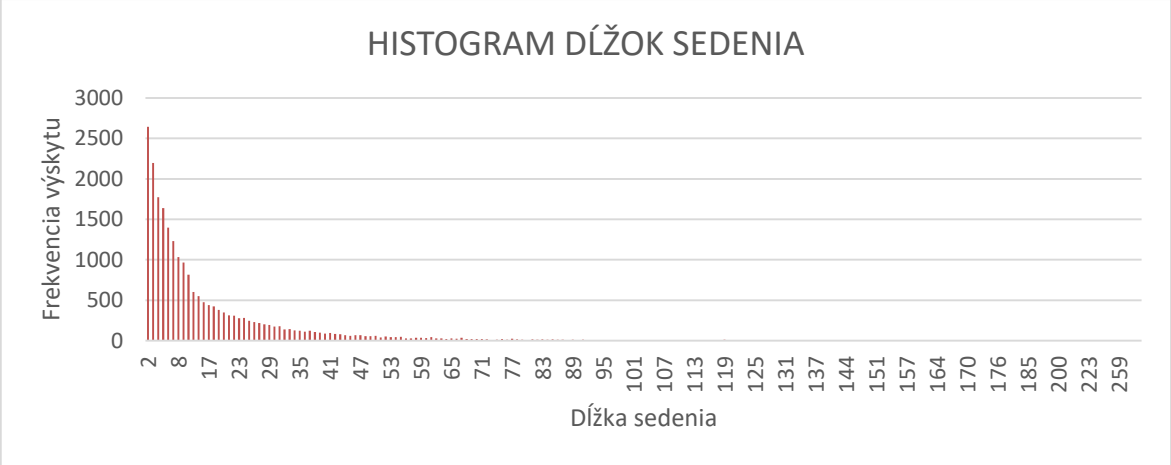
Obrázok 4 Histogram počtu návštev stránok používáteľmi

Na obrázku Obrázok 5 je znázornená distribúcia počtu sedení používateľov. Z hľadiska našej metódy je vhodné aby sa používatelia vracali čo najviac. V tom prípade je totiž nachádzaných viac skupinových vzorov a sú aj viac využívané.



Obrázok 5 Histogram počtu sedení používateľa

Takisto je dôležitý pohľad na distribúciu dĺžok sedení (Obrázok 6). Tu vidíme rýchle klesanie počtu sedení s väčšou dĺžkou. Tento fakt ovplyvňuje vyhodnocovanie. Pri zväčšovaní počtu odporúčaných položiek a veľkosti vyhodnocovacieho okna dochádza k odfiltrovaniu väčšieho množstva sedení, ktoré nemôžu byť použité na vyhodnocovanie. Zostávajú tak čoraz dlhšie sedenia, kde predpokladáme intuitívne vyššiu úspešnosť metódy, pretože sa jedná o aktívnejších používateľov s väčším počtom vykonaných akcií.



Obrázok 6 Histogram dĺžok sedení používateľov

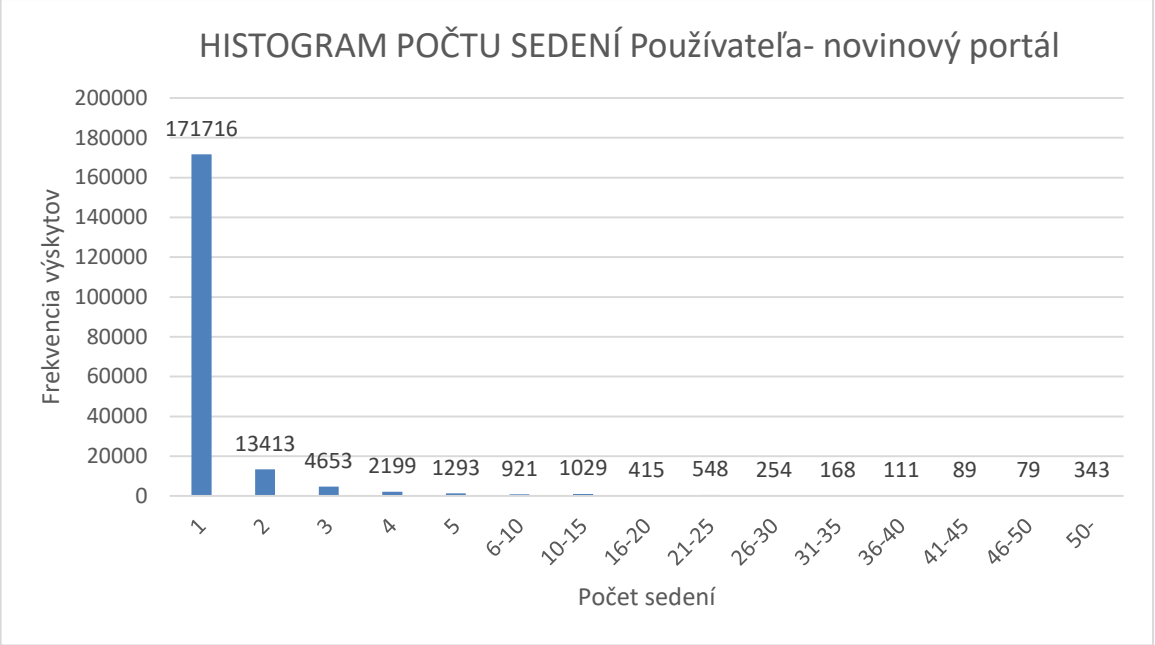
Dataset – novinový portál

Sedenia sú zoradené v chronologickej časovej následnosti. Výsledný predspracovaný dataset pozostáva z:

- 334 257 sedení,
- 199 196 používateľov,
- dáta sú zbierané v období od 01/03/2015 do 01/07/2015. To je 122 dní. Je to oveľa kratšie obdobie ako pre dataset ALEF ale zároveň s oveľa väčším počtom akcií.
- 85 rôznych položiek (kategorií stránok),
- priemerná dĺžka sedenia je 3 navštívené stránky

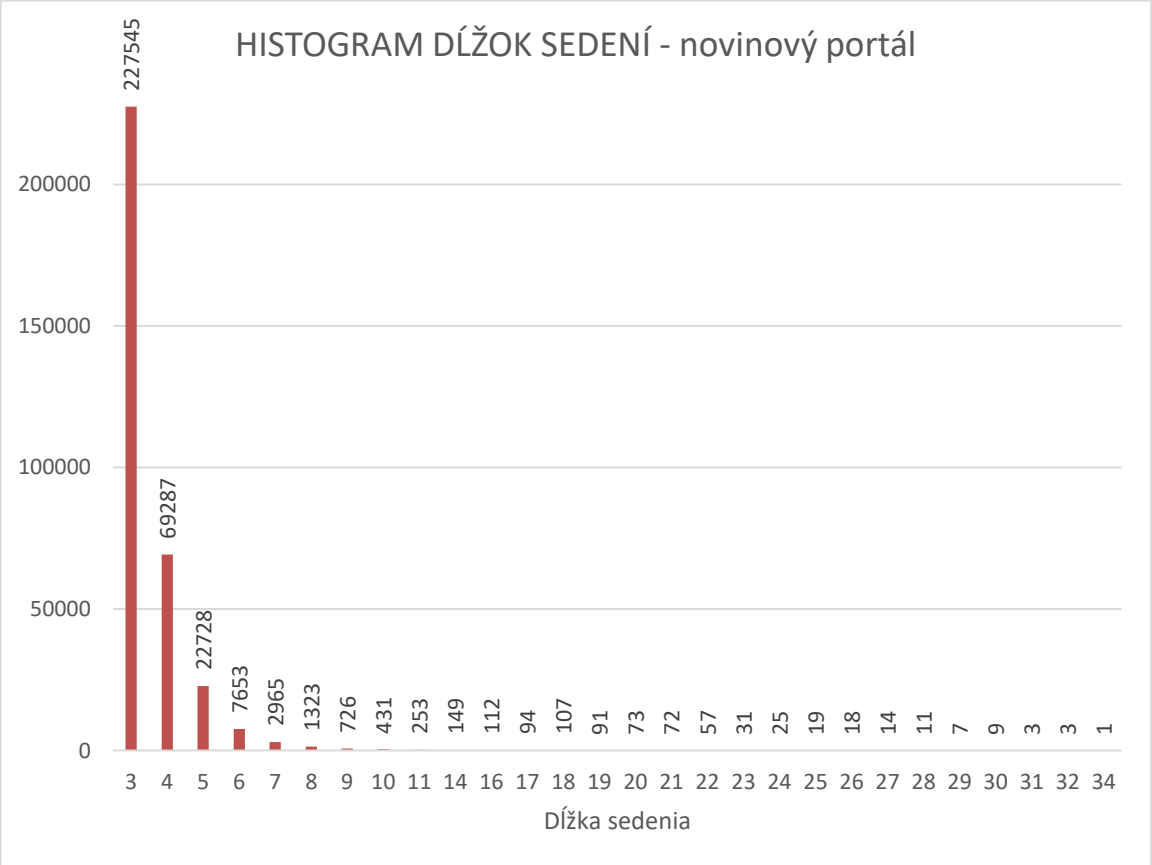
Oproti datasetu ALEF je tento dataset značne väčší čo sa týka počtu sedení aj počtu používateľov. Jedná sa o doménu kde sú typické veľmi krátke sedenia.

Na obrázku Obrázok 5 je znázornená distribúcia počtu sedení používateľov. Z hľadiska našej metódy je vhodné aby sa používatelia vracali čo najviac. Oproti datasetu ALEF je v tejto doméne práve tento problém, keďže veľmi veľké množstvo používateľov má zaznamenaných len málo sedení.



Obrázok 7 Histogram počtu sedení používateľa novinového portálu.

Na grafe je znázornená distribúcia dĺžok sedení. Pripomíname, že príliš krátke sedenia dĺžky 1 a 2, ktoré sú v tejto doméne veľmi časté sú v tejto podobe datasetu už odstránené. Oproti datasetu ALEF sú sedenia väčšinou kratšie.



Obrázok 8 Histogram dĺžok sedení používateľov v novinovom portáli.

7.2 Vyhodnotenie úspešnosti odporúčania pre jednotlivé skupiny parametrov v datasete ALEF

V tejto časti vyhodnocujeme úspešnosť odporúčania pre rôzne konfigurácie parametrov s použitím datasetu ALEF (opísaný v časti 7.1.4). Samostatne budeme vyhodnocovať jednotlivé skupiny parametrov uvedené v časti 5.3. Na záver vyberieme najlepšiu konfiguráciu s ktorou budeme pokračovať v ďalších experimentoch. Celá metodológia, ktorou sa riadime je popísaná v časti 7.1.1. Keďže detailné vyhodnotenie je pomerne rozsiahle v tejto podkapitole uvádzame len najzaujímavejšie poznatky. Viac detailných pozorovaní uvádzame v prílohe E a prílohe F a detailné výsledky v prílohe na elektronickom médiu.

7.2.1 Vyhodnotenie pre skupinu parametrov algoritmu hľadania frekventovaných množín

V tejto časti sa budeme venovať vyhodnoteniu úspešnosti odporúčania s rôznymi konfiguráciami skupiny parametrov pre algoritmus hľadania uzavretých frekventovaných množín v prúde dát *IncMine*. Ide o parametre (bližšie sú opísané v časti 7.1.4) a ich hodnoty:

- **Minimálna podpora** *(skr. ms)*: {0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1}
- **Miera uvoľnenia** *(skr. rr)*: {0.1, 0.5, 0.9}
- **Dĺžka segmentu** *(skr. sl)*: {25, 50, 100, 150, 200, 500}
- **Maximálna dĺžka množiny** *(skr. mil)*:{10}
- **Veľkosť okna** *(skr. ws)*: {5,10,15}

Spolu je to 272 rôznych konfigurácií. Čo sa týka hodnôt minimalnej podpory tá vyjadruje minimálnu podporu v rámci jedného segmentu. Teda ak dĺžka segmentu je napr. 50 a hodnota minimálnej podpory je 0.02 a menšia, tak je každá množina v segmente považovaná za frekventovanú. Kombinácie parametrov, kde bude teda hodnota $l * ms \leq 1$, budeme ignorovať pre vyhodnocovanie.

Tabuľka 3 Hodnoty skúmaných parametrov 5 najlepších konfigurácii podľa metriky presnosť. ms je minimálna podpora, rr je miera uvoľnenia, sl je dĺžka segmentu, mil je maximálna dĺžka množiny, ws je veľkosť okna.

Konfigurácia (dočasné ozn.)	ms	rr	sl	mil	ws
1	0.05	0.1	25	10	15
2	0.04	0.5	50	10	10
3	0.03	0.5	50	10	10
4	0.05	0.5	25	10	15
5	0.04	0.1	50	10	10

Na základe metriky *presnosť* sme vybrali najlepšie konfigurácie uvedené v Tabuľka 3. Ďalej sme sledovali vplyv jednotlivých parametrov a kombinácií parametrov na výsledky odporúčania. Zistili sme, že konfigurácie s kratšou dĺžkou segmentov dosahovali priemerne lepšie výsledky (*presnosť)* ako dlhšie segmenty. Tiež, že konfigurácie s príliš nízkou

hodnotou podpory a príliš vysokou hodnotou podpory dosahovali priemerne horšie výsledky. Príliš nízka hodnota *ms* spôsobuje generovanie veľkého množstva vzorov čo výrazne spomaľuje výpočet až na spodnú hranicu obmedzenia rýchlosti (15 transakcií za sekundu), čo má tiež za následok zhoršovanie výsledkov (pozri spôsob obmedzenia rýchlosti v časti 5.2.3). Príliš vysoká hodnota *ms* zas spôsobí, že veľa vzorov, ktoré by prispeli k dobrému výsledku sa nenájde. Ďalej sme zistili, že najlepšie výsledky sú dosahované pri nízkych hodnotách minimálnej podpory a zároveň nízkych hodnotách dĺžky segmentov. A tiež pre kombináciu krátkych segmentov a väčšej dĺžky okna. To znamená, že často dochádza k aktualizácií vzorov nad menšími časťami dát a počet segmentov v rámci okna je vyšší, čo znamená, že podpora vzorov sa počíta k dlhšej histórií dát.

Čo sa týka rýchlosti spracovania transakcií vo vybraných konfiguráciach, tak všetky konfigurácie okrem konfigurácie 1 (rýchlosť 61.45 transakcií za sekundu) spadli až na spodnú hranicu minimálnej požadovanej rýchlosti (15 transakcií za sekundu). Konfigurácia 1 má oproti konfigurácií 2 o polovicu kratšiu dĺžku segmentu a o *0.01* vyššiu hranicu minimálnej podpory. Je zaujímavé, že takáto relatívne malá zmena spôsobí významnú zmenu čo sa týka rýchlosti spracovania aj presnosti odporúčania. Ak teda je dôležitá rýchlosť a postačuje odporúčanie jednej položky je dobrou voľbou konfigurácia 1, ktorá je najpresnejšia práve pri odporúčaní jednej položky ak nie je rýchlosť až tak kritická a dôležitá je presnosť pri odporúčaní 2 a viac položiek tak je lepšou voľbou konfigurácia 2. Do záverečného vyhodnocovania sme teda vybrali hodnoty parametrov práve z konfigurácií 1 a 2.

7.2.2 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov zhlukovania

V tejto časti sa budeme venovať vyhodnoteniu úspešnosti odporúčania s rôznymi konfiguráciami skupiny parametrov pre algoritmus zhlukovania *CluStream*. Ide o parametre (bližšie sú opísané v časti 5.3) a ich hodnoty:

- **Počet zhlukov** *(skr. gc)*: {2, 4, 6, 8}
- **Hraničný počet zmien v modeli používateľa** *(skr. tcu)*: {5,10,15}
- **Hraničný počet zmien v mikrozhlukoch** *(skr. tcm)*: {50, 100, 200, 400, 800}
- **Maximálny počet mikrozhlukov***(skr. mmc)***:** {100,1000}

Spolu je to 120 rôznych konfigurácií. Nastavenie ostatných parametrov sme vybrali z jednej z najlepších konfigurácií z predchádzajúcej časti (konfigurácia 1 v Tabuľka 3**Error! Reference source not found.**), ktorá nebola najlepšia čo sa týka celkovej presnosti, ale dosahovala lepšie výsledky z hľadiska rýchlosti spracovania (zvolili sme teda kompromis medzi rýchlosťou a presnosťou).

Na základe metriky *presnosť* sme vybrali najlepšie konfigurácie uvedené v Tabuľka 4 aj s dočasným označením konfigurácií. Všetky najlepšie konfigurácie majú hodnotu parametra *tcu=5* a hodnotu parametra *tcm=100*. Čo sa týka počtu skupín sú hodnoty rôzne ale neobsahujú hodnotu 2. Podobne pre parameter *tcm* neobsahujú hodnoty nižšie ako 200.

Tabuľka 4 Najlepšie konfigurácie vybrané zo skupiny parametrov zhlukovania.

Konfigurácia (dočasné ozn.)	gc	tuc	tcm	Mmc
1	6	5	400	100
2	8	5	400	100
3	8	5	800	100
4	4	5	400	100
5	4	5	200	100

Nenašli sme nejaký jasný vzor medzi vybranými konfiguráciami čo sa týka výslednych hodnôt metrík *presnosť* a *ndcg* pre rôzne počty odporúčaných položiek. Rozdiely sú pomerne malé v metrike *presnosť* aj *ndcg.* Na prvých dvoch miestach v oboch metrikách sa striedajú konfigurácie 1 a 2.

Ďalej sme sledovali vplyv hodnôt jednotlivých parametrov a dvojíc parametrov na výsledky odporúčania. Zistili sme, že konfigurácie s počtom skupín 2 sú mierne horšie ako konfigurácie s vyšším počtom skupín, kde už väčšie rozdiely nebadáme. Je to spôsobené zrejme tým, že počet skupín 2 je príliš malý aby sa priblížil skutočnému rozdeleniu používateľov do skupín a pri väčšom počte skupín sú identifikované aj niektoré ďalšie dôležité skupiny.Takisto sme zistili, že najlepšie výsledky sú v konfiguráciach s najmenšou hodnotou parametra *tcu.* To znamená, že častejšie dochádza k aktualizácií mikrozhlukov a tým pádom aj k častejšiemu makrozhlukovaniu a tiež, že používateľ je posudzovaný (teda zaradzovaný do skupín) podľa jeho čo najaktuálnejšieho správania (posledných 5 sedení). Pre parameter *mmc* je situácia celkom zrejma, zistili sme, že vyšší počet mikrozhlukov neviedol k lepším výsledkom. Čo sa týka dvojíc hodnôt parametrov tak sme zistili, že najlepšia kombinácia hodnôt parametrov *gc* (počet skupín) a *tcdiff* (minimálny počet aktualizácií mikrozhlukov) je pri počte skupín *rc = 4* a *tcm = 200.* Čo sa týka rozdielov v rýchlostiach, tie už nie sú také značné ako pri rôznych konfiguráciach zo skupiny parametrov pre hľadanie frekventovaných množín, ale najlepšiu rýchlosť (58.41 transakcií za sekundu) dosiahla konfigurácia 3. Do záverečného vyhodnocovania sme teda vybrali hodnoty parametrov práve z konfigurácií 1 a 2 a kvôli vyššej rýchlosti aj konfigurácie 3.

7.2.3 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov odporúčania

V tejto časti sa budeme venovať vyhodnoteniu odporúčania v spojitosti s rôznymi konfiguráciami skupiny parametrov pre odporúčanie. Táto skupina je zvláštna, keďže pozorujeme len jeden parameter hoci tu patrí aj parameter *rc,* teda počet odporúčaných položiek, ktorého rôzne hodnoty ale sledujeme implicitne pri každom behu aj v ostatných skupinách parametrov. Ide teda o jediný parameter:

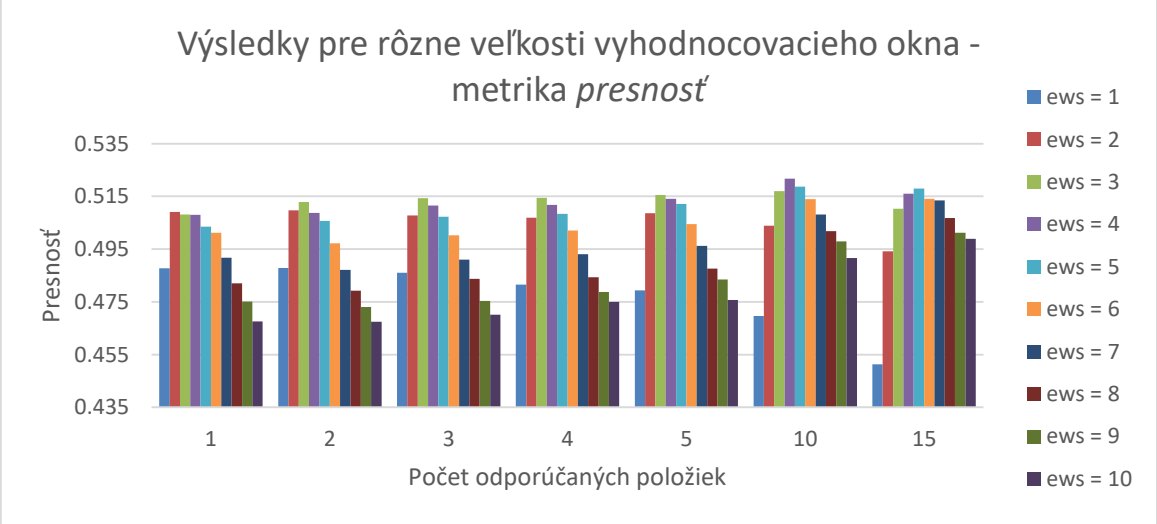
- **Veľkosť okna vyhodnocovania** *(skr. ews)*: {1,2,3,4,5,6,7, 8, 9, 10}

Nastavenie hodnôt ostatných parametrov sme zobrali z najlepšej konfigurácie čo sa týka presnosti vybranej z predchádzajúcej skupiny parametrov zhlukovania (konfigurácia 1 v Tabuľka 4).

Zmena hodnoty parametra *ews* mení aj množinu sedení, ktoré sú vyhodnocované. Čím dlhšie je vyhodnocovacie okno tým menej sedení má dostatočnú dĺžku aby mohli byť vyhodnotené. To je závažný fakt, ktorý má vplyv na relevantnosť porovnávania jednotlivých konfigurácií.

Na Graf 2 možno vidieť hodnoty metriky *presnosť* pre rôzne počty odporúčaných položiek a rôzne veľkosti vyhodnocovacieho okna. Môžeme si zvlášť všimnúť, že hodnoty parametra *ews* < *2* majú horšie výsledky než ostatné a hodnoty *ews* > *7* majú horšie výsledky najmä pri malom počte odporúčaných položiek. Malé vyhodnocovacie okno dáva priestor možnosti výberu veľkého množstva vzorov správania (aj kratších aj dlhších), ktoré majú s ním prienik a môžu byť zapojené do odporúčania. To je však zrejme aj čiastočne obmedzením, pretože okno je príliš malé aby rozpoznalo charakter sedenia a zámer používateľa. Pri väčších oknách sa *presnosť* opäť znižuje, čo je zrejme spôsobené aj tým, že neexistuje dostatok dlhších vzorov, ktoré by vedeli dané správanie používateľa správne charakterizovať.

Najlepšie výsledky *presnosti* sú pri hodnotách *ews* > *1* a zároveň *ews* < *7* pri počte odporúčaných položiek menej ako 10. Pri väčšom počte odporúčaných položiek sa javia najlepšie veľkosti *ews* > *2* a zároveň *ews* < *8*.



Graf 2 Výsledky metriky presnosť pre rôzne veľkosti vyhodnocovacieho okna ews pre rôzne počty odporúčaných položiek.

Tento parameter je teda špecifický a do výsledného hľadania najlepšej konfigurácie (časť 7.2.5) sme vybrali hodnoty *ews* z množiny {2, 3, 4, 5}, pri ktorých nebola množina vyhodnocovaných sedení priveľmi zmenšená a zároveň boli dosiahnuté najlepšie výsledky presnosti.

7.2.4 Vyhodnotenie úspešnosti odporúčania pre ostatné parametre metódy

V tejto časti sa budeme venovať vyhodnoteniu odporúčania v spojitosti s rôznymi konfiguráciami skupiny osatných parametrov metódy. Ide o parametre (bližšie sú opísané v časti 5.3) a ich hodnoty:

- **Hraničná hodnota rozdielu identifikátorov zhlukovania** *(skr. tcdiff)*: {1,2,3,4, 5,6,7,8}
- **Minimálna rýchlosť** *(skr. mts)*: {15}

V tejto skupine sme sa rozhodli testovať rôzne hodnoty len prvého parametra *tcdiff*. Rôzne hodnoty parametra pre obmedzenie minimálnej rýchlosti budeme testovať až vo vyhodnotení rýchlosti spracovania (7.3.2), kde sledujeme k akému zhoršeniu metriky *presnosť* bude dochádzať pri zrýchľovaní spracovania transakcií na už vybranej celkovo najlepšej konfigurácií. Nastavenie hodnôt ostatných parametrov sme zobrali z konfigurácie vybranej z predchádzajúcej skupiny parametrov odporúčania, kde sme pozorovali len jeden parameter určujúci veľkosť vyhodnocovacieho okna. Vybrali sme nastavenie vyhodnocovacieho okna s veľkosťou 2, ktoré príliš nezmenšuje množinu vyhodnocovaných sedení a zároveň dosahuje jedny z najlepších výsledkov metriky *presnosť*. Celkovo k makrozhlukovaniu dochádza 9 krát preto sme testovali rôzne hodnoty *tcdiff* po túto hodnotu

Sledovali sme zlepšovanie výsledkov so stúpajúcou hodnotou tohto parametra až po hodnotu 5. Staré modely používateľov by mali byť odstraňované aby sa tak zabránilo potencionálnemu zaplneniu pamäte a spomaľovaniu spracovania po dlhšej dobe spracovania. Dataset ALEF zachycuje pomerne dlhé obdobie počas ktorého došlo k viacerým zmenám záujmov konkrétnych používateľov (ako napr. ukončenie predmetu a začiatok nového predmetu). V tomto prípade zrejme existuje istá časová hranica po ktorej je vhodné model používateľa v prípade nečinnosti odstrániť pretože jeho stará verzia je už nerelevantná vzhľadom na nové záujmy používateľa. Inak to môže byť v iných doménach kde záujmy používateľov zostávajú s väčšou pravdepodobnosťou dlhšiu dobu rovnaké ako napr. novinový portál.

Do záverečného vyhodnotenia sme vybrali hodnotu parametra *tcdiff* = 5, pri ktorej boli výsledky o niečo málo lepšie než pri ostatných a pri ktorej sa sledovaný rastúci trend v skúmanom datasete pozastavil.

7.2.5 Celkové vyhodnotenie úspešnosti odporúčania pre najlepšie konfigurácie

Na základe vybraných najlepších konfigurácií zo všetkých skupín parametrov ako sme ich uvádzali v predchádzajúcich častiach sme skonštruovali priestor prehľadávania v ktorom budeme hľadať celkovo najlepšiu konfiguráciu. Ide o parametre a ich hodnoty:

- **Minimálna podpora** *(skr. ms):* {0.04, 0.05}
- **Miera uvoľnenia** *(skr. rr)*: {0.1, 0.5}
- **Dĺžka segmentu** *(skr. sl)*: {25, 50}
- **Maximálna dĺžka množiny** *(skr. mil)*: {10}

- **Veľkosť okna** *(ws)*: {10,15}
- **Počet zhlukov** *(skr. gc)*: {6, 8}
- **Hraničný počet zmien v modeli používateľa** *(skr. tcu)*: {5}
- **Hraničný počet zmien v mikrozhlukoch** *(skr. tcm)*: {400, 800}
- **Maximálny počet mikrozhlukov***(skr. mmc)***:** {100}
- **Veľkosť okna vyhodnocovania** *(skr. ews)*: {2,3,4,5}
- **Hraničná hodnota rozdielu identifikátorov zhlukovania** *(skr. tcdiff)*: {5}
- **Minimálna rýchlosť** *(skr. mts)*: {15}

V Tabuľka 5 uvádzame 7 konfigurácií, ktoré boli najlepšie pre niektorú z rôznych kombinácií hodnôt parametrov *ews* (veľkosť vyhodnocovacieho okna) a *rc* (počet odporúčaných položiek). Jediná z uvedených hodnôt ktorá sa medzi najlepšími konfiguráciami nenachádza je hodnota miery uvoľnenia *rr = 0.5*.

Tabuľka 5 Tabuľka zobrazuje hodnoty parametrov pre 7 najlepších konfigurácií celkovo pre dataset ALEF.

ID konfigurácie	ms	rr	sl	ws	gc	tcu	tcm	mmc	tcdiff	mts
1	0.05	0.1	25	15	6	5	400	100	5	15
2	0.04	0.1	50	15	8	5	800	100	5	15
3	0.05	0.1	50	15	8	5	400	100	5	15
4	0.04	0.1	50	10	6	5	400	100	5	15
5	0.05	0.1	50	15	6	5	800	100	5	15
6	0.04	0.1	50	10	8	5	800	100	5	15
7	0.05	0.1	50	15	6	5	400	100	5	15

Výsledky sme sledovali samostatne pre rôzne hodnoty parametra *ews*, ktorý, ako sme už spomínali, mení veľkosť priestoru vyhodnocovania. Zistili sme, že konfigurácia 1 vyniká naprieč rôznymi veľkosťami vyhodnocovacieho okna pri menšom počte odporúčaných položiek (<= 3). Pri väčšom počte odporúčaných položiek sa najlepšie konfigurácie striedajú. Konfigurácia 1 je výnimočná nielen kvôli svojím výsledkom čo sa týka metrík *presnosť* a *ndcg* ale aj čo sa týka rýchlosti. Všetky ostatné konfigurácie padli až na spodnú hranicu minimálnej požadovanej rýchlosti. Z tohto dôvodu sme sa rozhodli konfiguráciu 1 zobrať ako celkovo najlepšiu konfiguráciu hodnôt parametrov pre dataset ALEF a pokračujeme s ňou v ďalšom vyhodnocovaní.

Na záver sme ešte vyhodnotili priemerný rozdiel medzi výsledkami metódy využívajúcej len globálne vzory a metódy využívajúcej kombináciu skupinových a globálnych vzorov. Ten sa pohybuje okolo 1.7% pre rôzne počty odporúčaných položiek a výnimkou nie je ani vybraná konfigurácia 1, kde sa tento rozdiel pohybuje okolo 1.6% (pozri tabuľku

Tabuľka 6 a tabuľku Tabuľka 7). Tieto výsledky potvrdené jednoduchým štatistickým t-testom súhlasia s hypotézou, ktorú sme uviedli na začiatku kapitoly. Vykonali sme jednoduchý t-test medzi populáciami hodnôt výsledkov metriky *presnosť* všetkých vyhodnocovaných konfigurácií metódy využívajúcej kombináciu skupinových a globálnych vzorov a metódy využívajúcej len globálne vzory. Dosiahnutý výsledok je štandardne vyhodnotený ako vysoko štatisticky signifikantný ($p < 0.0001$, $t=9.8366$, $df=7740$ a rozdiel stredných hodnôt je 0.0165).

Tabuľka 6 Tabuľka zachytáva rozdiel medzi priemernými hodnotami metriky presnosť pre dve metódy odporúčania získané zo všetkých vyhodnocovaných konfigurácií a pre rôzne počty odporúčaných položiek v stĺpcoch.

	P@1	P@2	P@3	P@4	P@5	P@10	P@15
Kombinácia skupinových a globálnych vzorov	44.15%	44.66%	44.71%	44.63%	44.53%	43.79%	42.59%
Iba globálne vzory	42.47%	43.03%	43.09%	43.02%	42.91%	42.13%	40.87%
Rozdiel	1.68%	1.63%	1.62%	1.61%	1.61%	1.66%	1.72%

Tabuľka 7 Tabuľka zachytáva rozdiel medzi priemernými hodnotami metriky presnosť pre dve metódy odporúčania získané z najlepšej konfigurácie a pre rôzne počty odporúčaných položiek v stĺpcoch. Hodnoty v bunkách sú spriemerované pre rôzne hodnoty parametra ews z {2,3,4};

	P@1	P@2	P@3	P@4	P@5	P@10	P@15
Kombinácia skupinových a globálnych vzorov	50.73%	50.98%	51.09%	51.12%	51.34%	51.56%	50.93%
Iba globálne vzory	49.07%	49.51%	49.46%	49.52%	49.73%	49.73%	49.00%
Rozdiel	1.66%	1.47%	1.64%	1.59%	1.61%	1.83%	1.92%

7.3 Výsledné vyhodnotenie pre dataset ALEF

V tejto časti pre najlepšiu konfiguráciu parametrov nájdenú v predchádzajúcej časti vyhodnocujeme vzťahy medzi globálnymi a skupinovými vzormi a rýchlosť spracovania transakcií s a bez zbierania skupinových vzorov.

7.3.1 Vyhodnotenie vzťahov medzi skupinovými a globálnymi vzormi správania

Tak ako sme to uviedli v časti 7.1.2 tak pri vyhodnocovaní vzťahov sledujeme rôzne prieniky vektorov úspešnosti odporúčania získaných pomocou skupinových vzorov, globálnych vzorov a pomocou kombinácie týchto vzorov tak ako sme to navrhli v časti 5.1.4. V nasledujúcej Tabuľka 8 sú uvedené veľkosti všetkých prienikov pre vybranú najlepšiu konfiguráciu z vyhodnotenia úspešnosti odporúčania pomocou metriky *presnosť* (v predchádzajúcej časti 7.2.1). Pre pripomenutie označenie *H* je nasledované troma bitmi označujúce v poradí tri množiny správnych odporúčaní získaných metódou využívajúcou len globálné vzory, len skupinové vzory a ich kombináciou. Napr. označenie *H111* označuje spoločný prienik správnych odporúčaní zo všetkých troch prístupov.

Tabuľka 8 Veľkosti jednotlivých prienikov vektorov úspešnosti odporúčaní (pojem je vysvetlený v časti 7.1.2)

Počet odporúčaných položiek	H111	H110	H101	H100	H011	H010	H001	H000
1	26.45%	0.10%	20.70%	2.02%	3.23%	2.63%	0.32%	44.54%
2	24.92%	0.59%	21.28%	2.57%	3.72%	3.37%	0.84%	42.71%
3	23.78%	0.89%	21.69%	2.94%	3.92%	3.87%	1.17%	41.74%
4	23.14%	1.19%	22.03%	3.19%	4.08%	4.06%	1.50%	40.81%
5	22.60%	1.40%	22.17%	3.36%	4.29%	4.27%	1.79%	40.12%
10	20.63%	2.09%	21.60%	3.82%	4.66%	4.73%	2.68%	39.78%
15	18.54%	2.41%	20.82%	3.93%	4.71%	4.60%	3.16%	41.83%

V aplikácií našej metódy sa snažíme kombinovať skupinové a globálne vzory. Keby sme dokázali pomocou nášho spôsobu kombinovania skupinových a globálnych vzorov vždy správne rozhodnúť, ktorú množinu odporúčaní použiť (teda buď tú získanú pomocou skupinových vzorov alebo globálnych) vedeli by sme dosiahnuť najvyšší možný počet správnych odporúčaní vypočítaný ako **H100+H101+H010+H011+H110+H111** (pre 3 odporúčané položky je to **58.40%**). V skutočnosti sa nám podarilo dosiahnuť našou metódou počet správnych odporúčaní vypočítaný ako **H101+H011+H001+H111** (pre 3 odporúčané položky je to **51.73%**), čo je tiež zlepšenie oproti počtu správnych odporúčaní generovaných len globálnymi vzormi (pre 3 odporúčané položky je to **50.44%**) alebo len skupinovými vzormi (pre 3 odporúčané položiek je to **33.21%**). V dôsledku kombinácie vzorov boli tiež generované správne odporúčania aj v niektorých ďalších prípadoch, kedy negenerovali správne odporúčania ani skupinové ani globálne vzory samostatne (pre 3 odporúčané položky je to **1.19%**).

Môžeme si tiež všimnúť, že výrazný podiel tvorí počet správných odporúčaní generovaných zároveň globálnymi aj skupinovými vzormi vypočítaný ako **H111+H110**

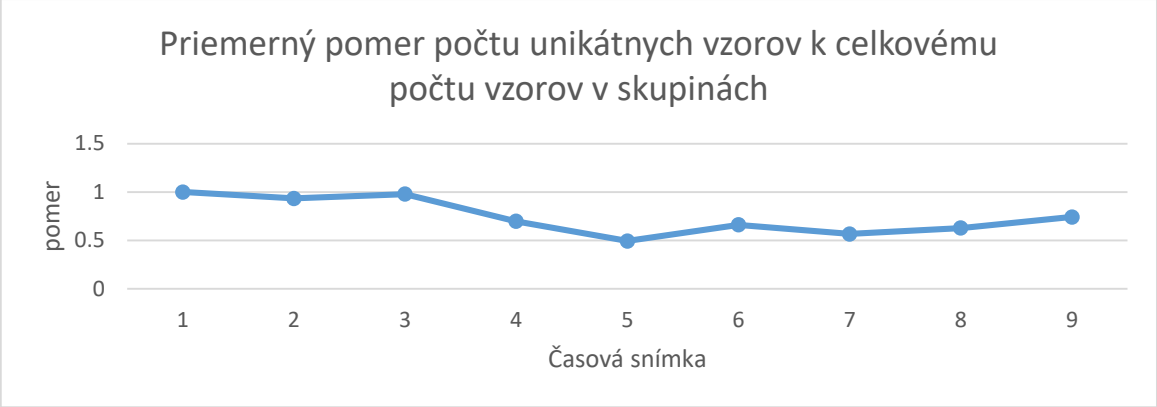
(pre 3 odporúčané položky je to **25.24%**). Toto môže poukazovať na podobnú výpovednú hodnotu skupinových aj globálnych vzorov pri úlohe odporúčania.

Počet správnych odporúčaní, ktoré dokázali vygenerovať len globálne vzory a skupinové vzory nie vypočítaný ako **H101+H100** (pre 3 odporúčané položky je to **25.19%**) je značne vyšší ako počet správnych odporúčaní, ktoré generovali skupinové vzory a globálne vzory nie vypočítaný ako **H011+H010** (pre 3 odporúčané položky je to **7.97%**). Toto môže byť spôsobené aj tým, že v skupinách nevznikalo tak vela silných vzorov ako v globálnom priestore, keďže aj príslušné dátové vzorky boli menšie. V prílohe F uvádzame detailnú tabuľku sumarizujúcu uvedené fakty a ďalšie získané fakty z analýzy dát z tabuľky Tabuľka 8.

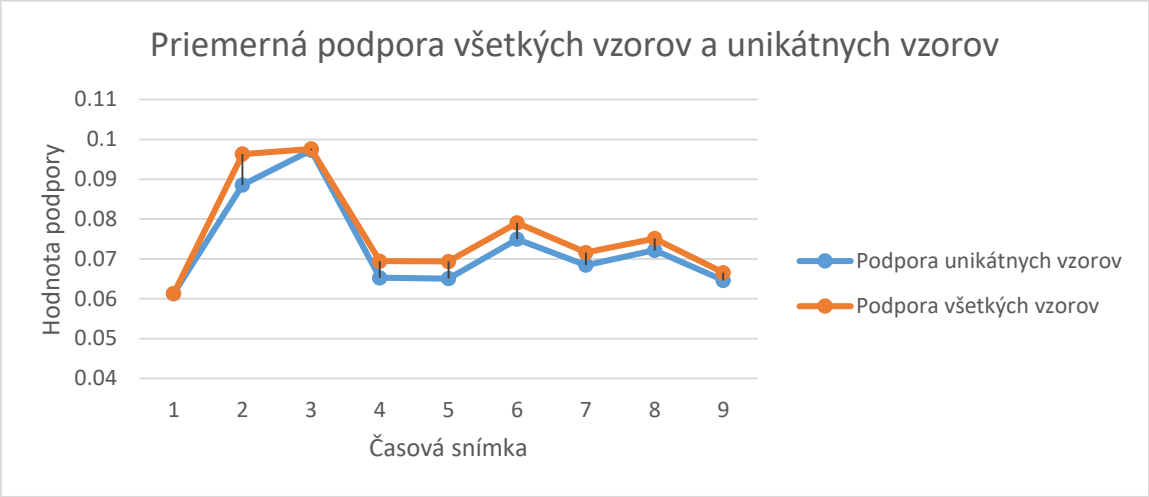
Sledovali sme tiež distribúciu počtu používateľov v skupinách a počty unikátnych vzorov v jednotlivých skupinách v pravidelných diskrétnych meraniach v čase spracovávania prúdu. Merania boli vykonané pravidelne pri dosiahnutí hodnoty *395* parametra *tcm*, teda tesne pred vykonaním ďalšieho makrozhlukovania. Za celý beh je to 9 snímok aktuálneho stavu. Všimli sme si prevahu jednej skupiny nad ostatnými vo väčšine snímok. Tiež sme sledovali výrazne kolísanie týchto počtov, zrejme korešpondujúce so zmenou záujmov používateľov, ktoré naša metóda zachytáva. To je tiež jednou z výhod hľadania aproximácie frekventovaných množín nad pohybujúcim sa oknom dát (ako sme to opísali v časti 4.1.2), že dokáže zachytávať aktuálne záujmy používateľov a ich zmeny v čase (teda prispôsobiť sa tzv. konceptuálnemu posunu). Grafy a detailnejšie výslekdy k spomenutým zisteniam uvádzame pre ich rozsiahlosť samostatne v prílohe F.

Na Graf 3 sledujeme aký je priemerný pomer počtu unikátnych vzorov v skupinách ku celkovému počtu vzorov. Vzor považujeme za unikátny ak sa taký vzor nevyskytuje v žiadnej inej skupine ani v množine globálnych vzorov. Tento pomer sa drží vždy nad hranicou 50%. Teda vždy aspoň polovica všetkých vzorov sú unikátne vzory. Každá skupina obsahuje v každej snímke minimálne 30% unikátnych vzorov naprieč skupinami. Tu sa nám už otvára odpoveď na stanovenú hypotézu na začiatku kapitoly, že skupinové vzory dokážu odhaliť nový pohľad na správanie používateľov v podobe unikátnych vzorov správania oproti globálnym vzorom správania.

Na Graf 4 Priemerná podpora všetkých vzorov a skupinových vzorov vidíme, že hodnota priemernej podpory unikátnych vzorov nie je malá a teda sú to silné vzory spomedzi všetkých, hoci je väčšinou o niečo málo menšia ako priemerná podpora všetkých vzorov.



Graf 3 Priemerný pomer počtu unikátnych vzorov v skupinách k celkovému počtu vzorov. Na osi x sú uvedené časové snímky.Snímka predstavuje moment v behu programu tesne pred makrozhlukovaním keď sa zozbierali uvedené dáta.



Graf 4 Priemerná podpora všetkých vzorov a skupinových vzorov. Na osi x sú uvedené časové snímky.Snímka predstavuje moment v behu programu tesne pred makrozhlukovaním keď sa zozbierali uvedené dáta.

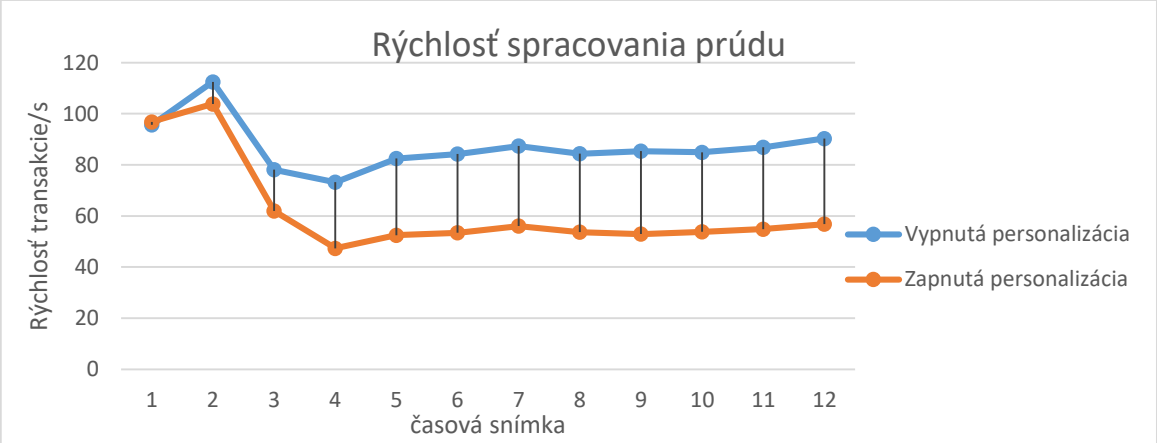
Pre vyhodnotenie vzťahov medzi globálnymi a skupinovými vzormi sme tiež sledovali hodnoty metriky *presnosť* pre odporúčanie pomocou globálnych vzorov, skupinových vzorov a ich kombinácie v snímkach počas behu programu. Pozorovali sme, že *presnosť* je nižšia pre skupinové vzory ako pre globálne vzory. To je spôsobené zrejme aj nedostatkom dát v skupinách, ktoré by generovali silné vzory s dobrou schopnosťou predikcie. Na grafoch v prílohe F uvádzame priebehy vývoja pre metriku *presnosť* v jednotlivých skupinách. V prílohe F tiež uvádzame príklad nájdených top vzorov v jednotlivých skupinách v niektorých časových snímkach v prehľadnej tabuľke, ktorá ilustruje, že dolovanie skupinových vzorov v prúde dát ponúka zaujímavý pohľad na správanie a zmeny správania používateľov v skupinách.

7.3.2 Vyhodnotenie rýchlosti spracovania transakcií

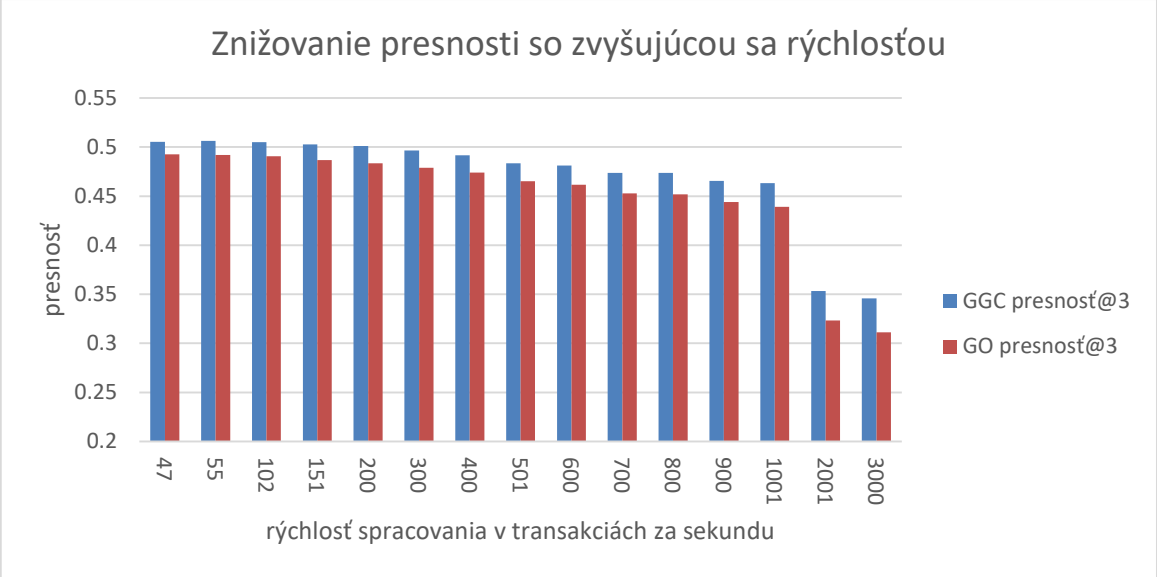
V tejto časti vyhodnocujeme kompromis medzi rýchlosťou spracovania transakcií v najlepšej vybranej konfigurácii podľa výsledkov metriky *presnosť* v úlohe odporúčania, ktorej vyhodnoteniu sa venovala časť 7.2.1. Na Graf 5 je možno vidieť porovnanie priemerných rýchlostí z 10 nezávislých behov so zapnutou personalizáciou (teda zhlukovanie a hľadanie skupinových vzorov správania) a z 10 nezávislých behov

s vypnutou personalizáciou. Možno si všimnúť, že rozdiel medzi rýchlosťami je viditeľný a jedná sa o rozdiel 35 transakcií za sekundu teda spomalenie o zhruba 35% pri poslednom meraní. Tento rozdiel hoci spočiatku má rastúci trend sa zdá byť na konci ustálený. Celkovo sme pri spomalení priemerne 35 transakcií za sekundu dosiahli zlepšenie o 1.67% *v presnosti* odporúčania (priemerne pre rôzne testované počty odporúčaných položiek) v prípade vybranej konfigurácie. Vzniká tu teda kompromis čo sa týka nižšej rýchlosti a vyššej *presnosti*. Toto je však v prípade, že nie je proces paralelizovaný tak ako sme to navrhli v časti 5.2.2, čo by malo viesť k zmenšeniu tohto rozdielu.

Okrem toho sme testovali účinok zvyšovania podmienky minimálnej požadovanej rýchlosti (vstupný parameter *mts*) na zmenu *presnosti*. Testovali sme hodnoty mts z množiny {50,100,150,200,300,400,500,600,700,800,900,1000,2000,3000}. Na grafe Graf 6 sú znázornené výsledné hodnoty presnosti pre 3 odporúčané položky a pre rôznu rýchlosť spracovania. Sledujeme očakávaný pokles presnosti pri zvyšovaní požiadavky na minimálnu rýchlosť a zachovanie a mierne zvyšovanie rozdielu presnosti medzi metódou využívajúcou len globálne vzory a metódou vyžívajúcou kombináciu skupinových a globálnych vzorov. V závislosti od požiadaviek aplikácie je teda možné prispôsobiť rýchlosť spracovania a požadovanej presnosti.



Graf 5 Rýchlosť spracovania prúdu dát.



Graf 6 Graf znázorňuje kompromis medzi znižujúcou sa rýchlosťou spracovania (na horizontálnej osi) a znižujúcou sa presnosťou pre 3 odporúčané položky. GGC označuje presnosť metódy využívajúcej kombináciu skupinových a globálnych vzorov a GO označuje presnosť metódy využívajúcej len globálne vzory správania.

8 Záver a zhodnotenie

Zoznam použitej literatúry

Agarwal, R. C., Aggarwal, C. C., & Prasad, C. C. (2001). A Tree Projection Algorithm for Generation of Frequent Item Sets. *Journal of Parallel and Distributed Computing*, 350-371.

Agarwal, R. C., Aggarwal, C. C., & Prasad, V. (2000). Depth first generation of long patterns. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (s. 108--118). ACM.

Aggarwal, C. C., Bhuiyan, M. A., & Hasan, M. A. (2014). Frequent Pattern Mining Algorithms:A Survey. In V. E. Lee, R. Jin, & G. Agrawal, *Frequent Pattern Mining* (s. 19-61). Springer International Publishing. doi:10.1007/978-3-319-07821-2

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2004). A Framework for Projected Clustering of High Dimensional Data Streams. *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30* (s. 852-863). Toronto, Canada: VLDB Endowment.

Aggarwal, C. C., Watson, T. J., Ctr, R., Han, J., Wang, J., & Yu, P. S. (2003). A Framework for Clustering Evolving Data Streams. *Proceedings of the 29th international conference on Very large data bases*, 81-92.

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proc. 20th int. conf. very large data bases, VLDB*, (s. 487-499).

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. *Int'l. Conf. Data Engineering (ICDE 95)*, 3–14.

Anandhi, D., & Irfan Ahmed, M. S. (2014). An improved web log mining and online navigational pattern prediction. *Research Journal of Applied Sciences, Engineering and Technology*, 1472-1479.

Ayres, J., Flannick, J., Gehrke, J., & Yiu, T. (2002). Sequential PAttern Mining Using a Bitmap Representation. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (s. 429-435). Edmonton, Alberta, Canada: ACM.

Barbará, D. (2003). Requirements for Clustering Data Streams. *ACM SIGKDD Explorations Newsletter*, 23-27.

Bayardo, J., & Roberto, J. (1998). Efficiently mining long patterns from databases. *ACM Sigmod Record*, 85--93.

Bieliková, M., Šimko, M., Barla, M., Tvarožek, J., Labaj, M., Móro, R., . . . Ševcech, J. (2014). ALEF: From Application to Platform for Adaptive Collaborative Learning. In *Recommender Systems for Technology Enhanced Learning: Research Trends and Applications* (s. 195-225). New York, NY: Springer New York. doi:10.1007/978-1-4939-0530-0_10

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). MOA: Massive Online Analysis. *The Journal of Machine Learning Research*, 1601-1604.

Bockermann, C., & Blom, H. (2012). Processing Data Streams with the RapidMiner Streams Plugin. *Proceedings of the 3rd RapidMiner Community Meeting and Conference*.

Burdick, D., Calimlim, M., & Gehrke, J. (2001). MAFIA: A maximal frequent itemset algorithm for transactional databases. *Data Engineering, 2001. Proceedings. 17th International Conference on* (s. 443-452). IEEE.

Calders, T., Dexters, N., Gillis, J. J., & Goethals, B. (2014). Mining frequent itemsets in a stream. *Information Systems*, 233-255.

Cao, F., Ester, M., Qian, W., & Zhou, A. (2006). Density-Based Clustering over an Evolving Data Stream with Noise. *Proceedings of the 2006 SIAM International Conference on Data Mining* (s. 328-339). SIAM.

Chang, J. H., & Lee, W. S. (2003). Finding recent frequent itemsets adaptively over online data streams. *KDD '03 - Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 487 - 492.

Chen, Y., & Tu, L. (2007). Density-based Clustering for Real-time Stream Data. *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (s. 133-142). New York, NY, USA: ACM.

Cheng, J., Ke, Y., & Ng, W. (2008). Maintaining frequent closed itemsets over a sliding window. *Journal of Intelligent Information Systems*, 191-215.

Chi, Y. a. (2006). Catch the moment: Maintaining closed frequent itemsets over a data stream sliding window. *Knowledge and Information Systems*, 265-294.

Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. (2004). Moment: Maintaining closed frequent itemsets over a stream sliding window. *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on* (s. 59-66). IEEE.

Etminani, K., Delui, A. R., Yenehsari, N. R., & Rouhani, M. (2009). Web Usage Mining: Discovery of the user's navagational patterns using SOM. *IEEE*, 244-249.

Facca, F. M., & Lanzi, P. L. (2003). Recent Developments in Web Usage Mining Research. In *Data Warehousing and Knowledge Discovery: 5th International Conference* (s. 140-150). Berlin: Springer Berlin Heidelberg. doi:10.1007/978-3-540-45228-7_15

Fournier-Viger, P. (07. 11 2016). Dostupné na Internete: SPMF: An Open-Source Data Mining Library: http://www.philippe-fournier-viger.com/spmf/index.php

Fu, Y., Shih, M.-Y., Creado, M., & Ju, C. (2002). Reorganizing web sites based on user access patterns. *International Journal of Intelligent Systems in Accounting, Finance & Management*, 39-53.

Han, J., Kamber, M., & Pei, J. (2012). 6 - Mining Frequent Patterns, Associations, and Correlations: Basic Concepts and Methods. In J. Han, M. Kamber, & J. Pei, *Data mining: concepts and techniques* (s. 243 - 278). Boston: Elsevier.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns Without Candidate Generation. *ACM Sigmod Record* (s. 1-12). ACM.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns Without Candidate Generation. *SIGMOD Rec.*, 1-12.

Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., & Hsu, M.-C. (2000). FreeSpan: frequent pattern-projected sequential pattern mining. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (s. 355-359). ACM.

Hassani, M., & Seidl, T. (2016). Using internal evaluation measures to validate the quality of diverse stream clustering algorithms. *Vietnam Journal of Computer Science*, 1--13.

Huntington, P., Nicholas, D., & Jamali, H. R. (2008). Website usage metrics: A re-assessment of session data. *Information Processing and Management, 44*(1), 358-372. doi:http://dx.doi.org/10.1016/j.ipm.2007.03.003

Jalali, M., Mustapha, N., Nasir Sulaiman, M., & Mamat, A. (2010). WebPUM: A Web-based recommendation system to predict user future movements. *Expert Systems With Applications, 37*, 6201-6212. doi:http://dx.doi.org/10.1016/j.eswa.2010.02.105

Kosala, R. a. (2000). Web Mining Research: A Survey. *ACM SIGKDD Explorations Newsletter*, 1-15.

Kranen, P., Assent, I., Baldauf, C., & Seidl, T. (2011). The ClusTree: indexing micro-clusters for anytime stream mining. *Knowledge and Information Systems*, 249-272.

Krempl, G., Spiliopoulou, M., Stefanowski, J., Žliobaite, I., Brzeziński, D., Hüllermeier, E., . . . Sievi, S. (2014). Open Challenges for Data Stream Mining Research. *ACM SIGKDD Explorations Newsletter*, 1-10.

Kum, H.-C., Pei, J., Wang, W., & Duncan, D. (2003). Approxmap: Approximate mining of consensus sequential patterns. *SDM Conference* (s. 311-315). SIAM.

Lee, V. E., Jin, R., & Agrawal, G. (2014). Frequent Pattern Mining in Data Streams. In V. E. Lee, R. Jin, & G. Agrawal, *Frequent Pattern Mining* (s. 199-220).

Liraki, Z., & Harounabadi, A. (2015). Predicting the Users ' Navigation Patterns in Web , using Weighted Association Rules and Users ' Navigation Information. *International Journal of Computer Applications*, 16-21.

Mobasher, B., Dai, H., Luo, T., Sun, Y., & Zhu, J. (2000). Integrating Web Usage and Content Mining for More Efective Personalization. 165-176.

Morales, G. D., & Bifet, A. (2015). SAMOA: Scalable Advanced Massive Online Analysis. *Journal of Machine Learning Research*, 149-153.

Pei, J., Han, J., & Mao, R. (2000). CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets. *ACM SIGMOD workshop on research issues in data mining and knowledge discovery*, (s. 21-30).

Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., & chun Hsu, M. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. *17th international conference on data engineering*, (s. 215-224).

Perkowitz, M., & Etzioni, O. (1997). Adaptive web sites: An AI challenge. *IJCAI International Joint Conference on Artificial Intelligence*, 16-21.

Pinto, H., Han, J., Pei, J., Wang, K., Chen, Q., & Dayal, U. (2001). Multi-dimensional sequential pattern mining. *Proceedings of the tenth international conference on Information and knowledge management* (s. 81-88). ACM.

Quadrana, M., & Mestre, R. G. (2012). Methods for Frequent Pattern Mining in Data Stream within the MOA System. Universitat Politecnica de Catalunya.

Quadrana, M., Bifet, A., & Gavaldà, R. (2015). An Efficient Closed Frequent Itemset Miner for the MOA Stream Mining System. *Ai Communications*, 143-158.

Savasere, A. a. (1995). *An efficient algorithm for mining association rules in large databases.* Georgia Institute of Technology.

Shen, W., Wang, J., & Han, J. (2014). Sequential Pattern Mining. In V. E. Lee, R. Jin, & G. Agrawal, *Frequent Pattern Mining* (s. 261-281).

Sisodia, D. S., & Verma, S. (2012). Web usage pattern analysis through web logs: A review. *JCSSE 2012 - 9th International Joint Conference on Computer Science and Software Engineering*, 49-53.

Spiliopoulou, M. a. (2003). A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis. *INFORMS J. on Computing*, 171--190.

Srikant, R., & Agrawal, E. (1996). Mining Sequential Patterns: Generalization and Performance Improvements. *5th International Conference on Extending Database Technology (EDBT '96)*, (s. 3-17).

Srivastava, M., Garg, R., & Mishra, P. K. (2014). Preprocessing Techniques in Web Usage Mining: A Survey. *International Journal of Computer Applications (0975 – 8887), Volume 97*, 1-9.

Teng, W. G., Chang, C. Y., & Chen, M. S. (2005). Integrating web caching and web prefetching in client-side proxies. *IEEE Transactions on Parallel and Distributed Systems*, 444-455.

Tsymbal, A. (2004). The Problem of Concept Drift: Definitions and Related Work. *Computer Science Department, Trinity College Dublin*.

Tzvetkov, P., Yan, X., & Han, J. (2005). TSP: Mining top-k closed sequential patterns. *Knowledge and Information Systems*, 438-457.

Vellingiri, J., Kaliraj, S., Satheeshkumar, S., & Parthiban, T. (2015). A Novel Approach for User Navigation Pattern Discovery and Analysis for Web Usage Mining. *Journal of Computer Science*, 372-382.

Yu, X., & Korkmaz, T. (2015). Heavy path based super-sequence frequent pattern mining on web log dataset. *Artif. Intell. Research*, 1–12.

Zaki, M. J. (2000). Scalable Algorithms for Association Mining. *IEEE Trans. on Knowl. and Data Eng.*, 372-390.

Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 31-60.

Zaki, M. J., & Gouda, K. (2003). Fast vertical mining using diffsets. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (s. 326--335). ACM.

Zaki, M. J., & Hsiao, C.-J. (2002). CHARM: An efficient algorithm for closed itemset mining. *Proceedings of the 2002 SIAM International Conference on Data Mining*, (s. 457--473).

PRÍLOHA A – Algoritmy dolovania frekventovaných množín v statických dátach

V tejto prílohe uvádzame podrobnejší opis niektorých známych algoritmov dolovania frekventovaných množín v statických datasetoch.

8.1 Algoritmus Apriori a jeho variácie

Názov tohto algoritmu, ktorý bol navrhnutý ako jeden z prvých algoritmov riešiacich úlohu hľadania frekventovaných množín v práci (Agrawal & Srikant, 1994), je odvodený od *apriori* znalosti, ktorú tento algoritmus používa. Táto *apriori* znalosť hovorí, že všetky podmnožiny frekventovaných množín sú tiež frekventované množiny. Algoritmus objavuje frekventované množiny iteratívne.

1. Najskôr sa nájdu všetky frekventované *1-množiny*. To si vyžaduje jeden prechod celou databázou – aby sa zistili počty každej *1-množiny*. Vyberú sa iba tie *1-množiny*, ktoré spĺňajú podmienku minimálnej podpory. Označme výsledné nájdené *1-množiny* ako L_1.
2. V ďalších iteráciách sa použije L_1 na nájdenie L_2 (čo je množina všetkých frekventovaných *2-množín*), L_2 sa použije na nájdenie L_3 a tak postupne až kým sa nenájdu žiadne ďalšie frekventované *k-množiny*.
3. Ak by sme nevyužili *a priori* znalosť tak by sme v každej iterácií z L_{k-1} museli generovať príliš veľké množstvo kandidátov na frekventované *k-množiny* patriace L_k.
4. Využitie *a priori* znalosti v algoritme *Apriori* spočíva v 2 krokoch:
    a. Generovanie kandidátov. Množina kandidátov C_k sa vytvorí spojením L_{k-1} s L_{k-1} . Táto operácia spojenia predpokladá, že položky v transakciách L_{k-1} sú zoradené. Spojenie prebehne pri dvojiciach transakcií, ktoré majú spoločných prvých k-2 položiek a rozdielne položky na pozícií k-1.
    b. Odstránenie kandidátov z C_k, ktoré nie sú frekventovanými množinami. Na základe *a priori* znalosti platí, že akákoľvek *(k-1)-množina*, ktorá nie je frekventovaná nemôže byť podmnožinou frekventovanej *k-množiny*. Skontrolujeme všetky *(k-1) podmnožiny* kandidáta. Ak nie sú všetky frekventované (čo vieme zistiť jednoducho keďže všetky frekventované *(k-1)-množiny* boli nájdené v predchádzajúcej iterácií) tak vieme, že ani kandidát nebude frekventovanou množinou.
5. Po vygenerovaní kandidátov je potrebné ešte prejsť každú transakciu v databáze aby sa vypočítala podpora jednotlivých kandidátov. Nakoniec sa kandidáti s nedostatočnou podporou z C_k odstránia a dostávame výslednú množinu frekventovaných *k-množín* L_k, ktorá sa stáva vstupom do ďalšej iterácie.

Aj keď *Apriori* výrazne zmenšuje množinu kandidátov na frekventované množiny , tak si nepočína dobre najmä pri dlhých frekventovaných vzoroch a nízkych hodnotách

minimálnej podpory. Nasledujúce tri uvedené problémy spôsobujú najvyššie straty na efektívnosti algoritmu (Han, Pei, & Yin, Mining Frequent Patterns Without Candidate Generation, 2000).

- V prvých iteráciách je problém s veľkým počtom generovaných kandidátov. Ak je počet frekventovaných *1-množín* 10^n tak počet generovaných kandidátov bude väčší ako 10^{2*n-1}.
- Ďalším problémom sú veľmi dlhé frekventované množiny. Napríklad na nájdenie frekventovanej množiny s dĺžkou 100 by sa muselo vygenerovať v priebehu algoritmu viac ako 2^{100} kandidátov.
- Výpočtovú náročnosť algoritmu tiež zvyšuje fakt, že sa vyžaduje v každej iterácií prechod celou databázou transakcií a hľadanie prítomnosti kandidátov v jednotlivých transakciách, kvôli výpočtu ich podpory. Na optimalizáciu tohto počítania bolo navrhnuté vylepšenie v práci (Agrawal & Srikant, Fast algorithms for mining association rules, 1994) kde sa používa dátová štruktúra *hash-tree* na ukladanie kandidátov C_k, kde list stromu je zoznam kandidátnych k-množín a vnútorné uzly sú hashovacie tabuľky. Konkrétna k-množina je namapovaná na konkrétny listový uzol pomocou aplikácie hašovaciích funkcií v tabuľkách od koreňa stromu až po listový uzol. V i-tej úrovni je táto funkcia aplikovaná na i-tu položku množiny. Položky v k-množinách sú usporiadané. Samotné počítanie prebieha nasledovne. Pre každú transakciu *t* sa v koreňovom uzle pokračuje vo vetvách, ktoré sa vyberú aplikáciou hash-funkcie na každú položku množiny. Vo vnútornych uzloch ak bola *i*-ta položka z *t* poslednou zahašovanou tak sa pokračuje vo vetvách, ktoré sa vyberú aplikáciou hašovacej funkcie na všetky položky za ňou. Pri každom nájdenom listovom uzle sa inkrementuje podpora všetkých jeho kandidátnych k-množín (Aggarwal, Bhuiyan, & Hasan, 2014).

Bolo navrhnutých viacero optimalizácií algoritmu Apriori. V pôvodnej práci (Agrawal & Srikant, 1994) boli navrhnuté optimalizácie *AprioriTID*, *AprioriHybrid.* Nech R(T, C_k) označuje množinu identifikátorov kandidátnych k-množín C_k, ktoré sa nachádzajú v transakcií T. V algoritme *AprioriTID* sa všetky takéto množiny uložia do novej databázy transakcií C'_k, ktorá sa použije namiesto C_k . Nová databáza je úspornejšia, transakcie sú v nej kratšie. V niektorých prípadoch nebude mať transakcia za podmnožinu žiadnu kandidátnu *k-množinu* a to hlavne v prípadoch keď je *k* väčšie. V tom prípade je transakcia vymazaná z C'_k čo má za následok zrýchlenie výpočtu. V niektorých prípadoch však bude mať transakcia za podmnožinu viac kandidátnych podmnožín čo je nežiadúce. Variácia *AprioriHybrid* používa toto vylepšenie až v neskorších iteráciach kedy je k-väčšie a teda prirodzene viacero transakcií z C'_k je odstraňovaných (Aggarwal, Bhuiyan, & Hasan, 2014).

8.2 FP-growth

FP-Growth (skrátené od ang. *frequent pattern growth*) je algoritmus navrhnutý v práci (Han, Pei, & Yin, 2000), ktorý sa pokúša prekonať popísané problémy algoritmu *Apriori*, ktoré spôsobujú jeho výpočtovú náročnosť spôsobenú najmä veľkým počtom generovaných kandidátov.

Autori v článku (Han, Pei, & Yin, 2000) navrhli dátovú štruktúru na ukladanie frekventovaných množín s názvom *FP-tree*. Pri konštrukcií *FP-tree* štruktúry vychádzajú z týchto poznatkov :

1. Je nutné vykonať minimálne jeden prechod databázou aby sa našli všetky frekventované 1-množiny.
2. Ak dokážu ukladať frekventované množiny nachádzajúce sa v jednotlivých transakciách do nejakej kompaktnej dátovej štruktúry tak nebude potrebné prechádzať celou databázou kvôli hľadaniu frekventovaných *k-množín* pre každé *k*.
3. Ak sa vo viacerých transakciách nachádza identická frekventovaná množina, môžu byť zlúčené a uložené ako jedna položka v dátovej štruktúre.
4. Predpokladajme, že položky v transakcií sú usporiadané zostupne podľa podpory jednotlivých položiek vzhľadom na celú databázu. Potom bude celkom veľká šanca, že viacero transakcií bude mať spoločný prefix. Ak majú dve transakcie spoločný prefix, tak túto spoločnú časť môžeme zlúčiť.

Spôsob konštrukcie *FP-tree* uvedieme na upravenom príklade z (Han, Pei, & Yin, Mining Frequent Patterns Without Candidate Generation, 2000). V Tabuľka 9 uvádzame príklad databázy transakcií s 5 transakciami, z ktorej zostrojíme *FP-tree.* Nech hodnota minimálnej podpory v tomto príklade je 3.

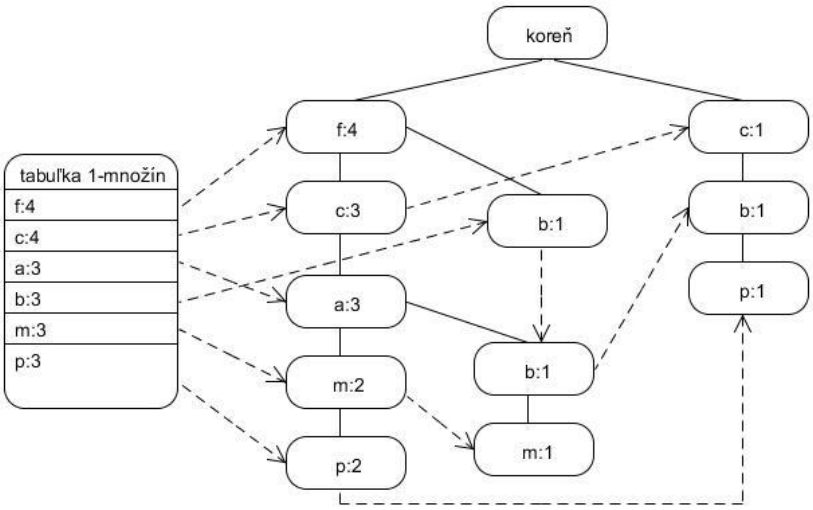
TID	Položky transakcie	Frekventované položky transakcie
1	f, a, c, d, g, i, m, p	f, c, a, m, p
2	a, b, c, f, l , m, o	f, c, a, b, m
3	b, f, h, j, o	f, b
4	b, c, k, s, p	c, b, p
5	a, f, c, e, l, p, m, n	f, c, a, m, p

Tabuľka 9 Príklad databázy transakcií

1. Najskôr prechádzame databázu a hľadáme frekvencie výskytov jednotlivých *1-množín*. Dostaneme tabuľku *1-množín* ako je uvedené na Obrázok 9, kde sú jednotlivé položky zoradené podľa počtu výskytov v databáze (počet výskytov je číslo uvedené pri položke).
2. Vytvoríme prázdny koreň stromu.
3. Znova prechádzame celú databázu transakcií.
    a. Pri prechode transakciou 1 zostrojíme prvú vetvu stromu (zľava): *<f:1,c:1,a:1, m:1, p:1>* s frekventovanými *1-množinami* z transakcie. Dôležité je, že zachovávame pri konštrukcií danej vetvy usporiadanie z tabuľky *1-množín*.

b. Pri prechode druhou transakciou by sme zostrojili vetvu <*f:1,c:1,a:1, b:1, m:1*>. Vidíme, že má spoločný prefix <*f,c,a*> s prvou vetvou. Zvýšime početnosť položiek so spoločného prefixu v prvej vetve a vytvorím podvetvu <*b:1,m:1*>, ktorú pripojíme k rodičovskému uzlu <*a:2*>
c. Podobne pri prechode treťou transakciou je zoznam frekventovaných položiek <*f:1,b:1*>. Spoločný prefix je iba <*f*>. Inkrementujeme početnosť uzlu <*f:2*> na <*f:3*> a vytvoríme nový uzol <*b:1*>, ktorý pripojíme k rodičovskému uzlu <*f:3*>.
d. Pri prechode štvrtou transakciou vytvoríme novú vetvu <*c:1, b:1,p:1*>. Keďže nezdieľa prefix s žiadnou inou transakciou, tak ju pripojíme priamo na koreň.
e. V poslednej transakcií máme vetvu <*f:1,c:1,a:1, m:1, p:1*>, ktorá je celá identická s prvou. Stačí teda inkrementovať počty v tejto vetve.

4. Aby bolo možné uskutočniť prechod stromom, z tabuľky *1-množín* zostrojíme ukazovatele na spájané zoznamy, ktoré spájajú uzly s rovnakým názvom *1-množiny* nachádzajúcej sa v strome.
5. Zostrojenú štruktúru vidíme na Obrázok 9.



Obrázok 9 Príklad zostrojenej štruktúry FP-tree.

V ďalšej časti práce (Han, Pei, & Yin, 2000) autori navrhli algoritmus, ktorý z takto vytvorenej štruktúry nájde všetky frekventované vzory, ktorý nazvali *FP-growth* (ang. *frequent pattern growth*).

Tento algoritmus využíva vlastnosti dátovej štruktúry *FP-tree*:

- Pre frekventovanú *1-množinu* je možné nájsť všetky frekventované množiny v ktorých sa vyskytuje nasledovaním liniek spájaného zoznamu, ktorý začína v tejto *1-množine*. Teda v našom príklade vieme napr. pre *1-množinu* <*p*> nájsť vetvy v ktorých sa nachádza a to sú : <*f:4, c:3, a:3, m:2, p:2*> a druhá <*c:1, b:1, p:1*>. Z prvej uvedenej vetvy vieme, že početnosť množiny <*f,c,a,m,p*> v databáze je 2, keďže prefix <*f,c,a,m*> sa nachádza spoločne s položkou *p* v databáze presne 2 krát.
- Na výpočet frekventovaných množín, ktoré obsahujú položku *i* je potrebné vo vetve stromu v ktorej sa nachádza uvažovať len časť vetvy, ktorej uzly sú predkami tejto

položky – teda prefixová časť vetvy. Tiež platí, že všetky uzly tejto prefixovej časti budú mať rovnakú početnosť ako položka *i*. Množinu všetkých prefixových časti pre všetky vetvy v ktorých sa položka *i* nachádza nazveme *podmienený základ vzoru* (ang. *conditional pattern base*). Označíme ju ako "*základ vzoru|i "*. Z tohto podmieneného základu vzoru vieme vypočítať všetky frekventované množiny obsahujúce položku *i* a to pomocou zostrojenia ďalšieho menšieho *FP-tree* a ďalším rekurzívnym hľadaním podmienených základov a konštruovaním ďalších *FP-tree*.

- Nech *a* je frekventovaná množina v databáze *D*. Nech *B* je *podmienený základ vzoru* a β je množina $\beta \subset B$. Potom platí že $\alpha \cup \beta$ je frekventovaná množina ak a iba ak je β frekventovaná množina.
- Pre *1-množiny* teda zostrojíme *podmienený základ vzoru*, čo je napríklad pre *1-množinu p* množina jej tzv. prefixových ciest s upravenou početnosťou. Teda : $\{< \mathrm{f}: 2, \mathrm{c}: 2, \mathrm{a}: 2, \mathrm{m}: 2 >, < \mathrm{c}: 1, \mathrm{b}: 1 >\}|p$.
  Pre *1-množinu m* je to zas: $\{< \mathrm{f}: 2, \mathrm{c}: 2, \mathrm{a}: 2 >, < \mathrm{f}: 1, \mathrm{c}: 1, \mathrm{a}: 1, \mathrm{b}: 1 >\} \mid m$.
- Ďalej pokračujeme rekurzívnym zostrojovaním podmienených *FP-tree* z nájdených *podmienených základov vzorov*. Pričom pre každú položku α_i z tabuľky frekventovaných *1-množín* nového *FP-tree* vytvárame rastúci vzor $\beta = \alpha_i \cup \alpha$, kde α je vzor, ktorý bol doposiaľ vytvorený (teda bol vstupom do rekurzívneho volania) pričom podpora β je rovná podpore položky α_i.
- Ďalej platí, že ak nový *FP-tree* má len jednu vetvu, tak nájdeme z nej všetky frekventované množiny tak, že nájdeme všetky kombinácie položiek tejto vetvy, ktoré spojíme s doposiaľ vytvoreným vzorom. Podpora každej kombinácie bude rovná podpore položky s minimálnou podporou v nej.
- Napríklad z podmieneného základu vzoru $\{< \mathrm{f}: 2, \mathrm{c}: 2, \mathrm{a}: 2, \mathrm{m}: 2 >, < \mathrm{c}: 1, \mathrm{b}: 1 >\}|\mathrm{p}$ zostrojíme *FP-tree*, ktorý bude mať len jednu vetvu *<c:3>*. Tak vieme povedať, že frekventovaná množina je *<c, p>* s podporou 3.
  Pre prípad rekurzívnej vetvy, ktorá začínala s *1-množinou m b*ude skonštruovaný *FP-tree* s jednou vetvou *<f:3,c:3,a:3>*. Z nej vieme nájsť všetky frekventované množiny obsahujúce *m*. Jednoducho vytvoríme z tejto vetvy všetky kombinácie, ktoré spojíme s doteraz nájdeným vzorom, ktorým je v tomto prípade zatiaľ len položka *m*. Budú to tieto frekventované množiny:
  <f,m>, <c,m>, <a,m>, <f,c,m>, <f,a,m>, <c,a,m>,<f,c,a,m>. Všetky budú mať podporu rovnakú ako je podpora najmenšej položky v kombinácií a to je vo všetkých prípadoch podpora položky *m*, ktorá je 2.

V uvedenej práci autori ukazujú, že algoritmus *FP-growth* je efektívnejší ako *Apriori* a jeho iné varianty založené na generovaní kandidátov podľa apriori vlastnosti aj v prípadoch krátkych aj dlhých frekventovaných množín.

8.3 Eclat

Eclat (z ang. Equivalence Class Transformation) je jeden zo 6 navrhovaných algoritmov dolovania frekventovaných množín v práci (Zaki M. J., 2000).
Všetky navrhované algoritmy sú postavené na podobnom princípe. Kľúčové prvky tohto princípu sú tieto :

1. Použitie vertikálneho formátu dát, teda indexu kde je pre každú množinu položiek uvedený zoznam transakcií v ktorých sa nachádza.
2. Rozdeľovanie priestoru prehľadávania do nezávislych častí.
3. Oddelenie úlohy rozdeľovania problému na menšie časti a úlohy hľadania frekventovaných množín v týcho častiach.
4. Minimalizovanie I/O operácií vďaka menšiemu počtu prechodov databázy. V najlepšom prípade len jeden.

Ako horizontálny formát dát je označovaný klasický formát, kde jeden záznam predstavuje jednu transakciu a pozostáva z identifikátora transakcie (skr. TID) ku ktorej je uvedený zoznam položiek transakcie. Ako vertikálny formát dát je označovaný formát, kde jeden záznam predstavuje množinu položiek ku ktorej je uvedený zoznam TID v ktorých sa nachádza. Transformácia horizontálneho formátu do vertikálneho formátu dát si vyžaduje jeden kompletný prechod databázou transakcií.

Platí, že TID zoznamy pre frekventované (k+1)-množiny je možné získať s využitím apriori znalosti prienikom všetkých párov TID zoznamov frekventovaných k-množín. Pričom platí, že kardinalita TID zoznamov bude so zvyšujúcim sa k klesať, čo prakticky znamená stale rýchlejší výpočet väčších frekventovaných množín. Kritický je teda najmä začiatok algoritmu, kedy z frekventovaných 1-množín generujeme frekventované 2-množiny.

Autori navrhli tiež spôsob ako výpočet rozdeliť do nezávislých častí a tak znížiť pamäťové zaťaženie (v pamäti nemusia byť ukladané všetky medzivýsledky) a umožniť aj prípadné paralelizovanie výpočtu. Nezávisle je možné hľadať všetky frekventované k+1-množiny pre jednotlivé triedy ekvivalentnosti na úrovni *k*. Dve *(k+1)*-množiny sú v rovnakej triede ekvivalentnosti definovanou reláciou θ_k ak zdieľajú rovnaký prefix dĺžky *k*. (predpokladáme, že TID zoznamy sú vždy udržiavané usporiadané).

Algoritmus môže byť potom založený na rekurzívnej dekompozícii každej triedy ekvivalentnosti definovanej reláciou θ_k do menších tried ekvivalentnosti. Samotné prehľadávanie priestoru riešení potom môže byť hľadaním do hĺbky alebo hľadaním do šírky. Pri hľadaní do šírky sa najskôr vygeneruje všetky k+1-frekventované množiny pre všetky triedy ekvivalentnosti na úrovni k.

8.4 CHARM

CHARM (Zaki & Hsiao, 2002) je algoritmus na dolovanie uzavretých frekventovaných množín, ktorý podobne ako algoritmus *Eclat* (Zaki M. J., 2000) využíva vertikálny formát transakčných dát. Nech P a Q sú množiny položiek. Nech $t(P)$ je zoznam identifikátorov transakcií v ktorých sa nachádza množina P. CHARM využíva tieto tri vlastnosti pre zmenšovanie prehľadávaného priestoru:

- Ak t (P) = t(Q), potom platí aj $t(P \cup Q) = t(P) \cap t(Q) = t(P)$
  To znamená, že každý výskyt P môžeme nahradiť $P \cup Q$ a odstrániť Q aj celú vetvu pod ním v prehľadávanom priestore, keďže obidve množiny majú rovnaký uzáver.
- Ak $t(P) \subset t(Q)$, potom platí aj: $t(P \cup Q) = t(P) \cap t(Q) = t(P) \neq t(Q)$. Takže môžeme nahradiť každý výskyt P množinou $P \cup Q$ pretože ak sa P vyskytuje v nejakej transakcií tak sa tam nachádza aj Q. Keďže však platí, že Q sa nachádza aj v iných transakciách tak nemožno odstrániť z prehľadávaného priestoru Q.
- Ak $t(P) \subset t(Q)$, potom platí aj $t(P \cup Q) = t(P) \cap t(Q) \neq t(Q) \neq t(P)$. V tomto prípade nemôže byť odstránená ani jedna množina, keďže P aj Q vedú k odlišným uzavretým frekventovaným množinám.

Vstupnými parametrami metódy je zoznam I dvojíc frekventovaných 1-množín s ich zoznamami identifikátorov transakcií a minimálna podpora.

1. Množiny v I sa zoradia podľa stúpajucej podpory (vypočítaná ako kardinalita množiny transakcií).
2. Pre každú množinu $P \in I$ sa algoritmus pokúsi rozšíriť ju o inú množinu $Q \in I$ a aplikovať uvedené 3 vlastnosti pre zmenšovanie prehľadávaného priestoru. Ak je novovytvorená množina z P a Q frekventovaná tak overuje či je uzavretá.
3. Algoritmus CHARM je volaný rekurzívne pre upravenú množinu I až pokiaľ táto nie je prázdna.

8.5 Citované práce

Aggarwal, C. C., Bhuiyan, M. A., & Hasan, M. A. (2014). Frequent Pattern Mining Algorithms:A Survey. In V. E. Lee, R. Jin, & G. Agrawal, *Frequent Pattern Mining* (s. 19-61).

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceeding VLDB '94 Proceedings of the 20th International Conference on Very Large Data Bases*, 487-499.

Han, J., Pei, J., & Yin, Y. (2000). Mining Frequent Patterns Without Candidate Generation. *SIGMOD Rec.*, 1-12.

Zaki, M. J. (2000). Scalable Algorithms for Association Mining. *IEEE Trans. on Knowl. and Data Eng.*, 372-390.

Zaki, M. J., & Hsiao, C.-J. (2002). CHARM: An efficient algorithm for closed itemset mining. *Proceedings of the 2002 SIAM International Conference on Data Mining*, (s. 457--473).

PRÍLOHA B – Algoritmy dolovania frekventovaných sekvencií v statických dátach

V tejto prílohe uvádzame podrobnejší opis niektorých známych algoritmov dolovania frekventovaných sekvencií v statických datasetoch.

8.6 GSP

8.7 SPADE

8.8 PrefixSpan

8.9 Citované práce

PRÍLOHA C – Algoritmy dolovania uzavretých frekventovaných množín v prúde dát

V tejto prílohe uvádzame podrobnejší opis niektorých známych algoritmov dolovania uzavretých frekventovaných množín v prúde dát.

8.10 Moment

Autori v práci . (Chi, Wang, Yu, & Muntz, 2004) navrhujú algoritmus hľadania uzavretých frekventovaných množín nad posúvajúcim sa oknom v prúde dát s názvom Moment. Je to prvá práca v ktorej navrhli inkrementálny prístup k dolovaniu uzavretých frekventovaných množín z prúdu dát. Okrem toho nájdené množiny nie sú len aproximáciou ale sú to exaktne nájdené uzavreté frekventované množiny aj napriek tomu, že algoritmus používa inkrementálne sa posúvajúce okno.

Autori navrhujú dátovú štruktúru CET (ang. closed enumeration tree), ktorá monitoruje uzavreté frekventované množiny a tiež tzv. okrajové množiny, ktoré tvoria hranicu medzi uzavretými frekventovanými množinami a ostatnými nefrekventovanými množinami. Každý uzol n_i tejto štruktúry reprezentuje množinu *I* s atribútom aktuálnej podpory *sup(I).* Táto štruktúra je podobná prefixovému stromu *FP-tree* používaného v algoritme *FP-Growth* (pozri ...). Rozdiel je, že v strome neudržiava všetky množiny ale len frekventované a okrajové. Pokiaľ nedochádza k zmene stavu množiny, tak stačí inkrementovať hodnotu podpory pre zahrnuté podmnožiny. V CET sú 4 typy uzlov (teda stavov množín):

- N_i je ***nefrekventovaný bránový uzol*** ak množina I, ktorú reprezentuje je nefrekventovaná. Rodičovský uzol N_j je frekventovaný a množina I vznikne spojením rodičovskej množiny J s jedným zo súrodeneckých uzlov J'.
- N_i je ***nenádejný bránový uzol*** ak I je frekventovaná množina a existuje uzavretá frekventovaná množina J taká, že $J \prec I \; a \; J \supset I \; a \; sup(I) = sup(J)$
- N_i je ***prechodný uzol*** ak *I* je frekventovaná množina a má ako potomka uzol N_j taký, že $sup(J) = sup(I) \; a \; N_j$ nie je nenádejný bránový uzol.
- N_i je ***uzavretý uzol*** reprezentujúci uzavretú frekventovanú množinu v aktuálnom okne. Môže to byť vnútorny uzol aj listový uzol stromu.

V práci ďalej dokazujú tieto tvrdenia o uzloch:

1. Ak je N_i ***nefrekventovaný bránový uzol*** tak akýkoľvek uzol N_j kde $J \supset I$ reprezentuje nefrekventovaný uzol (Vyplýva zo známej *apriori* vlastnosti). Tieto uzly N_j tak môžu byť odstránené (Takto sa výrazne znižuje počet uzlov v štruktúre). Uzol N_i predstavuje okrajový uzol, ktorý si algoritmus „pamätá" a ak sa stane frekventovaným tak vie z neho nájsť ďalšie frekventované množiny.
2. Ak je N_i ***nenádejný bránový uzol*** potom N_i nie je uzavretý uzol a žiadny z jeho potomkov nie je uzavretý. Preto sú potomkovia uzla N_i odrezaný, pretože uzol N_i si "pamätá" hranicu a vieme z neho odvodiť ďalšie neuzavreté frekventované množiny.

3. Ak je N_i ***prechodný uzol*** tak N_i nie je uzavretý a má potomkov uzavreté uzly.

Uzol N_i je teda štruktúrou :

- Informácia o type
- Množina *I*
- Podpora pre množinu *I – support(I)*
- Suma identifikátorov transakcií v ktorých sa *I* nachádza (*tid_sum(I))*

Na urýchlenie častej operácie kontroly, či daný uzol je alebo nie je nenádejný bránový uzol, navrhli použiť hash tabuľku so všetkými uzavretými frekventovanými množinami. Kľúčom je dvojica (*support(I), tid_sum(I)*).

Štruktúra CET sa inkrementálne aktualizuje vychádzajúc z aktuálneho okna. Aktuálne okno je reprezentované štruktúrou FP-tree (ako sme ju opísali v časti) s tým rozdielom, že obsahuje všetky množiny aj tie nefrekventované a navyše obsahuje tabuľku ukazovateľov TID na koncový uzol korešpondujúci s danou transakciou uloženou v strome.

Procedúra s názvom *Explore(N_i, min_support)* prehľadáva podstrom CET štruktúry do hĺbky, aktualizuje a označuje typy uzlov. Podprocedúra *leftcheck(N_i)* vie určiť či je N_i ***nenádejný bránový uzol*** tak, že zisťuje v hash tabuľke, či predtým už existovala uzavretá množina s rovnakou podporou ako N_i a obsahujúca *I.*
Pre každú prichádzajúcu transakciu T algoritmus aktualizuje hodnotu podpory a môže zmeniť typ korešpondujúceho uzla N_i v CET štruktúre.

- Ak N_i bol ***nefrekventovaný bránový uzol***. A teraz sa stáva frekventovaným tak pre každého ľavého súrodenca skúma či má byť vytvorený nový potomok s množinou obsahujúcou podmnožinu *I*. Keďže N_i bol okrajový uzol tak musí byť ďalej skúmaný aj jeho celý odrezaný podstrom pomocou procedúry *Explore.*
- Ak N_i bol ***nenádejný bránový uzol*** tak sa môže stať "nádejným". Ako nádejný uzol označujú autori uzol, ktorý v CET nachvíľu nie je ani ***nefrekventovaný bránový uzol*** ani ***nenádejný bránový uzol***. V tom prípade treba preskúmať jeho odrezaný podstrom pomocou procedúry *Explore.*
- Ak N_i bol ***prechodný uzol*** tak sa môže stat ***uzavretým***, Ak T obsahuje *I* ale žiadny z potomkov N_i , ktoré majú rovnakú podporu ako N_i pred príchodom T.
- Ak N_i bol ***uzavretý uzol*** tak zostane uzavretý.

Pre každú odchádzajúcu transakciu T algoritmus aktualizuje hodnotu podpory a môže zmeniť typ korešpondujúceho uzla N_i v CET štruktúre.

- Ak N_i bol ***nefrekventovaný bránový uzol*** tak sa nezmení.
- Ak N_i bol ***nenádejný bránový uzol*** môže sa stať nefrekventovaným a teda bude odstránený.
- Ak N_i bol ***nádejný uzol*** tak sa môže stať nenádejným (ak T obsahuje *I*). Použije sa procedúra *leftcheck(N_i).*

- Ak N_i bol ***uzavretý uzol*** tak môže prestať byť uzavretým uzlom. Zistíme to kontrolou hodnôt podpory potomkov tohto uzla.

8.11 CLAIM

8.12 Citované práce

Chi, Y., Wang, H., Yu, P. S., & Muntz, R. R. (2004). Moment: Maintaining closed frequent itemsets over a stream sliding window. *Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on* (s. 59-66). IEEE.

PRÍLOHA D – Algoritmy zhlukovania v prúde dát

V tejto prílohe uvádzame podrobnejší opis niektorých známych algoritmov zhlukovania v prúde dát.

8.13 Denstream

V práci (Cao, Ester, Qian, & Zhou, 2006) autori uvádzajú nové pojmy ako jadrový mikrozhluk (ang. core micro-cluster),odľahlý mikrozhluk (ang. outlier micro-cluster), ktoré umožňujú štatisticky reprezentovať zhluky rôznych tvarov.
Dáta prichádzajúce v prude môžu prechádza rôznymi zmenami. Meniť sa môžu počty zhlukov a tiež ich tvary. Dôležitou požiadavkou na zhlukovací algoritmus sa teda stáva schopnosť odhaľovať zhluky rôznych tvarov a rôzneho počtu zhlukov (bez nutnosti definovať používateľom).

8.14 ClusTree

8.15 Citované práce

Cao, F., Ester, M., Qian, W., & Zhou, A. (2006). Density-Based Clustering over an Evolving Data Stream with Noise. *Proceedings of the 2006 SIAM International Conference on Data Mining* (s. 328-339). SIAM.

PRÍLOHA E – Detailné vyhodnotenie úspešnosti odporúčania pre dataset ALEF

V tejto časti uvádzame detailné opisy vyhodnotenia úspešnosti odporúčania pre dataset ALEF.

8.16 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov hľadania frekventovaných množín

V tejto časti sa budeme venovať vyhodnoteniu úspešnosti odporúčania s rôznymi konfiguráciami skupiny parametrov pre algoritmus hľadania uzavretých frekventovaných množín v prúde dát *IncMine*. Ide o parametre (bližšie sú opísané v časti 7.1.4) a ich hodnoty:

- **Minimálna podpora** *(skr. ms)*: {0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1}
- **Miera uvoľnenia** *(skr. rr)*: {0.1, 0.5, 0.9}
- **Dĺžka segmentu** *(skr. sl)*: {25, 50, 100, 150, 200, 500}
- **Maximálna dĺžka množiny** *(skr. mil)*:{10}
- **Veľkosť okna** *(skr. ws)*: {5,10,15}

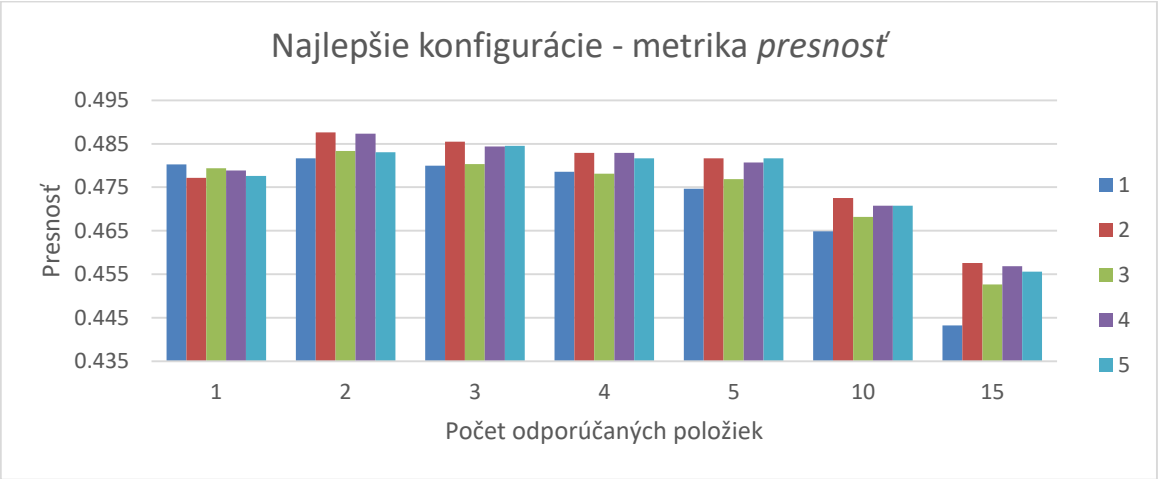
Spolu je to 270 rôznych konfigurácií. Čo sa týka hodnôt minimalnej podpory tá vyjadruje minimálnu podporu v rámci jedného segmentu. Teda ak dĺžka segmentu je napr. 50 a hodnota minimálnej podpory je 0.02 a menšia tak je každá množina považovaná za frekventovanú. Kombinácie parametrov kde bude teda hodnota $l * ms \leq 1$ budeme ignorovať.

Na **Error! Reference source not found.** možno vidieť hodnoty metriky *presnosť* pre rôzne počty odporúčaných položiek. Vybrali sme tie konfigurácie, ktoré boli medzi dvoma najlepšími výsledkami v odporúčaní pre jednotlivé počty odporúčaných položiek. V **Error! Reference source not found.** sú uvedené hodnoty skúmaných parametrov v týchto konfiguráciach aj s dočasným označením konfigurácií. Hodnoty metrík pre menšie počty odporúčaných položiek sú podobné, ale zvlášť konfigurácia *1* dosahuje najlepší výsledok pri počte odporúčaných položiek *1*. Pri väčšom počte odporúčaných položiek dosahuje najlepšie výsledky konfigurácia *2*. Keď sa ďalej pozrieme na výsledky metriky *NDCG* pre tieto konfigurácie uvedené na **Error! Reference source not found.** opäť dosahuje najlepšie výsledky konfigurácia 2.

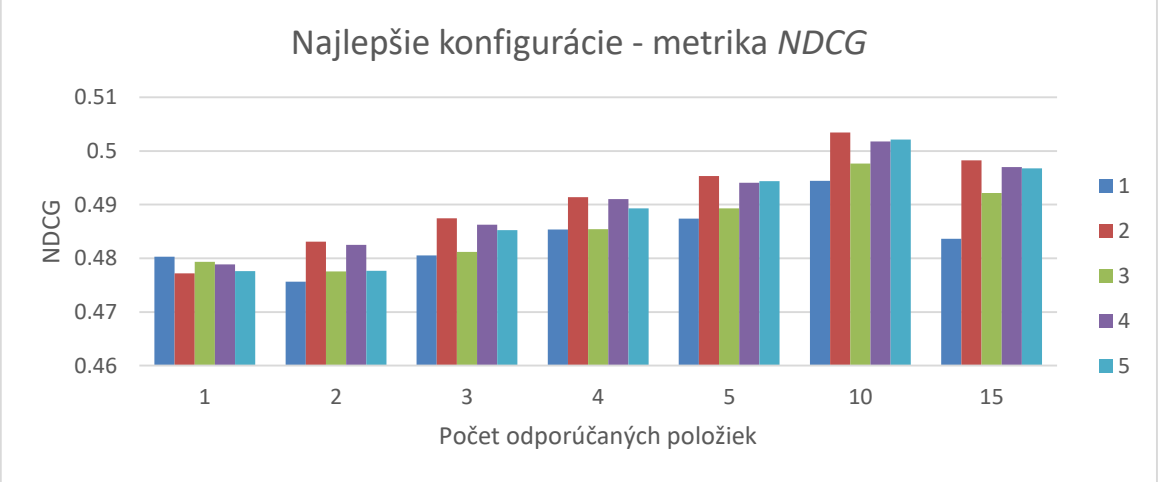
Tabuľka 10 Hodnoty skúmaných parametrov 5 najlepších konfigurácii

Konfigurácia (dočasné ozn.)	ms	rr	sl	mil	ws
1	0.05	0.1	25	10	15
2	0.04	0.1	50	10	15
3	0.04	0.1	50	10	10
4	0.03	0.5	50	10	15

5	0.05	0.1	50	10	15



Graf 7 Hodnoty metriky presnosť pre 5 najlepších konfigurácií zo skupiny parametrov hľadania frekventovaných množín.



Graf 8 Hodnoty metriky NDCG pre 5 najlepších konfigurácií zo skupiny parametrov hľadania frekventovaných množín.

Ďalej sme sledovali vplyv hodnôt jednotlivých parametrov a dvojíc parametrov na výsledky odporúčania. Zistili sme, že konfigurácie s kratšou dĺžkou segmentov dosahovali priemerne lepšie výsledky (*presnosť)* ako dlhšie segmenty. Najlepšie výsledky dávali konfigurácie s dĺžkou segmentu 50. Tiež, že konfigurácie s príliš nízkou hodnotou podpory a príliš vysokou hodnotou podpory dosahovali priemerne horšie výsledky. Príliš nízka hodnota *ms* spôsobuje generovanie veľkého množstva vzorov čo výrazne spomaľuje výpočet až na spodnú hranicu obmedzenia rýchlosti (15 transakcií za sekundu), čo má tiež za následok zhoršovanie výsledkov (pozri spôsob obmedzenia rýchlosti v časti 5.2.3). Príliš vysoká hodnota *ms* zas spôsobí, že veľa vzorov, ktoré by prispeli k dobrému výsledku sa nenájde. Takisto sme pozorovali, že priemerne horšie výsledky dávali tiež konfigurácie s vyššou hodnotou *rr* a vyššou hodnotou *ws*. Tu si možno všimnúť, že hoci sme sledovali takýto trend, tak medzi najlepšími konfiguráciami sú práve konfigurácie s vysokou hodnotou *ws*. Jednotlivé parametre sa navzájom ovplyvňujú. Napríklad zvýšením hodnoty

dĺžky segmentu *sl* sa mení účinok minimálnej podpory *ms*, ktorá sa počíta práve vzhľadom na dĺžku segmentu. Sledovali sme teda aj vplyv rôznych dvojíc parametrov na výsledky.

V Tabuľka 11 môžeme sledovať priemerné výsledky metriky *presnosť* pre rôzne kombinácie hodnôt parametrov *ms* a *sl*. Možno pozorovať, že najlepšie výsledky sú pri nízkych hodnotách minimálnej podpory a zároveň nízkych hodnotách dĺžky segmentov.

Tabuľka 11 Priemerné výsledky metriky presnosť pre rôzne kombinácie hodnôt parametrov ms a sl.

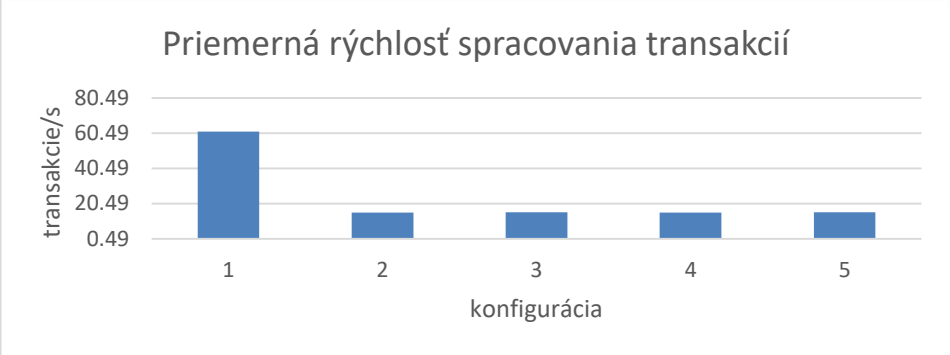
ms vs. *sl*	0.005	0.01	0.02	0.03	0.04	0.05	0.1
25						44.43%	41.44%
50				46.60%	46.38%	44.96%	41.07%
100			45.28%	44.00%	42.11%	40.66%	34.96%
150		43.71%	42.89%	40.18%	38.84%	37.93%	30.63%
200		44.01%	40.87%	40.76%	38.31%	35.75%	28.86%
500	37.69%	38.81%	36.92%	35.35%	32.74%	30.59%	28.40%

V ďalšej Tabuľka 12 si môžeme zas všimnúť, že najlepšie výsledky sú dosahované pre kombináciu krátkych segmentov a väčšej dĺžky okna. To znamená, že často dochádza k aktualizácií vzorov nad menšími časťami dát a počet segmentov v rámci okna je vyšší, čo znamená, že podpora vzorov sa počíta k dlhšej histórií dát.

Tabuľka 12 Priemerné výsledky metriky *presnosť* pre rôzne kombinácie hodnôt parametrov *sl* a *ws*

sl vs *ws*	25	50	100	150	200	500
5	0.4018	0.4355	0.4222	0.4092	0.3978	0.3432
10	0.4354	0.454	0.4202	0.3882	0.3821	0.3533
15	0.451	0.453	0.3996	0.3736	0.363	0.3343

Na Graf 9 vidíme priemernú rýchlosť spracovania transakcií jednotlivými konfiguráciami. Všetky konfigurácie okrem konfigurácie 1 spadli až na spodnú hranicu minimálnej požadovanej rýchlosti. Konfigurácia 1 má oproti konfigurácií 2 o polovicu kratšiu dĺžku segmentu a o *0.01* vyššiu hranicu minimálnej podpory. Je zaujímavé, že takáto relatívne malá zmena spôsobí významnú zmenu čo sa týka rýchlosti spracovania aj presnosti odporúčania. Ak teda je dôležitá rýchlosť je dobrou voľbou konfigurácia 1 ak nie je rýchlosť až tak kritická a dôležitá je presnosť tak je lepšou voľbou konfigurácia 2. Do konečného vyhodnocovania teda vyberáme hodnoty z konfigurácií 1 a 2.



Graf 9 Priemerná rýchlosť spracovania transakcií.

8.17 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov zhlukovania

V tejto časti sa budeme venovať vyhodnoteniu úspešnosti odporúčania s rôznymi konfiguráciami skupiny parametrov pre algoritmus zhlukovania Clustream. Ide o parametre (bližšie sú opísané v časti 0) a ich hodnoty:

- **Počet zhlukov** *(skr. gc)*: {2, 4, 6, 8}
- **Hraničný počet zmien v modeli používateľa** *(skr. tuc)*: {5,10,15}
- **Hraničný počet zmien v mikrozhlukoch** *(skr. tcm)*: {50, 100, 200, 400, 800}
- **Maximálny počet mikrozhlukov***(skr. mmc)***:** {100,1000}

Spolu je to 120 rôznych konfigurácií. Nastavenie ostatných parametrov sme vybrali z jednej z najlepších konfigurácií z predchádzajúcej časti (konfigurácia 1 v **Error! Reference source not found.**), ktorá nebola najlepšia čo sa týka celkovej presnosti, ale dosahovala lepšie výsledky z hľadiska rýchlosti spracovania (zvolili sme teda istý kompromis medzi rýchlosťou a presnosťou).

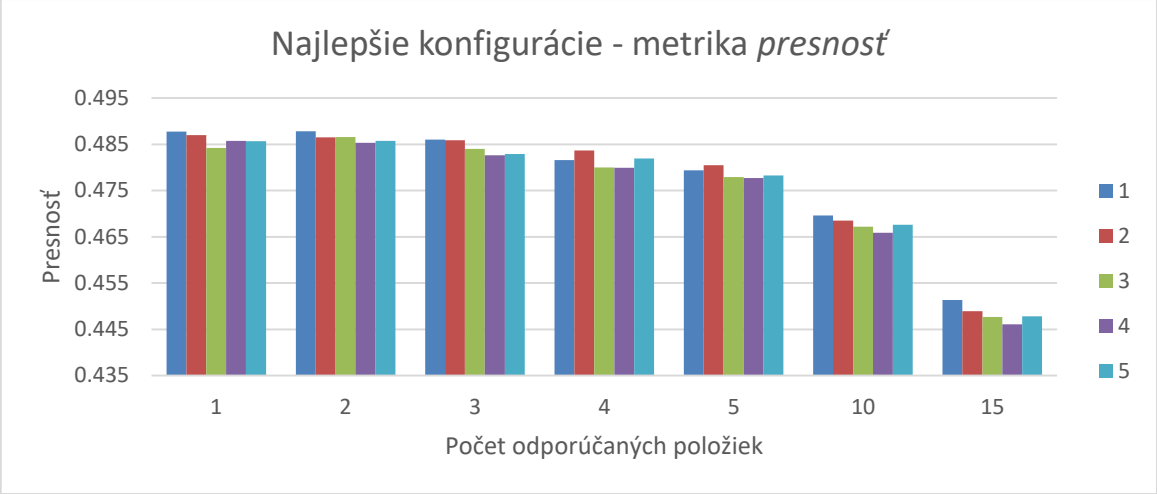
V

Tabuľka 13 sú uvedené hodnoty skúmaných parametrov v týchto konfiguráciach aj s dočasným označením konfigurácií. Všetky najlepšie konfigurácie majú hodnotu parametra *tcu=5* a hodnotu parametra *tcm=100.* Čo sa týka počtu skupín sú hodnoty rôzne ale neobsahujú hodnotu 2. Podobne pre parameter *tcm* neobsahujú hodnoty nižšie ako 200.

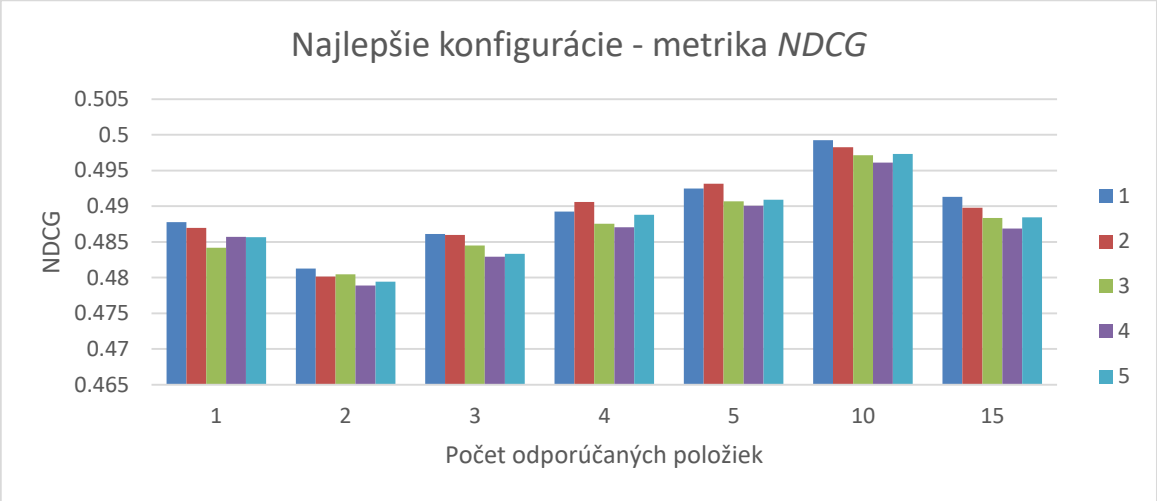
Na Graf 10 a Graf 11 možno vidieť hodnoty metriky *presnosť,* resp. *ndcg* pre rôzne počty odporúčaných položiek. Nie je vidno nejaký jasný vzor medzi vybranými konfiguráciami. Rozdiely sú pomerne malé v metrike *presnosť* aj *ndcg.* Na prvých dvoch miestach v oboch metrikách sa striedajú konfigurácie 1 a 2.

Tabuľka 13 Najlepšie konfigurácie vybrané zo skupiny parametrov zhlukovania.

Konfigurácia (dočasné ozn.)	gc	tuc	tcm	mmc
1	6	5	400	100
2	8	5	400	100
3	8	5	800	100
4	4	5	400	100
5	4	5	200	100



Graf 10 Metrika presnosť pre 5 najlepších konfigurácií zo skupiny parametrov zhlukovania.



Graf 11 Metrika NDCG pre 5 najlepších konfigurácií zo skupiny parametrov zhlukovania.

Ďalej sme sledovali vplyv hodnôt jednotlivých parametrov a dvojíc parametrov na výsledky odporúčania. Zistili sme, že konfigurácie s počtom skupín 2 sú mierne horšie ako konfigurácie s vyšším počtom skupín, kde už väčšie rozdiely nebadáme. Je to spôsobené zrejme tým, že počet skupín 2 je príliš malý aby sa priblížil skutočnému rozdeleniu používateľov do skupín a pri väčšom počte skupín sú identifikované aj niektoré ďalšie dôležité skupiny.Takisto sme zistili, že najlepšie výsledky sú v konfiguráciach s najmenšou hodnotou parametra *tuc*. To znamená, že častejšie dochádza k aktualizácií mikrozhlukov a tým pádom aj k častejšiemu makrozhlukovaniu a tiež, že používateľ je posudzovaný (teda zaradzovaný do skupín) podľa jeho čo najaktuálnejšieho správania (posledných 5 sedení). Čo sa týka parametra *tcm* tak príliš nízke hodnoty (50 a 100) a príliš vysoká hodnota (800) boli horšie ako hodnoty 200 a 400 (pozri tabuľku Tabuľka 15). Pre parameter *mmc* je situácia celkom jasná, zistili sme, že vyšší počet mikrozhlukov neviedol k lepším výsledkom. Jednotlivé parametre sa navzájom ovplyvňujú. V tabuľke Tabuľka 14 vidíme, že najlepšia kombinácia hodnôt parametrov *gc* (počet skupín) a *tcm* (minimálny počet aktualizácií mikrozhlukov) je pri počte skupín 4 a *tcm = 200.*

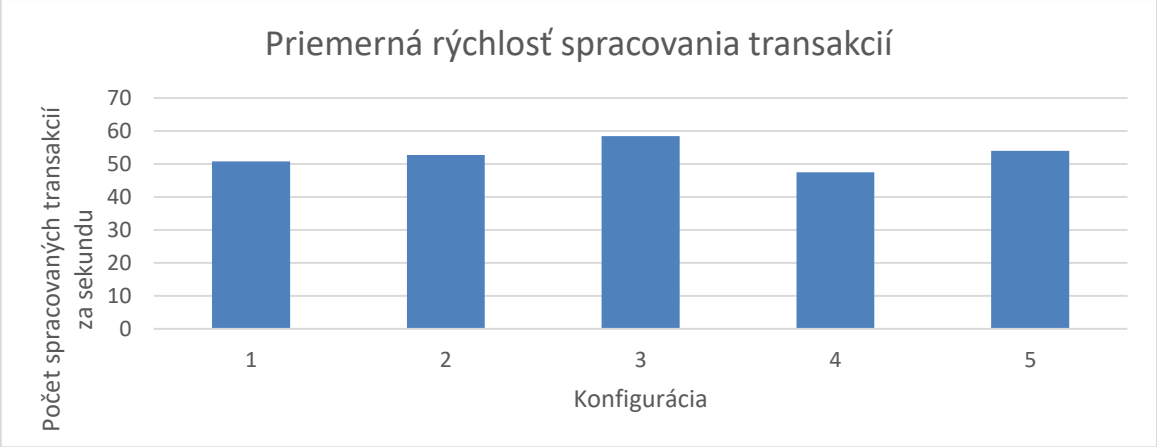
Tabuľka 14 Priemerné výsledky metriky presnosť pre rôzne kombinácie hodnôt parametrov gc a tcm.

gc vs. tcm	2	4	6	8
50	46.57%	46.66%	46.62%	46.62%
100	46.65%	46.86%	46.84%	46.82%
200	46.77%	47.02%	46.91%	46.91%
400	46.66%	46.77%	46.85%	46.84%
800	46.26%	46.42%	46.40%	46.41%

Tabuľka 15 Priemerné výsledky metriky presnosť pre rôzne kombinácie hodnôt parametrov tcm a tuc.

tcm vs. tuc	50	100	200	400	800
5	47.01%	47.12%	47.40%	47.41%	47.07%
10	46.57%	46.80%	46.81%	46.78%	46.29%
15	46.26%	46.46%	46.50%	46.16%	45.76%

Na Graf 12 vidíme priemernú rýchlosť spracovania transakcií jednotlivými konfiguráciami. Čo sa týka rozdielov v rýchlostiach, tie už nie sú také značné ako pri rôznych konfiguráciach zo skupiny parametrov pre hľadanie frekventovaných množín, ale najlepšiu rýchlosť (58.41 transakcií za sekundu) dosiahla konfigurácia 3. Do záverečného vyhodnocovania sme teda vybrali hodnoty parametrov práve z konfigurácií 1 a 2 a kvôli vyššej rýchlosti aj konfigurácie 3.



Graf 12 Priemerná rýchlosť spracovania transakcií pre najlepšie konfigurácie zo skupiny parametrov zhlukovania.

8.18 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov odporúčania

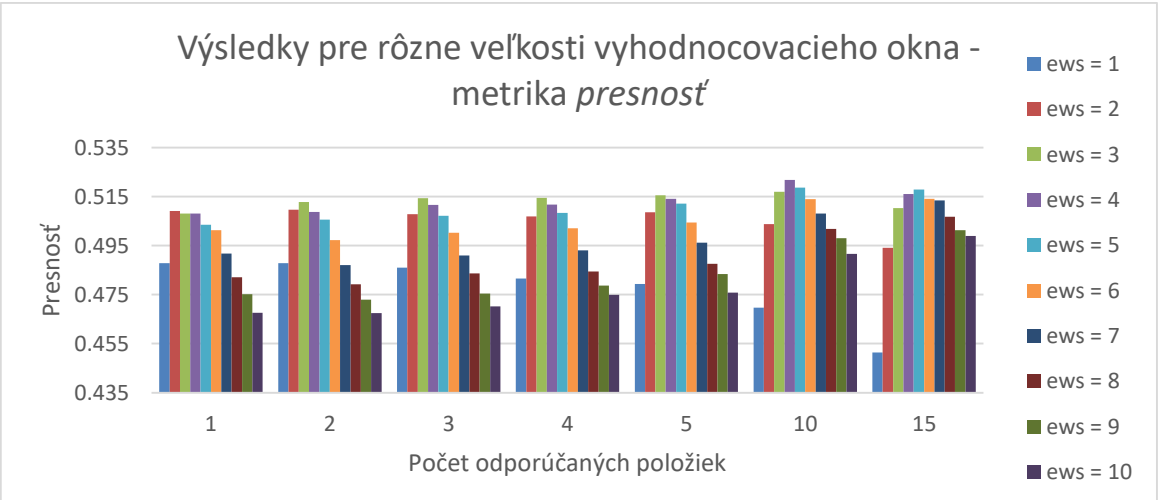
V tejto časti sa budeme venovať vyhodnoteniu odporúčania v spojitosti s rôznymi konfiguráciami skupiny parametrov pre odporúčanie. Táto skupina je zvláštna, keďže pozorujeme len jeden parameter hoci tu patrí aj parameter *rc* teda počet odporúčaných položiek, ktorého rôzne hodnoty ale sledujeme implicitne pri každom behu aj v ostatných skupinách parametrov. Ide teda o jediný parameter:

- **Veľkosť okna vyhodnocovania** *(skr. ews)*: {1,2,3,4,5,6,7, 8, 9, 10}

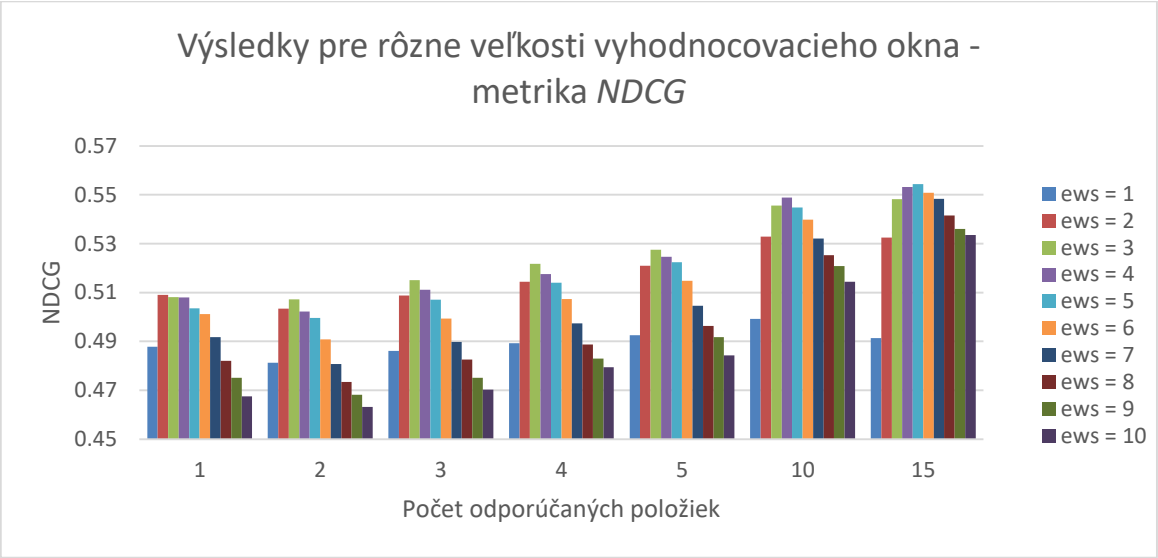
Nastavenie hodnôt ostatných parametrov sme zobrali z najlepšej konfigurácie čo sa týka presnosti vybranej z predchádzajúcej skupiny parametrov zhlukovania (konfigurácia 1 v tabuľke Tabuľka 15).

Zmena hodnoty parametra *ews* mení aj množinu sedení, ktoré sú vyhodnocované. Čím dlhšie je vyhodnocované okno tým menej sedení má dostatočnú dĺžku aby mohli byť vyhodnotené. To je závažný fakt, ktorý má vplyv na relevantnosť porovnávania jednotlivých konfigurácií.

Na Graf 13 a Graf 14 a možno vidieť hodnoty metriky *presnosť,* resp. *ndcg* pre rôzne počty odporúčaných položiek a rôzne veľkosti vyhodnocovacieho okna. Môžeme si zvlášť všimnúť, že hodnoty parametra *ews* < *2 a ews* > *7* majú horšie výsledky najmä pri malom počte odporúčaných položiek. Malé okno dáva priestor možnosti výberu veľkého množstva vzorov správania (aj kratších aj dlhších), ktoré majú prienik s týmto oknom a môžu byť zapojené do odporúčania. To je však zrejme aj čiastočne obmedzením, pretože okno je príliš malé aby rozpoznalo charakter sedenia a zámer používateľa. Pri väčších oknách sa *presnosť* opäť znižuje, čo je zrejme spôsobené aj tým, že neexistuje dostatok dlhších vzorov, ktoré by vedeli dané správanie používateľa správne charakterizovať.
Najlepšie výsledky *presnosti* sú pri hodnotách *ews* > *1 a zároveň ews* < *7* pri počte odporúčaných položiek < 10. Pri väčšom počte odporúčaných položiek sa javia najlepšie veľkosti *ews* > *2* a zároveň *ews* < *8*.



Graf 13 Výsledky metriky *presnosť* pre rôzne veľkosti vyhodnocovacieho okna *ews* pre rôzne počty odporúčaných položiek.



Graf 14 Výsledky metriky NDCG pre rôzne veľkosti vyhodnocovacieho okna ews pre rôzne počty odporúčaných položiek.

Tento parameter je teda špecifický a do výsledného hľadania najlepšej konfigurácie (časť 7.2.5) sme vybrali hodnoty *ews* z množiny {2, 3, 4, 5}, pri ktorých nebola množina vyhodnocovaných sedení priveľmi zmenšená a zároveň boli dosiahnuté najlepšie výsledky presnosti.

8.19 Vyhodnotenie úspešnosti odporúčania pre ostatné parametre metódy

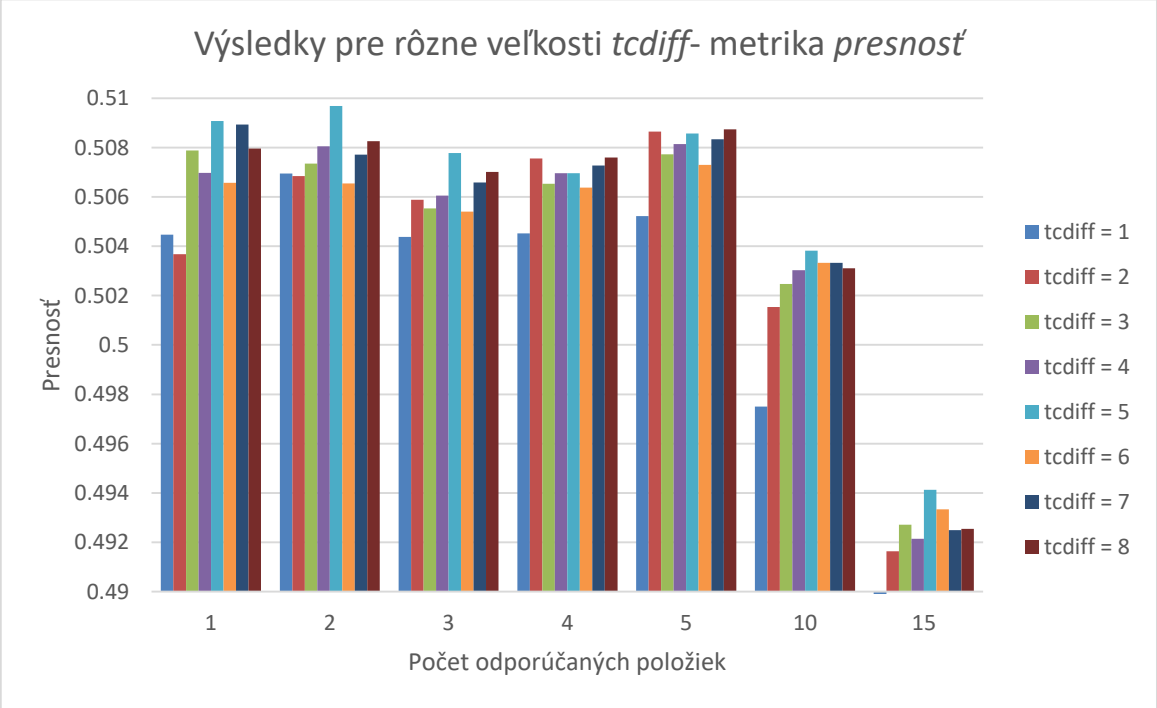
V tejto časti sa budeme venovať vyhodnoteniu odporúčania v spojitosti s rôznymi konfiguráciami skupiny osatných parametrov metódy. Ide o parametre (bližšie sú opísané v časti 0) a ich hodnoty:

- **Hraničná hodnota rozdielu identifikátorov zhlukovania** *(skr. tcdiff)*: {1,2,3,4, 5,6,7,8}
- **Minimálna rýchlosť** *(skr. mts)*: {15}

V tejto skupine sme sa rozhodli testovať rôzne hodnoty len prvého parametra *tcdiff*. Rôzne hodnoty parametra pre obmedzenie minimálnej rýchlosti budeme testovať až vo vyhodnotení rýchlosti spracovania (7.3.2), kde sledujeme k akému zhoršeniu metriky *presnosť* bude dochádzať pri zrýchľovaní spracovania transakcií na už vybranej celkovo najlepšej konfigurácií. Nastavenie hodnôt ostatných parametrov sme zobrali z konfigurácie vybranej z predchádzajúcej skupiny parametrov odporúčania, kde sme pozorovali len jeden parameter veľkosti vyhodnocovaného okna. Vybrali sme nastavenie vyhodnocovacieho okna s veľkosťou 2, ktorá príliš nezmenšuje množinu vyhodnocovaných sedení a zároveň dosahuje jedny z najlepších výsledkov metriky *presnosť*. Celkovo k makrozhlukovaniu dochádza 9 krát preto sme testovali rôzne hodnoty *tcdiff* po túto hodnotu.

Sledovali sme, že skúmané hodnoty tohto parametra nijak zvlášť neovplyvňovali celkové výsledky (pozri Graf 15). To je dobré znamenie z toho pohľadu, že odstraňovanie neaktívnych používateľov z pamäte v skúmanej doméne už po krátkej dobe nečinnosti zdá sa nejak zásadne neovplyvňuje celkové výsledky.

Do záverečného vyhodnotenia sme vybrali hodnotu parametra *tcdiff* = 5, pri ktorej boli výsledky o niečo málo lepšie než pri ostatných.



Graf 15 Výsledky metriky presnosť pre rôzne hodnoty parametra tcdiff a rôzne počty odporúčaných položiek.

8.20 Celkové vyhodnotenie úspešnosti odporúčania pre najlepšie konfigurácie

Na základe vybraných najlepších konfigurácií zo všetkých skupín parametrov ako sme ich uvádzali v predchádzajúcich častiach sme skonštruovali priestor prehľadávania v ktorom budeme hľadať celkovo najlepšiu konfiguráciu. Ide o parametre a ich hodnoty:

- **Minimálna podpora** *(skr. ms)*: {0.04, 0.05}
- **Miera uvoľnenia** *(skr. rr)*: {0.1, 0.5}
- **Dĺžka segmentu** *(skr. sl)*: {25, 50}
- **Maximálna dĺžka množiny** *(skr. mil)*: {10}
- **Veľkosť okna** *(ws)*: {10,15}
- **Počet zhlukov** *(skr. gc)*: {6, 8}
- **Hraničný počet zmien v modeli používateľa** *(skr. tuc)*: {5}
- **Hraničný počet zmien v mikrozhlukoch** *(skr. tcm)*: {400, 800}
- **Maximálny počet mikrozhlukov***(skr. mmc)***:** {100}
- **Veľkosť okna vyhodnocovania** *(skr. ews)*: {2,3,4,5}
- **Hraničná hodnota rozdielu identifikátorov zhlukovania** *(skr. tcdiff)*: {5}
- **Minimálna rýchlosť** *(skr. mts)*: {15}

V Tabuľka 5 uvádzame 7 konfigurácií, ktoré boli najlepšie pre niektorú z rôznych kombinácií hodnôt parametrov *ews* (veľkosť vyhodnocovacieho okna) a *rc* (počet odporúčaných položiek). Jediná z uvedených hodnôt ktorá sa medzi najlepšími konfiguráciami nenachádza je hodnota miery uvoľnenia *rr = 0.5*.

Tabuľka 16 Tabuľka zobrazuje hodnoty parametrov pre 7 najlepších konfigurácií celkovo pre dataset ALEF.

ID konfigurácie	ms	rr	sl	ws	gc	tuc	tcm	mmc	tcdiff	mts
1	0.05	0.1	25	15	6	5	400	100	5	15
2	0.04	0.1	50	15	8	5	800	100	5	15
3	0.05	0.1	50	15	8	5	400	100	5	15
4	0.04	0.1	50	10	6	5	400	100	5	15
5	0.05	0.1	50	15	6	5	800	100	5	15
6	0.04	0.1	50	10	8	5	800	100	5	15
7	0.05	0.1	50	15	6	5	400	100	5	15

Výsledky sme sledovali samostatne pre rôzne hodnoty parametra *ews,* ktorý, ako sme už spomínali, mení veľkosť priestoru vyhodnocovania. dNa grafoch Graf 16 a Graf 17 uvádzame výsledky metriky *presnosť* a *ndcg* pre rôzne hodnoty parametra *ews*. Na grafe Graf 18 vidíme porovnanie priemerných rýchlostí jednotlivých konfigurácií. Môžeme si všimnúť konfiguráciu 1, ktorá vyniká naprieč rôznymi veľkosťami vyhodnocovacieho okna pri menšom počte odporúčaných položiek (<= 3). Pri väčšom počte odporúčaných položiek sa najlepšie konfigurácie striedajú. Konfigurácia 1 je výnimočná nielen kvôli svojím výsledkom čo sa týka metrík *presnosť* a *ndcg* ale aj čo sa týka rýchlosti. Všetky

ostatné konfigurácie padli až na spodnú hranicu minimálnej požadovanej rýchlosti okrem konfigurácie 1. Z tohto dôvodu sme sa rozhodli konfiguráciu 1 zobrať ako celkovo najlepšiu konfiguráciu hodnôt parametrov pre dataset ALEF a pokračujeme s ňou v ďalších experimentoch.

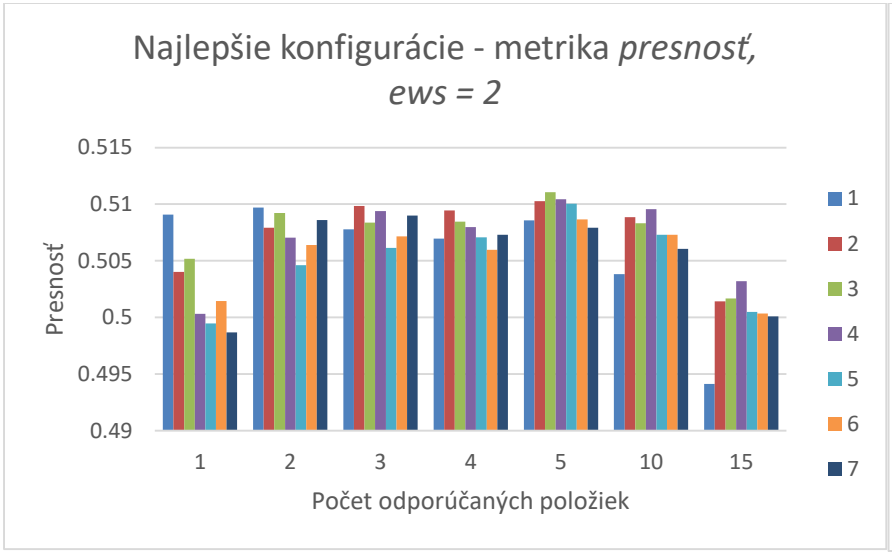
Na záver sme ešte vyhodnotili priemerný rozdiel medzi výsledkami metódy využívajúcej len globálne vzory a metódy využívajúcej kombináciu skupinových a globálnych vzorov. Ten sa pohybuje okolo 1.7% pre rôzne počty odporúčaných položiek a výnimkou nie je ani vybraná konfigurácia 1 kde sa tento rozdiel pohybuje okolo 1.6% (pozri tabuľku Tabuľka 6 a tabuľku Tabuľka 7). Tieto výsledky potvrdené jednoduchým štatistickým t-testom súhlasia s hypotézou, ktorú sme uviedli na začiatku kapitoly. Vykonali sme jednoduchý t-test medzi populáciami hodnôt výsledkov metriky *presnosť* všetkých vyhodnocovaných konfigurácií metódy využívajúcej kombináciu skupinových a globálnych vzorov a metódy využívajúcej len globálne vzory. Dosiahnutý výsledok je štandardne vyhodnotený ako vysoko štatisticky signifikantný ($p < 0.0001$, $t=9.8366$, $df=7740$ a rozdiel stredných hodnôt je 0.0165).

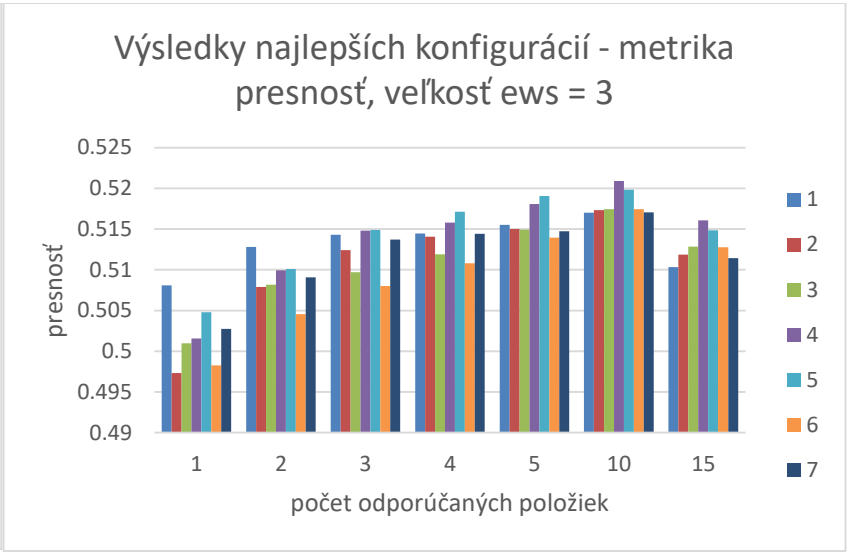
Tabuľka 17 Tabuľka zachytáva rozdiel medzi priemernými hodnotami metriky presnosť pre dve metódy odporúčania získané zo všetkých vyhodnocovaných konfigurácií a pre rôzne počty odporúčaných položiek v stĺpcoch.

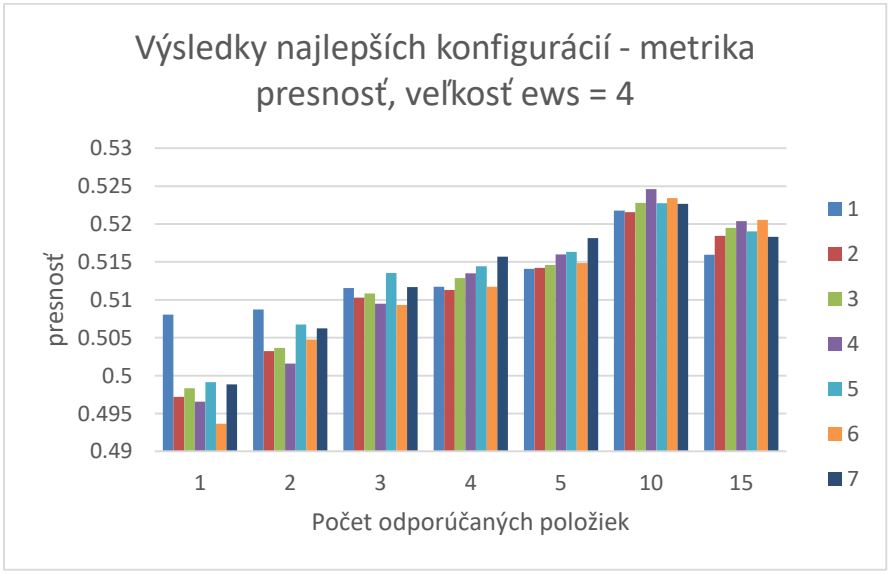
	P@1	P@2	P@3	P@4	P@5	P@10	P@15
Kombinácia skupinových a globálnych vzorov	44.15%	44.66%	44.71%	44.63%	44.53%	43.79%	42.59%
Iba globálne vzory	42.47%	43.03%	43.09%	43.02%	42.91%	42.13%	40.87%
Rozdiel	1.68%	1.63%	1.62%	1.61%	1.61%	1.66%	1.72%

Tabuľka 18 Tabuľka zachytáva rozdiel medzi priemernými hodnotami metriky presnosť pre dve metódy odporúčania získané z najlepšej konfigurácie a pre rôzne počty odporúčaných položiek v stĺpcoch. Hodnoty v bunkách sú spriemerované pre rôzne hodnoty parametra ews z {2,3,4};

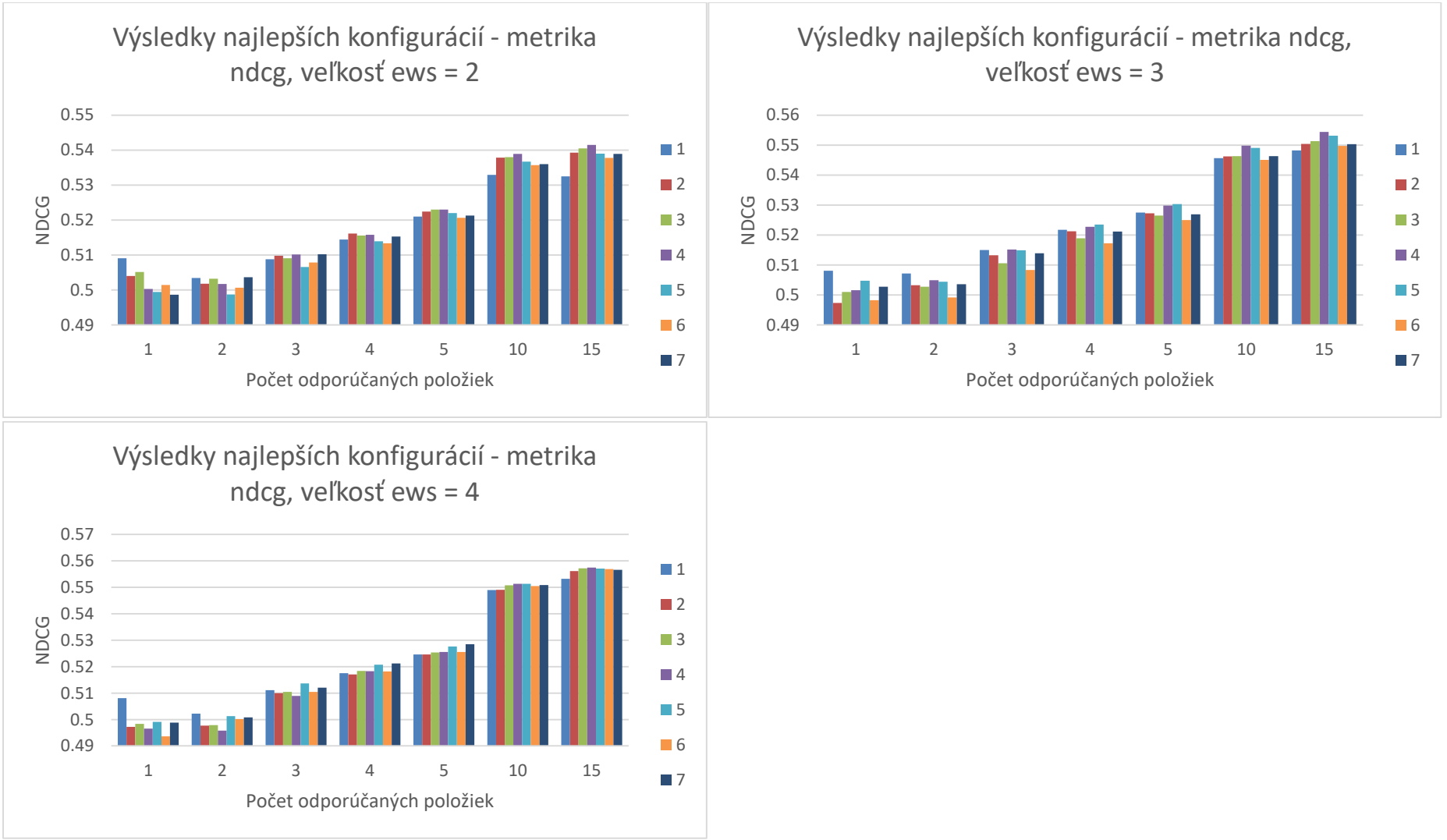
	P@1	P@2	P@3	P@4	P@5	P@10	P@15
Kombinácia skupinových a globálnych vzorov	50.73%	50.98%	51.09%	51.12%	51.34%	51.56%	50.93%
Iba globálne vzory	49.07%	49.51%	49.46%	49.52%	49.73%	49.73%	49.00%
Rozdiel	1.66%	1.47%	1.64%	1.59%	1.61%	1.83%	1.92%



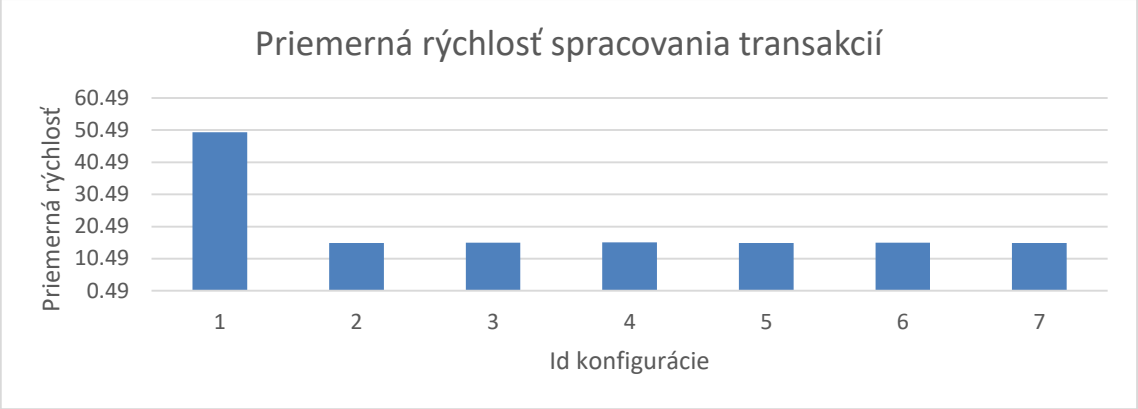




Graf 16 Grafy znázorňujúce výsledky najlepších 7 konfigurácií v metrike presnosť a pri rôznych hodnotách parametra ews.



Graf 17 Grafy znázorňujúce výsledky najlepších 7 konfigurácií v metrike NDCG a pri rôznych hodnotách parametra ews.



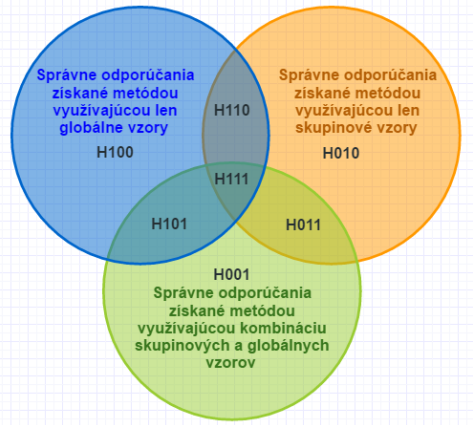
Graf 18 Graf znázorňuje priemerné rýchlosti jednotlivých konfigurácií v jednotkách počet transakcií za sekundu.

PRÍLOHA F – Dodatočné materiály k výslednému vyhodnoteniu pre dataset ALEF

V tejto časti uvádzame niektoré podrobnejšie grafy a tabuľky z celkového vyhodnotenia pre dataset ALEF.

8.21 Vyhodnotenie vzťahov medzi globálnymi a skupinovými vzormi [dodatočné materiály]

Uvádzame Tabuľka 19, ktorá sumarizuje zaujímavé fakty získané z rôznych prienikov množín vektorov úspešnosti odporúčania (pre odporúčania získané metódy využívajúcou len globálne vzory, len skupinové vzory a kombináciu skupinových vzorov s globálnymi) a pre najlepšiu nájdenú konfiguráciu pre dataset ALEF. Množiny sú označené písmenom *H* nasledovaným troma bitmi, ktoré označujú použitie globálnych, skupinových vzorov a ich kombinácie. Rôzne prieniky sú pre pripomenutie znázornené na Vennovom diagrame (Graf 1).



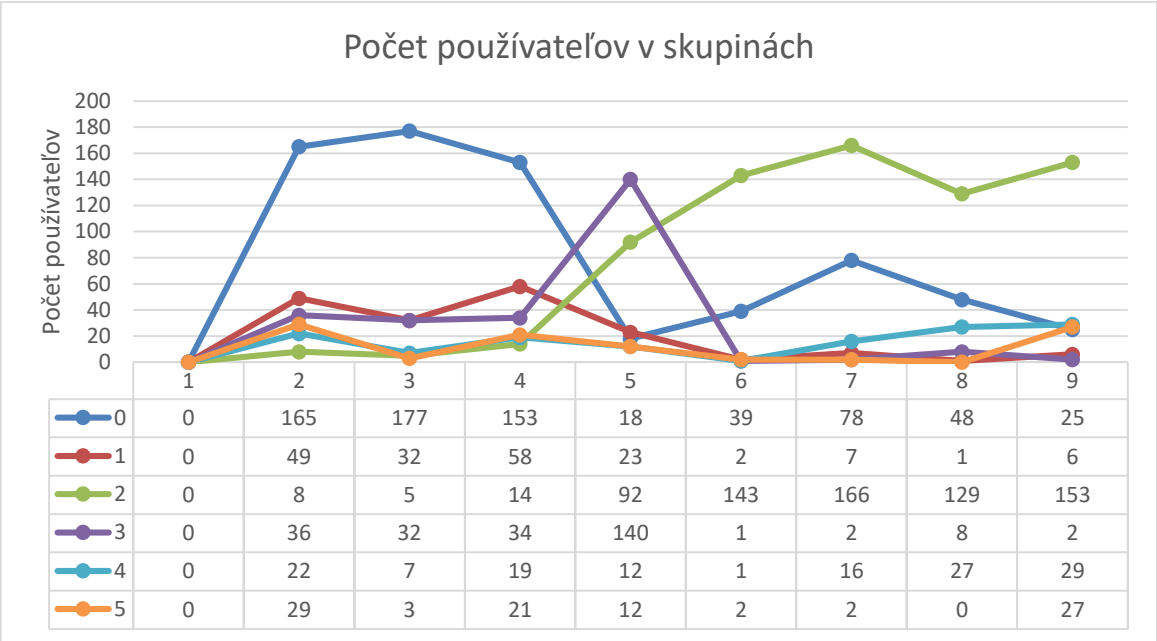
Graf 19 Vennov diagram znázorňuje označenia jednotlivých množín vektorov úspešnosti odporúčania a ich prienikov.

Na grafe 2 uvádzame distribúciu počtu používateľov v jednotlivých skupinách počas spracovania prúdu dát. Na grafe 3 uvádzame počty unikátnych vzorov v jednotlivých skupinách počas spracovania prúdu dát.

Tiež uvádzame príklad nájdených top vzorov v jednotlivých skupinách v niektorých snímkach v Tabuľka 20. Ilustrujeme tak, že dolovanie skupinových vzorov v prúde dát ponúka zaujímavý pohľad na správanie a zmeny správania používateľov v skupinách.

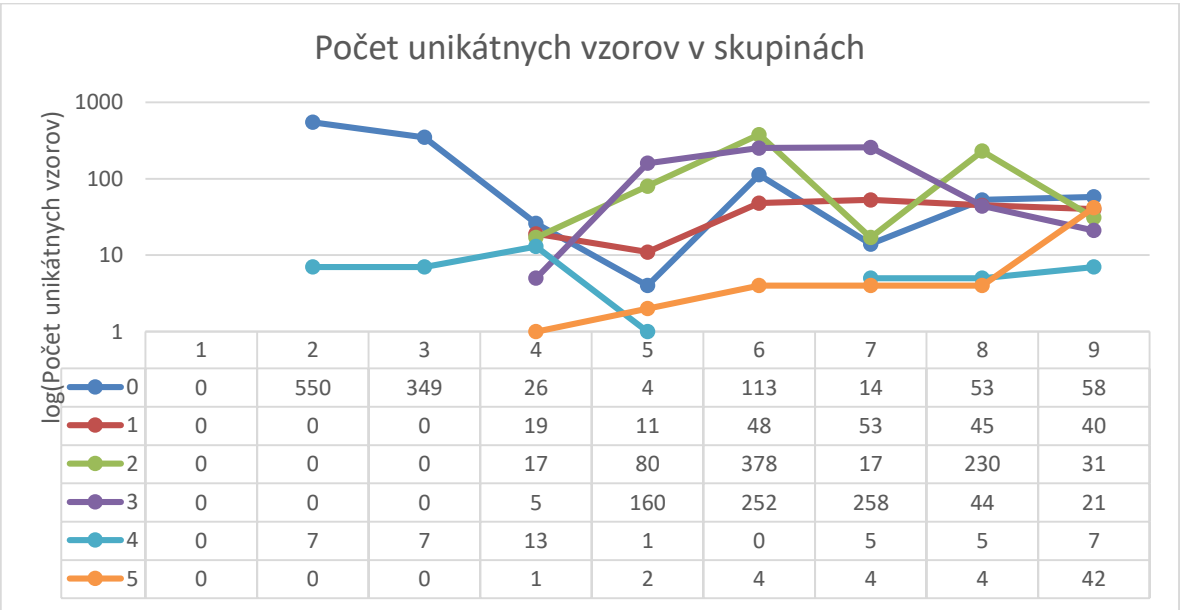
V ostatných grafoch na konci prílohy uvádzame priebehy vývoja pre metriku *presnosť* pre 2 odporúčané položky v jednotlivých skupinách počas spracovania prúdu dát. Nech GGC je označenie pre metódu využívajúcu globálne aj skupinové vzory, GO označenie pre metódu využívajúcu iba globálne vzory a OG metódu využívajúcu iba skupinové vzory.

Pre všetky uvedené grafy a tabuľky platí, že merania sú uskutočňované v časových snímkach. Časové snímky predstavujú určitý bod v čase počas spracovania prúdu dát, ktorý sa pravidelne opakuje. Konkrétne je to tesne pred ďalším makrozhlukovaním (vtedy by už malo byť dosť používateľov zaradených do skupín a mali by byť identifikované dostatočne silné aktuálne vzory).



	1	2	3	4	5	6	7	8	9
0	0	165	177	153	18	39	78	48	25
1	0	49	32	58	23	2	7	1	6
2	0	8	5	14	92	143	166	129	153
3	0	36	32	34	140	1	2	8	2
4	0	22	7	19	12	1	16	27	29
5	0	29	3	21	12	2	2	0	27

Graf 20 Distribúcia počtu používateľov v skupinách v jednotlivých časových snímkach. Na osi x sú uvedené časové snímky.Snímka predstavuje moment v behu programu tesne pred makrozhlukovaním keď sa zozbierali uvedené dátata.



	1	2	3	4	5	6	7	8	9
0	0	550	349	26	4	113	14	53	58
1	0	0	0	19	11	48	53	45	40
2	0	0	0	17	80	378	17	230	31
3	0	0	0	5	160	252	258	44	21
4	0	7	7	13	1	0	5	5	7
5	0	0	0	1	2	4	4	4	42

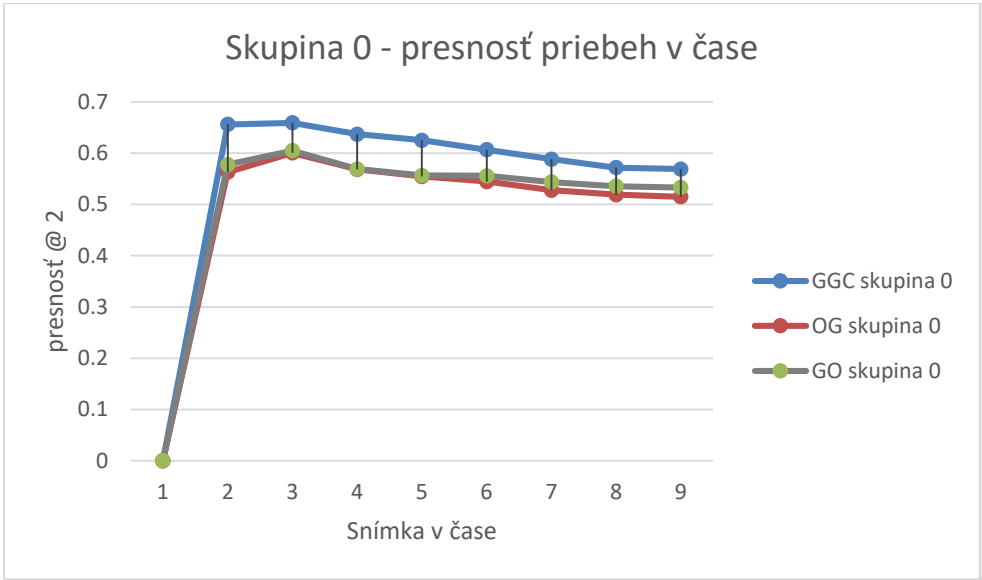
Graf 21 Počty unikátnych vzorov v skupinách zaznamenané v jednotlivých časových snímkach na logaritmickej škále (kvôli lepšej prehľadnosti). Na osi x sú uvedené časové snímky.Snímka predstavuje moment v behu programu tesne pred makrozhlukovaním keď sa zozbierali uvedené dáta.

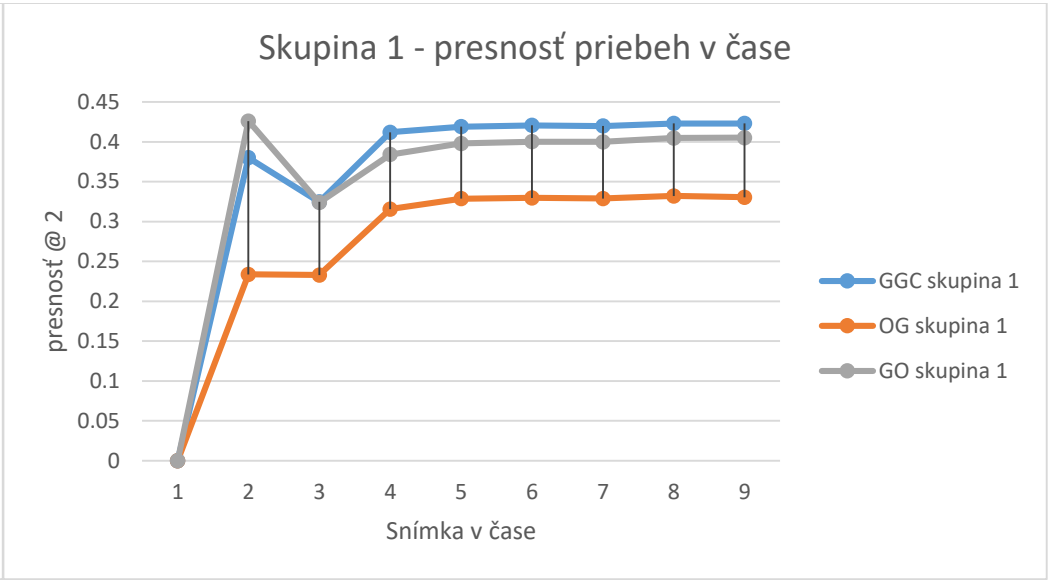
Tabuľka 19 Suma rôznych prienikov vektorov správnosti odporúčania získaných pomocou globálnych skupinových aj kombinovaných vzorov.

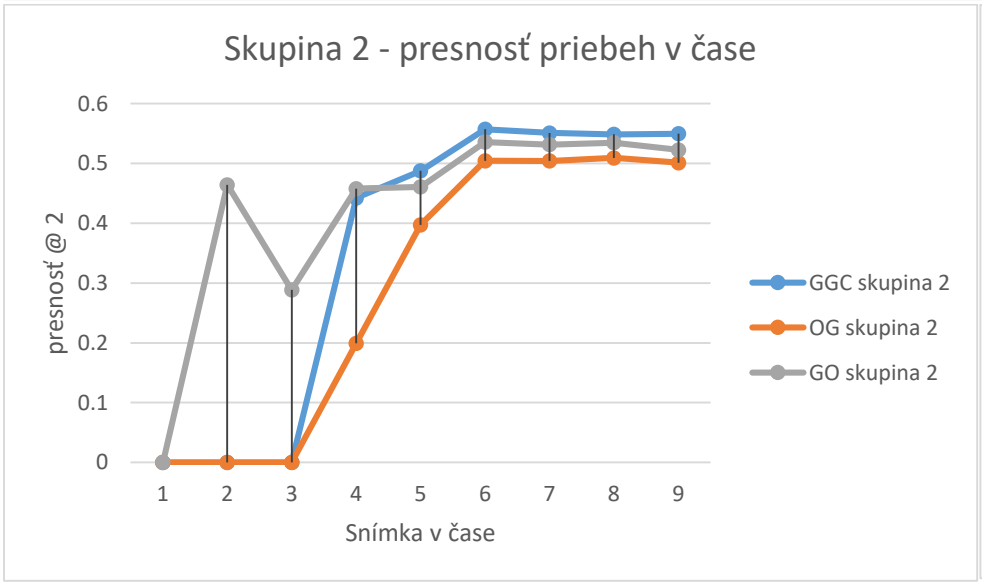
ID	*POPIS*	*Spôsob výpočtu*	P@1	P@2	P@3	P@4	P@5	P@10	@15
1	Počet správnych odporúčaní len pomocou skupinových vzorov a nie globálnych:	**H011 + H010**	**5.86%**	**7.17%**	**7.97%**	**8.33%**	**8.82%**	**9.96%**	**10.27%**
2	Počet správnych odporúčaní len pomocou globálnych vzorov a nie skupinových:	**H101+H100**	**22.72%**	**24.09%**	**25.19%**	**25.82%**	**26.31%**	**26.98%**	**27.32%**
3	Počet správnych odporúčaní len pomocou kombinovaných vzorov a nie iných vzorov:	**H001**	**0.32%**	**0.84%**	**1.19%**	**1.54%**	**1.84%**	**2.85%**	**3.49%**
4	Počet správnych odporúčaní len pomocou skupinových vzorov, ktoré kombinovaná metóda nevedela odporučiť	**H010**	**2.63%**	**3.41%**	**3.95%**	**4.16%**	**4.40%**	**5.02%**	**5.08%**
5	Počet správnych odporúčaní pomocou globálnych vzorov, ktoré kombinovaná metóda nevedela odporučiť	**H100**	**2.02%**	**2.60%**	**3.00%**	**3.27%**	**3.47%**	**4.05%**	**4.34%**
6	Počet správnych odporúčaní získaných pomocou prieniku odporúčaní z globálnych a skupinových vzorov.	**H111+H110**	**26.55%**	**25.77%**	**25.24%**	**24.91%**	**24.74%**	**24.12%**	**23.13%**
7	Teoreticky najvyšší možný počet správnych odporúčaní, ktoré by mohla kombinovaná metóda dosiahnuť.	**H111+H110+H101+ H100+H011+H010**	**55.14%**	**57.02%**	**58.40%**	**59.06%**	**59.88%**	**63.91%**	**60.72%**
8	Počet správnych odporúčaní pomocou našej metódy	**H101+H011+H001+ H111**	**50.71%**	**51.26%**	**51.73%**	**51.96%**	**52.41%**	**52.62%**	**52.13%**
9	Počet správnych odporúčaní pomocou globálnych vzorov	**H111+H110+H101+ H100**	**49.27%**	**49.85%**	**50.44%**	**50.73%**	**51.06%**	**51.10%**	**50.45%**
10	Počet správnych odporúčaní pomocou skupinových vzorov	**H111+H011+H110+ H010**	**32.41%**	**32.93%**	**33.21%**	**33.24%**	**33.57%**	**34.09%**	**33.40%**

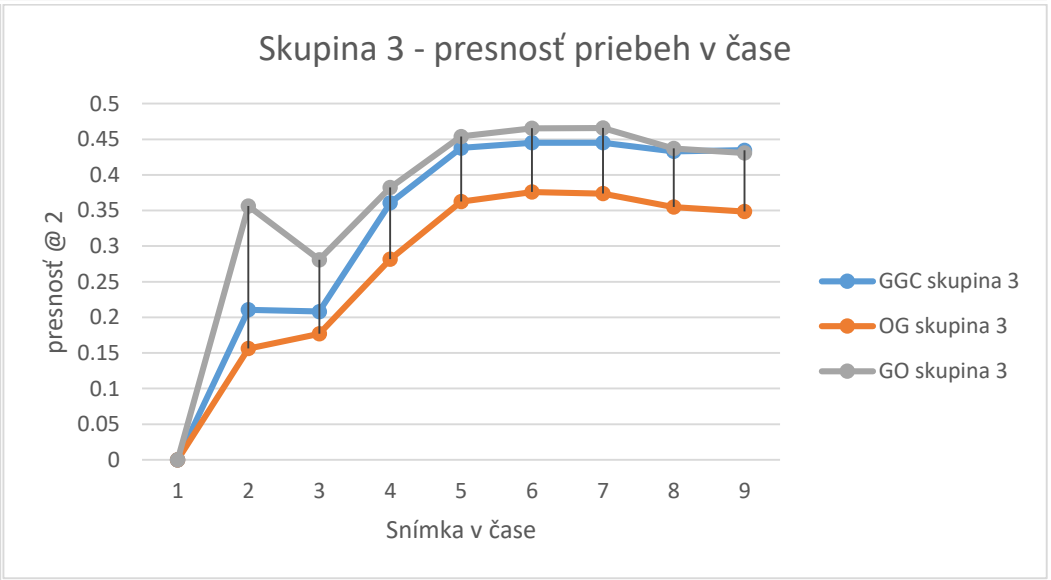
Tabuľka 20 Ukážky vybraných najsilnejších globálnych a skupinových vzorov vo vybraných snímkach. -1 označuje globálne vzory.

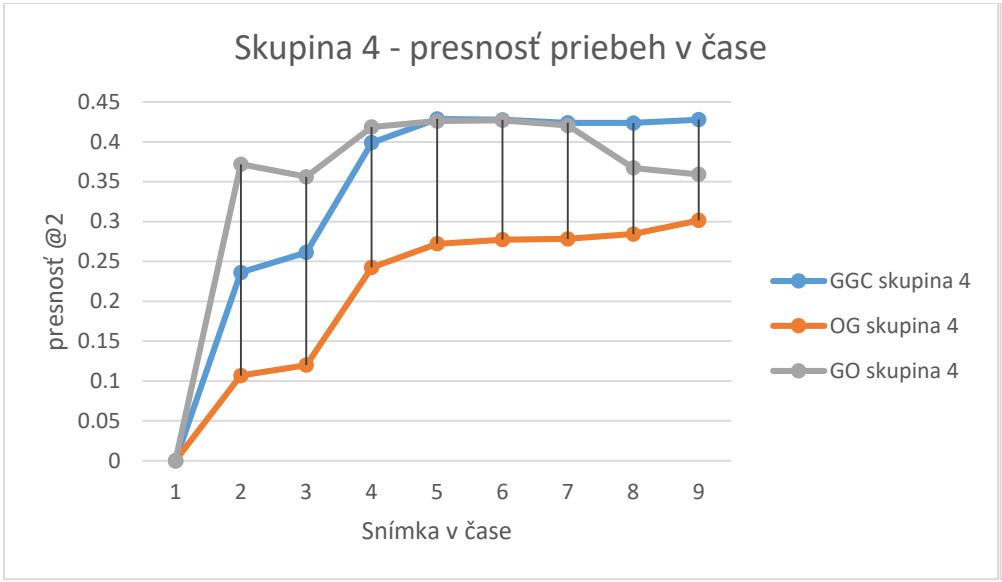
		ID SNÍMKY		
		7 (2014-03-01 14:25:11)	**8(2014-04-03 14:09:06)**	**9 (2014-06-17 15:10:58)**
SKUPINA	**-1**	[1.3 Softvér],[1.1 Čo je to softvérové inžinierstvo?]	[5.1.4 Diagram modelu údajov], [5.1.3 Entitno-relačný diagram]	[FLP:Predikát op], [Operátory]
		[1.4 Dôležité skutočnosti k vývoju softvéru],[1.1 Čo je to softvérové inžinierstvo?]	[5 Štruktúrne modely], [4 Funkcionálne modely]	[3.6 UML diagramy], [3 Modelovanie softvéru]
	0	[5 Štruktúrne modely],[6 Modely správania]	[4 Funkcionálne modely],[4.2.2 Notácia pre model prípadov použitia]	[4.2 Model prípadov použitia],[4.2.1 Použitie modelu prípadov použitia]
		[PSI - question 4], [PSI - question 144]	[4 Funkcionálne modely],[4.2.1 Použitie modelu prípadov použitia]	[5 Štruktúrne modely],[4 Funkcionalne modely]
	1	[5.1 Model údajov],[5.1.2 Základné elementy modelu údajov],[5.1.3 Entitno-relačný diagram]	[1.4.2 Dôsledky a príčiny],[1.4.1 Zákonité a nezákonité problémy]	[1.4.2 Dôsledky a príčiny],[1.4.1 Zákonité a nezákonité problémy]
		[5.1 Model údajov],[5.1.1 Typy modelov údajov]	[1.4.1 Zákonité a nezákonité problémy]	[1.4.1 Zákonité a nezákonité problémy]
	2	[1.2 Softvérové inžinierstvo - dôležité pojmy]	[4.2.1 Použitie modelu prípadov použitia],[4.2.2 Notácia pre model prípadov použitia]	[3.1 Model softvérového systému]
		[FLP:Príklad SUCIN], [FLP:Základné prvky jazyka lisp]	[5 Štruktúrne modely], [4 Funkcionálne modely]	[3.3 Rôzne pohľady - rôzne modely softvéru]
	3	[5 Štruktúrne modely],[6 Modely správania]	[FLP:Funkcionál REDUCE],[FLP:Funkcionál MAPCAR],[FLP:Funkcionál FUNCALL]	[FLP:Funkcionály FIND-IF a COUNT-IF],[FLP:Funkcionál MAPCAR],[FLP:Funkcionál FUNCALL]
		[4 Funkcionálne modely],[6 Modely správania]	[FLP:Funkcionál REDUCE],[FLP:Funkcionál MAPCAR],[FLP:Forma FUNCTION]	[FLP:Funkcionál REDUCE],[FLP:Funkcionál MAPCAR],[FLP:Funkcionál FUNCALL]
	4	[1.3 Softvér],[1.1 Čo je to softvérové inžinierstvo?]	[FLP:Funkcionál REDUCE],[FLP:Funkcionál MAPCAR],[FLP:Forma FUNCTION],[Funkcionál FUNCALL],[Funkcionály REMOVE-IF a REMOVE-IF-NOT]	[FLP:Počítanie],[FLP:Redukcia],[FLP:Zobrazenie]
		[1.4 Dôležité skutočnosti k vývoju softvéru],[1.1 Čo je to softvérové inžinierstvo?]	[FLP:Funkcionál REDUCE],[FLP:Funkcionál MAPCAR],[FLP:Forma FUNCTION],[Programové schémy],[Funkcionál FUNCALL],[Funkcionály REMOVE-IF a REMOVE-IF-NOT]	[FLP:Počítanie],[FLP:Filter],[FLP:Zobrazenie]
	5	[3.1 Model softvérového systému],[3.2 Aspekty modelovania softvéru]	[3.1 Model softvérového systému],[3.2 Aspekty modelovania softvéru]	[5 Štruktúrne modely], [6 Modely správania]
		[5 Štruktúrne modely],[6.3.1 Použitie stavového diagramu],[PSI - question 7], [PSI - question 2], [PSI - question 44]	[5 Štruktúrne modely],[6.3.1 Použitie stavového diagramu],[PSI - question 7], [PSI - question 2], [PSI - question 44]	[5 Štruktúrne modely]

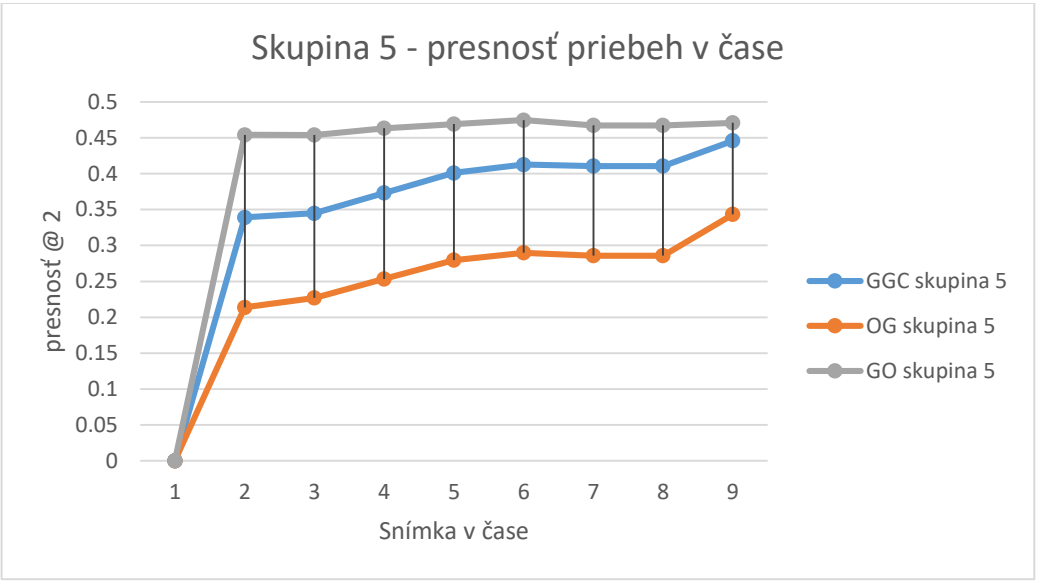

Skupina 0 - presnosť priebeh v čase
presnosť @ 2
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1 2 3 4 5 6 7 8 9
Snímka v čase
GGC skupina 0
OG skupina 0
GO skupina 0


Skupina 1 - presnosť priebeh v čase
presnosť @ 2
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1 2 3 4 5 6 7 8 9
Snímka v čase
GGC skupina 1
OG skupina 1
GO skupina 1


Skupina 2 - presnosť priebeh v čase
presnosť @ 2
0.6
0.5
0.4
0.3
0.2
0.1
0
1 2 3 4 5 6 7 8 9
Snímka v čase
GGC skupina 2
OG skupina 2
GO skupina 2


Skupina 3 - presnosť priebeh v čase
presnosť @ 2
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1 2 3 4 5 6 7 8 9
Snímka v čase
GGC skupina 3
OG skupina 3
GO skupina 3


Skupina 4 - presnosť priebeh v čase
presnosť @2
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
9
Snímka v čase
GGC skupina 4
OG skupina 4
GO skupina 4


Skupina 5 - presnosť priebeh v čase
presnosť @ 2
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
1
2
3
4
5
6
7
8
9
Snímka v čase
GGC skupina 5
OG skupina 5
GO skupina 5

PRÍLOHA G – Detailné vyhodnotenie úspešnosti odporúčania pre dataset SACBEE

V tejto časti uvádzame niektoré podrobnejšie grafy a tabuľky z celkového vyhodnotenia pre dataset SACBEE.

8.22 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov hľadania frekventovaných množín

V tejto časti sa budeme venovať vyhodnoteniu úspešnosti odporúčania s rôznymi konfiguráciami skupiny parametrov pre algoritmus hľadania frekventovaných množín. Ide o parametre (bližšie sú opísané v časti 7.1.46) a ich hodnoty:

- **Minimálna podpora (ms):** {0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1}
- **Miera uvoľnenia (rr):** {0.1, 0.5, 0.9}
- **Dĺžka segmentu (fsl):** {25, 50, 100, 150, 200, 500}
- **Maximálna dĺžka množiny (mil):** {10}
- **Veľkosť okna (ws)**: {5,10,15}

Spolu je to 270 rôznych konfigurácií. Čo sa týka hodnôt minimalnej podpory tá vyjadruje minimálnu podporu v rámci jedného segmentu. Teda ak dĺžka segmentu je napr. 50 a hodnota minimálnej podpory je 0.02 a menšia tak je každá množina považovaná za frekventovanú. Kombinácie parametrov kde bude teda hodnota $l * ms \leq 1$ budeme ignorovať.

8.23 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov zhlukovania

V tejto časti sa budeme venovať vyhodnoteniu úspešnosti odporúčania s rôznymi konfiguráciami skupiny parametrov pre algoritmus zhlukovania *CluStream*. Ide o parametre (bližšie sú opísané v časti 0) a ich hodnoty:

- **Počet zhlukov** *(skr. gc)*: {2, 4, 6, 8}
- **Hraničný počet zmien v modeli používateľa** *(skr. tuc)*: {5,10,15}
- **Hraničný počet zmien v mikrozhlukoch** *(skr. tcm)*: {50, 100, 200, 400, 800}
- **Maximálny počet mikrozhlukov***(skr. mmc)***:** {100,1000}

Spolu je to 120 rôznych konfigurácií. Nastavenie ostatných parametrov sme vybrali z jednej z najlepších konfigurácií z predchádzajúcej časti (konfigurácia 2 v **Error! Reference source not found.**), ktorá dosahovala najlepšie výsledky pre menšie počty odporúčaných položiek a zároveň bola veľmi rýchla.

8.24 Vyhodnotenie úspešnosti odporúčania pre skupinu parametrov odporúčania

V tejto časti sa budeme venovať vyhodnoteniu odporúčania v spojitosti s rôznymi konfiguráciami skupiny parametrov pre odporúčanie. Táto skupina je zvláštna, keďže pozorujeme len jeden parameter hoci tu patrí aj parameter *rc* teda počet odporúčaných položiek, ktorého rôzne hodnoty ale sledujeme implicitne pri každom behu aj v ostatných skupinách parametrov. Ide teda o jediný parameter:

- **Veľkosť okna vyhodnocovania** *(skr. ews)*: {1,2,3,4,5,6,7, 8, 9, 10}

Nastavenie hodnôt ostatných parametrov sme zobrali z najlepšej konfigurácie čo sa týka presnosti vybranej z predchádzajúcej skupiny parametrov zhlukovania (konfigurácia 1 v tabuľke Tabuľka 15).

Zmena hodnoty parametra *ews* mení aj množinu sedení, ktoré sú vyhodnocované. Čím dlhšie je vyhodnocované okno tým menej sedení má dostatočnú dĺžku aby mohli byť vyhodnotené. To je závažný fakt, ktorý má vplyv na relevantnosť porovnávania jednotlivých konfigurácií.

8.25 Vyhodnotenie úspešnosti odporúčania pre ostatné parametre metódy

V tejto časti sa budeme venovať vyhodnoteniu odporúčania v spojitosti s rôznymi konfiguráciami skupiny osatných parametrov metódy. Ide o parametre (bližšie sú opísané v časti 5.3) a ich hodnoty:

- **Hraničná hodnota rozdielu identifikátorov zhlukovania** *(skr. tcdiff)*: {1,2,3,4, 5,6,7,8,9,10}
- **Minimálna rýchlosť** *(skr. mts)*: {15}

V tejto skupine sme sa rozhodli testovať rôzne hodnoty len prvého parametra *tcdiff*. Rôzne hodnoty parametra pre obmedzenie minimálnej rýchlosti budeme testovať až vo vyhodnotení rýchlosti spracovania (7.3.2), kde sledujeme k akému zhoršeniu metriky *presnosť* bude dochádzať pri zrýchľovaní spracovania transakcií na už vybranej celkovo najlepšej konfigurácií. Nastavenie hodnôt ostatných parametrov sme zobrali z konfigurácie vybranej z predchádzajúcej skupiny parametrov odporúčania, kde sme pozorovali len jeden parameter určujúci veľkosť vyhodnocovacieho okna. Vybrali sme nastavenie vyhodnocovacieho okna s veľkosťou 2, ktoré príliš nezmenšuje množinu vyhodnocovaných sedení a zároveň dosahuje jedny z najlepších výsledkov metriky *presnosť*.

8.26 Celkové vyhodnotenie úspešnosti odporúčania pre najlepšie konfigurácie

Na základe vybraných najlepších konfigurácií zo všetkých skupín parametrov ako sme ich uvádzali v predchádzajúcich častiach sme skonštruovali priestor prehľadávania v ktorom budeme hľadať celkovo najlepšiu konfiguráciu. Ide o parametre a ich hodnoty:

- **Minimálna podpora** *(skr. ms):* {0.005, 0.03, 0.05}
- **Miera uvoľnenia** *(skr. rr)*: {0.1, 0.5}
- **DÍžka segmentu** *(skr. sl)*: {25, 50, 500}
- **Maximálna dÍžka množiny** *(skr. mil)*: {10}
- **Veľkosť okna** *(ws)*: {15}
- **Počet zhlukov** *(skr. gc)*: {6, 8}
- **Hraničný počet zmien v modeli používateľa** *(skr. tcu)*: {5}
- **Hraničný počet zmien v mikrozhlukoch** *(skr. tcm)*: {400, 800}
- **Maximálny počet mikrozhlukov***(skr. mmc)***:** {100,1000}
- **Veľkosť okna vyhodnocovania** *(skr. ews)*: {1,2,3,4,5}
- **Hraničná hodnota rozdielu identifikátorov zhlukovania** *(skr. tcdiff)*: {15}
- **Minimálna rýchlosť** *(skr. mts)*: {15}

PRÍLOHA H – Dodatočné materiály k výslednému vyhodnoteniu pre dataset SACBEE

V tejto časti uvádzame niektoré podrobnejšie grafy a tabuľky z celkového vyhodnotenia pre dataset SACBEE.

PRÍLOHA I – Plán pre DP III

V tejto prílohe uvádzame plan práce v poslednej fáze projektu v bodoch, ktoré su zoradené podľa priority.

1. **Vyhodnotenie na datasete z inej domény.** Plánujeme použiť na vyhodnotenie ďalší dataset z inej domény. Pravdepodobne z domény novín.
2. **Článok. Zostavenie a zaslanie článku na konferencie IIT.src, UMAP prípadne RecSys.**
3. **Technická dokumentácia a implementácia.** Plánujeme dotvoriť technickú dokumentáciu s podrobnejším diagramom tried. Tiež doplniť dokumentáciu generovanú z kódu. A doplniť krátky návod na použitie.
4. **Dokončenie niektorých častí analýzy**. Hlavná časť analýzy je v podstate hotová. Pre úplnosť však chceme dokončiť ešte analýzu niektorých zaujímavých algoritmov v prílohách.
5. **Integrácia do MOA.** Integrácia je už funkčná, chceme len trocha upraviť tak aby ju bolo možné spúšťať z GUI, ktoré MOA ponúka.
6. **Implementácia prototypu pre distribuované prostredie.** Už sme začali vytvárať návrh a chystáme sa implementovať našu metódu v rámci Apache Storm a s použitím úložiska Redis. Z časového hľadiska a aj z problémov, ktoré distribuovaná povaha tohto návrhu prináša neočakávame že budeme schopní aj toto riešenie vyhodnotiť tak detailne ako už vytvorený experimentálny prototyp. Skôr sa jedná o návrh a prvý prototype možného riešenia v praxi.

PRÍLOHA J – Obsah elektronického media

- *source* - obsahuje zdrojový kód imlementácie experimentálneho prototypu a skripty predspracovania dát.
- *dokumenty* - obsahuje kompletny text a prílohy.
- *vysledky-ALEF* - obsahuje detailne získané výsledky.
- *vysledky-SACBEE*