Fast recognition and application of Web users' behavioral patterns*

Tomas Chovanak, Ondrej Kassak, Maria Bielikova
Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovicova 2, 842 16 Bratislava, Slovakia
{name.surname} @stuba.sk

ABSTRACT

Understanding of website user behavior is a crucial assumption for improving the website and user experience with it. Typical and repeating features of behavior during user's visit of website can be represented through behavioral patterns. In this work we represent behavioral patterns as frequent itemsets of actions frequently performed by users in their browsing sessions. Behavioral patterns have wide usage. They can be used to create recommendations, predict user's intentions (which can be subsequently used to cache predicted pages), improve website design, structure to complex understanding of users' behavior. This work responds to actual trend of Web personalization, focusing on needs of individual users and also to trend of data streams usage enabling processing of high number of data incoming in large volumes. In this paper we propose a method for behavioral patterns recognition combining global patterns with patterns specific to groups of similar users. Proposed method was evaluated indirectly through recommendation task. We performed several experiments over data from e-learning and news domains. Our results clearly show that combination of common global patterns and specific group patterns reaches higher prediction precision than its components used individually. Inclusion of group patterns also brings only constant computational load, which supports its maintenance in production usage.

CCS CONCEPTS

• Information systems → World Wide Web → Web mining • Computing methodologies → Machine learning → Learning paradigms → Supervised learning

KEYWORDS

Behavioral Patterns, Frequent Itemsets, Clustering, Data Stream; Recommendation, Data mining

1 INTRODUCTION

Understanding of behavior of website users is crucial for the site personalization and adaptation of its content and structure. Every user is unique and his actions subject to his actual aim, context etc. But when we look at behavior of multiple users together, we would be able to observe some regularities and actions typical for specific situations. These regularities are in general known as the behavioral patterns. The behavioral patterns may be modelled in various ways as frequent itemsets [1], frequent sequences of actions [2] or association rules [1]. These patterns can be applied to user groups of various sizes. For example, in e-learning domain, users will be probably segmented according to different learning specializations, paths or learning speed. Another example could be from news domain where users probably will be segmented according to their common preferences for different content categories (global news, local news, entertainment, sports etc.). The best segmentation of users however may be hidden and not so clear as we outlined. Smart detecting of groups of users with similar behavior and their typical patterns may lead to better understanding of users' intentions.

Knowledge of the behavior (which may be represented by behavioral patterns) of web users have broad utilization in different applications as outlined in [3]. It may be used for supporting website personalization, predicting users' behavior, caching web pages or making business decisions such market segmentation.

User actions within the website are implicit and therefore objective source of information directly describing his behavior. Mining the data from the actions results into objective information or even knowledge about users' behavior.

Nowadays a huge amount of web usage data is generated especially within large websites with many users (i.e. big news portals, social networks). As this data come in a massive and potentially infinite stream, it has become important to be able to process them as a fast in-memory process. If we want to effectively gain knowledge about recent users' behavior and immediately use it, it is suitable to use methods able to process the data by a single pass. Data stream algorithms are built upon models that are updated incrementally in an online time. Traditional data mining algorithms however usually require more than one pass over all instances in database, which decrease their usability.

Goal of this work is to respond to current trends of Web personalization and to focus on needs of individual users by discovering behavioral patterns not only on the level of global site community, but also smaller communities of similarly behaving users. We believe that identification of communities of similarly behaving users will help to discover specific behavioral patterns, which cannot be recognized on the global level. Using knowledge about both types of behavioral patterns and even their combination will enhance their quality and usability in different applications. For this reason, we propose an innovative method for identification

of global and also group behavioral patterns and their mutual combination specialized for recommendation of user's future visited pages. At first, our method uses data stream clustering algorithm for segmenting active users to several groups according their actual behavior. Next, it uses algorithm for mining global and group behavioral patterns (represented as frequent closed itemsets) from data stream. As a method application, we use identified patterns for recommending interesting pages for users to visit. Proposed method is able to detect recent behavior of global community and also smaller communities, identify behavioral patterns, combine them and recommend interesting pages to the users.

The rest of this paper is organized as follows. In section 2, we describe three different research areas related to method we propose: frequent patterns mining over data stream, clustering over data stream and recommendation. In section 3, we describe details of proposed method and its input variable tuning. In section 4, we describe evaluation of our method by the recommendation task and the methodology of performed experiments. We conclude the paper in section 5.

2 Related Work

Our method deals with three main tasks: segmenting users to groups, mining frequent patterns, recommendation of user's future visited pages. In this section, we describe actual state-of-the-art approaches specialized for these tasks and discuss their usability for our method.

2.1 Predicting users' behavior and recommendation

Behavioral patterns can be used as an input for wide scale of tasks as data analysis, future user actions prediction, classification or personalized recommendation.

In [4] WebPUM method was proposed, which represents behavioral patterns as partitions resulting from graph algorithm applied on user navigation graph, where nodes represent web pages and edges their mutual weighted connections. The weights are calculated based on intensity and frequency of page pairs being visited in same sessions. Based on the graph, the patterns are identified and pages recommended to users according their actual behavior.

Next interesting method predicting users' future steps was proposed in [5]. In this system fuzzy c-means clustering is used to find behavioral patterns represented as association rules.

In [6] authors propose new heuristic used to identify users' sessions. They use DBscan algorithm to cluster users' sessions into clusters representing behavioral patterns. DBScan is able to reveal otherwise ignored patterns because of their low support but high confidence when represented as association rules. Finally, the approach uses inverted index to effectively predict users' behavior online.

2.2 Mining frequent closed itemsets over data stream

Several algorithms were proposed for mining frequent closed itemsets task. Frequent itemset can be considered as simple, but effective representation of behavioral pattern which is less complex than e.g. frequent sequences. Frequent itemsets can be further used to compute association rules, which may be considered as more complex behavioral patterns representation and may lead to new knowledge about users' behavior. We focus on algorithms mining only frequent closed itemsets, which are complete and not redundant representations of all frequent itemsets [7].

Existing algorithms can be classified according to window model they use [8]. There exist several representations as landmark window containing all items from start of the stream or sliding window containing only most recent elements. Algorithms could be mining exact set of frequent itemsets or approximate set of frequent itemsets. Approximate mining is much more effective because it doesn't have to track all itemsets (frequent and not frequent) in history (compared to exact frequent itemset mining) and is able to well respond to conceptual drift.

First algorithm for incremental mining of closed frequent itemsets over a data stream is MOMENT [9]. It mines exact frequent itemsets using sliding window approach. It has become a baseline for solutions proposed later. It uses in-memory prefix-tree-based data structure called closed enumeration tree, which effectively stores information about infrequent itemsets, nodes that are likely to become frequent, and closed itemsets.

A successor of MOMENT algorithm called NEWMOMENT represents itemsets and window as bitsets [10]. It allows usage of efficient bitwise operations as for example to count support of itemsets or perform shift of sliding window.

CLOSTREAM uses different data structures and approach to mine exact closed frequent itemsets over sliding window than MOMENT [11] It uses Cid List and the SET function to find the closed itemsets similar to actual transaction. Unlike previous approaches it does not take so much time to search from a tree structure, because it only needs to intersect transaction with the specific closed itemsets [11].

Abandoning requirement to mine exact frequent itemsets helps to design fast algorithms for mining approximation of frequent closed itemsets like IncMine proposed in [12]. They use relaxed minimal support threshold to keep infrequent itemsets that are promising to become frequent later. They use update per batch policy that is different to all other algorithms we described here. It results in better time-per-transaction at risk of temporarily loosing accuracy of the maintained set while each batch is being collected [13]. In [13] authors use inverted index to efficiently address stored itemsets with IncMine algorithm.

Next algorithm named CLAIM for approximate frequent closed itemsets mining was proposed in [14]. This algorithm solves problem when conceptual drifts appear frequently and they slow down algorithm, by redefining frequent itemset definition and proposing usage of support value intervals considered as same value.

2.3 Clustering over data stream

Several algorithms were proposed for task of data stream clustering. CluStream algorithm is based on two (online and offline) components. Online microclustering component performs fast transformation of incoming data instances into compact approximate statistical representation. Offline macroclustering component uses this representation to get final results of clustering on demand [15]. This approach is adapted in other works using different macroclustering algorithms and altering microclustering phase slightly, like density based algorithm Denstream proposed in [16] that uses DBScan as macroclustering algorithm and defines new concepts of core-microclusters and outlier-microclusters. Denstream is able to detect clusters of arbitrary shapes, while it requires no assumption on the number of clusters. CluStream approach is adapted also in HPStream – projected clustering for high-dimensional data streams proposed in [17]. It outperforms basic CluStream with high-dimensional streaming data.

There are also other algorithms not based on CluStream. For example, another density based clustering algorithm D-Stream proposed in [18]. It maps input data into a density grid. Offline component clusters the grid. It adopts decaying technique to capture dynamic changes of a data stream. ClusTree [19] is parameter free algorithm that automatically adapts to the speed of the data stream with usage of compact and self-adaptive index structure for maintaining stream summaries. It incorporates the age of the objects to reflect the greater importance of more recent data.

3 Method for mining personalized behavioral patterns over a data stream

In this section, we describe method for online mining of behavioral patterns from the user activity within the website. The method combines global patterns, identified from behavior of all website users, with the group patterns determined for dynamically identified groups of similar users. Based on identified behavioral patterns, proposed method is able to recommend to individual users the pages they are probably going to visit in current session, cache these pages in advance etc.

Our method comprises of three logical components. First component ensures clustering of users into groups based on similarity of their behavior. Second component is used for searching for global and group behavioral patterns represented as closed frequent itemsets over data stream. Third component ensures application of found patterns. In this paper we focus to recommendation task, because of wide usage possibilities of process results (e.g., recommendation of interesting pages, caching probable future visits in advance). The third component of our method in addition ensures evaluation of the used application task. All these components are joined into process described in section 3.2 and illustrated in Figure 1.

We implemented the method into MOA framework [20], which contains implementations of data stream frequent patterns mining algorithm *IncMine* and data stream clustering algorithm *Clustream*. We use these algorithms to quick prove of concepts proposed in this paper. Later on, we plan to experiment with other algorithms and compare them with actual method results.

3.1 Clustering and user model representation

Important part of designed process is user clustering according to their similar behavior in recent past. Proposed method uses Clustream algorithm consisting of fast updated microclusters phase and macroclustering phase where k-means or other traditional clustering algorithms can be applied in microclusters to find the final clusters. As our method needs to perform macroclustering regularly as part of data streaming processing, this phase should be fast enough. Therefore, we use k-means, which is fast and easy to adjust algorithm. The defined process, however, should be considered as a framework and macroclustering algorithm as well as frequent patterns mining algorithm should be matter of choice depending on usage domain requirements.

As a data stream could be potentially infinite we need to prevent possible memory leak caused by adding new user models to memory and not maintaining the unused ones. We represent users' behavior simply as different frequency spectrums of their recent actions. The actions are stored to queue with limited capacity.

User model *u* consists of following attributes:

- *uid* : user identifier.
- *aq*: actions queue. Queue with limited capacity to store actions user took in recent history.
- *gid*: group identifier. Id of group where user was classified to in last macroclustering.
- *nsc*: new sessions count. Number of new sessions of this user since last macroclustering.
- *lmid*: last macroclustering id. Identifier of last macroclustering performed when this user model was active.

Macroclustering phase is performed in regular intervals (always after defined number of microclusters updates). *lmid* (last macroclustering id) represents a global macroclusterings counter, which is incremented always after new macroclustering phase is performed. User *u* is assigned to the specific group (identified by *u.gid*) with his first session after new macroclustering (if his *u.lmid* < *lmid*) and his *u.lmid* is set to same value as global *lmid.* To maintain memory size at constant size, inactive users have to be regularly removed. For this reason, after macroclustering phase, every user model *u* where *lmid* - *u.lmid* > *tcdiff* (threshold of clustering identifiers difference) is deleted.

3.2 User session processing

In this section we describe user session processing. The process is designed as a framework, where individual components are independent and can be replaced by another implementation (e.g., different clustering or frequent patterns mining algorithm) (Figure 1). Every user session, represented as set of actions, is continuously loaded from the data stream. User actions could vary from webpage visits to product purchases, shopping basket manages (adding or removing items), etc.

Proposed method uses evaluation approach interleaved test-then-train [20] where each individual example can be used to test the model before it is used for training. This way method is tested

on the whole dataset. Therefore, at first, user session is used for recommendation.

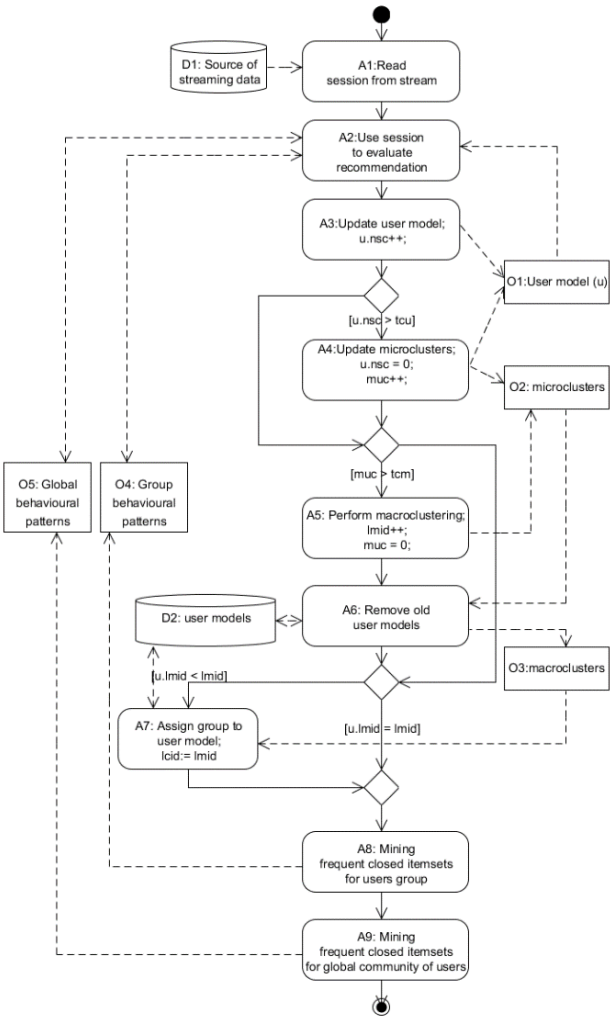

Figure 1: Activity diagram displaying processing of user sessions within proposed method. Diagram elements are tagged with prefixes describing their types: actions are tagged as A, data objects as O, data sources as D.

Next, the session is used for update of user model *u*. Actions from current session are added to queue in user model and *u.nsc* (new sessions count) counter is incremented. If *u.nsc* is greater than input parameter *tcu* (threshold number of changes in user model) then *u.nsc* is nulled and microclusters are updated with instance generated from *u.aq* (user model's actions queue) and *u.muc* (microclusters updates counter) is incremented by 1. If number of updates in microclusters (*muc*) is greater than given threshold *tcm* (threshold number of changes in microclusters) then *muc* is nulled and macroclustering is performed.

With macroclustering *lmid* (last macroclustering id) is incremented. And every old user model *u*, meeting condition that $lmid - u.lmid > tcdiff$, (where *tcdiff* parameter is threshold of clustering identifiers difference) is deleted from memory. Next if user model *u* has *u.lmid* attribute value other than current global *lmid* then *u.gid* is assigned identifier of group he belongs to according to last macroclustering performed. Lastly, user session becomes an input for algorithm mining frequent patterns (both global and group).

3.3 Application of behavioral patterns

As we mentioned before, behavioral patterns can be applied in wide scale of tasks. In this paper we used them to recommend pages to users based on their previous behavior in the actual session. Let the session be represented as vector of user actions $S = < a_1, a_2, \ldots, a_n >$ .Found behavioral patterns are $P = \{P_1, P_2, \ldots P_m\}$. Each pattern P_i is represented as set of actions $P_i = \{a_1, a_2, \ldots, a_k\}$. Let *ews* be size of evaluation window part of session. Condition $ews > 0 \wedge ews < |S|$ must be true to use actual session for evaluation of recommendation. Let first *k* actions of *S* be dedicated as evaluation window $W_e = < a_1, a_2, \ldots, a_k >$ and all other actions from *S* as testing part $T_e = < a_{k+1}, \ldots, a_n >$. Let r be number of recommended items. If condition $n >= (ews + r)$ is not met, then *S* is ignored for evaluation. We use following strategy to choose behavioral patterns according to actual evaluation window:

1. For each P_i in *P* (containing global patterns and patterns from group user belongs to) approximate support value normalized to *<0,1>* interval is computed. Let's mark it as $support(P_i)$. Size of intersection of P_i with W_e is determined with *LCS* algorithm (least common subset). It is also normalized to *<0,1>* interval. Let's mark it as $lcs(P_i, W_e)$.
2. All patterns P_i (global and group) are sorted. First by size of intersection with W_e descending. Second by descending support value.
3. Let *M* be map of items and their *"votes"*. By iterating over all patterns votes values of items are updated. Votes value of item *i* that is contained in pattern P_i and not in W_e is incremented by $support(P_i) * lcs(P_i, W_e)$.
4. Finally, *M* is sorted descending by votes values and best *r* items are picked to be recommended to user.

3.4 Summary of parameters used

One of critical parts of designed process is to properly set up the input parameters of individual method parts (clustering algorithm, frequent patterns mining algorithm, etc.). As there exist high number of input values combinations, the method tuning could have exponential complexity. To optimize such a process, there should be used some optimizations as for example tuning individual method parts separately etc. For this reason, we divided method input parameters into four categories according their purpose (Table 1).

We use *IncMine* algorithm [13] implemented in MOA framework for mining frequent closed itemsets over data stream. We search for best settings of its input parameters: *ms*, *rr*, *sl*, *ws*.

Minimal support (*ms*) is value corresponding to σ parameter in *IncMine* algorithm that is used to compute progressive function of minimal support (MST) for different segments of actual window. Relaxation rate (*rr*) is value corresponding to *r* parameter in *IncMine* algorithm that is used to compute relaxed MST, which prevents it from deleting potentially frequent itemsets. Segment length (*sl*) is number of transactions (in our case these are user sessions) in one batch update. Window size (*ws*) is number of segments sliding window consists of.

We use *Clustream* algorithm using k-means macroclustering implemented in MOA framework. We search for best setting of its input parameters: number of clusters, maximal microclusters count, together with input parameters specific to our method related to clustering part: threshold number of changes in user model, threshold number of changes in microclusters (we explained their usage in process description 3.2).

Category of recommendation parameters contains: *ews*, *rc*. Evaluation window size (*ews*) represents number of actions in user session used to identify best patterns to use with recommendation. All other actions in user session following this window are used as test set to evaluate generated recommendations. Recommendation count (*rc*) represents number of actions recommended to user.

Category of general method parameters contains: mts, tcdiff. As mentioned in [21] mining of frequent itemsets over the data stream requires balancing requirements for method accuracy and effectivity according to needs of specific task it will be used in. Our method offers option to set minimal required speed as input parameter *mts*. Threshold of clustering ids difference represents maximal difference between global macroclusterings counter (*lmid*) and counter in user model u (*u.lmid*). If this threshold is exceeded user model is marked as inactive and deleted.

Table 1: Summary of parameters.

abbr.	full name	category
ms	minimal support	IncMine
rr	relaxation rate	IncMine
sl	segment length	IncMine
ws	window size	IncMine
gc	groups count	clustering
tcu	threshold number of changes in usermodel	clustering
tcm	threshold number of changes in microclusters	clustering
mmc	maximal microclusters count	clustering
ews	evaluation window size	recommendation
rc	recommendations count	recommendation
mts	minimal transactions per second	general
tcdiff	threshold of clustering ids	general

4 Evaluation

As mentioned before, we evaluate proposed method indirectly by recommendation of items for user to visit within the actual session. We compare results of 3 different methods for behavioral patterns identification. First method generates recommendations using global patterns only (*GL*), second identify patterns specific for groups of users with similar behavior (*GR*) while third, our proposed method, combines previous two into new hybrid method (*GG*).

Recommendation results are compared based on *precision* metric. The recommendation is generated and evaluated for every session *S* from used datasets. The session is divided into training window *A* (its length is equal to input parameter *ews*) and testing window *B*. Let set of recommended items be *R*. We compute *precision* as

$$precision = \frac{|R \cap B|}{|R|} \quad (2)$$

Processing speed results are compared based on speed metric we defined as average number of sessions processed in a second. It is computed as

$$speed = \frac{number\ of\ processed\ sessions}{processing\ time\ [s]} \quad (3)$$

4.1 Datasets

We evaluated proposed method on two datasets from domains with different characteristics. First used dataset come from e-learning system ALEF (Adaptive Learning Framework) [23], second dataset belongs to news portal (NP).

Preprocessing of both datasets consisted mainly of users' sessions identification and omitting of too short sessions (1 action long).

ALEF dataset contains 24k user sessions from 870 users. The actions were performed between 26/10/2010 and 30/04/2013 (917 days but only 737 active). There is in average 33.37 sessions per active day with 15 actions in average per session. In this e-learning system, users could visit 2072 different learning objects (web pages).

In NP dataset there exist 334k sessions. Data was collected between 01/03/2015 and 01/07/2015 (122 active days). In average there exist 2739.8 sessions per day with 3 actions in average per session, which is significant difference from ALEF dataset. For simplicity of evaluation we removed all user sessions with length less than 3 as these sessions bring noise and lower the quality of mined. In NP there is 199k users. Many of them are much less active than users in ALEF and they don't return to the site so often if even. In news portal there exist several thousands of pages (articles) and they are active for too short time to be able to recognize behavioral patterns over their visits. For this reason, we abstract the pages into 85 categories (e.g., culture, sport) and recognized patterns over them. This abstraction brings significant patterns recognition method speed (lower number of possible actions and patterns) and patterns quality (higher patterns occurrence) increase in comparison to ALEF dataset.

4.2 Searching for best configurations

Considering number of input parameters our method takes (described in section 3.4), we used grid search approach to find most promising configurations maximizing recommendation precision. To reduce space of possible parameter values combinations, we tuned method parts independently based on their

category (Table 1). In this section we describe best found configurations of individual categories of parameters.

Results from ALEF dataset are affected by minimal speed requirements (*mts*), which we set to 15 transactions per second. Patterns for ALEF are discovered in high dimensional space of all possible pages, which slows down the processing. For this reason, multiple parameter configurations met the minimal speed requirements and have to be skipped. In NP dataset, processing is much faster due to lower dimensionality of actions (page categories) and therefore *mts* value we set doesn't affect the results quality. In Table 2 there are enumerated all parameters' values we searched for.

Let h be number of recent sessions stored in memory. It is computed as: $h = ws * sl$. Evaluation of every parameter configuration starts after certain number of transactions are processed. It is computed as: $h_m = \max_{c \in C} h_c$. Where C is set of all searched configurations of parameters. This approach is used for the reason that the configuration with smaller h is able to recommend sooner than configurations with higher h value. For first category (IncMine algorithm) we evaluated 270 different parameter configurations. In both domains we observed that configurations with longer window size (ws = 15) and shorter segment length (sl = 25) reach higher precision.

Next, we observed that too small value of *ms* causes generation of too many patterns, which causes slowing down the method and failing in minimal method speed requirement in ALEF domain. On the other hand, too high *ms* value causes generating of small number of patterns only, which could cause missing of some potentially important patterns.

Table 2: Parameters values used as search space in grid search.

abbr.	Values
ms	0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1
rr	0.1, 0.5, 0.9
sl	25, 50, 100, 150, 200, 500
ws	5,10,15
qc	2, 4, 6, 8
tcu	5,10,15
tcm	50, 100, 200, 400, 800
mmc	100,1000
ews	1,2,3,4,5,6,7, 8, 9, 10
rc	1,2,3,4,5,10,15
mts	15
tcdiff	1,2,3,4, 5,6,7,8,9,10

Tuning the parameters from second category used for clustering brought 120 different configurations. We observed that clustering users to small number of groups (<4) was less successful, because some groups were incorrectly joined together. In both domains, we observed best results when using smaller value of *tuc* (threshold of user model updates) in combination with higher *tcm* (threshold of microclusters updates). It means that microclusters are updated more often for recent users' behavior and also number of microclusters updates required, to perform macroclustering, is reasonably higher to capture enough information to be able to cluster users correctly. For third category of parameters (recommendation), we tuned only a parameter *ews* (evaluation window size). Different values of other parameters from this category (*rc*) were evaluated implicitly for every configuration to be able to compare all results for different number of recommended items. Let l_s be length of session *s*. For recommendation evaluation we could only use sessions with $l_s >= ews + rc$. That's why tuning of *ews* and *rc* changes the number of sessions usable for the evaluation. We cannot directly compare parameter configurations with different values of *ews* and *rc* (it would be only possible if we omit great number of short sessions). We observed that for ALEF dataset, the best results were accomplished for *ews* > 2 and *ews* < 6. For NP with many short sessions, *ews=1* performed the best results for recommendation of less than 4 items. Recommendation of multiple items, however, reached better results with higher *ews* value.

For last category of parameters (general), we observed that deleting inactive user models too soon (when *tcdiff* is low), after new macroclustering is performed, cause worse results. In both domains, we observed trend of precision increase after increasing the *tcdiff* threshold. In ALEF dataset ascending precision trend was visible until *tcdiff = 5*. Student behavior in ALEF is highly bound to actual course studied. For this reason, their user model should be removed after finishing the course. In news domain, an ascending trend in precision (with ascending *tcdiff*) was visible for all evaluated *tcdiff* values (1-10). In news domain we represent user actions as category visits, which are stable and thus usable for longer time periods. In this case, the models should remain until users are active in the website.

4.3 Search results

Finally, according to performed search, for each dataset, we chose the best configuration maximizing precision metric and processing speed (Table 3). For these best settings, we compared proposed method to the baseline methods using global and group only patterns (Table 4).

Table 3: Best configurations of parameters for both domains.

ms	ews	rr	sl	W	gc	Tc	Tcm	mmc	tcdiff
ALEF									
0.0	2	0.1	25	15	6	5	400	100	5
NEWSPAPERS PORTAL									
0.0	1	0.5	25	15	8	5	800	1000	15

We performed statistical t-test to compare precisions of baseline methods. For ALEF dataset our method reached significant increase of precision when compared to method using only global patterns ($p < 0.0001$, $t = 9.7153$, $df = 7600$, difference of means 0.0165) and also compared to method using only group patterns ($p < 0.0001$, $t=103.2904$, $df=7600$, difference of means 0.2143). For NP dataset we reached also significant increase compared to method using only global patterns ($p < 0.0001$, $t=3.8976$, $df=12598$, difference of means 0.0086) and also compared to method using only group patterns ($p < 0.0001$, $t=239.6845$, $df=12598$, difference of means 0.4137).

Table 4: Differences in precision of methods for best configuration in ALEF and NP. GG marks recommendation method using global and group patterns combination. GL marks method using global patterns only. GR marks method using only group patterns.

	P@1	P@2	P@3	P@4	P@5	P@10
ALEF						
GG	50.91%	50.97%	50.78%	50.70%	50.86%	50.38%
GL	48.94%	49.16%	49.12%	49.07%	49.04%	48.42%
GR	38.07%	37.98%	37.82%	37.67%	37.79%	37.13%
GG – GL	1.97%	1.81%	1.66%	1.63%	1.82%	1.96%
GG -GR	12.84%	12.99%	12.96%	13.02%	13.07%	13.25%
NEWSPAPERS PORTAL						
GG	65.08%	53.90%	55.79%	55.73%	54.79%	51.15%
GL	63.86%	52.81%	54.66%	54.23%	52.76%	48.89%
GR	13.35%	11.09%	15.78%	20.35%	23.65%	12.18%
GG - GL	1.21%	1.09%	1.13%	1.50%	2.21%	2.26.%
GG – GR	50.51%	41.71%	38.88%	30.82%	31.58%	50.34%

4.3 Evaluating relations between group and global patternsFor each user session, we evaluate separately results of recommendation approach that is using global behavioral patterns only (*GL*), group behavioral patterns only (*GR*) and using their combination (*GG*). This results in three different sets of successful recommendations (Figure 2).

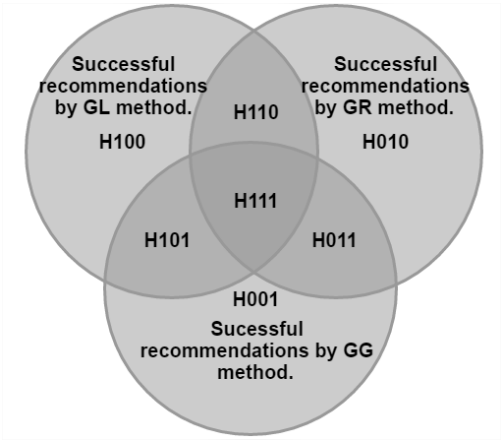

Figure 2: Venn diagram displaying sets of successful recommendations gained by different methods and marking of their intersections. Each part of the diagram is marked as H (hits) following by three bits. First bit marks set of successful recommendations gained with *GL*, second with *GR* and third with *GG*.

To be able to compare results of combined *GG* method not only to its parts (*GL*, *GR*), we present also the best theoretical combination of both individual methods (Table 5). In this case, the better result of *GL* and *GR* is chosen for every session. Based on this information, it is possible to evaluate the quality of used approaches combination.

From the experiment results, we observe quite large intersection of successful recommendations generated based on global and group patterns (*H110*). Number of successful recommendations generated by usage of global patterns only (*H100*) is much higher than number of successful recommendations generated with usage of group patterns only (*H010*). The reason is that small number of sessions within groups results into low quality patterns (low support). We observed that precision is higher in larger groups with many active users (several hundreds).

In addition to overall precision, we observed recommendation precision inside each user group during the stream processing. The results were logged in regular intervals before new macroclustering (in total, 9 measurements were realised during stream processing in ALEF and 18 measurements in NP) (Figure 3).

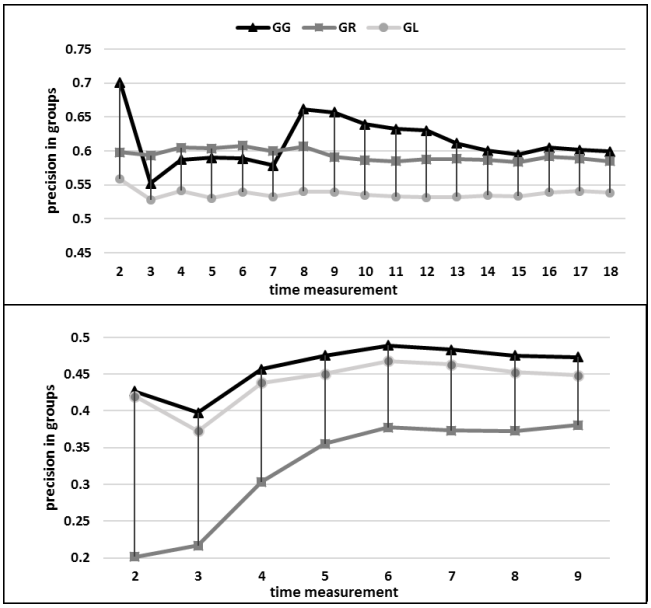

Figure 3: Average recommendation precision within groups for NP (upper) and ALEF datasets (lower). Precision is computed as average of precisions inside each group. *GG* marks method using global and group patterns combination. *GL* marks method using only global patterns. *GR* means method using only group patterns.

From these results is visible that *GG* for both datasets reached the highest precision, which shows that it is suitable to combine global and group patterns. The difference, however, lies in results of *GL* and *GR*. In ALEF dataset, *GL* reached better precision in comparison to *GR*, while in the case of NP dataset, the situation is opposite. It is caused by the number of users in the datasets. In NP, there exist high amount of users, so they can be clustered into highly similar and quality groups. For this reason and despite the short sessions, there can be identified quality patterns. In ALEF, there are less users, who in addition perform more specific sessions and thus it is unable to create such quality behavioral patterns. Despite this restriction, group patterns are useful, as can be seen from results of *GG*, which outperformed the *GL* method. Results of both methods are able to find unique patterns.

Table 5: Facts computed from sets of successful recommendations from different methods.

Description of	Computed as	ALEF					NEWSPAPERS PORTAL				
		P@1	P@2	P@3	P@4	P@5	P@1	P@2	P@3	P@4	P@5
Hypothetic situation, where more precise result is always chosen (from *GL* and *GR*)	H111+H110+H101+ H100+H011+H010	**55.14**	**57.02**	**58.40**	**59.06**	**59.88**	**66.79**	**55.92**	**59.33**	**60.65**	**61.00**
GG	H101+H011+H001+ H111	**50.71**	**51.26**	**51.73**	**51.96**	**52.41**	**64.88**	**53.68**	**55.68**	**55.66**	**54.91**
GL	H111+H110+H101+ H100	**49.27**	**49.85**	**50.44**	**50.73**	**51.06**	**63.69**	**52.62**	**54.58**	**54.19**	**52.76**
GR	H111+H011+H110+ H010	**32.41**	**32.93**	**33.21**	**33.24**	**33.57**	**13.06**	**10.86**	**15.43**	**19.88**	**23.07**
Recommendations generated uniquely of group patterns and not global patterns.	H011 + H010	**5.86**	**7.17**	**7.97**	**8.33**	**8.82**	**3.10**	**3.30**	**4.74**	**6.46**	**8.24**
Recommendations generated uniquely of global patterns and not group patterns.	H101+H100	**22.72**	**24.09**	**25.19**	**25.82**	**26.31**	**53.72**	**45.06**	**43.90**	**40.78**	**37.93**
Recommendations generated uniquely by combination of global and group patterns.	H001	**0.32**	**0.84**	**1.19**	**1.54**	**1.84**	**0.08**	**0.49**	**0.80**	**1.24**	**1.70**

4.4 Evaluating speed

For approaches specialized in data stream processing, computation time represent one of the crucial criterions of usability. For this reason, we observed, in addition to recommendation precision, also the processing speed of proposed method. logged in regular time intervals. To be able to consider computational cost of user clustering, group patterns creation and their combination with global patterns, we compared it to result of GL method computing global patterns only (Figure 4).

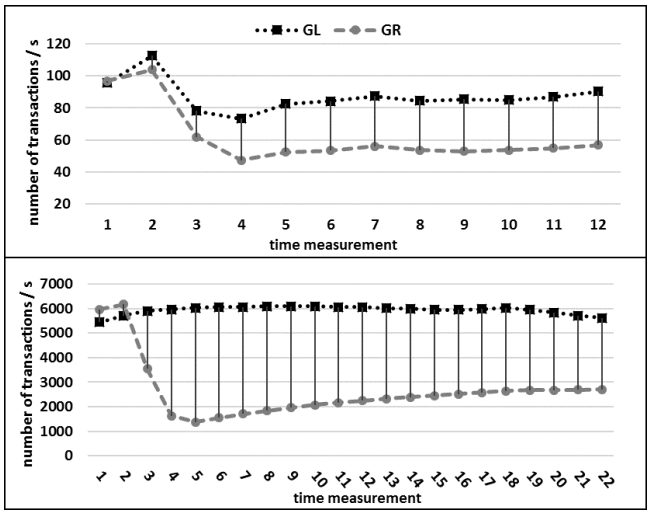

Figure 4: Average processing speed of ALEF (upper) and NP datasets (lower). The measurements are taken every 2000 transactions.

From the results is visible that additional cost caused by clustering and searching for group patterns is constant and thus maintainable in production. Another finding came from comparison of results between both domains. We can see that computation is faster in order in the case of NP dataset. The reason is the number of possible items, from which are identified the behavioral patterns (tens of categories in NP, hundreds of pages in ALEF). This in addition shows up, how items abstraction helps to significantly reduce computation time.

5 CONCLUSIONS

In this paper we propose a method, which is able to perform multiple tasks over data stream: segmenting users to groups dynamically, searching for group and global behavioral patterns and applying these patterns in recommendation task.

The method was evaluated on e-learning and news data, which have highly different characteristics (total size, number of users, different average length of session). We performed multiple experiments mainly with purpose to investigate how useful is combining global and group patterns. The method was evaluated indirectly through recommendation task, where we observed precision and speed (which is important because of data stream processing) of the method.

GG reached significant increase of precision when compared to GL and GR with both datasets.

In addition to overall precision, we observed recommendation precision inside each user group during the stream processing. In NP dataset, *GR* method outperforms *GL* within groups thanks to high amount of users forming high quality groups and therefore quality patterns. In ALEF dataset, *GL* method outperforms *GR* because there are less users performing more specific sessions and thus it is unable to create such quality patterns. However, proposed method (*GG*) outperformed *GR* and *GL* in both datasets within groups.

We showed that additional computational cost of GG compared to GL, caused by clustering and searching for group patterns, is constant and thus maintainable in production. Also that, abstraction of items from which are identified behavioral patterns results in significantly faster computation.

Next, we plan to evaluate proposed method with multiple different algorithms for mining frequent itemsets (top-k frequent itemsets mining algorithm without minimum support input [22]) and clustering (D-Stream and Clustree) and compare results. We

will also try to parallelize individual tasks of the method, which can help to make method even faster. Input data stream could be copied to 3 separate branches – user clustering, mining of group behavioral patterns and mining of global behavioral patterns.

REFERENCES

[1] Agrawal, Rakesh, Tomasz Imieliński, and Arun Swami. "Mining association rules between sets of items in large databases." Acm sigmod record. Vol. 22. No. 2. ACM, 1993.

[2] Agrawal, Rakesh, and Ramakrishnan Srikant. "Mining sequential patterns." Data Engineering, 1995. Proceedings of the Eleventh International Conference on. IEEE, 1995.

[3] Facca, Federico Michele, and Pier Luca Lanzi. "Recent developments in web usage mining research." International Conference on Data Warehousing and Knowledge Discovery. Springer Berlin Heidelberg, 2003.

[4] Jalali, M., Mustapha, N., Nasir Sulaiman, M. and 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

[5] Liraki Z. and Harounabadi A. 2015. Predicting the Users' Navigation Patterns in Web, using Weighted Association Rules and Users' Navigation Information. International Journal of Computer Applications. 110.12, 16-21.

[6] Anandhi, D., and MS Irfan Ahmed. "An Improved Web Log Mining and Online Navigational Pattern Prediction." Research Journal of Applied Sciences, Engineering and Technology 8.12 (2014): 1472-1479.

[7] Wang, Jianyong, and Jiawei Han. "BIDE: Efficient mining of frequent closed sequences." Data Engineering, 2004. Proceedings. 20th International Conference on. IEEE, 2004.

[8] Quadrana, Massimo, Albert Bifet, and Ricard Gavalda. "An efficient closed frequent itemset miner for the MOA stream mining system." AI Communications 28.1 (2015): 143-158.

[9] Chi, Yun, et al. "Moment: Maintaining closed frequent itemsets over a stream sliding window." Data Mining, 2004. ICDM'04. Fourth IEEE International Conference on. IEEE, 2004.

[10] Li, Hua-Fu, et al. "A new algorithm for maintaining closed frequent itemsets in data streams by incremental updates." Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06). IEEE, 2006.

[11] Yen, Show-Jane, et al. "A fast algorithm for mining frequent closed itemsets over stream sliding window." Fuzzy Systems (FUZZ), 2011 IEEE International Conference on. IEEE, 2011.

[12] Cheng, James, Yiping Ke, and Wilfred Ng. "Maintaining frequent closed itemsets over a sliding window." Journal of Intelligent Information Systems 31.3 (2008): 191-215.

[13] Quadrana, Massimo, Albert Bifet, and Ricard Gavalda. "An efficient closed frequent itemset miner for the MOA stream mining system." AI Communications 28.1 (2015): 143-158.

[14] Song, Guojie, et al. "CLAIM: An efficient method for relaxed frequent closed itemsets mining over stream data." International Conference on Database Systems for Advanced Applications. Springer Berlin Heidelberg, 2007.

[15] Aggarwal, Charu C., et al. "A framework for clustering evolving data streams." Proceedings of the 29th international conference on Very large data bases-Volume 29. VLDB Endowment, 2003.

[16] Cao, Feng, et al. "Density-based clustering over an evolving data stream with noise." Proceedings of the 2006 SIAM international conference on data mining. Society for Industrial and Applied Mathematics, 2006.

[17] Aggarwal, Charu C., et al. "A framework for projected clustering of high dimensional data streams." Proceedings of the Thirtieth international conference on Very large data bases-Volume 30. VLDB Endowment, 2004.

[18] Chen, Yixin, and Li Tu. "Density-based clustering for real-time stream data." Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2007.

[19] Kranen, Philipp, et al. "The ClusTree: indexing micro-clusters for anytime stream mining." Knowledge and information systems 29.2 (2011): 249-272.

[20] Bifet, Albert, et al. "Moa: Massive online analysis." Journal of Machine Learning Research 11.May (2010): 1601-1604.

[21] Lee, Victor E., Ruoming Jin, and Gagan Agrawal. "Frequent pattern mining in data streams." Frequent Pattern Mining. Springer International Publishing, 2014. 199-224.

[22] Tzvetkov, Petre, Xifeng Yan, and Jiawei Han. "TSP: Mining top-k closed sequential patterns." Knowledge and Information Systems 7.4 (2005): 438-457.

[23] Bieliková, Mária, et al. "ALEF: from application to platform for adaptive collaborative learning." Recommender Systems for Technology Enhanced Learning. Springer New York, 2014. 195-225.