Fast recognition and application of Web users' behavioral patterns*

T. Chovaňák
Istituto Officina dei Materiali
P.O. Box 1212
Italy
gubbiotti@corporation.com

O. Kašsák
Dipartimento di Fisica e Geologia
P.O. Box 6221
Italy
malago@affiliation.org

M.Bieliková
Dipartimento di Fisica e Scienze
P.O. Box 5000
Italy
fin@affiliation.org

ABSTRACT

Behavioral patterns can be understood as typical and repeating features of user's behavior during their visit of website. In this work we represent behavioral patterns as frequent itemsets of actions frequently taken by user's in their sessions. Frequent source of knowledge about behavior of users are web logs and actions taken during their visits to website aggregated to sessions. Whole process of processing web logs, finding behavioral patterns and their analysis is also known as Web Usage Mining. Found behavioral patterns may be used to create recommendations, predict user's intentions (which can be used to cache predicted pages), as support for website design change or complex understanding of website users' behavior. Existing methods of Web Usage Mining usually search for behavioral 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 and also 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 behavior of global community of users, but also about actual behavior and changes in behavior of smaller user communities. We evaluate contribution of combining global and group behavioral patterns in their application in recommendation task. We also observe way this method is able to detect unique behavior of specific groups of users in domain of e-learning system and newspapers web portal.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics • **Networks** → Network reliability

KEYWORDS

ACM proceedings, text tagging

1 INTRODUCTION

Understanding website users' behavior precisely is crucial for better personalization and adaptation of website content and structure. Every user is less or more different from others but often there are groups of users with similar behavior in some specific situations and time. Detecting groups of users with similar behavior and their behavioral patterns may lead to better understanding of users' intentions.

Web logs and users' actions stored there are often source of implicit information about users' behavior. Whole process of processing web logs, finding behavioral patterns and their analysis is known as Web Usage Mining (WUM). WUM is process consisting of 5 phases:

1. Collecting data from data sources
2. Preprocessing data
3. Patterns discovery
4. Analysis, validation and use of discovered patterns

One of main advantages of using WUM process to search for behavioral patterns is that data inputs consists of objective user feedback - actions taken by users. We can represent user session as set of actions taken by user (items) and behavioral patterns as frequent itemsets of actions gathered from preprocessed logs.

One of important challenges nowadays is fast in-memory processing of data generated as potentially infinite stream. Data streaming algorithms are built upon models that are incrementally updated with incoming data instances. Traditional data mining algorithms usually require more than one scan of all instances in database. In streaming data every instance should be processed only once. With data stream evolution conceptual drift can appear. Conceptual drift is important problem when performing data mining with streaming data. It is derived from changes in real environment where streaming data instances are created. Factors activating such changes are usually hidden and unknown. Conceptual drift is problem mainly with models created from old data before the drift happened which are consequently improper.

Goal of this work is to respond to current trends of Web personalization and focusing on needs of individual users by discovering behavioral patterns not only from global community od website users but also from smaller communities of users. We propose process with regard to requirements of data streaming algorithms, that is integrating data stream clustering algorithm used for segmenting active users to groups according their actual behavior with algorithm mining frequent closed itemsets over data stream that is able to respond quickly to conceptual drifts and is used to discover actual behavioral patterns represented as frequent closed itemsets from global community of users and from detected groups of users. Proposed method is able to detect recent behavior of global community and smaller communities and also changes of their behavior in time.

There are many applications of such discovered knowledge. For Web personalization as such, for predicting users' behavior, for caching web pages to memory, as important input when changing design of website or making some business decisions such e.g. about market segmentation.

We apply discovered knowledge about users' behavior in recommendation task and evaluate gain from combination of behavioral patterns for global community with patterns discovered for smaller communities against approach where we use only patterns discovered for global community in this specific task.

The paper is organized as follows. In §2, we review three related research directions. In §3, we describe method for mining global and group behavioral patterns over data stream and use them in recommendation task. In §4, we describe methodology of experiments and data we used to evaluate proposed method and simultaneously results we observed. We conclude the paper in §5.

2 Related Work

In this section we review few related research works. Our proposed process is combination of existing stream data mining algorithms from which every has its own research area. It's new application of WUM process that is able to run as whole in online environment. We apply extracted knowledge to recommend items to users, but there are other possible applications to it.

2.1 Existing applications of WUM to predict users' behavior and recommend items

Found knowledge about users' behavior with some of different representations of behavioral patterns can be applied to predict next steps of user or recommend items or products.

In [1] WebPUM method was proposed which represents behavioral patterns as partitions resulting from graph partitioning algorithm applied on user navigation graph, where nodes are web pages, connections between them have weight computed from time and frequency information of users' sessions. Weight of connection rises as more frequently two pages appear in sessions together and smaller time gap between them and more time users' spent on pages. They use gained patterns to recommend pages to users according their actual behavior.

Next interesting method predicting users' next steps was proposed in [2]. It is sophisticated system using fuzzy c-means clustering to find behavioral patterns represented as association rules.

In [3] 3 new contributions to enhance WUM process were proposed. First is heuristic to identify users' sessions. Second is usage of DBScan algorithm to cluster users' sessions. DBScan is able to reveal otherwise ignored patterns because of their low support but high confidence when represented as association rules. Their last proposed contribution is using inverted index to effectively predict users' behavior online.

2.2 Algorithms for mining frequent closed itemsets over data stream

Our method represents behavioral patterns as closed frequent itemsets extracted from fast streaming data. Several algorithms were proposed for this task. We focused on algorithms mining only frequent closed itemsets which are complete and not redundant representation of all frequent itemsets. That significantly reduces size of search space.

Existing algorithms can be classified according to window model they use. It could be 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 can be much more effective because it doesn't have to track all itemsets (frequent and not frequent) in history and is able to respond to conceptual drift well.

First algorithm for incremental mining of closed frequent itemsets over a data stream was MOMENT, proposed in [4]. It mines exact frequent itemsets using sliding window approach. It has become a reference for solutions proposed later. They propose 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.

Algorithm NEWMOMENT proposed in [5] makes MOMENT more efficient. It represents itemsets and window as bitsets. This representation allows usage of efficient bitwise operations for example to count support of itemsets or perform sliding of window with bitwise shift.

CLOSTREAM proposed in [6] uses different data structures and approach to mine exact closed frequent itemsets over sliding window than MOMENT.

Abandoning requirement to mine exact frequent itemsets helps to design fast algorithm for mining approximation of frequent closed itemsets like that proposed in [7] called IncMine. 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 [8]. In [8] They use inverted index to efficiently address stored itemsets with IncMine algorithm.

Next algorithm named CLAIM for approximate frequent closed itemsets mining was proposed in [9]. This algorithm is trying to solve problem when conceptual drifts appear frequently and slows down algorithm by redefining frequent itemset definition and proposing usage of support value intervals considered as same value.

2.3 Algorithms for clustering over data stream

Our method clusters user models into groups according their similar behavior. We adapt approach proposed in [10] where authors introduced Clustream framework for clustering over data stream. It is based on online microclustering component performing fast transformation of incoming data instances into compact approximate statistical representation and offline

component using this representation to perform macroclustering and get results of clustering on demand.

3 Method for mining personalized behavioral patterns over data stream

In this section, we describe method for mining behavioral patterns common to global community of users and behavioral patterns common to dynamically identified groups of users over data stream and their application in recommendation task.

3.1 Input data format

We consider one data instance in data stream as one user's session represented as set of actions user made identified by unique identifiers. Actions could be diverse like etc. visits of webpages, buying of products, adding or removing items to or from shopping basket in e-shop.

3.2 User model representation

Important part of designed process is clustering of users to groups according their similar recent behavior. As data stream could be potentially infinite we need to prevent possible memory leak caused by adding new user models to memory and not deleting old. We represent users' behavior simply as different frequency spectrums of their recent actions. We use queue with limited capacity to store this actions. Capacity of this queue is one of input parameters. As we said we use Clustream framework for clustering user models consisting of fast update of microclusters part and macroclustering part where k-means or other traditional clustering algorithm can be applied to found final clusters. Macroclustering is performed on regular basis. Every macroclustering has its id number assigned.

User model *u* consists of this fields:

- *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 user model when this user model was active.

Let current macroclustering id be *lmid* (last macroclustering id). User *u* is assigned new group identifier on his first session after new macroclustering was performed. So only if his *u.lmid* is other than *lmid*. Always after new macroclustering is performed every user model *u* where *lmid* - *u.lmid* > *tcdiff*, where *tcdiff* (threshold of clustering identifiers difference) is input parameter, are deleted to prevent memory leak.

3.3 Behavioral pattern representation

We represent behavioral patterns as frequent closed itemsets with computed support value. We discriminate between global and group behavioral patterns. Global behavioral patterns are mined from behavior of whole community of users. Group behavioral patterns are patterns mined from behavior of concrete segment of community of users identified with clustering part of designed process.

3.4 Application of behavioral patterns

As we said found behavioral patterns could be applied as input to diverse other tasks. We apply found behavioral patterns to recommend pages to users in session. Let actual session be represented as vector of user actions $S = < a_1, a_2, \dots, a_n >$. Found behavioural patterns are $P = \{P_1, P_2, \dots P_m\}$. Each pattern P_i is represented as set of actions $P_i = \{a_1, a_2, \dots, a_k\}$. 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 $W_e = < a_1, a_2, \dots, a_k >$ and all other actions from *S* as testing part $T_e = < a_{k+1}, \dots, a_n >$. 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 approximate support value is computed. Let mark it as $support(P_i)$. Intersection of P_i with W_e is found with *LCS* algorithm (Least common subset). Let's mark it as $lcs(P_i, W_e)$.
2. All patterns (global and group) are sorted. First by size of intersection with We descending. Second by support value descending.
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 $support(P_i) * lcs(P_i, W_e)$ both normalized to <0,1> interval.
4. Finally, *M* is sorted descending by votes values and best *r* items are picked to be recommended to user.

3.4 User session processing

In this section we describe how every user session is processed. On Figure 1 is activity diagram for this process. This process is designed with regards to evaluation and experiments we perform on it. It is framework where individual components can be changed (etc. different clustering or frequent closed itemsets mining algorithm). User session represented as set of actions is loaded into process from data stream. First it is used in recommendation task as we proposed in previous section and other evaluation purposes as we describe them later. Next user model *u* is updated. 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 with appended id of group user belongs to is input to algorithm for mining frequent closed itemsets for specific group and same user session alone is input to same algorithm but performed for all users.

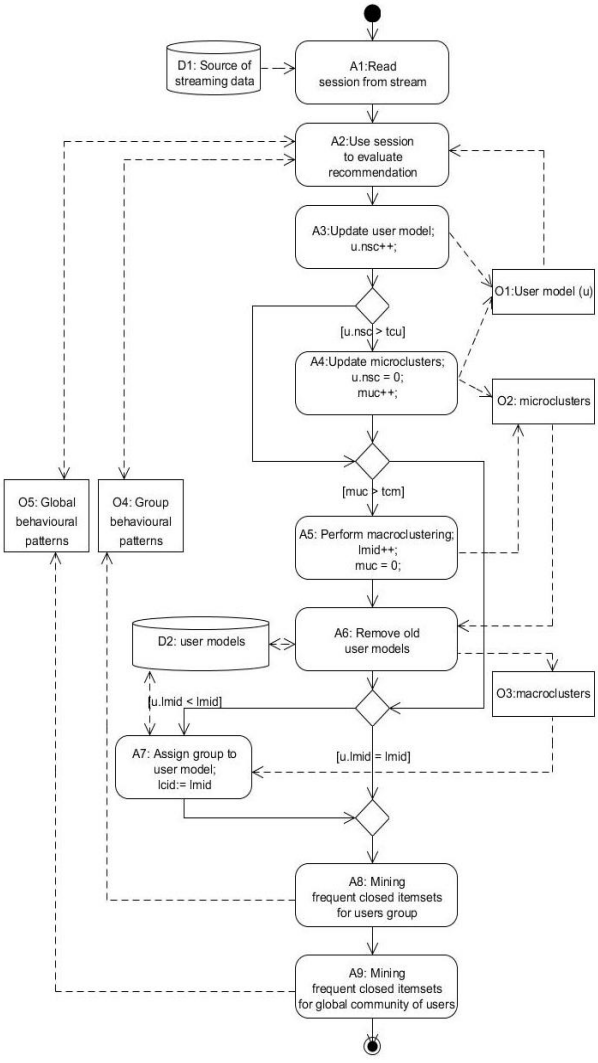

Figure 1: Activity diagram displaying processing of user sessions. Prefix A means action, prefix O means data object, prefix D means data source.

Some parts of designed process could be parallelized. Data stream could be copied to 3 separate branches. One branch performing clustering part of process, second branch performing mining of group behavioral patterns and third branch performing mining of global behavioral patterns. For our evaluation purposes sequential process was used. But in real application parallelization could be easily adapted to perform processing even faster.

3.4 Speed regulation

As mentioned in [11] task of mining frequent itemsets requires balancing requirements for accuracy and effectivity. Higher speed means worse accuracy and better accuracy means lower speed. User should have option to set this balance according to his actual needs. In our method we give to user option to set minimal required speed as input parameter *mts* (minimal transactions per second). As we mentioned, *IncMine* algorithm processes transactions in batches. Every update of batch (segment) is critical part. Let *actspeed* be actual average speed of processing in transactions per second. Let *ctrans* be actual number of processed transactions from start of measuring. Then:

$$actspeed = \frac{ctrans}{mts} \quad (1)$$

Let *tstart* be time when measuring started. Let *tsupdate* be time when update of batch started. We compute maximal allowed time of update *tmax* before each update as:

$$tmax = actspeed - tsupdate + tstart \quad (2)$$

When frequent itemsets update in current batch takes more than *tmax* it is simply stopped. It means some frequent closed itemsets won't be discovered. In next section we evaluate what effect does it have on accuracy when changing *mts* parameter value. Interesting solution to this problem would be using other algorithm for mining frequent closed itemsets from batch of transactions with approach of mining *k* most frequent itemsets because there is no need to set minimal support threshold like one proposed in [12] and best patterns are discovered first. Implementing and evaluating this kind of solution is actually outside of scope. As we see in evaluation part high dimensionality is problem that primarily causes speed deceleration. But with moderate number of different possible user actions logged solution is very fast.

3.4 Summary of parameters used

One of critical parts of designed process is number of input parameters. There are input parameters for clustering algorithm and for mining frequent closed itemsets algorithm and some parameters required by process itself. We divided these parameters into four categories according their purpose. Summary of parameters and categories is in Table 1. We use *IncMine* algorithm as implemented in [13] for mining frequent closed itemsets over data stream. It has its own parameters. Minimal support 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 is value corresponding to *r* parameter in *IncMine* algorithm that is used to compute relaxed MST which prevents from deleting potentially frequent itemsets. Segment length is number of transactions in one batch update. Window size is number of segments sliding window consists of. We use

Clustream framework as implemented in MOA framework. Important input parameter is desired number of clusters. For clustering also other threshold parameters are important we already mentioned in process description (*tcu*, *tcm*).

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 difference	general

4 Experiments

In this section we investigate how combining of global patterns with group patterns using our method is useful specifically when applied to recommend items to users in domain of education system ALEF and in domain of newspaper portal.

4.1 Datasets

First dataset we use is from weblog of education system ALEF (Adaptive Learning Framework) [14]. Preprocessing mainly consisted of users' sessions identification cleaning and removing too short sessions (1 action). Dataset consists of 24594 user sessions of 870 users and their actions from 26/10/2010 to 30/04/2013 (917 days but only 737 active). It is average 33.37 sessions per active day. There are 2072 different learning objects users could visit. We represent actions as visits to these objects. Average session length is 15. Second dataset is from anonymous news portal (we will refer to it as NP from now on). Users' sessions there are much shorter. For simplicity of evaluation we removed all sessions with length less than 3. Data was collected from 01/03/2015 to 01/07/2015 (122 active days). There are 334 257 sessions of 199 196 users. It is average 2739.8 sessions per day. There are 85 categories of pages. We represent actions as visits to these categories. ALEF dataset has much higher dimensionality which, as we will see, causes significant speed deceleration.

4.2 Methodology

We evaluate recommendation with *precision* metric. Evaluation always starts from transaction number computed as $max(ws) * max(sl)$ from all used configurations of *ws* and *sl* parameters values, because some configuration with shorter windows and segments are able to recommend sooner than configurations with longer *ws* and *ls*. We evaluate every session *S*. It is separated to evaluation window *A* and testing set *B*. Let number of recommended items be *N*, and set of recommended items *R*. Let *H* be bitwise vector of length *N*, where value 1 on *i-th* position means that *i-th* recommended item belongs to $R \cap B$. We compute *precision* as

$$precision = \frac{|R \cap B|}{N} \quad (3)$$

For each user session we evaluate separately results of recommendation using only global behavioral patterns, only group behavioral patterns and using both of them in the way we proposed. This results in three different sets of successful recommendations. We keep track of all intersections of this sets as you can see in Figure 2. This gives us few interesting results we describe later.

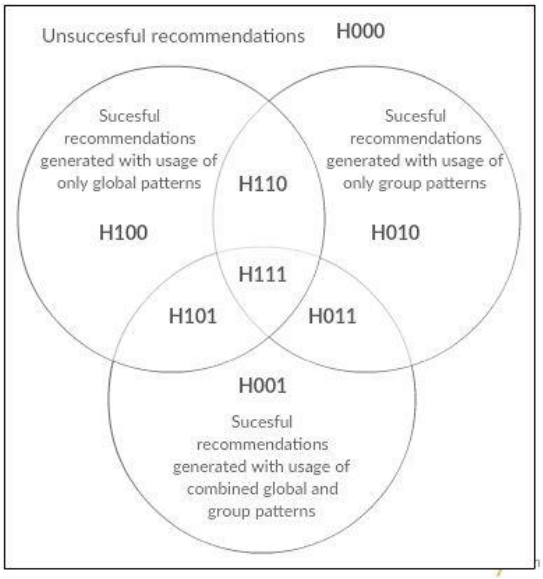

Figure 2: Venn diagram displaying sets of successful recommendations gained by different methods and marking of their intersections.

Considering number of input parameters our method takes, we decided to use grid search approach to find most promising configurations of parameters and observe correlations between different parameters values and results of recommendation. To reduce search space, we divided parameters into four categories (see Table 1) and performed search for most promising configurations separately for this categories and finally for combinations of best configurations from them.

4.2 Searching for best configurations

In this section we describe our observations from evaluations of individual parameter categories and after that we show results of evaluating best configuration from different aspects. Results from ALEF domain are affected by minimal speed requirements we set

to 15 transactions per second. Patterns for ALEF are discovered in high dimensional space which significantly slows down the processing. Many configurations are so slow that minimal requirements are met which results in worse precision than gained without using this threshold. Results from newspapers domain are not affected by our speed restriction. In Table 2 are all values we searched best configuration for.

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
gc	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

For first parameters category for *IncMine* algorithm we evaluated 270 different configurations. In both domains we observed that configurations with longer window size (*ws* = 15) and shorter segment length (*sl* = 25) are better in precision. It means remaining history of 375 sessions in memory. With average number of sessions per active day this is something more than 3 hours in newspapers domain and about 14 days in ALEF. In newspapers domain content is changing very fast which may cause frequent change of patterns compared to ALEF users progressing in their studies usually on week by week basis (or longer). Next we observed that too small *ms* value causes generating too many patterns, that causes slowing down method under minimal speed requirement in ALEF domain where we search for high dimensional patterns which slows down method significantly. But too high *ms* value causes generating only few patterns missing some potentially important patterns.

For second parameters category for clustering we evaluated 120 different configurations. We observed that segmenting to small number of groups (<4) was slightly less successful, probably because some important groups were wrongly interleaved together in that case. In both domains we observed best results when using smaller *tuc* (threshold of user model updates) in combination with higher *tcm* (threshold of microclusters updates). It means microclusters are updated more often with more actual users' behavior and number of microclusters updates required to perform macroclustering is reasonably higher to capture enough information to be able to segment users correctly.

For third category of parameters for recommendation we evaluated only values of parameter *ews* (evaluation window size). Different values of other parameter from this category (*rc*) were evaluated implicitly for every configuration. Changing *ews* and *rc* caused change in number of valid sessions to evaluate. It means we cannot compare these correctly on same evaluation set. Although we observed that best results were accomplished for *ews* > 2 and *ews* < 6 in ALEF. In newspapers domain, where there are many very short sessions *ews=1* performed much better for recommending one item exclusively. But higher ews values performed better with rare longer sessions. This is interesting knowledge as we could for example use *ews* of size 1 to predict exclusively next action after first visited item and later use *ews* of greater sizes to predict more actions.

For last category of parameters, we observed that deleting old user models too soon after new macroclustering is performed can cause worse results. In both domains we observed trend of ascending precision with moving *tcdiff* threshold higher. Old user models should be removed after some time to prevent possible memory leak. Also retaining many user models in memory implicates slowing down of method. In ALEF ascending trend seems to end with *tcdiff = 5* value, but this trend could be better understood having larger dataset. In ALEF students change their study subjects and old user model representing user who studied something and finished his study of subject should be removed because they are not actual. In newspapers domain, situation is different because readers probably don't change their behavior so regularly so it's better to remove old user models as late as given memory requirements permit.

Table 3: Differences in precision of methods for best configuration in ALEF and NP. GGC marks method using global and group patterns combination. GO marks method using only global patterns. DIFF marks difference. Precision is averaged for different values of ews.

	P@1	P@2	P@3	P@4	P@5	P@10
ALEF						
GGC	**50.73%**	**50.98%**	**51.09%**	**51.12%**	**51.34%**	**51.56%**
GO	**49.07%**	**49.51%**	**49.46%**	**49.52%**	**49.73%**	**49.73%**
DIF	**1.66%**	**1.47%**	**1.64%**	**1.59%**	**1.61%**	**1.83%**
NEWSPAPERS PORTAL						
GGC	**39.10%**	**47.36%**	**53.98%**	**58.19%**	**61.33%**	**64.40%**
GO	**37.65%**	**45.85%**	**52.53%**	**56.59%**	**59.47%**	**63.40%**
DIF	**1.46%**	**1.51%**	**1.45%**	**1.60%**	**1.87%**	**1.00%**

Finally, according performed search and observations, we described here, we chose best configuration in precision metric for each domain. In Table 3 we show differences in precision of chosen best configurations (in Table 4) between method using combination of group and global behavioural patterns and method using only global patterns.

Table 4: Best configurations of parameters for both domains.

ms	rr	sl	ws	gc	tcu	tcm	mmc	tcdiff
ALEF								
0.05	**0.1**	**25**	**15**	**6**	**5**	**400**	**100**	**5**
NEWSPAPERS PORTAL								
0.05	**0.5**	**25**	**15**	**8**	**5**	**800**	**1000**	**15**

Table 5: Facts computed from sets of successful recommendations from different methods using only global patterns or using only group patterns or both.

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
Precision gained if always perfect at choosing between usage of group or global patterns	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**
Precision gained using our method	H101+H011+ H001+H111	**50.71**	**51.26**	**51.73**	**51.96**	**52.41**	**64.88**	**53.68**	**55.68**	**55.66**	**54.91**
Precision gained with usage of only global patterns	H111+H110+ H101+H100	**49.27**	**49.85**	**50.44**	**50.73**	**51.06**	**63.69**	**52.62**	**54.58**	**54.19**	**52.76**
Precision gained with usage of only group patterns	H111+H011+ H110+H010	**32.41**	**32.93**	**33.21**	**33.24**	**33.57**	**13.06**	**10.86**	**15.43**	**19.88**	**23.07**
Precision gained with 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**
Precision gained with 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**
Precision gained with 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**

We performed statistical t-test for average precision values of both methods in from all evaluated parameters configurations. It tells us results are significant with $p < 0.0001$, $t=9.8366$, $df=7740$ a difference of means 0.0165 for ALEF. In newspapers portal domain results are significant with $p < 0.0001$, $t=3.8976$, $df=12598$ a difference of means 0.0086.

4.3 Evaluating relations between group and global patterns

As our methodology said we kept track of three separate sets (and their intersections) of successful recommendations given by method using only global patterns, method using only group patterns and method using both of them. To see how we mark them see Figure 2. We found out some interesting facts as you can see in Table 5. Theoretically if we always knew which patterns set (group or global) to use we could get higher precision than we get with our proposed approach to combine these sets of patterns, although it outperforms method using only global or only group patterns. There are few successful recommendations given only by method combining group and global patterns. We observe quite large intersection of successful recommendations generated with global patterns and group patterns. Number of successful recommendations generated with usage of only global patterns and not group patterns is much larger than number of successful recommendations generated with usage of only group and not global patterns. This could be caused by small number of sessions in groups used to train model which results in small number of strong enough and actual patterns as we see that precision is higher in groups that has larger number of active users (see figure ...).

Next we kept track of precision of recommendation inside each group of users during stream processing in discrete measurements taken always before new macroclustering (it is 9 measurements taken during stream processing in ALEF and 18 measurements in

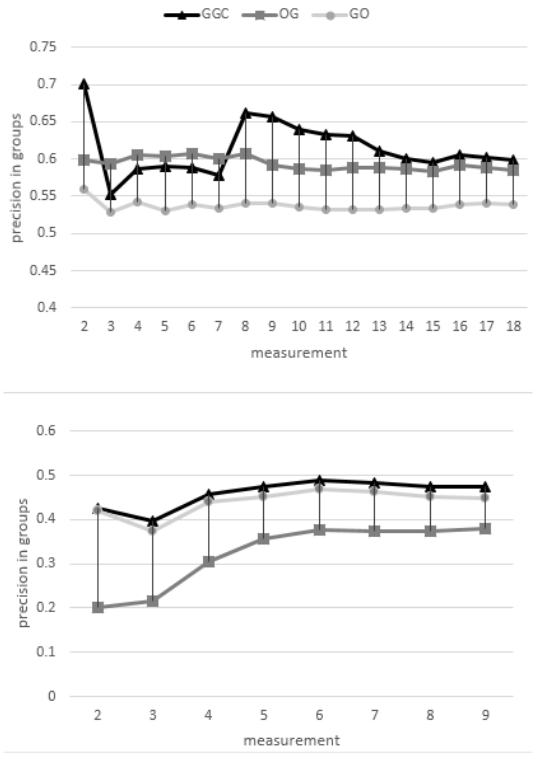

Figure 3: Precision inside groups in NP (upper chart) and ALEF (bottom chart). Precision is computed as average precision of all evaluated numbers of recommended items configurations and all detected groups of users. GGC marks method using global and group patterns combination. GO marks method using only global patterns. OG means method using only group patterns.

NP). In ALEF, global patterns are better in precision even inside groups and our hybrid method is slightly better or very similar. In NP domain it's opposite, group patterns are better than global. It means that group patterns in NP domain are strong enough to describe behavior of users with assigned group better than global patterns. On Figure 3 you can see only average values of precision during processing for all evaluated values of *rc* (number of recommended items) parameter, but we observed described trends with all values of *rc* independently.

On Figure 4 you can see ratio of number of unique patterns from all groups to number of all patterns. We consider pattern unique if there is no other such pattern in other groups or in set of global patterns. This tells us that group patterns are able to detect aspects of users' behavior not visible by global patterns.

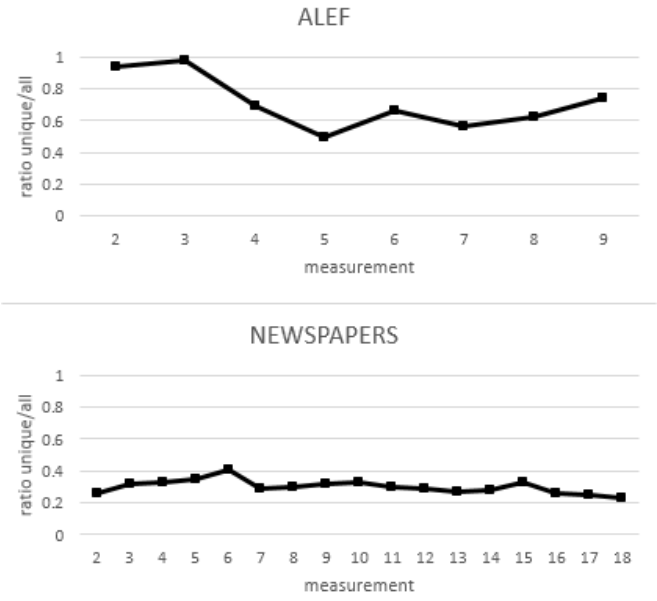

Figure 4: Ratio of number of unique patterns from all groups to number of all patterns in ALEF.

4.4 Evaluating speed

Finally, we evaluate more precisely speed of method using best chosen configuration. We run best configuration independently with evaluation component unplugged for 20 times measuring speed during each run in regular intervals. On Figure 5 you can see comparison of average speed results of method using proposed combination of group and global patterns and method using only global patterns. Of course we observe some cost caused by additional clustering and searching for group patterns but this difference doesn't appear to be growing with time. Next we tried to change parameter *mts* to see how much results are worse when speeding up the processing. As you can see on figure the difference between methods is retained with ascending mts values.

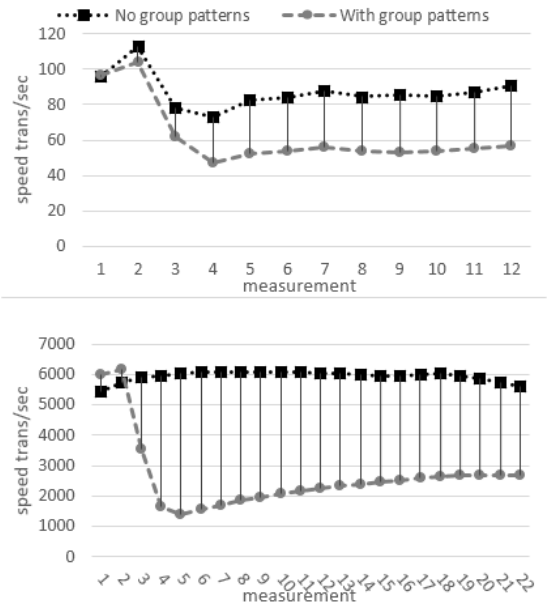

Figure 4: Average speed during processing of ALEF and NP dataset. Speed measurements taken every 2000 transactions.

5 CONCLUSIONS

In summary, we have performed both an experimental and theoretical study of the spin eigenmodes in dipolarly coupled bi-component cobalt and permalloy elliptical nanodots. Several eigenmodes have been identified and their frequency evolution as a function of the intensity of the applied magnetic field has been measured by Brillouin light scattering technique, encompassing the ground states where the cobalt and permalloy dots magnetizations are parallel or anti-parallel, respectively. In correspondence to the transition between the two different ground states, the mode frequency undergoes an abrupt variation and more than that, in the anti-parallelstate, the frequency is insensitive to the applied field strength. The experimental results have been successfully interpreted by the dynamic matrix method which permits to calculate both the mode frequencies and the spatial profiles.

REFERENCES

[1] 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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[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] 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.

[10] 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.

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

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

[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] 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.