Mining frequent generalized patterns for Web personalization

Panagiotis Giannikopoulos
University of Peloponnese,
Department of Computer Science and Technology, Tripoli, Greece
cst04006@uop.gr

Iraklis Varlamis
Athens University of Economics and Business,
Department of Informatics,
Athens, Greece
varlamis@aueb.gr

Magdalini Eirinaki
San Jose State University, Computer Engineering Department,
San Jose, CA, US,
magdalini.eirinaki@sjsu.edu

Abstract. In this paper we present FGP, an algorithm that combines the powers of an association rule mining algorithm (FP-Growth) and a generalized pattern mining algorithm (GP-Close) in order to efficiently generate rules from transaction data. Our Frequent Generalized Pattern (FGP) algorithm considers that all items that appear in a set of transactions, belong to categories organized in a taxonomy. It takes as input the transaction database and the taxonomy of categories and produces generalized association rules that contain transaction items and/or item categories. This algorithm is particularly useful for personalizing web sites with continuously updated content, such as, blog aggregators, or news portals. In this context, the transaction database contains user click-stream information and the hierarchy of item types is a thematic taxonomy of web pages. The algorithm generates frequent itemsets comprising of both web pages and categories. The results are used to generate association rules and consequently recommendations for the users. We experimentally evaluate the proposed algorithm using web log data collected from a newspaper web site.

1. INTRODUCTION

The role of recommendations is very important in everyday transactions. When buying a product, or reading a newspaper article, one would like to have recommendations on related items. To achieve this, recommendation engines first build a predictive model, by discovering itemsets or item sequences with high support among users. Recommendations are subsequently generated by matching new transaction patterns to the predictive model. Most current approaches in web personalization consider that a web site consists of a finite number of web pages and build their predictive models based on this assumption [10]. The Web, however, is a continuously evolving environment, and this assumption does no longer hold. Social networking structures, such as blog aggregators, and news portals are typical examples of this situation since their content is updated on a regular basis. As a result, the traditional, usage-based approach that takes as input the navigation paths recorded on the web page level is not as effective. Since most predictive models are based on frequent itemsets, the more recent a page is, the more difficult it is to become part of the recommendation set; at the same time, such pages are more likely to be of interest for the average user. This problem can be addressed by generalizing the page-level navigation patterns to a higher, aggregate level [3, 9].

In this work, we address the aforementioned problem by modifying and combining two algorithms that have been proposed in different contexts. The first algorithm, FP-Growth [5], considers a database of user transactions that comprise one or more unordered items (itemsets) and a minimum support threshold. The algorithm processes the transaction database and mines the complete set of frequent itemsets (whose frequency surpasses the threshold). FP-Growth considers the support of each item in the set to be equal to one. We extend the algorithm so that it assigns different weights to every item in the set depending on its importance in the transaction. The algorithm considers no relation between items in the database, but this is not the case in the web, where items in a web site are (conceptually) hierarchically organized. This characteristic is tackled by the second algorithm, GP-Close [6, 7], that was proposed independently from FP-Growth. GP-Close considers a hierarchical organization of all items in the transaction database and uses this information to produce generalized patterns. The two algorithms are very efficient and solve many of the problems of pattern mining, such as the costly generation of candidate sets and the over-generalization of rules.

In this paper, we combine the forces of the two algorithms in one efficient generalized pattern mining algorithm, which: a) extends the main structure of FP-Growth, the FP-Tree, to include weight information about items, thus producing a weighted FP-Tree (*WFP-Tree*) and, b) addresses the problem of continuously updated content by using the WFP-Tree and the taxonomic information about a web site's content as input to the GP-Close algorithm, and generates generalized recommendations. We experimentally evaluate our approach using web log data and content collected from a newspaper's web site.

The paper is organized as follows. First, we provide an overview of the related research in the area of pattern and association rule mining, as well as in the area of personalizing news sites. We briefly describe the fundamentals of the FP-Growth and GP-Close algorithms, and we present the details of the FGP algorithm in Sections 3 and 4 respectively. In Section 5, we discuss a proof-of-concept implementation and present preliminary experimental results. We conclude with our plans for future work in Section 6.

2. RELATED WORK

Numerous approaches exist that address the problem of personalizing a web site. An extensive overview can be found in [10]. Here we overview those that generalize the predicted patterns using a hierarchy.

The problem with sites such as blogs or news portals, is that their content is continuously updated. Moreover, in the case of blog aggregators we have less control on the tags assigned to each item. Since they do not belong to a hierarchy we need to put extra effort to assign them to a hierarchy node (i.e. using semantic relatedness between the tags and the node [13]). Some approaches are based on the preference information explicitly provided by the users [2, 4]. However, users' interests change from time to time. In the existence of this concept-drift issue, either web users should continuously update their preferences, or the system will eventually fail to present useful, personalized recommendations. We can see that this is a situation analogous to the cold-start problem, that appears when a system should make predictions in the absence of any transaction history. The cold-start problem has been addressed mainly in the context of collaborative filtering systems [8, 12], by creating hybrid recommender systems that take into account both the content of the site and the user ratings or profiles. When there are is not adequate user-based information, similarities between the content can be used to make predictions. The idea of integrating the content in the recommendation process has also been addressed by generalizing the page-level navigation patterns to a higher, aggregate level, with the aid of a topic hierarchy. In a previous work, we have proposed the mapping of all user sessions to the topics of a hierarchy [3]. Those generalized sessions were then used as input to the Apriori algorithm [1], in order to generate category-based recommendations. In [11], the authors proposed a similar framework for semantic web sites, where the content was annotated using an ontology. This framework focused on web mining instead of personalization tasks. In [9] an approach focusing on recommending academic research papers was proposed. The authors mapped the user profiles as well as the research papers to ontology terms, and input those data in a collaborative filtering recommender. Considering the shortcomings of this technique, which are lack of scalability and data sparsity [10], we opted for an association rule mining algorithm. As compared to Apriori or its extensions, AprioriTid and AprioriHybrid [1], the FP-Growth algorithm is more efficient in that it does not generate candidate itemsets, but rather adopts a pattern-fragment growth method. Moreover, we use the topic hierarchy as an inherent component of our algorithm, and adapt the GP-Close mechanism in order to produce generalized recommendations. It is important to note that, compared to previous techniques our recommendations include a combination of pages and page categories.

3. FP-GROWTH AND GP-CLOSE

3.1. The FP-Growth algorithm

The details of the FP-Growth algorithm can be found in the related bibliography [5]. In the following we present its basic steps using a running example. This same example is employed in order to demonstrate the differences between FP-Growth and our algorithm, FGP.

In the first step FP-Growth scans the transaction database, finds all frequent items (minimum support is 3 in our example) and orders them in descending frequency order. In a second database scan, the FP-tree is constructed. Each transaction is mapped to a path in the FP-tree. For the items already in the tree, the count of the respective nodes in the path is updated, whereas new nodes are added for the remaining items. For items belonging to more than one frequent itemsets, all their appearances in the tree are linked. An index table containing all frequent items sorted in descending global frequency order, points to the first appearance of each item in the FP-tree. The FP-Tree resulting from the transaction database of Table 1 is shown in Figure 1.

TID	Itemset	Ordered frequent items (min freq=3)
100	f, a, c, a, d, g, i, a, m, c, p	f, c, a, m, p
200	a, b, c, f, c, l, a, m, o	f, c, a, b, m
300	b, f, h, j, o, f	f, b
400	b, c, k, s, p, c, b	c, b, p
500	a, c, f, c, e, l, f, p, m, n, a	f, c, a, m, p

Table 1. A sample transaction database.

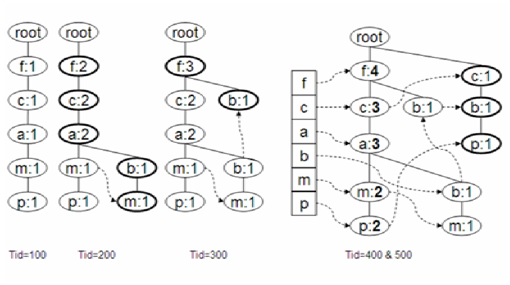

Figure 1. The steps of constructing an FP-Tree.

As proven in [5], the FP-Tree is adequate for mining frequent patterns and can replace the database. In order to compute the support of a k-itemset, FP-Growth starts scanning the tree for the less frequent items in the set. The items in the path from the root to the item under examination form the conditional pattern base of the item and their support equals the support of the item under examination (count adjustment). Table 2 contains the conditional pattern base for the FP-Tree in Figure 1.

Item	Conditional pattern base	Conditional FP-Tree
p	{fcam:2, cb:1}	{c:3}|p
m	{fca:2, fcab:1}	{f:3, c:3, a:3}|m
b	{fca:1, f:1, c:1}	{}
a	{fc:3}	{f:3, c:3}|a
c	{f:3}	{f:3}|c
f	{}	{}

Table2. The conditional pattern base and FP-tree.

3.2. The GP-Close algorithm

The GP-Close algorithm takes as input a transaction database *DB* and a taxonomy *T*, containing all items of *DB*. Using a minimum support threshold, it generates a tree *GT* that contains all the generalized frequent item-sets. Children of a node in the GT expand their parent item-set by adding one item.

The first step of the algorithm is to locate all frequent 1-itemsets and generate all their frequent generalizations by looking up to *T*. After sorting them in a support increasing manner, it gradually expands them to *n*-itemsets, by combining smaller sets and updating support count. Two pruning techniques prevent from exploring unnecessary combinations: the Subtree pruning and the Child pruning. The details of the algorithm and an explanation of the pruning techniques are available in [6].

4. THE FGP ALGORITHM

Consider that all items in the transaction database of Table 1 are articles in a news site and that the taxonomy of topics depicted in Figure 2 exists for this site (numbers correspond to topic ids, and letters to article ids). For simplicity, we consider that each article belongs to a single topic.

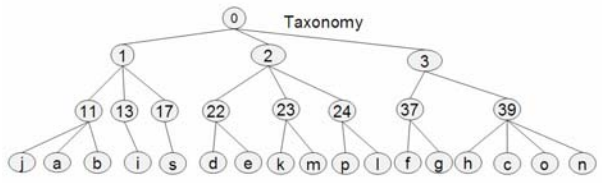

Figure 2. The taxonomy of items

4.1. Pre-processing: item weighting

We should point out that the information we store in the FP-Tree differs from that of the original implementation. In the original paper [5], each transaction identifier (TID) stores only one occurrence for each node. However, in the case of web log files, a user might visit a Web page more than once during a session. Repetitiveness signifies the importance of a page for a specific user, thus the input format is modified to include <*pageID, weight, support*> triplets, instead of merely pageID information.

Although a page's importance in a session depends on the number of repetitive visits, its importance in the whole database is related to the number of distinct sessions it appears in. Thus, analogous to term weighting in document collections (tf/idf), we consider the weight of a page in a session to be the number of its appearances in the session divided by the total number of page hits in the session (page frequency) and the support of a page to be the number of sessions that contain this page (inverse session frequency).

TID	Session items (PID, hits)	Hits/session
100	(a,3), (c,2), (f,1), (d,1), (g,1), (i,1), (m,1), (p,1)	11
200	(a,2), (c,2), (b,1), (f,1), (l,1), (m,1), (o,1)	9
300	(f,2), (b,1), (h,1), (j,1), (o,1)	6
400	(b,2), (c,2), (k,1), (s,1), (p,1)	7
500	(a,2), (f,2), (c,2), (e,1), (l,1), (p,1), (m,1), (n,1)	11

Table3. The web log entries grouped by session

The result of this processing for Table 1 is depicted in Table 3, which is consequently mapped to the WFP-Tree.

4.2.The FGP Algorithm

The FGP algorithm takes as input a transaction database (as in Table 3) and a hierarchy (as in Figure 2) and constructs a set of generalized association rules as follows:

1) Scans the transaction database and constructs the WFP-Tree
2) Finds frequent 1-itemsets using the WFP-Tree
3) Creates frequent generalized 1-itemsets using the hierarchy
   a) Sorts 1-itemsets in increasing support order
   b) Prunes Children: While creating the generalization tree prunes 1-item generalizations that have support equal to a frequent 1-itemset already in the tree
4) Combines 1-itemsets to generate the complete generalized itemsets tree.
   a) Prunes subtrees: If a n-itemset A can be subsumed by an identified k-itemset B already in the tree with n$\subset$k and support(A)<=support(B) then A and its corresponding subtree is pruned.

In what follows, we use the running example of Section 3 to demonstrate the various steps of the proposed algorithm.

4.2.1 Construction of the WFP-Tree

For constructing the WFP-Tree we parse the transaction database and calculate the weight of each individual page in a transaction. We aggregate the weights of the remaining page ids and store a reference to the header table. Transactions are stored in decreasing weight order. The resulting WFP-Tree for the database in Table 3 is depicted in Figure 3 and is used instead of the transaction database in the remaining steps of the algorithm.

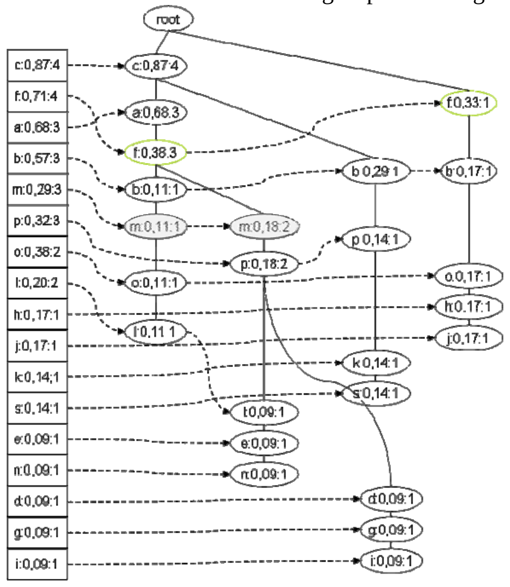

Figure 3. The Weighted FP-Tree

4.2.2 Find frequent 1-itemsets and their generalizations

The header table, which accompanies the WFP-Tree contains references to every page in the tree. We use this table and the taxonomic information presented in Figure 2, in order to find frequent 1-itemsets and to produce the corresponding frequent generalized 1-itemsets. These itemsets are, in essence, the frequently visited categories in the database.

Since categories correspond to more than one page, in order to find the total weight for each category (internal node in the taxonomy tree) we find all the corresponding pages (leaf nodes) in the taxonomy tree. We then process the index file, from bottom to top, in order to locate all the appearances of the leaf nodes in the WFP-Tree and sum their weights.

For computing the support of a topic (i.e. the number of transactions that contain at least one page from this topic), we examine all appearances of the corresponding pages in the WFP-Tree. Transactions that contain many pages from the same topic, are counted only once in the support of this topic.

For example, the support for category 11 is computed based on pages j, b and a. First we aggregate the appearances of j (1), which is lower in the index table, then of b (1+1 + 1-1, due to j) and consequently those of a (3-1 since b has been added). The total support for category 11 is consequently 5, which corresponds to the number of transactions that contain at least one of {j, b, a}. The weight of 11 is 1.42, which is the sum of the weights of j, b and a.

4.2.3 Prune 1-itemset generalizations

In this step we prune 1-itemsets and consequently their generalizations, when they do not have high support (e.g. support < 3 in our example).

Furthermore, in order to avoid the combinatorial explosion of the GP-Close when it searches for all frequent *n*-itemsets, we prune those frequent 1-item generalizations that have the same support as their specializations. For example, the support of category 37, comprising pages f and g is 4, which equals to the support of f. As a result the generalization of 37 is pruned from the final tree and so do all the combinations of 37.

In order to prune the frequent 1-item generalizations we sort all frequent 1-itemsets in increasing support order. If a generalization has the same support with its specialization, then it is pruned from the tree. The first level of the tree containing the frequent generalized 1-itemsets appears in Figure 4.

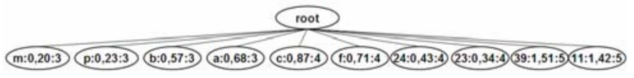

Figure 4. Frequent generalized 1-itemsets

4.2.4 Find frequent k-itemsets

We gradually combine the frequent 1-itemsets to produce larger sets. We compute their support and weight and prune sets that do not meet the minimum support requirements. The support for the itemset *K* is computed over the WFP-Tree as follows:

Suppose that L_z is the set of all leaf nodes for item *z*. Of course, if *z* is a page then $L_z=\{z\}$.

```
1.construct $LS=\{L_z\}:\forall z \in K$ support$_K$=0
2.for $L_1 \in LS$ , the first set of pages in LS
3.$\forall i \in L_1$ find $i_{ALL}$ : all appearances of $i$ in WFP-Tree
4. $\forall i_x \in i_{ALL}$ if contain(subnodes(i$_x$) ,LS-L$_1$)
           then support$_K$=support$_K$+supportlast
```

where the method `contain()` parses the list of subnodes of i_x until at least a page from all the sets in (LS-L_1) is found, and `supportlast` is the support of the last page checked. If we have reached the end of a subnodes list and we have not found a page for every set then `supportlast=0`.

For example, if *K*={f, 24} then *LS*={{f},{p,l}}. We check all appearances of f and search for either p or l in the sub node lists. The support for *K* is 1 (the support of left shaded m in figure 3) plus 2 (the support of the right shaded m in figure 3) plus 0 (the rightmost f does not contain p or l in its node list). A support of 3 is above the minimum threshold in our example, so {f,24} is a frequent 2-itemset. The weight of this itemset is the aggregate of the weights of all WFP-Tree nodes involved in the support counting (0.38+0.11+0.11+0.09+0.09=0.88).

4.2.5 Prune redundant subtrees

It is obvious that certain combinations will be pruned due to insufficient support. For example, a scan in the WFP-Tree gives to {m,p} a support of 2, which is unacceptable. Thus, {m,p} and its subtree are directly pruned. All the 2-itemsets generated for {m} are listed in Figure 5.

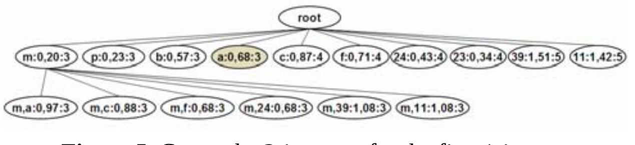

Figure 5. Create the 2-itemsets for the first 1-itemset

A second pruning strategy is applied in this step. According to this, when a *k*-itemset has equal support to a *(k+1)*-itemset and is a subset of this itemset then it is a subsumed one and must be pruned. For example, the shaded node a in figure 5 is pruned. This strategy further reduces the combinations to check in the next expansion step.

The complete expansion of the first *1*-itemset results in pruning most of the *n*-itemsets created (*n*>1). Figure 6 illustrates the result of this expansion, where all shaded nodes are pruned. Expansion continues with the remaining *1*-itemsets.

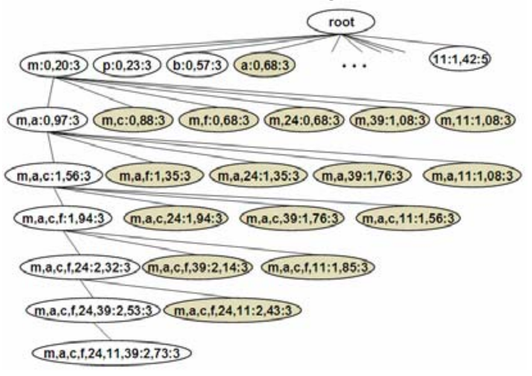

Figure 6. Expansion of the first 1-itemset and subtree pruning

When the tree of sets cannot be further expanded, each node in the tree is exported as a *frequent k-itemset*, which can be used as a rule for recommendation.

4.2.6 Generate recommendations

The frequent *k*-itemsets are subsequently used to generate recommendations for an active user *A*. Given that *A*'s navigation history includes (*k-r*) items from a specific *k*-itemset, the system selects and recommends the remaining items. *r* is a parameter of the system and can take integer values between 1 and *k*. In the case that the predicted item is a category, the system can provide a top-n list of the pages that belong to this category. The criteria for the selection can be the page popularity, its recency, etc.

5.EXPERIMENTAL EVALUATION

5.1 Performance testing

In order to evaluate the performance of the FGP algorithm we test it over the web log files of a news site (www.reporter.gr), collected over a 31 days' period (August 2006). We preprocess the log files in order to clean them from noisy entries (bot entries, invalid requests etc.) and then sessionize them. The log file of a day takes the form of the transaction database presented in Table 3. Each page in the web site belongs to a topic and the hierarchy of topics is used as input to our algorithm.

In average users perform 8708 article hits per day in 882 sessions. The average session comprises of 8.5 pages. FGP algorithm takes 17 seconds in average to process the log file of a day and produces 56 generalized frequent *k*-itemsets in average (*k*>=2). The number of itemsets is bigger compared to those produced by FP-Growth (7 in average for the same transaction set). This shows that FGP is able to generate more itemsets. Moreover, the two pruning strategies avoid redundancies and accelerate the tree creation.

5.2 Validity of results

The output of the FGP algorithm is a set of frequent *k*-itemsets, each one associated with a weight and a support score. A recommendation engine can use these frequent *k*-itemsets against web usage patterns: when a user's pattern matches the (*k-1*) items in the set, then the *k*-th item is suggested to the user, as a recommended hyperlink. The recommendation is considered successful if the user clicks on the hyperlink.

We measure the accuracy of the recommendations generated by FGP as follows: we produce frequent *k*-itemsets by applying FGP to the log file of a certain day and evaluate the rules against the web log file of the day after that. We repeat the same process for every pair of consecutive days and find the average values, performing in essence a 30-fold cross-validation. We validate the itemsets produced from one day's logs only against the logs of the subsequent day, since the life of article ids in the logs is short and rules containing solely article ids will have limited support.

In the experiments, we do not use itemsets' support and weight information when counting for sessions matching an itemset.

We define the *session coverage* (*SC*) of a set of rules (frequent *k*-itemsets) measured against a set of sessions as the number of sessions that match at least one rule in the set divided by the total number of sessions, as shown in formula (1).

$$SC = \frac{validSessions}{allSessions} \quad (1)$$

The results of our experiments are depicted in Figure 7. The horizontal axis corresponds to the day used for generating the frequent *k*-itemsets and the vertical axis shows the percentage of sessions that match at least one rule (session coverage). The results in Figure 7 show that the coverage of the generalized itemsets is larger than that of page-level ones. The average coverage for generalized itemsets produced by FGP is almost 20% (thick line in the graph), and it lowers into 12% when page-level itemsets are only used (thin line).

For those sessions that match to at least one rule we count the total number of rules being matched and present the average values per day in the graph of Figure 8. Values are strongly related to the size of the recommendation set since matching more rules means providing more useful recommendations to the end-users. The average number of rules, produced by FGP, that match a session for the complete dataset is approximately 12, whereas the value for FP-Growth is 5.

6 CONCLUSIONS & FUTURE WORK

In this paper, we presented the FGP algorithm, which takes as input a database of transactions comprising items in a hierarchy and the hierarchy of items, and produces the set of frequent k-itemsets comprising items and/or categories from the hierarchy. The set consists of all itemsets above a minimum support threshold and their generalizations but omits redundant generalizations. In the current implementation we combined two state of the art algorithms: FP-Growth for frequent itemset creation and GP-Close for itemset generalization and pruning of redundancies. The performance evaluation of FGP proved that it is fast and produces many useful itemsets, while avoiding redundancies.

An extensive evaluation of the algorithm, against other similar approaches, using more benchmark data sets is in our next plans. We are currently working on extending the algorithm to work with multiple category assignments for each item and for more complex hierarchies. This is typically the case of blog aggregator services, where content is provided by different authors and tagged using a variety of tags. We also design a user-based evaluation by implementing a recommendation engine for a news site which employs our algorithm.

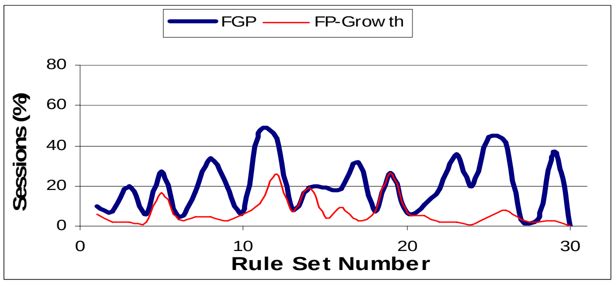

Figure 7. Session coverage

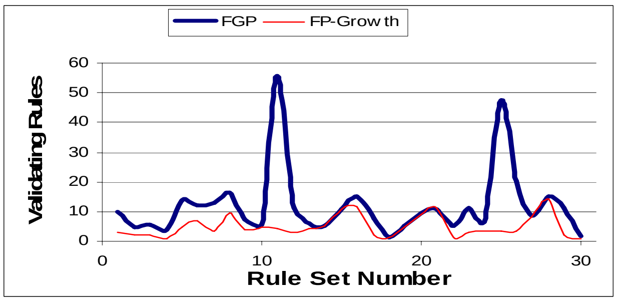

Figure 8. Valid itemsets per session

REFERENCES

[1] R. Agrawal, R. Srikant, Fast Algorithms for Mining Association Rules, *VLDB*(1994)

[2] E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas, I. Vlahavas, PersoNews: A Personalized News Reader Enhanced by Machine Learning and Semantic Filtering, *ODBASE* (2006)

[3] M. Eirinaki, M. Vazirgiannis, I. Varlamis, Sewep: Using site semantic and a Taxonomy to enhance the web personalization process, *KDD*(2003)

[4] E. Gabrilovich, S. Dumais, E. Horvitz, Newsjunkie: Providing Personalized Newsfeeds via Analysis of Information Novelty, *WWW* (2004)

[5] J. Han, J.Pei, Y. Yin,R. Mao,Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Min. Knowl. Discov.* 8, 1 (2004)

[6] T. Jiang, A.H. Tan, Mining RDF Metadata for Generalized Association Rules, *Database and Expert Systems Applications*, LNCS Vol. 4080 (2006)

[7] T. Jiang, A. Tan, K. Wang, Mining Generalized Associations of Semantic Relations from Textual Web Content, *TKDE*, 19(2), (2007)

[8] X. Lam, T. Vu, T. Le, A. Duong, Addressing cold-start problem in recommendation systems, *ICUIMC* (2008)

[9]S. Middleton, N. Shadbolt,D. De Roure, Ontological User Profiling in Recommender Systems, *TOIS*, 22(1) (2004)

[10] B. Mobasher, Data Mining for Personalization. *In The Adaptive Web: Methods and Strategies of Web Personalization*, LNCS Vol. 4321(2007)

[11] D. Oberle, B. Berendt, A. Hotho, J. Gonzalez, *Conceptual User Tracking*, *AWIC* (2003)

[12] A. Schein, A. Popescul, H. Lyle, Methods and Metrics for Cold-Start Recommendations, *SIGIR* (2002)

[13] G. Tsatsaronis, I. Varlamis, M. Vazirgiannis, "Word Sense Disambiguation with Semantic Networks", to appear in the *11th Int. Conf. on Text Speech and Dialog (2008).*