Mining frequent generalized patterns for Web personalization in the presence of topic taxonomies

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

Iraklis Varlamis
Harokopio University of Athens, Department of Informatics and Telematics, Athens, Greece
varlamis@hua.gr

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

Abstract. The Web is a continuously evolving environment, since its content is updated on a regular basis. As a result, the traditional usage-based approach to generate recommendations that takes as input the navigation paths recorded on the web page level, is not as effective. Moreover, most of the content available online is either explicitly or implicitly characterized by a set of categories organized in a taxonomy. Using this information, the page-level navigation patterns can be generalized to a higher, aggregate level. In this direction, we present the Frequent Generalized Pattern (FGP) algorithm. FGP takes as input the transaction data and a hierarchy of categories that characterize them, and produces generalized association rules that contain transaction items and/or item categories. The algorithm can be applied to the log files of a typical web site; however, it can be more helpful in a Web 2.0 application where content is semantically annotated. Given a transaction database and a hierarchical organization of the browsed items, the algorithm generates frequent itemsets comprising of both web pages and categories. The results can be used to generate association rules and subsequently recommendations for the users. We also introduce FGP+, which is an extension of the FPG algorithm and can take as input more complex structures than taxonomies. This extension addresses the unique characteristics of Web 2.0 sites such as feed aggregators or digital libraries. In such sites, the topic or tag information attached in a page hardly forms a taxonomy. We experimentally evaluate the proposed algorithms 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 (Mobasher, 2007). The Web, however, is a continuously evolving environment and this assumption does no longer hold. News portals are typical examples of this situation since they update their content 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 (Eirinaki et. al. 2003; Mobasher, 2007).

In this work, we present the FGP algorithm, to address the aforementioned problem. The FGP algorithm is in essence the result of the modification and combination of two algorithms that have been proposed in different contexts. The first one, FP-Growth (Han et. al. 2004), is given 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. In this work, we extend the algorithm so that it assigns different weights to every item in the set depending on its importance in the transaction. We should note that the FP-Growth algorithm does not consider any relation between items in the database. This, however, is not the case in the web, where items in a web site are (conceptually) hierarchically organized. This intrinsic characteristic of the web can be tackled by the second algorithm, GP-Close (Jiang and Tan, 2006; Jiang et. al., 2007). 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.

The FGP algorithm works efficiently in the case of web sites that have a well-defined underlying hierarchy of topics, such as news portals. Many Web 2.0 sites, however, present a more complex underlying structure. For instance, feed aggregators summarize and present content that is collected from multiple sources. In such sites, the content is not necessarily categorized into predefined categories (Inform 2007), being described by user-defined tags instead. This collaborative tagging process results into folksonomies (Voss 2007; P. Heymann and H. Garcia-Molina 2006) that differentiate from the traditional top-down taxonomies. The more complex structure of folksonomies, the use of plurals, the synonym polysemy and specificity of tagging raise new issues for the recommendation engines. In this context, we propose an extension of the FGP algorithm, named FGP+ that takes a more composite topic hierarchy as input, and supports multiple category assignments per topic.

In brief, the contributions of our work are outlined in what follows:

- We modify and combine the forces of FP-Growth and GP-Close in one efficient generalized pattern mining algorithm, named FGP which:
  - extends the frequent-pattern tree, the main structure of the FP-Growth algorithm, to include weight information about items, thus producing a weighted FP-Tree (*WFP-Tree*)
  - addresses the problem of continuously updated content by using the WFP-Tree and the taxonomic information related to the web site's content as input to GP-Close, and generates generalized recommendations
- We present an extension of FGP, named FGP+ that supports an extended taxonomy of topic categories and/or multiple category assignments per item. The extended algorithm directly addresses the special characteristics of social networking applications.
- We experimentally evaluate our approach using data 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. We motivate the need for an extension of the basic algorithm, and convey our solution in Section 5. In Section 6, we discuss our implementation and present the experimental results. We conclude with our plans for future work in Section 7.

2. RELATED WORK

There exist numerous approaches that address the problem of personalizing a web site. An extensive overview can be found in (Mobasher, 2007). In this paper, we overview those that are more similar to ours with regards to: a) the personalization of news sites and b) the abstraction of the generated patterns using a hierarchy.

Both research projects (Antonellis et. al., 2006; Banos et. al., 2006; Katakis et. al. 2008, Gabrilovich et. al., 2004) and commercial sites, such as Spotback (http://spotback.com) and Topix (http://www.topix.net), have attempted to address the need of personalizing the content of a news site according to users' preferences. Most of those approaches, however, are based on the preference information explicitly provided by the users. However, users' interests change from time to time. In the existence of this concept-drift issue (Tsymbal, 2004; Katakis et. al., 2008), 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, which 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 (Lam et. al., 2008; Schein et. al., 2002), by creating hybrid recommender systems that take into account both the content of the site and the user ratings or profiles. When there 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 (Eirinaki et. al., 2003). Those generalized sessions were then used as input to the Apriori algorithm (Agrawal and Srikant, 1994), in order to generate category-based recommendations. Oberle et. al. (2003) 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 (Middleton et. al., 2004) 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 used those data as input to a collaborative filtering recommender.

Considering several shortcomings of collaborative filtering, such as data sparsity and lack of scalability (Mobasher, 2007), we opted for an association rule mining algorithm as the core of the personalization process. Compared to Apriori or its extensions, namely, AprioriTid and AprioriHybrid (Agrawal and Srikant, 1994), the FP-Growth algorithm is more efficient in that it does not generate candidate itemsets, but adopts a pattern-fragment growth method instead. 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 taking as input hierarchical, as well as complex taxonomies. In contrast to existing techniques that recommend either pages or categories, FGP and its extension generate frequent itemsets comprising of both of them. Thus, it supports the generation of recommendations that 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 (Han et. al. 2004). In what follows we present an overview of the algorithm 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.

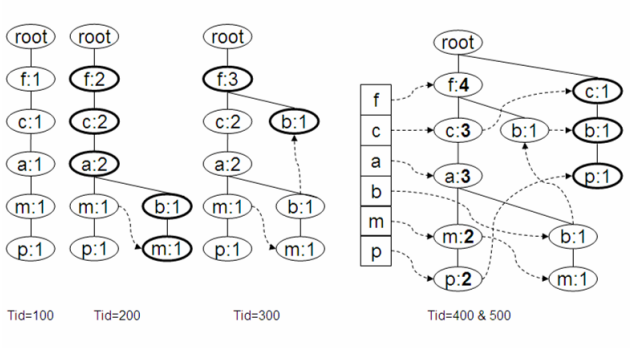

Figure 1. The steps of constructing an FP-Tree.

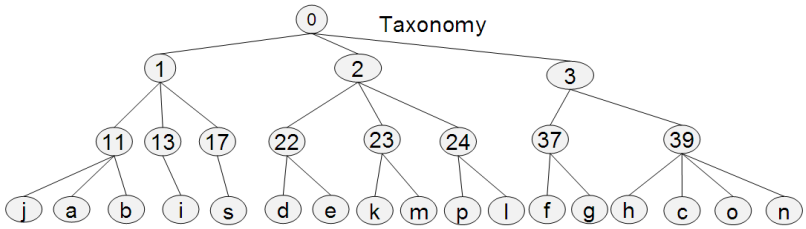

Figure 2. The taxonomy of items

As proven by Han et al. (2004), the FP-Tree is adequate for mining frequent patterns and can replace the transaction database. In order to compute the support of a k-itemset, FP-Growth scans the tree in order to find 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 of 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	{}	{}

Table 2. 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 called *closure enumeration tree* (CET) that contains all the generalized frequent itemsets. The children of a node in the CET expand their parent itemset 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-Closure pruning. The details of the algorithm and an explanation of the pruning techniques are available in (Jiang and Tan, 2006).

4. The FGP algorithm

In order to demonstrate how the FGP algorithm functions we use the running example introduced in Section 3. 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). Without loss of generality we assume that each article belongs to a single topic. We show how the algorithm is extended to handle multiple category assignments in Section 5.

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 (Han et al. 2004), 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 database is related to the number of distinct sessions it appears in. Thus, analogously to term weighting in document collections (i.e. 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)	Total 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

Table 3. 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 taxonomy (as in Figure 2) and constructs a set of generalized association rules as follows:

1) Scan the transaction database and construct the WFP-Tree
2) Find frequent 1-itemsets using the WFP-Tree
3) Create frequent generalized 1-itemsets using the hierarchy
   a) Sort 1-itemsets in increasing support order
   b) Prune Children: While creating the generalization tree prune 1-item generalizations that have support equal to a frequent 1-itemset already in the tree
4) Combine 1-itemsets to generate the complete generalized itemsets tree.
   a) Prune 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 demonstrate the implementation of the FGP algorithm on the WFP-Tree and put light on the details of support counting, tree generation and pruning, using the running example introduced before.

4.2.1 Construction of the WFP-Tree

In order to construct the *WFP-Tree*, the transaction database is parsed and the support and weight for each individual page in a transaction is calculated. The algorithm then aggregates the weights of the remaining page ids and stores a reference to the header table. The transactions are stored in decreasing weight order. The final result for the database in Table 3 is depicted in Figure 3. The *WFP-Tree* can replace the original transaction database in the remaining steps of the algorithm.

4.2.2 Discovering frequent 1-itemsets and their generalizations

The header table, which accompanies the *WFP-Tree*, contains a reference to every page in the tree. This table, along with the taxonomy, is used as input in order to find *frequent 1-itemsets* and 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), FGP finds all the corresponding pages (leaf nodes) in the taxonomy tree. It subsequently processes 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.

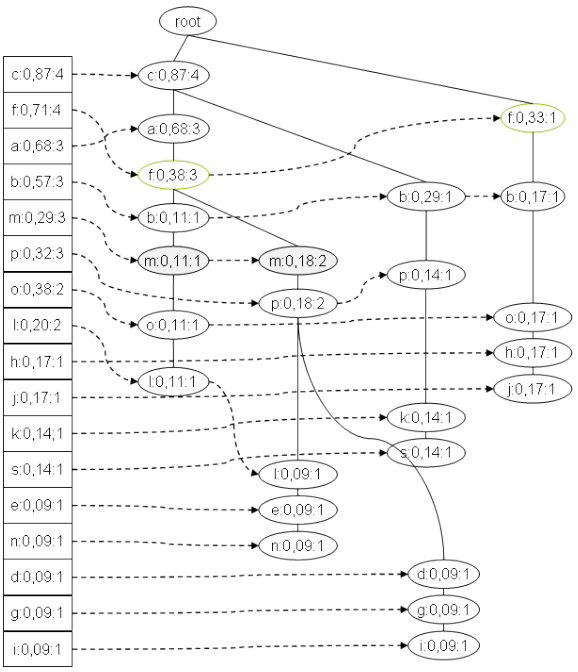

Figure 3. The Weighted FP-Tree

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

For example, the support for category 11 is computed based on pages j, b and a. First the algorithm aggregates the appearances of j (1), which is lower in the header 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 Pruning 1-itemset generalizations

In this step all 1-itemsets and their generalizations that do not have high support (e.g. support < 3 in our example) are being pruned.

Furthermore, in order to avoid the combinatorial explosion of the GP-Close when it searches for all frequent *n*-itemsets, FGP also prunes those frequent 1-item generalizations that have the same support as one of their specializations. For example, the support of category 37, comprising of 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 the algorithm sorts all frequent 1-itemsets in increasing support order. If a generalization has the same support with one of its specializations, then it is pruned from the closure enumeration tree. The first level of the tree containing the frequent generalized 1-itemsets is shown in Figure 4.

Figure 4. Frequent generalized 1-itemsets

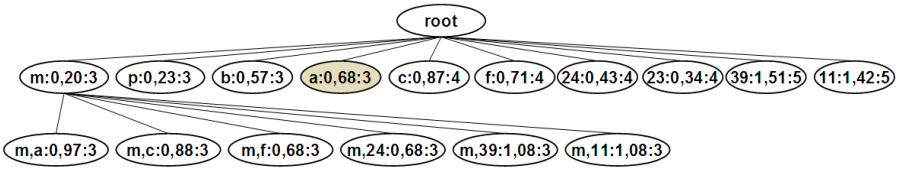

Figure 5. Creating the 2-itemsets for the first 1-itemset

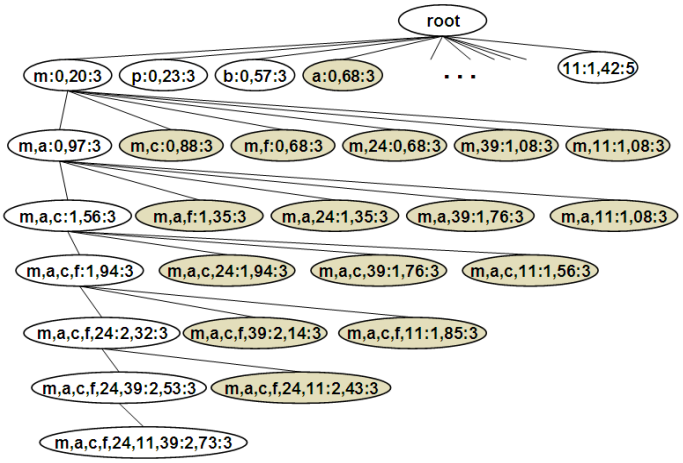

Figure 6. Expansion of the first 1-itemset and subtree pruning

4.2.4 Discovering frequent k-itemsets

FGP incrementally combines the frequent 1-itemsets to generate larger sets. After computing their support and weight, the sets that do not meet the minimum support requirements are pruned. 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. If z is a page then $L_z=\{z\}$.

1. construct $LS=\{L_z\}:\forall z\in K$ $\text{support}_K=0$
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 $\text{support}_K=\text{support}_K+\text{supportlast}$

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 the end of a subnodes list has been reached without finding a page for every set then `supportlast=0`.

To provide an example, we calculate the support of K={f, 24}. We first construct 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 l in Figure 3) + 2 (the support of the leftmost occurrence of p in the WFP-tree) + 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, which means f and l in the leftmost branch (0.38+0.11) and p and l (0.18+0.09) in the second leftmost sub-branch (which shares f as an ancestor). The total weight for K is 0.76.

4.2.5 Pruning redundant subtrees

It is obvious that certain combinations will be pruned due to insufficient support. For example, a scan in the WFP-Tree of Figure 3 gives to {m,p} a support of 2, which is below the specified threshold. Thus, {m,p} and its subtree are directly pruned. All the 2-itemsets generated from {m} are listed in Figure 5.

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 can be pruned. For example, the shaded node a in Figure 5 is pruned. This strategy further reduces the possible combinations than need to be checked in the next expansion step.

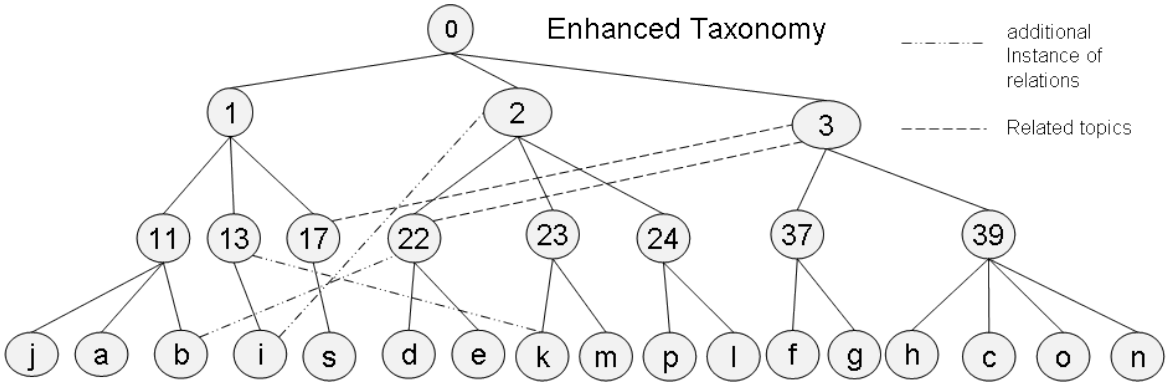

Figure 7. The modified taxonomy of Figure 2, in which it is allowed for a node to have more than one parents.

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.

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 to generate recommendations. For example, similarly to association rules-based recommendations, the recommender system can find the *k-itemset* that is more similar to the current user's navigation, by comparing the *k-1* items (pages or categories) to the current visit. The system can then recommend the *k*-th item, if it is a page (e.g. a news article), or the most popular/recent pages belonging to the *k*-th item, if it is a category.

5. The FGP+ algorithm

The FGP algorithm can be used for generating recommendations in sites where the categories are organized in a hierarchy and each page is characterized by a single category. However, in the case of feed aggregators or digital library mediators, the underlying connectivity of categories is more complex (i.e. taxonomies are enhanced with "related category" links and categories have multiple direct ancestors) and pages (or items in general) are assigned to multiple categories. In what follows, we propose FGP+, which is an extension of the FGP algorithm that addresses the aforementioned issues.

5.1 Major differences between a newspaper site and an aggregator

In the newspaper world, things are quite simple: there is a collection of articles, each one belonging to a single category only, and a hierarchy of categories, strictly defined by the administrator so that every node has one parent at most. Nevertheless this is not the case in Web 2.0 sites where both the content and the category "tagging" is user-controlled. Even when the users select from a list of predefined tags to assign to their articles (e.g. blog posts), they may choose more than one per item. Moreover those tags may belong to inner categories of the taxonomy (e.g. page i in Figure 7, which is assigned to categories 2 and 13)

In several cases the tags assigned to items are different but share similar meaning. Such related or synonym categories can be shown in the taxonomy with horizontal relations (e.g. the relation between categories 3 and 17 in Figure 7). This implies that we need a method to map words to their synonyms in the taxonomy.

5.2 Principal extensions to the FGP algorithm

The modifications in the taxonomy do not affect the original transaction database and the structure of the WFP-Tree. The taxonomy tree, however, has to be enhanced with additional relations allowing each node to have more than one parent. Thus a *taxonomy graph* should be created, that will maintain its directed acyclic nature. Additionally, each category can have a set of synonyms associated with it, in order for the mining algorithm to address cases where different users describe the same content using different category tags.

The construction of the extended tree structure and especially the computation of support and weight for a composite itemset, comprising both pages and categories, is not a straightforward task. In fact, even in the case when we consider no "related topic" edges for the taxonomy graph, the latter should be constructed very cautiously so as to take into account multiple parent assignments for the same internal or leaf node.

Moreover, when we further extend our data structure, allowing each node to contain a list of synonyms associated with it, things become really challenging. For example, Figure 8 presents a taxonomy enhanced with synonymous/related categories information. Furthermore, both leaf and inner categories have instances (pages, denoted with letters).

In what follows, we discuss the different situations that need to be handled during the construction of the extended closure enumeration tree and the actions that are taken for the computation of support and weight (the relations between nodes are depicted in the taxonomy graph of Figure 8):

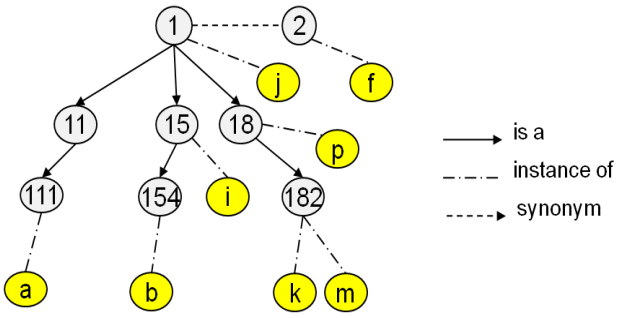

Figure 8. The taxonomy graph now supports a list of synonyms for each node, whereas the instances (words from actual tags) do not necessarily belong to the same level.

- *Compute the support and weight of categories containing items that have a parent/child relationship* (e.g. {18, 182} in Figure 8). In this case we can either, a) ignore the subclass relation in the creation of the closure enumeration tree, i.e. regard the two categories as unrelated, or b) consider that either the parent or the child is supported when an instance of the child category is found (e.g. k). In our implementation, we follow the first alternative and ignore the existence of the child nodes, as far as the parent one is concerned when computing the support and weight. For example, the support and weight computation for the pair {18, 182} of the closure enumeration tree of Figure 8 occur independently for the two nodes, taking the sessions containing p and m, or p and k but not those that contain m and k but not p, into account.
- *Handle nodes having descendant/ancestor relationships* (e.g. {1, 111} in Figure 8), which is in essence a generalization of the parent/child relationship. Thus, we follow the same solution as before. For example, we will not take into consideration the items tagged with 111 during the support and weight computation of category 1.
- *Handle synonym/related relationships* (e.g. nodes {1} and {2} in Figure 8). The corresponding values for support and weight should be aggregated over all items belonging to synonym categories. For example, the values of support and weight for any of the two categories {1} and {2} should be the aggregate of the corresponding values of instances j and f.
- *Handle synonyms in descendant/ancestor relations* (e.g. {2, 182} in Figure 8). We are currently facing the problem of support and weight computation for nodes, which, despite seeming irrelevant, possess an ancestor/descendant relationship indeed. In order to identify such relations, we must first replace each instance of a synonym by its principal term. For example, node {2} should be replaced by {1} which is easily identified as an ancestor of 182.

In order to deal with the utilization of synonym tags by the users (e.g. in blog posts), we can keep a list of synonyms associated with each principal category (for example in Figure 8, {1} is a principal category, whereas {2} corresponds to a synonymous one). This list should be used in order to replace all synonyms by their principal categories, prior to applying FGP+. In this way, the articles belonging to the synonymous category will be considered as members of the principal one, too.

6. EXPERIMENTAL EVALUATION

6.1 Performance testing

In order to evaluate the performance of the FGP and FGP+ algorithms we use the web log files of a news site (www.reporter.gr) collected over a 31 days' period (during August 2006). The log files were cleaned, preprocessed, and sessionized, based on the assumption that the web pages viewed by a user within half an hour belong to the same session. The log files were transformed into a transaction database as the one shown in Table 3. Each page in the web site belongs to a topic and the hierarchy of topics was used as input to our algorithm. Table 4 shows the statistics of our log file set.

Total number of files	31
Avg num of hits per day	8708
Avg num of sessions per day	882
Avg session length (in page hits)	8.5
Avg num of k-item sets per day (FP-Growth)	7
Avg num of generalized k-item sets per day	56

Table 4. Log files processing statistics

When no pruning is used, the time needed for the creation of the Closure Enumeration Tree (CET) is 21.04 seconds and the tree contains 1707 rules on average, against only 17.2 seconds and 281 rules obtained by applying the two pruning techniques (Child-Closure pruning and Subtree pruning) to FGP. This shows that the two pruning strategies avoid redundancies and accelerate the tree creation.

In order to evaluate FGP+, we use the same set of log files, but this time we take into account the tag information assigned by the site owners. Although in this data set the tag information is centrally controlled by the administrators, it has some useful features that allow us to test the extension of the FGP algorithm, since a) multiple tags are assigned per article, and b) the tags correspond to topics in the aforementioned hierarchy, in which it is allowed for a node to have more than one parents. As a result, we are able to use FGP+ and find frequent generalized itemsets comprising of both articles and tags.

6.2 Validity of the results of the FGP algorithm

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. Furthermore, if this element corresponds to a category, the n most recent articles belonging to it are recommended, thus providing a solution to the cold-start problem (Lam et. al., 2008; Schein et. al., 2002). We could even propose the (k-r) items in the pattern, provided that the user has requested the rest r, where r is a system parameter.

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 following day. 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 a 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 our experiments, we do not make use of the support and weight information of itemsets when counting for sessions matching a set of elements.

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 9. The horizontal axis corresponds to the day used for generating the frequent *k*-itemsets, whereas the vertical axis shows the percentage of sessions that match at least one rule (session coverage). The results in Figure 9 show that the coverage of the generalized itemsets is larger than that of the page-level ones. The average coverage for the generalized itemsets produced by FGP is almost 29% (dashed line in Figure 9), and it lowers into 5.4% when page-level itemsets are only used (thin line in Figure 9).

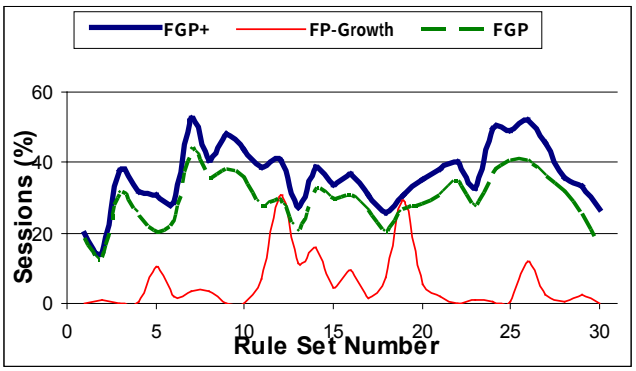

Figure 9. Session coverage (24 hours)

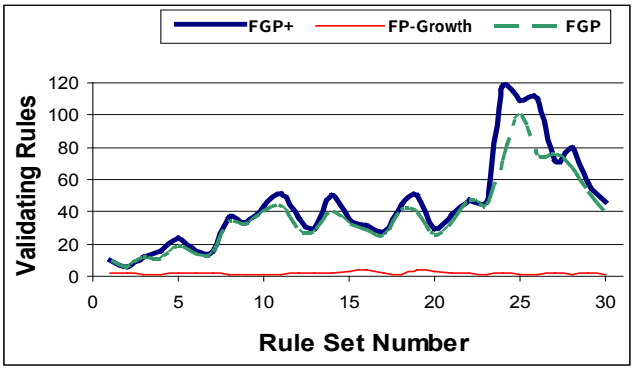

Figure 10. Valid itemsets per session (24 hours)

In a second set of experiments, we count the total number of rules being matched, for those sessions that satisfy at least one rule for both FGP and FP-Growth. The average values per day are shown in Figure 10 (dashed and thin line for FGP and FP-Growth respectively). We should point out that the number of matching rules is strongly related to the size of the recommendation set, since the more rules that are matched, the more recommendations will be provided to the end-users. As shown in Figure 10, the average number of rules, produced by FGP, that match a session for the complete dataset is approximately 37, while the value for FP-Growth is almost 2.

After the initial evaluation, we proceeded in evaluating our algorithm by utilizing the half-day (i.e. 12 hours) log as a training set (i.e. for rule generation) and the next half as a test set (i.e. for evaluation). This approach makes sense in continuously updated sites, such as news sites, where new articles are published every few hours and visitors tend to read the most recent of them. The results, shown in Figures 11 and 12, prove this necessity. This is also an indication that in a recommendation engine, the rule database should be constantly refreshed to capture the readers' shift of interest.

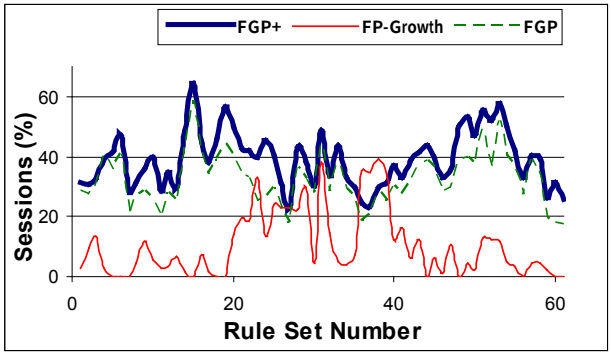

Figure 11. Session coverage (12 hours)

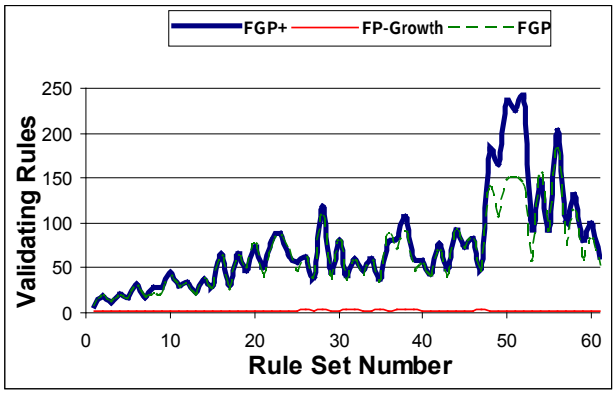

Figure 12. Valid itemsets per session (12 hours)

More specifically, as depicted in Figure 11, the average coverage for the generalized itemsets produced by FGP increases to almost 32,3% (dashed line in the graph), while when page-level itemsets are only used (thin line) the coverage raises but is still smaller (hardly reaches an average of 10.5%). Similarly, as shown in Figure 12, the average number of rules produced by FGP that match at least one session for the complete dataset almost doubles (raises to 63), whereas the value for FP-Growth remains 2.

6.3 Working with enhanced hierarchies

In order to test the ability of FGP+ to work with more complex taxonomies, we used the tag information and repeated the same set of experiments using the 24 and 12 hour log files. The results of FGP+ are illustrated with a thick line in Figures 9 to 12.

Table 5 summarizes the average values for the three techniques. According to these results, the average coverage for the generalized itemsets produced by FGP+ raises to 36.3% when full day logs are used and to 39% when we use half day logs.

Average values		FP-Growth	FPG	FPG+
24 hours	Coverage (%)	5.48	29.18	36.30
	Rules per session	1.82	37.70	44.02
12 hours	Coverage (%)	10.48	32.29	39.02
	Rules per session	1.92	62.82	73.88

Table 5. Summary of average values for the three algorithms

7 CONCLUSIONS & FUTURE WORK

In this paper, we presented the FGP algorithm, which takes as input a database of transactions consisting of items that are organized in a taxonomy, as well as the taxonomy itself, and produces a 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 modified and combined two state-of-the-art algorithms: FP-Growth for frequent itemset creation and GP-Close for itemset generalization and pruning of redundancies. The proposed algorithm, as well as its extension named FGP+, is capable to deal with taxonomies of various levels of complexity, ranging from simple ones (i.e. taxonomy tree of a newspaper site) to more complicated ones (i.e. taxonomy graph of a feed aggregator), allowing each node to have more than one parents. FGP+ also handles multiple category assignment and a list of synonyms for each concept. The performance evaluation of FGP and FGP+ has shown that they produce many useful itemsets, while avoiding redundancies.

An extensive evaluation of FGP+ against more web log data sets is in our next plans. We also plan to perform a user-based evaluation by implementing a recommendation engine on top of the FGP algorithm and use it on a news feed aggregator.

REFERENCES

R. Agrawal and R. Srikant (1994), *Fast Algorithms for Mining Association Rules in Large Databases*. In Proceedings of the 20th international Conference on Very Large Data Bases (September 12 - 15, 1994). J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Very Large Data Bases. Morgan Kaufmann Publishers, San Francisco, CA.

I. Antonellis, C. Bouras and V. Poulopoulos (2006), *Personalized News Categorization Through Scalable Text Classification*, in 'Proc. of the 8th Asia-Pacific Web Conference- Frontiers of WWW Research and Development (APWeb 2006)', Vol. 3841 of Lecture Notes in Computer Science, Springer-Verlag New York, Inc., Harbin, China.

E. Banos, I. Katakis, N. Bassiliades, G. Tsoumakas and I. Vlahavas (2006), *PersoNews: A Personalized News Reader Enhanced by Machine Learning and Semantic Filtering*. 5th International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE 2006), Springer-Verlag, Montpellier, France, 2006

M. Eirinaki, M. Vazirgiannis and I. Varlamis (2003), *SEWeP: using site semantics and a taxonomy to enhance the Web personalization process.* In Proceedings of the Ninth ACM SIGKDD international Conference on Knowledge Discovery and Data Mining (Washington, D.C., August 24 - 27, 2003). KDD '03. ACM, New York, NY, 99-108. DOI= http://doi.acm.org/10.1145/956750.956765

E. Gabrilovich, S. Dumais and E. Horvitz (2004), *Newsjunkie: providing personalized newsfeeds via analysis of information novelty*. In Proceedings of the 13th international Conference on World Wide Web (New York, NY, USA, May 17 - 20, 2004). WWW '04. ACM, New York, NY, 482-490. DOI= http://doi.acm.org/10.1145/988672.988738

J. Han, J.Pei, Y. Yin and R. Mao (2004), *Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*, Data Min. Knowl. Discov. 8,1. DOI= http://dx.doi.org/10.1023/B:DAMI.0000005258.31418.83

P. Heymann and H. Garcia-Molina (2006), *Collaborative creation of communal hierarchical taxonomies in social tagging systems*. Preliminary Technical Report, InfoLab, Stanford, 2006. Avalaible online on 26-9-2008 at: http://heymann.stanford.edu/taghierarchy.html.

T. Jiang and A.H. Tan (2006), *Mining RDF Metadata for Generalized Association Rules: knowledge discovery in the semantic web era.* In Proceedings of the 15th international Conference on World Wide Web (Edinburgh, Scotland, May 23 - 26, 2006). WWW '06. ACM, New York, NY, 951-952. DOI= http://doi.acm.org/10.1145/1135777.1135960

T. Jiang and K. Wang (2007), *Mining Generalized Associations of Semantic Relations from Textual Web Content*. IEEE Trans. on Knowl. and Data Eng. 19, 2 (Feb. 2007), 164-179. DOI= http://dx.doi.org/10.1109/TKDE.2007.36

Inform Inc.(2008), *Inform's Essential Technology Platform*, White paper, Accessed on 26-9-2008 at: http://www.inform.com/contents/pdf/informwhitepaper.pdf

I. Katakis, G. Tsoumakas and I.Vlahavas (2008), *An Ensemble of Classifiers for coping with Recurring Contexts in Data Streams*, Poster at the 18th Europeen Conference on Artificial Intelligence (ECAI 2008), Patras, Greece, July 21-25, 2008

I. Katakis, G. Tsoumakas, E. Banos, N. Bassiliades and I. Vlahavas (2008), *An adaptive personalized news dissemination system*, Journal of Intelligent Information Systems, Springer, DOI - 10.1007/s10844-008-0053-8

X. Lam, T. Vu, T. Le and A. Duong (2008), *Addressing cold-start problem in recommendation systems.* In Proceedings of the 2nd international Conference on Ubiquitous information Management and Communication (Suwon, Korea, January 31 - February 01, 2008). ICUIMC '08. ACM, New York, NY, 208-211. DOI= http://doi.acm.org/10.1145/1352793.1352837

S. Middleton, N. Shadbolt and D. De Roure (2004), *Ontological User Profiling in Recommender Systems*, ACM Transactions on Information Systems 22, 1, 54-88. DOI= http://doi.acm.org/10.1145/963770.963773

B. Mobasher (2007), *Data Mining for Personalization*. In The Adaptive Web: Methods and Strategies of Web Personalization, Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.). Lecture Notes in Computer Science, Vol. 4321, PP. 90-135, Springer, Berlin-Heidelberg, 2007

D. Oberle, B. Berendt, A. Hotho and J. Gonzalez (2003), *Conceptual User Tracking*, In Proceedings of the Atlantic Web Intelligence Conference (AWIC) Madrid, Spain, Springer, Lecture Notes in Computer Science, volume 2663.

A. Schein, A. Popescul and H. Lyle (2002), *Methods and Metric for Cold-Start Recommendations*, In Proceedings of the 25th Annual international ACM SIGIR Conference on Research and Development in information Retrieval, SIGIR '02. ACM, 253-260. DOI= http://doi.acm.org/10.1145/564376.564421

G. Tsatsaronis, I. Varlamis and M. Vazirgiannis (2008), *Word Sense Disambiguation with Semantic Networks*, in Proceedings of the 11th International Conference on Text, Speech and Dialogue, (TSD 2008), 8-12 September 2008, Brno, Czech Republic.

A. Tsymbal (2004), *The problem of concept drift: definitions and related work*, University of Dublin, Technical Report. Available at https://www.cs.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-15.pdf on 26-9-2008

J. Voss (2007). *Tagging, Folksonomy & Co - Renaissance of Manual Indexing?*. Proceedings of the International Symposium of Information Science: 234–254