RP-Tree: Rare Pattern Tree Mining

Sidney Tsang, Yun Sing Koh, and Gillian Dobbie

The University of Auckland
{stsa027,ykoh,gill}@cs.auckland.ac.nz

Abstract. Most association rule mining techniques concentrate on finding frequent rules. However, rare association rules are in some cases more interesting than frequent association rules since rare rules represent unexpected or unknown associations. All current algorithms for rare association rule mining use an Apriori level-wise approach which has computationally expensive candidate generation and pruning steps. We propose RP-Tree, a method for mining a subset of rare association rules using a tree structure, and an information gain component that helps to identify the more interesting association rules. Empirical evaluation using a range of real world datasets shows that RP-Tree itemset and rule generation is more time efficient than modified versions of FP-Growth and ARIMA, and discovers 92-100% of all the interesting rare association rules.

Keywords: Rare Pattern Mining, FP-Growth, Information Gain.

1 Introduction

Association rule mining techniques are used to extract useful information from databases. The set of association rules that can be extracted from a database can be divided using a support threshold into frequent and rare association rules. Both frequent and rare association rules present different information about the database from which they are found, since frequent rules focus on patterns that occur frequently, while rare rules focus on patterns that occur infrequently. In many domains, events that occur frequently may be less interesting than events that occur rarely, since frequent patterns represent the known and expected while rare patterns may represent unexpected or previously unknown associations, which is useful to domain experts. For example, in the area of medicine, the expected, frequent responses to medications are less interesting than exceptional, rare responses which may indicate adverse reactions or drug interactions.

Algorithms such as Apriori [1] can be used to find both frequent and rare association rules, but the latter requires the minimum support threshold to be set to a low value. However this may cause a combinatorial explosion of itemsets as the number of patterns that meet minimum support becomes insurmountable. Given n items, the number of possible itemsets is $2^n - 1$.

There are three possible types of rare itemsets: first, itemsets which consist of rare items only; second, itemsets which consist of both rare and frequent items; and third, itemsets which consist of only frequent items which fall below the minimum support threshold. We refer to itemsets of the first and second types as *rare-item itemsets*. Rare-item itemsets are generally more interesting than itemsets of the third type, which we

A. Cuzzocrea and U. Dayal (Eds.): DaWaK 2011, LNCS 6862, pp. 277–288, 2011.

call *non-rare-item itemsets*. This is because frequent items occur commonly in the database, and there may be many non-rare-item itemsets that do not represent any interesting connection between items since the items only occurred together by chance. Empricial evidence for the claim that rare-item itemsets are more interesting is given in the results in Section 4. For now, we will illustrate this with the following simple example.

Suppose a database of patient symptoms contains the rare itemsets "1:{elevated heart rate, fever, skin bruises, low blood pressure}" and "2:{muscle pain, tinnitus, sneezing, heartburn}", where all items other than "low blood pressure" are frequent items. Itemset 1 is a rare-item itemset, and itemset 2 is a non-rare-item itemset. Itemset 1 contains a subset of the symptoms of sepsis, will produce a rule such as "{elevated heart rate, fever, skin bruises} $\rightarrow$ low blood pressure" that highlights the association between the different three former symptoms with low blood pressure, which is a symptom of severe sepsis. However, rules generated from itemset 2, such as "{muscle pain, tinnitus, heartburn} $\rightarrow$ sneezing" does not give any useful information, since all these symptoms are individually common, and have simply occured together by chance.

The key contribution of the paper is a novel algorithm called RP-Tree that finds rare-item itemsets using a tree structure. Unlike previous level-wise approaches, RP-Tree does not need to generate and test all plausable combinations of rare itemsets, which is more efficient. We empirically show that RP-Tree finds rare itemsets and association rules more effciently than existing algorithms, and identifies 92-100% of rare association rules that meet a confidence and lift threshold. The second contribution of this paper is an extension to RP-Tree that reduces the number of uninteresting association rules generated by excluding items that are poor at predicting the occurrence of rare items. To our knowledge, RP-Tree is the first rare association rule mining algorithm that uses a tree structure.

The paper is organized as follows. In Section 2 we look at previous work in the area of rare association rule mining. In Section 3 we present basic concepts and definitions for rare association rule mining and discuss our novel RP-Tree approach. Section 4 describes the experimental results. Finally, Section 5 concludes the paper.

2 Related Work

Current rare itemset mining approaches are based on level-wise exploration of the search space similar to the Apriori algorithm [1]. In Apriori, k-itemsets (itemsets of cardinality k) are used to generate $k+1$-itemsets, which are then pruned using the downward closure property. Apriori terminates when there are no new $k+1$-itemsets remaining after pruning. Rarity, AfRIM, ARIMA and Apriori-Inverse are four algorithms that detect rare itemsets. They all use level-wise exploration similar to Apriori.

Troiano et al. [2] notes that rare itemsets are at the top of the search space, so that bottom-up algorithms must first search through many layers of frequent itemsets. To avoid this, Troiano et al. proposed the Rarity algorithm that begins by identifying the longest transaction within the database and uses them to perform a top-down search for rare itemsets, thereby avoiding the lower layers that only contain frequent itemsets.

In Rarity, potentially rare itemsets (candidates) are pruned in two different ways. Firstly, all k-itemset candidates that are the subset of any of the frequent $k+1$-itemsets

are removed as a candidate, since they must be frequent according to the downward closure property. Secondly, the remaining candidates have their supports calculated, and only those that have a support below the threshold are used to generate the $k-1$-candidates. The candidates with supports above the threshold are used to prune $k-1$-candidates in the next iteration.

Adda et al. [3] proposed AfRIM that uses a top-down approach similar to Rarity. Rare itemset search in AfRIM begins with the itemset that contains all items found in the database. Candidate generation occurs by finding common k-itemset subsets between all combinations of rare $k+1$-itemset pairs in the previous level. Candidates are pruned in a similar way to the Rarity algorithm. Note that AfRIM examines itemsets that have zero support, which may be inefficient.

Szathmary et al. [4] proposed two algorithms that together can mine rare itemsets. As part of those two algorithms, Szathmary et al. defines three types of itemsets: minimal generators (MG), which are itemsets with a lower support than its subsets; minimal rare generators (MRG), which are itemsets with non-zero support and whose subsets are all frequent; and minimal zero generators (MZG), which are itemsets with zero support and whose subsets all have non-zero support. The first algorithm, MRG-Exp, finds all MRG by using MGs for candidate generation in each layer in a bottom up fashion. The MRGs represent a border that separates the frequent and rare itemsets in the search space. All itemsets above this border must be rare according to the antimonotonic property. The second algorithm, ARIMA, uses these MRGs to generate the complete set of rare itemsets. This is done by merging two k-itemsets with $k-1$ items in common into a $k+1$-itemset. ARIMA stops the search for non-zero rare itemsets when the MZG border is reached, since above that there are only zero rare itemsets.

Apriori-Inverse [5] proposed by Koh et al. is used to mine perfectly rare itemsets, which are itemsets that only consist of items below a maximum support threshold (maxSup). Apriori-Inverse is similar to Apriori, except that at initialisation, only 1-itemsets that fall below maxSup are used for generating 2-itemsets. Since Apriori-Inverse inverts the downward-closure property of Apriori, all rare itemsets generated must have a support below maxSup. In addition, itemsets must also meet an absolute minimum support, for example 5, in order for them to be used for candidate generation. Since the set of perfectly rare-rules may only be a small subset of rare itemsets, Koh et al. also proposed several modifications that allow Apriori-Inverse to find near-perfect rare itemsets. The methods are based on increasing maxSup during itemset generation, but using the original maxSup during rule generation.

All of the above algorithms use the fundamental Apriori approach, which has potentially expensive candidate generation and pruning steps. In addition, these algorithms attempt to identify all rare itemsets, and as a result spend a significant amount of time searching for non-rare-item itemsets. However, we will show that these non-rare-item itemsets do not tend to give us interesting rare association rules.

The proposed RP-Tree algorithm is an improvement over these existing algorithms in three ways. Firstly, RP-Tree avoids the expensive itemset generation and pruning steps by using a tree data structure, based on FP-Tree, to find rare patterns. Secondly, RP-Tree focusses on rare-item itemsets which generate interesting rules and does not spend time looking for uninteresting non-rare-item itemsets. Thirdly, RP-Tree is based

on FP-Growth, which is efficient at finding long patterns, since the task is divided into a series of searches for short patterns. This is especially beneficial since rare patterns tend to be longer than frequent patterns.

3 Rare Pattern Tree Mining

In this section we first discuss basic concepts and the definition of rare-item itemsets. We then describe our proposed RP-Tree algorithm and present a simple example. Finally we describe the modification to RP-Tree using an information gain threshold.

3.1 Basic Concept: Rare Itemsets

Let the set of items $\mathcal{I} = \{i_1, i_2, ...i_m\}$, and the transactional database $\mathcal{D} = \{t_1, t_2, ...t_n\}$ where every $t \subseteq \mathcal{I}$. An association rule is an implication $X \rightarrow Y$ such that $X \cup Y \subseteq \mathcal{I}$ and $X \cap Y = \emptyset$. X is the antecedent and Y is the consequent of the rule. The *support* of $X \rightarrow Y$ in $\mathcal{D}$ is the proportion of transactions in $\mathcal{D}$ that contains $X \cup Y$. The *confidence* of $X \rightarrow Y$ is the proportion of transactions in $\mathcal{D}$ containing X that also contains Y. The *lift* of $X \rightarrow Y$ is confidence($X \rightarrow Y$) / support (Y).

The minRareSup threshold is a noise filter, whereby items that are below this threshold are considered as noise. An itemset is a rare itemset if it has support less than the minimum frequent support threshold (minFreqSup) but above or equal to the minimum rare support threshold (minRareSup). As mentioned in Section 1, rare itemsets can be divided into types: *rare-item itemsets* which refers to itemsets that consist of only rare items and itemsets that consist of both rare and frequent items; and *non-rare-item itemsets* which consist of only frequent items which fall below the minimum support threshold.

For instance, suppose there were 4 items $\{a, b, c, x\}$ with supports a = 0.80, b = 0.30, c = 0.50, and x = 0.12, with minFreqSup = 0.15 and minRareSup = 0.05. If the itemset $\{a, b, c\}$ had a support of 0.09, then this itemset would be a non-rare-item itemset ((1) above) since all items are frequent, and its support lies between minFreqSup and minRareSup. The itemset $\{a, x\}$ would be a rare-item itemset ((2) above) assuming that the support of $\{a, x\} > 0.05$, since the itemset includes the rare item x.

Formally, an itemset X is a *rare itemset* iff

$$\text{support}(X) < \text{minFreqSup}, \text{support}(X) \geq \text{minRareSup}$$

An itemset X is a *non-rare-item itemset* iff

$$\forall x \in X, \text{support}(x) \geq \text{minFreqSup}, \ \text{support}(X) < \text{minFreqSup}$$

An itemset X is a *rare-item itemset* iff

$$\exists x \in X, \text{support}(x) < \text{minFreqSup}, \ \text{support}(X) < \text{minFreqSup}$$

3.2 RP-Tree Algorithm

FP-Growth, proposed by Han et al. [6] is a frequent itemset mining algorithm which uses a frequent-pattern tree (FP-Tree) to store a set of database transactions and reduces the required number of database scans to 2. The first scan is used to find the set of items in the database with support over the minimum frequent support threshold; the second is used to construct the initial FP-tree.

The RP-Tree algorithm, shown in Algorithm 1, is a modification of the FP-Growth algorithm. RP-Tree performs one database scan to count item support, similar to FP-Growth. During the second scan, RP-Tree uses only the transactions which include at least one rare item to build the initial tree, and prunes the others, since transactions that only have non-rare items cannot contribute to the support of any rare-item itemset. For example, if $\{x, y, z\}$ was the set of rare items for a given database, minRareSup and minFreqSup, a transaction will have to contain at least one of x, y or z to avoid being pruned.

Note that the ordering of items in each transaction during insertion into the initial tree is according to the item frequency of the original database (and not the database with pruned transactions). This is because rare items in the reduced database may have higher supports than frequent items. If item frequencies of the reduced database were used for transaction item ordering, a frequent item may become the child of a rare item, which invalidates property 1 below.

Using this initial tree, RP-Tree constructs conditional pattern bases and conditional trees for each rare item only. Each conditional tree and the corresponding rare item are then used as arguments for FP-Growth (simplified version shown in Algorithm 2). The threshold used to prune items from the conditional trees is minRareSup. The union of the results from each of these calls to FP-Growth is a set of itemsets that each contain a rare-item, or rare-item itemsets.

The result of RP-Tree is the complete set of rare-item itemsets. This is because:

1. Rare-items will never be the ancestor of a non-rare item in the initial tree due to the tree construction process.
2. All itemsets that involve a particular item a can be found by examining all nodes of a and the nodes of all items that have a lower support than a in the initial tree.

Since RP-Tree examines all rare-item nodes in the initial tree, and all nodes that have a lower support than a rare-item are themselves rare items, RP-Tree must find all rare-item itemsets.

RP-Tree Example. Applying RP-Tree to database $\mathcal{D}$ in Table 1, the support ordered list of all items is $\langle(a{:}7), (i{:}6), (b{:}5), (c{:}4), (l{:}4), (d{:}3), (f{:}3), (e{:}2), (g{:}2), (h{:}1), (j{:}1), (k{:}1), (m{:}1)\rangle$. Using minFreqSup = 4 and minRareSup = 1, only the items $\{d, f, e, g\}$ are rare, and included in $rareItems$.

During construction of the initial RP-Tree, only transactions 1, 3, 4, 5, and 6 are used, since the remaining transactions do not contain any rare items and cannot contribute to any of the result itemsets. In addition, since the support of items h, j, k and m falls below minRareSup, these items are ignored during RP-Tree construction. The initial tree constructed using FP-Growth, which only ignores items that fall below minRareSup, will use all transactions, as shown in Figure 1. This tree has 8 additional

Algorithm 1. RP-Tree

Input: $\mathcal{D}, minRareSup, minFreqSup$;
Output: $results$; // Set of rare-item itemsets

Initialisation:
$allItems \leftarrow \{$all unique items in $\mathcal{D}\}$;
countSupport($allItems$); // First scan of database
$rareItems \leftarrow \{i \in allItems \mid i.supp \geq minRareSup \wedge i.supp < minFreqSup\}$;
$rareItemTrans \leftarrow \{t \in \mathcal{D} \mid \exists r \cdot r \in rareItems \wedge r \in t\}$;
$tree \leftarrow$ constructTree($rareItemTrans$); // Second scan of database

Mining:
$results = \emptyset$;
for item a in $tree$ **do**
  if $a \in rareItems$ **then**
    construct a's conditional pattern-base and then a's conditional FP-Tree $Tree_a$;
    $results \leftarrow results \cup$ FP-Growth($Tree_a, a$);
  end if
end for
return $results$;

Algorithm 2. FP-Growth (without single prefix path optimisation)

Input: $tree, \alpha$;
Output: $results$; // All itemsets generated from $tree$

$results \leftarrow \emptyset$;
for item a in $tree$ do **do**
  generate pattern $\beta \leftarrow a \cup \alpha$ with $support = a.support$;
  $results \leftarrow results \cup \beta$
  construct βs conditional pattern-base and then β's conditional FP-tree $tree_\beta$;
  if $Tree_\beta \neq \emptyset$ **then**
    $results \leftarrow results \cup$ FP-Growth($tree_\beta, \beta$);
  end if
end for
return $results$;

nodes compared to the tree built using RP-Tree from the reduced transaction set (shown in Figure 2(a)). The additional nodes are frequent items that correspond to transactions pruned by RP-Tree. To find the rare-item itemsets, the initial RP-Tree is used to build conditional pattern bases and conditional RP-Trees for each rare item $\{d, f, e, g\}$. The conditional tree for item g is shown in Figure 2(b). Each of the conditional RP-Trees and the conditional item are then used as parameters for the FP-Growth algorithm, for example, FP-Growth($Tree_g, g$).

Table 1. Transaction database $\mathcal{D}$

TID	Transactions	TID	Transactions	TID	Transactions
1	{a, b, c, d, f}	5	{a, c, e, f}	9	{i, l, k}
2	{a, c}	6	{a, b, d, g}	10	{i, b, l}
3	{a, c, d, f, g}	7	{i, b, c, l, j}	11	{i, m}
4	{a, b, e, h}	8	{a, i, l}	12	{i}

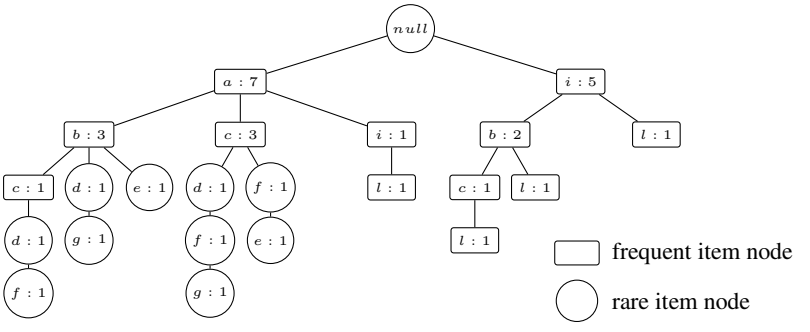
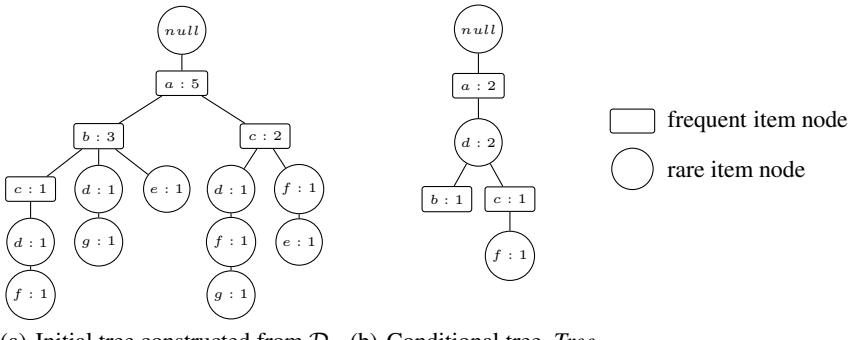


Fig. 1. Pattern tree constructed from database $\mathcal{D}$ using FP-Tree



(a) Initial tree constructed from $\mathcal{D}$ (b) Conditional tree, $Tree_g$

Fig. 2. Pattern trees constructed from database $\mathcal{D}$ using RP-Tree

3.3 RP-Tree with Information Gain

Rules that predict the occurrence of rare-items are more interesting than rules that predict the occurrence of frequent items. To identify these rules, RP-Tree has been extended using an information gain component (RP-Tree-IG) to remove frequent items that are not good predictors of rare items. This is done by treating rare items as classifications and each frequent item as a separate attribute. Transactions that contain more than one rare item (and class) are converted into multiple transactions during the information gain calculation so that the transaction contains only 1 rare item. For example, $\{a, b, d, e\}$, where d and e are rare, is split into $\{a, b, d\}$ and $\{a, b, e\}$.

Information gain [7] is calculated as: $\text{IG}(X) = \text{Entropy}(Y) - \text{Entropy}(Y \mid X)$ where Y is the set of rare items, and X is a frequent item. Frequent items that do not have an information gain higher than a pre-defined threshold are not used for itemset generation. Specifically, line 11 in Algorithm 1 and line 4 in Algorithm 2 becomes:

$$\text{for item } a \in tree \text{ where } IG(a) \geq minIG$$

where $minIG$ is the minimum information gain threshold an item must meet to be used for generating itemsets. Using the previous example, the classes Y are $\{d, f, e, g\}$, and attributes X are $\{\{a\}, \{b\}, \{c\}\}$. The information gain of c is $IG(c) = 1.971 - 1.842 = 0.129\, bits$. If, for example, $minIG = 0.1$, then c would be used for generating itemsets.

4 Experimental Results

In our experiments we compared the performance of ARIMA, FP-Growth, and RP-Tree with and without the information gain component. We also generated rules from these itemsets and compared the quality of these rules using the seven interest measures examined in [8]: χ^2, lift, confidence (all and max), coherence, cosine and Kulczynski (abbreviated to kulc). The equations for calculating these measures are shown in Table 2. All algorithms were implemented in Java and executed on an Intel Core 2 Duo 2.33 GHz machine with 4GB of RAM running Windows 7.

In these experiments ARIMA and FP-Growth were modified in order to obtain comparable results within a reasonable time. The ARIMA algorithm was modified in two ways. Firstly, the absolute minimum support (corresponding to minRareSup in RP-Tree) may now be greater than 1. An itemset must meet this support threshold to be included in the result. This is necessary to allow the removal of noisy itemsets, and to allow ARIMA to finish in a reasonable time without setting minFreqSup to an extremely low value. Secondly, candidate support count is done by building a tree structure with candidates from each level, which is more efficient than iterating through each itemset for each transaction. The FP-Growth algorithm was modified to find rare itemsets by generating all itemsets that meet minRareSup, then removing all itemsets that exceed minFreqSup.

Experiments comparing RP-Tree and RP-Tree-IG with Rarity and AfRIM have been omitted since they use a level-wise approach similar to ARIMA, and their performance compared to ARIMA has already been reported in detail in [2] and [3]. It is sufficient to note that AfRIM and Rarity can perform several orders of magnitude faster than ARIMA under specific conditions, such as a low number of rare itemsets, or when there is a small number of items. However, in general, AfRIM performs 2-3 times faster, while Rarity performs about 30 times faster than ARIMA.

Nine datasets from UCI Repository [9] were used in the experiments: Connect-4, Congressional Voting Records (Voting), Primary Tumor (Tumor), Zoo, Teaching Assistant Evaluation (Teaching), Flags, Adult, Dermatology and Soybean Large (Soybean).

4.1 Itemset Generation Performance

In this section we compare the time taken for itemset generation for ARIMA, FP-Growth, RP-Tree and RP-Tree-IG. We use the same minFreqSup and minRareSup threshold across all experiments.

Table 2. Rule Interest Measures [8]

Measure	Definition
χ^2	$\sum \frac{(observed-expected)^2}{expected}$
Lift$(X \rightarrow Y)$	$\frac{sup(X \cup Y)}{sup(X)sup(Y)}$
AllConf$(X \rightarrow Y)$	$\frac{sup(X \cup Y)}{max(sup(X),sup(Y))}$
MaxConf$(X \rightarrow Y)$	$max\{\frac{sup(X \cup Y)}{sup(X)}, \frac{sup(X \cup Y)}{sup(Y)}\}$
Coherence$(X \rightarrow Y)$	$\frac{sup(X \cup Y)}{sup(X)+sup(Y)-sup(X \cup Y)}$
Cosine$(X \rightarrow Y)$	$\frac{sup(X \cup Y)}{\sqrt{sup(X)sup(Y)}}$
Kulc$(X \rightarrow Y)$	$\frac{sup(X \cup Y)}{2}(\frac{1}{sup(X)} + \frac{1}{sup(Y)})$

Table 3. Time taken for itemset generation

	ARIMA			FP-Growth			RP-Tree			RP-Tree-IG		
Dataset	Itemsets	Time (s)	Rel. time	Itemsets	Time (s)	Rel. time	Itemsets	Time (s)	Rel. time	Itemsets	Time (s)	Rel. time
Connect-4	46428	292.53	32.30	46428	9.06	1.00	35494	8.72	0.963	57	2.58	0.285
Voting	1437652	1099.35	230.86	1437652	4.76	1.00	225634	0.87	0.184	16	0.56	0.119
Tumor	1111993	699.12	178.53	1111993	3.92	1.00	309698	0.94	0.239	17	0.15	0.039
Zoo	484139	496.12	194.94	484139	2.55	1.00	117934	0.87	0.343	6	0.05	0.019
Teaching	281	0.38	4.28	281	0.09	1.00	118	0.10	1.067	12	0.05	0.573
Flags	233533	962.66	372.55	233533	2.58	1.00	185	0.33	0.129	8	0.01	0.004
Adult	72658	340.60	47.37	72658	7.19	1.00	57463	6.45	0.898	50	3.65	0.508
Dermatology	-	-	-	4.90E+08	1419.537	1.00	9.12E+07	265.206	0.187	2.37E+06	270.562	0.191
Soybean	-	-	-	1.67E+08	1184.961	1.00	4.34E+06	15.902	0.013	1.40E+05	14.913	0.013

Table 3 shows, for each dataset, the number of itemsets generated, the absolute time taken for each algorithm. The relative time compared to FP-Growth is also given, with relative time for FP-Growth is set to 1.0. The differences in the number of itemsets are due to the types of rare itemsets each algorithm can generate. ARIMA and FP-Growth both generate the complete set of rare itemsets, RP-Tree generates only rare-item itemsets, and RP-Tree-IG generates rare-item itemsets using items that meet an information gain threshold. Note that the results for ARIMA for the Dermatology and Soybean datasets have been excluded since execution did not complete within 2 hours. The runtime for ARIMA is more than 32 times longer than FP-Growth in all datasets except Teaching. This is due to the Teaching dataset being a fairly small dataset, and overhead takes up a large proportion of the time taken. Realistically, most real world datasets would be of a reasonable size and the overhead is negligible. We also see that time taken for RP-Tree-IG is consistently less than that of RP-Tree, which is in turn less than FP-Growth. The time taken for ARIMA is significantly longer than FP-Tree due to the computationally expensive candidate generation and pruning steps. The differences in time taken for RP-Tree and RP-Tree-IG are the result of pruning transactions without any rare items, and pruning itemsets without any rare items above the minIG threshold, respectively. Transaction pruning reduces the size of the initial tree generated, and reduces the amount of computation required and the number of rare itemsets found.

4.2 Changes in Rule Quality

Rules were generated from the itemsets found using ARIMA/FP-Growth (both of which generate the complete set of rare itemsets), RP-Tree and RP-Tree-IG. For all algorithms, parameters are: minFreqSup = 15% and minRareSup = 5. For RP-Tree with information

Table 4. UCI Datasets

FP-Growth without single prefix path optimisation (removing frequent patterns)												
Dataset	Itemsets	Rules	Time (s)	Support	Conf	χ^2	Lift	AllConf	Coherence	Cosine	Kulc	MaxConf
Connect-4	9.55E+05	243	1317	19.374	0.999	911.416	49.296	0.014	0.014	0.097	0.507	0.999
Voting	2.11E+06	46	425	56.935	0.918	56.094	1.918	0.273	0.267	0.50	0.596	0.918
Tumor	3.40E+06	26858	489	18.099	0.939	12.643	1.686	0.097	0.096	0.281	0.518	0.939
Zoo	6.49E+05	102932	59	11.497	0.951	16.157	2.283	0.272	0.266	0.483	0.611	0.951
Teaching	281	53	<1	6.849	0.992	19.608	3.408	0.170	0.170	0.347	0.581	0.992
Flags	1.50E+08	137133	25394	6.546	0.997	4.064	1.619	0.054	0.054	0.224	0.526	0.997
Adult	1.88E+06	83	1640	1822.120	0.944	2427.051	4.345	0.090	0.088	0.204	0.517	0.944
Dermatology	4.90E+08	29613244	186224	34.352	0.938	10.964	1.375	0.132	0.131	0.343	0.535	0.938
Soybean	1.67E+08	47519598	64834	40.013	1.000	21.678	1.471	0.192	0.192	0.435	0.596	1.000
RP-Tree												
Dataset	Itemsets	Rules	Time (s)	Support	Conf	χ^2	Lift	AllConf	Coherence	Cosine	Kulc	MaxConf
Connect-4	8.96E+05	243	1270	19.374	0.999	911.416	49.296	0.014	0.014	0.097	0.507	0.999
Voting	3.38E+05	46	46	56.935	0.918	56.094	1.918	0.273	0.267	0.500	0.596	0.918
Tumor	1.41E+05	26858	154	18.099	0.939	12.643	1.686	0.097	0.096	0.281	0.518	0.939
Zoo	1.79E+05	102932	15	11.497	0.951	16.157	2.283	0.272	0.266	0.483	0.611	0.951
Teaching	118	49	<1	6.673	1.000	20.319	3.506	0.172	0.172	0.347	0.586	1.000
Flags	4.69E+07	137048	6517	6.533	0.997	4.061	1.619	0.054	0.054	0.223	0.526	0.997
Adult	1.85E+06	80	1614	1623.950	0.945	2513.977	4.469	0.087	0.085	0.197	0.516	0.945
Dermatology	9.12E+07	29593672	26005	34.340	0.938	10.966	1.376	0.132	0.131	0.343	0.535	0.938
Soybean	4.34E+06	47368525	1150	40.000	1.000	21.704	1.472	0.192	0.192	0.435	0.596	1.000
RP-Tree with Information Gain (minIG = 0.25)												
Dataset	Itemsets	Rules	Time (s)	Support	Conf	χ^2	Lift	AllConf	Coherence	Cosine	Kulc	MaxConf
Connect-4	10803	0	3	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
Voting	613987	46	11	56.935	0.918	56.094	1.918	0.273	0.267	0.5	0.596	0.918
Tumor	500	2	< 1	9.000	1.000	15.487	2.748	0.071	0.071	0.266	0.536	1.000
Zoo	47364	1140	1	12.039	0.960	24.611	2.808	0.348	0.342	0.551	0.654	0.960
Teaching	51	8	<1	7.125	1.000	58.796	8.138	0.416	0.416	0.591	0.708	1.000
Flags	8167	12	<1	11.750	0.993	41.297	4.245	0.261	0.259	0.49	0.627	0.993
Adult	19397	7	2	1151.286	0.958	1070.297	2.029	0.047	0.047	0.184	0.502	0.958
Dermatology	615585	63869	14	39.003	0.961	13.887	1.336	0.148	0.147	0.37	0.554	0.961
Soybean	292424	257477	8	40.000	1.000	28.510	1.620	0.211	0.211	0.457	0.606	1.000

gain minIG is set to 0.25. Only rules that met the minimum confidence of 0.9 and lift of 1.0 were included for analysis. Table 4 shows the number of rules retained for each dataset and algorithm, the average support and confidence, and the average values for each of the seven measures listed in Table 2.

For the Connect-4 and Adult datasets, FP-Growth and RP-Tree found a very similar number of itemsets. For the remaining seven datasets, RP-Tree generated significantly fewer itemsets compared to FP-Growth, ranging from 42.0% for Adult to 2.60% for Soybean. However, the number of rules that met the confidence and lift thresholds were either identical or very similar. This shows that the set of rare itemsets that are ignored by RP-Tree does not tend to be interesting, since these itemsets do not generate rules that meet both the confidence and lift thresholds. Since fewer itemsets are generated by RP-Tree compared to FP-Growth, the time required for rule generation is also less: for example, time taken for rule generation for RP-Tree for the soybean dataset is reduced to 1.8% of that for FP-Growth, while the number of association rules retained is 99.7% of FP-Growth. Overall the time taken for RP-Tree is lower than FP-Growth.

The Information Gain component for RP-Tree results in far fewer rules than RP-Tree for all datasets except Voting, with no change, and tends to generate rules that are of higher quality. For five datasets (Zoo, Teaching, Flags, Soybean and Dermatology) there are increases in most of the seven interest measures. However, for one dataset (Adult), the measures decreased. There were no rules generated at all for Connect-4. The

reduction in the number of rules is due to the minIG threshold reducing the number of items that can participate in itemsets. Overall, the information gain component tends to selectively retain rules that are more interesting according to the seven interest measures.

Case Study. From the Teaching dataset, FP-Growth, RP-Tree and RP-Tree-IG generated 53 rules, 49 rules, and 8 rules respectively. The interest measures of the 4 additional rules generated by FP-Growth from non-rare-item itemsets are shown in Table 5. Rules 1 and 2 have lower than average values for confidence and all interest measures. Rules 3 and 4 have lower values for confidence, lift, kulc and maxConf; and higher values for χ^2, allConf, coherence and cosine. 8 association rules were generated using RP-Tree-IG. Of the 8 rules, 6 had higher than average values for confidence and all 7 interest measures compared to FP-Growth, while the remaining 2 had lower values.

From the Adult dataset, FP-Growth, RP-Tree and RP-Tree-IG generated 83 rules, 80 rules and 7 rules respectively. The 3 additional rules generated by FP-Growth had lower than average values for confidence, χ^2, lift and maxConf; and higher values for allConf, coherence, cosine and kulc, as shown in Table 5 . The 7 rules generated all had several measures that were lower than the average compared to FP-Growth.

The omission of non-rare-item itemsets by RP-Tree only has a small effect on the number and quality of association rules generated compared to FP-Growth, since the additional rules are of average quality and are few in number compared to the overall number of rules generated.

Table 5. Non-rare-item itemsets generated by FP-Growth

Dataset	Rule ID	Confidence	χ^2	Lift	AllConf	Coherence	Cosine	Kulc	MaxConf
Teaching	1	0.900	0.585	1.114	0.074	0.073	0.258	0.487	0.900
	2	0.900	0.227	1.062	0.070	0.070	0.252	0.485	0.900
	3	0.900	21.385	3.315	0.220	0.214	0.444	0.560	0.900
	4	0.900	21.385	3.315	0.220	0.214	0.444	0.560	0.900
Adult	1	0.928	88.662	1.034	0.160	0.158	0.386	0.544	0.928
	2	0.908	208.679	1.062	0.170	0.168	0.393	0.539	0.908
	3	0.915	29.765	1.019	0.164	0.161	0.387	0.539	0.915

5 Conclusions and Future Work

We present a new method for finding rare association rules in large databases. To our knowledge, this is the first algorithm that uses a tree structure to mine rare itemsets. Our algorithm finds a subset of all rare itemsets, which we call rare-item itemsets. We evaluated our method by comparing the quality of association rules generated against those generated using the FP-Growth algorithm from 9 datasets. We found that, in the majority of cases, RP-Tree generated far fewer itemsets for some datasets compared to FP-Growth. This meant that rule generation took much less time for RP-Tree than FP-Growth. However, at the same time, there was very little reduction in the number of rules that met the minimum confidence and lift thresholds. This shows that rare-item itemsets are more interesting since they contribute to almost all the rules that pass the thresholds, and the omission of non-rare-item itemsets by RP-Tree does not reduce rule quality, and in most cases, improves the overall rule quality in the set.

In our future work, we intend to find other ways of focusing on more potentially interesting association rules, such as rules that contain only rare items as the consequent. In addition, we intend to investigate the effect of the minRareSup on the quality of rules generated by RP-Tree, and to find ways of dealing with noise and removing coincidental non-rare-item itemsets.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, Santiago, Chile, pp. 487–499 (1994)
2. Troiano, L., Scibelli, G., Birtolo, C.: A fast algorithm for mining rare itemsets. In: Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications, pp. 1149–1155. IEEE Computer Society Press, Los Alamitos (2009)
3. Adda, M., Wu, L., Feng, Y.: Rare itemset mining. In: Proceedings of the Sixth International Conference on Machine Learning and Applications, ICMLA 2007, pp. 73–80. IEEE Computer Society Press, Los Alamitos (2007)
4. Szathmary, L., Napoli, A., Valtchev, P.: Towards rare itemset mining. In: Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2007, vol. 01, pp. 305–312. IEEE Computer Society, Los Alamitos (2007)
5. Koh, Y.S., Rountree, N.: Finding sporadic rules using apriori-inverse. In: Ho, T.-B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 97–106. Springer, Heidelberg (2005)
6. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD 2000, pp. 1–12. ACM, New York (2000)
7. Mitchell, T.M.: Machine Learning, pp. 57–60. McGraw-Hill, New York (1997)
8. Wu, T., Chen, Y., Han, J.: Association mining in large databases: A re-examination of its measures. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 621–628. Springer, Heidelberg (2007)
9. Frank, A., Asuncion, A.: UCI machine learning repository (2010), `http://archive.ics.uci.edu/ml`