Unsupervised Rare Pattern Mining: A Survey

Yun Sing Koh, University of Auckland
Sri Devi Ravana, University of Malaya

Association rule mining was first introduced to examine patterns among frequent items. The original motivation for seeking these rules arose from need to examine customer purchasing behaviour in supermarket transaction data. It seeks to identify combinations of items or itemsets, whose presence in a transaction affects the likelihood of the presence of another specific item or itemsets. In recent years, there has been an increasing demand for rare association rule mining. Detecting rare patterns in data is a vital task, with numerous high-impact applications including medical, finance, and security. This survey aims to provide a general, comprehensive, and structured overview of the state-of-the-art methods for rare pattern mining. We investigate the problems in finding rare rules using traditional association rule mining. As rare association rule mining has not been well explored, there is still specific groundwork that needs to be established. We will discuss some of the major issues in rare association rule mining and also look at current algorithms. As a contribution we give a general framework for categorizing algorithms: Apriori and Tree based. We highlight the differences between these methods. Finally we present several real-world application using rare pattern mining in diverse domains. We conclude our survey with a discussion on open and practical challenges in the field.

Categories and Subject Descriptors: H.2.8 [**Database Management**]: Database Applications – Data mining; A.1 [**Introductory and Survey**]

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Association Rule Mining, Infrequent Patterns, Rare Rules

ACM Reference Format:
Yun Sing Koh and Sri Devi Ravana, 2016. Unsupervised Rare Pattern Mining: A Survey. *ACM Trans. Knowl. Discov. Data.* 0, 0, Article 0 (2016), 31 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

The main goal of association rule mining is to discover relationships among sets of items in a transactional database. Association rule mining, introduced by Agrawal et al. [1993] aims to extract interesting correlations, frequent patterns, associations or casual structures among sets of items in transaction databases or other data repositories. The relationships are not based on intrinsic properties of the data themselves but rather based on the co-occurrence of the items within the database. These associations between items are also known as association rules.

Two major types of rules can be found in a database: frequent and rare rules. Both frequent and rare association rules present different information about the database in which they are found. Frequent rules focus on patterns that occur frequently, while

Author's addresses: Y. Koh, Department of Computer Science, University of Auckland. S.D. Ravana, Department of Information Systems, Faculty of Computer Science & Information Technology, University of Malaya.

DOI: http://dx.doi.org/10.1145/0000000.0000000

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. Examples of mining infrequent itemsets include identifying relatively rare diseases, predicting telecommunication equipment failure, and finding associations between infrequently purchased supermarket items. 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. Suppose a database of patient symptoms contains a rare itemset *elevated heart rate, fever, skin bruises, low blood pressure*, where all items other than *low blood pressure* are frequent items. This itemset will produce a rule: *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 are symptoms of severe sepsis [Tsang et al. 2011]. Motivated by the crucial and critical information that may be derived from rare patterns, there has been a spur of research in this area.

The main difficulty in finding rare patterns or rare association rule is the mining process can be likened to finding a needle in a haystack [Weiss 2004]. The difficultly of searching for a small needle is multiplied by the fact that the needle is obscured by a large amount of hay strands. Likewise, rare rules pose difficulties for data mining systems for a variety of reasons. The fundamental problem is the lack of data associated with rare cases. Rare rules tend to cover only a few examples. The lack of data makes detecting rare cases difficult and, even when rare case are detected, it is hard for us to meaningfully generalize the results. Generalization is a difficult task when we are tasked with identifying regularities from few data points.

In classical association rule mining, all frequent itemsets are found, where an itemset is said to be frequent if it appears at least a given percentage of the time in all transactions called minimum support (minsup). Then association rules are found in the form of $A \rightarrow B$ where AB is a frequent itemset. Strong association rules are derived from frequent itemsets and constrained by minimum confidence (minconf). Confidence is the percentage of transactions containing A that also contain B. An example of an association rule, $A \rightarrow B$ is 90% of transactions that contain item A also contain item B. 30% of all transactions contain item A. 27% of all transactions contain these two items together. Here the support of the rule is 27% and the confidence is 90%. Confidence is the conditional probability of a transaction containing A also containing B. Currently there are many association rule mining algorithms dedicated to frequent itemset mining. These algorithms are defined in such a way that they only find rules with high support and high confidence. Most of these approaches adopt an Apriori-like approach, which is based on anti-monotonic Apriori heuristics [Agrawal and Srikant 1994]. This means if any length k-itemset is not frequent in the database, its superset length $(k+1)$-itemsets can never be frequent. The Apriori algorithm iteratively generates sets of candidate itemsets and checks their corresponding occurrence frequencies in the database.

Common association rule mining algorithms, such as Apriori, discover valid rules by exploiting support and confidence requirements and using the minimum support threshold to prune the combinatorial search space. Two major problems may arise when we apply these techniques. Intuitively to discover rare rules, every transaction has to be inspected to generate the support of all combinations of items. However, this significantly increases the workload during candidate set generation, construction of tree nodes and testing. The number of rules generated would be considerable, and not all the rules generated may be interesting. Patterns with items having significantly different support levels would be produced, which usually have weak correlations.

However if we set the minimum support threshold too high, interesting rare rules will never be generated. Rare rules with low support would never be considered in the itemset generation.

Using frequent pattern mining approaches, items that rarely occur are in very few transactions are normally pruned out. Thus, infrequent itemsets warrant special attention as they are more difficult to find using traditional data mining techniques, thus new approaches need to be developed to cater for these specific types of itemsets. Recently, there have been some growing interests in developing techniques to mine rare patterns.

1.1. Types of Rare Patterns

Overall there are two major categories of the types of rare patterns: basic and extended. In Figure 1 we show the roadmap of the categories of rare pattern. In the basic category, we have association rule mining and rare patterns. In the extended category, there is a range of other patterns including subgraph, probabilistic, and sequence patterns. Based on pattern diversity, rare pattern mining can be classified using the following criteria basic and extended patterns.

Basic Rare Patterns: Rare patterns can also be mapped into basic association rules, or other kinds of rare rules based on constraints or interestingness measures. The major advantage on focusing on the basic type of patterns and this category of mining techniques is they are widely applicable in different application domains. Furthermore, we can also utilize those techniques on different data types. For example we can apply rare pattern techniques on a sequential dataset. Sequential datasets have additional temporal information compared to a normal transactional dataset. Using basic approaches on a sequential datasets has its disadvantages such that we would strip away some underlying information, for example, the temporal information in a sequential dataset. In this survey we will concentrate on the basic type of rare patterns and the current mining techniques in the area. We aim to provide and discuss a large variety of research in the area.

Extended Rare Patterns: Overall most of the current research in the field concentrates on finding the basic type of rare patterns. There exists research concentrating on finding the patterns in the extended category. The main reason behind this could be the computational complexity of these mining techniques in detecting rare occurrences of events. For example, frequent graph mining techniques can suffer from costly subgraph isomorphism testing and generate an enormous number of candidates during the mining process. This problem is exacerbated when we try to look at rare graph mining. Rare subgraph pattern techniques have been used to find useful substructures in a graph database. For example, network security administrators can conduct a pattern (subgraph) matching over the network traffic data to detect possible malicious attacks which are relatively uncommon compared to the normal network traffic.

Some rare patterns may involve sequences and structures. For example, by studying the order in which rare events occur we may be able to find sequential patterns, that is, rare subsequences in a sequence of ordered events. For example, when a diagnosing a patients condition, the context is reflected in the sequence of conditions that has appeared across a time period. By mining sequential patterns of symptoms we can capture a patient's condition. In this way, a patient's condition can be found at a more abstract level and it gets easier to track and detect changes in the patient's conditions. For instance, *high temperature* $\rightarrow$ *low blood pressure* $\rightarrow$ *high pulse rate* in women who has just given birth could indicate an onset of a secondary postpartum hemorrhage which is a relatively rare condition which can cause mortality.

In some datasets we need to consider the existential uncertainty of rare itemsets, indicating the probability that an itemset occurs in a transaction, makes traditional

techniques inapplicable. For example a medical dataset may contain a table of patient records, each of which contains a set of symptoms and/or illnesses that a patient suffers from. Applying rare association analysis on such a dataset allows us to discover any potential correlations among the symptoms and illnesses for rare diseases. In these cases, symptoms would best be represented by probabilities based on historical data statistics that indicate their presence in the patients' records. These type of data is known as uncertain data.

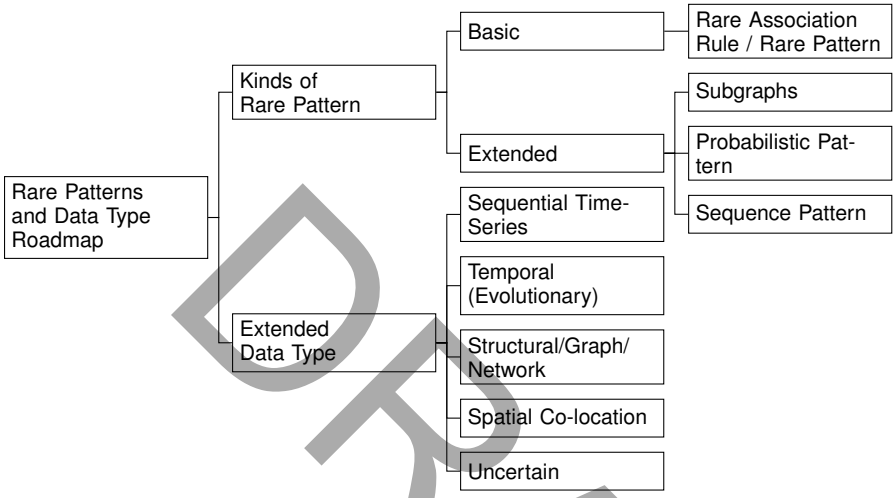

Fig. 1. Roadmap for Types Rare Patterns

We can also apply those techniques on other kinds of data types. For example we can apply rare pattern techniques on a sequential dataset [Hu et al. 2014]. Sequential datasets have additional temporal information. In sequential time-series data analysis, researchers can discretize time-series values into multiple levels or intervals. In doing so small fluctuations and differences in value can be ignored. The data can then be summarized into sequential patterns, which can be indexed to facilitate similarity search or comparative analysis, which are normally used in recommender systems. Whereas we can also use temporal or evolutionary data, whereby the data is dynamic and may evolve over time [Huang et al. 2014]. Suppose an analyst is modelling abnormal user behaviours for detecting fraudulent call patterns, processing the cause of frauds can change dynamically as fraudsters continuously find new ways to break the system.

We may also mine rare structural patterns [Lee et al. 2015] which are substructures from a structured dataset. Note that structure is a general concept that covers many different kinds of structural forms including directed graphs, undirected graphs, network, lattices, trees, sequences, sets, or combinations of such structures. A general pattern may contain different elements such as subsequence, subtree, or subgraph.

Pattern analysis is useful in the analysis of spatial co-location data. Rare spatial co-location patterns [Huang et al. 2006] represent rare relationships among events happening in different and possibly nearby locations. For example, these can help determine if a certain rare but deadly disease is geographically co-located with certain areas, to establish patterns of potential epidemics occurring.

1.2. Our Contributions

This survey is an attempt to provide a structured and broad overview of extensive research in rare pattern mining spanning different application domains. In this survey we will take an in-depth look at the issues in rare pattern mining, applications of rare pattern mining, current rare pattern mining approaches, an evaluation of the techniques, and a discussion on the open challenges in the area.

We divide the current rare pattern mining techniques into two types: static and dynamic. We proposed that techniques that deal with static data are divided into four categories: variable support threshold, without support threshold, consequent constraint-based rule mining, and tree based approaches. Techniques that deals with dynamic data are divided into two categories: with and without drift detection.

We identify the advantages and disadvantages of the techniques in each category. We also provide a detailed discussion of the application domains where rare pattern mining techniques have been used. We discuss the similarity between rare pattern mining and a related research area of anomaly detection. We also looked at the open challenges in rare pattern mining.

1.3. Organization

In Section 2 we identify the various aspects that determine the formulation of the problem and highlight complexity associated with rare patten mining. In Section 3 we briefly describe the different application domains where rare pattern mining has been applied. In subsequent sections we provide a categorization of rare pattern mining techniques based on the research area which they belong to. The majority of techniques can be categorized as techniques for detecting rare rules in static data, and are cataloged in Section 4. The techniques for detecting rare rules in dynamic data are cataloged in Section 5. We compare and contrast the characteristics of the techniques in Section 6. We present some discussion on the similarity and differences between rare pattern mining techniques and anomaly detection in Section 7. Section 8 contains our concluding remarks.

2. BASIC CONCEPT: RARE PATTERN PROBLEM

The most popular pattern discovery method in data mining is association rule mining. The associations between items are also known as association rules. Association rule mining was also initially known as frequent pattern mining [Agrawal et al. 1993]. The following is a formal statement of association rule mining for a transaction database.

Let $I = i_1, i_2, \ldots, i_m$ be the universe of items. A set $X \subseteq I$ of items is called an itemset or pattern. In particular, an itemset with k items is called a k-itemset. In association rule mining, the main function of a unique transaction ID is to allow an itemset to occur more than once in a database. Every transaction contains a unique transaction ID tid. A transaction $t = (tid, X)$ is a tuple where X is an itemset. A transaction $t = (tid, X)$ is said to contain itemset Y if $Y \subseteq X$. Let $\{t_1, t_2, \ldots, t_n\}$ be the set of all possible transactions called T. A transaction database D is a set of transactions, such that $D \subseteq T$. In effect, D is really a multi-set of itemsets. The *count* of an itemset X in D, denoted by $\text{count}(X, D)$, is the number of transactions in D containing X. The *support* of an itemset X in D, denoted by $\text{sup}(X, D)$, is the proportion of transactions in D that contain X, *i.e.*,

$$\text{sup}(X, D) = \frac{\text{count}(X, D)}{|D|}$$

where $|D|$ is the total number of transactions in D.

Definition 2.1 (*Rare Itemset*). Given a user-specified minimum support threshold minsup $\in [0, 1]$, X is called a rare itemset or rare pattern in D if $\text{sup}(X, D) \leq minsup$.

Typically the problem of mining rare patterns is to find the meaningful set of rare patterns in a transaction database with respect to a given support threshold.

Definition 2.2 (*Association Rule*). An association rule is an implication of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$, and $X \cap Y = \emptyset$.

The left-hand side (LHS) X is referred to as the *antecedent* of the rule, and the right-hand side (RHS) Y as the *consequent*. The rule $X \rightarrow Y$ has support of s in the transaction set D, if $s = \text{sup}(XY, D)$. XY refers to an itemset that contains both X and Y, and is always used in association rule mining literature instead of $X \cup Y$. The rule $X \rightarrow Y$ holds in the transaction set D with *confidence* c where $c = \text{conf}(X \rightarrow Y, D)$,

$$\text{conf}(X \rightarrow Y, D) = \frac{\text{sup}(XY, D)}{\text{sup}(X, D)}.$$

Given a transaction database D, a support threshold *minsup* and a confidence threshold *minconf*, the task of association mining is to generate all association rules that have support and confidence above the user-specified thresholds. This is known as the *support-confidence* framework [Agrawal et al. 1993].

However, it is difficult to find association rules with low support but high confidence using Apriori-like methods. In order to find these rules, the minimum support threshold would need to be set quite low to enable rare items to be included. The Apriori heuristic would be able to reduce the size of candidate itemsets if the minimum support is set reasonably high. However, in situations with abundant frequent patterns, long patterns, or a low minimum support threshold, an Apriori-like algorithm may still suffer from the following nontrivial cost: most of the items would be allowed to participate in itemset generation. This will have an effect on the scalability of the Apriori algorithm. It is costly to handle a huge number of candidate items. It is time consuming to repeatedly scan the database and check each of the candidate itemsets generated. The complexity of the computation will increase exponentially with regard to the number of itemsets generated.

Traditional association rule analysis, such as Apriori, focus on mining association rules in large databases with a single minsup. Since a single threshold is used for the whole database, it assumes that all items in the database are of the same nature and/or have similar frequencies. As such, Apriori works best when all items have approximately the same frequency in the data. Apriori exploits the downward closure property that states that if an itemset is frequent so are all its subsets. This leads to a weakness in Apriori. Consider the case where the user-defined minsup of $\{A, B, C\}$ is 2% and the user-defined minsup of $\{A, B\}$, $\{A, C\}$, and $\{B, C\}$ is 5%. It is possible for $\{A, B, C\}$ to be frequent with respect to its minsup but none of $\{A, B\}$, $\{A, C\}$, $\{B, C\}$ to be frequent with respect to their minsup. $\{A, B\}$, $\{A, C\}$, $\{B, C\}$ may have support below 5% but $\{A, B, C\}$ may still be above 2%. In this case itemset $\{A, B, C\}$ is considered frequent because of the different minsup values. In reality, some items may be very frequent while others may rarely appear. Hence minsup should not be uniform because deviation and exceptions generally have a much lower support than general trends. Note that support requirements vary as the support of items contained in an itemset varies.

Given that the existing Apriori algorithm assumes a uniform support, rare itemsets can be hard to find. Items that rarely occur are in very few transactions and will be pruned because they do not meet the minsup threshold. In data mining, rare itemsets may be obscured by common cases. Weiss [2004] calls this relative rarity.

This means that items may not be rare in the absolute sense but are rare relative to other items. This is especially a problem when data mining algorithms rely on greedy search heuristics that examine one item at a time. Since rare cases may depend on the conjunction of many conditions, analysing any single condition alone may not be interesting [Weiss 2004].

As a specific example of the problem, consider the association mining problem in determining if there is an association between *buying a food processor* and *buying a cooking pan* [Liu et al. 1999b]. The problem is that both items are rarely purchased in a supermarket. Thus, even if the two items are almost always purchased together, when either one is purchased, this association may not be found. Modifying the minsup threshold to take into account the importance of the items is one way to ensure that rare items remain in consideration. In order to find this association minsup must be set low. However setting this threshold low would cause a combinatorial explosion in the number of itemsets generated. Frequently occurring items will be associated with one another in an enormous number of ways simply because the items are so common that they cannot help but appear together. This is known as the rare item problem [Liu et al. 1999b]. It means that using the Apriori algorithm, we are unlikely to generate rules that may indicate rare events of potentially dramatic consequence. For example, we might prune out rules that indicate the symptoms of a rare but fatal disease due to the frequency of occurrence not reaching the minsup threshold.

As rare rule mining is still an area that has not been well explored, there is some groundwork that needs to be established. A real dataset will contain noise, possibly at levels of low support. Normally, noise has low support. In Apriori, setting a high minimum support threshold would cut the noise out. Inherently we are looking for rules with low support that could make them indistinguishable from coincidences (that is, situations where items fall together no more often than they would normally by chance). Apriori is the most commonly used association mining technique. However it is not efficient when we try to find low support rules. Using Apriori, we would still need to wade through thousands of itemsets (often having high support) to find the rare itemsets that are of interest to us. Another approach was to use a tree based approach. Most tree based rare pattern mining approaches follow the traditional FP-Tree [Han et al. 2000] approach. It is a two-pass approach and is affordable when mining a static dataset. However in a data stream environment, a two-pass approach is not suitable. The overview of mining techniques in Sections 4 and 5 is shown in Table I.

Table I. Categorization of Rare Pattern Mining Approaches

	Apriori Based	Tree Based
Static	Sections 4.1- 4.3	Section 4.4
Dynamic	-	Section 5

In the next section we look at some application areas where rare pattern mining have been utilized. Rare pattern mining has been used in many different applications including detecting suspicious uses and behaviors in the context of web applications, finding infrequent patterns on pre-processed wireless connection records, and finding patterns that look for adverse drug reactions.

3. APPLICATION OF RARE PATTERN MINING IN VARIOUS DOMAINS

Rare patterns can be applied in various domains including biology, medicine and security. In the security field, normal behaviour is very frequent, whereas abnormal or suspicious behaviour is less frequent. When analysing a database where the behaviour of

people in sensitive places such as airports are recorded, if we model those behaviours, a possible outcome would be to find that normal behaviours can be represented by frequent patterns and strange or suspicious behaviours by rare patterns. In the medical field, by analysing clinical databases a user can discover rare patterns or trends that will assist doctors to make decisions about the clinical care. In this section, we discuss some of the applications of rare pattern mining techniques.

3.1. Detecting Suspicious Web Usage

Adda et al. [2012] adopted an approach to find rare patterns to detect suspicious uses and behaviors. The mining system Rare Pattern Mining for Suspicious Use Detection (RPMSUD) [Adda et al. 2012] is mainly composed of an engine that unobtrusively analyses the usage data of a running web application and detects suspicious activities. The authors made an assumption that a web application usage is dominated with repetitive access/requests and that rare access patterns are a source of potential abnormal usage that may be related to security issues.

The algorithm behind the RPMSUD system is composed of an event queue processing process that periodically analyses the events that occur in a given period of time called mining cycles. They consider a usage pattern suspicious if it is detected during different mining cycles. The user has to determine the duration of a cycle and the number of cycles to consider before issuing an alert. After each cycle the queue is emptied. The number of cycles multiplied with the duration of a cycle is called the mining window. After each mining window, the set of mined patterns is saved and then emptied.

3.2. Detecting Network Attacks from Wireless Connection Records

WiFi Miner [Rahman et al. 2010] finds infrequent patterns on pre-processed wireless connection records using an infrequent pattern mining algorithm based on Apriori. They proposed the Online Apriori-Infrequent algorithm which does not use the confidence value parameter and efficiently uses only frequent and rare patterns in a record to compute an anomaly score for the record to determine whether this record is anomalous or not on the fly. The algorithm has an Anomaly Score Calculation which assigns a score to each wireless packet.

Computationally the proposed Online Apriori-Infrequent algorithm improves the join and prune step of the traditional Apriori algorithm with a constraint. The constraint avoids joining itemsets not likely to produce frequent itemsets as their results, thereby improving efficiency and run times significantly. An anomaly score is assigned to each packet (record) based on whether the record has more frequent or infrequent patterns. Connection records with positive anomaly scores have more infrequent patterns than frequent patterns and are considered anomalous packets.

To test the application the authors created a wireless network with two other PCs where one was the victim and another one was the attacker PC. Within a five minutes time window they have captured around 19,500 wireless packets, which were generated as a result of some innocent activities. They gathered around 500 anomalous packets which contained different kinds of crafted attacks like passive attacks, active attacks, Man-In-the-Middle attack. Then, they launched these attacks from the attacker PC to the victim PC. They then tested the system with crafted intrusions and compared it with other two other benchmark systems and found our system to be more efficient.

3.3. Detecting Adverse Drug Reactions

Ji et al. [2013] proposed a framework to mine potential causal associations in electronic patient data sets where the drug-related events of interest occur infrequently. They developed and incorporated an exclusion mechanism that can effectively reduce

the undesirable effects caused by frequent events. Specifically, they proposed a novel interestingness measure, exclusive causal-leverage, based on a computational, fuzzy recognition-primed decision model. On the basis of this new measure, a data mining algorithm was developed to mine the causal relationship between drugs and their associated adverse drug reactions. The proposed exclusive causal-leverage measure looks for itemset pairs $< X, Y >$. It denotes that X has causality with Y. A large value indicates a stronger causal association. If its value is negative, it indicates a reverse causality. If a value close to zero it shows no causal association between X and Y.

To evaluate the algorithm, a dataset needs to be preprocessed to obtain different types of information: a list of all drugs in the database and the support count for each drug, and a list of all symptoms in the database and the support count for each symptom. The lists of drugs and symptoms are needed to form all possible drug-symptom pairs whose causal strengths will be assessed. In the experiment, the data was obtained from Veterans Affairs Medical Center in Detroit, Michigan. "Event" data such as dispensing of drug, office visits, and certain laboratory tests were retrieved for all the patients. The results showed that the exclusive causal-leverage measure outperformed traditional interestingness measures like risk ratio and leverage because of its ability to capture suspect causal relationships and exclude undesirable effects caused by frequent events.

3.4. Detecting Abnormal Learning Problems in Students from Educational Dataset

Romero et al. [2010] explore the extraction of rare association rules when gathering student usage data from a Moodle system. In the education domain, infrequent associations can be of great interest since they are related to rare but crucial cases. The rules produced can allow the instructor to determine frequent/abnormal learning problems that should be taken into account when dealing with students with special needs. Thus, this information could help the instructor to discover a minority of students who may need specific support with their learning process. The authors used the Apriori-Inverse [Koh and Rountree 2005] and Apriori-Rare [Szathmary et al. 2007] algorithms detailed in Section 4.1. In the experiments, the authors have evaluated the relation/influence between the on-line activities and the final mark obtained by the students.

4. DETECTING RARE RULES IN STATIC DATA

In this section we look at techniques that work on a static dataset. Classic pattern mining techniques, such as Apriori, rely on uniform minimum support. These algorithms either miss the rare but interesting rules or suffer from congestion in itemset generation caused by low support. Driven by such shortcomings, research has been carried out in developing new rule discovery algorithms to mine rare rules. Figure 2 shows the taxonomy of the current techniques.

There have been several approaches taken to ensure that rare items are considered during itemset generation. One of the approaches is association rule mining with *variable support threshold*. In this approach, each itemset may have a different support threshold. The support threshold for each itemset is dynamically lowered to allow some rare items to be included in the rule generation. Section 4.1 discuss some of the research [Liu et al. 1999a; Yun et al. 2003; Tao et al. 2003; Wang et al. 2003; Seno and Karypis 2001; Koh and Rountree 2005; Pillai et al. 2013; Szathmary et al. 2007; Sadhasivam and Angamuthu 2011; Hoque et al. 2013].

A uniform minimum support threshold is not effective for datasets with a skewed distribution because uniform minimum support thresholds tend to generate many trivial patterns or miss potential low-support patterns. Hence another approach uses association rule mining *without support threshold*, but it usually introduces another con-

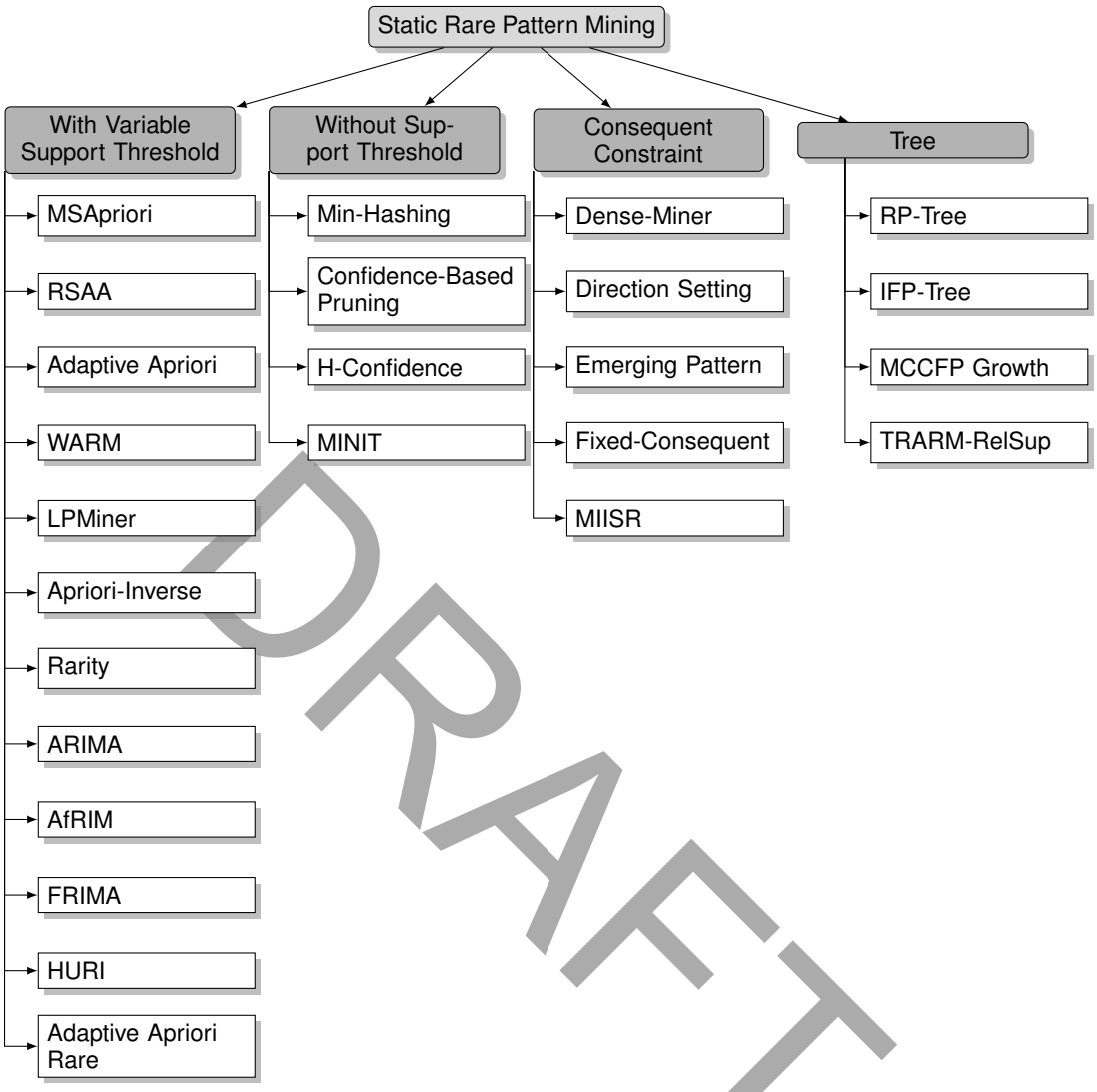

Fig. 2. Static Rare Pattern Mining

straint to solve the rare item problem. In Section 4.2 we discuss some of the algorithms that focus on mining low support rules without using support thresholds [Cohen et al. 2001; Wang et al. 2001; Xiong et al. 2003; Haglin and Manning 2007].

Another way to encourage low-support items to take part in candidate rule generation is by imposing item constraints. For example, it is possible to provide a list of those items that may or may not take part in a rule and then modify the mining process to take advantage of that information [Srikant et al. 1997; Ng et al. 1998; Bayardo et al. 2000; Rahal et al. 2004; Li et al. 1999]. One of the restrictions that may be imposed is called *consequent constraint-based rule mining* discussed in Section 4.3.

Recently tree based approaches have been developed to find rare association rules. Section 4.4 discuss some of the algorithms focusing on rare pattern mining using *tree*

based approaches [Tsang et al. 2011; Gupta et al. 2012; Kiran and Krishna Reddy 2012; Lavergne et al. 2012].

4.1. With Variable Support Threshold

In this approach, the minimum support threshold is lowered using different techniques to allow some rare items to be included in rule generation. Each itemset is given a different minimum support threshold. The support of a rare itemset would be lower than frequent itemsets. Thus the minimum support threshold for the rare itemset is set lower to allow the itemset to be considered.

For example the sales of *espresso machine* may appear in 1% of the time in a departmental store's transaction and the sales of *coffee grinder* appears in about 1.2%. Suppose both *coffee grinder* and *espresso machine* appears 0.8%. Given that we derived that the rule *espresso machine* $\rightarrow$ *coffee grinder* appears with confidence of 80%. If the minimum support threshold for was set to 1%, we would not be able to detect this potentially interesting rule. To overcome using the variable support threshold, each item *espresso machine* or *coffee grinder* may be given a different minimum support threshold. For example if we set the minimum support threshold for *coffee grinder* as 0.5% and the minimum support threshold for *espresso machine* as 0.8%, then the itemsets {*espresso machine*, *coffee grinder*} will be considered for rule generation. Another approach would be using two different thresholds such that we consider items that are below a minimum support threshold, thus using this variation we can find the rule *espresso machine* $\rightarrow$ *coffee grinder* as both the items in the rules would be considered in the during rule generation.

4.1.1. Multiple Minimum Support Thresholds (MSApriori). Liu et al. [1999a] deal with the rare item problem by using multiple minimum support thresholds. In their research premise, they note that some individual items can have such low support that they cannot contribute to rules generated by Apriori, even though they may participate in rules that have very high confidence. They overcome this problem with a technique whereby each item in the database can have its own minimum item support (MIS). By providing a different MIS for different items, a higher minimum support can be set for rules that involve frequent items and lower minimum support for rules that involve less frequent items. The minimum support of an itemset is the lowest MIS among those items in the itemset. For example, let MIS(i) denote the MIS value of item i. The minimum support of a rule R is the lowest MIS value of items in the rule. A rule, R: $AB \rightarrow C$ satisfies its minimum support if the rule has an actual support greater or equal to: $\min(\text{MIS}(A), \text{MIS}(B), \text{MIS}(C))$.

However, consider four items in a dataset, A, B, C, and D with MIS(A) = 10%, MIS(B) = 20%, MIS(C) = 5%, and MIS(D) = 4%. If we find that $\{A, B\}$ has 9% support at the second iteration, then it does not satify $\min(\text{MIS}(A), \text{MIS}(B))$ and is discarded. Then potentially interesting itemsets $\{A, B, C\}$ and $\{A, B, D\}$ will not be generated in the next iteration. By sorting the items in ascending order of their MIS values, the minimum support of the itemset never decreases as the length of an itemset grows, making the support a more general support constraint. In general, it means that a frequent itemset is only extended with an item having a higher (or equal) MIS value. The MIS for each data item i is generated by first specifying LS (the lowest allowable minimum support), and a value $\beta, 0 \leq \beta \leq 1.0$. MIS($i$) is then set according to the following formula:

$$MIS(i) = \max(\beta.\textbf{sup}(i, D), LS)$$

The advantage of the MSApriori algorithm is that it has the capability of finding some rare-itemset rules. However, the actual criterion of discovery is determined by the user's value of β rather than the frequency of each data item.

4.1.2. Relative Support Apriori (RSAA). Determining the optimal value for β could be tedious especially in a database with many items where manual assignment is not feasible. Thus [Yun et al. 2003] proposed the RSAA algorithm to generate rules in which significant rare itemsets take part, without any user defined thresholds. This technique uses *relative support*: for any dataset, and with the support of item i represented as sup(i, D), relative support ($RSup$) is defined as:

$$RSup(\{i_1, i_2, \ldots, i_k\}, D) = \frac{\textbf{sup}(\{i_1, i_2, \ldots, i_k\}, D)}{\min(\textbf{sup}(i_1, D), \textbf{sup}(i_2, D), \ldots, \textbf{sup}(i_k, D))}$$

Thus this algorithm increases the support threshold for items that have low frequency and decreases the support threshold for items that have high frequency.

Using a non-uniform minimum support threshold leads to the problem of choosing a suitable minimum support threshold for a particular itemset. Each item within the itemset may have a different minimum support threshold. MSApriori and RSAA sort the items within the itemset in non-decreasing order of support. Here the support of a particular itemset never increases and the minimum support threshold never decreases as the itemset grows.

4.1.3. Adaptive Apriori. Wang et al. [2003] proposed Adaptive Apriori which has a variable minimum support threshold. Adaptive Apriori introduces the notion of support constraints (SC) as a way to specify general constraints on minimum support. In particular, they associate a support constraint with each of the itemsets. They consider support constraints of the form $SC_i(B_1, \ldots, B_s) \geq \theta_i$, where $s \geq 0$. Each B_j, called a bin, is a set of items that need not be distinguished by the specification of minimum support. θ_i is a minimum support in the range of $[0 \ldots 1]$, or a function that produces minimum support. If more than one constraint is applicable to an itemset, the constraint specifying the lowest minimum support is chosen. For example, given $SC_1(B_1, B_3) \geq 0.2$, $SC_2(B_3) \geq 0.4$, $SC_3(B_2) \geq 0.5$, and $SC_0() \geq 0.9$, if an itemset contains $\{B_1, B_2, B_3\}$, the minimum support used is 0.2. However, if the itemset only contains $\{B_2, B_3\}$ then the minimum support is 0.4.

The key idea of this approach is to $push$ the support constraint following the dependency chain of itemsets in the itemset generation. For example, we want to generate itemset $\{B_0B_1B_2\}$, which uses SC_3, which is 0.5. $\{B_0B_1B_2\}$ is generated by using $\{B_0B_1\}$ with SC_0 and $\{B_1B_2\}$ with SC_3. This requires the minsup, which is 0.5 from $\{B_0B_1B_2\}$, to be pushed down to $\{B_0B_1\}$, and then pushed down to $\{B_0\}$ and $\{B_1\}$. The pushed minimum support is 0.5, which is lower than the specified minsup for $\{B_0B_1\}$, $\{B_0\}$, or $\{B_1\}$, which is 0.9. The pushed minimum support of each itemset is forced to be equal to the support value corresponding to the longest itemset.

4.1.4. Weighted Association Rule Mining (WARM). We can determine the minimum support threshold of each itemset by using a weighted support measurement. Each item or itemset is assigned a weight based on its significance. Itemsets that are considered interesting are assigned a larger weight. Weighted association rule mining (WARM) [Tao et al. 2003] is based on a weighted support measurement with a weighted downward closure property. They propose two types of weights: item weight and itemset weight. Item weight $w(i)$ is assigned to an item representing its significance, whereas itemset weight $w(X)$ is the mean of the item weight. The goal of using weighted support is to make use of the weight in the mining process and prioritise the selection of targeted itemsets according to their significance in the dataset, rather than by their frequency alone. The weighted support of an itemset can be defined as the product of the total weight of the itemset (sum of the weights of the items) and the weight of the fraction of transactions that the itemset occurs in. In WARM, itemsets are no longer

simply counted as they appear in a transaction. The change in the counting mechanism makes it necessary to adapt the traditional support to a weighted support. An itemset is significant if its support weight is above a pre-defined minimum weighted support threshold. Tao et al. [2003] also proposed a weighted downward closure property as the adjusted support values violate the original downward closure property in Apriori. The rules generated in this approach rely heavily on the weights used. Thus to ensure the results generated are useful, we have to determine a way to assign the item weights effectively.

4.1.5. LPMiner. Previous approaches vary the minimum support constraint by using a particular weighting method using either the frequency or significance of the itemsets. Here we will discuss a different approach, LPMiner [Seno and Karypis 2001], which varies the minimum support threshold. It uses a pattern length-decreasing support constraint which tries to reduce support so that smaller itemsets which have higher counts are favoured over larger itemsets with lower counts. Seno and Karypis propose a support threshold that decreases as a function of itemset length. A frequent itemset that satisfies the length decreasing support constraint can be frequent even if the subsets of the itemset are infrequent. Hence the downward closure property does not hold. To overcome this problem, they developed a property called smallest valid extension (SVE). In this property, for an infrequent itemset to be considered it must be over a minimum pattern length before it can potentially become frequent. Exploiting this pruning property, Seno and Karypis propose LPMiner based on the FP-tree algorithm [Han et al. 2000]. This approach favours smaller itemsets; however, longer itemsets could be interesting, even if they are less frequent. In order to find longer itemsets, one would have to lower the support threshold, which would lead to an explosion of the number of short itemsets found.

4.1.6. Apriori-Inverse. Apriori-Inverse proposed by Koh and Rountree [2005] is used to mine rare rules which they called perfectly sporadic 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 the Aprori-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 in order for them to be used for candidate generation.

Perfectly sporadic rules consist only of items falling below the maxsup threshold. They may not have any subsets with support above *maxsup*. They define perfectly sporadic rules as the following $A \rightarrow B$ in transaction database D is perfectly sporadic for maxsup s, and minconf c if and only if:

$$\text{conf}(A \rightarrow B, D) \geq c, \quad \text{and } \forall x : x \in (AB), \quad \text{sup}(x, D) < s$$

That is, support must be *under* maxsup and confidence must be *at least* minconf, and no member of the set of AB may have support above maxsup. Perfectly sporadic rules thus consist of antecedents and consequents that occur rarely (that is, less often than maxsup) but, when they do occur, tend to occur together (with at least minconf confidence). For instance, suppose we had an itemset AB with $\text{sup}(A, D) = 0.12$, $\text{sup}(B, D) = 0.12$, and $\text{sup}(AB, D) = 0.12$, with maxsup $= 0.12$ and minconf $= 0.75$. Both $A \rightarrow B$ (confidence $= 1.0$) and $B \rightarrow A$ (confidence $= 1.0$) are perfectly sporadic in that they have low support and high confidence.

While this is a useful definition of a particularly interesting type of rule, it certainly does not cover all cases of rules that have support lower than maxsup. For instance, suppose we had an itemset AB with $\text{sup}(A, D) = 0.12$, $\text{sup}(B, D) = 0.16$, and $\text{sup}(AB, D) = 0.12$, with maxsup $= 0.12$ and minconf $= 0.75$. Both $A \rightarrow B$ (confidence

$= 1.0$) and $B \rightarrow A$ (confidence $= 0.75$) are sporadic in that they have low support and high confidence, but neither is perfectly sporadic, due to B's support being too high. 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.

4.1.7. A Rare Itemset Miner Algorithm (ARIMA). Szathmary et al. [2007] proposed two algorithms that can be used together to 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 their subsets; minimal rare itemset (mRI), 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.

Their approach mines rare itemsets using three algorithms: Apriori-Rare, MRG-Exp and ARIMA. Apriori-Rare is a modification of the Apriori algorithm used to mine frequent itemsets. Apriori-Rare generates a set of all minimal rare itemsets, that correspond to the itemsets usually pruned by the Apriori algorithm when seeking for frequent itemsets. The MRG-Exp algorithm, finds all mRIs by using MGs for candidate generation in each layer in a bottom up fashion. The MRIs 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 anti-monotonic property. ARIMA takes as input the set of rare mRIs and generates the set of all rare itemsets split into two sets: the set of rare itemsets having a zero support and the set of rare itemsets with non-zero support. Rare itemsets are generated by growing each itemset in the mRI set. In fact, if an itemset is rare then any extension of that itemset will result a rare itemset. To make the distinction between zero and non-zero support itemsets, the support of each extended itemset is calculated. Essentially, ARIMA generates 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, which represents the border above which there are only zero rare itemsets.

4.1.8. Apriori Based Framework for Rare Itemset Mining (AfRIM). Adda et al. [2007] proposed AfRIM that uses a top-down approach. 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. Those that remain after pruning are rare itemsets. Note that AfRIM traverses the search space from the very top and examines itemsets that have zero support, which may be inefficient.

4.1.9. Rarity. Troiano et al. [2009] notes that rare itemsets are at the top of the search space like AfRIM, so that bottom-up algorithms must first search through many layers of frequent itemsets. In order to avoid this, Troiano et al. proposed the Rarity algorithm that begins by identifying the longest transaction within the database and uses it 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 candidates, 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.

4.1.10. Frequent Rare Itemset Mining Algorithm (FRIMA). FRIMA was proposed by Hoque et al. [2013] uses bottom-up approach and attempts to find the frequent as well as rare itemsets based on the downward closure property of Apriori. FRIMA takes a dataset and minsup as input and finds frequent and rare itemsets as outputs. FRIMA initially scans the dataset once and finds all the frequent and rare single itemsets. Based on the support count it categorizes the single itemsets as zero itemsets having support count zero, frequent itemsets having support count greater than minsup and rare itemsets having support count less than minsup. The algorithm then generates three candidate lists. The first candidate list is generated from the frequent single-itemset, second candidate list is generated from the rare itemset and the third list is generated by combining frequent and rare itemsets. Now, these three lists are combined to make one single list and make one database scan to find the zero, frequent and rare itemsets of size 2. These frequent and rare 1 and 2-itemsets are maintained in a dynamic structure and used towards generation of frequent and rare itemsets greater than 2.

4.1.11. Mining High Utility Rare Itemsets (HURI). High Utility Rare Itemset Mining (HURI) algorithm was proposed by Pillai et al. [2013] to generate high utility rare itemsets of users' interest. HURI is a two-phase algorithm, Phase 1 generates rare itemsets and Phase 2 generates high utility rare itemsets, according to users' interest. In the first phase, the key idea was to generate rare itemsets are generated by considering those itemsets which have support value less than the maximum support threshold using Apriori-Inverse [Koh and Rountree 2005]. In the second phase, the itemsets found in Phase 1 are pruned based on the utility value. Here high utility rare itemsets having utility value greater than the minimum utility threshold are generated.

4.1.12. Automated Apriori-Rare. Sadhasivam and Angamuthu [2011] proposed a method that adopts both Apriori and MSApriori algorithms for frequent and rare item generation called Automated Apriori-Rare. It mines frequent items belonging to three different item groups namely Most interesting Group (MiG), Somewhat interesting Group (SiG), and Rare interesting Group (RiG). MiG and RiG use calculated levelwise automated support thresholds like Apriori, whereas RiG uses item-wise support thresholds like MSApriori. The itemsets which have support greater than the average support of items in that level are added to the MiG at that level.

The itemsets which have less support than average support in a level may turn to be somewhat interesting. Itemsets which have lower support than the average support are noted as a somewhat interesting itemset. The threshold used for filtering the SiG is known as MedianSup. Rare interesting items are the items with less support and high confidence. The itemsets which have less support than AvgSup and MedianSup should be considered when searching for interesting rare itemsets. The transactions which consist of rare itemsets should be separated into a group rare items transactions and the rare itemsets are found by scanning the itemsets in that group. The remaining transactions are used to find the MiGs and SiGs.

4.1.13. Observation. The approaches in this section try to vary the support constraint in some fashion to allow some rare items to be included in frequent itemset generation. MIS, RSAA, Adaptive Apriori, WARM, ARIMA, FRIMA, and AfRIM all use a weighting method of some form to adapt or customize the support threshold, whereas LPMiner uses a pattern length decreasing support constraint and Apriori-Inverse, HURI, and FRIMA constrain the rules generated using two different support bounds. Like Apriori and MSApriori, RSAA, Adaptive Apriori, WARM, LPMiner, MCCFP-growth, AfRIM, Automated Apriori-Rare are exhaustive in their generation of rules, and so it is necessary to spend time looking for rules with high support and high confidence. If the varied minimum support value is set close to zero, they will take a similar amount

of time to that taken by Apriori to generate low-support rules in amongst the high-support rules. These methods generate all rules that have high confidence and high support. In order to include rare items, the minsup threshold must be lower, which consequently generates an enormous set of rules consisting of both frequent and infrequent items.

4.2. Without Support Threshold

The previous section discussed some of the approaches that use a variable support threshold to include some rare items in rule generation. But to ensure each rare item is considered, the minimum support threshold must still be pushed low, resulting in a combinatorial explosion in the number of rules generated. In order to overcome this problem, some researchers have proposed to remove the support-based threshold. Instead they use another constraint such as similarity or confidence-based pruning.

Using the example of rule *espresso machine* $\rightarrow$ *coffee grinder* where the confidence is 80%. We obtained two other rules *coffee tamper* $\rightarrow$ *coffee grinder* with confidence is 80% and *coffee tamper, espresso machine* $\rightarrow$ *coffee grinder* with confidence is 90%. Using these techniques without minimum support such as confidence pruning we can see note that both the rules stated above would be valid. However if we obtained two other rules *decalcifying tablets* $\rightarrow$ *coffee grinder* with confidence is 80% and *decalcifying tablets, espresso machine* $\rightarrow$ *coffee grinder* with confidence is 70%; the second rule *decalcifying tablets, espresso machine* $\rightarrow$ *coffee grinder* would not meet the confidence constraints. The confidence of the second rule is lower than the first rule thus not providing us with additional information.

4.2.1. Min-Hashing and its variations. Variations on the Min-Hashing technique were introduced by Cohen et al. [2001] to mine significant rules without any constraint on support. Transactions are stored as a 0/1 matrix with as many columns as there are unique items. Rather than searching for pairs of columns that have high support or high confidence, Cohen et al. [2001] search for columns that have high *similarity*, where similarity is defined as the fraction of rows that have a 1 in both columns when they have a 1 in either column. This is also known as *latent semantic indexing* in Information Retrieval. Although this is easy to do by brute force when the matrix fits into main memory, it is time-consuming when the matrix is disc-resident. Their solution is to compute a hashing signature for each column of the matrix in such a way that the probability that two columns have the same signature is proportional to their similarity. After signatures are calculated, candidate pairs are generated, and then finally checked against the original matrix to ensure that they do indeed have strong similarity. It should be noted that the hashing solution will produce many rules that have high support and high confidence, since only a *minimum* acceptable similarity is specified. It is not clear whether the method will extend to rules that contain more than two or three items, since $\binom{m}{k}$ checks for similarity must be done, where m is the number of unique items in the set of transactions, and k is the number of items that might appear in any one rule.

Removing the support requirement entirely is an elegant solution, but it comes at a high cost of space: for n transactions containing an average of k items over m possible items, the matrix will require $n \times m$ bits, whereas the primary data structure for Apriori-based algorithms will require $n \times \log_2 m \times k$ bits. Note that itemsets with low similarity may still produce interesting rules. For example, in dataset D we are given items A, B, and C where $\text{sup}(A, D) = 0.50$, $\text{sup}(B, D) = 0.50$, and $\text{sup}(C, D) = 0.50$. If $\text{sup}(A\neg BC, D) = 0.25$ and $\text{sup}(\neg ABC, D) = 0.25$, $R_1 : A \rightarrow B$ and $R_2 : B \rightarrow C$, it should be interesting but no pairs of items are strongly similar to each another.

4.2.2. Confidence-Based Pruning. Another constraint known as *confidence-based* pruning was proposed by Wang et al. [2001]. It finds all rules that satisfy a minimum confidence, but not necessarily a minimum support threshold. They call the rules that satisfy this requirement *confident rules*. The problem with mining confident rules is that, unlike support, confidence does not have a downward closure property. Wang et al. [2001] proposed a confidence-based pruning that uses the confidence requirement in rule generation. Given three rules $R_1 : A \rightarrow B$, $R_2 : AC \rightarrow B$, and $R_3 : AD \rightarrow B$, R_2 and R_3 are two specialisations of R_1, having additional items C and D. C and D are exclusive and exhaustive in the sense that exactly one will hold up in each itemset but they will not appear together in the same itemset. The confidence of R_2 and R_3 must be greater than or equal to that of R_1. We can prune R_1 if neither R_2 nor R_3 is confident. This method has a universal-existential *upward* closure. This states that if a rule of size k occurs above the given minimum confidence threshold, then for every other attribute not in the rule (C and D in the given example), some specialisation of size $k + 1$ using the attribute must also be confident. They exploit this property to generate rules without having to use any support constraints.

4.2.3. H-Confidence. Xiong et al. [2003] try to improve on the previous confidence-based pruning method. They propose the *h-confidence* measure to mine *hyperclique* patterns. A hyperclique pattern is a type of association containing objects that are highly affiliated with each other; that is, every pair of objects in a hyperclique pattern is guaranteed to have a cosine similarity (uncentered correlation coefficient) above a certain level. They show that *h-confidence* has a cross-support property which is useful for eliminating candidate patterns having items with widely different supports. The h-confidence of an itemset $P = \{i_1, i_2, \ldots, i_m\}$ in a database D denoted by *hconf(P, D)*, is a measure that reflects the overall affinity among items within the itemset.

A hyperclique pattern P is a strong-affinity association pattern because the presence of any item $x \in P$ in a transaction strongly implies the presence of $P\backslash\{x\}$ in the same transaction[1]. To that end, the *h-confidence* measure is designed specifically for capturing such strong affinity relationships. Nevertheless, even when including hyperclique patterns in rule generation, we can also miss interesting patterns. For example, an itemset $\{A, B, C\}$ that produces low confidence rules $A \rightarrow BC$, $B \rightarrow AC$, and $C \rightarrow AB$, but a high confidence rule $AB \rightarrow C$, would never be identified.

4.2.4. Minimally Infrequent Itemset (MINIT). Haglin and Manning [2007] designed an algorithm called MINIT (Minimally Infrequent Item set) for mining minimal infrequent items. The algorithm works by sorting and ranking items based on the support. At each step, the items having support less than minimum support are chosen and only the transactions containing those items are selected for further processing. Initially, a ranking of items is prepared by computing the support of each of the items and then creating a list of items in ascending order of support. Minimal τ-infrequent itemsets are discovered by considering each item in rank order, recursively calling MINIT on the support set of the dataset with respect to the item considering only those items with higher rank than item, and then checking each candidate minimal infrequent itemset against the original dataset. One mechanism that is used to consider only higher-ranking items in the recursion is to preserve a vector indicating which items remain feasible at each level of the recursion.

4.2.5. Observation. Similar to the approaches in the previous section, these algorithms suffer from the same drawback of generating all the frequent rules as well as the rare

[1]The $\backslash$ denotes exclusion, where $P\backslash\{x\}$ returns the pattern P excluding $\{x\}$.

rules. In most of these approaches we would need a post-pruning method to filter out the frequent rules or the trivial rules produced.

4.3. Consequent Constraint-Based Approach

Using a variable support threshold or no support threshold would generate frequent rules as well as rare rules. Here we will discuss an approach that tries to generate only rare rules. The research below imposes a restriction called *consequent constraint-based rule mining*. In this approach, an item constraint is used which requires mined rules to satisfy a given constraint. In this section we will discuss five slightly different methods that use this approach: Dense-Miner, DS (Direction Setting) rules, EP (Emerging Pattern), Fixed-Consequent ARM (Association Rule Mining), and MIISR (Mining Interesting Imperfectly Sporadic Rules).

Using this technique we are detecting rules with consequent items that are predetermined. For example, given a medical dataset we would like to detect symptoms that lead to a rare disease such as *Meningitis* that appears less than $1.38 \times 10^{-4}\%$ of the time in the population. We set the consequent of the rule as *Meningitis*. We then produce rules that fulfill this particular constraint. The items in the antecedent of the rule may be rare or frequent. In this example, we obtain the rule *stiff neck*, *rash*, *headaches*, *aversion to light* $\rightarrow$ *Meningitis*. In this particular rule, we can expect that the symptoms *stiff neck*, *rash*, *headaches* would occur relatively frequently in the dataset, whereas *aversion to light* would be a less common symptom. Together all four symptoms would point to *Meningitis*.

4.3.1. Dense-Miner. Bayardo et al. [2000] noted that the candidate frequent itemsets generated are too numerous in *dense* data, even when using an item constraint. A dense dataset has many frequently occurring items, strong correlations between several items, and many items in each record. Thus Bayardo et al. [2000] use a consequent constraint-based rule mining approach called Dense-Miner. They require mined rules to have a given consequent C specified by the user.

They also introduce an additional metric called *improvement*. The key idea is to extract rules with confidence above a *minimum improvement* value greater than any of the simplifications of a rule. A simplification of a rule is formed by removing one or more items from its antecedent. Any positive minimum improvement value would prevent unnecessarily complex rules from being generated. A rule is considered overly complex if simplifying its antecedent results in a rule with higher confidence. The *improvement* of a rule $A \rightarrow C$ is defined as the minimum difference between its confidence and the confidence of any proper sub-rule with the same consequent.

$$\text{improvement}(A \rightarrow C, D) = \text{conf}(A \rightarrow C, D) - \max\{\text{conf}(A' \rightarrow C, D) | A' \subset A\}$$

If the *improvement* of a rule is greater than 0, then removing any non-empty combination of items from the antecedent will lower the confidence by at least the improvement. Thus every item and every combination of items present in the antecedent of a rule with a large improvement are important contributors to its predictive ability. In contrast, it is considered undesirable for the *improvement* of a rule to be negative, as it suggests that the extra elements in the antecedent detract from the rule's predictive power.

In their approach, rules with consequent C must meet the user-specified minimum support, confidence, and improvement thresholds. A rule is considered to have a large improvement if its improvement is at least the specified minimum improvement. Dense-Miner's advantage over previous systems is that it can utilise rule constraints to efficiently mine consequent-constrained rules at low support.

4.3.2. Direction Setting. Liu et al. [1999b] proposed a rule pruning technique similar to Dense-Miner. Their approach uses a chi-square significance test which tries to prune out spurious and insignificant rules that appear by coincidence rather than random association. They also introduce *direction setting* rules. If the antecedent and consequent of a rule are positively associated, we say that the direction is 1, whereas if they are negatively associated we say that the direction is -1. If the antecedent and consequent are independent, then the direction is 0. The *expected directions* for $R : AB \rightarrow C$ is the set of directions for both $R_1 : A \rightarrow C$ and $R_2 : B \rightarrow C$. If R_1 and R_2 have direction of -1 but R has the direction of 1, R would be considered surprising. The rule R is a direction setting (DS) rule if it has a positive association (direction of 1) and its direction is not an element of the set of *expected directions*. For R to be a DS rule, the direction for both R_1 and/or R_2 should not be 1.

4.3.3. Emerging Pattern. The Emerging Pattern (EP) method was proposed by Li et al. [1999]. Given a known consequent C, a dataset partitioning approach is used to find *top* rules, *zero-confidence* rules, and μ*-level confidence* rules. The dataset, D, is divided into sub-datasets D_1 and D_2; where D_1 consists of the transactions containing the known consequent and D_2 consists of transactions which do not contain the consequent. All items in C are then removed from the transactions in D_1 and D_2. Using the transformed dataset, EP then finds all itemsets X which occur in D_1 but not in D_2. For each X, the rule $X \rightarrow T$ is a *top* rule in D with confidence of 100%. On the other hand, for all itemsets, Z, that only occur in D_2, all transactions in D which contain Z must not contain C. Therefore $Z \rightarrow C$ has a negative association and is a *zero-confidence* rule. For μ*-level confidence* rules $Y \rightarrow C$ the confidences are greater than or equal to $1 - \mu$. The confidences of μ-level rules must satisfy:

$$\frac{\sup(Y, D_1)|D_1|}{\sup(Y, D_1)|D_1| + \sup(Y, D_2)|D_2|} \geq 1 - \mu$$

Note that $\sup(Y, D_1)|D_1|$ is the number of times itemset YC appears together in dataset D and $\sup(Y, D_1)|D_1| + \sup(Y, D_2)|D_2|$ is the number of times itemset Y appears in dataset D. This approach is considered efficient as it only needs one pass through the dataset to partition and transform it. Of course, in this method one must supply C.

4.3.4. Fixed-Consequent Association Rule Mining. Rahal et al. [2004] proposed a slightly different approach. They proposed a method that generates the highest support rules that matched the user's specified minimum without having to specify any support threshold. Fixed-consequent association rule mining generates *confident minimal* rules using two kinds of trees [Rahal et al. 2004]. Given two rules, R_1 and R_2, with confidence values higher than the confidence threshold, where R_1 is $A \rightarrow C$ and R_2 is $AB \rightarrow C$, R_1 is preferred, because the antecedent of R_2 is a superset of the antecedent of R_1. The support of R_1 is necessarily greater than or equal to R_2. R_1 is referred to as a *minimal* rule ("simplest" in the notation of [Bayardo et al. 2000]) and R_2 is referred to as a *non-minimal* rule (more complex). The algorithm was devised to generate the *highest* support rules that match the user specified minimum confidence threshold without having the user specify any support threshold.

4.3.5. Mining Interesting Imperfectly Sporadic Rules (MIISR). Koh et al. [2006] proposed an approach to find rare rules with candidate itemsets that fall below a maxsup (maximum support) level but above a minimum absolute support value. They introduced an algorithm called Apriori-Inverse to find sporadic rules efficiently, for example, a rare association of two common symptoms indicating a rare disease. They later proposed another approach called MIISR. In that approach, the consequent of these rules is an

item below maxsup threshold and the antecedent has support below maxsup but may consist of individual items above maxsup. In both approaches they use minimum absolute support (minabssup) threshold value derived from an inverted Fisher's exact test to prune out noise. At the low levels of co-occurrences of candidate itemsets that need to be evaluated to generate rare rules, there is a possibility that such co-occurrences happen purely by chance and are not statistically significant. The Fisher's exact test provided a statistically rigorous method of evaluating significance of co-occurrences and was thus an integral part of their approach.

4.3.6. Observation. These algorithms are only useful when we have prior knowledge that a particular consequent is of interest. Most rare items occur infrequently and thus may go undetected. This makes it unsuitable for generating rare item rules efficiently because we want to generate rules without needing prior knowledge of which consequents ought to be interesting.

4.4. Tree Based Approach

All of the above algorithms use the fundamental generate-and-test approach used in Apriori, which has potentially expensive candidate generation and pruning steps. In addition, these algorithms attempt to identify all possible rare itemsets, and as a result require a significant amount of execution time.

The techniques in this section produces rules which are from subsets of the techniques discussed from Sections 4.1 to 4.3. The main difference is that the techniques proposed in this section use a tree based data structure similar to FP-growth [Han et al. 2000] that removes the need for expensive candidate generations.

4.4.1. Rare Pattern Tree (RP-Tree). RP-Tree algorithm was proposed by Tsang et al. [2011] as a solution to these issues. RP-Tree avoids the expensive itemset generation and pruning steps by using a tree data structure, based on FP-Tree, to find rare patterns. RP-Tree finds rare itemsets using a tree structure, using a two-pass approach. RP-Tree performs one database scan to count item support. 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.

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 focuses 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. As RP-Tree uses a two pass approach it is not suitable in a data stream environment. Moreover RP-Tree is developed to generate rules produced from rare-item itemset, whereby the itemsets consist of only rare items or consist of both rare and frequent items. Thus RP-Tree only generates a specific type of rare rules. They would miss out on rules generated from itemsets that were rare but consist of individually frequent items.

4.4.2. Inverse FP- Tree (IFP-Tree). The Inverse FP-tree (IFP-tree) [Gupta et al. 2012] is a variant of the FP-tree. The techniques mines minimally infrequent itemsets (MII) which is an extension to MINIT [Haglin and Manning 2007]. An itemset is a MII if and only if it is infrequent and all its subsets are frequent. The algorithm produces the same type of rules in MINIT but the data structure used is different. The IFP-min algorithm that used IFP-Tree recursively mines minimally infrequent itemsets by di-

viding the IFP-tree into two sub-trees: projected tree and residual tree. The projected database corresponds to the set of transactions that contains a particular item. A potential minimal infrequent itemset mined from the projected tree must not have any infrequent subset. The itemset itself is a subset since it is actually the union with the item of the projected tree that is under consideration. The IFP-min algorithm considers the projected tree of only the least frequent item (lf-item). A residual tree for a particular item is a tree representation of the residual database corresponding to the item, *i.e.*, the entire database with the item removed. Similar to the projected tree, the IFP-min algorithm considers the residual tree of only the lf-item. The use of residual trees reduces the computation time. It was shown that IFP-tree is more suitable for large dense datasets, whereas it does not work as well for small dense datasets compared to MINIT.

4.4.3. Maximum Constraint based Conditional Frequent Pattern-growth (MCCFP). Another multiple minimum support framework technique proposed was maximum constraint model by Kiran and Krishna Reddy [2012]. In this research, each item has a specified support constraint, called minimum item support (MIS). A minsup of a pattern is then represented with the maximal MIS value for all items contained in the pattern. Thus, each pattern can satisfy a different minsup depending upon the items within it. Their proposed model is based on an Apriori-like approach with granular computation of bit strings for frequent pattern mining. Their proposed approach called Maximum Constraint based Conditional Frequent Pattern-growth (MCCFP-growth), utilizes prior knowledge of the items MIS values which was user defined to generate frequent patterns on the transactional database using a single pass.

MCCFP-growth is an FP-growth-like approach. In MCCFP-growth, the initial tree is constructed from all items in the transaction dataset. MCCFP-growth uses multiple minsups to discover complete set of frequent patterns by using a single scan on the dataset. MCCFP-growth constructs a tree in descending order of items based on MIS value. Since MCCFP-growth constructs a tree using the descending order of items based on MIS, it requires relatively more memory. Despite this it is still more efficient as compared to Apriori based approaches because it avoids the combinatorial explosion of candidate generation and multiple scans on a transactional dataset.

4.4.4. Targeted Rare Association Rule Mining Using Itemset Trees and the Relative Support Measure (TRARM-RelSup). Lavergne et al. [2012] proposed a novel targeted association mining approach to rare rule mining using the itemset tree data structure (TRARM-RelSup). This algorithm combines the efficiency of targeted association mining querying with the capabilities of rare rule mining; this results in discovering a more focused, frequent and rare rules for the user, while keeping the complexity manageable. TRARM-RelSup combines the efficiency of the itemset tree for targeted association mining with the strategy of using relative support in order to efficiently find targeted rare association rules. Targeted association mining discovers rules based upon the interests of users and, therefore, can generate more focused results. The itemset tree data structure was developed to facilitate the targeted rule mining process.

4.4.5. Observation. Tree based approaches are more efficient than the other types of approaches as they do not require a candidate generation phase used in other approaches. However building the tree for rare pattern mining is more complicated.

4.5. Key Differences

In this section we highlight the key differences between the approaches. We look at the advantages of a particular approach compared to the other approaches in this section.

— **With Variable Support Threshold.** Most of the techniques in this approach produces a more general set of rules, as compared to techniques in the other approaches: without support threshold and consequent constraint-based approach. These techniques would often produce a large number of rules compared to the two other approaches, which works well in a general exploratory mining. Thus generating a large number of rules which provides more coverage over information that can be found in the dataset.
— **Without Support Threshold.** This approach normally adopts specific additional pruning constraints such as confidence based pruning. Using the additional constraints, rules generated are normally more actionable as compared to the more generalized rules in the previous approach. The techniques in this approach uses particular constraints to prune rules that it deems too trivial or do not provide additional information.
— **Consequent Constraint-Based Approach.** This approach works well when the users have existing knowledge of existing rare items to target, which means that this approach would be more efficient in terms of time complexity as it only produces a specific set of the rules.
— **Tree Based Approach.** The biggest advantage of tree based approach is that it is a variant of FP-Tree approaches which is a more efficient tactic of generating rare rules as the minimum support threshold has to be set low. Overall the techniques in this approach also assimilate a variation of the different constraints used in the other approaches

5. DETECTING RARE RULES IN DYNAMIC DATA

Ever since its inception, data stream mining has remained one of the more challenging problems within the data mining discipline. This is mainly due to the nature of data streams being continuous and unbounded in size as opposed to traditional databases where data is static and stable. Data from a wide variety of application areas ranging from online retail applications such as online auctions, telecommunications call data, credit card transactions, sensor data and climate data are a few examples of applications that generate vast quantities of data on a continuous basis. Data produced by such applications are highly volatile with new patterns and trends emerging on a continuous basis. The unbounded size of data streams is considered the main obstacle to processing data streams. As it is unbounded, it makes it infeasible to store the entire data on disk. Furthermore, we would want to process data streams near real time. This raises two issues. Firstly, a multi-pass algorithm cannot be used because the entire dataset would need to be stored before mining can commence. In addition, scanning the input data more than once is considered to be costly and inefficient. Secondly, obtaining the exact set of rules that includes both frequent and rare rules from the data streams is too expensive.

Furthermore it is not feasible to use static mining techniques, even if we ran it on a recent batch of data. Current static techniques require multiple passes. Due to the high speed nature of online data streams, they need to be processed as fast as possible; the speed of the mining algorithm should be faster than the incoming data rate, otherwise data approximation techniques, such as sampling, need to be applied which will decrease the accuracy of the mining results [Jiang and Gruenwald 2006]. The mining method on data streams needs to adapt to their changing data distribution; otherwise, it will cause a concept drifting problem. Thus, enforcing mining to the most recent batch of data, may put the technique at risk of unnecessarily mining and requiring additional system resources when there is no change in the distribution thus not yielding anymore interesting rules. Moreover if the batch size is set to large, a concept drift may have occurred which means that new rare patterns have emerge in the stream but we

would only detected it when mining is carried at the end of the batch of data, thus increasing the delay time taken to detect the new rules. We now formally define the notion of concept drift.

Definition 5.1 (*Drift Detection*). Let $S_1 = (t_1, t_2, ..., t_m)$ and $S_2 = (t_{m+1}, ..., t_n)$ with $0 < m < n$ represent two samples of instances from a stream with population means μ_1 and μ_2 respectively. Then the drift detection problem can be expressed as testing the null hypothesis H_0 that $\mu_1 = \mu_2$ that the two samples are drawn from the same distribution against the alternate hypothesis H_1 that they arrive from different distributions with $\mu_1 \neq \mu_2$.

In practice the underlying data distribution is unknown and a test statistic based on sample means needs to be constructed by the change detector M. If the null hypothesis is accepted incorrectly when a change also known as drift has occurred then a false negative is said to have taken place. On the other hand if M accepts H_1 when no change has occurred in the data distribution then a false positive is said to have occurred.

In data streams we can only look at the transactions within the stream once, thus, a one-pass approach is necessary. This rules out the possibility of building a tree based on the frequency of items within the data stream. The frequency of an item may change as the stream progresses. There are four scenarios which we need to consider in a stream [Tsang et al. 2011]:

Scenario 1. A frequent item x at time T_1 may become rare at time T_2.
Scenario 2. A rare item x at T_1 may become frequent at time T_2.
Scenario 3. A frequent item x at T_1 may remain frequent at time T_2.
Scenario 4. A rare item x at T_1 may remain rare at time T_2.

T_1 represents a point in time, and T_2 represents a future point in time after T_1. Moreover concepts may change over time in data streams, most current techniques in pattern mining for data streams do not detect these changes. Figure 3 illustrates a sketch of the taxonomy.

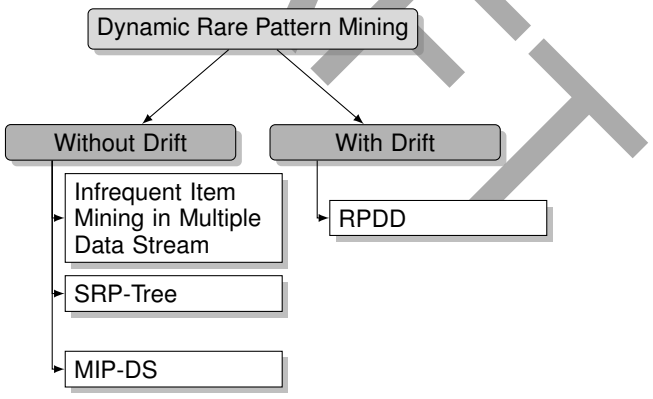

Fig. 3. Dynamic Rare Pattern Mining

5.1. Detecting Rare Rules in Dynamic Data without Drift

In this section we discuss approaches which detect rare rules in data stream without considering drifts. In these algorithms mining is are made at fixed intervals of time,

thus adapting to changes in data. The accuracy of such predictive models is hindered by the ability to set the correct interval.

5.1.1. Infrequent Item Mining in Multiple Data Streams. Saha et al. [2007] addressed two major issues in infrequent pattern mining. The first issue is scalability in terms of memory requirements and pattern selection over time horizons that vary in span. They developed a framework for detecting infrequent patterns in stream data which allows the discovery of patterns at different time resolutions. In general, the framework selects the infrequent patterns and stores them into a data structure that allows unique patterns across the stream to be stored at the top of the data structure. One very important aspect of this framework is its ability to handle multiple data streams and allows the discovery of infrequent patterns across multiple data streams.

As the underlying approach of the algorithm uses an online hierarchical structure, which allows us to map the concise description of the infrequent patterns within the data stream. Based on this, the structure allows the data stream to be incrementally processed.

5.1.2. Streaming Rare Pattern Tree (SRP-Tree). Streaming Rare Pattern Tree (SRP-Tree) [Huang et al. 2012] captures the complete set of rare rules in data streams using only a single scan of the dataset using a sliding window approach. SRP-Tree technique is a one-pass only strategy which is capable of mining rare patterns in a static database or in a dynamic data stream. In the case of mining data streams, this technique is also capable of mining at any given point of time in the stream and with different window and block sizes. The authors propose two variations of the technique where the first one involves using a novel data structure called the Connection Table which allows us to efficiently constrain the search in our tree, and keep track of items within the window. The second variation improves upon the first and requires the tree to go through restructuring before mining with FP-Growth.

5.1.3. Mining Minimal Infrequent Patterns from Data Streams (MIP-DS). A multi pass algorithm MIP-DS was proposed by Hemalatha et al. [2015] to mine minimal infrequent patterns. The proposed technique works in two phases. In Phase 1, a sliding window is used to bound the dataset and minimal infrequent patterns are mined from the observed stream. In Phase 2, minimal infrequent pattern deviation factor is computed for each minimal infrequent pattern, which captures the deviation of the support from a specified user defined minimum support threshold. The sum of minimal infrequent pattern deviation factor of minimal infrequent patterns for each transaction is computed and weighted using a transaction weighting factor.

5.2. Detecting Rare Rules in Dynamic Data with Drift

With the previous techniques we would run the risk of either mining too frequently which is an expensive task, or too infrequently where changes in the data stream are only detected ith significant latency. Finding the appropriate “sweet” spot for mining is close to an impossible task without incorporating a drift detector. The mining method of data streams needs to adapt to their changing data distribution; otherwise, its results may be inaccurate due to concept drift. In an environment of continuous, high speed, and changing data distribution characteristics, the analysis results of data streams often keep changing as well. Therefore, mining of data streams should be an incremental process to keep up with the changes in the data stream.

5.2.1. Rare Pattern Drift Detector (RPDD). The Rare Pattern Drift Detector (RPDD) [Huang et al. 2014] algorithm is designed to find rare pattern changes from unlabeled transactional data streams intended for unsupervised association rule mining. Overall the framework has two components. One of the components is the processing of stream

data. They detail the actual drift detection component of the algorithm. The authors introduce a novel M measure, a consolidated numerical measure that represents the state of item association. The selected rare items from the stream processing component will be individually tracked and monitored. Essentially, each selected item will have a separate process that monitors the item associations it has with other items in the stream using the M measure.

5.3. Key Differences

In this section we highlight the key differences between the two approaches. We analyse and compare the two approaches in this section.

— **Dealing with dynamic data without drift.** Most of the current stream data mining methods in this approach have user define parameters that are required before their execution, such as the size of the sliding window; however, most of them do not mention how users can adjust these parameters while the system is running. It is not desirable or feasible for users to wait until a mining algorithm to complete before they can reset the parameters. Moreover none of the current techniques in this approach addresses the concept drift problem, which is crucial effective stream mining.

— **Dealing with dynamic data with drift.** The technique in this approach allows users to detect changes of pattern in the stream when there is a change in data distribution. The main drawback of the current proposed technique is it focuses on detecting drifts or changes in data distribution of targeted rare items, and thus only monitoring a subset of possible rare rules.

6. EVALUATION OF THE DIFFERENT TECHNIQUES

In this section we compare and contrast the different techniques which we discussed above. We evaluate the techniques by comparing them based on five different characteristics: complexity, base algorithm, constraints, input data, and output itemsets. In

Table II. Summary of the Different Techniques

Technique	Complexity: Pass			Base Algorithm			Constraints			Input: Data		Output: Itemsets	
	Single	Two	Multiple	Candidate Generation	Variation of FP-Growth	Others	Mulitple Support Threshold	Pruning Constraints	Weights/ Ranking	Static	Stream	Frequent & Rare	Rare
MSApriori			x	x			x			x		x	
RSAA			x	x			x			x		x	
WARM			x	x					x	x		x	
LPMiner		x			x				x	x		x	
Apriori-Inverse			x	x			x			x			x
ARIMA			x	x			x	x		x		x	
AfRIM			x	x			x			x		x	
Rarity			x	x				x		x			x
FRIMA			x	x			x			x		x	
HURI			x	x					x	x		x	
Automated Apriori-Rare			x	x				x		x		x	
Min-Hashing			x			x		x		x		x	
Confidence-Based Pruning			x	x				x		x		x	
H-Confidence			x	x				x	x	x		x	
MINIT			x	x				x		x		x	
Dense-Miner			x	x				x		x		x	
Direction Setting			x	x				x		x		x	
Emerging Pattern			x	x				x		x		x	
MIISR			x	x				x		x			x
RP-Tree		x			x		x			x			x
IFP-Tree		x			x			x	x	x			x
MCCFP	x				x		x	x		x			x
TRARM-RelSup		x			x		x			x		x	
Infrequent Item Mining in Multiple Data Streams	x				x						x		x
SRP-Tree	x				x		x				x		x
MIP-DS			x		x				x		x		x
RPDD	x				x		x				x		x

complexity we look at the number of passes the technique carries out over the dataset. In base algorithm, we look at the underlying algorithm that the technique is based on such as Apriori-like candidate generation, variation of FP-Growth or inverted index. In constraints, we look at whether the technique requires multiple support thresholds, additional rule constraints, or additional weights. We then compare the type of input taken by the techniques and the type of itemsets produced by the technique. Table II shows the evaluation of the techniques. The 'x' marks the characteristic that is fulfilled by the technique.

7. RARE PATTERN MINING VS ANOMALY DETECTION

A similar area to rare association rules is anomaly detection. Both of these areas have some similar characteristics but are distinctly different. Anomalies are patterns in data that do not conform to a well-defined notion of normal behaviour. Like rare pattern mining, techniques in this category make the implicit assumption that normal instances are far more frequent than anomalies in the test data. Since frequent events are usually considered normal, anomalies are usually rare. Based on the normal events, data mining methods can generate, in an automated manner, the models that represents normalcy and identify deviations, which could be considered anomalies [Chandola et al. 2009].

We can compare anomaly detection and rare association rules based on three parts: input, method, and output. For anomaly detection, input is generally a collection of data instances (also referred as object, record, point, vector, pattern, event, case, sample, observation, entity). Each data instance can be described using a set of attributes. The attributes can be of different types such as binary, categorical or continuous. On contrary, rare association rules are normally generated from a categorical data set. However we can also represent the input in rare pattern mining as vectors, whereby, each itemset is represented as a vector, while each item corresponds to one dimension in a vector. The methods used in anomaly detections can be divided into supervised, semi-supervised and unsupervised techniques. The methods used in rare association rules are typically unsupervised. The outputs produced by anomaly detection techniques are normally one of the following two types: scores and labels. Rare pattern mining produces descriptive rules which represent dependencies. In order for anomaly detection techniques to generate patterns like association rules, we can use the join distribution and conditional distribution, both of which can be provided by statistical methods in some anomaly detection techniques.

8. CONCLUSIONS AND OPEN CHALLENGES

In this survey, our aim has been to provide a comprehensive review of rare pattern mining. This survey describes various problems associated with mining rare patterns and methods for addressing these problems. Overall the techniques can be categorized into two major groups: detection techniques for static data and detection techniques for dynamic data.

Following our taxonomy in Figure 2, we surveyed various static rare pattern mining techniques. The static detection methods were divided into four categories: (i) variable support threshold, (ii) without support threshold, (iii) consequent constraint-based approach, and (iv) tree based approach. The first category pushes constraints on to the minimum support threshold, whereas the second category uses statistical measures to pre-determine the support threshold. The third category pushes constraints onto the structure of the rule generated, defining the type of rules produced. The last category, uses a tree based approach which is a more efficient type of data structure in terms of performance. Figure 3 illustrates the dynamic rare pattern mining technique. The

dynamic detection methods were divided into two categories: (i) with drift detection and (ii) without drift detection.

8.1. Open Challenges

While a significant amount of research on rare mining is available, much of this work is still in its infancy. That is, there are no well-established, proven, methods for generally handling rare cases. We expect research on this topic to continue, and accelerate, as increasingly more difficult data mining tasks are tackled. In this part we provide a discussion of open challenges in the area.

— **Rare pattern mining on data streams.** While rare pattern mining has been explored in static data, there has been only a few techniques that worked in a dynamic stream environment. It is difficult to differentiate a rare pattern from an emerging pattern that may be frequent in the future. We are interested in the patterns that remain rare or infrequent over either the processed chunk of the data stream or the entire stream. For example, colds and flu cases tend to be less frequent at the beginning of autumn as compared to winter. Thus reporting that a rule such as *Vitamin* $\rightarrow$ *Codral (Cold and Flu Medication)* as rare would be premature as it is considered as an emerging trend and not rare. Current stream mining techniques are unable to signal differences between the two types of rules. Vice versa the patterns may also become frequent during a different time period and then become rare again.
— **Distinguishing noise from signal.** When we deal with rare patterns, we need to be able to distinguish between chance occurrences and real rare pattern mining. Noise is an inherent problem when we deal with rare occurrences of instances. This problem does not exist when we deal with frequent pattern mining.
— **Choosing the right time window.** Many algorithms for time evolving data require a time window for computation of normal/frequent patterns. One of the open questions is how to choose the window to discover the different rare patterns. if a window is set too small rare patterns may be obscured and not indistinguishable from noise, whereas if a window is set too large the rare patterns will only be found after the entire window has been processed, and causes a delay between when the rare pattern occurred and when it is found.
— **Scalable real time rare pattern mining.** One of the most important future challenges is to develop scalable approaches for streaming data. Specifically research should focus on algorithms that are sub-linear to the input or the very least linear.
— **Rare Pattern Mining in Probabilistic Datasets.** In uncertain transaction databases, the support of an itemset is uncertain; it is defined by a discrete probability distribution function, where each itemset has a frequentness probability. There has been research in detecting frequent itemsets mining in uncertain data [Aggarwal et al. 2009; Leung and Brajczuk 2010; Tong et al. 2012]. However, it would pay to harvest probabilistic rare rules as well. The problem of detecting probabilistic rare rules are by far more complicated as probability calculations for rare itemsets are less consistent compared to frequent itemsets.

ACKNOWLEDGMENTS

This work was partially supported by the University of Auckland (FRDF Grant 3702232) and University of Malaya and Ministry of Education (High Impact Research Grant UM.C/625/1/HIR/MOHE/FCSIT/14).

REFERENCES

Mehdi Adda, Lei Wu, and Yi Feng. 2007. Rare Itemset Mining. In *Proceedings of the*

Sixth International Conference on Machine Learning and Applications (ICMLA '07). IEEE Computer Society, Washington, DC, USA, 73–80.

Mehdi Adda, Lei Wu, Sharon White, and Yi Feng. 2012. Pattern Detection with Rare Item-set Mining. *CoRR* abs/1209.3089 (2012).

Charu C. Aggarwal, Yan Li, Jianyong Wang, and Jing Wang. 2009. Frequent Pattern Mining with Uncertain Data. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '09)*. ACM, New York, NY, USA, 29–38.

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining Association Rules Between Sets of Items in Large Databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data (SIGMOD '93)*. ACM, New York, NY, USA, 207–216.

Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th International Conference on Very Large Databases, VLDB '94*, Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). 487 – 499.

Roberto J. Bayardo, Rakesh Agrawal, and Dimitrios Gunopulos. 2000. Constraint-Based Rule Mining in Large, Dense Databases. *Data Mining and Knowledge Discovery* 4, 2/3 (2000), 217 – 240.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)* 41, 3, Article 15 (2009), 58 pages.

Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. 2001. Finding Interesting Association Rules Without Support Pruning. *IEEE Transactions on Knowledge and Data Engineering* 13 (2001), 64 – 78.

Ashish Gupta, Akshay Mittal, and Arnab Bhattacharya. 2012. Minimally Infrequent Itemset Mining using Pattern-Growth Paradigm and Residual Trees. *CoRR* abs/1207.4958 (2012).

David J. Haglin and Anna M. Manning. 2007. On Minimal Infrequent Itemset Mining. In *Proceedings of the 2007 International Conference on Data Mining, DMIN 2007, June 25-28, 2007, Las Vegas, Nevada, USA*, Robert Stahlbock, Sven F. Crone, and Stefan Lessmann (Eds.). CSREA Press, 141–147.

Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining Frequent Patterns Without Candidate Generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, SIGMOD '00*. ACM Press, 1 – 12.

C. Sweetlin Hemalatha, V. Vaidehi, and R. Lakshmi. 2015. Minimal infrequent pattern based approach for mining outliers in data streams. *Expert Systems with Applications* 42, 4 (2015), 1998 – 2012.

N. Hoque, B. Nath, and D.K. Bhattacharyya. 2013. An Efficient Approach on Rare Association Rule Mining. In *Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012)*, Jagdish Chand Bansal, Pramod Kumar Singh, Kusum Deep, Millie Pant, and Atulya K. Nagar (Eds.). Advances in Intelligent Systems and Computing, Vol. 201. Springer India, 193–203.

Zhongyi Hu, Hongan Wang, Jiaqi Zhu, Maozhen Li, Ying Qiao, and Changzhi Deng. 2014. *Discovery of Rare Sequential Topic Patterns in Document Stream*. Chapter 60, 533–541.

David Huang, Yun Sing Koh, and Gillian Dobbie. 2012. Rare Pattern Mining on Data Streams. In *Data Warehousing and Knowledge Discovery*, Alfredo Cuzzocrea and Umeshwar Dayal (Eds.). Lecture Notes in Computer Science, Vol. 7448. Springer Berlin Heidelberg, 303–314.

David Tse Jung Huang, Yun Sing Koh, Gillian Dobbie, and Russel Pears. 2014. Detect-

ing Changes in Rare Patterns from Data Streams. In *Advances in Knowledge Discovery and Data Mining*, Vincent S. Tseng, Tu Bao Ho, Zhi-Hua Zhou, Arbee L.P. Chen, and Hung-Yu Kao (Eds.). Lecture Notes in Computer Science, Vol. 8444. Springer International Publishing, 437–448.

Yan Huang, Jian Pei, and Hui Xiong. 2006. Mining Co-Location Patterns with Rare Events from Spatial Data Sets. *GeoInformatica* 10, 3 (2006), 239–260.

Yanqing Ji, Hao Ying, John Tran, Peter Dews, Ayman Mansour, and R. Michael Massanari. 2013. A Method for Mining Infrequent Causal Associations and Its Application in Finding Adverse Drug Reaction Signal Pairs. *IEEE Transactions on Knowledge and Data Engineering* 25, 4 (April 2013), 721–733.

Nan Jiang and Le Gruenwald. 2006. Research Issues in Data Stream Association Rule Mining. *SIGMOD Rec.* 35, 1 (March 2006), 14–19.

R.Uday Kiran and Polepalli Krishna Reddy. 2012. An Efficient Approach to Mine Rare Association Rules Using Maximum Items Support Constraints. In *Data Security and Security Data (Lecture Notes in Computer Science)*, LachlanM. MacKinnon (Ed.), Vol. 6121. Springer Berlin Heidelberg, 84–95.

Yun Sing Koh and Nathan Rountree. 2005. Finding Sporadic Rules Using Apriori-Inverse. In *PAKDD (Lecture Notes in Computer Science)*, Tu Bao Ho, David Cheung, and Huan Liu (Eds.), Vol. 3518. Springer, 97–106.

Yun Sing Koh, Nathan Rountree, and Richard O'Keefe. 2006. Mining Interesting Imperfectly Sporadic Rules. In *PAKDD (Lecture Notes in Computer Science)*, Wee Keong Ng, Masaru Kitsuregawa, Jianzhong Li, and Kuiyu Chang (Eds.), Vol. 3918. Springer, 473–482.

Jennifer Lavergne, Ryan Benton, and Vijay V. Raghavan. 2012. TRARM-RelSup: Targeted Rare Association Rule Mining Using Itemset Trees and the Relative Support Measure. In *Foundations of Intelligent Systems*, Li Chen, Alexander Felfernig, Jiming Liu, and Zbigniew W. Raś (Eds.). Lecture Notes in Computer Science, Vol. 7661. Springer Berlin Heidelberg, 61–70.

Gangin Lee, Unil Yun, Heungmo Ryang, and Donggyu Kim. 2015. Multiple Minimum Support-Based Rare Graph Pattern Mining Considering Symmetry Feature-Based Growth Technique and the Differing Importance of Graph Elements. *Symmetry* 7, 3 (2015), 1151.

Carson Kai-Sang Leung and Dale A. Brajczuk. 2010. Efficient Algorithms for the Mining of Constrained Frequent Patterns from Uncertain Data. *SIGKDD Explor. Newsl.* 11, 2 (May 2010), 123–130.

Jinyan Li, Xiuzhen Zhang, Guozhu Dong, Kotagiri Ramamohanarao, and Qun Sun. 1999. Efficient Mining of High Confidence Association Rules without Support Thresholds. In *Principles of Data Mining and Knowledge Discovery (Lecture Notes in Computer Science)*, Jan M. Žytkow and Jan Rauch (Eds.), Vol. 1704. Springer Berlin Heidelberg, 406–411.

Bing Liu, Wynne Hsu, and Yiming Ma. 1999a. Mining Association Rules with Multiple Minimum Supports. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 337 – 341.

Bing Liu, Wynne Hsu, and Yiming Ma. 1999b. Pruning and Summarizing the Discovered Associations. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '99*. ACM Press, 125 – 134.

Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. 1998. Exploratory mining and pruning optimizations of constrained associations rules. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data, SIGMOD '98*. ACM Press, 13 – 24.

Jyothi Pillai, O.P. Vyas, and Maybin Muyeba. 2013. HURI A Novel Algorithm for

Mining High Utility Rare Itemsets. In *Advances in Computing and Information Technology*, Natarajan Meghanathan, Dhinaharan Nagamalai, and Nabendu Chaki (Eds.). Advances in Intelligent Systems and Computing, Vol. 177. Springer Berlin Heidelberg, 531–540.

Imad Rahal, Dongmei Ren, Weihua Wu, and William Perrizo. 2004. Mining Confident Minimal Rules with Fixed-Consequents. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI'04*. IEEE Computer Society, 6 – 13.

Ahmedur Rahman, C.I. Ezeife, and A.K. Aggarwal. 2010. WiFi Miner: An Online Apriori-Infrequent Based Wireless Intrusion System. In *Knowledge Discovery from Sensor Data*, Mohamed Medhat Gaber, Ranga Raju Vatsavai, Olufemi A. Omitaomu, João Gama, Nitesh V. Chawla, and Auroop R. Ganguly (Eds.). Lecture Notes in Computer Science, Vol. 5840. Springer Berlin Heidelberg, 76–93.

Cristóbal Romero, José Raúl Romero, Jose María Luna, and Sebastián Ventura. 2010. Mining Rare Association Rules from e-Learning Data. *Educational Data Mining* (2010), 171–180.

Kanimozhi SC Sadhasivam and Tamilarasi Angamuthu. 2011. Mining rare itemset with automated support thresholds. *Journal of Computer Science* 7, 3 (2011), 394.

B. Saha, M. Lazarescu, and S. Venkatesh. 2007. Infrequent Item Mining in Multiple Data Streams. In *Seventh IEEE International Conference on Data Mining Workshops*. 569–574.

Masakazu Seno and George Karypis. 2001. LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing Support Constraint. In *Proceedings of the 2001 IEEE International Conference on Data Mining ICDM*, Nick Cercone, Tsau Young Lin, and Xindong Wu (Eds.). IEEE Computer Society, 505 – 512.

Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. 1997. Mining Association Rules with Item Constraints. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining, KDD '97*. 67 – 73.

Laszlo Szathmary, Amedeo Napoli, and Petko Valtchev. 2007. Towards Rare Itemset Mining. In *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence - Volume 01 (ICTAI '07)*. IEEE Computer Society, Washington, DC, USA, 305–312.

Feng Tao, Fionn Murtagh, and Mohsen Farid. 2003. Weighted Association Rule Mining using weighted support and significance framework. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03*. ACM Press, 661 – 666.

Yongxin Tong, Lei Chen, Yurong Cheng, and Philip S. Yu. 2012. Mining Frequent Itemsets over Uncertain Databases. *Proc. VLDB Endow.* 5, 11 (July 2012), 1650–1661.

Luigi Troiano, Giacomo Scibelli, and Cosimo Birtolo. 2009. A Fast Algorithm for Mining Rare Itemsets. In *Proceedings of the 2009 Ninth International Conference on Intelligent Systems Design and Applications (ISDA '09)*. IEEE Computer Society, Washington, DC, USA, 1149–1155.

Sidney Tsang, Yun Sing Koh, and Gillian Dobbie. 2011. RP-Tree: Rare Pattern Tree Mining. In *DaWaK (Lecture Notes in Computer Science)*, Alfredo Cuzzocrea and Umeshwar Dayal (Eds.), Vol. 6862. Springer, 277–288.

Ke Wang, Yu He, and David W. Cheung. 2001. Mining confident rules without support requirement. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*. ACM Press, 89 – 96.

Ke Wang, Yu He, and Jiawei Han. 2003. Pushing Support Constraints Into Association Rules Mining. *IEEE Transactions Knowledge Data Engineering* 15, 3 (2003), 642 – 658.

Gary M. Weiss. 2004. Mining with rarity: a unifying framework. *SIGKDD Exploration Newsletter* 6, 1 (2004), 7 – 19.

Hui Xiong, Pang-Ning Tan, and Vipin Kumar. 2003. Mining Strong Affinity Association Patterns in Data Sets with Skewed Support Distribution. In *ICDM*. IEEE Computer Society, 387 – 394.

Hyunyoon Yun, Danshim Ha, Buhyun Hwang, and Keun Ho Ryu. 2003. Mining Association Rules on Significant Rare Data Using Relative Support. *The Journal of Systems and Software* 67, 3 (15 September 2003), 181 – 191.