A New Approach for Clustering Alarm Sequences in Mobile Operators

Selcuk Sozuer
OBSS
Istanbul, Turkey
Email: selcuk.sozuer@obss.com.tr

Engin Zeydan
Türk Telekom Labs
Istanbul, Turkey
Email: engin.zeydan@turktelekom.com.tr

Cagri Etemoglu
Türk Telekom Labs
Istanbul, Turkey
Email: cagriozgenc.etemoglu@turktelekom.com.tr

***Abstract*—Telecom Networks produce huge amount of daily alarm logs. These alarms usually arrive from different regions and network equipments of mobile operators at different times. In a typical network operator, Network Operations Centers (NOCs) constantly monitor those alarms in a central location and try to fix issues raised by intelligent warning systems by performing a trouble ticketing based management system. In order to automate rule findings, different sequential rule mining algorithms can be exploited. However, the number of sequential rules and alarm correlations that can be generated by using these algorithms can overwhelm the NOC administrators since some of those rules are neither utilized nor reduced appropriately by the non-customized sequential rule mining algorithms. Therefore, additional efficient and intelligent rule identification techniques need to be developed depending on the characteristic of the data. In this paper, two new metrics that is inspired from document classification approaches are proposed in order to increase the accuracy of the sequential alarm rules. This approach utilizes new definition of identifying transactions as alarm features and clustering the alarms by their occurrences in built transactions. Experimental evaluations demonstrate that up to 61% accuracy improvements can be achieved through utilizing the proposed appropriate metrics compared to a sequential rule mining algorithm.**

***Keywords*—telecommunication, network management, clustering, alarm correlation, rule mining.**

I. Introduction

In a typical mobile operator, NOC receive many types of alarm messages due to faults/failures in one or many components or communication links in their network infrastructure. Moreover, the characteristics of alarm data constantly varies due to changes in network equipments, software and network load throughout day. For example, when a network device's battery level becomes low, many different forms of alarms can be generated which may be correlated with that specific event based on topology and time of the day. These triggered alarms form a structure or they belong to a special kind of a set. Most of the time, finding out event structures (sets) which represent a specific phenomenon is obtained through years of experience of NOC experts [1]. For this reason, finding rules that are reflecting the most effective ones can be a challenging task. Inside mobile operators, telecommunication experts create those composite sets intuitively by observing alarm log sequences in a long period of time, possibly months or even years and learning curve gets longer as the characteristics of network changes.

Our main goal in this paper is to study different weight methods for discovering clusters in a given alarm sequence data in order to increase the efficiency of the sequential rules and perform better alarm rule summarization compared to traditional sequential alarm rule mining methods such as RuleGrowth [2], FPGrowth [3]. Given a set of observed alarm sequences, it is important to identify the most critical alarms that are potentially causes of other sequential alarms. Heuristically, if a rule, say, {ACDC FAULT, MAINS BREAKDOWN WITH BATTERY BACK } => { BATTTERY LOW} (means showcasing that when the base station power goes down and the battery back up power warning occur, battery power low alert will arrive next) is found, since their occurrence depends on some phenomenon (possible a root cause event), these alarms are elements of that specific event and should be in the same cluster. Therefore, we are particularly interested in developing various weight vectors for efficient clustering of correlated alarms in alarm specific domain in order to increase the accuracy of existing sequential rules. Through proposed approach, by creating an automatized solution NOCs can be less dependent on the expert's experience and more reliable methodology can be followed in contrast to an intuitive one.

A. Related Work and Our Contribution

For alarm flow analysis inside telecommunication networks, alarm correlation problem has been discussed in different papers [4] [5] [6] [7] [8] [9] [10] (see also [11] for an overview). There are also various works that have studied temporal data mining (see [12] for a survey). Some of these papers have considered only sequential data mining techniques which find the most frequented event sequences in a sequence database [7] [13]. GSP [13] and WINEPI [7] were the first algorithms to apply Apriori algorithm [14] to find sequential association rules temporally. These algorithms require a user-defined sliding-time window duration to traverse the data. Most of the sequential data mining methods proposed so far are based on discovering temporal relations between events in discrete time series [13]. These methods are based on finding the frequencies of sequences of events and generating the rule candidates against the database [2]. Others have concentrated on prediction problems for sequential rule mining on several different application domains. Some of these approaches are based on rule correlation, fuzzy logic, coding correlation, Bayesian networks, artificial neural networks [2] [7] [15] [16] [17].

Exploiting clustering algorithms inside mobile operators

for alarm correlation problem has been studied in the literature in various papers [8] [18] [19] [20] [21] [22] [23] [24]. In [18], a clustering based alarm correlation approach leveraging alarms' sequential proximity is proposed. In [19], the authors are studying data mining algorithms in order to detect correlated alarm patterns in a GSM/DCS mobile telephone network. The authors in [20] are studying the problem of analyzing, interpreting and reducing the number of these alarms before trying to localize the faults leveraging clustering approaches. In [21], the authors remark about difficulty of applying clustering algorithms into alarm domain due to availability of categorical and numerical attributes and introduce modified adaptive resonance theory network algorithm for mining correlation rules of alarm messages. In [22], the authors propose an algorithm to identify faults by clustering alarms based on their identification and periodicity in order to considerably reduce the alarms. The authors in [23] have proposed a hybrid framework by allowing High Performance Clusters for alert detection in system logs. A forecasting system is proposed for high-severity system event messages before they occur by mining relevant event patterns from past problem occurrences in [24].

Compared to the above works, our main contribution is to highlight the potential of further possible exploitation of sequential alarm rules through proposed different weight metrics with clustering similar to the *term frequency and inverse document frequency (TF.IDF)* approach used for document classification/clustering [25]. On the operational domain of the data, a new approach is developed that defines transactions as a feature of every item (alarm) and new weight metrics for clustering algorithms in order to select sequential rules with higher accuracy. The success of the different weight metrics with clustering approaches are studied based on the *alarm and cluster rule match rate* measures for validation. In addition, the validation of the reduced rules' set based on the clustering approach with proposed different weight metrics are also compared.

The organisation of the remainder of paper is as follows: Section II introduces the system architecture and data model definitions. Section III discusses the proposed method as well as the new weight metrics that are utilized within clustering algorithms. Section IV discusses the experimental results and accuracy evaluations. Finally, Section V gives the conclusions.

II. System Architecture and Model

The main goal of this section is to find out reasonable features that can represent alarms for clustering. First, an overview of the proposed AlarmAVEA architecture is briefly discussed. Then, the data model which will be used as input in the clustering process is described. Finally, by assuming every transaction to be a feature in our approach, two weight metrics, *Term Frequency (TF)* and *Inverse Item Frequency (IIF)* are introduced for clustering the alarms.

A. AlarmAVEA System Architecture

The architecture diagram showing the proposed AlarmAVEA application with clustering unit is depicted in Fig. 1. The proposed architecture is intended for offline analysis. In this architecture, there are four main units: *Alarm Log Management Unit*, *Hadoop Platform Management Unit*, *Processing Unit* and *Visualization Component*. *Alarm Log Management Unit* is the main data entry point of the system which handles the alarm log management. It has two main functionality: First, alarm logs are collected from Operations Support Systems (OSS) interface into a local directory through offline log collector, and then these alarm logs are loaded into the Hadoop Distributed File System (HDFS) in batch mode. *Hadoop Platform Management Unit* is responsible for storing the raw data and processing the raw data through Offline Data Processing (Extract, Transform and Load (ETL)) (e.g. cleansing, parsing, formatting, etc purposed). It is one of the main component of the system for storing and processing the alarm data in an effective manner. The *Processing Unit* consists of a clustering module which takes the raw data from *Hadoop Platform Management unit* and performs the proposed clustering approach. *Clustering modules*' main output is the rule results of the proposed methods discussed in this paper. The processed output of clustering unit is sent back again into Hadoop Platform Management unit for further retrieval. The output of the clustering module be used by the visualization component where the operational units are able to visualize notified alarms from registered rules and observe new alarm clusters.

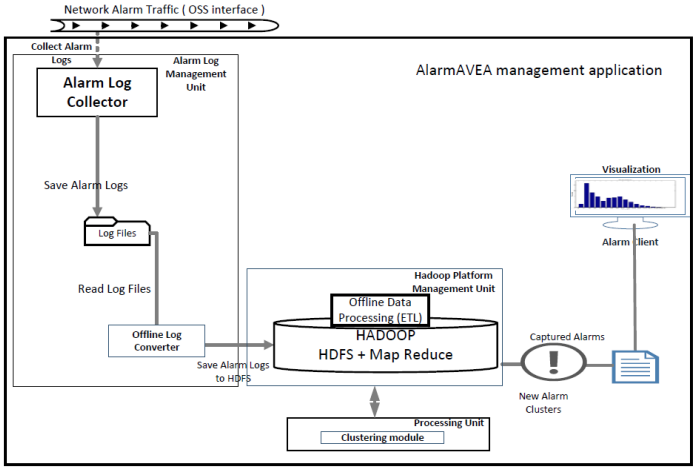
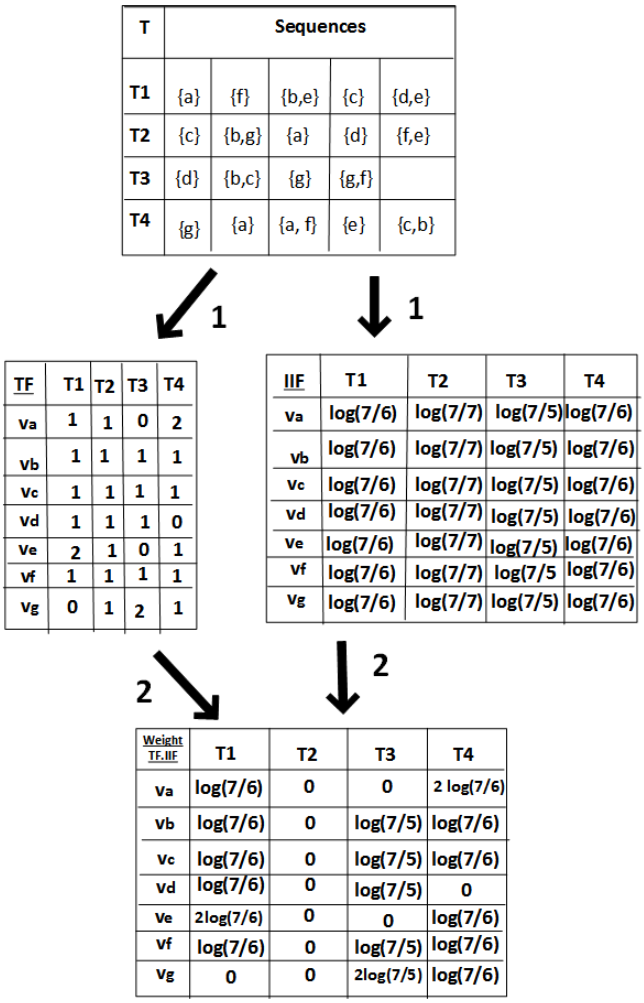


Fig. 1. General architecture diagram of AlarmAVEA management application.

B. Data model and definitions

In this paper, we consider the problem of clustering correlated sequences and comparing its performance with the sequential rule mining algorithms. We both formally define the problem and give a clarifying example in this section. Let a sequence database, *SeqDB* be the set of event sequences denoted by $SeqDB = \{T_1, T_2, ..., T_K\}$ where $T_i =< i, s_i >$ represents a *transaction*, i is the identifier of a transaction, each $s_i = \{X_1, X_2, ..., X_m\}$ is a subset of event occurrence itemsets from $I = \{i_1, i_2, ..., i_n\}$ where $X_1, X_2, ..., X_m \subseteq I$ and K is the total number of transactions [2] (Note that in the context of network alarm operations, each item corresponds to a specific alarm in a given $SeqDB$). Hence, the length of the sequence set $SeqDB$ is denoted by $|SeqDB| = K$. Every itemset X_k has a special attribute called $timestamp$, which records the time when the item occurred. Note in Figure 2, each row corresponds to a transaction whereas each column



T	Sequences				
T1	{a}	{f}	{b,e}	{c}	{d,e}
T2	{c}	{b,g}	{a}	{d}	{f,e}
T3	{d}	{b,c}	{g}	{g,f}	
T4	{g}	{a}	{a, f}	{e}	{c,b}

1

1

TF	T1	T2	T3	T4
Va	1	1	0	2
Vb	1	1	1	1
Vc	1	1	1	1
Vd	1	1	1	0
Ve	2	1	0	1
Vf	1	1	1	1
Vg	0	1	2	1

IIF	T1	T2	T3	T4
Va	log(7/6)	log(7/7)	log(7/5)	log(7/6)
Vb	log(7/6)	log(7/7)	log(7/5)	log(7/6)
Vc	log(7/6)	log(7/7)	log(7/5)	log(7/6)
Vd	log(7/6)	log(7/7)	log(7/5)	log(7/6)
Ve	log(7/6)	log(7/7)	log(7/5)	log(7/6)
Vf	log(7/6)	log(7/7)	log(7/5	log(7/6)
Vg	log(7/6)	log(7/7)	log(7/5)	log(7/6)

2

2

Weight TF.IIF	T1	T2	T3	T4
Va	log(7/6)	0	0	2 log(7/6)
Vb	log(7/6)	0	log(7/5)	log(7/6)
Vc	log(7/6)	0	log(7/5)	log(7/6)
Vd	log(7/6)	0	log(7/5)	0
Ve	2log(7/6)	0	0	log(7/6)
Vf	log(7/6)	0	log(7/5)	log(7/6)
Vg	0	0	2log(7/5)	log(7/6)

Fig. 2. An example showing the weight = $TF.IIF$ calculation for each item in an event sequence database when $|SeqDB| = K = 4$ (top) and the obtained item vector with *TF.IIF* weights (bottom).

represents a specific itemset that occurs at the indicated time instance. Note also that a specific itemset X_i can consist of multiple items each of which corresponds to a specific alarm. The item[1] i_j log format collected from the OSS interface has the format as listed in Table I. The item fields that are used in this paper are *ADDITIONAL-TEXT*, *EVENT-TIME* and *CLEARANCE-TIMESTAMP*. Note that we do not employ any type of feature selection to pre-process the data before the clustering process, but rely on these fields. The *ADDITIONAL-TEXT* field contains many information related to *ALARM-LOCATION*, *ALARM-NAME*, *NODE-NAME* (which represents the unique name of the equipment), etc. For our analysis, data related to *NODE-NAME* and *ALARM-NAME* are considered. EVENT-TIME field is the time at which the alarm has been initiated and *CLEARANCE-TIMESTAMP* reflects the field where the alarm has been clarified by the operator center or automatically by the machine itself. Note that in our database, we are assuming that all the alarms have been cleared.

The transaction $T_k, \forall k \in \{1, 2, ..., K\}$ in a given $SeqDB$ is formed as follows: Every item i_j in a given s_i has the fields as listed in Table I. The transaction T_k is formed depending on the item life-cycle of each item in a given sequence database. The lifecycle of an item starts at *EVENT-TIME* and ends at *CLEARANCE-TIMESTAMP*. Hence, the lifecycle duration of an item is defined as *ILC* = | *CLEARANCE-TIMESTAMP* - *EVENT-TIME*|. During the lifecycle duration ILC, all the items that have the *EVENT-TIME* between this duration are put into transaction T_k and sorted in time order. Note also that due to domain characteristics, some of the rules that apply to items in the alarm domain are as follows: Every item triggers a sequence of itemsets in a given transaction during items' lifestyle duration. Moreover, whenever an item i_j is triggered, the same item i_j is not observed again in a given transaction during its life-cycle duration.

[1]alarm and item will be used interchangeably throughout the paper.

III. TF.IIF WEIGHT METRIC AND PROPOSED METHOD

In this paper, we study clustering the items i_j based on their occurrences in a given transaction $T_k, \forall k \in \{1, 2, ..., K\}$ among the sequence database $SeqDB$. We use the item vectors $\mathbf{v}_{i_j} \in V$ and $j = \{1, 2, ..., N\}$ where N is the number of distinct items $i_j \subseteq I$ and V is the item vector set that is constructed for each distinct item vector in $SeqDB$. Using above notations, two different numerical statistic measures are used to reflect how important a given transaction T_i is in a given $SeqDB$: TF, *transaction frequency* and IIF, *inverse item frequency*. Define f_{i_j,T_k} to be the *frequency*, i.e. number of occurrence of item (i.e. *ALARM-NAME*) i_j in a given transaction T_k. Then, the transaction frequency of transaction T_k for the given item i_j (denoted here as $TF(i_j, T_k)$ in short), given as:

$$TF(i_j, T_k) = f_{i_j,T_k}. \tag{1}$$

$Inverse-item-frequency$ (denoted here as $IIF(i_j, T_i)$ in short), is defined as follows: Suppose the item i_j appears in n_i of the N distinct items of the $SeqDB$. Then

$$IIF(i_j, T_i) = log_2(\frac{N}{n_i}). \tag{2}$$

We define the item vector as $\mathbf{v}_{i_j}(TF.IIF) = [w_{TF.IIF}(i_j, T_1) \ w_{TF.IIF}(i_j, T_2) \ ... \ w_{TF.IIF}(i_j, T_K)]]$ or $\mathbf{v}_{i_j}(TF) = [w_{TF}(i_j, T_1) \ w_{TF}(i_j, T_2) \ ... \ w_{TF}(i_j, T_K)]$ depending on the selected $TF.IIF$ or TF weights respectively. Hence using this definition, the items with the highest $TF.IIF$ or TF scores are the items that best characterizes the item vector $\mathbf{v}_{i_j}$. For example, for TF weight, the item vector defines the number of occurrences of item i_j in all the transactions of $SeqDB$ and a transaction T_k has high $TF.IIF$ score for an item i_j if this transaction appears in relatively few other distinct items, but appears in this one, and when it appears for an item it tends to appear many times. Figure 2 shows the corresponding *TF* and *IIF* values for example transactions listed at step number 1. Note that *TF* indicates a measure of the relative frequency of the occurrence of a given transaction T_k in item i_j whereas *IIF* is an indicator of the closeness of each distinct transaction T_k with regards to their occurrences for a given item i_j.

An example of the transaction formation is illustrated in Fig. 2 at different time intervals. In this example, e.g. $T1$ is

Field Name	Description
ADDITIONAL-TEXT	Consists many items including Alarm NODE-NAME, ALARM-LOCATION and ALARM-NAME
NODE-NAME	Network equipment producing the alarm (e.g. base station, base station controller, transmission lines, etc)
ALARM-TYPE	Defines type of alarm (e.g. equipment alarm, communications alarm, environmental alarm, etc)
ALARM-NAME	Alarm identification
EVENT-TIME	Shows the alarms' initiation time
PERCEIVED-SEVERITY	Defines alarms' priority type (e.g. warning, major, minor)
TERMINATION-TIMESTAMP	Defines alarms' clearance time, i.e. termination time
CLEARANCE-TIMESTAMP	Defines alarms' clearance time

TABLE I. AN EXAMPLE OF ALARM LOG FIELD NAMES AND THEIR DESCRIPTIONS

constructed based on the sequentially arriving alarms during the *ILC* of a, i.e. when alarm a occurs all the remaining alarms f, b, e, c and d, e occurs sequentially until alarm a is cleared from the system. For example, when *ACDC FAULT* occurs in a given time, *DC POWER MAJOR, DC POWER MINOR* alarms follow it until *ACDC FAULT* alarm is cleared off. The intuition behind this way of transaction formation is to better correlate the alarms that are related to each other while forming a rule. Note also that at alarms d and e arrive at the same time. After first step, the middle part of Figure 2 illustrates all TF and IIF values calculated from this transaction database. Consider the item vector $\mathbf{v}_d$ for the item d where $v_d(TF) = [1\ 1\ 1\ 0]$, which means that for the given four transactions T_1, T_2, T_3 and T_4, the item d occurs $1, 1, 1$ and 0 times respectively at each of them. Its IIF value is calculated as $\mathbf{v}_d(IIF) = [log(7/6)\ log(7/7)\ log(7/5)\ log(7/6)]$ since $N = 7$ and item d appears $6, 7, 5$ and 6 times in the given vector space V. After these calculations, e.g. the weight vector of d is calculated as $w_{v_d}(TF.IID) = [log(7/6)\ 0\ log(7/5)\ 0]$ and $w_{v_d}(TF) = [1\ 1\ 1\ 0]$.

The intuition behind the above calculations of weights is to achieve meaningful set of clusters that represent items with different closeness definitions. In order to have items that fall into the same range, the input vector needs to be normalized. After selecting the weights according to the above calculations for each distinct items, the normalization step takes place as follows:

$$\mathbf{v}_{i_j}^{norm} = \frac{\mathbf{v}_{i_j}}{||\mathbf{v}_{i_j}||} \quad j = 1, 2, ..., N. \tag{3}$$

The above normalization step will help to ensure that the items with large number of presence in transactions will not dominate the items with smaller number of presence in transactions for the clustering algorithm. After the normalization step, the proposed method runs a standard clustering algorithm (e.g. K-means is chosen due to its simplicity without lose of generality) in order to obtain clusters with items that are highly related to each other depending on the selected TF or $TF.IIF$ weights. For the clustering process, the item definition is alarm and features define what kind of characteristics or properties such as time, equipment, transaction number, etc. represent the item. A summary of the whole proposed clustering method is detailed in Algorithm 1. After the alarm clusters are formed, in the evaluation results section, these clusters are leveraged for eliminating the insignificant alarm rules in order to increase the accuracy of the previously built sequential rules. Note that the proposed method utilizes clustering algorithms on *ALARM-NAMEs* for each different *NODE-NAME*.

Algorithm 1 Proposed clustering method

Inputs:
$SeqDB$: Event Sequence Database;
Outputs:
Items *(ALARM-NAMEs)* and their cluster indexes;
Method:
For each *NODE-NAME*:

1) Scan for all items in the database and construct the transactional database $\{T_1$,T_2,...,$T_K\}$ using item lifecycle definition based on Section II-B for all items.
2) Calculate the TF and $TF.IIF$ weights of each item in the sequential transactional database using Eq. (1) and Eq. (2) respectively.
3) Construct the item vector $\mathbf{v}_{i_j}$ based on the calculated based on the TF and $TF.IIF$ weights
4) Normalize the item vector and obtain $\mathbf{v}_{i_j}^{norm}$ using Eq.(3)
5) Run K-means clustering algorithm with the normalized item vector and number of clusters M as input

End

IV. EXPERIMENT AND EVALUATION RESULTS

In previous sections, the details about input data creation steps for the clustering algorithm are presented. In this section, we will demonstrate our clustering approach and its benefits by simulating it with real-experimental alarm data. As mentioned before, the main goal of clustering processing is to validate alarm rules which are generated using the sequential rule mining algorithm in [2][26]. For the experiments, we have utilized scikit-learn machine learning library in Python [27] as well as SPMF, an open-source data mining mining library of [28]. In this experiment, we worked on logs which are from $10/08/2012$ to $23/08/2012$ based on the Nokia's radio access network logs and logs on $24/08/2012$ are used for calculating prediction accurancy. All log files are preprocessed before clustering. Total number of lines is 369345 and totally 268 unique alarms (items) are extracted from these log files. Using a sequential rule mining algorithm, i.e. RuleGrowth of [2], $\kappa = 15207$ unique sequential alarm rules are generated from the log files having 268 unique alarms. The feature formation steps for clustering process are done using steps $2 - 4$ in Algorithm 1. After those steps, clustering process takes place at step 5. Two important aspects about clustering process are: i) Clustering process will be run per *NODE-NAME*. So, the transaction size, hence, features could be different for each NODE-NAME. ii) If a *NODE-NAME* has less than 10 distinct alarms, clustering is not run for that *NODE-NAME*. In this

Training Data	Prediction Count	Rule Occurrence Count	Accuracy
2012/08/10 - 23 (RuleGrowth)	2440281	443426	25.4 %
2012/08/10 - 23 (RuleGrowth, eliminated lower confidences)	1442345	306454	30.1 %
2012/08/10 - 23 (TF clustered)	1073962	228852	33.3 %
2012/08/10 - 23 (TF clustered and eliminated lower confidences)	583564	167989	40.9 %
2012/08/10 - 23 (TF.IIF clustered)	1124082	238318	33.6 %
2012/08/10 - 23 (TF.IIF clustered and eliminated lower confidences)	610427	173830	41.1 %

TABLE II. RESULTS OF PREDICTION-RULE MODEL FOR SEQUENTIAL RULES

experiment, we chose K-Means clustering algorithm due to its efficiency and scalability [29].

A. Results

In this section, the results of running the above proposed methods are discussed. After clustering process, two different results per different weight metrics are produced. Two metrics are also used to measure these results:

- Number of clusters matched with at least one rule, α
- Number of sequential alarm rules matched with at least one cluster, β.

Using the above notations, *Cluster Rule Matched Rate*, γ, and *Alarm Rule Matched Rates*, δ, are calculated as,

$$\gamma = \frac{\alpha}{\kappa}, \;\; \delta = \frac{\delta}{\kappa}.$$

TF Weight Results: Using this metric, 525 unique clusters are formed from all the *NODE-NAMEs* in $SeqDB$. Cluster results are evaluated by using previously found sequential alarm rules using RuleGrowth algorithm of [2]. Number of clusters matching with at least one rule is 458 and the number of alarm rules matching with at least one cluster is 11457. Here, matching refers to the case when the items in a given rule (both first half and second half) are present in a given cluster. So, *Cluster Rule Matched Rate* and *Alarm Rule Matched Rates* are 87.24% and 75.34% respectively (since the number of unique alarm rule $\kappa = 15207$).

TF.IIF Weight Results: By using this metric, 538 unique clusters are calculated. Cluster results are evaluated by using alarm rules as mentioned in section 4.1. The clusters matched with at least one rule count is 471 and alarm rules matched with at least one cluster count is 9586. So *Cluster Rule Matched Rate* and *Alarm Rule Matched Rate* are calculated as 87.55% and 63.04% respectively (since the number of unique alarm rule $\kappa = 15207$).

Note that *TF* and *TF.IIF* weight results will be used in evaluation section in order to reduce the number of sequential alarm rules and calculate the accuracy of these approaches.

B. Accuracy Evaluation

For the evaluation, we implement a prediction-occurrence model with the alarm rules. This model is based on the rule occurrences on the transactions. If the first half of a sequential rule of node occurs in a transaction, then the prediction *"the other half of this sequential rule will also occur in this transaction"* is made. If this half of sequential rule also occurs, the prediction is correct. Then, the correctness of the prediction is calculated as *accuracy*. The data set contains data from August 10 to August 24, 2012. The rules are created from the first days (the data from August 10 to August 23) and predictions are made for the following day (the data of August 24, 2012). During the experiments, the clustered alarm rules with *TF* and *TF.IIF* methods are also used. For making good predictions, we eliminate the rules which have lower confidence levels. Note that confidence levels are evaluated based on the results obtained from a traditional sequential rule mining algorithm, e.g. RuleGrowth. All results are shown as Table II. In summary, Table II shows which data is used for rule creation as training data, the *prediction count* which is the number of the first half of a sequential rule occurrences on the transactions, *rule occurrence count* which is the number of the second halves of a sequential rules following the first halves occurrences on the transactions. The accuracy is calculated based on the average of rule occurrence counts over prediction counts for transactions in order to measure how correct is the prediction.

Table II also shows that using rules from RuleGrowth algorithm in [26] [2], the accuracy is 25.4% and 30.1% when lower confidences are eliminated. Using the TF clustered approach, after eliminating the rules that do not match to any cluster, the accuracy level has increased to 33.3% and 40.9% when lower confidences are eliminated. Moreover, when $TF.IIF$ clustered approach is used to eliminate inefficient rules, the accuracy level becomes 33.6% and 41.1% when lower confidences are eliminated. These results demonstrate that through appropriate metric utilization, inefficient rules of sequential rule mining approaches can be eliminated in order to achieve up to 61% accuracy improvements.

V. CONCLUSION

In this paper, we proposed different numerical weight metrics and utilized clustering algorithms in order to increase the accuracy efficiency of sequential rules deducted using sequential rule mining algorithms. In summary, new measures are proposed to differentiate different clustering methods and the accuracy results of the reduced rule set has been compared with the accuracy results of the sequential rule mining algorithms. We evaluated the summarization performance of the clustering algorithms using the proposed metrics and obtained up to 61% accuracy improvements. As a future work, the evaluation of an overall clustering that includes all data for all nodes, which would reduce the number of clusters overall, and might uncover other generalizations can be studied.

ACKNOWLEDGMENT

This work has been sponsored by TUBITAK TEYDEB 1501 no. 3130411 project. We'd like to thank Bilkent University for providing validation results.

REFERENCES

[1] I. T. Frisch, M. Malek, and S. S. Panwar, *Network management and Control*, vol. 2. Springer Science & Business Media, 2013.

[2] P. Fournier-Viger, R. Nkambou, and V. S.-M. Tseng, "Rulegrowth: mining sequential rules common to several sequences by pattern-growth," in *26th ACM Symposium On Applied Computing, SAC'11*, pp. 956–961, 2011.

[3] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *SIGMOD Rec.*, vol. 29, pp. 1–12, May 2000.

[4] I. Rouvellou and G. W. Hart, "Automatic alarm correlation for fault identification," in *Proceedings IEEE INFOCOM '95, Fourteenth Annual Joint Conference of the IEEE Computer and Communications Societies: Bringing Information to People.*, 1995.

[5] S. Wallin, "Chasing a definition of alarm," *Journal of Network and Systems Management*, vol. 17, pp. 457–481, 2009.

[6] J. E. P. Duarte, M. A. Musicante, and H. H. Fernandes, "Anemona: a programming language for network monitoring applications," *International Journal of Network Management*, vol. 18, no. 4, pp. 293–300, 2008.

[7] H. Mannila, H. Toivonen, and A. Inkeri Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge Discovery*, vol. 1, pp. 259–289, Jan. 1997.

[8] M. Klemettinen, H. Mannila, and H. Toivonen, "Rule discovery in telecommunication alarm data," *Journal of Network and Systems Management*, vol. 7, pp. 395–423, 1999.

[9] H. Wietgrefe, K.-D. Tuchs, K. Jobmann, and et al., "Using neural networks for alarm correlation in cellular phone network," in *International Workshop on Applications of Neural Networks in Telecommunications*, 1997.

[10] T. Li and X. Li, "Novel alarm correlation analysis system based on association rules mining in telecommunication networks," *Inf. Sci.*, vol. 180, pp. 2960–2978, Aug. 2010.

[11] D. M. Meira, *A Model For Alarm Correlation in Telecommunication Networks*. PhD thesis, Federal University of Minas Gerais., 2000.

[12] S. Laxman and P. S. Sastry, "A survey of temporal data mining," in *SADHANA, Academy Proceedings in Engineering Sciences*, 2006.

[13] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the Eleventh International Conference on Data Engineering*, ICDE '95, (Washington, DC, USA), pp. 3–14, IEEE Computer Society, 1995.

[14] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993* (P. Buneman and S. Jajodia, eds.), pp. 207–216, ACM Press, 1993.

[15] G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth, "Rule discovery from time series," in *KDD'98*, pp. 16–22, 1998.

[16] D. Lo, S.-C. Khoo, and L. Wong, "Non-redundant sequential rules-theory and algorithm," *Inf. Syst.*, vol. 34, pp. 438–453, June 2009.

[17] M. Yu, W. Li, and L.-j. W. Chung, "A practical scheme for mpls fault monitoring and alarm correlation in backbone networks," *Comput. Netw.*, vol. 50, pp. 3024–3042, Nov. 2006.

[18] Y. Liu, J. Zhang, X. Meng, and J. Strassner, "Sequential proximity-based clustering for telecommunication network alarm correlation," in *Advances in Neural Networks-ISNN 2008*, pp. 30–39, Springer, 2008.

[19] K.-D. Tuchs and K. Jobmann, "Intelligent search for correlated alarm events in databases," in *Integrated Network Management Proceedings, 2001 IEEE/IFIP International Symposium on*, pp. 285–288, IEEE, 2001.

[20] J.-H. Bellec, M. Kechadi, *et al.*, "Feck: A new efficient clustering algorithm for the events correlation problem in telecommunication networks," in *Future Generation Communication and Networking (FGCN 2007)*, vol. 1, pp. 469–475, IEEE, 2007.

[21] H.-C. Chao, C.-M. Hsiao, W.-S. Su, C.-C. Hsu, and C.-Y. Wu, "Modified adaptive resonance theory for alarm correlation based on distance hierarchy in mobile networks," in *Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific*, pp. 1–4, Sept 2011.

[22] J. Bellec, M. T. Kechadi, and J. Carthy, "A new efficient clustering algorithm for network alarm analysis," in *International Conference on Parallel and Distributed Computing Systems, PDCS 2005, November 14-16, 2005, Phoenix, AZ, USA*, pp. 638–643, 2005.

[23] A. Makanju, E. E. Milios, *et al.*, "Interactive learning of alert signatures in high performance cluster system logs," in *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pp. 52–60, IEEE, 2012.

[24] A. Clemm and M. Hartwig, "Netradamus: A forecasting system for system event messages," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pp. 623–630, IEEE, 2010.

[25] C. D. Manning, P. Raghavan, H. Schütze, *et al.*, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[26] O. F. Celebi, E. Zeydan, I. Ari, O. Ileri, and S. Ergut, "Alarm sequence rule mining extended with a time confidence parameter," in *Poster Proceedings Volume of Industrial Conference on Data Mining 2014 (ICDM'14)*, pp. 26–35, 2014.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, *et al.*, "Scikit-learn: Machine learning in python," *The Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[28] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "Spmf: a java open-source pattern mining library," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3389–3393, 2014.

[29] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied statistics*, pp. 100–108, 1979.