Mining of an Alarm Log to Improve the Discovery of Frequent Patterns

Françoise Fessant, Fabrice Clérot, and Christophe Dousson

France Télécom R&D, 2 avenue P. Marzin, 22307 Lannion, France
{francoise.fessant, fabrice.clerot, christophe.dousson}
@francetelecom.com
http://www.rd.francetelecom.com

Abstract. In this paper we propose a method to pre-process a telecommunication alarm log with the aim of discovering more accurately frequent patterns. In a first step, the alarm types which present the same temporal behavior are clustered with a self organizing map. Then, the log areas which are rich in alarms of the clusters are searched. The sublogs are built based on the selected areas. We will show the efficiency of our preprocessing method through experiments on an actual alarm log from an ATM network.

1 Introduction

Telecommunications networks are growing in size and complexity, which means that a constantly increasing volume of notifications must be handled by the management system. Most of this information is produced spontaneously by equipment (e.g. status change and dysfunction detection) and this message flow must be preprocessed to make an effective management possible.

Filters based on a per-notification basis fail to perform an adequate information pre-processing required by human operators or by management application softwares which are not able to process such amount of events. A pre-processing stage must "thin" this information stream by suppressing redundant notifications and/or by aggregating relevant ones. Numerical time constraints must also be taken into account since time information is *apropos* for the telecommunications alarm propagation. Many works deal with different approaches and propose more or less complex intelligent filtering: one can use some efficient rule-based languages [1], and/or object-based techniques [2]. More specific techniques are devoted to capture time constraints between alarms [3], [4]. In any case, the problem of expertise acquisition remains the same: how to feed the filtering system? Which aggregation rules are relevant?

A way to filter the information flow is to exploit the logs collected from telecommunications equipment. We have developed a tool called FACE (Frequency Analyzer for Chronicle Extraction) that performs a frequency-based analysis in order to extract "frequent patterns" from the logs. We only suppose that all the events are time-stamped. The searched patterns are sets of event patterns with time constraints between them (these sets are called "chronicle models") and the frequency criterion is defined as an user-defined minimal frequency threshold [5].

P. Perner (Ed.): ICDM 2004, LNAI 3275, pp. 144–152, 2004.

Identifying the most frequent chronicle models is relevant to reduce the number of alarms displayed to the operator: if a chronicle corresponds to a dysfunction, the corresponding set of alarms is aggregated before being displayed to the human operator; if not, the corresponding set of alarms is filtered. At the moment, we do not use any extra knowledge about the domain; the rule qualification (aggregation or filtering) is performed by an expert at the end of the discovering process.

Real experiments on telecommunications data show that most of the discovered chronicles are relevant and some of them have a real benefit for the experts. However, the chronicle process implemented in FACE, based on the exhaustive exploration of the chronicle instances in the alarm log, is very memory-space consuming and the main factor of this explosion is the size of the processed event logs.

To deal with that problem in the current implementation of FACE, the operational experts manually select some alarms types and/or some time periods in the alarms logs in order to extract a more manageable sublog to be processed by the tool. The purpose of this communication is to describe and to evaluate a pre-processing method to automatically extract relevant sublogs from an initial alarm log to simplify the use of the software and alleviate the memory saturation effect.

The pre-processing stage we propose can be decomposed in two main steps. Firstly we group together the alarm types which exhibit the same temporal behavior. Then we search for each group the areas of the log that are *rich* in alarms of the observed group. The sublogs are deduced from the alarms in these areas. The following sections describe our data pre-processing and the experimental results obtained on an actual alarm log from an ATM network.

2 Data Description and Representation

Previous experiments with FACE on alarm logs of various origins have revealed that the instances of chronicles tend to be distributed in a clustered way along time. This can be explained by the fact that most chronicles are indicative of faulty network states, which do not happen randomly in time but only happen for short periods of time before being corrected.

The data representation introduced below takes this into account, gives a great importance to the temporal behavior of the alarms and aims at grouping in sublogs alarms with similar temporal behaviors.

In the first data preparation phase the entire log is split in slices of fixed duration. Each slice, called *period,* can contain a variable number of alarms or can also be empty. The periods can be seen as unit elements of the log; they form a partition of it. The periods constitute the base on which the alarms will be described and from which the sublogs will be built.

The alarm types are then represented through their temporal evolution along the periods. The cumulative profile of an alarm type is computed by adding up its occurrences in successive periods and it is normalized in order to make it independent of the total number of occurrences. Such a description takes into account the time aspect present in alarm generation.

Our experiments are achieved with ATM (Asynchronous Transfer Mode) network data. We have about one-month alarms log with 46600 alarms dispatched through 12160 different types. An alarm type is characterized by an alarm message (a string)

and a date and time of occurrence; 75% of the alarm types weight less than 3 occurrences. We fixed the duration of a period to 15 minutes and we obtained 2317 periods which contain about 16 alarms on the average. The figure 1 shows the occurrences of one alarm type of the log on all the periods and the figure 2 shows the corresponding normalized cumulative profile.

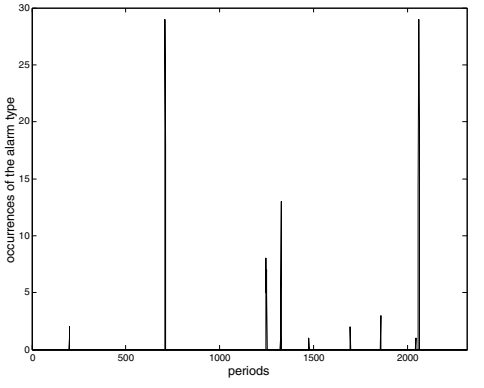

Fig. 1. Occurrence of an alarm type on all the log periods

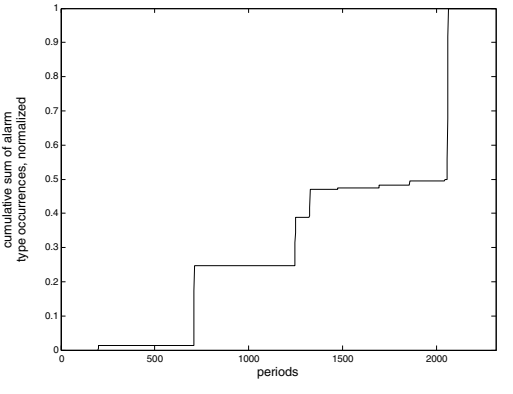

Fig. 2. Normalized cumulative profile of the alarm type of Figure 1 on all the log periods

The purpose of the next step of the method is to group together the alarm types which have the same temporal behavior.

3 Clustering of the Cumulative Profiles with a Self-Organizing Map

The alarms types are grouped together with a Self Organizing Map (SOM). A SOM is a useful tool for exploratory data analysis made up of a set of nodes organized into a

2-dimensional grid (the *map*). Each node has fixed coordinates in the map and adaptive coordinates (the *weights*) in the input space. The input space is relative to the variables setting up the observations. The self-organizing process slightly moves the nodes coordinates in the data definition space -i.e. adjusts weights according to the data distribution. This weight adjustment is performed while taking into account the neighboring relation between nodes in the map. The SOM has the well-known ability that the projection on the map preserves the proximities: close observations in the original multidimensional input space are associated with close nodes in the map. For a complete description of the SOM properties and some applications, see [6] and [7].

After learning has been completed, the map is segmented into clusters, each cluster being formed of nodes with similar behavior, with a hierarchical agglomerative clustering algorithm. This segmentation simplifies the quantitative analysis of the map [8].

We consider only the frequent alarm types -i.e. alarm types making at least 3 occurrences through the log: a chronicle being by definition characterized by instances with several occurrences in the log, infrequent alarm types cannot take part in the chronicle creation. We end up with a database of 3042 cumulative profiles (the observations) described on the space of the 2317 periods. We experiment with a 9x9 square map with hexagonal neighborhoods (experiments on SOM have been done with the SOM Toolbox package for Matlab [9]).

The clustering of the map has revealed 10 different groups of observations with similar behavior. Figure 3 details the characteristic behavior of the clusters. We plot the mean cumulative profile for each of the 10 clusters; each of the clusters being identified by a number. The mean cumulative profile of a cluster is computed by the mean of all the observations that have been classified by the nodes of the map belonging to the cluster.

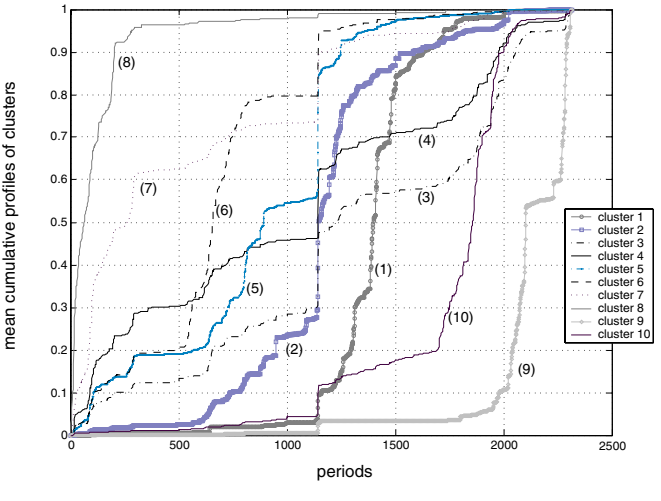

Fig. 3. Mean cumulative profiles of the clusters

The sharper the rise of a cumulative profile, the stronger the accumulation of the alarms. The visual inspection of the figure shows that the clusters indeed correspond to very different accumulation behaviors: some clusters correspond to accumulations on rather limited sets of periods (cluster 8, accumulating at the very beginning of the log, cluster 9 accumulating at the very end or cluster 1 accumulating in the middle, for instance); other clusters show a more even distribution in time (clusters 3 and 4, for instance).

At this stage we end up with a limited number of groups that characterize and summarize the whole alarm types: all the alarms types classified in a cluster accumulate in the same way through the log.

4 Sublogs Construction

Some periods of the log are more significant than others for a given cluster of alarm types. We define the importance of a period for a cluster by the number of alarms of the cluster included in the period, in proportion of all the alarms this period contains (with this definition, we tend to favor some periods producing few alarms but essentially alarms of the cluster).

We proceed as follows to select the most interesting periods for a cluster: we first order the periods according to their importance for the studied cluster; the periods which have the highest importance values are ranked first. The subset of periods corresponding to the cluster is simply built by adding the periods in order of decreasing importance until a given proportion of the alarms of the cluster are contained in the sublog.

The figure 4 illustrates the result of the period selection process for cluster 1. We arbitrarily fixed the stopping criterion to 90% (i.e. we retain the periods until the sublog

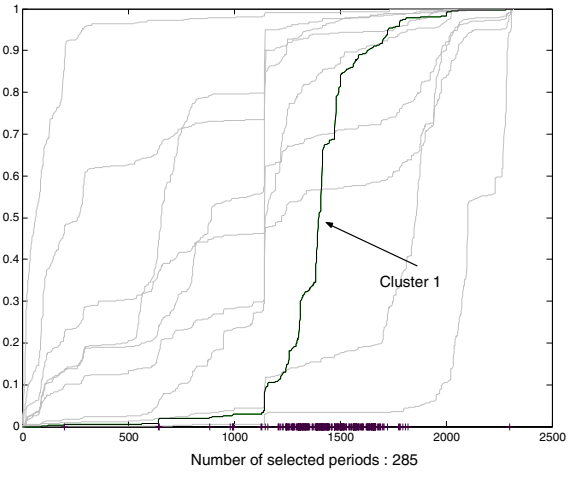

Fig. 4. Selected periods for an alarm type cluster

contains 90% of the alarms of the cluster). For the cluster given here as an example in the figure, we select 285 periods of the log and this corresponds to 7% of the whole alarms. A cross on the x-axis marks the selected periods. These selected periods clearly coincide with the accumulation areas of the cluster (essentially in the middle of the log for the cluster).

The figure 5 plots the location of the selected periods in the log for each cluster. The comparison with figure 3 shows that the selected periods for a given cluster correspond to the areas of alarm accumulation for this cluster.

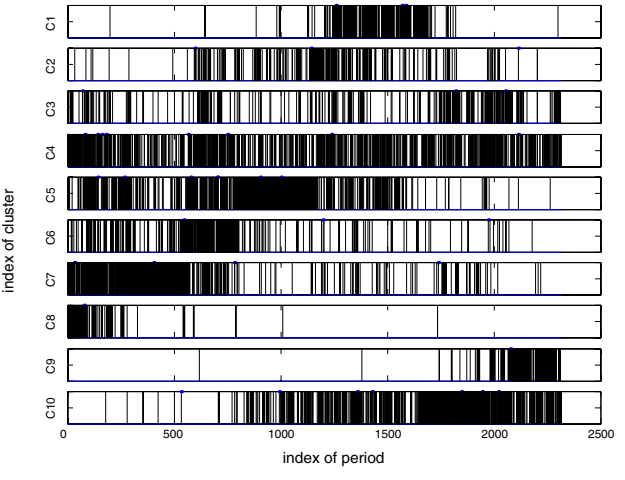

Fig. 5. Time location of the selected periods for the cluster

The number of periods which have to be taken into account to reach 90% of the alarms of a cluster vary a lot from a cluster to another, but whatever the observed cluster, the amount of selected periods never goes beyond 35% of the whole alarms.

Table 1. Number of periods and alarms in each sublog

Sublog index	Number of periods	Number of alarms
1	285	3079
2	287	9996
3	334	14204
4	783	16283
5	741	16083
6	361	11879
7	608	14725
8	136	4476
9	163	7043
10	658	10318
Total	4356	108086

At this step of the process, we have isolated as many subsets of periods as alarm type clusters. The sublog attached to a cluster of alarms is simply obtained by re-ordering in time the alarms of the corresponding subset. Note that we keep all the alarms belonging to the periods, including those of infrequent alarm types.

The Table 1 gives the number of periods in each subset and the corresponding number of alarms (recall that the entire log consists of 2317 periods and 46600 alarms).

Let us remark that our selection method considers each cluster independently of the others and, therefore, one period can be assigned to several subsets. The effect is that the sum of alarms over the all sublogs is about 2.5 the number of alarms of the initial log.

5 Experimental Results

In this section we report the experimental results we obtained when running FACE algorithm on the initial log and on the sublogs designed as previously explained (from now on we call them SOM sublogs). FACE learning algorithm requires two input parameters: the time window t_w which sets the maximum distance between the events of a chronicle model and the minimum threshold of the chronicle instances n_{qmin} which gives the minimum number of instances a chronicle model must have in the log to be considered as frequent. Time window t_w is fixed to 15 seconds and we choose to vary the value of n_{qmin}.

We report in figure 6 the number of different chronicle models discovered on the entire log and on t he SOM sublogs as a function of n_{qmin}. We also plot this number for normal sublogs. To build normal sublogs we simply manually split the whole log into several successive slices with about the same number of alarms in each slice. We

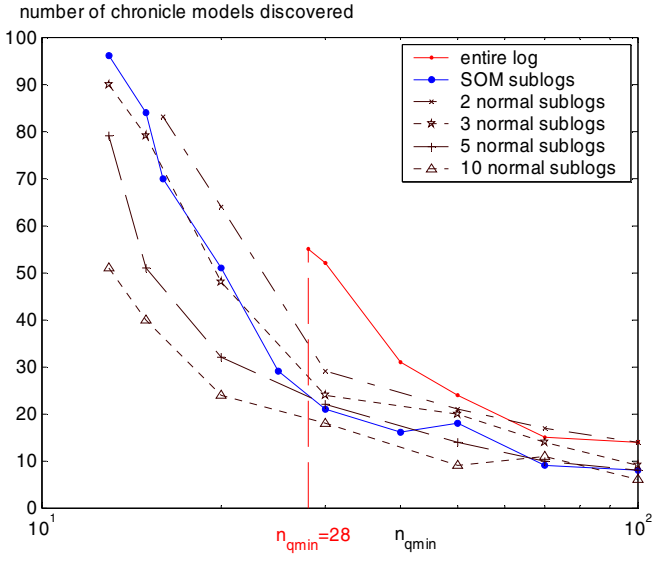

Fig. 6. Number of different chronicle models discovered

experiment with four different sets of sublogs (the log is successively split in 2, 3, 5 and 10 sublogs, we proceed as operational experts usually do to extract more manageable sublogs from an alarm log).

All experiments reported below have been run on the same computer (a Pentium 4 with 1.7 GHz of CPU and 1 Go of RAM) with no other application running than FACE.

We can observe that the entire log is impossible to process for n_{qmin} below 28; the reason is a lack of memory-space of the computer. The treatment remains possible on sublogs for much smaller values of n_{qmin} (the limit is reached for n_{qmin}=16 for the set of 2 sublogs and n_{qmin} =13 for the other sets of sublogs, and it is never possible to go below this value).

We are able to discover more chronicles with the sublogs than with the entire log (except for the set of 10 normal sublogs): for a given n_{qmin}, we discover more chronicles with the entire log than with the sublogs, but the exploration can be carried on sublogs for much lower n_{qmin}. Moreover, a detailed analysis of the discovered chronicles shows that we find from the sublogs all the chronicles we have found from the entire log and also many new chronicles: we do not loose any chronicle with SOM sublogs.

We now compare the number of discovered chronicle models for low values of n_{qmin}, with normal sublogs and with SOM sublogs. We can see that the highest number of chronicles discovered with normal sublogs is obtained for the set of 3 sublogs. SOM sublogs show a better performance. We found more chronicles with SOM sublogs. Therefore, the improvement obtained with the SOM sublogs cannot be attributed to the sequential processing of many smaller logs instead of the entire log and to the ability to reach low frequencies. This improvement has to be attributed to the data processing method adopted to build relevant sublogs.

Let us remark that the manual search of the good number of sublogs that will lead to the discovery of the maximum number of chronicles is a very time consuming process. The processing method we propose automatically extracts the relevant sublogs.

These experimental results validate our data processing method which takes into account the fact that chronicle instances do not happen randomly in time but in a rather clustered way; using this pre-processing we recover all the chronicle models discovered without pre-processing and we can analyze the log in a more accurate way and discover new chronicles. We also have observed that the total processing time of the SOM sublogs remains of the same order of magnitude as for the entire log.

6 Conclusion

The system FACE is very helpful for experts to discover monitoring knowledge from alarm logs. However, the chronicle discovery process implemented in FACE (based on the exploration of all the possibilities of chronicle instances) has the limitation to be very memory-space consuming; the main factor of this explosion being the size of the alarm log.

In this paper we have proposed a method to automatically extract relevant sublogs from an alarm log to simplify the use of the software and alleviate this memory saturation effect.

The method is decomposed in several steps. The first step consists in the description of the alarm types in a suitable representation that takes into account the temporal evolution of the alarm types through the log. Then the alarm types which have the same temporal behavior are grouped together with a self-organizing map. The result of the clustering is a description of the whole alarm types in few groups. Finally, the areas of the log that are the more relevant for the clusters of alarm types are isolated and the alarms in these areas are grouped in sublogs. The sublogs can be processed with FACE.

We presented experiments on an actual ATM log. The proposed data processing method turns out to be very effective: with the sublogs we have found the same chronicles that we were able to find while processing the whole log. Moreover we can carry the search on sublogs to a point that is impossible to reach with the entire log and obtain many new chronicles. We also have observed that the total duration process of the sublogs always remain moderate. Another interesting point is that the user can get intermediate results after the processing of each sublog and future research will consider the adaptation of the learning parameters of the tool independently for each sublog in order to discover the maximum number of chronicles.

References

1. Möller, M., Tretter, S., Fink, B.: Intelligent filtering in network-management systems. Proceedings of the 4th International Symposium on Integrated Network Management (1995) 304-315
2. Nygate, Y.A.: Event correlation using rule and object base techniques. Proceedings of the 4th International Symposium on Integrated Network Management (1995) 279-289
3. Dousson, C.: Extending and Unifying Chronicle Representation with Event Counters. Proceedings of the 15th ECAI (2002) 257-261
4. Jakobson, G., Weissman, M.: Real-time telecommunication network management: extending event correlation with temporal constraints. Proceedings of the 4th International Symposium on Integrated Network Management (1995) 290-301
5. Dousson, C., Vu Duong, T.: Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. Proceedings of the 16th IJCAI (1999) 620-626
6. T. Kohonen. Self-Organizing Maps. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (2001)
7. E. Oja. S. Kaski. Kohonen maps. Elsevier (1999)
8. Vesanto, J., Alhoniemi, E.: Clustering of the Self-Organizing Map. IEEE Transactions on Neural Networks. 11 (3) (2000) 586-600
9. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: SOM Toolbox for Matlab, Report A57 Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, (2000)