Temporal Classification for Fault-prediction in a real-world Telecommunications Network

Mohammad Jaudet, Naeem Iqbal, Amir Hussain[1], Kamran Sharif[2]
Dept. of Electrical Engineering, PIEAS, Islamabad, Pakistan
[1]Dept. of Computing Science, University of Stirling, Stirling FK9 4LA, Scotland, UK
[2]DE NMS/DXX Optical Fiber System, PTCL Complex, Rawal Pindi, Pakistan
jaudet@pieas.edu.pk, naeem@pieas.edu.pk, ahu@cs.stir.ac.uk, kamran.sharif@ptcl.net.pk

Abstract

This paper presents a new temporal classification approach for fault-prediction in a Telecommunications Network. The countrywide data network of Pakistan Telecom (PTCL) has been selected as a basis for the investigation of classification algorithms to predict faults before they stop a large number of users' circuits from normal operation. The main problems addressed are the evaluation of alarms and development of new machine learning tools to help overcome the interoperability issues. The motivation behind this work is to assist human operators and minimize the cost of the alarm evaluation process.

Keywords: *Network Management, Classification, Prediction and Decision Tree Induction.*

1. Introduction

PTCL (Pakistan Telecom) network consists of various communication devices operating at different levels. The main transmission medium is the optical fiber and data is carried by a combination of DWDM and SDH equipment. This backbone allows the establishment of different services such as ATM (Asynchronous Transfer Mode), PSTN (Public Switched Telephone Network), and PDN (Public Data Network) spread over the whole country. Purchase of equipment from different vendors has enhanced the difficulty of resource management in each service. In this study, we investigate the PTCL's Public Data Network and aim to develop a software suite for fault-prediction in such telecommunications networks.

Effective fault prediction in a telecommunications network requires a thorough investigation of alarms. Alarms are generated by communication devices and collected by monitoring processes. Devices usually generate alarms when conditions of their operation change due to some malfunctioning or configuration update. An alarm signal contains information about the problem encountered and time of occurrence of the problem. Modern devices are smart enough to detect problems related to data streams passing through their communication interfaces. But these features alone do not guarantee smooth operation of a network. Monitoring processes known as network management systems have to do a lot of work based on data collected from these devices.

Network management becomes more difficult when dealing with devices from different manufacturers. This situation requires additional efforts from the operators in order to detect communication problems, as end-to-end circuits do not appear under a single console. One of the objectives of this research is therefore to develop a single console based monitoring process to help detect problems in the network. The heart of this monitoring process is based on data mining algorithms which will be able to detect faults appearing in the domains of different network management systems. This partially developed suite of applications is also capable of predicting network faults before they stop a large number of users' circuits from normal operation.

In this paper, we have employed and adapted TimeSleuth [1] (a C4.5 [2] variant) software for analysis of alarm messages and fault-prediction in a selected telecommunications network, namely PTCL. State-of-the-art temporal decision trees and temporal rules have been developed and inducted for prediction of alarms related to different types of network devices. Add-on modules have been developed to enable dynamic reconstruction of temporal decision trees for online operation. A new C module has also been developed and integrated with the software to help evaluate alarms and predict upcoming events directly acquired from backend databases of network management systems. Certain other modules (C4.5/TimeSleuth [1] suite) used to evaluate the decision tree generated by TimeSleuth, and the TimeSleuth's front-

end has also been modified in order to process large decision trees and generation of causal rules.

The rest of this paper is organized as follows: an overview on related work is given in section 2. Our proposed approach for handling alarms is described in section 3, and the results of our approach are given in Section 4. Some concluding remarks and future work recommendations are finally outlined in Section 5.

2. Related Work

Alarm sequences are a kind of irregularly observed data with symbolic values rather than numeric ones. Irregularly observed numeric data is treated differently [3, 4] from regularly observed data [5, 6]. An irregularly sample dataset is usually transformed to frequency domain via a variant of Discrete Fourier Transform [7]. A reverse transform returns the dataset with equally spaced samples. This however does not guarantee the availability of original samples in the new set of samples. Recently, development in research related to modeling of chaotic processes, results of prediction have significantly improved for irregularly sampled datasets [8]. When we interact with irregularly sampled alarms, we can not approximate alarm symbols. We, however, could approximate time differences between consecutive alarms.

When attempting to approximate time between consecutive alarms, we could try classification techniques via feed forward neural networks [9, 10]. Classification techniques [11, 12, 13] are available in most of research areas such as neural networks, data mining and machine learning. In data mining and machine learning research, alarm sequences are called temporal event sequences. Temporal event sequences consist of symbolic values and time stamps. In most cases the dimension of time is used for keeping things in order. Temporal event sequences are also called symbolic, categorical and ordinary sequences in literature. Temporal event sequences have been treated mostly in business under market basket research and their application in robotics is growing. Temporal event sequences are also modeled using Markov Models, Hidden Markov Models [14] and Recurrent Neural Networks [15].

There are two main methodologies treating such sequences in data mining and machine learning research. One is the template-based approach [16] and the other one is classification based [11]. Template based methods are mainly based on variants of Apriori [17, 14] algorithm. Here, association rules are discovered using frequent patterns in event sequences. Time widows of different lengths containing events are considered as baskets of items. Sometimes infrequent patterns are also used for discovery of rules. Classification techniques group events based on similarity automatically. Classification allows predictions and is done by different algorithms in artificial intelligence research. Classical classification algorithms do not consider temporal nature of event sequences and treat them as a set of unordered items. This is the same in Apriori based methodologies.

Decision trees provide means for classification and prediction of tasks. They have been widely used for classification and extracting classification rules. A variety of different algorithms for pruning trees and classification rules have been developed over the time. C4.5 [2] and Ripper [18] are classical examples of such decision trees' based algorithms. C4.5 is a widely used decision tree and rule generator. Classical C4.5 algorithm does not distinguish the passage of time between instances and needs to be modified. With suitable modifications [19] C4.5 can preserve temporal order and discover causal rules between sequences of events. Causal relations between input and output help us to understand and predict the behavior of a system. Finding such relations is called causal [20] rules discovery and now widely accepted as temporal rules discovery.

Modifications to C4.5 lead to an unsupervised tool TimeSleuth [1] to find temporal or causal rules [21]. This tool uses a modified C4.5 algorithm to accept flattened input streams. The flattening process is similar to delaying inputs. This helps in finding causal relations between more than one pair of events. Increasing the temporal windows size increases execution time as well as demand for memory. This however increases quality of association rules that are critical for good prediction. The main advantage of TimeSleuth is that it can work as unsupervised tools for classification and temporal rules generation. TimeSleuth has been proven to outperform its competing algorithms TETRAD [22] and CaMML [23, 24] in finding causal rules. Before attempting to use a classification based methodology, we also attempted a neural network [25] based approach for prediction of alarms. Our neural network based approach predicted time spacing between consecutive alarms but was not suitable for prediction of an alarm symbol.

Timeweaver [26] is another algorithm developed for prediction of rare equipment failures based on alarm processing. It was based on a genetic algorithm and was considered better than non-temporal version of C4.5. We have requested the data sets and implementation of Timeweaver used in previous comparison [26]. This would help us in comparing results of Timeweaver with TimeSleuth in the future for efficiency and quality of rules. Alarm processing has also been attempted twice using Apriori algorithm. One was made in the University of Helsinki [27] and the other was carried out at IBM research [28]. Besides these attempts, a neural network based methodology was developed at University of Hannover [29] for alarm minimization.

3. Alarm Sequence Processing

Prediction targets in our study are telecommunications network alarms such as a communications link is broken between two communication devices. A prediction here is an estimation of a categorical or nominal value in the near future (e.g., communication link will be down within five minutes). Network alarms can be collected directly from a communication device locally and remotely. Such alarms could also be collected from a network management system associated with that communication device. Network management systems keep topological information and receive thousands of alarms in short time periods. It is common to find more than one such network management system in average sized telecommunications networks.

Network management systems store alarms and topological information on their respective database servers. The structure of database to store this kind of information was different in all network management systems we encountered in our study. We took care of this problem with the help of a UDB (unified database) [25] that stores alarms and topological information in a homogenous format. The UDB is kept in synch with databases of network management systems through a synchronizing process. Data analysis techniques to analyze data collected from more than one network management systems [25] may reveal useful patterns characterizing network problems.

Next thing in alarm processing is the identification of alarms that need to be examined. We discussed this with domain experts at PTCL and decided to follow three of main alarms. This is particularly useful in identifying what could be a possible reason leading to a certain condition. This allows discovery of alarm sequences that were never considered critical by domain experts previously. We made sure that these selected alarms belong to each type of device installed on the network. The first selected alarm was chosen to represent the "CRC errors from far-end" condition. The second alarm represents "Missing interface signal" condition. In third alarm, we have chosen to examine "NTU line fault" condition. We refer to these events, namely all occurrences of link failure alarm, as the set of target events [30, 31].

Once target alarms are identified the next thing in the pipeline is finding alarms or set of alarms that precede a target alarm. One approach is to use fixed time windows ending at target alarms and mine association rules for identifying alarms occurring before that target alarm. This type of approach [27, 28] is similar to Apriori algorithm if we replace the market basket with fixed time windows. In another approach [26] a genetic algorithm was trained to learn alarm sets preceding particular alarms. On the other hand, we use transformed alarms to predict target alarms and extract rules. Discovered rules in our case are called temporal or causal and not association rules. We then delay alarms to improve the prediction results and find rules associated with this delaying mechanism. We decide number of delayed inputs based on evaluation of discovered rules. Based on these final rules, we could find frequent alarms happening prior to a critical alarm.

Prediction of alarms requires availability of training samples and test samples for each type of device in the network. For this purpose, we induce a decision tree based on training samples and then analyze results of prediction based on test samples. Once a good enough decision tree is ready for deployment in a production environment, we start a new thread of device monitoring daemon. Such a thread runs on the monitoring machine, then starts gathering current alarms from UDB and finally begins analysis of incoming alarm data online. It has a built in alarm evaluation mechanism and sends audible alarms and popup windows about anticipated critical alarms. The daemon accepts path to decision tree data, device type and number of delays. Figure 1 shows an illustration of processes running on different machines. All these processes can also run on a single computer.

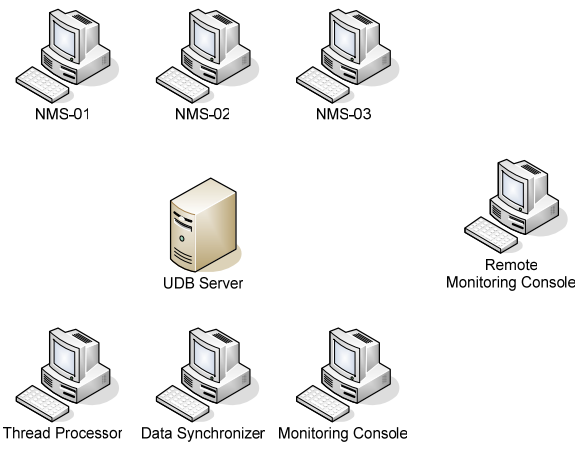

Figure 1: Monitoring Processes

A typical alarm data contains many different fields but most important fields are those having information about time of occurrence, type of device, device identification, alarm type, severity and textual description. These fields are very important for building up training and test samples. Type of device allows separation of device type specific samples. Time of occurrence and device identification fields allow temporal alignment of alarms with respect to a particular device. We divided our past data into two different time windows. One time window is used for generating training samples and other window for setting up test samples. We now transform alarms for each device into samples where each sample contains

current alarm type, time between the current alarm and next alarm.

Once transformation of alarms takes place for each device, we combine the data for the different types of devices. We do this by appending transformed data for each device one after another for each type of device. This way we prepare training and test samples for each type of device. We then filter out training and test samples where the incoming alarm is not critical. This allows prediction of critical alarms based on temporal spacing between two consecutive alarms. A temporal tool like TimeSleuth can now generate temporal decision tree and decision rules.

Delaying a single alarm does not allow finding frequent alarms happening before a critical alarm. However, if we delay alarms from 9-10 alarms for our database, we get improved prediction in alarm detection. For this purpose, we transform and format alarms for each device in a way that each sample contains alarm at time t1, time difference, alarm at time t2, time difference, alarm at time t3, and so on. Generation of final training and test samples is the same as before in case of delaying one alarm. We now filter out training and test samples where the last alarm in each sample is non-critical. Once a production tree is ready, we could build sets of alarms frequently happening before a critical alarm based on inducted rules. Please note that ordering of alarms here is preserved while in case of [27, 28] it is not. Same thing is true with [26] as well.

4. Empirical Findings

We report results obtained from the country wide data communication network managed by the largest telecom operator in Pakistan. We collected alarms stored in databases with network management systems. The collected data had around 60,000 alarms which had occurred over the previous four days. There were 30 devices of three different types speaking mostly. Alarms from the first three days were used as source for training samples, and alarms from fourth day served as source for generation of samples for test purposes. We investigated effects of delaying alarms or increasing time windows in prediction of alarms as symbol. The temporal rules associated with these trials were also extracted and tested

TimeSleuth needs all training samples in memory for generation of decision trees. Its rule generator also needs memory to hold decision tree for rule generation. In order to leverage its memory overhead, we first generated preliminary rules with all possible sequences. This lead to establishment of global classes of alarm sequences. Accuracy of rules for this classification tree and its rules stayed around 80-85%. We also delayed alarms to form a decision tree and its corresponding rules to detect frequent alarm groups. Accuracy for prediction of groups stayed around ninety percent. We started with a 60,000 alarm messages collected during four days. After consulting the fault management manual we filtered out frequent warning messages automatically handled by a device or its associated network management system.

Further removal of less critical alarms as a target class reduced alarm messages considerably. This removal took place during the preprocessing stage and not at the raw alarm messages level. We concentrated only on three of the main critical alarms happening before a link failure. Each alarm record contains critical alarm and the alarm before it. The record also contains time difference between these two alarms. For analysis, we divided these sequences into two, namely, the training sequences and the test sequences. TimeSleuth accepts training and test samples in separate files. So for each type of device, we prepared one data file containing training samples and the other file containing test samples.

We regenerated decision trees and causal rules for each device type using this reduced data sets. The time window, initially set to one produced good results for alarm prediction and maximum error was found to be below three percent. We also generated trees and rule sets for time window size between 1 and 10. These experiments improved our results both in terms of prediction of alarm symbols and quality of extracted rules. Table 1 summarizes our results about prediction of alarm symbols for three different types of devices. The column with heading “Direct” lists the results after delaying one alarm, this same as time window 1. The other column “Delayed Inputs” lists the results after delaying ten alarms. Results in this column correspond to time window 10. Please note that these results do not come from a traditional c4.5 but a different c4.5 that preserves temporal order between symbols being predicted.

Table 1: Results of prediction from decision tree

Device/ Type	% Error in prediction of alarm Symbol/ID using DT	
	Direct	Delayed Inputs
1(130)	2.4	1.3
2(40)	1.3	0.4
3(47)	0.0	0.0

Table 2 and Table 3 list number of discovered rules and results of predictions based on those rules respectively. While understanding and analyzing this c4.5 code, it was found that the routines loadtree and savetree had problems. Both Table 2 and Table 3 have additional columns showing slightly different results. These results are given due to the fact that programs consult and consultr were not able to correctly test samples during interactive sessions. Both of these programs load decision tree saved by temporal c4.5 program.

The temporal tree once built in the memory worked fine for both symbol prediction and rule generation. It was also able to test both training and test samples. We fixed this loading routine in c4.5rules so that the rules be generated and evaluated with same tree produced by c4.5 decision tree generator. Due to these modifications we have listed outcomes with the original code and the modified code. Column with additional title TS-C4.5r lists results from c4.5rules available with time sleuth. The other column TS-C4.5r' lists results from modified code.
Table 2 presents number of rules discovered for time window 1 and 10. For time window 1 we were able to find rules to find which alarm is to come after this alarm and what is time interval for which this rule is valid. The other explanation of this type of rule could be that the next alarm b is expected to appear in this much time (< c seconds) if the current alarm is a. The alarm interval could be of the form >c and <d seconds too. Results of time window 10 or delaying ten alarms are given in Delayed Inputs column. Discovered rules in this case now cover impact of consecutive alarms to expect an alarm after occurrence of several others.

Table 2: Extraction of rules

Device/ Type	**No of Rules for Symbol Prediction**			
	Direct		**Delayed Inputs**	
	TS-C4.5r	*TS-C4.5r'*	*TS-C4.5r*	*TS-C4.5r'*
1(130)	3	3	5	6
2(40)	11	9	13	5
3(47)	14	14	67	2

In Table 3, we have provided the results of prediction based on discovered rules for both time window 1 and time window 10. Table3 also provides errors in evaluation of rules on training samples along with those listed for test samples separately. A symbol ♣ is used to identify results after evaluating training patterns using the discovered rules. The symbol ♦ identifies results for evaluation of test patterns.

Table 3: Results of prediction using extracted rules

Device/ Type	**% Error in prediction of alarm Symbol/ID using Extracted Rules**			
	Direct		**Delayed Inputs**	
	TS-C4.5r	*TS-C4.5r'*	*TS-C4.5r*	*TS-C4.5r'*
1(130)	1.8	1.8	1.6♣, 2.7♦	1.6♣, 1.9♦
2(40)	1.3	1.3	0.4♣, 5.0♦	2.2♣, 1.9♦
3(47)	0.0	0.0	0♣, 20.5♦	0♣, 20.5♦

5. Conclusions and Future Work

In this paper we have presented a temporal alarm prediction scheme based on a predictive tool known as TimeSleuth. TimeSleuth has been successfully employed as unsupervised tool for classification and prediction. Separation of time between consecutive alarms was the main variable of interest for such a symbolic time series data. Results based on this c4.5 variant were quite accurate in terms of alarm symbol prediction. However not all prediction rules were good enough for prediction in real-time environment. Empirical results show, how a predictive algorithm can be successfully employed in prediction of critical events. Efforts are now being made to use more variables including topological dependencies.

Our goal is to develop tools to transform the input data so as to enable the use of different predictive algorithms available in literature. This will enrich our understanding and applicability of predictive algorithms. The results would increase the amount of information necessary to determine the root cause of a problem and the amount of evidence to perform accurate predictions. We are currently working on modeling the whole PTCL access network in order to predict faults based on these temporal/causal rules. Once this complete software system becomes operational, it will be able to assist PTCL network operators to minimize downtime associated with faults in a multi-vendor equipment based environment.

5. References

[1] Karimi, K., and Hamilton, H.J., TimeSleuth: A Tool for Discovering Causal and Temporal Rules, 14th IEEE International Conference On Tools with Artificial Intelligence (ICTAI2002), Washington DC, USA, November 2002.

[2] Quinlan, J. R., "C4.5: Programs for Machine Learning", Morgan Kaufmann, 1993.

[3] Parzen, E, "Time Series Analysis of Irregularly Observed Data: Proceedings of a Symposium Held at Texas A&m University", Springer-Verlag, 1984.

[4] Bloomfield, P., "Fourier Analysis of Time Series: An Introduction", Wiley-Interscience; 2nd edition, 2000.

[5] Brockwell, P.A., Davis, R.A., "Introduction to Time Series and Forecasting", Springer-Verlag, 2002.

[6] Weigend, A.S., Gershenfeld, N.A., (editors), "Time Series Prediction: Forecasting the Future and Understanding the Past", Perseus, 1994.

[7] Liang, H., "Adaptive Fourier Analysis for unequally spaced Time Series Data", Ph.D. thesis, Virginia Polytechnic Institute and State University, 2002.

[8] Mathias, A. , Grond, F., Guardans, R., Seese, D., Canela, M., and Diebner, H.H., Algorithms for Spectral Analysis of Irregularly Sampled Time

Series, Journal of Statistical Software, 11, Issue 2, 2004.

[9] Haykin, S, Neural Networks: A Comprehensive Foundation (2nd Edition), Prentice Hall, India, 1998.

[10] Luo, F and R. Unbehauen, Applied Neural Networks for Signal Processing, Cambridge Press, Cambridge, 1998.

[11] Mitchell, T., "Machine Learning", McGraw Hill, 1997.

[12] Michie, D., Spiegelhalter, D.J., Taylor, C.C., "Machine Learning, Neural and Statistical Classification", Prentice Hall, 1994.

[13] Hastie, T., Tibshirani, R., Friedman, J.H., "The Elements of Statistical Learning", Springer; 1 edition 2001.

[14] Dunham, M.H., "Data Mining: Introductory and Advanced Topics", Pearson, 2003.

[15] Mandic, D.P. and Chambers, J.A., "Recurrent Neural Networks for Prediction", John Wiley & Sons, Chichester, 2001.

[16] Hand, D.J., Mannila, H., Smyth, P., "Principles of Data Mining", MIT Press, 2000

[17] Han, J., Kamber, J., "Data Mining: Concepts and Techniques", Elsevier, 2001.

[18] Cohen, W.W., Fast Effective Rule Induction in Proceedings of the Twelfth International Conference on Machine Learning 1995: 115-123.

[19] Karimi, K., Hamilton, H.J., RFCT: An Association-Based Causality Miner, The Fifteenth Canadian Conference on Artificial Intelligence (AI'2002), Calgary, Alberta, Canada, May 2002.

[20] Pearl, J., "Causality: Models, Reasoning and Inference", Cambridge University Press. 2000.

[21] Karimi, K., Hamilton, H.J., Using TimeSleuth for Discovering Temporal/Causal Rules: A Comparison of Methods, The Sixteenth Canadian Artificial Intelligence Conference (AI'2003), Halifax, Nova Scotia, Canada, June 2003. 175-189.

[22] Scheines, R., Spirtes, P., Glymour, C., Meek, C., Tetrad II: Tools for Causal Modeling, Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[23] Kennett, R.J., Korb, K.B., Nicholson, A.E., Seabreeze Prediction Using Bayesian Networks: A Case Study, Proc. Fifth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'01). Hong Kong, April 2001.

[24] Wallace, C. S., Korb, K. B., Learning Linear Causal Models by MML Sampling, Causal Models and Intelligent Data Management, Springer-Verlag, 1999.

[25] Jaudet, M., Hussain, A., Iqbal, N., Neural Networks for Fault-prediction in a Telecommunications Network, IEEE INMIC – 2004.

[26] Weiss, G. M., Timeweaver: a genetic algorithm for identifying predictive patterns in sequences of events., In Proceedings of the Genetic and Evolutionary Computation Conference (Orlando, Florida), edited by W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, pp. 719-725. San Francisco, CA: Morgan Kaufmann.

[27] Klemettinen, M., A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases, Ph.D. thesis, University of Helsinki, 1999.

[28] Vilalta, R., Apte, C.V., Hellerstein, J.L., Ma, S., and Weiss, S.M., Predictive algorithms in the management of computer systems, IBM Systems Journal, Volume 41, Number 3, 2002.

[29] Wietgrefe, H., Tuchs, K.D., Jobmann, K., et. al: "Using Neural Networks for Alarm Correlation in Cellular Phone Systems", International Workshop on Applications of Neural Networks to Telecommunications 1997 (IWANNT'97), Melbourne, Australia June 1997

[30] Weiss, G.M. and Hirsh, H., Learning to Predict Rare Events in Event Sequences, Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA (1998), pp. 359–363.

[31] Vilalta, R., Ma, S. and Hellerstein, J., Rule Induction of Computer Events, Proceedings of the 12th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, Springer-Verlag, Lecture Notes in Computer Science (2001).