The Application of Neural Networks to Predicting the Root Cause of Service Failures

Robert Harper
Moogsoft Ltd
River Reach, 31-35 High Street
Kingston-Upon-Thames, UK
rob@moogsoft.com

Philip Tee
Moogsoft Inc
1265 Battery St
San Francisco, CA 94111
phil@moogsoft.com

***Abstract*—The principal objective when monitoring compute and communications infrastructure is to minimize the *Mean Time To Resolution* of service-impacting incidents. Key to achieving that goal is determining which of the many alerts that are presented to an operator are likely to be the root cause of an incident. In turn this is critical in identifying which alerts should be investigated with the highest priority.**

Noise reduction techniques can be employed to reduce the quantity of alerts a network operator needs to examine but even in favorable scenarios there may be multiple candidate alerts that need to be investigated before the root cause of the incident can be accurately identified, resolved and full service resumed.

The current contribution describes a novel technique, *Probable Root Cause*, that applies supervised machine learning in the form of Neural Networks to determine the alerts most likely to be responsible for a service-impacting incident.

An evaluation of different models and model parameters is presented. The effectiveness of the approach is demonstrated against sample data from a large commercial environment.

I. Introduction

Network infrastructures together with the applications and services that run on them produce huge quantities of event data on an ongoing basis. In the realm of fault localization, the number of events generated on a reasonably sized commercial infrastructure can be of the order of hundreds per second while volumes in the largest enterprises can be many tens of thousands per second. There are countless event types each of which deliver differing amounts of information to a network operator. For example, some events may represent application *heartbeats*; others may be the result of active polling on a device to a fixed schedule to report hardware usage statistics; others may be asynchronously delivered whenever a threshold is reached or a particular scenario has been detected, an SNMP LinkDown trap for example. It is the role of a network operator to decode the alerts that are presented, to determine whether a service-impacting incident exists, which alerts are indicators of it and subsequently which alerts represent the underlying problem that needs to be remedied.

There are several techniques, loosely termed *noise reduction*, that can reduce the volume of alerts that an operator needs to examine, [1], [2]. At its simplest, noise reduction can be achieved via manual exclusion or *blacklisting*, [3], a process that is impractical at scale. Other techniques exist that use machine learning or rules-based approaches to group alerts that are related to the same incident [4]. These groups of alerts are created without regard for whether the alerts represent the symptoms of the outage, or its underlying cause.

Noise reduction is an invaluable tool for an operator, but it doesn't solve the problem of which alert to action first, resulting in a *Mean Time To Resolution* (MTTR) that is higher than desired. The techniques used to identify the reason for an outage are collectively known as *Root Cause Analysis* (RCA) as described in [5]. Many RCA techniques were invented when network and application topology changes were infrequent and rely upon a behavioral models of the infrastructure. The advent of technologies such as virtualization, micro-services, network traffic shaping etc. have created huge changes in the way that infrastructures and services are deployed and managed. In modern infrastructures, which combine highly dynamic network and application topologies and large volumes of event data, traditional RCA techniques have become impractical and largely obsolete [6], [7]. In particular, the demands placed upon RCA by the adoption of virtual datacenter technologies can introduce the *Overlay Network* fault diagnosis issue [8] and render behavioral models untenable. Further, none of these models leverage the expertise of the operations staff who have considerable practical understanding of which events are suspicious in terms of root cause.

In this paper we describe a new approach to predicting root cause alerts called *Probable Root Cause*[1] (PRC). Rather than analyzing network and application topologies alongside known failure scenarios to generate a static model [5], PRC applies supervised machine learning to alert data and associated operator feedback to *estimate* the probability that an alert is causal in any given failure scenario. Thresholds can be applied to this value to change the remediation activity undertaken.

We begin in section II by describing the details of the technique we are using to leverage neural networks for root cause prediction. In particular how we present features to the model in a way that is amenable to training and determination of root cause. In section III the methods used to train and validate the model are presented with particular emphasis on optimization of the input features and training set to achieve

[1]This work is supported by Moogsoft Inc. and describes patented features if its products.

the most favorable results against a fixed set of validation data. Section IV presents favorable results of the validated model against a set of pre-determined test data and a discussion of the utility of PRC in the context of a live operational environment.

II. Background

A. Definitions & Nomenclature

Throughout this paper the following definitions will be used:

Event An instance of a failure or status message.
Alert A state-based grouping of identical events.
Situation A group of alerts representing a service failure.

Events, alerts and situations are represented within management systems as a collection of name-value pairs. A minimal set of operationally useful fields for alert and situation records are shown in Tables I and II respectively.

The PRC technique described here trains two models. The first, *Alert PRC*, assigns a probability to each alert taken in isolation and based purely on its individual attributes and regardless of whether the alert forms part of a situation or not. This value acts as a predictor that operators can monitor before an outage has fully developed and act on preemptively to prevent a more serious outage occurring. The second model, *Situation PRC* also assigns a probability to an alert but based on its context, i.e. the situation in which the alert is observed. This value acts as a call to action and gives an operator a clear indicator as to which alerts should be actioned first in any situation. In a live PRC deployment, feedback from operators is required to identify the root cause and symptomatic alerts of a resolved situation and to train the model further.

B. Feature Extraction

The values of alert and situation attributes need to be converted into meaningful feature vectors for input into the learning algorithm. Not all of an alert's or a situation's attributes are informative when determining the root cause of an outage, in fact some attributes are only assigned a value once the likelihood of root cause has been determined. For example, the operator to whom a situation is assigned or the position of an alert is in its operational workflow are only given values once it is deemed actionable, attributes such as these are an output of PRC rather than an input to it.

When converting the value of an attribute into a feature vector the primitive type of the value but also its practical interpretation need to be considered. The values of some attributes represent distinct and unconnected categories, for example, the values for *Type* may be *Hardware*, *Network*, *Application* etc., where each value represents a distinct category. Other attributes require a different interpretation: *Count* is a numerical value with a natural order to it; *Source* is the host name of a device containing embedded conventions that a trained model needs to understand. In addition, while time is stored as a numerical delta from an epoch, derived features such as *hour of the day* or *minute of the hour* are generally more useful features. In the context of a situation, so-called *Ensemble Features* may be required. Ensemble features are not explicit attributes of a situation but are derived from the set of alerts it contains, for example, alerts that occur earlier in a situation may be more likely to be the root cause. The feature types required to encapsulate different attribute types are shown in Table III.

Conversion of the numerical and categorical feature types into feature vectors follow standard processes. Ordered numbers require no special conversion and categorical features are converted using so-called *One Hot Encoding*. However, both long and short textual feature types require special treatment.

Several operational requirements influenced the approach to feature extraction within PRC, including: transferability of a trained model from one operational environment to another and the ability for the model to learn incrementally as new training examples are received. The practical consequence of these requirements is that the size of the feature vectors need to be fixed at the beginning of the training process and that the training and classification processes must be able to accommodate new, previously unseen tokens as new data is presented to the model.

A common method of converting text based values to a feature vector is to adopt a token-counting approach. The text is divided into distinct tokens, such as words, phrases or shingles and the number of occurrences of each token is recorded in the feature vector. The length of the feature vector required to model an attribute therefore depends upon the number of distinct tokens across the entire corpus of that attribute. In this approach, when a new token is observed, that token can neither be incorporated into the model, restricting the development of the model over time, nor can its value impact the classification of the alert to which the token belongs. To avoid these issues we adopt *feature hashing*, [9], also known as the *hashtrick*, for all text features.

The differing treatment of short and long text features is defined by the tokenization strategy and the size of the vector that the corpus is collapsed onto. For attributes with longer values such as Description, the text is split into words and it is be expected that the vector size will be relatively large. Experiments conducted during development of PRC suggested that a feature vector sizes of at least 128 were required to provide the appropriate balance between accuracy and computational effort. For short text fields such as Source, tokenization is based on n-character shingles where $n \geq 2$, experiments during development of the technique suggested feature vectors of size between 16-48 were required. The optimal sizes for each feature and the tokenization regime form part of a model validation exercise, for the current work this process is described in Section III-D.

C. Feature Vector Construction

The Alert and Situation PRC models take different feature vectors, but both are constructed from the same fundamental components. The Alert PRC model takes instances of the *Alert Feature Vector*, $\mathbf{X}_{\mathbf{Alert}}$, the Situation PRC model takes instances of the *Situation Alert Feature Vector*, $\mathbf{X}_{\mathbf{SitAlert}}$.

1) Alert Feature Vector: We define S as the set of all alert attributes and T as the set of attributes upon which the model

is to be trained or the value of root cause predicted, such that $T \subset S$. Each attribute in T is converted into its own feature vector, $\mathbf{X_t}$ in a d_t dimensional vector space V_t, according to the type of the attribute value and its practical interpretation as described in Section II-B. $\mathbf{X_{Alert}}$ is constructed from the concatenation of these feature vectors embedded in a new vector space $F = \bigoplus_{t \in T} V_t$, which we write as:

$$\mathbf{X_{Alert}} = (\mathbf{X_{Alert}} \| \mathbf{X_t}) \, \forall t \in T \tag{1}$$

such that the dimensionality of $\mathbf{X_{Alert}}$, $d_{Alert} = \sum_{t \in T} d_t$

2) Situation Alert Feature Vector: The feature vector for a situation, $\mathbf{X_{Situation}}$ is defined as:

$$\mathbf{X_{Situation}} = \sum_{X_{Alert} \in Q} \mathbf{X_{Alert}} \tag{2}$$

where Q is the set of alerts in a situation. $\mathbf{X_{SitAlert}}$ is the concatenation of the feature vector for the alert, $\mathbf{X_{Alert}}$, and the situation in which it exists, $\mathbf{X_{Situation}}$ such that:

$$\mathbf{X_{SitAlert}} = \mathbf{X_{Situation}} \| \mathbf{X_{Alert}} \tag{3}$$

D. Neural Network Architecture

The neural network used in this work is a multi-layer perceptron, [10], [11], with a single hidden layer. The size of the input layer, N_{Input}, is governed by the dimensionality of $\mathbf{X_{Alert}}$ or $\mathbf{X_{SitAlert}}$. As PRC is fundamentally a binary classification problem the output layer, N_{Output}, contains 2 nodes. The size of the hidden layer, N_{Hidden}, is chosen automatically using the convention $N_{Hidden} = 2\,(N_{Input} + N_{Output})\,/3$.

Non-categorical features in the input feature matrix are normalized to zero-mean and unity variance. Standard sigmoid activation, back-propagation and L2 regularization is employed as described in [10], [11] alongside the conjugate gradient optimization routine described in [12].

III. Experiments

In this section we describe the process of training, validating and subsequently testing the Alert and Situation PRC models.

A. Training Data

The evaluation process followed standard practices for training and validating a classifier. The labelled data was split in the ratios 60:20:20 between training, validation and test data. The content of each dataset remained fixed for each trial and in cases where only part of the training set was used, the models were tested using the complete validation and test datasets.

In the current work a single piece of labelled data refers to a situation in which the root cause alert, or alerts, have been identified. In general a situation will contain at least one root cause alert and a greater number of symptomatic alerts. In the data used here there is an average of about nine alerts per situation and the non-root cause alerts were labelled as symptoms. In practice, the number of alerts in a situation is unconstrained and it does not automatically follow that a non-root cause alert should be labelled as a symptom.

A total of approximately 3000 labelled situations were extracted from a live deployment of an Incident.MOOG system, [13]. Alert labelling was facilitated automatically via a root cause attribute assigned by a topology based, legacy system available in the deployment environment. A live PRC deployment would rely upon operator feedback to label the root cause alerts and their symptoms.

B. Evaluation Measures

The trained PRC models are initially evaluated using the F_1 score. There are known limitations with the F_1 score in certain classification scenarios, [14], but the deficiencies were considered acceptable during model validation given the ease with which F_1 can be calculated. A more comprehensive evaluation of the validated model in Section IV uses additional measures including the Receiver Operating Characteristic (ROC), [15].

C. Feature Selection

For the purposes of this investigation and owing to a considerable depth of knowledge of the labelled data, the feature selection process was a manual one. The *Description* and *Agent* attributes, Table I, were identified as the most discriminatory features and are used throughout the current work. Agent was treated as *Categorical Text*, while the Description was encoded as *Long Text*, see Table III and Section II-B. Alert and Situation feature vectors were constructed as described in Section II-C.

D. Description Feature Size

An important attribute of PRC is the ability to project an arbitrary set of tokens onto a fixed length feature vector. Figure 1 shows the variation of F_1 score with the length of the Description feature count for both the Alert and Situation PRC models. In common with all results presented here the accuracy of the Alert PRC model is slightly below that of the Situation PRC model. In both cases the F_1 score has a value of approximately 0.80 for a feature count of only 8, climbing to approximately 0.90 for a feature count of 64. Higher feature counts show the F_1 score reaching an asymptotic value of approximately 0.93. The experiments described subsequently were all performed using a Description feature count of 128. The relative accuracy of the model at low feature counts was a little surprising, it is postulated that this may be due to a relatively small number of alert templates being responsible for root cause alerts. Further investigation of this finding is required but is not covered in this work.

E. Training Set Size

The impact of training set size on the Alert and Situation PRC Models are shown in Figures 2 and 3 respectively. Both plots display the F_1 score against the number of labelled situations in the training set and display similar characteristics. For small training sets both models have the characteristics of a high variance model, specifically, low error when evaluated against the training data, ($F_1 \approx 1.0$), but a relatively high error when tested against the validation data, ($F_1 \approx 0.8$). As

TABLE I
TYPICAL FIELDS IN AN EVENT OR ALERT RECORD

Name	Type	Description
Agent	Text	The name of the agent that created the event e.g. (SCOM, NAGIOS, SNMP TRAP etc.)
Class	Text	Classification for the event in a hierarchy to identify simple event ontologies, see *Type*
Count	Number	A running count of the number of de-duplicated events for this alert
Description	Text	A human interpretable summary of the event
Location	Text	The geographic or physical location of the agent e.g. NYC Data Centre, (51.407139, -0.307321), London
Manager	Text	A general identifier of the event generator or intermediary process e.g. NAGIOS, SCOM
Owner	Number	The system wide ID for the operator that is currently assigned to resolve this alert
Severity	Number	An indicator, in the range of 0-5, of a event's importance, by convention 0 is the lowest severity.
Source	Text	A unique human readable hostname for the source of the event
State	Number	The current state of the alert, primarily used to indicate where the alert is in an operational workflow
Timestamp	Time	The timestamp of when the event occurred
Type	Text	A sub-classification for the event in a hierarchy to identify simple event ontologies, see *Class*

TABLE II
TYPICAL FIELDS IN A SITUATION RECORD

Name	Type	Description
Created	Time	When the situation was created
Description	Text	A human readable summary of the situation
Moderator	Number	The operator ID that is responsible for resolution of this situation
Priority	Number	A hint, in the range of 0-5, to indicate the importance of this situation
Processes	Text/Number	The processes that are impacted by this situation
Queue	Text	A user-defined workflow categorisation of the situation
Services	Text/Number	The high-level business services that are impacted by this situation
State	Number	The current state of the alert, primarily used to indicate where the alert is in an operational workflow

TABLE III
FEATURE TYPES

Feature Type	Example Attribute	Description
Time	Created	The timestamp expressed as 'hour of the day' and 'minute of the hour' for example
Ordered Number	Count	A numerical field with a link between the magnitude of the value and its meaning
Categorical Number	Severity	A numerical field where there is no logical link between the different values
Categorical Text	Agent	A Text field with with no logical link between words regardless lexical similarity
Short Text	Source	A treatment that aims to extract patterns from within individual tokens
Long Text	Description	A word/phrase based treatment that aims to extract patterns from the text as a whole

the size of the training set is increased the F_1 score and hence the underlying error between the training and validation sets converge to approximately 0.91 and 0.93 for the Alert and Situation PRC models respectively. We consider these F_1 scores to demonstrate an acceptable level of accuracy for the current model, consequently, for all further experiments and tests the complete set of training data, comprising approximately 1750 training examples was used.

IV. MODEL EVALUATION

Section III described the processes used to train and validate the Alert PRC and Situation PRC models. In this section we evaluate the optimized models against a set of test data, the same models were used for all test cases.

Raw statistics for the performance of the Alert PRC and Situation PRC models are shown in Tables IV and V respectively for a variety of *Root Cause Threshold* (RC Threshold) values. RC Threshold is defined as the probability above which

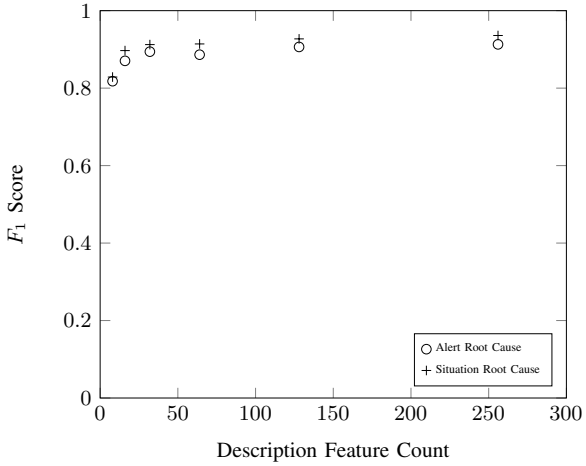

Fig. 1. F_1 Score vs. Description Feature Size

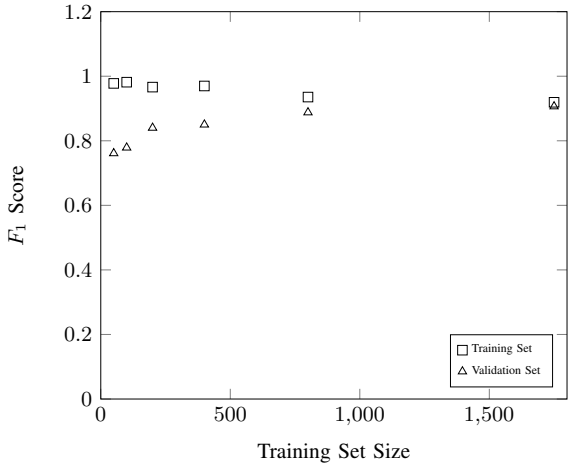

Fig. 2. Alert PRC : F_1 Score vs. Training Set Size

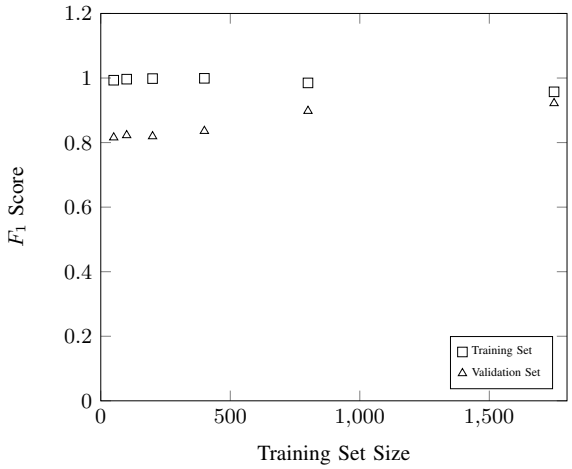

Fig. 3. Situation PRC : F_1 Score vs. Training Set Size

an alert is classified as a root cause and below which it is considered a symptom.

Figure 4 shows the variation of the F_1 score with RC Threshold. As was also observed in Section III, the accuracy of the Situation PRC model is consistently higher than the Alert PRC model. The additional context available in the Situation PRC model allows a different root cause probability to be predicted for identical alerts that appear in different situations. In the same scenario the Alert PRC model is only be able to predict a single value for the probability of root cause. Closer examination of Figure 4 shows a remarkably consistent F_1 score for both models for values of RC Threshold between 0.2 and 0.7. At values of RC Threshold outside these bounds there is a considerable drop-off in accuracy. Interesting however, is the higher rate of drop-off in the Alert PRC model, we propose that the inherent lack of context in the Alert PRC model is the cause.

All learned models exhibit conflicting accuracy characteristics so it is important to find an acceptable compromise when applying a model to a real-world application. Intuition

TABLE IV
ALERT PRC : ACCURACY STATISTICS

RC Threshold	Precision	FPR	TPR	F_1 Score	ROC Area
0.01	0.623	0.183	0.994	0.766	0.812
0.05	0.774	0.088	0.985	0.867	0.899
0.10	0.817	0.067	0.979	0.891	0.913
0.20	0.844	0.055	0.971	0.903	0.918
0.30	0.870	0.044	0.964	0.914	0.922
0.40	0.887	0.037	0.951	0.918	0.917
0.50	0.898	0.032	0.928	0.913	0.898
0.60	0.914	0.026	0.911	0.912	0.887
0.70	0.929	0.020	0.877	0.902	0.859
0.80	0.942	0.015	0.786	0.857	0.774
0.90	0.967	0.007	0.632	0.765	0.628
0.95	0.991	0.001	0.477	0.644	0.477
0.99	0.997	0.000	0.374	0.544	0.374

suggests that the RC Threshold value should be 0.5, however, that assumption provides no control over the number of true positive or false positive results. In an operational environment capturing more root cause alerts and consequently reducing the MTTR of critical outages may be acceptable, even at the expense of more false positives. The data in Tables IV and V and the corresponding ROC plot in Figure 5 are used to demonstrate this. The ROC plot shows the trade-off between the *True Positive Rate* (TPR) and the *False Positive Rate* (FPR), the closer the curve to the top left corner of the plot the more accurate the model, both models exhibit very favourable characteristics in this regard. The conclusion of [16] is that the optimal algorithm for a problem is the one that gives the lowest area under the ROC curve. By extension we postulate that the optimal characteristic *within* a model is the one which captures the largest area within that model's ROC curve. Examination of Tables IV and V show that the TPR for both models rises as the RC Threshold value falls, however FPR rises at a slower rate. The maximum value for the area within the ROC curve for each value of RC Threshold is shown and has a maximum value of 0.92 at an RC Threshold of 0.3 for the Alert PRC model and 0.93 at an RC Threshold of 0.2 for the Situation PRC model. These points are highlighted in Tables IV and V and in the magnified area of Figure 5. For the data presented here, an RC Threshold potentially as low as 0.1 could be chosen without compromising the efficiency of the fault resolution process because of too many false positive results. Ultimately the choice of RC Threshold depends purely on the operational requirements.

V. CONCLUSION

In this paper we have presented a novel technique for predicting the root cause alerts of service failures in compute and communication infrastructures, *Probable Root Cause*. The technique does away with the need for the behavioral models common in legacy RCA systems and consequently is well suited to rapidly changing, virtualized infrastructures that are common today. The technique uses alert data alongside operator feedback to train a neural network and hence leverages

TABLE V
SITUATION PRC : ACCURACY STATISTICS

RC Threshold	Precision	FPR	TPR	F_1 Score	ROC Area
0.01	0.767	0.093	0.987	0.863	0.895
0.05	0.841	0.057	0.978	0.904	0.910
0.10	0.871	0.045	0.973	0.919	0.929
0.20	0.900	0.033	0.964	0.931	0.932
0.30	0.920	0.026	0.955	0.937	0.930
0.40	0.931	0.022	0.946	0.939	0.926
0.50	0.933	0.021	0.935	0.934	0.916
0.60	0.944	0.017	0.921	0.932	0.905
0.70	0.956	0.013	0.899	0.927	0.888
0.80	0.963	0.010	0.866	0.912	0.858
0.90	0.969	0.008	0.795	0.874	0.789
0.95	0.986	0.003	0.713	0.828	0.711
0.99	0.994	0.001	0.544	0.703	0.544

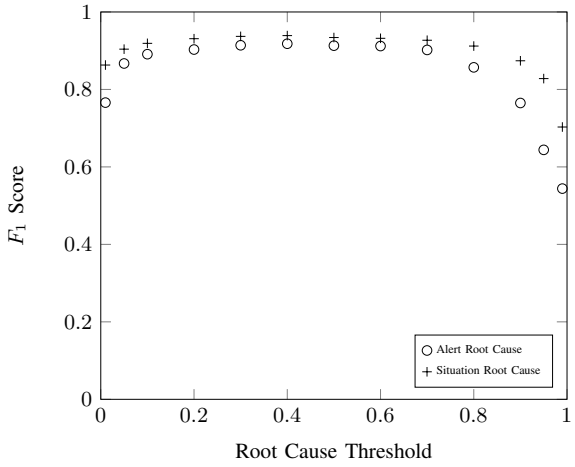

Fig. 4. F_1 Score vs. Root Cause Threshold

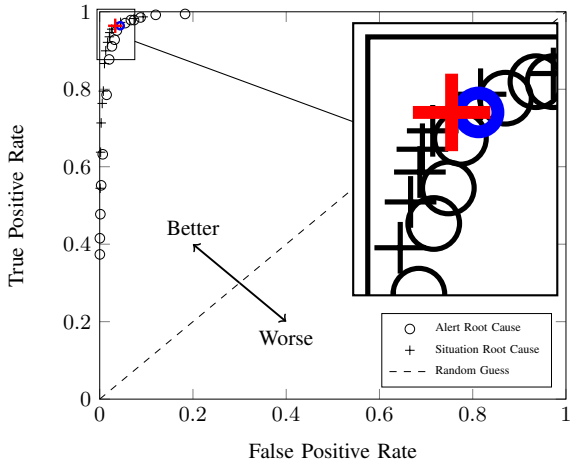

Fig. 5. Receiver Operating Characteristic

the expertise of the operations staff charged with maintaining strict service levels.

The underlying premise of PRC and the methods used to formulate the models were trained, validated and tested against labelled data obtained from a large commercial environment. The validated models show a high level of accuracy as measured by the F_1 score and ROC. We would expect equivalent performance if the models were deployed into an environment with similar event sources and failure scenarios. Deployment in a substantially different environment would require a new training and validation exercise.

The models display some operationally significant characteristics, which allow the root cause threshold to be lowered significantly below expected levels such that the rate of true positive results increased without the number of false positive results increasing to unacceptable levels.

In order to meet operational requirements and reduce computational overhead, PRC employs techniques to control the dimensionality of a variable size feature space. It was demonstrated that highly-dimensional text attributes could be projected onto low dimensional feature spaces without sacrificing accuracy when measured by F_1 score or ROC, we intend to conduct further research into this intriguing result.

REFERENCES

[1] R. Harper, "Entropy & The Science of Noise," 2016. [Online]. Available: https://www.moogsoft.com/whats-new/entropy-noise/

[2] P. Tee, G. Parisis, and I. Wakeman, "Towards an approximate graph entropy measure for identifying incidents in network event data," in *IEEE/IFIP Network Operations and Management Symposium*, 2016.

[3] L. Metcalf and J. M. Spring, "Blacklist Ecosystem Analysis Spanning Jan 2012 to Jun 2014," *ACM Digital Library*, 2014.

[4] P. Bodík, "Automating Datacenter Operations Using Machine Learning," Ph.D. dissertation, 2010.

[5] M. Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," *Science of Computer Programming*, 2004.

[6] M. Miyazawa and K. Nishimura, "Scalable root cause analysis assisted by classified alarm information model based algorithm," in *7th International Conference on Network and Service Management*, 2011.

[7] Smarts, "Downstream Suppression is Not Root Cause Analysis," Tech. Rep., 2002.

[8] Y. Tang, E. Al-Shaer, and K. Joshi, "Reasoning under Uncertainty for Overlay Fault Diagnosis," *IEEE Transactions on Network and Service Management*, 2012.

[9] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009.

[10] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 2007.

[11] M. A. Nielsen, "Neural Networks and Deep Learning." [Online]. Available: http://neuralnetworksanddeeplearning.com

[12] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *ESAIM: Mathematical Modelling and Numerical Analysis - Modélisation Mathématique et Analyse Numérique*, 1969.

[13] Moogsoft, "Incident.MOOG Documentation," 2016. [Online]. Available: http://docs.moogsoft.com

[14] D. M. W. Powers and Ailab, "Evaluation: from Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies*, 2011.

[15] P. A. Flach, "The geometry of ROC space: understanding machine learning metrics through ROC isometrics," in *Proc. 20th International Conference on Machine Learning*, 2003.

[16] A. P. Bradley and A. P., "The use of the area under the ROC curve in the evaluation of machine learning algorithms," *Pattern Recognition*, 1997.