CueT: Human-Guided Fast and Accurate Network Alarm Triage

Saleema Amershi[†‡], **Bongshin Lee**[†], **Ashish Kapoor**[†], **Ratul Mahajan**[†], **Blaine Christian**[*]

[†] Microsoft Research
One Microsoft Way
Redmond, WA 98052
{bongshin, akapoor, ratul}
@microsoft.com

[‡]Computer Science & Engineering
DUB Group,
University of Washington
Seattle, WA 98195
samershi@cs.washington.edu

[*]Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
blainech@microsoft.com

ABSTRACT

Network alarm triage refers to grouping and prioritizing a stream of low-level device health information to help operators find and fix problems. Today, this process tends to be largely manual because existing tools cannot easily evolve with the network. We present CueT, a system that uses interactive machine learning to learn from the triaging decisions of operators. It then uses that learning in novel visualizations to help them quickly and accurately triage alarms. Unlike prior interactive machine learning systems, CueT handles a highly dynamic environment where the groups of interest are not known a-priori and evolve constantly. A user study with real operators and data from a large network shows that CueT significantly improves the speed and accuracy of alarm triage compared to the network's current practice.

Author Keywords

Interactive Machine Learning, Visualization, Triage.

ACM Classification Keywords

H.5.2. [User Interfaces]: Graphical User Interface; I.2.6 [Learning].

General Terms

Algorithms, Design, Human Factors.

INTRODUCTION

Triaging alarms is the first line of defense for modern computer networks. Triage is the process by which the multitude of low-level alarms (e.g., high link utilization, fan failure) generated by individual devices are grouped and prioritized. These groups are investigated by network engineers for problems with the network. Grouping related alarms, which likely stem from the same problem, helps engineers by reducing the number of issues they need to investigate and giving them a broader view of the problem than what an individual alarm provides. It is critical that triage is fast and accurate so engineers are not misled and problems can be identified and resolved quickly.

Many automated systems have been developed for alarm triage (see [9, 17] for reviews). However, despite years of effort, these systems are never fully accurate due to the complexity of the problem. Large networks have thousands of diverse devices, each generating a different set of alarms. Further, because each network is different and the set of devices within a network changes with time, it is very challenging to develop systems that work across networks. Therefore, to cope with the inherent inaccuracy of automated systems, networks invariably employ human operators (so called "Tier 1" operators) for alarm triage. These operators are required to sift through the thousands of alarms per day that can be missed by automation.

We explore a fundamentally different approach for alarm triage. Since human operators need to be involved in the triage process anyway, we ask if we can learn from their actions and in turn use that learning to assist them. By learning continuously and in situ from operator actions, the assistance provided would better fit the network and its unique practices as well as evolve with the network.

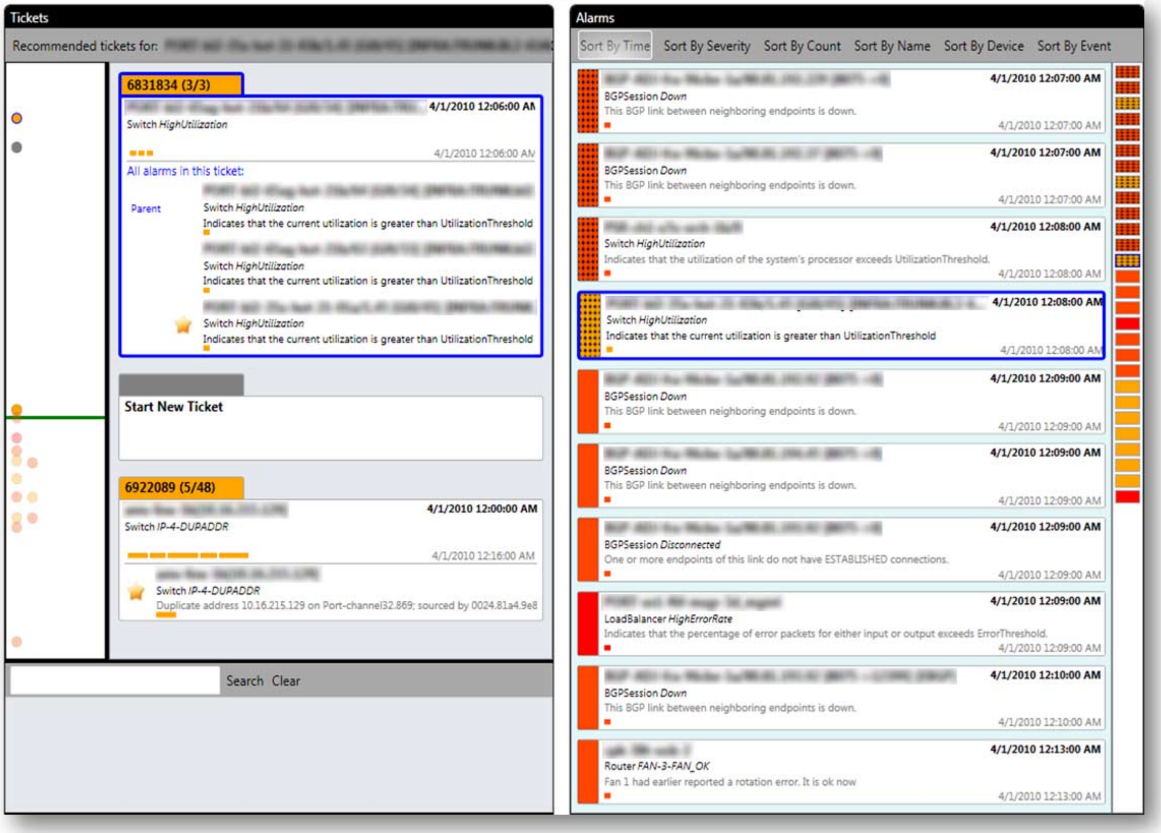
We develop CueT (Figure 1), a system that combines novel visualizations with interactive machine learning for fast and accurate alarm triage. CueT maintains a constantly-updating, machine-learning-based model for the triage process. As alarms arrive, CueT provides an operator with recommendations on how to group the alarms based on its model at that instant. The operator inspects the recommendations along with CueT's visualization of its confidence in them, and decides how to triage. CueT then learns from these operator actions and updates its model.

Our evaluation of CueT demonstrates the effectiveness of our approach. Testing with human operators and real data from a large, global network with approximately 15,000 devices, we show that CueT significantly improves the accuracy and reduces the time required for alarm triage.

The contributions of this paper are:

- A novel approach to alarm triage. Our approach also applies to many other scenarios where a human needs help with organization of a continuous data stream (e.g., RSS information feeds, email management, social

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
Copyright 2011 ACM 978-1-4503-0267-8/11/05....$10.00.



Tickets

Recommended tickets for: [redacted]

6831834 (3/3)

[redacted] **4/1/2010 12:06:00 AM**
Switch *HighUtilization*
4/1/2010 12:06:00 AM

All alarms in this ticket:

Parent
[redacted]
Switch *HighUtilization*
Indicates that the current utilization is greater than UtilizationThreshold

[redacted]
Switch *HighUtilization*
Indicates that the current utilization is greater than UtilizationThreshold

[redacted]
Switch *HighUtilization*
Indicates that the current utilization is greater than UtilizationThreshold

Start New Ticket

6922089 (5/48)

[redacted] **4/1/2010 12:00:00 AM**
Switch *IP-4-DUPADDR*
4/1/2010 12:16:00 AM

[redacted]
Switch *IP-4-DUPADDR*
Duplicate address 10.16.215.129 on Port-channel32.869; sourced by 0024.81a4.9e8

Search Clear

Alarms

Sort By Time Sort By Severity Sort By Count Sort By Name Sort By Device Sort By Event

[redacted] **4/1/2010 12:07:00 AM**
BGPSession *Down*
This BGP link between neighboring endpoints is down.
4/1/2010 12:07:00 AM

[redacted] **4/1/2010 12:07:00 AM**
BGPSession *Down*
This BGP link between neighboring endpoints is down.
4/1/2010 12:07:00 AM

[redacted] **4/1/2010 12:08:00 AM**
Switch *HighUtilization*
Indicates that the utilization of the system's processor exceeds UtilizationThreshold.
4/1/2010 12:08:00 AM

[redacted] **4/1/2010 12:08:00 AM**
Switch *HighUtilization*
Indicates that the current utilization is greater than UtilizationThreshold
4/1/2010 12:08:00 AM

[redacted] **4/1/2010 12:09:00 AM**
BGPSession *Down*
This BGP link between neighboring endpoints is down.
4/1/2010 12:09:00 AM

[redacted] **4/1/2010 12:09:00 AM**
BGPSession *Down*
This BGP link between neighboring endpoints is down.
4/1/2010 12:09:00 AM

[redacted] **4/1/2010 12:09:00 AM**
BGPSession *Disconnected*
One or more endpoints of this link do not have ESTABLISHED connections.
4/1/2010 12:09:00 AM

[redacted] **4/1/2010 12:09:00 AM**
LoadBalancer *HighErrorRate*
Indicates that the percentage of error packets for either input or output exceeds ErrorThreshold.
4/1/2010 12:09:00 AM

[redacted] **4/1/2010 12:10:00 AM**
BGPSession *Down*
This BGP link between neighboring endpoints is down.
4/1/2010 12:10:00 AM

[redacted] **4/1/2010 12:13:00 AM**
Router *FAN-3-FAN_OK*
Fan 1 had earlier reported a rotation error. It is ok now
4/1/2010 12:13:00 AM

Figure 1. CueT's interface. Alarms stream in from the network and are displayed on the right. CueT's triage recommendations for each alarm appear on the left along with a visualization of CueT's confidence in those recommendations (far left). Device Names and other information are blurred for security reasons.

network updates). It is designed to reside in a dynamic environment, where it learns from each human action to help with organizing the incoming data stream.

- The design and implementation of CueT, a system that embodies our approach in the context of triaging network alarms. CueT tightly integrates visualization with the underlying computation, in contrast to existing work on network monitoring which tends to focus on one aspect or the other. CueT is based on an interactive machine learning technique operating in a highly dynamic environment; existing work assumes a fixed set of groups that are known a-prior.
- Evaluation of CueT in a real-world environment with real operators and its comparison against the current practice of manual triage, showing that our approach leads to faster and more accurate triage.

RELATED WORK

Most research in network monitoring and alarm correlation has focused either on visualizations or automated solutions in isolation. Researchers have recently proposed visualization systems for network monitoring and diagnosis (e.g., [7, 14]). For example, the Visual-I system [7] uses visual grouping and scatterplots to highlight correlations between multiple devices and problems. While these systems leverage human visual and cognitive abilities to process data and look for patterns, our approach combines visualization and interactive machine learning in order to further reduce the cognitive effort of manually identifying patterns in large data sets. Helping automate pattern discovery can increase operator efficiency and accuracy, necessary for these time critical problems.

Automated alarm correlation (also related to root cause analysis and fault localization) has long been an active area of research because of the complexity of the problem and the potential impact on the industry [9, 17]. Most solutions to this problem have taken the form of expert defined rules or models (e.g., [1, 3, 9, 12, 15]). Such approaches require manual configuration of rules which can be difficult to obtain, are not robust to new situations, and require frequent maintenance [9, 17]. In contrast, our interactive machine learning approach can handle general alarm correlations that require alarm grouping based on similarity.

Some researchers have explored automatically learning rules or models from data that can then be used for automated alarm correlation and filtering (e.g., [13, 17]). However, most of these approaches require extensive training periods and must be retrained whenever the network topology changes [17]. In contrast, our approach is based on a dynamically changing model that is constantly learning from human guidance. We argue that human input is critical during the alarm correlation process because of the inherently inconsistent and ambiguous nature of the problem [9], which is a contributing factor to why alarm correlation is still an open area of research.

While today's commercial systems employ some of these previous techniques for alarm triage and filtering [5, 10, 18], most companies still require teams of Tier 1 operators to manually triage the thousands of remaining alarms unhandled by these existing systems. Furthermore, most of these systems tend to use a tabular representation of the data, which can make it difficult for Tier 1 operators to find patterns and group alarms (e.g., Figure 2). Our approach combines interactive machine learning with an appropriate visualization in order to deal with general alarm triage that is difficult to handle with existing approaches.

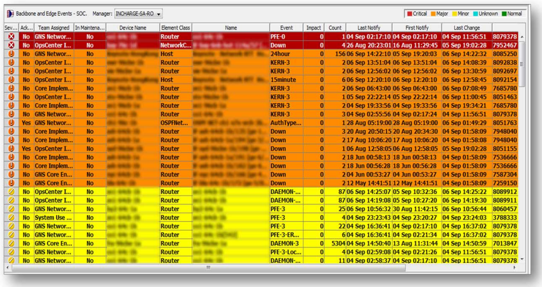
Our work is closely aligned with evolving research on interactive machine learning [2, 4, 6, 8]. All of these works assert the importance of human involvement and propose interactive systems for various applications (e.g., image segmenting [6], grouping documents [2, 4] and image search [8]). While all of these focus on a pool-based static environment, where the categories of interest and the pool of unlabeled data are known a-priori, our work involves a more challenging scenario. CueT deals with a highly dynamic environment where data is constantly streaming in and out and the set of labels to be assigned is constantly changing over time. Such dynamic scenarios, to the best of our knowledge, have not been addressed earlier in the realm of interactive machine learning.

UNDERSTANDING THE TRIAGE PROCESS

We visited the network operations center of a large organization on multiple occasions to understand current triage practices. We first talked informally with center managers to get a broad understanding of the process. We then closely observed two Tier 1 operators (1 female) for an hour each as they were triaging alarms using the current tool employed at this network operations center, and asked them questions during and after the observations.

The current tool used at this operations center is a combination of two pieces: (1) an automated rule-based engine that selects alarms to be shown to Tier 1 operators and labels those alarms (e.g., alarm severity level) and (2) a table-based UI for operators to triage alarms using labels and other features (Figure 2).

Our investigation revealed three main failings of the current process. First, the rule-based engine is hard to configure and keep up-to-date as the network evolves. As a result, thousands of alarms must still be manually triaged by operators every day. While triaging, the operators largely ignored the tool's output, except for some basic information like alarm severity that is used for prioritizing which alarm to focus on next. Much of this information is not computed by the tool but is directly configured by network managers.



Backbone and Edge Events - SOC. Manager: INCHARGE-SA-RO ▾

🟥 Critical 🟧 Major 🟨 Minor 🟦 Unknown 🟩 Normal

Sev...	Ack...	Team Assigned	In Maintena...	Device Name	Element Class	Name	Event	Impact	Count	Last Notify	First Notify	Last Change	
⊗	No	GNS Networ...	No	[redacted]	Router	[redacted]	PFE-0	0	1	04 Sep 02:17:10	04 Sep 02:17:10	04 Sep 11:56:51	8079378
⊗	No	OpsCenter I...	No	[redacted]	NetworkC...	[redacted]	Down	0	4	26 Aug 20:23:01	16 Aug 11:29:45	05 Sep 19:02:28	7952467
❗	No	GNS Networ...	No	[redacted]	Host	[redacted]	24hour	0	156	06 Sep 14:22:10	05 Sep 19:20:03	06 Sep 14:22:32	8085250
❗	No	OpsCenter I...	No	[redacted]	Router	[redacted]	KERN-3	0	2	06 Sep 13:51:04	06 Sep 13:51:04	06 Sep 14:08:39	8092838
❗	No	OpsCenter I...	No	[redacted]	Router	[redacted]	KERN-3	0	2	06 Sep 12:56:02	06 Sep 12:56:02	06 Sep 13:30:59	8092697
❗	No	OpsCenter I...	No	[redacted]	Host	[redacted]	15minute	0	6	06 Sep 12:20:10	06 Sep 12:20:10	06 Sep 12:58:45	8092154
❗	No	Core Implem...	No	[redacted]	Router	[redacted]	KERN-3	0	2	06 Sep 06:43:00	06 Sep 06:43:00	06 Sep 07:08:49	7685780
❗	No	OpsCenter I...	No	[redacted]	Router	[redacted]	KERN-3	0	1	05 Sep 22:22:14	05 Sep 22:22:14	06 Sep 11:00:45	8051463
❗	No	Core Implem...	No	[redacted]	Router	[redacted]	KERN-3	0	2	04 Sep 19:33:56	04 Sep 19:33:56	04 Sep 19:34:21	7685780
❗	No	GNS Networ...	No	[redacted]	Router	[redacted]	KERN-3	0	3	04 Sep 02:55:56	04 Sep 02:17:24	04 Sep 11:56:51	8079378
❗	Yes	GNS Networ...	No	[redacted]	OSPFNet...	[redacted]	AuthType...	0	1	28 Aug 05:19:00	28 Aug 05:19:00	06 Sep 01:49:29	8051763
❗	No	Core Implem...	No	[redacted]	Router	[redacted]	Down	0	3	20 Aug 20:50:15	20 Aug 20:34:30	04 Sep 01:58:09	7948040
❗	No	Core Implem...	No	[redacted]	Router	[redacted]	Down	0	2	17 Aug 10:06:20	17 Aug 10:06:20	04 Sep 01:58:08	7948040
❗	Yes	OpsCenter I...	No	[redacted]	Router	[redacted]	Down	0	1	06 Aug 12:58:05	06 Aug 12:58:05	05 Sep 19:02:28	8051155
❗	No	Core Implem...	No	[redacted]	Router	[redacted]	Down	0	2	18 Jun 00:58:13	18 Jun 00:58:13	04 Sep 01:58:09	7536666
❗	No	Core Implem...	No	[redacted]	Router	[redacted]	Down	0	2	18 Jun 00:56:28	18 Jun 00:56:28	04 Sep 01:58:09	7536666
❗	No	GNS Core En...	No	[redacted]	Router	[redacted]	Down	0	2	04 Jun 00:53:27	04 Jun 00:53:27	04 Sep 01:58:09	7587304
❗	No	GNS Core En...	No	[redacted]	Router	[redacted]	Down	0	2	12 May 14:41:51	12 May 14:41:51	04 Sep 01:58:09	7259150
⊘	No	OpsCenter I...	No	[redacted]	Router	[redacted]	DAEMON-...	0	87	06 Sep 14:25:07	05 Sep 10:32:36	06 Sep 14:25:22	8089912
⊘	No	OpsCenter I...	No	[redacted]	Router	[redacted]	DAEMON-...	0	87	06 Sep 14:19:08	05 Sep 10:27:20	06 Sep 14:19:30	8089911
⊘	No	GNS Networ...	No	[redacted]	Router	[redacted]	PFE-3	0	25	06 Sep 10:56:32	30 Aug 11:42:15	06 Sep 10:56:44	8060457
⊘	No	System Use ...	No	[redacted]	Router	[redacted]	PFE-3	0	4	04 Sep 23:23:43	04 Sep 23:20:27	04 Sep 23:24:03	3788333
⊘	No	GNS Networ...	No	[redacted]	Router	[redacted]	PFE-3	0	22	04 Sep 16:36:41	04 Sep 02:17:10	04 Sep 16:37:02	8079378
⊘	No	GNS Networ...	No	[redacted]	Router	[redacted]	PFE-3-ER...	0	6	04 Sep 16:36:41	04 Sep 02:21:34	04 Sep 16:37:02	8079378
⊘	No	GNS Core En...	No	[redacted]	Router	[redacted]	DAEMON-3	0	5304	04 Sep 14:50:40	13 Aug 11:31:44	04 Sep 14:50:59	7013847
⊘	No	GNS Networ...	No	[redacted]	Router	[redacted]	PFE-3-Loc...	0	4	04 Sep 02:59:08	04 Sep 02:21:26	04 Sep 11:56:51	8079378
⊘	No	GNS Networ...	No	[redacted]	Router	[redacted]	DAEMON-...	0	11	04 Sep 02:58:37	04 Sep 02:17:10	04 Sep 11:56:51	8079378

Figure 2. Current tool for displaying alarms in the network operations center we observed. A separate tool with a similar interface is used for displaying tickets.

Second, the triage process is highly manual partly because network managers often have specific instructions on how operators should temporarily handle certain types of alarms. These instructions are based on how managers expect the network to behave as they are making changes behind the scenes. They are typically conveyed in a formal document known as a *whiteboard*, updated daily and that operators are instructed to review before beginning a workday. Such special case instructions make it hard to develop tools for assisting with alarm triage. This difficulty applies not only to the traditional methods, but also to our approach as short-term special cases create noise in our learning model.

Third, the table-based visualization for the current triage process is rudimentary and inefficient. The alarms are presented as rows in a table that can be ordered by alarm attributes (e.g., severity). Another table in a different window shows the recently created alarm groups. The operators need to switch between these two windows to handle an alarm.

In addition, we learned that some alarm attributes are more important than others for the triage process. Understanding the importance of alarm attributes helps us not only choose the right features for the machine learning algorithm but also design appropriate visual representations.

While we studied only one network in detail, based on conversations with the network managers (who have a broader view of the industry and have also worked at other networks), we understand that the processes at this network are similar to those of many others. Next, we describe how CueT addresses these problems by interleaving human interaction with machine learning and visualization.

CUET

CueT consists of two interacting components: (1) a stream-based interactive machine learning engine for making triage recommendations and (2) an interface to assist operators in inspecting and interpreting those recommendations and feeding operator actions back into the learning engine. We will refer to the alarm currently being triaged as the "incoming alarm," a group of one or more related alarms that have already been triaged as a "ticket," and the current set of tickets that are being used for recommendations (and have not been discarded yet) as the "working set of tickets."

Stream-Based Interactive Machine Learning

One key aspect of our problem scenario is that it resides in a dynamic environment, where both the alarms and tickets being generated are constantly changing. Thus, our scenario is fairly different than the classical machine learning setting, where the classes are fixed and known a-priori, and the classifier operates in an assumed static environment. We tackle the challenges due to the dynamic environment and ever evolving set of classes (i.e., the working set of tickets and the alarms within those tickets).

Our approach builds on nearest neighbor classification. In particular, CueT provides triage recommendations for an incoming alarm via a nearest-neighbor strategy that orders the working set of tickets by their similarity to that alarm. Similarity is measured using a distance function that can adapt based on operator actions.

While the nearest neighbor strategy can help match incoming alarms with existing tickets, there is no easy way to identify that a new ticket needs to be spawned. We extend the classification module to include a mechanism for recommending when an incoming alarm should spawn a new ticket and include this recommendation in CueT's ordered list of recommendations. New tickets are interactively spawned and added to the working set when a human operator judges that an incoming alarm is unrelated to any existing ticket (i.e., part of a new problem). Additionally, old tickets are dynamically discarded from the working set over time, simulating the effect of problems represented by those tickets as being resolved (and reducing interference among existing tickets in the working set).

Recommending Existing Tickets

CueT measures similarity between the incoming alarm and a ticket in the working set of tickets by computing the average distance between the incoming alarm and each alarm in the ticket. CueT then orders the tickets by their average distance to the incoming alarm and uses this ordering of tickets as its triage recommendations.

Distance between alarms is measured using 17 individual distance metrics, each of which represents alarm similarity along a feature attribute. These 17 distance metrics were found to be relevant and important for triage during our initial observations of Tier 1 operators. The operators typically inspect the following alarm attributes:

- *Device Name* (e.g., ab1-cd2-ef3-gh4)
- *Device Type* (e.g., Router, Switch)
- *Element Type* (device part needing attention, e.g., Port)
- *Name* (includes Device Name and additional information such as the Element that needs attention, e.g., Port-ab1-cd2-ef3-gh4)
- *Severity* (an integer ranging from 1 to 5 representing highest to lowest priority, respectively)
- *Event Name* (e.g., Fan-3-Failed, High Utilization)

From these attributes, CueT computes 17 string-based distance metrics described below. Our simulations in the following section show that these string-based distance metrics effectively capture operators' practice of visually comparing the attribute values of alarms. Further, because large organizations often follow standard device naming conventions [16] (e.g., the "ab1" in ab1-cd2-ef3-gh4 typically indicates the location of the device), some of our string-based metrics implicitly encode topological information about the underlying network structure (e.g., device ab1* is likely to be near device ab2*). If organizations do not include topological information in device names, such information is typically available in network configuration data, from where it can be easily extracted and used to compute the following metrics.

For alarm attributes *Device Name*, *Name*, and *Event Name*, as well as the four standard component parts of the *Device Name* (e.g., ab1-cd2-ef3-gh4 is divided into ab1, cd2, ef3, and gh4), CueT computes two string-based distance metrics (amounting to fourteen metrics in total): the edit distance and the longest common substring (LCS) converted to a distance according to:

$$d_{i,j} = \text{maxlength}\,(i, j) - s_{i,j}$$

where $s_{i,j}$ is the length of the longest common substring between strings i and j. We include both edit distance and LCS because they have complementary strengths. For example, LCS is a good measure for strings that encode location. Devices "ab1*" and "ac1*" are likely in different locations. For these, LCS distance (which is 2) better captures that these are different than edit distance (which is 1). As described below, our method of learning the combination of these individual metrics will reduce the effect of any irrelevant metric (edit distance in this case).

For alarm attributes *Device Type*, *Element Type*, and *Severity*, CueT computes one string matching distance metric each (amounting to three metrics in total). This distance metric returns 0 if the attribute values are the same or 1 if they are different.

We combine these 17 distance metrics using Mahalanobis distance, which parameterizes distance between any alarms $\boldsymbol{u}$ and $\boldsymbol{v}$, represented as d dimensional feature vectors, by a $d \times d$ positive definite covariance matrix A:

$$Distance\,(\boldsymbol{u}, \boldsymbol{v}) = (\boldsymbol{u} - \boldsymbol{v})^{\mathrm{T}} A\,(\boldsymbol{u} - \boldsymbol{v})$$

This function effectively weights the 17 distances by the matrix A, which encodes their relative importance for alarm classification and the correlations between them.

We learn the parameters of the matrix A from operators using an online metric learning algorithm [11], originally derived for nearest neighbor classification in static environments. We extend the procedure to dynamic scenarios where both the number and type of classes are varying. In particular, given a stream of alarms, each labeled with the ticket that it was triaged into, we incrementally update the matrix A by encoding the labels as constraints indicating that the incoming alarm and each alarm in the target ticket should be near each other. When an alarm spawns a new ticket, no update is made to the matrix A (however, this does change the working set of tickets). To learn the parameters of A, we initialize it to the identity matrix (and set the regularization parameter η to .001) and then update the parameters as we observe triage actions. We continue this process for N alarms, where N is determined empirically from our simulation experiments described below, and then fix the distance function. The final covariance matrix A_N is used in making recommendations for the remaining data.

Intuitively, the parameters learned for the matrix A reflect the importance and correlations among the individual distance metrics, to best explain the human operator's actions. The advantage of learning the matrix A from data is that it does not require expert tuning, which can be difficult to obtain and does not evolve with the network [9, 17].

Recommending Starting a New Ticket

CueT maintains a threshold distance for starting a new ticket based on information about when operators spawn new tickets. When an operator spawns a new ticket for an incoming alarm, the distance between this alarm and the nearest ticket in the working set is stored. We experimented with several strategies for computing the threshold distance from these stored distances. They include taking the minimum and average over various window sizes of the most recently stored distances (including a window of all the distances). We found that taking the minimum within a window of the five most recent stored distances performs best. That a small window size performs better than larger sizes likely stems from the fact that thresholds need to reflect the dynamically changing distribution of tickets in the metric space.

For each incoming alarm, CueT computes the latest threshold distance using the strategy above and inserts a "start new ticket" recommendation into its ordered list of recommendations according to this distance.

Spawning New Tickets and Discarding Old Tickets

When an operator determines that an incoming alarm is part of a new problem, a new ticket is created and added to the working set. CueT also automatically discards old tickets, simulating the resolution of problems. We use a windowing mechanism to discard old tickets. In particular, we fix the window size to N, which is the number of alarms used for learning our covariance matrix. Any time the number of unique alarms in the working set of tickets exceeds N we remove the oldest ticket in the set. Spawning new tickets and discarding old ones means that the working set of tickets used for machine learning based recommendations is continually evolving as an operator interacts with CueT.

Simulation Experiments

In this section, we present the results of experiments that simulate human interaction with CueT's interactive machine learning component. For our experiments, we obtained alarm triage data from a network operations center at a large organization that monitors a network with approximately 15,000 devices. This data was labeled by Tier 1 operators through their manual triage process. To evaluate CueT's interactive machine learning component over a long period, we use data spread across several months. In particular, we use data from the first day of each month between January and August 2010 (inclusive) except for May and July when the network had recording problems. This data set contains 338,218 alarms of which 8,719 are unique and are mapped to 1,281 unique tickets.

To simulate human interaction and compute the accuracy of CueT's learning, we processed alarms in the data in the order in which they occurred. For each alarm, we first compute an ordered list of recommendations that we use to measure accuracy. Then, we obtain the actual label for the alarm and either add the alarm to an existing ticket or create a new ticket. If we add the incoming alarm to an existing ticket and we have observed fewer than N alarms, we update A_N as described previously. If we start a new ticket, we update the threshold distance for starting new tickets and update the working set of tickets. Finally, if we determine that the working set has exceeded the window size of N alarms, we discard the oldest ticket in the set.

We measure recommendation accuracy for each incoming alarm by comparing the recommendations to the ground truth (all of the alarm triage data observed before reaching the current incoming alarm, without discarding any tickets). There are two types of correct recommendations:

- *Correct New Ticket*. CueT recommends a new ticket be created and the alarm's label shows that a human operator actually created a new ticket for that alarm.
- *Correct Existing Ticket*. CueT recommends an existing ticket and that ticket is the alarm's actual label.

If neither of these is true, we record the recommendation as incorrect. Note that because the nature of the problem we are dealing with requires that we operate in a moving window, some of our errors may be the result of discarding tickets (e.g., recommending a new ticket when the correct ticket is in the ground truth but no longer exists in the working set of tickets).

Because multiple tickets may be the same distance away from an incoming alarm, we compute recommendation accuracy as whether or not the alarm's actual label appeared within the set of ticket recommendations a given distance away from the alarm or closer. For example, if CueT predicts two different ticket recommendations as being equally closest to the incoming alarm ("Top 1 distance" away) and if the correct label is one of the two tickets then we consider this a correct recommendation at the Top 1 distance. Similarly, a recommendation is correct at the Top 2 distance if the correct label is within the set of tickets recommended at the Top 2 distance away from the incoming alarm or less (e.g., at the Top 1 distance). We

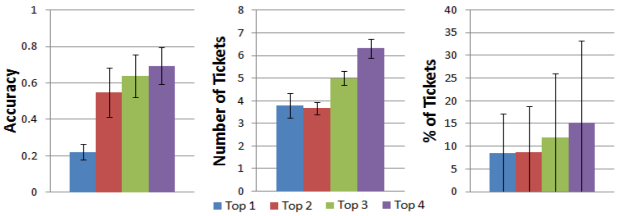


Figure 3. CueT's accuracy (left), number (middle), and % of tickets presented (right) within Top 1, 2, 3, 4 distances from each incoming alarm, averaged over all simulation trials.

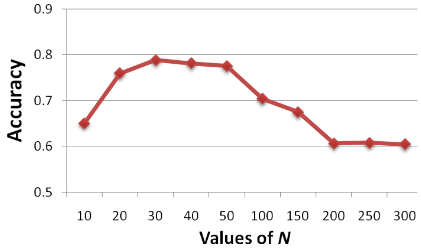


Figure 4. CueT's accuracy at the Top 3 distances for various window sizes *N*. *N*=30 achieves peak performance.

experimented with accuracy within the Top 1, 2, 3 and 4 distances from the incoming alarm.

We ran ten simulations over our data, varying the number of alarms N used in both learning the distance function parameters A_N as well as in the window size for discarding old tickets. Figure 3 (left) illustrates CueT's accuracy within the set of tickets recommended at the Top 1, 2, 3, and 4 distances from incoming alarms averaged over all the simulation trials. Figure 3 (middle and right) shows the average number and percentage of tickets (out of N) presented at each of the Top 1 to 4 distances. From these results it appears that presenting tickets in the Top 3 distances achieves a good balance between relatively high accuracy and a small number of tickets being presented. Therefore, for our user study described later, we fix CueT to recommend tickets for an incoming alarm within the Top 3 distances from that alarm.

CueT's accuracy at the Top 3 distances over the various values of N that we experimented with (10, 20, 30, 40, 50, 100, 150, 200, 250, and 300) appears to peak at an N value of 30 alarms (Figure **4**). Therefore, for our user study, we set N=30 in our interactive machine learning engine.

CueT Interface

CueT's interface (Figure 1) consists of two main views: the *Alarm View* on the right displays alarms as they stream in from the network and the *Ticket View* on the left displays CueT's ticket recommendations.

When operators click on an alarm in the Alarm View, CueT displays its recommendations for that alarm in the Ticket View and illustrates its confidence in those recommendations with the *Ticket Distance Overview*. Operators can then inspect the recommendations and visualizations to determine how to triage the alarm (that is, either add it to an existing ticket or start a new one). Operators triage the alarm by dragging and dropping it onto the appropriate ticket in the Ticket View. CueT's interface also contains a *Search View* (bottom left in Figure 1) through which operators can search for existing tickets by entering a search string as they do with their current system for triaging alarms. Operators can add alarms to tickets appearing in the Search View just as in the Ticket View.

Alarms and Alarm View

Our initial observations of operators helped us determine the importance of certain alarm attributes. *Severity* and *Notification Time*, most viewed by operators when determining which alarm to triage next, are displayed prominently. As shown in Figure 1 (right), Severity (on a scale of 1-5) is encoded on the left of an alarm by color: red (most severe), orange-red, orange, yellow, and white (least severe). The Notification Time attribute is emphasized in bold and displayed at the top right-hand side of the alarm.

Operators often work under a service level agreement (SLA) that defines the time limit within which an alarm must be triaged. CueT highlights when an alarm has passed this limit by overlaying a pattern on the Severity display as in the alarms at the top of the Alarm View in Figure 1.

Operators also sometimes use the *Count* attribute of an alarm to determine which alarm to deal with next. The Count attribute represents the number of duplicate alarms observed and many duplicates sometimes signal a severe problem. CueT represents Count by the length of the horizontal bar (color coded by Severity of the alarm) at the bottom of the alarm. In addition, if duplicates are observed, the time when the last duplicate was observed is displayed in gray at the bottom right of the alarm.

Alarm *Name* is also emphasized in the alarm in bold as this attribute is used most often when comparing alarms to existing tickets and deciding how to triage. The rest of the alarm information, including *Device Type*, *Event Name*, and a description of the event is displayed less saliently in the alarm in gray. This layout allows operators to visually scan and compare alarms by Severity, Time, and Count while still being able to digest the rest of the alarm information in a compact representation.

Alarms are displayed in the Alarm View as they stream in from the network. Since operators often miss important alarms that appear off of the screen, the Alarm View includes an Alarm Overview (far right in Figure 1) that provides awareness of all alarms still requiring triage even if they are off the screen. This overview displays one rectangle per alarm, color coded by that alarm's Severity and possibly the pattern if that alarm has passed its SLA. The heights of the rectangles automatically adjust so as to always display all of the alarms currently available for triage. The rectangles are presented in the order that alarms are displayed in the Alarm View, where alarms can be sorted by any attribute of the alarm. This allows the overview to act as a scroll bar for easy alarm navigation.

Tickets, Ticket View, and Ticket Distance Overview

Tickets are a collection of related alarms (Figure 1 left). Each ticket has a parent alarm, which is manually determined by a human operator and typically represents either the most severe or the first alarm in the ticket. The label at the top of the ticket is color coded by Severity of the ticket's parent alarm. The ticket label also includes the ticket's unique ID, automatically generated by the system at the time of creation, and two numbers in parenthesis; the number of unique alarms within that ticket followed by the total number of duplicates across those alarms. Unique alarms and duplicates are dually represented in the ticket as a series of horizontal bars (similar to the Count display in the alarm representation), one for each unique alarm, and again color coded by Severity of the alarm and reflecting the number of duplicates for that alarm by its length.

Immediately below the ticket label is information about the ticket's parent alarm along with the ticket description. Below the parent alarm is the best matching alarm within the ticket to the incoming alarm (next to the star icons in Figure 1). This serves as an explanation for why CueT is recommending that an operator triage an alarm into a given ticket. Thus, ticket representations displayed in the Ticket View are tailored for each alarm. Operators can also click on a ticket to display all the alarms currently grouped within the ticket. New ticket recommendations are displayed as empty ticket stubs with a gray label and text within the ticket displaying "Start New Ticket."

Each time the operator clicks on an alarm to triage in the Alarm View, CueT generates its ticket recommendations for the selected alarm and displays them in the Ticket View. They are shown in order of increasing distance from the alarm as per the distance function described previously. By default, and as determined by our simulation results, the Ticket View initially displays only the tickets within the Top 3 distances from the alarm being triaged as this helps to balance operator load and the probability of these tickets containing the correct recommendation. CueT allows operators to reveal more tickets to inspect using the Ticket Distance Overview visualization.

The Ticket Distance Overview (far left in Figure 1) is designed to provide operators with an estimate of its confidence in its recommendations so that operators can determine the necessity of inspecting more tickets. Each bubble in the Ticket Distance Overview corresponds to a ticket. The vertical position of the bubbles relative to the top of the overview reflects the distance between the alarm being triaged and each ticket within the working set. That is, the closer the bubble is to the top, the better a match the corresponding ticket is for the alarm with respect to our distance function. Also vertical positions of the bubbles correspond to the ordered list of recommendations in the Ticket View. Vertical distances are normalized so as to fit all of the existing tickets within the display. Thus, the positions of the bubbles display relative distances between tickets rather than absolute distances. Horizontal positioning is only used to minimize overlap of bubbles that are of equal or near equal distance to the incoming alarm.

Bubbles that are positioned near each other are comparable in terms of their similarity to the alarm currently being triaged. In this case, the overview should encourage operators to inspect all of the comparable tickets. To enable operators to inspect more tickets when necessary, CueT provides a horizontal green bar allowing them to set the distance threshold for the tickets to be displayed in the Ticket View. It divides the bubbles into a visible region (above the bar and corresponding to visible tickets in the Ticket View) and the invisible region (faded bubbles below the bar and corresponding to tickets not currently visible in the Ticket View). Operators can drag this bar vertically to reveal and inspect other tickets within the Ticket View.

USER STUDY

We conducted a user study to examine the effectiveness of CueT for alarm triage as compared to the traditional method of manually ticketing alarms. For the Traditional condition, we replicated the commercial system used by our network operators (Figure 2). As in the commercial system, participants could keyword search the tabular view of existing tickets to add incoming alarms or create new tickets. Adding alarms to existing tickets or creating new tickets is achieved by right clicking on a row in the table of alarms and selecting the corresponding action from a popup menu. The commercial system has separate windows for displaying alarms and searchable tickets, both using a tabular format. To avoid the overhead of switching between windows (a problem we observed during our initial observations), our version of the Traditional interface combines both of these views in one window as in CueT. This combination provides a fairer evaluation of the current practice, as the two-window issue is easy to fix.

Data, Study Design, and Equipment

For our study we used part of the data that we used for our simulation experiments (January 1, 2010 data). To compare two interfaces, we created two data sets from this data. To ensure that each set includes comparable numbers, distributions and types of alarms and tickets, we extracted alternating unique alarms. We also simulated the correct assignment of alarms to tickets for alarms not being shown as CueT relies on a dynamically changing working set of tickets. Therefore, *Data Set 1* included odd alarms (while we simulated the correct assignment for even alarms) and *Data Set 2* included even alarms (where we simulated the correct assignment for odd alarms). Each data set contained 80 unique alarms to be triaged by our participants as our pilot study showed that was a manageable number of alarms to triage in about 20 minutes. For demonstration and practice in each condition, we also created two additional data sets (*Demo Sets 1* and *2*) from the April 1, 2010 data using this same approach.

We conducted a within-subjects study, with each participant performing alarm triage using both CueT and the Traditional method. We compared *CueT* to the *Traditional* method in terms of accuracy, speed, and user preference. To avoid a learning effect, we counterbalanced the presentation order of the two interfaces.

We ran participants individually or in pairs depending on their schedule (two pairs). Each participant worked on a 2.7 GHz dual-core Windows 7 laptop with 4 GB RAM. We attached a 20.1'' Samsung monitor at a resolution of 1200x1600 (i.e., in a portrait orientation) to each laptop, as well as a mouse and keyboard. We turned the laptop away from participants so they could only look at the attached monitor and use the attached mouse and keyboard. When we ran pairs, we faced their desks away from each other to minimize disturbance.

Procedure and Participants

Before each condition, the experimenter demonstrated each interface using *Demo Set 1*. Then the participants were allowed to practice triaging alarms using *Demo Set 2* until they were comfortable with the interface and had practiced triaging several alarms, for a maximum of 5 minutes.

In each condition, participants were asked to triage all 80 alarms presented as accurately and quickly as possible and in the order that they normally would (i.e., they could triage the alarms in any order, but are encouraged to triage high severity alarms and alarms that have passed their SLA first). All interface actions were time-stamped and logged.

After each condition, participants were given a short questionnaire about the interface they just used. The questionnaire included 7-point Likert scale questions asking for the participants' level of agreement with statements about the interface (e.g., "Overall, I am satisfied with this system.") and specific questions about the CueT interface if they had just used that interface (e.g., "The ticket recommendations were useful."). It also asked participants to list three things that they liked and three that they would like improved about the interface. At the end of the session, a final questionnaire asked participants to select which interface they preferred and explain why.

The experiment lasted about 90 minutes and participants were given a gratuity of $20 worth of dining coupons. To encourage participants to triage alarms quickly and accurately, we also offered a prize of an additional $20 worth of dining coupons for the person who performed the best in terms speed and accuracy in each condition.

We recruited eleven people (two female, ages 28 to 44) plus one male pilot from the network operations team. Our participants were not currently working as Tier 1 operators, though six of them were self-proclaimed experts at the alarm triage process and four said they were proficient. One said he was a beginner. We could not recruit active Tier 1 operators because of their tight work schedule and because many work outside of the country.

Results

Performance: Accuracy and Speed

We analyze our logged data in terms of accuracy and speed. *Accuracy* is computed as the percentage of alarms correctly triaged out of the total presented. Correctness of participant labels is measured against the ground truth labels.

We compute two measures for speed: *Time on Screen* and *Time to Ticket*. The former is the time between when an alarm appeared on screen and when the participant completed the triage operation for that alarm. Along with *Accuracy*, it is a key measure of triage performance and is used to formulate service level agreements (SLAs) that the monitoring team offers. For instance, a possible guarantee may be that for 95% of alarms *Time on Screen* would be under 5 minutes. Note that both *CueT* and *Traditional* present multiple alarms on screen simultaneously and operators need not triage alarms in the order in which they appear on screen. Therefore, *Time on Screen* is affected by the order in which an operator decides to triage an alarm. Thus, for a detailed view of triage behavior, we also study *Time to Ticket*, which is time between successive triage actions regardless of order.

For *Accuracy*, *Time on Screen*, and *Time to Ticket*, we perform paired-samples *t* tests. We report means and standard deviations throughout.

Our analyses showed that participants were able to triage alarms faster with *CueT* than with the *Traditional* interface while maintaining the same level of accuracy. Participants were significantly faster with *CueT* than *Traditional* in terms of *Time on Screen* (*M*=107.7s, *SD*=127.7s vs. *M*=277.8s, *SD*=168.5s, *t*(10)=4.43, *p*=.001) and in terms of *Time to Ticket* (*M*=10.1s, *SD*=2.69s vs. *M*=12.9s, *SD*=3.79s, *t*(10)=3.26, *p*=.009). There was no significant difference in terms of accuracy between the *CueT* and *Traditional* conditions (*M*=71.8%, *SD*=17% vs. *M*=76.4%, *SD*=8%).

Our data included a type of alarm that required special handling. Operators are usually instructed to always create a new ticket for each such alarm, regardless of similarity to other alarms. Only a few participants asked us how to triage such alarms, to which we responded that they should triage as normal. The logged data shows that such alarms were handled unevenly by participants. Some rapidly created new tickets without inspecting recommendations (in CueT) or searching related tickets (in the CueT or Traditional condition), while others triaged based on similarity. These alarms reduce CueT's accuracy because its model does not handle exceptional cases. Despite that our results show that CueT's accuracy is no worse and its speed is much better.

We re-did our analysis after removing 39 of these exceptional alarms from the data to evaluate CueT's performance in the absence of special cases. Our corrected analyses of variance show that participants were still faster with *CueT* but also more accurate than the *Traditional* condition (Figure 5). The accuracy with *CueT* versus the

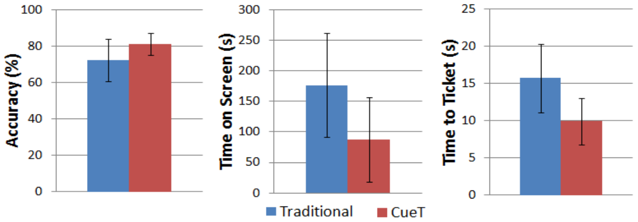


Figure 5. Accuracy (left), Time on Screen (middle), and Time to Ticket (right) comparisons. All differences are significant. Error bars represent standard error.

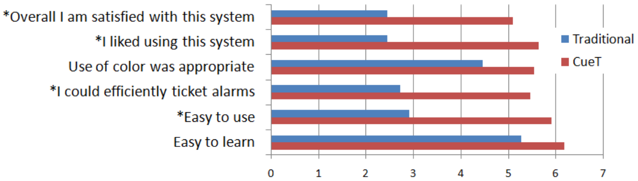


Figure 6. 7-point Likert scale questionnaire results. Stars indicate the question was significantly different.

Traditional condition was 81.3% (*SD*=6%) vs. 72.4% (*SD*=12%), (*t*(10)=2.29, *p*=.045). The participants were significantly faster with *CueT* at triaging in terms of both *Time on Screen* (*M*=86.9s, *SD*=69.2s vs. *M*=176.2s, *SD*=85.2s, *t*(10)=4.63, *p*=.001) and *Time to Ticket* (*M*=9.9s, *SD*=3.1s vs. *M*=15.7s, *SD*=4.6s, *t*(10)=6.52, *p*<.001).

Subjective Preference

We analyze our post-condition questionnaires using Friedman Chi-Square tests. *CueT* was favored significantly more than *Traditional* in terms of overall satisfaction (χ^2(1,*N*=11)=9.0, *p*=.003), how much participants liked using the system (χ^2(1,*N*=11)=11.0, *p*=.001), whether they felt that they could efficiently ticket alarms with the system (χ^2(1,*N*=11)=6.4, *p*=.011), and whether they felt the system was easy to use (χ^2(1,*N*=11)=11.0, *p*=.001) (Figure 6).

Regarding *CueT*-specific features, participants tended to agree with the statements "The ticket recommendations were useful" (5.81 avg.) and "The Distance Overview was useful" (5.36 avg.). In addition, all of our participants chose *CueT* as their preferred system for network alarm triage.

DISCUSSION AND FUTURE WORK

Our results show that real network operators can triage alarms significantly faster with CueT than with their traditional method. When considering general alarms as well as exceptional cases, CueT reduces the *Time on Screen* of alarms by 61.2% on average. When excluding exceptional cases the savings on this measure are 50.7%. Savings are lower without exceptional cases because such cases require an action that can be performed quickly and without deliberation (e.g., rapidly creating new tickets without looking for similar tickets). *Time on Screen* is affected by the order in which an operator triages alarms. Thus, CueT's Alarm Overview, designed to facilitate awareness of alarms remaining in the queue of alarms to be triaged, likely contributed to the savings on this measure. In listing the things they liked about CueT, one participant commented that "Having the alert change colors as it approached SLA helps with prioritization."

For *Time to Ticket*, CueT enables a savings of 21.6% when considering general and exceptional alarms, and a savings of 36.8% when considering only the general alarms that CueT is designed for. Considering that participants performed only 4.6 manual searches on average throughout their session with CueT (compared to 84.6 searches on average with the traditional method) and only one participant ever ticketed an alarm via dragging and dropping the alarm onto CueT's Search View, participants relied on CueT's recommendations to triage. Therefore, as *Time to Ticket* measures the time between ticket actions, this savings in time can be attributed to CueT's ability to provide operators with suggestions about how to ticket each alarm rather than having to manually search for tickets. To put this in perspective, assuming 10K alarms per day and a time savings of 5.8s per alarm (36.8%), CueT's estimated cumulative time savings using this measure amounts to about 20 operator days per month.

As our observations of operators revealed, exceptional cases are a reality in network operations. CueT currently cannot automatically exclude exceptional cases from its dynamically changing model. Remarkably, when considering general alarms along with exceptional cases, no significant decrease in overall accuracy is observed. This suggests that CueT does not adversely affect operator ability to deal with exceptional cases. Furthermore, although including exceptional cases in CueT's dynamic model may cause interference in recommendation accuracy for general alarms, CueT still performed significantly better than the traditional method for the general case, by 9%. This result points at the robustness of using triage recommendations from human-guided interactive machine learning based models. As interference can negatively affect CueT's recommendation accuracy, it is fair to regard CueT's performance results from our evaluation as a lower bound on its potential for improving alarm triage. In fact, it may be possible for network administrators to reduce some of this noise by creating temporary rules to automatically remove exceptional alarms. Despite the potential for interference, nine of our eleven participants commented that CueT's recommendation ability was one of the things they most liked about the system (e.g., "Recommendations done by the system for appropriate tickets was very useful.").

In terms of their subjective preference, all participants preferred CueT over the traditional method. The only questions for which there was no significant difference between CueT and the traditional method were whether the use of color was appropriate and whether the system was easy to learn. As with CueT, the traditional method also makes use of color coding to indicate severity of alarms. CueT however takes an additional step in overlaying information about the operator's SLA on the colored area of an alarm. In terms of being easy to learn, our participants were all already familiar with the traditional system of manually triaging alarms. Despite this, they were still able to learn and start using CueT in a matter of minutes.

We believe CueT's tight integration between interactive machine learning and visualization is key to its success. As with other distance-based recommendation systems, multiple tickets in CueT can be equally distant from an alarm. Presenting recommendations as a traditional list would therefore require arbitrarily ordering tickets and would likely mislead operators. To ensure high accuracy, CueT's Ticket Distance Overview visualization was specifically designed to show estimates of recommendation quality and encourage operators to inspect comparable tickets. Further, the coupling between machine learning and visualization makes it easy for operators to provide feedback to the system and keep the model up-to-date. However, additional studies that compare CueT's machine learning with and without visualization would shed more light on the value of this integration. We suggest a longitudinal investigation of operator confidence in the recommendations and the effects on operator vigilance in carefully inspecting those recommendations with and without visualization.

CONCLUSION

We present CueT, a system that combines novel visualizations and interactive machine learning to deal with a highly dynamic environment where the groups of interest are not known a-priori and evolve constantly. We implement CueT in the context of triaging network alarms to assist network operators in the complex task of alarm triage. Our user study indicates that CueT increases operator accuracy as well as speed compared to the current approach. All of our study participants preferred CueT over the traditional method, a sentiment that is reflected in one participant's comment: "the new system compared to the old is hands down better." While CueT is designed for triaging alarms, we believe that the lessons learned from our work readily extend to other scenarios where humans need to organize continuous streams of data.

ACKNOWLEDGEMENTS

We thank the staff at the network operations center, especially Spencer Watkins, for explaining their current practices to us and letting us observe their work. We also thank our study participants for their time and feedback.

REFERENCES

1. Appleby, K., Goldszmidt, G., and Steinder, M. Layered Event Correlation Engine for Multi-Domain Server Farms. *Proc. INM 2001*, IEEE (2001), 329-344.
2. Basu, S., Fisher, D., Drucker, S.M., and Lu, H. Assisting Users with Clustering Tasks by Combining Metric Learning and Classification. *Proc. AAAI 2010.*
3. Brugnosi, S., Bruno, G., Manione, R., Montariolo, E., Paschetta, E., and Sisto, L. An Expert System for Real Time Fault Diagnosis of the Italian Telecommunications Network. *Proc. INM 1993*, IEEE (1993), 617-628.
4. desJardins, M., MacGlashan, J., and Ferraioli, J. Interactive Visual Clustering. *Proc. IUI 2007*, ACM Press (2007), 361-364.
5. EMC Ionix, http://www.emc.com/products/family/ionix-family.htm
6. Fails, J.A. and Olsen, Jr., D.R. Interactive Machine Learning. *Proc. IUI 2003*, ACM Press (2003), 39-45.
7. Fisher, D., Maltz, D.A., Greenberg, A., Wang, X., Warncke, H., Robertson, G., and Czerwinski, M. Using Visualization to Support Network and Application Management in a Data Center. *Proc. INM 2008*, IEEE (2008), 1-6.
8. Fogarty, J., Tan, D., Kapoor, A., and Winder, S. CueFlik: Interactive Concept Learning in Image Search. *Proc. CHI 2008*, ACM Press (2008), 29-38.
9. Gardner, R.D. and Harle, D.A. Methods and Systems for Alarm Correlation. *Proc. GLOBECOM 1*, IEEE (1996), 136-140.
10. HP OpenView, http://openview.hp.com
11. Jain, P., Kulis, B., Dhillon, I.S., and Grauman, K. Online Metric Learning and Fast Similarity Search. *Proc. NIPS 2008*, (2008), 761-768.
12. Jakobson, G. and Weissman, M.D. Alarm Correlation: Correlating multiple network alarms improves telecommunications network surveillance and fault management. *IEEE Network 7*, 6 (1993), 52-59.
13. Klementtinen, M., Mannila, H., and Toivonen, H. Rule Discovery in Telecommunication Alarm Data. *J. Network and Systems Management 7*, 4 (1999), 395-423.
14. Lakkaraju, K, Yurcik, W., and Lee, A.J. NVisionIP: Network Visualizations of System State for Security Situational Awareness. *Proc. VizSEC/DMSEC 2004*, ACM Press (2004), 65-72.
15. Liu, G., Mok, A.K., and Yang, E.J. Composite Events for Network Event Correlation. *Proc. INM 1999*, IEEE (1999), 247-260.
16. Spring, N., Mahajan, R., Wetherall, D., and Anderson, T. Measuring ISP Topologies with Rocketfuel. *Proc. SIGCOMM 2002*, ACM Press (2002), 133-145.
17. Steinder, M. and Sethi, A.S. A Survey of Fault Localization Techniques in Computer Networks. *Science of Computer Programming 53*, (2004), 165-194.
18. Yemini, S., Kliger, S., Mozes, E., Yemini, Y., and Ohsie, D. High Speed and Robust Event Correlation. *IEEE Communications Magazine 34,* 5 (1996), 82-90.