Human-Guided Machine Learning for Fast and Accurate Network Alarm Triage

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

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

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

[*]Microsoft Corporation
Redmond, WA
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 rule-based tools cannot easily evolve with the network. We present CueT, a system that uses interactive machine learning to constantly 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. Our evaluations with real operators and data from a large network show that CueT significantly improves the speed and accuracy of alarm triage.

1 Introduction

Network alarm triage refers to grouping and prioritizing a stream of low-level device health information (e.g., high link utilization and fan failure alarms) to help operators find and fix problems in computer networks. It is critical that triage is fast and accurate so operators are not misled and problems can be identified and resolved quickly.

Many automated systems have been developed for the alarm triage problem [Gardner and Harle, 1996; Steinder and Sethi, 2004], mostly taking the form of expert defined rules or models. 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 over 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) to triage the remaining thousands of alarms per day that can be missed by automation.

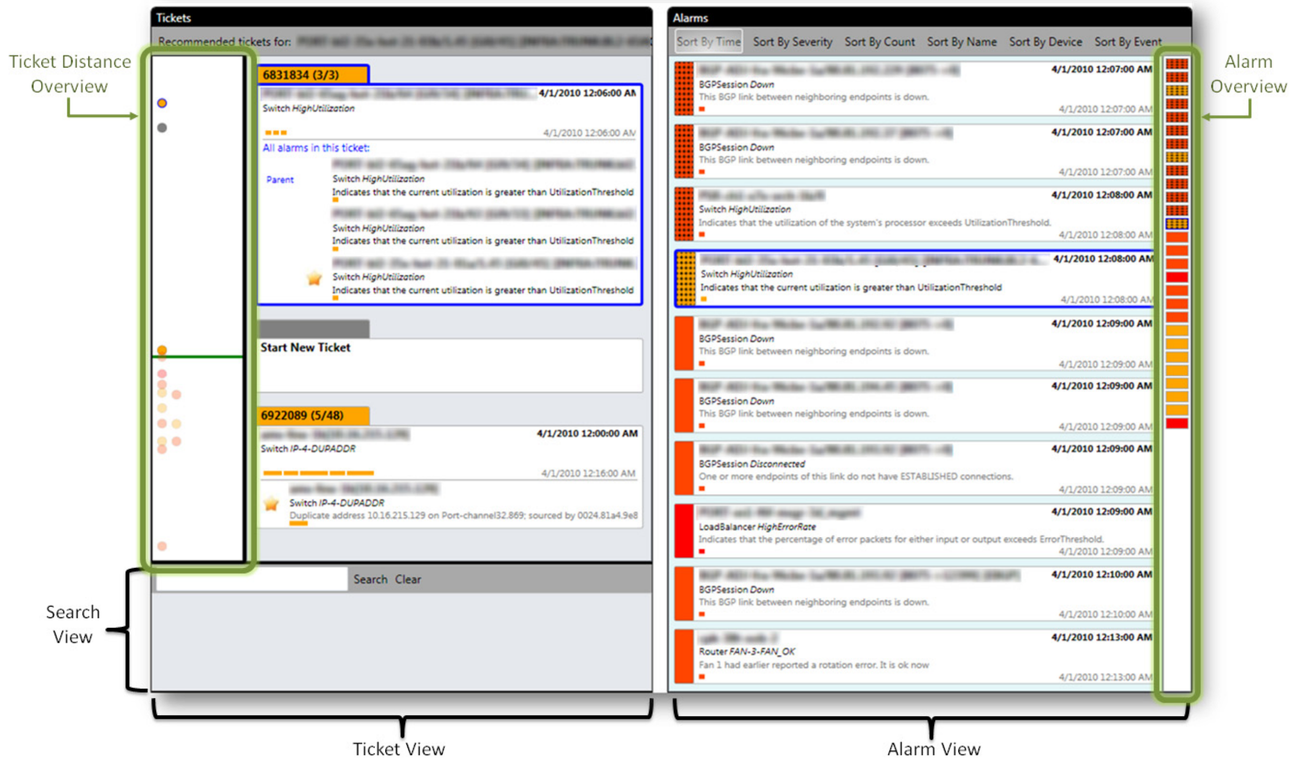
We explore a fundamentally different approach for alarm triage, using interactive machine learning to constantly learn from operator triage actions and in turn assist them by providing recommendations on how to group incoming alarms based on the learned model at that instant. Our approach is designed for highly dynamic environments where the groups are not known *a priori* and evolve constantly. It can also be applied to other scenarios where people need help organizing a continuous data stream (e.g., RSS feeds, email management, social network updates).

We implement our approach in the context of the alarm triage problem with a system we call CueT (Figure 1). Our simulation experiments with real data from a large, global network with approximately 15,000 devices show the potential of our constantly-updating machine-learning-based approach for improving the alarm triage process. However, as with all automation, machine-learning-based systems are never perfectly accurate. Therefore, in real-world applications of our approach, it is critical that operators carefully inspect and compare recommendations before deciding how to finally proceed. To this end, CueT includes a novel visualization that conveys its recommendation confidence to further help operators decide how to triage. Our evaluation with human operators shows that CueT's combination of interactive machine learning with novel visualizations significantly improves the accuracy and reduces the time required for alarm triage.

2 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. For example, Visual-I [Fisher *et al.*, 2008] uses visual grouping and scatterplots to highlight correlations between multiple devices and problems. 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



Recommended tickets for:

6831834 (3/3)

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

All alarms in this ticket:

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

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

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

Start New Ticket

6922089 (5/48)

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

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

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

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

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

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

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

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

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

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

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

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.

potential impact on the industry [Gardner and Harle, 1996; Steinder and Sethi, 2004]. Most solutions have taken the form of expert defined rules or models (e.g., [Gardner and Harle, 1996; Jakobson and Weissman, 1993]). Such approaches require manual configuration of rules which can be difficult to obtain, are not robust to new situations, and require frequent maintenance [Gardner and Harle, 1996; Steinder and Sethi, 2004]. In contrast, our 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., [Klementtinen *et al.*, 1999; Steinder and Sethi, 2004]). However, most of these approaches require extensive training periods and must be retrained whenever the network topology changes [Steinder and Sethi, 2004]. 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 [Gardner and Harle, 1996], a key reason why alarm correlation is still an open area of research. While today's commercial systems employ some of these previous techniques [EMC Ionix [1]; Yemini *et al.*, 1996], most companies still require teams of Tier 1 operators to manually triage the thousands of remaining alarms unhandled by existing systems.

Our work is closely aligned with evolving research on interactive machine learning, asserting the importance of human involvement and proposing interactive systems for various applications (e.g., image segmentation [Fails and Olsen, 2003], document grouping [Basu *et al.*, 2010], and image search [Fogarty *et al.*, 2008]). While previous work has focused on pool-based static environments where the categories of interest and the pool of unlabeled data are known *a priori*, our work involves a more challenging scenario where data is streaming in and out and the set of labels is constantly changing over time.

3 CueT

CueT consists of two interacting components: (1) an interface to assist operators in inspecting triage recommendations and feeding operator actions back into the learning system and (2) a stream-based interactive machine learning engine for making triage recommendations. 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."

[1] http://www.emc.com/products/family/ionix-family.htm

3.1 CueT Interface

CueT's interface (Figure 1) consists of two main views: the Alarm View on the right and the Ticket View on the left.

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. Each time an 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 along with a visualization illustrating its confidence in those recommendations with the Ticket Distance Overview (far left in Figure 1).

Tickets are a collection of related alarms. 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. Immediately below the ticket label (at the top of each ticket in the Ticket View) 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 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 displaying "Start New Ticket."

Operators can inspect CueT's ticket recommendations and the Ticket Distance Overview visualization to determine how to triage an incoming alarm (i.e., either add it to an existing ticket or start a new one). Each bubble in the Ticket Distance Overview corresponds to a ticket and its vertical position relative to the top of the overview reflects the similarity 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 currently being triaged. Bubbles that are positioned near each other are comparable in terms of their similarity to the alarm. In this case, the overview should encourage operators to inspect all comparable tickets before triaging.

The Ticket View initially displays only the tickets closest to the alarm being triaged (where 'closest' is determined by our simulations discussed in the following sections) as this helps to balance operator load and the probability of these tickets containing the correct recommendation. However, CueT allows operators to reveal more tickets to inspect using the Ticket Distance Overview.

Operators triage an 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.

3.2 Stream-Based Interactive Machine Learning

We tackle the challenges due to the highly dynamic environment and ever-evolving set of classes (i.e., the working set of tickets and the alarms within those tickets) by building on nearest neighbor classification. CueT provides triage recommendations for an incoming alarm by ordering the working set of tickets by their similarity to that alarm. Similarity is measured using a distance function that adapts based on operator actions. We extend classification to include a mechanism for recommending when an incoming alarm should spawn a new ticket.

Recommending Existing Tickets

CueT makes recommendations by ordering the working set of tickets by the average distance between the incoming alarm and each alarm in the ticket. Distance between alarms is measured using 17 individual string-based distance metrics, each of which represents similarity along an alarm attribute. Our simulations described later show that these string-based distance metrics effectively capture operators' practice of visually comparing the attribute values of alarms. The attributes operators typically inspect are: *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 information about the Element needing attention, e.g., Port-ab1-cd2-ef3-gh4), *Severity* (from 1 to 5 representing highest to lowest priority, respectively), and *Event Name* (e.g., High Utilization).

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

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

where $s_{i,j}$ is the length of the LCS 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 ***u*** and ***v***, represented as *d* dimensional feature vectors, by a *d*×*d* positive definite covariance matrix A:

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

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 operator actions, extending an online metric learning algorithm [Jain *et al.*, 2008] originally derived for static environments 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 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 changes the working set of tickets). To learn the parameters of A, we initialize it to the identity matrix (setting 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 simulations 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 of 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 [Gardner and Harle, 1996; Steinder and Sethi, 2004].

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 including taking the minimum and average of the most recently stored distances or of all the distances. We found that taking the minimum within the five most recently stored distances performs best.

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.

3.3 Simulation Experiments

For our experiments simulating human interaction with CueT's interactive machine learning component, 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: 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 (as devices typically generate duplicate alarms when problems occur) 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 incoming alarm, without discarding any tickets due to our fixed window size N). Note that the nature of the problem we are dealing with requires that we operate in a moving window. Therefore, 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).

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 tickets 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. We 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 2 (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 2 (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 within the Top 3 distances achieves a good balance between relatively high accuracy and a small number of tickets being presented.

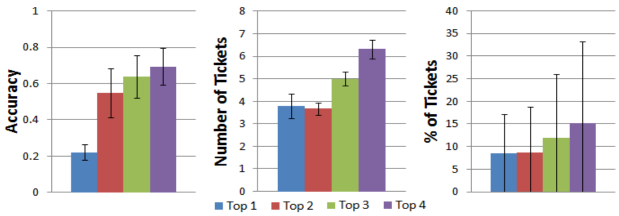


Figure 2. CueT's accuracy (left), number (middle), and % of tickets presented (right) within Top 1, 2, 3, 4 distances from each incoming alarm, averaged over all simulation trials.

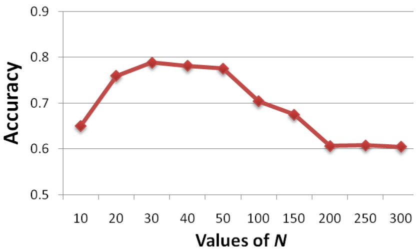


Figure 3. CueT's accuracy at the Top 3 distances for various window sizes N. N=30 achieves peak performance.

Therefore, for our user study described later, we fix CueT to recommend tickets for an incoming alarm (in the TicketView) 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 3). Therefore, for our user study, we set N=30 in our interactive machine learning engine.

4 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. For our study we used part of the data that we used for our simulation experiments (January 1, 2010 data). We ran it as a within-subjects study, with each participant performing alarm triage using both CueT and the Traditional method. To avoid a learning effect, we counterbalanced the presentation order of the two interfaces. We recruited eleven people (two female, ages 28 to 44) plus one male pilot from the network operations team. Each participant worked on an identical dual-core Windows 7 machine attached to a 20.1'' monitor at a resolution of 1200x1600 (i.e., in a portrait orientation). We measured accuracy, speed, and user preference. For more details about the study design and results refer to [Amershi *et al.*, 2011].

4.1 Results

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 triaged 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 will be under 5 minutes. Since operators need not triage alarms in the order in which they appear on screen, Time on Screen is affected by the order in which an operator decides to triage alarms. Therefore, we also study Time to Ticket, which is the time between successive triage actions regardless of order.

For *Accuracy*, *Time on Screen*, and *Time to Ticket*, we perform paired-samples *t* tests and 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 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 (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 these alarms, to which we responded that they should triage as normal. The logged data shows that these 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 this, 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 to evaluate CueT's performance in the absence of special cases. Our corrected analyses show that participants were still faster with *CueT* but also more accurate than with *Traditional* (M=81.3%, SD=6% vs. M=72.4%, SD=12%, *t*(10)=2.29, *p*=.045) (Figure 4). They 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).

We analyze our user preference questionnaires using Friedman Chi-Square tests. CueT was favored significantly more than Traditional in terms of overall satisfaction

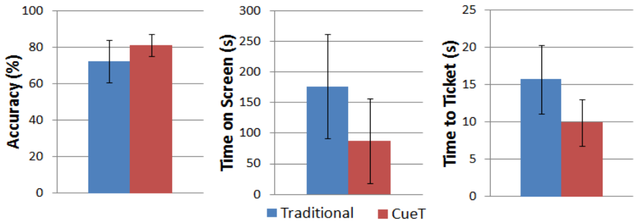


Figure 4. Accuracy (left), Time on Screen (middle), and Time to Ticket (right) comparisons. All differences are significant. Error bars represent standard error.

(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).

Regarding CueT-specific features, participants tended to agree with the statements "The ticket recommendations were useful" (5.81 avg. on a 7-point Likert scale, 7 being the highest level of agreement) and "The Distance Overview was useful" (5.36 avg.). In addition, all of our participants chose CueT as their preferred system for alarm triage.

5 Discussion and Conclusion

Our results show that 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 are 50.7%. Savings are lower without exceptional cases because these require an action that can be performed quickly and without deliberation (e.g., rapidly creating new tickets without looking for similar tickets).

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. To put this in perspective, assuming 10K alarms per day and a time savings of 5.8s per alarm (36.8%), the estimated cumulative time savings using this measure amounts to about 20 operator days per month.

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, CueT still performed significantly better than the traditional method for the general case, by 9%. This 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 as a lower bound on its potential for improving alarm triage.

We believe CueT's tight integration of 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 which would likely mislead operators. To ensure high accuracy, the Ticket Distance Overview was designed to illustrate 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 and keep the model up-to-date.

CueT demonstrates the importance of human involvement to complement and improve automated systems that are never fully accurate due to the complexity of the problem. While CueT is designed for triaging alarms, we believe that the lessons learned readily extend to other scenarios where humans need to organize continuous streams of data.

Acknowledgments

We thank the staff at the network operations center and our study participants for their time and feedback.

References

[Amershi *et al.*, 2011] Saleema Amershi, Bongshin Lee, Ashish Kapoor, Ratul Mahajan, and Blaine Christian. CueT: Human-Guided Fast and Accurate Network Alarm Triage. To appear in *Proc. CHI 2011*.

[Basu *et al.*, 2010] Sumit Basu, Danyel Fisher, Steven M. Drucker, and Hao Lu. Assisting Users with Clustering Tasks by Combining Metric Learning and Classification. *Proc. AAAI 2010*.

[Fails and Olsen, 2003] Jerry Alan Fails and Dan R. Olsen, Jr. Interactive Machine Learning. *Proc. IUI 2003*, pages 39-45.

[Fisher *et al.*, 2008] Danyel Fisher, David A. Maltz, Albert Greenberg, Xiaoyu Wang, Heather Warncke, Geroge Robertson, and Mary Czerwinski. Using Visualization to Support Network and Application Management in a Data Center. *Proc. INM 2008*, pages 1-6.

[Fogarty *et al.*, 2008] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. CueFlik: Interactive Concept Learning in Image Search. *Proc. CHI 2008*, pages 29-38.

[Gardner and Harle, 1996] Robert D. Gardner and David A. Harle. Methods and Systems for Alarm Correlation. *Proc. GLOBECOM 1996*, pages 136-140.

[Jain *et al.*, 2008] Prateek Jain, Brian Kulis, Inderjit S. Dhillon, and Kristen Grauman. Online Metric Learning and Fast Similarity Search. *Proc. NIPS 2008*, pages 761-768.

[Jakobson and Weissman, 1993] Gabriel Jakobson and Mark D. Weissman. Alarm Correlation: Correlating multiple network alarms improves telecommunications network surveillance and fault management. *IEEE Network*, 7(6): 52-59, 1993.

[Klementtinen *et al.*, 1999] Mika Klemettinen, Heikki Mannila, and Hannu Toivonen. Rule Discovery in Telecommunication Alarm Data. *J. Network and Systems Management*, 7(4): 395-423, 1999.

[Steinder and Sethi, 2004] Malgorzata Steinder and Adarshpal S. Sethi. A Survey of Fault Localization Techniques in Computer Networks. *Science of Computer Programming*, 53(2): 165-194, 2004.

[Yemini *et al.*, 1996] Shaula Alexander Yemini, Shmuel Kliger, Eyal Mozes, Yechiam Yemini, and David Ohsie. High Speed and Robust Event Correlation. *IEEE Communications Magazine*, 34(5): 82-90, 1996.