Incorporate Visualization and Statistical Analysis to Improve Operator

Ujjal Aloke Sarkar
PhD Scholar, Dept. of Computer Science
Shri Venkateshwara University , Gajraula (UP)

Abstract: Web applications have software and configuration failures that lower reduce availability. The main reasons for recovering from failures are the interval between these failures occur and the intervals at which field operators detect failures. We introduced a set of tool operator ability to detect the presence of a failure. An automatic anomaly detection searches for user behavior changes that indicate site failures in the HTTP access log, and the visualizer helps operators quickly detect and diagnose problems. Visualization addresses key issues regarding how to gain operator confidence in autonomic computing to embrace new tools. Evaluations performed using Ebates.com HTTP logs show that these tools can enhance fault detection and reduce inspection time. Our approach is applications to generic and can be applied to any web application without the need for instrumentation.

Keyword: Web Applications, HTTP, Naive Bayes Classifier, Anomaly Detection and Visualization

I. INTRODUCTION

Web applications are becoming increasingly complex and difficult to manage. In particular, it is difficult to detect non-stop application level errors that cause user visible damage without special case checks, but it can lead to temporary or permanent site abandonment. Up to 75% of the time it takes to recover from these failures is used to detect them [4]. New concerns about statistical anomaly detection and pattern recognition [10] are expected to reduce the manual settings and adjustments required by current monitoring tools, and statistical methods (and sometimes false positive, reducing the operator's confidence in the monitoring system.

Rather than ignoring this basic fundamental trust issue, but rather than eliminate the human being from the loop, but to sense the division of labor that the operator is dividing into such a failure. We assign the computer the biggest feature of the computer, statistical analysis of log data. From the human operators are provided with tools to leverage the system experience and expertise to interpret and respond to alerts generated by the analysis tools. By exploiting the fact that humans are better at visual pattern recognition, visualization helps the operator interpret failure alarms and identify possible causes, and allow to identify them quickly by maintaining a low by allowing her to rapidly identify them as such.

In determining what type of analysis to perform on the site log, we found that the end users of the site were a good "detector" of the site failure in that behavior typically changes when they encounter a malfunction. For example, if the link from the/shopping cart page to the/checkout page is broken, users cannot access the/checkout page. Similarly, if a particular page is not loaded or rendered correctly, the user can click reload multiple times to resolve the issue. Such behavior is recorded in the HTTP logs, we can build statistical models of normal access patterns and then detect anomalies in user behavior. Because HTTP logging is common to application-generic, our approach can be applied to other web applications without the need for additional tools.

II. CONTRIBUTIONS

It provides a visualization tool that allows operators to quickly detect anomalies or potential problems in the field in real time and to review or investigate problem alerts reported by automated inspection systems. In order to explain the latter ability, we applied a relatively familiar anomaly detection technology to detect failures other than server log failures from the actual medium-sized Internet site Ebates.com; Look for anomalies in end-user behavior. As a possible indicator of failure. The information from these anomaly detectors is provided to the visualization tool, allowing the operator to visually identify anomalies in the context of previous and current traffic patterns and associate anomaly score information with the traffic timeline. Unlike traditional visualization systems, whose structure usually reflects system architecture, the visualization of our tools drives from the measurement of the site access behavior of users that are easy for publishers to understand. We find that using a combination of visualization and analysis tools, the Ebates operator can detect and find many real site problems a few hours or days earlier than the real site did. We make the following specific contributions.

• We use informative-rich visualizations to resolve the problem of operator confidence in statistical learning algorithms. The synergy between of visualization and automatic detection allows an operator to easily use human pattern verify the product by our monitoring system.

•When a site becomes available, it monitors user behavior and automatically detects anomalies, but individual applications and

features beginning to fail. This enables us to quickly detect and identify application-level failures from a real systems. Ebates.com.

• Visualization of information was not based on the standard system architecture, but is based on metrics based on "black-box" user behavior. These user-oriented metrics provide a better understanding of the site situation and align it with our statistical algorithms. This match build-up a trust relationship. Our method only uses HTTP logs to monitor user activity, so it can be used with any web application.

Section 2 outlines our approach to the combination of visualization and automated statistical anomaly detection. Section 3 describes the algorithm itself, experiment settings and methods, and evaluation metrics. Section 4 focuses on the relative strengths and weaknesses of the different algorithms and the use of visualization to allow the operator to rapidly bring her experience and judgment into play to resolve ambiguities in failure reporting across the different algorithms. Section 5 discusses the key aspects of our goal-based results to help businesses work more efficiently with automatic detection technology. Then we review some related work, outline possible future directions, and draw conclude.

III. APPROACH: COMBINING ANOMALY DETECTION AND VISUALIZATION

Our anomaly detection approach is relatively simple: We decided to look for the sudden change of the top 40 popular pages (which cover about 98% of traffic on Ebates). Basically, this problem is

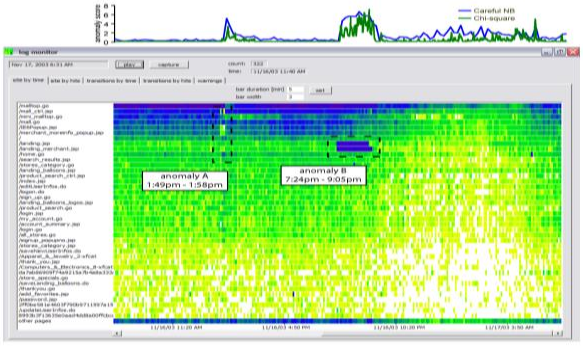

Figure 1. An annotated screenshot of the visualization tool (Note: If possible, figures 1, 2 and 3 should be viewed in color.) The horizontal axis is time at 5-minute intervals. Each horizontal bar represents the number of hit count for one of the 40 most requested pages, and the bottom bar represents the number of hits for all other page combined. A blue tiles indicate that the corresponding page was received within 5 minutes interval> 100 hits during that 5-minute interval; green > 10 hits, yellow > 1 hit, white (no tile) zero hits. The graph on the top shows the corresponding anomaly scores. In this screenshot, there are two

the baseline learning of hit frequency, detection of baseline deviation (abnormality), and the degree of influence of "lifetime" anomaly on baseline (i.e., the sensitivity with which the baseline itself shifts in response to recent and/or anomalous data). In addition, if an unusual frequency shift is detected, you need to determine which page is most likely to be involved in the exception and identify the problem. To perform this analysis we use two statistical methods: Naive Bayes classification and the χ 2 (Chi-square) test. (The details of these algorithms are described in section 3).Other anomaly detection methods such as Support Vector Machines) may perform better results, but it is not easy to use determine which page is the most anomalous. On the other hand, both Naïve Bayes and $\chi 2$ can quantify the anomaly for each page. The emphasis on "real-time" interactions with the data to distinguish between visualization and static graphic representations. In this example, as shown in Figure 1, the operator can drill down into the visually important flow anomalies in Figure 2 and see page-conversion rates during the anomaly period. The anomaly detection algorithm described above also sends information to the visualization tool. As will be explained later, the algorithm reports once a minute on whether abnormal behavior has been detected during that minute. To avoid impacting the operator with an unusual alarm lasting several minutes, combine the exceptions that follow the first exception into a single warning. In addition, for each alert, the tool reports changes in conversion rates between the most unusual pages, scores from anomaly detection algorithms, and most unusual pages. For example, if the operator clicks the “Warnings” tab in Figure 1 after selecting the anomaly marked as B in the diagram, the warning alert panel will display the following:

exceptions from data set 1. A (1:49 pm to 1:58 pm) and B (7:24 pm to 9: 05 pm).

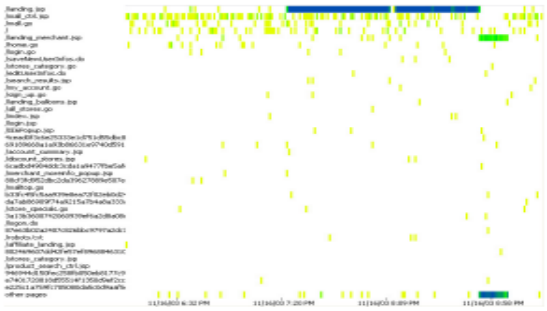

Figure 2. The page transitions from/landing-merchant.jsp page between data set 1, in 1-minute intervals. A sudden increase of transitions to/landing.jsp (the top most row) at 7:24 pm is an evident from the figure. This represents an alternate view of parts of the time period from figure 1.

IV. TEST DATA AND METHODOLOGY

In this section describes the HTTP log was analyzed, to give provides detailed information on the analysis algorithm, and describes the evaluation method before proceeding with the experimental results.

4.1. HTTP Access Logs

A typical three-tier Internet application consists of a web servers, a tier of an application logic server tier, and a persistent storage server. Common web servers include Apache and Microsoft IIS, and application servers can be framework-based, such as Java 2 Enterprise Edition (J2EE) servers, or customer- written in-house; can be provide to database such as Oracle or MySQL or by a file server.

Ebates.com provides 5 sets of (anonymous) access logs recorded by the web server layer. Each contains of HTTP traffic to three web servers over a continuous period of 7 to 16 days. Each period contains at least one web application failure and a fairly regular "normal" application operation. The access log contains the following information for each user request: Apache server time stamp, local URL of the page being accessed, URL parameters (part of the URL passed as a parameter to scripts on the active pages), session ID, application server that served the request, and anonymized user ID.

4.2. Analysis Using χ 2 –test

Intuitively, we might expect the under normal circumstances, the vector of page hits collected at different time intervals should be taken from the same distribution, or more generally, the number of pages collected during the current time interval the historical norm. The χ 2 test [12] can be used to calculate the two data vectors A = (a 1, ..., an) and B = (b 1, ..., bn) come from different distributions, it has used to detect network traffic in anomalies [14].

The test is performed in the following:

1. Let Sa = Pn = 1, Sb = Pn i = 1, and = i + as.

2. Calculate the expected value of each ai and bi. EAi = siSa / (Sa + Sb), EBi = siSb / (Sa + Sb).

3. Calculate the total χ 2 value of the two vectors: χ P 2 = n i=1(ai − EA i) 2/EA i + (bi − EB i) 2/EB i chi-squared value of the sum of two vectors.

4. The significance of the computer was tested in calculated by using a χ2 distribution with n-1 degrees of freedom.

5. Finally, the anomaly score is calculated as -log (1-s).

From the access log, first determine the site N of the most popular (by hit count) N pages (we used N = 40). Calculate the hit rate vector C = (c 1, ..., c 40) of each page during the current time interval and make it a "historical normal" vector H = (h 1, ... h 40) compare with the same pages of hit rate. Different type of traffic anomaly is different, it may take several hours to become evident, and so the length of the time interval will varying in length from 1 to 20 minutes. Since by definition, if each page is not clicked more than 5 times, the χ2 test is invalid and pages with fewer than 5 clicks are excluded. The actual algorithm, executed once a minute is as follows (where t is the current time):

1. The historical traffic pattern H of all over previous data is calculated, except for the time interval marked as anomalous. (At first, we assumed that all intervals are normal.)

2. For every t0∈ {1,2,…, 20} :

(a) Computer current traffic pattern C from time interval ht−t0, ti

(b) Compare C and H using the χ2 test. If the significance of the test exceeds 0.99, then the interval ht-t0, ti is marked as anomalous.

When the period is declared to be anomalous, we assign an anomalous score to each page based on the page's contribution to the total χ2 value: ((ci − EC i) 2/EC i + (hi − EH i) 2/EH i).

We also, significant changes in page transitions that occurred at the beginning of the anomaly were also detected. The traffic before the exception (time interval ht0-t, t0i, where t0 is the start of the anomaly) and the exception period (ht0, t1i, where t1 is the current time).Therefore, every time is anomalous traffic we use the transitions before and between the top 40 pages of the anomaly using the χ 2 – test.

4.3. Analysis Using Naive Bayes Classifier

The second type of analysis involves training a simple Bayesian classifier [6] to detect anomalies. Similarly, use the access log to calculate the number of clicks per unit time for each of the top N pages (c 1, ..., c N) of the site during the current time interval. The ci's is normalized by dividing it by the total number of hits in the interval to be in the range 0 to 1. We also calculated the difference between the total number of hits for all remaining pages on the site and the total number of past hits. By using period and current period simple Bayes models, we simplify the (incorrect) assumption that all 42 (= N + 2) features are conditionally independent. However, although Naive Bayes is often used successfully in practice, this theoretical requirement is rarely met. Divide the time into 10 minute intervals and use this classifier to determine if the current time interval is normal (S = s +) or abnormal (S = s-). The conditional probability for each feature f i given S = s + is modeled by a Gaussian distribution, where the maximum likelihood estimates from the previous time interval are the mean μ i and variance σ 2 for each feature used to estimate i.

If we know a priori which time intervals are abnormal and which are normal (in machine learning, when we tag the data), then p (fj | S = s +) and p (fj | S) Calculate The mean and variance of = s + is negligible. p (fj | S = s-) uses maximum likelihood estimation (MLE). However, as is often the case with real systems, the data is not marked (that is, it does not know which period is abnormal), so you need to do unsupervised learning. In the absence of the s-tag example, we decided to model the conditional probability p (fj | S = s-) using a uniform distribution over the range of possible values (ie 0 to 1).

The standard method of using Expected Maximization (EM) to simultaneously learn the value of s and p (f |s) is too slow to use for large amounts of data in real time, we so approximate in two different ways methods.

• Unweighted learning (Eager NB): We estimate μi and σ 2 i assuming that each previous time interval is normal, ie, "marking" each previous interval as S = s +. This is a reasonable first-order assumption as long as most conditions are actually normal, that is, failures are rare. However, if an error occurs and it is not resolved, treating this technique as an anomaly will cause it to "adapt" abnormally quickly. Therefore, we call it a "craving" learner.

• Probabilistically-weighted learning (Careful NB): When estimating μi and σ2i, we weight each past time interval with its normal probability. Thus, the more unusual the time interval that occurs, the less time it is included in the normal behavioral model. The method will continue to detect anomalies, but if a long-lived "abnormality" is indeed a new steady state, this method is longer to adapt to take time. Therefore, we call it a "careful" learner.

In our Naive Bayes approach, we did not learn prior probabilities of normal and abnormal time intervals. Instead, use a priori as a parameter to trade-off between low false positive rate (for low P rob (anomalous) and high abnormal rate (for high P rob (anomalous). The classifier reports an anomaly score for each time period; this score is calculated as –log (P rob(f|normal)/n, where n is the number of features used. To localize the most likely features that caused the anomaly, we assign an anomaly score to each feature fi as –log (P rob (fi |normal).

In many cases, operator intervention may be required to determine if a long-term anomaly is in fact a new steady state. Therefore, this cannot be determined automatically, but allows the operator to visualize the raw data and anomaly scores reported by each algorithm.

4.4. Methodology

The logs we received are collected in the past, and the failure events reflected there are diagnosed and processed. Thus, our approach involves the following steps:

1. With the little detection algorithm with little or no knowledge of what happened to the data set. For each data set, the model is initialized at the beginning of the data set and is trained online without training.

2. For each anomalous period (and some normal areas as well), use our visualization tool in conjunction with the Anomaly Score chart reported by the Anomaly Detection feature to check the traffic patterns of that period.

3. used the visualizations and diagrams, discuss with the CTO and operations engineers about at Ebates each reported event, reproducing as much as possible "what happens in the event".

4. Based on these arguments, each anomaly reported is classified as a true positive, a false positive (clearly attributable to a non-fault event, such as a failure-free update to the Web site), or a possible false positive (one we could not attribute to an event, or more often, that we could attribute but we could not unambiguously determine whether or not the associated event was a failure).

5. Based on these arguments, determine when the tool can be detected or blocked when the event occurs as the tool is deployed.

4.5. Evaluation Metrics

Traditionally, fault detection has been evaluated for accuracy and detection time. Accuracy is defined as true positives divided by total positives, ie T P / (T P + F P). Here, true positives are the number of actual failures detected and the number of identified events for which the error did not fail. However, these metrics have problems when dealing with real data from complex services.

First, part of the motivations behind our work are that existing detection technologies cannot detect certain types of outage failures. Thus, if the operator collates the results based on the best available knowledge, the list of known faults may not be all. Second, some of the false positives are non-fault events that change the user behavior, or events that cause actual performance anomalies that are not fatal at moderate loads, for example, but cause failures that the user sees at heavy loads. To be conservative, we count such incidents as "false positives" in our evaluation. Finally, the notion of "detection time" assumes that there is no failure before that and there is definitely a moment of failure that are not failsafe, especially those that occur only when the load increases, it is unknown how to select this point. Furthermore, even if it is assumed that the time of failure is clear, the actual knowledge to determine what that time is lacking.

The information we have is the time of the specific failure actually detected by the prior art (automatically or manually detected by Ebates employees) and the cause of the failure determined by Ebates employees (localized information) including. When measuring true and false positives, we detect all faults detected by

Ebates employees and consider it best not to misclassify other events. We also measure advance warning. This is because the staff has identified potential failures before making decisions and the staff has determined whether such pre-alerts can help mitigate or avoid the failure. Finally, we ask a question: Once the staff gets the localization information provided by our algorithm, how useful is it in determining the cause of the failure. In our measurements, Ebates employees reported qualitative responses from 1 (not useful) to 10 (very useful). (We did not collect localized score data for Eagle NB, one of our algorithms.)

V. DISCUSSION OF RESULTS FOR EACH DATA SET

In this section describes the combined results of anomaly detection and visualization for each ofn the datasets we examined. We can summarized to all over the differences of our techniques to each other algorithm to detect specific obstacles, and for the visualization of the role of the operator to clarify "what actually happens", and the implication for combining both types of operator information.

5.1. Data Set 1: Account Page

The major issue this week was crashing after the website slowed down due to the account page. Bad account pages have been around for a long time, but this period hits pages more than ever, as customers log in to view their accounts after sending them a refund check quarterly. Ebates first identified the problem on Monday at 6:23 am (data set 3), and Ebates diagnosed at around 12 noon and deleted the offending page at around 12:30 pm. The problem reappeared on Day 4 from 5:38 am to 7:13 am.The staff of Ebates tried to re-open the page but it was deleted. Around 8 pm on the 6th day, this page has a lot of quality assurance to check the fix on 2 servers on the live site. The correction was verified at 11:21 pm on the sixth day. Given this chronology reconstructed from Ebates's information, we now compare it to the behavior of our algorithm.

Day 2. We detected a mid-size anomaly (A) on Day 2 at 1:49pm and a significant anomaly (B) on the same day from 7:24pm until 9:05pm, centered around two pages not directly related to account activity. The number of hits to these pages increased from less than 5 every minute to about 50 hits an unusual B can be seen in Figure 1. After expanding this exception and switching to the transition view (Figure 2), you can see that the page mode accessed by the HTTP session has changed significantly immediately after accessing/logging to merchant.jsp. This is very important information for the operator. It was later discovered that these two exceptions correspond to database alerts generated by the Ebates database monitoring system. This is because the quarterly batch jobs cause a large load on the database. Figure 4 shows the algorithm's anomaly score over time and three database alerts displayed on Ebates that lasted 16 hours on this data set. It is important to note that the anomaly was detected at 7:24 pm (about 700 times in the figure) about 100 minutes before the third database alert is issued. Although the reported anomaly page is not directly related to the account page, Ebates staff may be alerted to possible problems with website operators by understanding this anomaly and may cause issues to be discovered earlier than they said there is.

Day 3. The next anomaly, the largest in the data set, starts at 11:07am on Day 3. This anomaly is shown in Figure 3. There are two exceptions (and the NB algorithm reports correctly). Most exception pages are account pages. At the moment, Ebates staff were aware of performance issues, but after 50 minutes we identified the cause of the exception alert where the two account pages were reported as the most abnormal. According to Ebates, this is a powerful diagnostic tip for the operator and may reduce the diagnostic time. The last major anomaly detected at 7:51 pm on the 6th day is due to intensive stress testing conducted in the field to validate the repair.

All three of these algorithms are also affected by significant nighttime anomalies that occur most of the night from midnight to 2am. In fact, all five datasets showed varying degrees of nighttime anomalies. For example, in Figure 1 you can notice the score for an unusual increase in this period. Later it turned out that these could correspond to the cash warming effects caused by the nightly cash flush (some employees do not know). There are other explanations for nighttime anomalies, such as the big difference due to the reduced number of hits. We will study this phenomenon further in the future. In summary, the anomalies and diagnostic information available in this case help to diagnose the cause of the performance degradation problem. You may also experience problems with your account page one hour before identification.

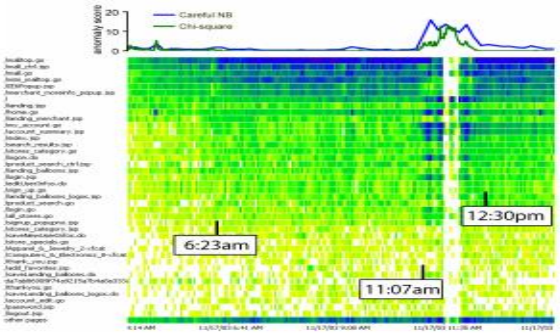

Figure 3. Visualization of anomalies followed by a site crash on day 3 of the data set 1. Ebates first discovered a problem (undiagnosed) at 6:23 am and Ebates had a problem at around 12:30 pm Diagnose and remove defects Account page our algorithm found the first exception at 11:07 AM and found.

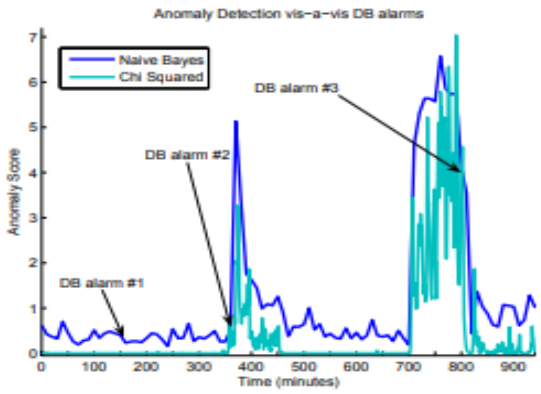

Figure 4. Anomaly Score and Database Alerts for Day 2 Dataset 1. The first database alert was not detected as an exception.

5.2. Data Set 2: Landing Loop

This data set includes a website crash with an error where the "login page" (site portal page) incorrectly redirects the user to the login page itself, eventually causing the site to crash. All three algorithms (careful NB, Eagle NB, and χ 2) correspond to the introduction of two new "login pages" to Ebates, two days and two days before Ebates detect the main problem A serious anomaly has been detected. All three on-site algorithms detected critical (and increased) anomalies at the site a few hours before the site crashed. Careful NB and Eagle NB provides localized information with a CTO rating of 8, with a rating of 1-10, to detect and diagnose problems to avoid crashes. χ2 has (possible) false positives. It was discovered early before the fall. Wishing for an NB, each NB has two (possibly) false positives. According to the CTO, even if the initial tests conducted 22 days before the accident did not represent the outbreak of the main virus, the warnings and location information provided at that time would diagnose the problem in the case of a serious failure very useful. It began to occur.

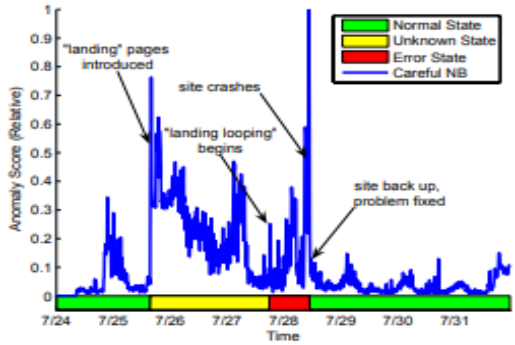

Figure 5. The discreet NB anomaly score over time in the data set 2 "Ling Cycle". The period from landing page introduction to Ebates's detection of landing loop problems was reported as an unknown system condition as it could not determine 100% whether this period was a problem for Ebates or not.

Figure 5. Which shows the careful NB anomaly scores over time for this data set, illustrating the effect of careful learning.When introducing problematic pages, careful NB detects significant changes in site traffic distribution. During this time, Careful NB has very low weights and incorporates it into the "normal" model, as this will significantly increase the anomaly score. Note that NB's sensitive traffic characteristics are still abnormal for the next two days. Thus, it is difficult for the NB to determine that this is actually a "normal" operation. Conversely, as shown in Figure 6 (which representing Data Set 3), the "new student with a thirst desires to work for all the same time periods as before, so the new, different traffic patterns are no longer anomalous. I strongly hope to conclude soon. All must contribute to the same profile that applies to 'normal' behavior. The third algorithm, χ 2 (shown in figure 7 of data set 5) works by detecting flow pattern changes over a relatively short period of time, and thus more bimodal behavior than careful NB.

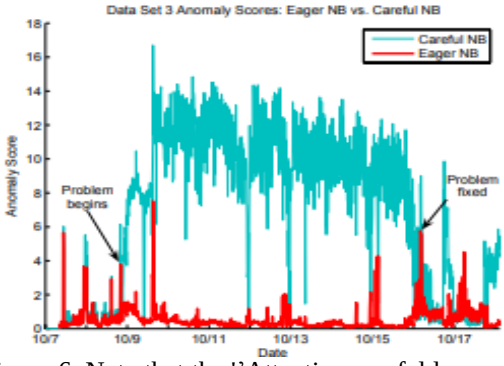

Figure 6. Note that the ''Attentive careful learner'' NB continues to detect the anomalies in all registered pages (data set 3) for all 7 days, "eager learner" Eager NB quickly decided that this new behavior is normal.

5.3. Data Set 3: Broken Signup

This data set does not contain crashes, the deployment of the new user registration page not detected by the Ebates operator is incomplete. When serving new users for more than a week, this page will display blank pages with errors instead of the expected site content, making it impossible for new users to access the site. This issue does not affect existing users.

Seven days before Ebates diagnosis, careful NB and χ2 were detected in the introduction of the problem. (Eager NB has also detected an anomaly at this point, but it is close to its noise threshold, so I don't think this is Eager NB's detection.) Attention NB is the problem's introduction time and the entire 7 days Provides localization information for in objection, Ebates said it was very useful for localization issues. Due to the length of the anomaly, Eager NB and χ 2 begin to treat the anomaly period as normal, but carefully NB ("Careful Learner") reports the entire 7 day anomaly as shown in Figure 6 I will keep doing. Another

example is that the operator's understanding of the system can help resolve the ambiguous results of algorithms that are sensitive to different time scales.

5.4. Data Set 4: Badly Optimized Page Bug

In dataset 4, a new page that invokes an inefficient database query crashed the site when the hit rate increased due to heavy email activity. As a result, the database was overloaded and the site failed. 4.5 hours before the accident (3 hours before Ebates staff found the problem), NB was alerted, and the enthusiastic NB detected the anomaly. Anomalous levels are initially low and increase as the problem gets worse. The χ 2 problem was detected 3 minutes after Ebates staff detected the problem. Ebates staff believes that they may have avoided crashing if they have unusual localization information provided by Careful NB.

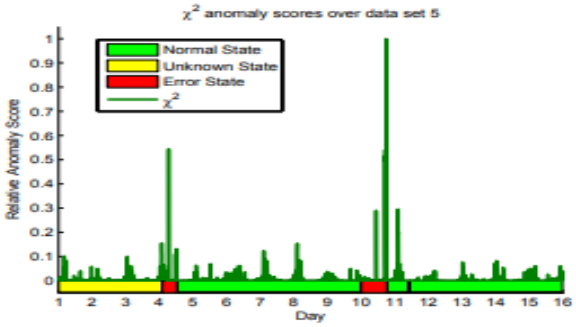

Figure 7. Anomalous score of χ 2 over time for data set 5 "Query with URL illegal and runaway". The main failure occurred on the 4 and 10 days.

5.5. Data Set 5: Bad URL and Runaway Query

Dataset 5 has two major mistakes. The first failure is due to a configuration error that caused one of Ebates's "shopping URLs" to be incorrectly mapped to a popular search engine site. All linked Ebates users will be sent back to Ebates where they will quickly and continuously generate shopping sessions until the site crashes.

χ2 detected this defect 5.5 hours before Ebates diagnosed the problem. Although both NB algorithms detect problems simultaneously, they are not used as detection of the resulting NB algorithm because the anomaly score is within the noise level. However, Ebates staff reiterated that the localization information provided by the NB algorithm would be a great diagnostic aid. The second major failure is that runaway queries have caused serious problems with database performance. All three algorithms detected this failure as soon as Ebates was detected. Figure 7 shows the behavior of the χ2 algorithm on this data set.

5.6. Summary of Results

Table 1 summarizes the overall results of our analysis. For five of the six major failures in log data, at least one algorithm detected the problem and provided useful localization information before Ebates diagnosed the problem. Our algorithm has seven glitches (four database alerts, a short break associated with one code push, one error reintroduction error page, and one large QA work to validate the fix at the live site) The performance at was low. Of the three missed faults (missing), two are database alerts and do not have a significant impact on the user. The third is a brief introduction, but the wrong page is deleted. Three known false positives are all fault-free code updates to the application. Predictable nighttime anomalies are not considered false positives as they are easily filtered out by time. We did not perform localization or advance-warning analysis on the minor faults.

Table 1. Summary of results for all five datasets In the case of a serious failure, χ2 has higher detection rates and fewer false positives than Careful NB or Eager NB, but discreet NB is more useful diagnostic information.

Major fault	Careful NB	Eager NB	χ 2
Faults Detected	5/6	4/6	6/6
Known FP's	1	1	1
Possible FP's	3	3	2
Detection rate	83%	67%	100%
Precision	56-83%	50-80%	67-86%
Local. Score	8.6/10	n/a	4/10
Minor faults	Careful NB	Eager NB	χ 2
Faults detected	4/7	4/7	4/7
Known FP's	3	3	0
Possible FP's	5	4	2
Detection rate	57%	57%	57%
Precision	33-57%	36-57%	67-100%

Table 2 summarizes the results grouped by data set. For each data set, the number of primary and secondary failures detected (total primary and secondary failures respectively), the number of known false positives and possible false positives, the early warning time

of critical failures (we The length of time from anomaly detection to Ebates' initial location-related failure by the Algebra's algorithm; and the usefulness of diagnostic information estimated by Ebates staff, ranging from 1 to 10, lowest to highest For range data set 2, it is not known if the test represents the actual occurrence of the problem (see section 4.2), otherwise the detection in the data set takes place approximately with the detection of Ebates.

Table 2. Dataset performance by using χ 2 for detection, NB careful for positioning.

Measure	DS1	DS2	DS3	DS4	DS5	Total
Major faults	1/1	1/1	1/1	1/1	2/2	6/6
Minor faults	3/5	0/0	1/1	0/0	0/1	4/7
Known FP's	0	0	1	0	0	1
Possible FP's	0	1	1	1	1	4
AWT	1h	50h?	7d	0m	5.5h,0m	avg: 37h
Local. score	8	8	9	10	8	avg: 8.6

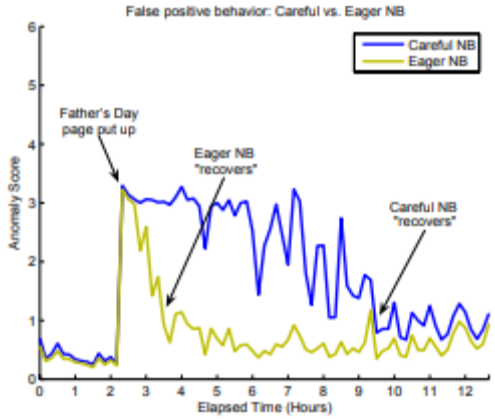

Figure 8. The different behaviors of prudent and enthusiastic learning when dealing with false positives. It takes 9 hours for NB to "recover" from false positives, and avid NB recovers in about 90 minutes.

VI. DISCUSSION: ROLE OF THE OPERATOR

6.1. Classifying False Positives

In our experience, operator support is essential to assert the basic facts of the alert system. In our case, we need their help to explain the failure data and validate the conclusions of the anomaly detector, but in general, the operator is most eligible to judge the anomaly or longevity, true Problem or false positive. Figure 8 shows the tradeoff between early detection and false positive. Careful NB and Eager NB have declared an exception with the introduction of a new, error-free "Father's Day" page on the site. There were no problems with this page, and it took Eager NB only 90 minutes to reach this conclusion. Similarly, even if Ebates only sent an email to all customers, even if the activity on the website increased, the anomalies detected by the site crash several hours before the exception page is generated by the operator will be rejected by the operator might realize that the localization information could help drill down on the problem and determine if there was really data set problem.

6.2. Detecting Different Types of Anomalies

Naive Bayes and χ2 respond to changes in different types of traffic patterns and help complementary tasks. Naive Bayes is sensitive to increasing frequency of infrequent pages. Since Naive Bayes models the hit frequency of each page as a Gaussian, 5% of the page growth is modeled as: It is unlikely. In our data set, NB was a useful diagnostic tool, as relatively infrequently accessed pages, were associated with many failures. In contrast, the χ 2 test is robust against changes in the number of hits on unpopular pages. This is because, for the validity of the test, it is necessary to exclude pages that did not receive at least 5 hits within a certain time. As a result, χ2 is sensitive to frequent page additions and reductions, but it is not very useful for accurate alignment. Bias for frequently accessed pages often reports these pages as the most unusual. How, most failures are usually affected at some point. For this reason, the summary of the results for each data set reported in Table 2 relies on naive Bayes for positioning with χ2 for detection and probability weighted learning (careful NB).The difference in behavior between the methods can be seen in Figure 9. This indicates that the data set 4.NB was able to detect this anomaly 3 hours before χ2 because it detected an increase in

the number of hits on a very rare page. The frequency of these pages is increasing and after 3 hours it will have a serious negative impact on the site. This results in a change of hits on the high frequency page, which is then detected by χ2. Thus, it is determined which algorithm has detected an anomaly to inform the operator about the nature of the anomaly, and based on experience, the operator learns to recognize such "patterns" which are later detected by the algorithm Can. one more. Similarly, our contribution is the extensive use of operator experience and an understanding of the system, and visualization to take advantage of the ability to quickly absorb visually presented information.

6.3. Reconstructing the Ground Truth

To precisely and calculate the accuracy, detection time, and false positive/false negative rates of our technology, we know what happened, when, and when to detect those "basic facts" is needed. The reconstruction of this information requires the cooperation of the operator, and even after viewing the system monitor log, email archive and chat log, the operator can only partially reconstruct certain events. Not sure exactly when the failure occurs (only when they were detected by Ebates staff), whether the three exceptions detected actually correspond to a temporary (but not detected) site problem It cannot be judged or it is false positive. This can make it difficult to determine if the anomaly detected before the actual failure is an early warning signal or an unrelated false alarm. I think this is an essential problem in dealing with actual fault data. Thus, instead of reporting “detection time” and “false detection rate”, alternative metrics based on the amount of pre-alerts provided by the tool are compared to the existing detection methods used by Ebates.

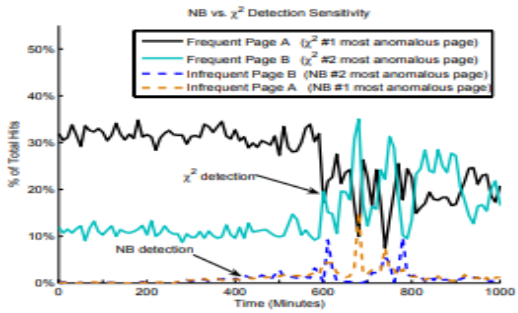

Figure 9. Is a graphical representation of the difference in detection sensitivity between and χ2. NB PWL is sensitive to increased clicks on infrequent pages. This will allow you to detect changes early and be a better locator. χ 2 is more sensitive to hitting more frequent page changes, making false positives.

VI. FUTURE WORK

Ebates was interested in introducing visualization tools to real-time data and working with us to develop better "basic facts" for evaluation. However, since the basic facts that are 100% accurate may not be realistic, we have repeated the experiment by acquiring similar data sets from the other two companies. We want to incorporate real-time feedback from operators into our models and generate smarter alerts. For example, if the current long-term anomaly indicates normal operation, the operator should be able to specify that this is actually a new normal operation. On the other hand, if the latest exception indicates normal behavior (e.g., page update), do not report a warning the next time a similar exception occurs. We are also extending the network traffic model to understand the correlation between the frequencies of pages.

VII. CONCLUSION

Despite the statistical analysis techniques for detecting and identifying Internet service failures, expert operator judgment eliminates disambiguate conflicting warnings, resolves obvious false warnings, and alerts with these algorithms help to explain the problem. The combination of visualization and anomaly detection and positioning makes more effective use of their expertise and experience to solve these problems, reducing the cost of classifying examination time, diagnostic workload, and false positives. In particular, we used Naive Bayes and χ2 tests to detect abnormal user traffic on real medium-sized internet sites. Our technology detects four of the six faults faster than the site staff, and visualization helps to understand the type and cause of the anomaly reported by the algorithm. There is a critical synergy between visualization and automatic detection from the perspective of the autonomic computing. Many traditional visualization tools are based on the configuration of the system. In contrast, our tools present information in a format that is useful to the operators so that they can monitor their systems and quickly determine if the visualization tools are effective and useful. From that foundation of trust, we may use the same visual form to automatically indicate suspicious behavior. The operator can immediately determine if these whether or not these warnings are useful. Without a visualization tool, many warnings will be displayed to determine if each operator trusts the tool. Because the detector and the visualizer use the same metric, it is easy and quick for the operator to determine if the alert is a false positive. As a result, false positives are much cheaper; they can be important behavioral changes that the operator may want to know, or can be easily excluded manually and visually. A quick visual inspection may actually provide a higher false positive rate in practice.

VIII. REFERENCES

[1]. Altaworks, Altaworks Panorama. http://www.altaworks.-com/solutionsPanorama.htm.

[2]. S. K. Card, J. D. Mackinlay, and B. Shneiderman. Readings in Information Visualization: Using Vision to Think. Morgan Kaumann, San Francisco, CA, 1999.

[3]. M. Chen, E. Kıcıman, E. Fratkin, A. Fox, and E. Brewer. Pinpoint: Problem determination in large, dynamic internet services. DSN 2002.

[4]. M. Y. Chen, A. Accardi, E. Kıcıman, D. Patterson, A. Fox, and E. A. Brewer. Path-based failure and evolution management. In NSDI, pages 309–322, 2004.
[5]. I. Cohen, J. S. Chase, M. Goldszmidt, T. Kelly, and J. Symons. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In 6th USENIX Symposium on Operating Systems Design and Implementation, San Francisco, CA, Dec 2004.
[6]. E. Eskin. Anomaly detection over noisy data using learned probability distributions. In Proceedings of the International Conference on Machine Learning, pages 255–262, 2000.
[7]. Hewlett Packard Corporation, HP OpenView Software. http://www.openview.hp.com.
[8]. IBM Corporation, IBM Tivoli Software. http://www.ibm.-com/software/tivoli.
[9]. C. Kallepalli and J. Tian. Measuring and modeling usage and reliability for statistical web testing. IEEE Transactions on Software Engineering, 27:1023–1036, 2001.
[10]. J. O. Kephart and D. M. Chess. The vision of autonomic computing. IEEE Computer, 36(1):41–50, 2003.
[11]. E. Kıcıman and A. Fox. Detecting application-level failures in component-based internet services. Technical report, Stanford, 2004.
[12]. G. W. Snedecor and W. G. Cochran. Statistical methods. Eighth Edition, Iowa State University Press, 1989.
[13]. J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: Discovery and applications of usage patterns from web data. SIGKDD Explorations, 1:12–23, 2000.
[14]. N. Ye, Q. Chen, S. M. Emran, and K. Noh. Chi-square statistical profiling for anomaly detection. In IEEE Systems, Man, and Cybernetics Information Assurance and Security Workshop June 6-7, 2000 at West Point, New York, pages 187–193, June 2000.