Adtributor: Revenue Debugging in Advertising Systems

Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah, *Microsoft*

https://www.usenix.org/conference/nsdi14/technical-sessions/presentation/bhagwan

This paper is included in the Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14).

April 2–4, 2014 • Seattle, WA, USA

ISBN 978-1-931971-09-6

Open access to the Proceedings of the 11th USENIX Symposium on Networked Systems Design and Implementation (NSDI '14) is sponsored by USENIX

Adtributor: Revenue Debugging in Advertising Systems

Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese,
Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah
Microsoft

Abstract

Advertising (ad) revenue plays a vital role in supporting free websites. When the revenue dips or increases sharply, ad system operators must find and fix the root-cause if actionable, for example, by optimizing infrastructure performance. Such revenue debugging is analogous to diagnosis and root-cause analysis in the systems literature but is more general. Failure of infrastructure elements is *only one* potential cause; a host of other *dimensions* (e.g., advertiser, device type) can be sources of potential causes. Further, the problem is complicated by *derived measures* such as costs-per-click that are also tracked along with revenue.

Our paper takes the first systematic look at revenue debugging. Using the concepts of explanatory power, succinctness, and surprise, we propose a new multi-dimensional root-cause algorithm for fundamental and derived measures of ad systems to identify the dimension mostly likely to blame. Further, we implement the attribution algorithm and a visualization interface in a tool called the Adtributor to help troubleshooters quickly identify potential causes. Based on several case studies on a very large ad system and extensive evaluation, we show that the Adtributor has an accuracy of over 95% and helps cut down troubleshooting time by an order of magnitude.

1 Introduction

Many free websites are supported today by revenue generated through advertisements (ads). Website ads can be of two types, namely, search and display. In the case of a search ad, an end user goes to a publisher website such as bing.com and enters a query phrase. The response to the query is a search results page that may contain one or more ads. If the user clicks on one of these ads, the publisher earns revenue. In the case of a display ad, an end user may visit a publisher website, such as cnn.com, where she might see ads at the top or sides of the page. The display of these ads earns revenue for the publisher.

Ad systems facilitate generation and accounting of millions of such search and display ads every day. Apart from *users* and *publishers* noted above, there are two other key constituents who interact with the ad system. The ads shown to the user are the result of an ad auction between various *advertisers* who bid to compete to have their ad displayed to the user. Also in the midst are various *fraud operators* [8] that try to usurp a fraction of the advertising revenue.

Ad systems manage the interaction between users, publishers, advertisers and fraud operators. Ad systems implement various ad-related algorithms that run the real-time ad auctions between the advertisers, return the winning ads to the publisher, monitor the user clicks, detect and remove potential fraudulent activity, compute the revenue from each displayed or clicked ad, charge the advertiser the appropriate bid amount, and pay the publishers. At the core of the ad system is a large-scale distributed system consisting of thousands of servers distributed across several data centers that execute the ad algorithms and manage the serving and accounting of ads.

The focus of this paper is on debugging ad systems. Typically, an ad system monitor issues an alert whenever a measure of interest is identified as anomalous (e.g., revenue or number of searches is down sharply). [1] Our goal is to automatically identify the potential root cause of this anomaly. We term our approach *revenue debugging*, even though it is applicable to several measures of interest to ad system operators, to acknowledge the prominence of the revenue metric. In this paper, we describe a new revenue debugging algorithm that analyses the large amount of data logged by the ad system and narrows down the scope of potential root-cause of an anomaly to a sub-component of the ad system for further investigation by a human troubleshooter.

Root-cause identification and diagnostics is an age-old problem in systems. Various performance root-causing tools have been proposed in the past [1, 2, 3, 10, 14, 15]. But all these solutions have focused on performance/failure debugging. Here, we address a similar yet more general problem: diagnostics in ad systems. While performance/failure of infrastructure systems components can be one possible root-cause for an anomalous measure, there may be various other root-causes that depend on other components that interact with the ad system. Consider the following examples.

1. Papal Election: We noticed that the papal election caused a revenue drop because many searches were made for non-monetizable query terms such as pope or papal

[1] Anomaly detection is a challenging problem in itself but is out of scope of this paper.

election, that advertisers typically do not bid for. The total number of ads shown dropped which resulted in an anomalous revenue drop. While identifying the root-cause as the papal election is not actionable, root-cause identification is still important as it eliminates an actionable root-cause such as the example below.

2. Browser Ad Failure: We found a revenue drop was caused by a manual error in updating a configuration file that had the side-effect of not showing ads on certain browser versions. In this case, quick identification helped rectify the configuration error, thereby restoring advertising revenue. A more extensive set of examples is depicted in Table 1 and discussed in Section 2.1.

The first challenge in ad systems debugging is sheer scale. There are hundreds of millions of searches and clicks every day; performing diagnostics at the level of a search or a click is not scalable (imagine running Magpie [3] or tracking a string of system calls through hundreds of system components for every click). Thus, for scalability reasons, ad system debugging operates over aggregates of various *measures*. These measures are typically counters aggregated over certain time intervals (e.g., revenue generated over the last 1 hour). Root-cause identification can only be triggered by anomalous behavior of these aggregate counters.

A second distinguishing characteristic of ad systems as compared to typical systems trouble shooting is the existence of multiple *dimensions*, and the need to first isolate the dimension that explains the anomaly. Measures such as revenue can be broken down or projected along different dimensions such as advertiser, browser, or data center. For instance, in Example 2, if revenue were projected along the browser dimension, one could observe that some browser versions were not generating their "typical" share of revenue. However, if the same revenue were sliced by the advertiser dimension, perhaps the distribution of revenue would not have changed significantly.

Typical systems root-causing algorithms such as SCORE [11] use succinctness (Occam's razor) and explanatory power (does the root-cause explain the change?) as their main parameters for optimization and do not have to account for multiple dimensions. To isolate anomalous dimensions, we introduce the notion of *surprise*, captured by quantifying the change in distribution of measure values across each dimension. For instance, in Example 2, change in distribution of revenue along the browser dimension is more surprising than the change in distribution of revenue along the advertiser dimension. Thus, *our first contribution in the paper is the root-causing algorithm described in Section 3* that uses surprise in addition to succinctness and explanatory power to identify root-causes in ad systems.

A third unique characteristic of ad systems is the prevalence of derived measures. Consider two fundamental measures: revenue per hour and number of clicks per hour. From these two measures, one can define a derived measure called cost-per-click that is simply revenue divided by number of clicks. Ad system operators monitor and track many such derived measures that are functions of various fundamental measures (see Figure 1). For example, the change in number of clicks and change in revenue may be small by themselves and not anomalous (e.g., less than 10%). However, *correlated* changes (e.g., revenue drops and simultaneously clicks increase, each by say 10%), are anomalous and is captured by the derived cost-per-click measure (20% change). As we discuss in Section 4, attributing a root-cause to derived measures is challenging. To address this, *we propose a novel partial-derivative inspired attribution solution for derived measures, our second contribution of the paper.*

The outcome of our root-cause identification algorithm is a set of candidates that potentially explain an anomaly. However, this is only the first step in the diagnosis process where a troubleshooter may, if appropriate, take actions to fix the issue. To help the troubleshooter quickly identify potential root-cause candidates, *we have implemented our root-cause identification algorithm and a graphical visualizer in a tool called the* Adtributor*, our third contribution of the paper*. Through experiences from a pilot deployment in a production system, we have refined the visualization interface and data representation techniques in Adtributor to further reduce turnaround time for troubleshooters.

Finally, we perform extensive evaluation of our root-causing algorithm. First, we tabulate and discuss a representative set of case studies that highlight the value of our root-causing tool. Second, we evaluate our algorithm on 128 anomaly alerts over 2 weeks of real ad system data and find that our algorithm achieves an accuracy of over 95%. In fact, Adtributor even found root-causes for a few anomalies that were missed by the manual troubleshooters. Further, the tool also speeds up the troubleshooting process by an order of magnitude.

2 Problem Statement

In this section, after providing a system overview, we show examples of real problems and their root-causes. Next, we state the problem more precisely and motivate our solution.

2.1 System Overview

Figure 1 shows a simplified representation of an ad system, and the entities such as users, fraud operators, publishers and advertisers that directly interact with the ad system. The ad system itself has various subcomponents, some of which we show.

While the logging infrastructure does track each

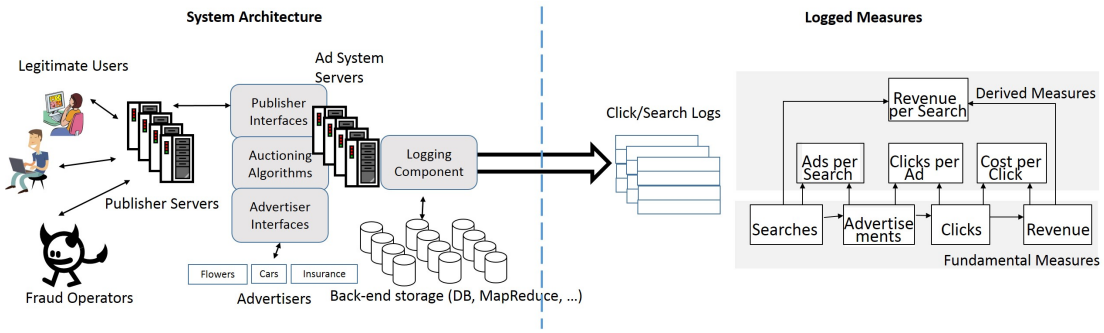

Figure 1: **A simplified representation of an ad system, and the measures it monitors.**

search request or ad-click, the sheer scale makes it hard to track down a problem at the individual request level. Instead, the system monitors a set of *aggregate measures*, as shown in Figure 1. From the raw logs, it first calculates, for each time interval, total searches received, total ads shown, total ad-clicks received, and total revenue from these clicks. These measures are all additive, and can be sliced along different dimensions. For instance, the total revenue is the sum of the revenue made from each advertiser using the system. The total revenue is also the sum of revenue received from different geographical regions where the ad system is active. We term such additive measures *fundamental measures*.

Additionally, the system also monitors a set of non-additive *derived measures*, which are functions of fundamental measures, such as ads-per-search (ads/searches), clicks-per-ad (clicks/ads), cost-per-click (revenue/clicks), and revenue-per-search (revenue/searches).

An anomalous rise or drop in any of these measures is an indication of a problem. Therefore, a diagnostic engine needs to first detect an anomaly, and then perform root-cause analysis. In this paper, we focus on the latter aspect of root-causing, while relying on well-known ARMA model-based methods [4] for anomaly detection. The anomaly detector generates a model-based prediction of measure values based on 8 weeks of historical data, taking into account normal time-of-day and day-of-week fluctuations. It then compares the actual value with the forecasted value – when the actual value of a measure is significantly different from the forecasted value, it generates an anomaly alert. The threshold difference above which we generate an alert is measured in terms of a percentage deviation from the expected value. In the current system, troubleshooters manually set this value based on experience. For each alert, our objective is to attribute the anomaly in a measure to a *dimension* and its corresponding *elements*. We define these terms next.

Dimension: A dimension is an axis along which a measure can be projected. For instance, we can project revenue along the axis of advertisers, and determine how much revenue comes in from each advertiser. The dimension in this case is "Advertiser". Derived measures can be similarly projected across dimensions. Some other dimensions are "Publisher", "Data Center", and "User Location". Typically, an ad system deals with dozens of such dimensions. Note that a dollar of revenue could be added to Advertiser 1 in one dimension, and Publisher 3 in a second dimension.

Element: Every dimension has a domain of values called elements. For instance, the "Advertiser" domain can have the following elements: {*Geico, Microsoft, Toyota, Frito-Lay, ...*}. The Publisher dimension may have elements: {*Bing, Amazon, NetFlix, ...*}.

Table 1 provides a number of problem examples we encountered, both actionable and not actionable, that need to be detected and root-caused to the appropriate dimensions and elements. Column 1 shows that problems can happen at various levels. Column 3 shows the anomalous measure. Column 4 shows the output of the root-cause analysis, the focus of this paper.

Note that Column 4 is only the first step towards root-causing, but it is essential as it gives the troubleshooter the best indication of where the problem actually lies. Other post-processing techniques (correlation engines, NLP techniques, manual investigation) use the output of the multi-dimensional analysis to perform a deeper dive into the issue to arrive at the final root-cause, shown in Column 5, but this aspect of root-causing is outside the scope of this paper. For instance, in row 9, while the multi-dimensional analysis did narrow down the problem to a few query strings, an administrator had to semantically interpret the strings to determine that the papal election was the cause.

2.2 Problem Definition and Scope

The multi-dimensional analysis problem of revenue debugging is to find the dimension and its elements that best explain an anomalous rise or fall in a measure. In this context, we need to define what constitutes the "best explanation" for an anomaly.

Consider the following example. The revenue of an ad system was forecasted to be $100 at a given time. In real-

Category	No.	Symptom	Faulty Dimension and Elements	Final Root-Cause
Infrastructure	1.	Ads shown dropped	Data Center: DC1	Deployment of certain updates to data center DC1 failed.
	2.	Revenue dropped	Log Server: L10, L11, L12	Bug caused abnormally large logs on these logging servers, and they went out of storage.
Ad System	3.	Revenue increased	Bucket: B1, B2	Buckets are A/B tests that are run on disjoint subsets of traffic to test new algorithms. Buckets B1 and B2 were using a different algorithm that increased the number of ads they showed.
	4.	Ads, revenue dropped	Browser: WB1	Configuration file error caused no ads to be shown to users who used web browser WB1. See Section 6.
Advertiser	5.	Cost-per-click, revenue increased	Advertiser: A1, A2, ..., An	These advertisers were all retail companies who increased their budgets during the holiday shopping season. This caused auction prices to go up, thereby increasing cost-per-click and revenue. See Section 6.
	6.	Cost-per-click dropped	Advertiser: Ax	A large advertiser Ax reduced their marketing budget drastically. This caused an overall drop in revenue, and clicks on ads from this advertiser. This made the cost-per-click drop anomalously.
Publisher	7.	Clicks-per-ad increased	Publisher: P1	One publisher launched a new UI with more ads shown on the top of the page than on the side. Users tend to click more on ads at the top of the page, and so this publisher reported more ad-clicks. See Section 6.
	8.	Revenue dropped	Publisher: P2, P3	Publishers P2 and P3 started blocking ads returned by the ad system to make for a cleaner UI. Their revenue dropped.
User	9.	Ads-per-search dropped	Query string: "pope", "papal election"	During the papal election, users searched for "Pope", "Papal election", etc. which are non-monetizable searches. These searches showed no ads, consequently the derived measure ads-per-search dropped.
	10.	Revenue dropped	User Location: New Orleans	A hurricane in New Orleans caused fewer searches from the affected geographical areas.
Fraud	11.	Searches increased	User-agent String	A large number of searches used an identical user-agent string. This was traced to a bot that was spoofing search requests and blindly replicating the user-agent string. See Section 6.

Table 1: **Some example issues that cause anomalies in advertising system measures.**

Data Center	Forecasted Revenue	Actual Revenue	Difference
X	$94	$47	$47
Y	$6	$3	$3
Total	$100	$50	$50

Table 2: **Revenue by Data Center**

Device Type	Forecasted Revenue	Actual Revenue	Difference
A1	$50	$24	$26
A2	$20	$21	-$1
A3	$20	$4	$16
A4	$10	$1	$9
Total	$100	$50	$50

Table 3: **Revenue by Advertiser**

Device Type	Forecasted Revenue	Actual Revenue	Difference
PC	$50	$49	$1
Mobile	$25	$1	$24
Tablet	$25	$0	$25
Total	$100	$50	$50

Table 4: **Revenue by Device Type**

ity, the actual revenue was only $50. An alert is triggered on the revenue measure, which brings a troubleshooters attention to the problem.

To find the root-cause when such problems occur, the ad system continuously tracks the revenue generated across a host of dimensions. For this scenario, consider three such dimensions: Data center (DC), Advertiser (AD), and Device type (DT). Tables 2, 3, 4 show the projection of revenue values along these dimensions, and the values attributed to the individual elements.

We now explain the semantics of these attributions. When the ad system receives a search query, it routes the query to a data center that in turn serves a number of ads in response. The revenue attributed to a data center is the total revenue received from clicks on ads that this data center serves. Each ad has an associated advertiser. When a user clicks an ad, the system charges the advertiser a pre-determined sum of money. The revenue attributed to the advertiser is the total cost of all such clicks made on the advertiser's ads. Users make search queries using a host of devices, which could be phones, tablets, or PCs. The revenue attributed to a device type is the sum total of all revenue that the ad system obtains from ad-clicks from that specific device-type.

The question that we seek to answer is: how do we pinpoint the revenue drop to the right dimensions and their elements? We restate the problem as follows:

"Find a Boolean expression, in terms of dimensions and their elements, such that the revenue drop attributed to the expression best explains the total drop in revenue."

While we examine how to determine "best" shortly, consider the following expressions that could explain the $50 revenue drop:

$$Revenue\_Drop(DC == X) = \$47 \quad (1)$$

$$Revenue\_Drop(AD == A1 \vee AD == A3 \vee AD == A4) = \$51 \quad (2)$$

$$Revenue\_Drop(DT == Mobile \vee DT == Tablet) = \$49 \quad (3)$$

For example, equation 2 states that the sum of the differences between the forecasted and actual revenues for rows 1, 3, and 4 of the advertiser table is $51, which is very close to the total revenue drop of $50.

In general, such expressions could include multiple

dimensions such as $Revenue\_Drop$ $(DT == PC \wedge DC == X)$ which refers to a revenue drop across PC users served ads from data center X. Based on about one year of monitoring alerts in ad systems we have observed, through manual study as well as through using an attribution algorithm that blames anomalies on multiple dimensions, that such cases where multiple dimensions contribute together to a root-cause are very rare. Therefore, for simplicity of exposition, in this paper, we limit our discussions to finding a Boolean expression that involves a single dimension and a set of its elements that explains the anomalous change.

To understand what constitutes the *"best"* dimension and a set of its elements, we studied several criteria. Consider the following strawman approach that motivates our final problem statement.

Strawman: Find the dimension and a set of its elements whose revenue drop is at least a threshold fraction, T_{EP}, of the total revenue drop, and is most succinct.

We quantify the *explanatory power* (EP) of a set of elements as the fraction of the measure change that it explains. We quantify *succinctness* (P) of a set of elements as the total number of elements in the expression. Therefore, the strawman will find the expression that has explanatory power of at least T_{EP}, and uses the smallest number of elements.

Occam's razor suggests that the most succinct set, as long as it explains the drop within a certain margin of error (T_{EP}), is the best explanation. By this argument, if T_{EP} is set to 0.9, the best dimension and set of elements among the three equations is in Equation 1, since the data center X alone can explain 94% of the total drop.

This approach, however, has deficiencies for root-causing in the presence of multiple dimensions. Though data center X's revenue drop is a high 94% of the total revenue drop, notice that both the forecasted and actual revenue are equally spread between the two data centers X and Y. Data center X provided 94% of the forecasted revenue ($94 out of $100), and actual revenue ($47 out of $50). Data center Y contributed 6% across both values. By comparison, in the device type dimension, device type *PC* contributed 50% of forecasted revenue ($50 out of $100), but 98% of actual revenue ($49 out of $50). The contributions of *Mobile* and *Tablet* device types also varies widely from 25% of forecasted revenue to 0% of actual revenue. The contributions vary along the advertiser dimensions as well, but not as much as they do along the device type dimension.

This large change in the contributions between forecasted and actual revenue from the different elements of the device type dimension is, in general, *surprising and unexpected*. Consequently, we propose that surprise is a better indication of a problem than if we only used succinctness and explanatory power of an expression. Say the root-cause of this revenue drop was due to a configuration file error which caused no ads to be shown on mobiles and tablets. While data center X would still show a huge drop in revenue because it provides 94% of all ads shown across devices, the actual root-cause is better explained by the device type dimension, and the elements *Mobile* and *Tablet*. In other words, the expression in Equation 3 is the best one, even though it is not the most succinct.

To capture this observation, our approach includes a notion of "surprise" (S) associated with an expression (Section 3 has the precise definition). Therefore, generalizing to any measure, our final revenue debugging problem statement can be captured in three steps:

- For a dimension, find all sets of elements that explain at least a threshold fraction, T_{EP}, of the change in the measure (have high explanatory power).
- Among all such sets for each dimension, find the sets that are *most succinct* in that dimension.
- Across all such sets for all dimensions, find the one set that is the *most surprising* in terms of changes in contribution.

Again, for the mock example, with T_{EP} = 0.9, the first step will narrow down the sets to {X} for Data Center, {A1, A3, A4} for Advertiser, and {Mobile, Tablet} and {PC, Mobile, Tablet} for Device Type. Step 2 will narrow down the sets for each dimension to {X}, {A1, A3, A4}, and {Mobile, Tablet}. Step 3 will then use the surprise metric to pick the Device Type dimension and its set {Mobile, Tablet} as the best explanation of the drop.

Our algorithm use a *per-element* threshold of the change in the measure, T_{EEP}, to add to the idea of succinctness. Not only do we want the smallest set of elements, we also want only those elements that contribute *at least* a fraction of T_{EEP} to the anomaly.

We show in Section 3.4 that solving this problem can take exponential time (in number of elements) in the worst case. Therefore, we use a greedy approach that solves this problem approximately.

3 Root-Cause Identification Algorithm

We start with some notation and use it to formally define explanatory power and surprise. We then describe the root-cause identification algorithm. While the algorithm remains the same for fundamental and derived measures, the way explanatory power and surprise are computed for derived measures is more complex and is discussed separately in Section 4.

3.1 Notation

The list of important terms used in this section and their notation are summarized in Table 5. Let the set of measures (e.g., revenue, number of searches)

Term	Notation	Example
Dimensions	$D = \{D_1, D_2, ..., D_n\}$	{Advertiser,Data center,...}
Cardinality of Dimension D_i	C_i	1000's for advertiser, 10's for Data center,...
Elements of Dimension D_i	$E_i = \{E_{i1}, E_{i2}, ..., E_{iC_i}\}$	{Flower123,...} for Advertisers
Measures	$M = \{m_1, m_2, ..., m_k\}$	{Revenue, Number of Searches,...}
Forecasted and Actual Values of measure m for element E_{ij}	$F_{ij}(m)$, $A_{ij}(m)$	Revenue for Flowers123: forecast = $100, actual = $90
Overall forecasted and actual values of measure m	$F(m)$, $A(m)$	Total revenue: forecast = $1,000,000 and actual = $900,000

Table 5: **Notation**

be denoted as $M = \{m_1, m_2, ..., m_k\}$ and let the set of dimensions (e.g., advertisers, data centers) be $D = \{D_1, D_2, ..., D_n\}$. Further, let the set of elements of a given dimension D_i be denoted as $E_i = \{E_{i1}, E_{i2}, ..., E_{iC_i}\}$ where C_i is the cardinality of dimension i. For example, E_{21} may be "Flowers123", an element of the advertiser dimension.

For each of the measures $m \in M$ of interest (including the fundamental and derived measures) and for each of the elements E_{ij}, we have access to the predicted or forecasted values, F_{ij}, as well as the actual observed values, A_{ij}. Note that, as discussed earlier, these values are aggregates corresponding to some time interval of interest (e.g., $100 revenue forecast, $90 revenue actual for element Flower123, dimension advertiser).

For fundamental measures such as revenue or number of searches, both the overall forecasted value for the measure, $F(m)$, as well as the overall actual value, $A(m)$, remain identical across all the dimensions (e.g., $100 forecasted and $50 actual revenue in the example in the previous section). For fundamental measures, the overall measure is simply the summation of value of the measures of the elements of the respective dimensions, but the same is not true for derived measures as they are not additive (Section 4).

Thus, given $F(m)$ and $A(m)$, the algorithm needs to output a potential root cause to explain the difference between the two. For this, it uses explanatory power and surprise, defined next.

3.2 Explanatory power

Explanatory power of an element can be defined as the percentage of change in the overall value of the measure that is explained by change in the given element's value. For fundamental measures, the explanatory power of an element j in dimension i is simply

$$EP_{ij} = (A_{ij}(m) - F_{ij})/(A(m) - F(m)) \quad (4)$$

For example, the total number of searches at a given hour deviates from a forecasted value of 1 million to 0.8 million, and the number of searches at the same hour at a particular data center, DC1, differs from its forecasted value of 0.5 million to 0.4 million, the explanatory power for element DC1 is (0.4-0.5)/(0.8-1) = 50%.

Note that, explanatory power for an element can be more than 100% or even negative, if the change in element is in opposite direction to overall change. However, the sum of explanatory powers of all elements of any dimension should sum up to 100%. Thus, explanatory power fully explains the change in the overall measure.

3.3 Surprise

As discussed in the example in Section 2, a dimension that has large change in its distribution (e.g., Device Type) is more likely to be a root-cause than the dimension that does not exhibit such a change (e.g., Data Center). We now formally define a measure of surprise to capture this notion.

For each element E_{ij}, let $p_{ij}(m)$ be the forecasted or prior probability value given by

$$p_{ij}(m) = F_{ij}(m)/F(m), \forall E_{ij} \quad (5)$$

Given a new anomalous observation, let $q_{ij}(m)$ be the actual or posterior probability value

$$q_{ij}(m) = A_{ij}(m)/A(m), \forall E_{ij} \quad (6)$$

Intuitively, the new observations for a given dimension are surprising if the posterior probability distribution is significantly different from the prior probability distribution. This difference between two probability distributions P and Q can be captured by the relative entropy or Kullback-Leibler (KL) divergence [12]. However, the use of KL divergence in our context has two issues. First, KL divergence is not symmetric. Second, KL divergence is only defined if, for all i, $q_i = 0$ only if $p_i = 0$, which does not hold in our setting (e.g., advertiser pauses his campaign).

Thus, instead of KL Divergence, we use a related measure called the Jensen-Shannon (JS) divergence [12] for computing surprise, defined as

$$D_{JS}(P,Q) = 0.5(\Sigma_i p_i \, log\frac{2p_i}{p_i+q_i} + \Sigma_i q_i \, log\frac{2q_i}{p_i+q_i})$$

Observe that $D_{JS}(P,Q)$ is symmetric and is finite even when $q_i = 0$ and/or $p_i = 0$. Further, $0 \leq D_{JS}(P,Q) \leq 1$, where 0 denotes no change in distribution between P and Q, with higher values denoting greater differences.

Thus, to compute surprise S_{ij} for element E_{ij}, we use $p = p_{ij}(m)$ and $q = q_{ij}(m)$ to compute

$$S_{ij}(m) = 0.5\,(p\,log(\frac{2p}{p+q}) + q\,log(\frac{2q}{p+q})) \quad (7)$$

3.4 Algorithm

```
1  Foreach m ∈ M // Compute surprise for all measures
2      Foreach E_ij // all elements, all dimensions
3              p = F_ij(m)/F(m) // Equation 5
4              q = A_ij(m)/A(m) // Equation 6
5              S_ij(m) = D_JS(p, q) // Equation 7
6  ExplanatorySet = {}
7  Foreach i ∈ D
8      SortedE = E_i.SortDescend(S_ij(m)) //Surprise
9      Candidate = {}, Explains = 0, Surprise = 0
10     Foreach E_ij ∈ SortedE
11         EP = (A_ij(m) − F_ij(m))/(A(m) − F(m))
12         if (EP > T_EEP) // Occam's razor
13             Candidate.Add += E_ij
14             Surprise += S_ij(m)
15             Explains += EP
16         if (Explains > T_EP) // explanatory power
17             Candidate.Surprise = Surprise
18             ExplanatorySet += Candidate
19             break
20 //Sort Explanatoryset by Candidate.Surprise
21 Final = ExplanatorySet.SortDescend(Surprise)
22 Return Final.Take(3) // Top 3 most surprising
```

Figure 2: **Root-Cause Identification Algorithm**

The root-cause identification algorithm seeks to solve the optimization problem specified in Section 2 using the above definitions of explanatory power and surprise.

Note that, obtaining the optimal solution to the problem in the worst case will take exponential time. This can be shown through a simple example: consider a set of size n where each element has an identical explanatory power and we require $n/2$ elements of the set to explain T_{EP}. In this case, every possible subset of cardinality $n/2$ is of minimum size possible (succinct) and has explanatory power of T_{EP}. Thus, we have to compare the surprise values of all these subsets (whose count is exponential in n) in order to find the subset that has the maximum surprise, the optimal solution.

Instead of enumerating various minimum cardinality subsets that have explanatory power of at least T_{EP} , our algorithm (Figure 2) uses the following greedy heuristic. In each dimension, after computing the surprise for all elements (lines 1–5), it first sorts the elements in descending order of surprise (line 8). It then adds each element to a candidate set as long as the element explains at least T_{EEP} of the total anomalous change by itself (lines 12–15). The parameter T_{EEP} helps control the cardinality of the set *(Occam's razor)*. For example, if T_{EEP} is 10% and T_{EP} is 67%, we can have at most 7 elements that explain anomalous change. Further, by examining elements in descending order of surprise, we greedily seek to maximize the surprise of the candidate set. The algorithm adds at most one candidate set per dimension (lines 16–19), as long as the set is able to explain a majority (T_{EP}) of the anomalous change *(explanatory power)*. Finally, the algorithm sorts the various candidate sets by their surprise value and returns the *top three most surprising* candidate sets as potential root-cause candidates (lines 21–22).

4 Derived Measures

Derived measures are functions of fundamental measures that are tracked by troubleshooters since they reveal more information than if one simply tracked the fundamental measures. In this section, we discuss how we compute explanatory power and surprise for derived measures.

4.1 Explanatory Power

While attributing contribution of an individual element to the overall value of a derived measure is important for root-cause identification, this is not as straightforward as computing the same for fundamental measures. In this section, we first start with a illustrative example that helps define explanatory power for derived measures and then present our solution to the derived measure attribution problem.

Example. Consider the hypothetical example in Tables 6 and 7 that shows revenue and number of clicks, respectively, for four different advertisers during an anomalous period. For these two fundamental measures, attribution of the overall change to each of the advertisers is simple using the explanatory power (equation 4) and is shown in the column labelled EP. Thus, for the revenue drop, one can attribute it to advertiser A1 (400%) while for the increase in clicks, one can attribute it to advertiser A2 (200%).

Let us assume that an anomaly is thrown on a measure if it differs from its expected value by at least 20%. Note that the overall revenue has gone down by 10% while the number of clicks is up 16%, neither of which exceeds the anomalous threshold. The corresponding cost-per-click values are shown in Table 8 and using the same 20% threshold, the overall cost-per-click (22.5% decrease) can be labelled anomalous Thus, one can see that derived measures can be useful in surfacing anomalies that are not surfaced by just examining fundamental measures. We confirm this quantitatively in Section 6.

The derived measure attribution problem is the following: how does one attribute the *drop* in overall cost-per-click from 0.2 (expected) to 0.155 (actual) to each of the advertisers? If one examines the individual cost-per-clicks of the advertisers in Table 8, we see that cost-per-click for advertisers A1, A2, A4, are *unchanged* while the cost-per-click for advertiser A3 has *increased*. Thus,

Adver-tiser	Forecasted Revenue	Actual Revenue	EP %
Overall	100	90	-10
A1	50	10	400
A2	0	0	0
A3	40	70	-300
A4	10	10	0

Table 6: **Revenue**

Adver-tiser	Forecasted Clicks	Actual Clicks	EP %
Overall	500	580	16
A1	100	20	-100
A2	200	360	200
A3	100	100	0
A4	100	100	0

Table 7: **Clicks**

Adver-tiser	Forecasted Cost/Click	Actual Cost/Click	EP %
Overall	0.2	0.155	-22.5
A1	0.5	0.5	125
A2	0	0	106
A3	0.4	0.7	-131
A4	0.1	0.1	0

Table 8: **Cost-per-click**

at first glance, it appears that none of the advertisers can be blamed for the overall drop but surely one or more of them must be responsible! Given this situation, how do we go about assigning explanatory power values for the change in cost-per-click to these advertisers?

Examining the fundamental measures does help shed more light. For example, even though cost-per-click of A1 is unchanged, A1 had a 5X drop compared to its forecasted values for both revenue and clicks. Given A1's cost-per-click (0.5) was higher than the overall value (0.2), the 5X reduction implies that A1 was indeed pulling down the overall cost-per-click. The fact that A1 explains some of the decrease in the overall derived measure can be further validated by observing that if we used A1's actual values but assume that the rest of the advertisers delivered their respective forecasted values, then the overall cost-per-click goes down to 60/420 = 0.143 for an impact of -29%.

Similarly, while A2 had 0 revenue as forecasted, A2 had a large increase in clicks, which ends up decreasing the overall cost-per-click. Again, if we used A2's actual values but keep the rest of the advertisers' measures to their forecasted value, the overall cost-per-click goes down to 100/660 = 0.152 for an impact of -24%. The above exercise of changing one advertiser's value at a time also suggests that A1 was more responsible for pulling down overall cost-per-click than A2 (since use of A1's actual values resulted in lower overall value than for A2).

Now consider A3. A3 had a higher revenue than forecasted without change in clicks, so A3 was clearly not contributing to the overall drop. Using A3's actual values in the above exercise would in fact increase the overall cost-per-click to 0.26, for an impact of +30%.

Finally, A4 had no change in either revenue or clicks. Therefore, A4 had no impact in overall cost-per-click.

Normalizing the individual impact values so that all the elements in total explain 100% of the overall change, the above exercise would give A1's explanatory power as 125%, A2's as 106%, A3's as -131% and A4's as 0%.

Summarizing the observations in the above example, one can see that an element's explanatory power for derived measures can be determined by computing a new derived measure value, where the actual value of the given element and forecasted values of all other elements are used, and comparing this derived measure value to the expected value of the derived measure.

Now, the question is how do we formalize this intuition in order to determine the explanatory power for arbitrary derived measures? We describe this next.

Derived measure attribution. Our solution to the derived measure attribution problem is adapted from *partial derivatives and finite-difference calculus*. Recall that a partial derivative is a measure of how a function of several variables changes when one of its variable changes. However, since we operate in the discrete domain, we use partial derivative equivalents from finite-difference calculus [13].

We formally define explanatory power of an element i for a derived measure, which is function $h(m_1,...,m_k)$ of fundamental measures $m_1, ..., m_k$, as the partial derivative with respect to i in finite-differences of $h(.)$, normalized so that the value across all elements of the dimension sum up to 100%.

While the above definition is general and applicable to derived measures that are arbitrary functions of fundamental measures (as long as they are differentiable in finite-differences), we now illustrate it through the specific example of derived functions of the form $A(m_1)/A(m_2)$, which make up many of the derived measures in ad systems (Figure 1). For example, for the cost-per-click derived measure, we have m_1 = revenue and m_2 = clicks.

The partial derivative in finite-differences of f(.)/g(.) is of the form $(\Delta f * g - \Delta g * f)/(g * (g + \Delta g))$, and is similar to continuous domain partial derivative, except for the extra Δg in the denominator.

Thus, explanatory power of element j for dimension i for derived measures of the form m_1/m_2 is given by

$$\begin{aligned} EP_{ij} \quad = \quad & ((A_{ij}(m_1) - F_{ij}(m_1)) * F(m_2) \\ & -(A_{ij}(m_2) - F_{ij}(m_2)) * F(m_1)) \\ & /(F(m_2) * \\ & (F(m_2) + A_{ij}(m_2) - F_{ij}(m_2))) \end{aligned} \quad (8)$$

We compute EP_{ij} for each of the elements using the above equation and normalize it so that they add up to 100%.

Table 8 shows the explanatory power computed using the above formula for each of the advertisers. We can see that the rank ordering of A1, A2, A4, and A3 and their respective explanatory power values for the attribution to the overall change agrees with the intuitive observations made earlier.

4.2 Surprise

Recall that we defined surprise for fundamental measures in Section 3.3 based on the relative entropy (specifically, JS divergence) between the prior and posterior mass functions of values for measure m. In this section, we seek to extend the notion of surprise to derived functions of multiple measures.

Consider the cost-per-click example in the previous section. A simple approach for computing surprise for derived measure is as follows. Just as for fundamental measures, one could compute prior and posterior probability values for cost-per-click for each element E_{ij}, say p_{ij}(cost-per-click) and q_{ij}(cost-per-click) and compute the surprise just as in Section 3.3.

However, such an approach will not work. Consider the example of advertiser A2 in Table 8. A2's cost-per-click was forecasted to be zero and the actual value was also 0. Thus, if one used the above approach to compute surprise for element A2, it would have a value of 0 (no surprise). However, we found that A2 had a high explanatory power of 106% for the overall change in cost-per-click due to changes in A2's number of clicks.

Examining the problem from the perspective of relative-entropy, given several measures, we first need to compute the joint probability distribution of the measures and then compute relative entropy of the joint probability distribution function. If the measures are independent, then the relative entropy (JS divergence as well) of the joint probability distribution is simply the sum of the relative entropy of the individual measure's probability distributions. In ad systems, the measures are not always strictly independent since some of them can be correlated (e.g., as the number of searches increase, revenue can be expected to increase). However, as an approximation, we assume that measures are independent, and compute the *surprise for derived measures as the summation of the surprise of the individual measures that are part of the derived function*.

5 Implementation and Experience

In this section, we describe our implementation of the above algorithms in the Adtributor tool and outline our experience with a pilot deployment in a production ad system.

5.1 Implementation

In our implementation, a database records, in real-time, counters for all measures, dimensions, and elements and exposes them as an OLAP service that supports multi-dimensional analytical queries [19]. When the system triggers an anomalous event, the Adtributor toolchain first gathers data relevant to the anomaly such as time of anomaly, measure, data for various measures, dimensions and elements. After the data has been queried from the database, Adtributor employs the root-cause algorithm to discover potential root-causes for the given anomaly.

Figure 3: **An example output of the Adtributor. Note: certain sensitive fields are masked.**

Recall that measures are not necessarily independent of each other. An anomaly on a certain measure could be correlated with changes in value of another. Therefore, we build a *dependency graph* of measures, and for a given anomalous measure, run the root-causing algorithm for every measure that correlates with it.

Adtributor filters the candidate set of root-causes (as described in Section 3) to produce the final list of root-causes. We use a T_{EP} value of 67% and a T_{EEP} value of 10%. These threshold values are driven by what the troubleshooters already use in the manual process. Also, our current implementation singles out a list of the top three dimensions. The troubleshooting experts recommended this number based on their own requirements and also on ease of visualization. With a smaller number they could miss useful information, while a larger number would lead to too much information for them to sift through.

The final output is a self-contained HTML5 application. Figure 3 shows an example of the output produced by the Adtributor toolchain. The visualization of the root-causes contains the following information:

- **Dependency Graph**: A graphical representation of dependencies between the different measures in the system (left half).
- **Measure Historical Graph**: A graph depicting the historical behavior of a measure (top right graph).
- **Element Root-Causes and Historical Graph**: For a given measure and dimension, the top elements that are root-causes. The element root-causes are grouped by dimensions with their historical graphs (under top right graph).

We arrived at the visualization requirements through iterative discussions with the troubleshooting experts. The dependency graph allows them to observe causality between the values of different measures, and the histori-

cal graphs per-dimension help them in making a more informed choice on what exactly was the root-cause. The entire Adtributor toolchain is implemented using .NET Framework 4 using 12,500 lines of code and executed automatically for each anomaly.

5.2 Deployment Experience

We conducted a pilot deployment of Adtributor between May 1, 2013 and May 10, 2013 with the troubleshooters who work with the production system on root-causing anomalies to understand the usefulness of Adtributor. This deployment was partially successful in helping the troubleshooters with their current processes. The findings of this pilot resulted in a set of improvements to our algorithm and visualization which led to significantly better performance as we show in our evaluation in Section 6.

Volatile dimensions: Various dimensions can be extremely volatile, and unexpected changes can occur in measures along these axes even though they are not necessarily the root-cause of the problem. Consider the example of an advertiser who frequently changes the budget allotment to their ads. When there is a revenue anomaly, this can sometimes cause the root-causing algorithm to pick the advertiser as a culprit even though the change coincidentally occurred just a little before the anomaly event. This drove us to improve our prediction algorithm for measures associated with elements of volatile dimensions by increasing the weightage given to large changes in the near-past in our prediction model, thereby fixing this problem to a large extent.

Visualization enhancements: The dependency graph of related measures was found to be very useful by the troubleshooters. However, the current view in the tool is limited to a small set of measures. There are hundreds of other measures being monitored within the ad system for which the dependencies are not known. We have therefore used a Bayesian structure learning algorithm [5] to infer a subset of these dependencies and plan to enhance the visualization of the dependency graph with these additional measures.

6 Evaluation

In the Section, we first describe four case studies in which multi-dimensional analysis is key to arriving at the final root-cause. Next, we provide a quantified evaluation of the accuracy of Adtributor, and the time savings we achieve with the tool.

6.1 Case Studies

Case 1: This was triggered by an anomalous drop in revenue. On performing the multi-dimensional analysis, we found that the dimension *Browser* was responsible. Figure 4 helps explain how Adtributor arrived at this result. It shows the percentage contribution to revenue along three dimensions – *Browser*, *Data Center*, and *Bucket* – for predicted revenue and actual revenue (see Table 1, example 3 for the definition of a bucket.). Notice that Browser 3's revenue contribution was predicted to be 12%, but its actual revenue was 0%! Similarly, Browser 1's contribution was predicted to be 60%, but was actually much higher at 74%. Neither the Data Center dimension nor the Bucket dimension show such surprising changes in contribution. This problem was actionable, since a further investigation revealed that a configuration error had caused no ads to be shown to users on Browser 3. Correcting the error fixed the problem and further loss in revenue.

Case 2: We noted an anomalous revenue increase at a particular time, which Adtributor attributed to a certain set of six advertisers. Two of these advertisers were airline ticket vendors, two were car rental agencies, and the remaining two were hotels. In aggregate, they fully explained the change in revenue. Delving into the issue, we noticed that these advertisers had deliberately increased their budgets for a certain period of time. The ads were appearing in a geographic region which had a long-weekend holiday approaching. Thus, we inferred that the advertisers were trying to capitalize and capture the attention of users as they performed vacation-related searches. Clearly, in this case, the sudden rise in revenue was attributed to advertiser behavior and not due to an actionable bug in the system. Several such anomalies also occur when advertisers deliberately drop their budgets as well.

Case 3: The total number of searches went anomalously high, and an analysis showed that most of the increase was attributed along the *User-agent string* dimension. From post-processing on this result, it was inferred that a majority of the searches with the repeated user-agent string were coming from a small range of IP addresses, and therefore, suspiciously characteristic of bot-traffic. In particular, the goal of this bot was to perform queries and collect information for search-engine optimization (SEO). This was an actionable issue which was fixed promptly by filtering the contribution of this traffic to the various metrics.

Case 4: We notice that sometimes, publishers change the placement of advertisements on their page, which make ads more (or less) conspicuous. This in turn causes a corresponding increase or decrease in revenue. These show up as revenue changes along the dimension of *Ad Position* on the page. For instance, if the publisher moves an ad meant to be shown on the side of the page to the top of the page, this presents itself as a surprising increase in revenue attributed to ads shown on the top. This is because users tend to click more on ads shown on the top of a page than they do on ads shown on the side.

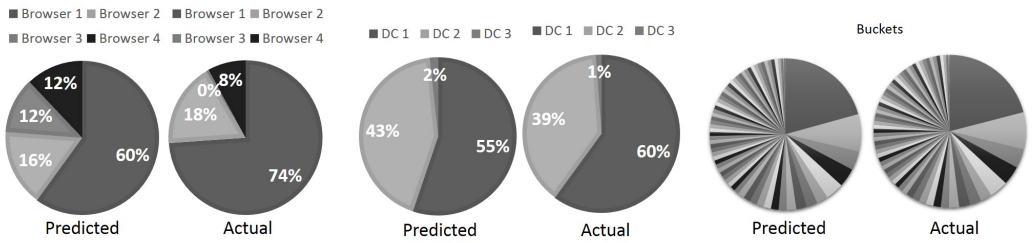

Figure 4: **Predicted and actual revenues for the Browser, Data center, and Bucket dimensions (Case Study 1).**

6.2 Comparative Study

Our quantitative evaluation is based on using Adtributor to root-cause problems in a widely deployed ad system. We evaluate all anomalies generated on a total of 12 measures, both fundamental and derived, across 33 dimensions. The results we present here use a subset of 128 valid anomalies generated by over a billion searches between September 1, 2013 and September 15, 2013 across 8 populations: PC and Mobile ad systems for USA, UK, France and Germany. For the purpose of this study, we do not consider false-positives in the anomaly generation process as they are weeded out by troubleshooters before applying the root-causing process[2]. 50% of the tested anomalies were generated solely on derived measures, with no related anomalies being generated on the respective fundamental measures that constitute the derived measure. This shows that using derived measures in aggregate root-cause analysis is extremely important.

We compare the output of Adtributor's multi-dimensional analysis with the output of the troubleshooting team that performs an in-depth and detailed analysis of these anomalies through manual means with the assistance of other tools (not Adtributor). Manually analyzing the cause of the anomalies has a number of advantages. The troubleshooters are aware of a large amount of information and domain-knowledge, and they frequently use this knowledge in the troubleshooting process. An automated tool such as Adtributor cannot possibly have an understanding of all of this. Further, Adtributor only narrows the scope of the root-cause (Column 4 of Table 1) – a manual process may still be necessary in many cases to identify the final root-cause (Column 4 of Table 1) since some of the data necessary to do this next step may not be available for the automated process (e.g., verifying whether the publisher indeed changed the position of ads).

However, the advantage of using Adtributor is that it aids the manual troubleshooting process by 1) using the multi-dimensional root-cause analysis to *exhaustively* check all possible dimensions (as we show, in a few cases, the manual process may overlook a dimension, leading to erroneous conclusions) and 2) Significantly faster processing to bubble up the top suspect candidates. For example, there are dozens of dimensions and some dimensions can have thousands of elements.

As described in Section 5, Adtributor displays the top three dimensions and their elements as potential suspects. We say that Adtributor matches the output of the manual root-causing process if it shows the same dimension and exactly the same elements as the manual process at any one of these three positions.

No. of anomalies	128
No. of matches	118 (1:81, 2:27,3:10)
Manual errors found	4
Adtributor's errors	5
Ambiguous	1
Adtributor accuracy	(118+4)/128=95.3%
Strawman accuracy (no surprise)	20.0%

Table 9: **Results summary from our comparison of Adtributor with manual scrutiny (and Strawman).**

Table 9 shows the results of the comparison between the output of Adtributor and the manual investigation. Of the 128 anomalies, Adtributor matched the results of the manual analysis in 118 cases. Of these, 81 (69%) matched in position 1, 27 (23%) matched only in position 2 and not in position 1, and 10 (8%) matched in position 3, and not in position 1 or 2. Of the 10 anomalies for which we did not match the manual output, we performed a deeper dive with the troubleshooting expert. On careful scrutiny, we found that out of the 10, 4 of the manual root-causes were erroneous, and Adtributor's output in position 1 was, in fact, the correct root-cause. This shows the utility of using a systematic algorithm, as in Adtributor, that exhaustively searching all dimensions to perform multi-dimensional root-cause analysis.

Out of the remaining 6 anomalies, the manual output was correct in 5 of them while the output of Adtributor was erroneous. In all of these cases, Adtributor suffered from a lack of domain knowledge, or the lack of knowledge of events external to the system which the troubleshooters were explicitly aware of. In one case (labelled ambiguous), however, the troubleshooter felt the dimension and elements blamed by Adtributor was

[2]Evaluating the number of false-positives and negatives would be to evaluate the anomaly detection algorithm which, as mentioned earlier, is out of scope of this paper.

as likely to be the true root-cause as the one obtained through manual analysis. In this case, he felt a further drill-down would be required to determine the correct root-cause. Taking the manual errors into account, Adtributor's *overall accuracy was (118+4)/128, or 95%*.

We also compare the potential time that could be saved using Adtributor compared to the first step in the manual troubleshooting process that identifies the dimension and elements that may be potential root-causes. Adtributor uses a multi-threaded implementation and caching to speed up the process of studying every dimension and every measure. It has a turnaround of approximately 3-5 minutes for each anomaly. The manual process of troubleshooting took between 13 minutes for the fastest anomaly to up to 231 minutes, with an average turnaround time of 73 minutes. Therefore, we conclude that Adtributor *speeds up the initial root-causing process by an order of magnitude*.

Finally, we show the value of using surprise by comparing our algorithm to the Strawman discussed in Section 2 that only uses succinctness and explanatory power. Compared to Adtributor's accuracy of 95%, we found that Strawman had an accuracy of only 20%. This clearly demonstrates the value of using surprise to identify the right dimension and elements as the root-cause.

7 Applicability beyond Ad Systems

We believe that the techniques introduced in this paper are general enough to be useful in other settings. For example,

Multi-dimensional analysis: Consider a web-server with a global audience that suddenly sees the number of hits drop sharply. Many of the dimensions considered in this paper such as data centers or CDNs, browsers, user locations, fraud operators/bots, etc. may all be potential root-causes that a multi-dimensional analysis can help disambiguate.

Derived measure attribution: Consider the following problem. The Mean-opinion-score (MOS) for VoIP calls has dropped and the investigators would like to understand which of the links in the route of the call is most responsible for this drop. Each link may have different amounts of delay, jitter, and loss percentages, and the MOS is a complex function of measures such as delay, loss, and jitter [6]. The use of the derived attribution technique can help compute the explanatory power of the drop in MOS for each of links.

8 Related Work

System and Network Root-Cause Analysis: Previous research has extensively studied root-causing performance and failure problems in systems and networks [14, 2, 10, 1, 21, 15, 3, 20, 11, 18]. Some of these use traces across individual requests through systems [3, 14] to diagnose problems, while others use aggregate counters of system performance or configuration values [15, 10, 20] to diagnose problems.

Distalyzer [14] is an example of the former category. It uses individual event logs and learns anomalous patterns between events that indicate a performance problem in a system component. Ganesha [15] is an example of the latter. It uses clustering approaches across aggregate measures, such as CPU usage, to build distinct profiles of MapReduce nodes. While our approach too uses aggregate measures, we intend to find more than performance problems or diagnose failures.

Q-Score [18] uses machine-learning to arrive at root-causes. We tried similar approaches and decided against them because selecting the right set of features to input to a stock machine-learning algorithm turned out to be a non-trivial task. Instead, we found that building a customized algorithm was simpler and better suited to analysis and feedback by our domain experts.

SCORE [11] localizes IP faults to underlying components using succinctness of explanation. Given a set of link failures as observation, it determines the smallest set of risk groups that explain failures. However, as we show, this approach is not enough to perform attribution across dimensions and a notion of surprise is essential to complete our solution.

Data Mining for Summarization: Previous work in data mining [17, 16, 7] has concentrated on summarizing multi-dimensional data in OLAP products. The objective is to provide an easily interpretable summary of the differences in data values across multiple dimensions. Such summarization techniques have been applied to network traffic summarization as well [9]. While data summarization across multiple dimensions is related to our work, it does not match our objective of finding surprising changes to perform root-cause analysis. In fact, our approach to root-cause analysis is complementary to these approaches and can be applied on the summaries that they generate.

9 Conclusion

We have described an algorithm, implementation, and evaluation of an approach that uses multi-dimensional analysis for root-causing problems in large-scale ad systems. We found that our approach has high accuracy (95%), helped identify more accurate root causes than the manual investigation in a few cases, and was able to reduce troubleshooting time significantly.

10 Acknowledgments

We would like to thank our shepherd VYAS SEKAR for his valuable comments and suggestions. We would also like to thank MURALI KRISHNA for helping us validate the output of Adtributor and determine its accuracy.

References

[1] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. Padmanabhan, and G. Voelker. NetPrints: Diagnosing Home Network Misconfigurations using Shared Knowledge. In *NSDI*, 2009.

[2] P. Bahl, R. Chandra, A. Greenberg, S. Kandula, D. A. Maltz, and M. Zhang. Towards Highly Reliable Enterprise Network Services Via Inference of Multi-level Dependencies. In *SIGCOMM*, 2007.

[3] P. Barham, A. Donnelly, R. Isaacs, and R. Mortier. Using Magpie for request extraction and workload modelling. In *Proceedings of USENIX OSDI*, 2004.

[4] G. Box, G. M. Jenkins, and C. Gregory. *Time Series Analysis: Forecasting and Control*. Prentice-Hall, 1994.

[5] D. M. Chickering. The winmine toolkit. Technical Report MSR-TR-2002-103, Microsoft, Redmond, WA, 2002.

[6] R. Cole and J. Rosenbluth. Voice over IP performance monitoring. *CCR*, Apr 2001.

[7] G. Cormode, F. Korn, S. Muthukrishnan, and D. Srivastava. Diamond in the rough: Finding hierarchical heavy hitters in multi-dimensional data. In *Proceedings of ACM SIGMOD*, 2004.

[8] V. Dave, S. Guha, and Y. Zhang. Measuring and fingerprinting click-spam in ad networks. In *Proceedings of ACM SIGCOMM*, 2012.

[9] C. Estan, S. Savage, and G. Varghese. Mining anomalies using traffic feature distributions. In *Proceedings of ACM SIGCOMM*, 2003.

[10] S. Kandula, R. Mahajan, P. Verkaik, S. Agarwal, and J. Padhye. Detailed diagnosis in computer networks. In *Sigcomm*. ACM, 2010.

[11] R. R. Kompella, J. Yates, A. Greenberg, and A. Snoeren. IP fault localization via risk modelling. In *Proceedings of USENIX NSDI*, 2005.

[12] J. Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, 1991.

[13] L. Milne-Thomson. *The calculus of Finite Differences*. Macmillan, 1933.

[14] K. Nagaraj, C. Killian, and J. Neville. Structured comparative analysis of systems logs to diagnose performance problems. In *Proceedings of USENIX NSDI*, 2012.

[15] X. Pan, J. Tan, S. Kavulya, R. Gandhi, and P. Narasimhan. Ganesha: blackBox diagnosis of MapReduce systems. *ACM SIGMETRICS Performance Evaluation Review*, 37(3):8–13, 2009.

[16] S. Sarawagi. Explaining differences in multidimensional aggregates. In *Proceedings of VLDB*, 1999.

[17] S. Sarawagi. iDiff: Informative Summarization of Differences in Multidimensional Aggregates. *Data Mining and Knowledge Discovery*, 5(4):255–276, 2001.

[18] H. H. Song, Z. Ge, A. Mahimkar, J. Wang, J. Yates, Y. Zhang, A. Basso, and M. Chen. Q-score: Proactive service quality assessment in a large IPTV system. In *Proceedings of ACM IMC*, 2011.

[19] E. Thomsen, G. Spofford, and D. Chase. *Microsoft OLAP solutions*. John Wiley & Sons, Inc., 1999.

[20] H. Wang, J. Platt, Y. Chen, R. Zhang, and Y. Wang. Automatic Misconfiguration Troubleshooting with PeerPressure. In *OSDI*, 2004.

[21] H. Yan, L. Breslau, D. Massey, D. Pei, and J. Yates. G-RCA: A Generic Root Cause Analysis Platform for Service Quality Management in Large IP Networks. In *Proceedings of ACM CoNext*, 2010.