Event Detection using Customer Care Calls

Yi-Chao Chen[1], Gene Moo Lee[1], Nick Duffield[2], Lili Qiu[1], Jia Wang[2]
The University of Texas at Austin[1], AT&T Labs – Research[2]

***Abstract*—Customer care calls serve as a direct channel for a service provider to learn feedbacks from their customers. They reveal details about the nature and impact of major events and problems observed by customers. By analyzing the customer care calls, a service provider can detect important events to speed up problem resolution. However, automating event detection based on customer care calls poses several significant challenges. First, the relationship between customers' calls and network events is blurred because customers respond to an event in different ways. Second, customer care calls can be labeled inconsistently across agents and across call centers, and a given event naturally give rise to calls spanning a number of categories. Third, many important events cannot be detected by looking at calls in one category. How to aggregate calls from different categories for event detection is important but challenging. Lastly, customer care call records have high dimensions (*e.g.*, thousands of categories in our dataset).**

In this paper, we propose a systematic method for detecting events in a major cellular network using customer care call data. It consists of three main components: (i) using a regression approach that exploits temporal stability and low-rank properties to automatically learn the relationship between customer calls and major events, (ii) reducing the number of unknowns by clustering call categories and using L_1 norm minimization to identify important categories, and (iii) employing multiple classifiers to enhance the robustness against noise and different response time. For the detected events, we leverage Twitter social media to summarize them and to locate the impacted regions. We show the effectiveness of our approach using data from a large cellular service provider in the US.

I. Introduction

Motivation: Customer care calls serve as a direct channel for a service provider to learn feedbacks from their customers. Customer care calls reveal important details about the nature, impact, and solution of major events and problems. By analyzing the customer care calls, the service provider can quickly detect important events so that they can (i) start and prioritize investigation of events to resolve the problems, (ii) deflect some of future customer care calls to reduce call volume by providing information about known events via alternative channels (*e.g.*, social media, web site, automated recorded messages), (iii) better manage work force in customer call centers to handle calls related to the detected events to reduce the call handling time and improve customer support.

Challenges: Automatically detecting major events in a cellular network using customer care calls poses the following significant challenges:

- Customers respond to an event in different ways. Some customers may not call at all, while others may call in but at different time depending on their impact, availability, and time-of-day. Therefore, the relationship between customers' calls and major events is blurred.
- Customer care call center records are labeled and input into system manually by call agents. Call agents are professionally trained to handle these customer calls, but may still face challenges in categorizing a call, depending on the information supplied by the customer, leading to inconsistent classification across agents and across call centers. Indeed a given event may naturally give rise to calls spanning a number categories. For example, the arrival of a new device may result in calls to pre-order, or to activate a new service, or to activate a new device.
- Many important events cannot be detected by looking at calls in one category. Aggregating calls from different categories is often necessary for event detection. However, the aggregation of categories should depend on the type of events, and the relationship between the categories and events also evolve over time as the call labeling process changes. Therefore it is important to automatically learn this relationship.
- There are a large number of possible categories by which the calls may be labeled. For example, our data set has 170 level-1 categories, 765 level-2 categories, and 2882 level-3 categories. Learning from such a large dimension of features is challenging.

Our approach: We first examine the problem by applying statistical methods and show that they alone are insufficient. Statistical techniques, such as EWMA, require a grouping of categories a priori and cannot automatically detect the grouping for a given type of events or anomalies, which we use inter-changeably. Principal Component Analysis (PCA) like approaches try to detect anomalies by projecting the categories into a lower dimensional subspace. However, PCA has several major weaknesses: (i) the orthogonal categories identified by PCA may not be the most relevant categories for the anomalies, (ii) PCA is sensitive to noise, and (iii) it is hard to determine the number of dimensions [17], [9], [8]. In general, they are not effective in automatically learning the relationship between anomalies and categories.

To address the issue, we develop a regression approach to learn the relationship between categories and anomalies using the training traces and use it to predict anomalies in the testing traces. It can be viewed as an inference problem $Ax = b$, where A represents the values of each categories, x denotes the weight (importance value) of the corresponding categories, and b is an indicator whether there is an anomaly. We find x to satisfy $Ax = b$ so that we can predict b in the future.

There are several significant challenges involved: (i) x may change over time, (ii) we may not have enough constraints to uniquely determine x, (iii) the extracted constraints may have significant noise and the best fitting to the constraints may not lead to accurate prediction for the future anomalies (*i.e.*, overfitting), (iv) the problem size is large and grows over time.

We address these challenges by developing a systematic

approach consisting of three main components: (i) a regression approach that leverages temporal stability and low rank properties of x, (ii) reducing the number of unknowns by clustering categories and identifying important clustered categories using L_1 norm minimization, and (iii) using multiple classifiers to enhance the robustness against noise and different customers' response time.

Even an event is detected with customer calls, it is hard to understand its nature and detailed impacted region because the customer call categories only have limited text information and the locations of calls can only be inferred by callers' area codes. To better understand the detected events, we leverage Twitter social media as a complementary source. With the augmented information from user feedbacks in Twitter, we summarize the events and localize the impacted regions.

Our main contributions can be summarized as follows:

- Understand the limitations of existing approaches.
- Develop a systematic approach that can effectively handle various practical issues, such as under-constraints linear systems, over-fitting, scalability, and varying response time.
- Evaluate the performance of our approach using the real traces from a large cellular provider in the US, and show the proposed approach can achieve 68% recall and 86% accuracy. The accuracy is limited in part by the fact that only a subset of data is available for our anomaly detection, while the ground truth is derived from more complete dataset. We expect the accuracy improves when more complete data is available.
- Conduct a feasibility study with social media as a complementary source to better understand the detected events.

II. Problem Formulation

Background: This study uses data derived from operational records of calls to customer support centers of a major mobile service provider. We now outline how these records are generated at the service center, and then describe a subset of the records that are used for this study.

When a customer calls the customer support, the call will first reach an Interactive Voice Response (IVR) system, an automated system configured with pre-defined menu. Based on the selected menu, the customer's call is either self-served or routed to one of the customer care call centers to be answered by an agent. Work force management for customer call centers are often performed according to the type of plan the customer has (*e.g.*, business or consumer, referred as "work group" in the paper) and what type of issues the customer has (*e.g.*, device, billing, performance issues).

Upon handling each customer care call, the call agent will open a case in the ticketing system, and label the case using a "three-level pre-defined categories" to indicate the customer's issue or need. Detailed notes are also input into the system based on the conversation with the customer. After the customer's need is satisfied and case is resolved, a "call resolution" code is also entered into the system. Although the detailed notes may provide more detailed information and help detect anomalies, they are used in this study due to the privacy issues and the challenge of using natural language to process them. Therefore, this paper focuses on using pre-defined categories of the calls to detect anomalies.

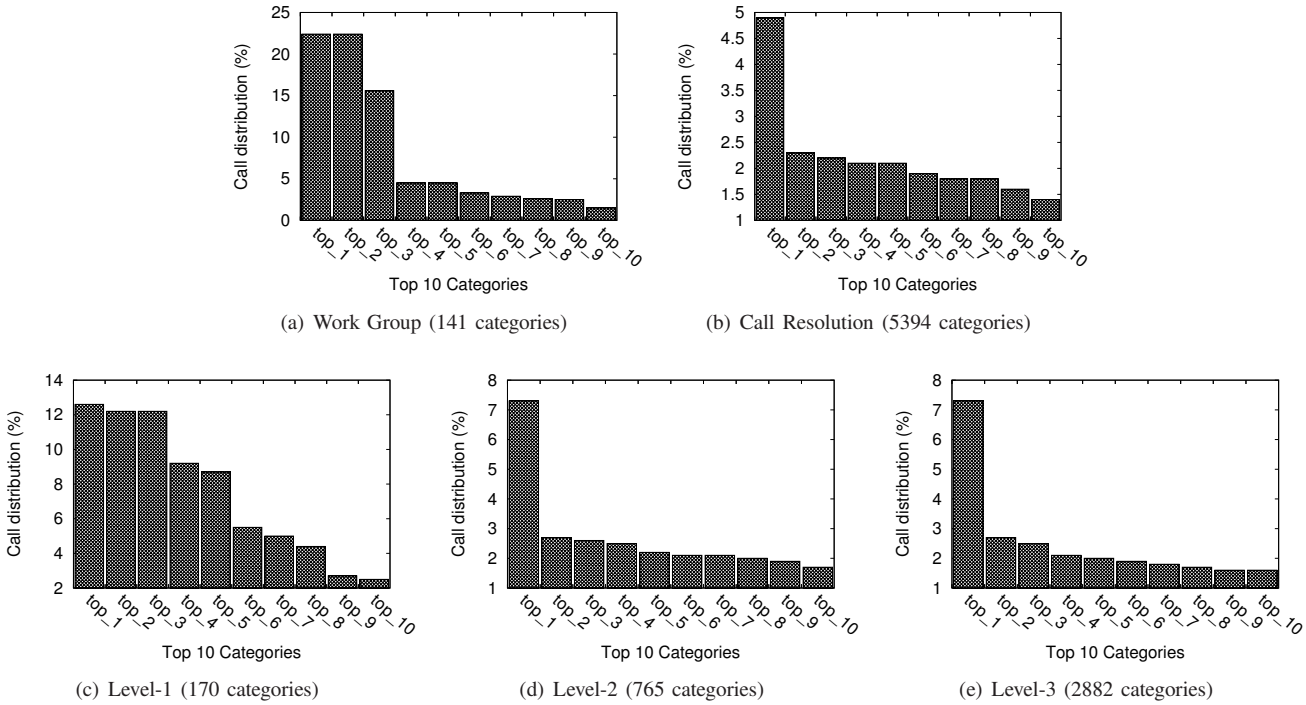
Customer care calls dataset: The data set for this study is derived from several million calls received at the service centers in 5 months during 2011. Each record in the dataset comprises the following subset of information related to a call: *work group*, *call resolution* (as described above) and the category ascribed to the call, comprising three levels customer need: *customer need level 1*, *customer need level 2*, and *customer need level 3*.

There are 141 work groups, 5394 call resolutions, 170 level-1 categories, 765 level-2 categories, and 2882 level-3 categories. Data do not contain any information concerning calls that did not progress beyond the IVR system. Figure 1 shows the normalized call distribution in the most popular 10 categories among *work group*, *call resolution* and 3 levels of *customer need*. We can observe that the top 10 categories account for 30-70% of calls. Moreover, top 10% categories account for 75-90% of calls. In spite of the significant amount of calls in top 10% categories, we cannot simply use these categories for anomaly detection because there are many anomalies dominated by the remaining 90% categories. For examples, "Technical" is a level-1 category which does not belong to the top 10% categories but it is one of the dominant factors in 73% of outage events observed from our dataset.

Real anomalies: In addition, we get ground truth from National Call Center Operations (NCCO) reports, in which anomalies are marked manually by monitoring the patterns of customer calls, activity of the IVR system, and network traffic. This process is extremely time-consuming and vulnerable to human errors, which motivates us to develop an automatic anomaly detector.

Table I shows an example of NCCO report. Each anomaly in NCCO reports contains the following information: an *event status* indicates the anomaly is first reported (*initial*), *updated*, or *resolved*; a *business unit* and *primary system* indicate which aspect of system is impacted by the anomaly; and a *region* indicates the impacted region. A *start time* labels the time when the anomaly is observed, and *resolved time* labels the time when the support team reports that the anomaly has been resolved. In this example, the anomaly is just reported, so the *resolved time* is unknown. There is also a description of the anomaly, *e.g.*, outage events or performance degradation. We designated the *start time* as the time at which an anomaly is detected. There could be a gap between the time that anomalies are observed in NCCO report and perceived by customers.

Since our approach uses only customer calls data to detect anomalies while the ground truth from NCCO reports is derived based on the more complete information (*e.g.*, activity of the IVR system and network traffic), we do not expect our approach can detect all anomalies.



(a) Work Group (141 categories)

(b) Call Resolution (5394 categories)

(c) Level-1 (170 categories)

(d) Level-2 (765 categories)

(e) Level-3 (2882 categories)

Fig. 1. The normalized number of calls in the most popular 10 categories among *work group*, *call resolution* and 3 levels of *customer need.*

Event Status: Initial **Business Unit(s)**: Mobility **Primary System**: Mobility: GSM Voice Networks **Region**: North East **Start Time**: month day year - time **Resolved Time**: Unknown **Issue Description**: Boston customers may experience no service or degraded service in the coverage area of the cell sites affected.

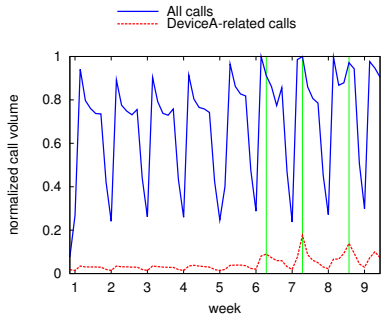
TABLE I
AN EXAMPLE OF NCCO REPORT

Issues: The call records give us information about calls in different categories with different metrics. Each category/metric gives us one timeseries. Our goal is to automatically detect anomalies or events using all the timeseries. A natural approach is to detect sudden changes in one or more timeseries. But finding an appropriate aggregation of the timeseries for accurate anomaly detection is challenging.

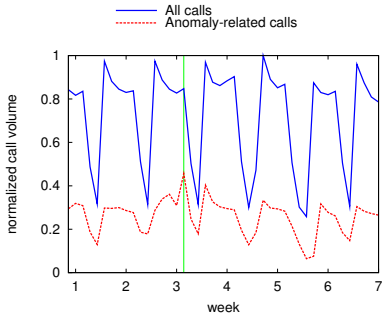
Simply aggregating all of them does not work well. Figure 2 show two examples. In the first example, there were 3 major events related to a release of new devices: *new device announcement*, *new device pre-order*, and *new device available*. If we consider all the customer care calls that are related to a new device, we can see clearly there are 3 spikes in the call volume corresponding to the above events. However, the events have little impact on the total call volume, and are difficult to detect using the total call volume. In the second example, 3G network outage occurred in South Florida on the second day of the third week. The anomaly can be detected using the weighted sum of call volume from categories "Technical", "Cannot Make or Receive Calls", "Voice", and "FLP" (Florida/Puerto Rico area), but not from the total call volume. Simply aggregating all calls is insufficient because (i) some events only have impact on a subset of customers and do not lead to significant changes in total volume, and (ii) even for the events that may potentially affect the total volume, the capacity of call centers limits the increase in total volume and makes it difficult to detect. Ideally in this case, we want to detect events before the capacity limit is reached so that we can increase call center capacity temporarily in response to the increased demand.

Another natural approach to detect anomalies is to use PCA. For example, [6] decomposes data into normal and abnormal subspaces using PCA by (i) applying PCA to the testing dataset and examining the projection of testing dataset on each principal axis in order, (ii) assigning the principal axis and all subsequent axes to anomalous subspace as soon as a projection is found to exceed a threshold (*e.g.*, 2x standard deviation from mean), and (iii) detecting spikes in the time series projected onto the anomalous subspace. We observe its precision (*i.e.*, fraction of predicted anomalies that are correct) is not much better than that of a random algorithm, which reports an anomaly by tossing a bias coin of 0.3 (*i.e.*, close to the fraction of time that has anomalies.) PCA performs poorly for several reasons. First, large anomalies can pollute the normal subspace [17] so PCA usually requires anomaly-free data for training [5]. This could be a problem for our dataset because there are average 1.8 anomalies per week in NCCO reports and it is hard to find a clean period for training. Second, determining the threshold for anomalous subspace is an open question [17], [8], [6]. Third, PCA is sensitive to noise, which is common in the customer care call dataset.

In general, the appropriate aggregation depends on the types of events. Due to a large number of possible types of events and evolving nature of events, it is infeasible to manually



(a) The three vertical lines indicate the time of 3 major events related to Device-A.



(b) An anomaly detected using the weighted sum of several categories

Fig. 2. The total call volume is insufficient to detect events.

determine the aggregation for each event type in advance. The above results call for a new method to automatically learn the mapping from the various input metrics to anomalies. The method should be (i) adaptive to new call labels and events, (ii) highly robust against noise, which is inherent in call data records due to different customers' responses to anomalies and inconsistent call labelings.

III. Our Approach

A. Problem Formulation

Our main problem can be formulated as follow. We have training and testing traces, where the training traces contain N-dimension input timeseries and the ground-truth about when anomalies take place and the testing traces contain new N-dimension input timeseries. Our goal is to determine when anomalies occur in the testing traces.

We use regression to approach the problem by casting it as an inference problem of $Ax = b$, where $A(t, i)$ denotes the number of calls of category i at time t, $x(i)$ denotes the weight (importance value) of the i-th category, $b(t)$ is an indicator whether there is an anomaly. We construct A and b from the training traces to solve for x. Then we plug in the estimated x and construct A from the testing traces to predict b for the testing trace. Essentially we view there exists a linear relationship between the categories values and the resulting anomalies, and we try to learn the linear coefficients x that automatically combines different metrics to predict the anomalies. As we will show in Section IV, a simple linear regression model works well, so we believe the linear assumption is reasonable. There are several significant challenges involved in realizing this scheme.

1. Dynamic x: As the categories and events evolve, the relationship between the inputs and the anomalies may also change. Therefore x can change over time.
2. Under-constraints: The number of categories can be much larger than the number of constraints derived from the training traces. So we have an under-constrained problem and there are an infinite number of solutions. Randomly picking one of them gives an equally good fit to the training data, but can give very different accuracy for the testing data. Our ultimate goal is to find x that can accurately predict the anomalies for the testing traces.
3. Over-fitting: Even if we are fortunate to get long enough training traces so that the number of categories is close to or smaller than the number of constraints, the weight estimation is still challenging because the solution that minimizes the fitting error based on the training traces is often not the one that gives the closest fit to the testing data. In other words, there can be over-fitting issues.
4. Scalability: There are thousands of categories or dimensions and thousands of time intervals. The scalability issue further exacerbates when we allow x to change. For example, if there are K different x's, the problem size further grows by a factor of K.
5. Varying customer response time: When an anomaly occurs, customers respond to it at different time depending on the impact of anomaly, the customers' own availability, time of day, and day of week. This blurs the relationship between A and b.

Below we first develop an approach to address the under-constraints and over-fitting issues while allowing x to change over time (Section III-B). Then we reduce the number of unknowns as well as handle the scalability issues by clustering categories and identifying important clustered categories (Section III-C). Finally, we use multiple classifiers to enhance the robustness against noise and different customers' response time (Section III-D).

B. Our Regression

To address the first challenge, we generalize our formulation to $A_d x_d = b_d$, where d denotes d-th day, $A_d(t, i)$ denote the value of i-th category from the traces at time t on the d-th day, $x_d(i)$ denote the weight of i-th variable on the d-th day, $b_d(t)$ denote whether there is an anomaly at time t on the d-th day, where 1 means anomaly and 0 means no anomaly.

To address the under-constraints and over-fitting issues, we cannot simply minimize the fitting error to the training data. Instead, we also impose additional structures on the solution. First, we expect the weight values x_d to be stable across consecutive days d. Second, we expect $x = [x_1 x_2 ... x_d]$ exhibits low-rank structure due to the temporal stability in x and the small number of dominant factors that cause the anomalies. Therefore, we try to find X, U, V that minimize the combined objective:

$$o(X, U, V) = \sum_d f(X) + \alpha \cdot g(X) + \beta \cdot h(X, U, V), \quad (1)$$

where $f(X)$ is the fitting error, $g(X)$ captures the degree of temporal stability, and $h(X, U, V)$ captures the error in approximating X as a product of two rank r matrices: U and V, where α and β give the relative weights of temporal stability and low-rank constraints, respectively, and r is the desired low rank. Next we elaborate on each term and how to select weights.

Fitting error: The fitting error is expressed as $f(X) = \sum_d \|A_d x_d - b_d\|_F^2$ where $\|\cdot\|_F$ is the Frobenius norm (with $\|Z\|_F = \sqrt{\sum_{ij} Z(i,j)^2}$ for any matrix Z.)

Incorporating temporal stability: To capture the temporal stability, we introduce a temporal transformation matrix T and define a penalty function as follows:

$$g(X) \triangleq \left\|M * T^T\right\|_F^2, \tag{2}$$

where $M = [x_1 x_2 ... x_d]$ merges all the column vectors into a matrix and T^T is the transpose of T. As in [29], we use a simple temporal transformation matrix to minimize the change in x between two consecutive days: $T = Toeplitz(0, 1, -1)$, which denotes the Toeplitz matrix with central diagonal given by 1, the first upper diagonal given by -1. That is,

$$T = \begin{bmatrix} 1 & -1 & 0 & \dots & \\ 0 & 1 & -1 & \ddots & \\ 0 & 0 & 1 & -1 & \ddots \\ \vdots & \ddots & \ddots & \ddots & \end{bmatrix}. \tag{3}$$

Incorporating low-rank constraints: Finally, to capture the low-rank nature of weight matrix, we introduce a penalty term function

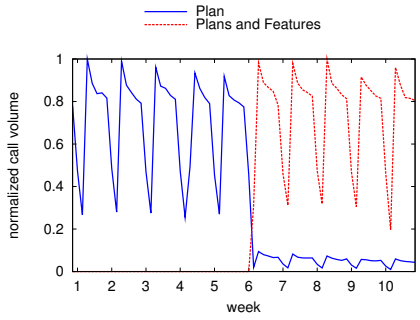
$$h(X, U, V) \triangleq \left\|M - U * V^T\right\|_F^2 \tag{4}$$

where $M = [x_1 x_2 ... x_d]$, U is a $N \times r$ unknown factor matrix, and V is a $d \times r$ unknown factor matrix, and r is the desired low rank. Minimizing the penalty term ensures M has a good rank-r approximation: $M \approx U * V^T$.

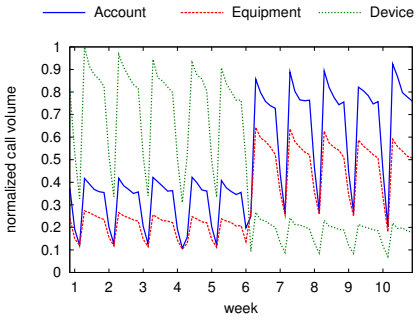
Selecting parameters: To decide the weights α and β, we use 6-week data as the training set to find the best weights (in terms of evaluation metrics discussed in Section IV) as follow. First, α and β are chosen to make fitting error, temporal stability, and low-rank constraint have similar order of magnitude. Second, we fix α, and keep increasing or decreasing β by a factor of 10 each time until the performance does not improve. β is updated as the one that gives the best performance seen so far. Third, similar to the second step, but this time we fix β and alter α. The second and third steps are repeated until the performance does not improve for the training traces. It usually takes 5 or fewer iterations. Then we apply the selected parameters to the testing trace for anomaly detection.

C. Reducing Categories

The regression problem described in Section III-B has thousands of variables for each day alone and the number further increases with the number of days. This imposes both scalability issue and exacerbates under-constraint issue.



(a) The number of calls labeled as "Plan" and "Plans and Features"



(b) The number of calls labeled as "Account", "Equipment", and "Device"

Fig. 3. Call centers change the way to label calls in the 2nd day of the 6th week.

1) Clustering Categories: To enhance scalability and minimize the impact of ambiguity and inconsistency in detecting events based on the customer needs categories, we need to cluster relevant categories in advance. It is infeasible to group categories manually because (i) the number of categories changes over time. Categories can be added or depreciated because of the emerging or ending of services/products; (ii) each customer care center can change the usage of categories; and (iii) there are too many categories.

One natural approach is to cluster categories base on the similarity between timeseries. However, the approach does not work. For example, Figure 3 shows timeseries of different categories. We can see that the call volume of category "Plans and Features" increases significantly while that of the other category "Plan" decreases on the same day, because the agents are asked to switch to use "Plans and Features" instead of "Plan". Similarly, we can see the call volume of "Device" decreases while that of "Account" and "Equipment" increases. Although these categories represent the same group of calls, their timeseries are not similar.

We cluster categories based on the similarity of their textual names since agents usually classify calls based on the textual names of categories and different agents may classify similar calls to different categories with similar texts. We treat each category name as a sequence of characters and adopt the Dice's coefficient [26] using bigram model [25], which is widely used in statistical natural language processing to measure string similarity. The Dice's coefficient s for two string x and y using the bigram model is computed as $s = \frac{2 \times n_t}{n_x + n_y}$, where n_t is the number of character the bigrams in both strings,

n_x and n_y are the numbers of big-rams in strings x and y, respectively. The value of s ranges between 0 and 1. A larger s indicates the two strings are more similar. For example, to calculate the similarity between strings "paid" and "payment", we find the sets of the bigrams as ("pa", "ai", "id") and ("pa", "ay", "ym", "me", "en", "nt"). These sets have 3 and 6 elements, while only 1 element is common. So we have $s = (2 \times 1)/(3 + 6) = 0.22$. By using a threshold of 0.3 for s, we cluster the customer categories into 96 at the first level, 354 at the second level, and 1165 at the third level. Our evaluate uses these newly computed categories.

2) Identifying Important Categories: Even after clustering, there are still a large number of clustered categories. We use the following three schemes to identify important categories.

Principal component analysis (PCA): PCA can also be used to identify important categories because it reduces dimensionality of a multivariate dataset by converting possibly correlated variables into linearly uncorrelated variables, which are called principal components. However, using PCA in this context has the same problem as it is used for anomaly detection mentioned in Section II. As a result, principal components may not identify the most important categories for anomaly detection. For example, the 3G network outage event shown in Figure 2 is dominated by categories "Technical", "Cannot Make or Receive Calls", "Voice", and "FLP". However, the PCA results show that in the top 10% principal components, the coefficients of "Cannot Make or Receive Calls" and "Voice" are small, which indicates they are not considered as important categories in PCA.

L_2 norm minimization: Another way of finding important categories is to cast it as an inference problem $Ax = b$, where $A(t, i)$ denotes the value of category i at time t, $x(i)$ denotes the weight (importance value) of the i-th category, $b(t)$ is an indicator whether there is an anomaly. Different from Section III-B, here we just need to filter out unimportant categories instead of determining the precise weights. So here we assume x is constant over time to have fewer unknowns.

We obtain A and b learned from the previous traces. Then we estimate x to best fit the relationship. A common metric for the best fit is L_2 *norm minimization*, defined as follows:

$$\min_x \|b - Ax\|_2^2 + \lambda^2 \|x\|_2^2, \tag{5}$$

where $\|x\|_2 = \sqrt{\sum_{k=1..n} \|x_k\|^2}$. It can be efficiently solved using a standard solver for linear least-squares problems. Then we filter out the categories whose weight x is within a threshold, which is set to 0 in our evaluation.

L_1 norm minimization: Another approach is to use L_1 *norm minimization*, defined as follow:

$$\min_x \|b - Ax\|_2^2 + \lambda^2 \|x\|_1. \tag{6}$$

where $\|x\|_1 = \sum_{k=1..n} \|x_k\|$. L_1 norm minimization is often used in situations where x is *sparse*, *i.e.*, x has only very few large elements and the other elements are all close to 0. This is well suited to our goal of identifying a small number of important factors. As shown in [2], the minimal L_1 norm solution often coincides with the sparsest solution for underdetermined linear systems. As we will show in Section IV, L_1 norm minimization performs the best since it explicitly maximizes sparsity. As before, we filter out the categories whose weight x is within a threshold, which is set to 0 in our evaluation.

D. Combining Multiple Classifiers

Need for multiple classifiers: Another important problem is what time scale we should use for anomaly detection. Ideally, we would like to capture all calls triggered by the same anomaly when learning the weight of the metrics. That is, A should include the characteristics of all calls corresponding to that anomaly. However, customers do not respond to an anomaly immediately and sometimes their response time may differ by hours. But simply using a large time window is not a good option since we can no longer detect anomalies in a fine time granularity.

To address both issues, we use a reasonably small bin size: 1 hour, but include calls made in previous n and next m hours as additional features. That is, we use $A_d(t - m, t - (m - 1), ..., t - 1, t, t + 1, ..., t + n)$, which denotes the values of N categories in the traces from time $t-m$ to time $t+n$. So there are altogether $(m + n + 1) \times N$ features and x_d also now has $(m+n+1) \times N$ elements, which are the weights of all these features. $b_d(t)$ remains the same as before (*i.e.*, whether there is an anomaly at time t).

However it is challenging to select m and n a priori. One set of values may work well on some data but not on others. Therefore we use multiple classifiers, where each classifier uses one set of m and n, and then we aggregate the results of all the classifiers. The intuition is that it is more likely to be a real anomaly if lots of classifiers claim so.

Aggregating multiple classifiers: We apply each classifier independently to the testing data and returns a binary time-series $pb(c, t)$. $pb(c, t) = 1$ denotes that there is an anomaly detected by classifier c at time t. $pb(c, t) = 0$ denotes there is no anomaly detected. We aggregate $pb(c, t)$ by assigning a weight w_c to each classifier. We detect an anomaly when $\sum_c w_c pb(c, t) > threshold$. Our evaluation uses a threshold of 0.3.

We calculate w_c for a classifier c by applying 2-fold cross-validation to the training data. The 2-fold cross-validation partitions the training data into two parts. In the first round, it uses the first partition for training and the second partition for testing. Since we know the ground truth in all training data, we can evaluate how the classifier performs in cross-validation by calculating the accuracy (*i.e.*, the fraction of correct prediction) in the second partition. Similarly, in the second round, we use the second partition of the training data for training, use the first partition for testing, and calculate the accuracy in the first partition. Therefore, with the 2-fold cross-validation we can get an average accuracy a_c in training set which gives us an estimate how the classifier may perform.

Then the weight of each classifier is assigned as the normalized accuracy: $w_c = a_c / \sum_c a_c$.

IV. Evaluation

We use the following metrics to quantify the accuracy:

$$recall = \frac{tp}{tp + fn} \quad (7)$$

$$precision = \frac{tp}{tp + fp} \quad (8)$$

where tp is the number of true positives (*i.e.*, correctly detected anomalies), fp is the number of false positives (*i.e.*, incorrectly detected anomalies), and fn is the number of false negatives (*i.e.*, missed anomalies). In addition, we integrate precision and recall into a single metric called *F-score* [27], which is the harmonic mean of precision and recall: *F-score* $= \frac{2}{1/precision + 1/recall}$. For all three metrics, larger values indicate higher accuracy. Unless otherwise specified, we use 30 classifiers.

Identification of important features: We first evaluate how different feature selection algorithms impact the performance. We vary the methods of identifying important categories while using multiple classifiers and temporal/low-rank based regression. We compare PCA, L_1 norm minimization, L_2 norm minimization, and random selection (*e.g.*, *Rand 1000* and *Rand 2000* randomly select 1000 and 2000 categories). L_1 norm selects 612 important categories and L_2 norm selects 980 important categories. As shown in Figure 4, L_1 norm consistently performs the best. It out-performs L_2 norm by 23%, PCA by 45%, Rand 2000 by 454% in terms of precision; out-performs L_2 norm by 10%, PCA by 32%, and *Rand 2000* by 1020% in terms of recall. *Rand 1000* selects too few categories and yields close to 0 precision and recall, so its bars are almost invisible from the figure.

Varying regression methods: Next we evaluate the impact of regression methods. We use L_1 norm minimization to select important categories and use multiple classifiers in all cases. We compare regression (i) that only uses fitting error as the objective (*Fit*), (ii) that uses fitting error and low rank (*Fit+LR*), (iii) that uses fitting error and temporal stability (*Fit+Temp*), (iv) that uses fitting error, temporal stability, and low rank (*Fit+Temp+LR*). In addition, as a baseline, we use random selection (Rand 0.3) that randomly determines if a given interval has an anomaly with a probability of 0.3 since around 30% of time intervals have anomalies. As shown in Figure 5, Fit+Temp+LR yields the highest accuracy: it out-performs Random, Fit, Fit+LR, Fit+Temp by 823%, 64%, 32%, 6%, respectively, in terms of F-score.

Using multiple classifiers: We evaluate how the number of classifiers affects the performance. As shown in Fig. 6, leveraging more classifiers can generally improves the accuracy, as we would expect. The improvement increases significantly initially and then tapers off. Since the computation cost increases with the number of classifiers, we use 30 classifiers as the default to trade off the benefit and cost.

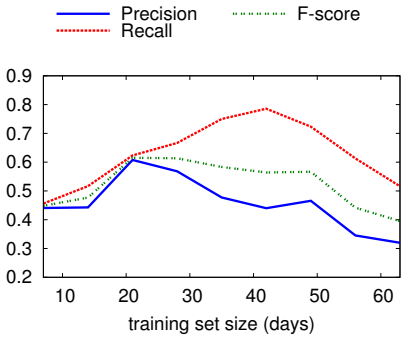


Fig. 7. Varying the size of training set.

Varying the sizes of training sets: We use the previous n days as the training set and detect anomalies on the next day. As shown in Figure 7, when n increases from 7 days to 42 days, the recall increases from 46% to 79%, but when n increases from 42 days to 63 days, the recall drops to 51%. Similarly, when n increases from 7 days to 21 days, the precision increases from 44% to 61%, but a further increase in n to 63-day reduces the precision to 32%. It is because when training set is larger, we have more constraints to find a better solution for Eq. 1 and therefore a higher accuracy. However, as the training set further increases and includes older dataset, the performance may degrade due to the evolving nature of the dataset. We plan to place higher weights to the constraints learned from more recent dataset to further improve the accuracy and robustness in the future.

Varying ratios of weight: Let $x(i,t)$ denote the weight (importance value) of the i-th category at time t. The change ratio of weight is $\sum_t \sum_i (x(i,t) - x(i,t-1))/x(i,t-1)$ when $x(i,t-1)$ is not 0. We see 84% of the changes is within 10%, and 2.5% of the changes is larger than 100%. This suggests x has significant temporal stability, but it can also adapt to different values in order to detect different types of anomalies across two consecutive days.

V. External Data: Social Media

So far, we focus on event detection using the call records, which are the direct feedbacks from customers. However, it is hard to understand the nature of detected events from customer calls due to the following reasons: (i) Categories from call records only have limited text information to describe the issues behind the calls. (ii) Locations of the called customers can be inferred by the area codes of their phone numbers. However, the coverage of each area code is not uniform. Moreover, the customers may use the area codes from their previous locations.

Benefits of using Twitter: To better understand the detected events, we leverage Twitter social media as an indirect channel to understand customers' experience of the service. There are mainly three reasons that make Twitter social media an attractive data source. First, Twitter data is massive; as of March 2012, Twitter has 500 million registered users [24]. Many people share their experiences of the services and products they are using. Second, user feedbacks are coming in near real-time. Compared with the efforts to report issues

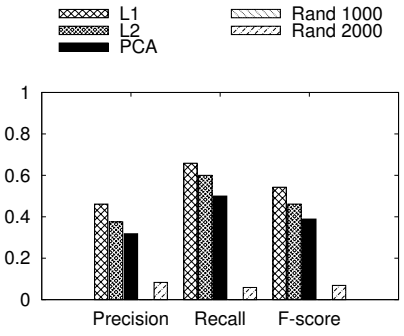


Fig. 4. Varying schemes to identify important features.

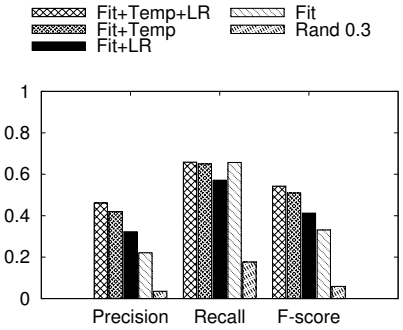


Fig. 5. Varying regression methods.

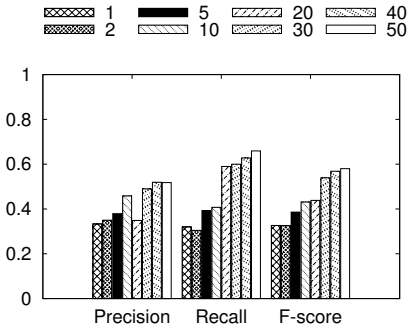


Fig. 6. Varying the number of classifiers.

through customer calls, it is much easier to express their experiences through *tweets*. Third, tweets may have richer context information than customer calls. They have many features which help to understand the various events.

Data description: Twitter data comes with a variety of features, which will help us to understand the nature of detected events. We use the following features of tweets:

- Timestamp;
- Text of original tweets
- Text of normalized tweets, which are the tweets converted into a standard format to ease processing;
- Username: a tweet's author;
- Twitter specific features: URLs, retweets (RT), #hashtags used to mark keywords or topics in a tweet, and @mentions, which are the tweets containing "@username" anywhere in the text;
- Location (city, state, latitude, longitude): Locations can come from the tweets themselves or Foursquare [3].

Understanding detected events: Once the anomalies are detected, we can leverage various features of tweets, such as keywords and locations, to summarize events as follow. We first find tweets that are related to customers experience by selecting tweets with hashtag #XXXFAIL or #XXXSUCK during the entire period, where XXX denotes the name of the provider. Then we try to identify important keywords that appear in the anomaly period. We use the metric, called Term Frequency - Inverse Document Frequency (TF-IDF) [22], to quantify the importance of a keyword. TF-IDF is defined as the number of occurrences of word-level 1-grams and 2-grams during the period of the anomaly (in the selected tweets) divided by the number of occurrences in the entire period (in the selected tweets). The intuition is that a keyword that appears frequently only in the anomaly period but not universally frequently is important. Table II shows some examples of events summarized using this approach.

To locate the impacted regions of the given anomaly, we gather the authors of the collected tweets, which contains n-grams with high TF-IDF scores during the time frame. Then we check the locations of the authors to get the impacted regions. Figure 8 shows an example of the tweet locations for the anomaly occurred at Miami, FL. Although some users may tweet about network incidents from other locations, as long as we see multiple tweets in a region, we can correctly locate the anomaly.

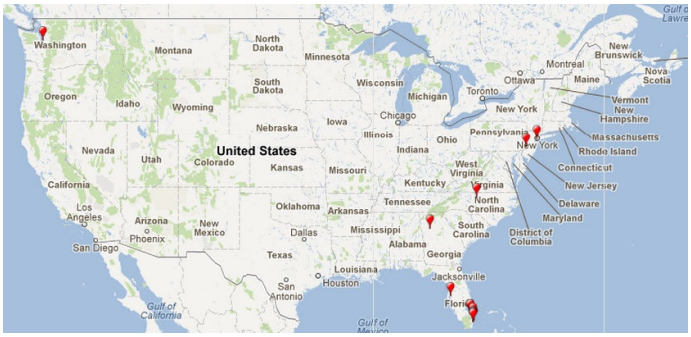


Fig. 8. Tweet locations of an detected event. Most tweets are located around Miami, Florida.

VI. Related Work

Network diagnosis: There has been significant work on anomaly detection and network diagnosis. PCA (*e.g.*, [4], [6], [7]) has been widely used for anomaly detection. [17] shows that PCA is sensitive to how many principal components are used. [18] shows that data poisoning can significantly degrade the performance of PCA. Barford et al. [1] uses wavelets to decompose an original signal into low-, mid-, and high-frequency components and then detect anomalies based on the high-frequency components. Zhang et al. [29] uses compressive sensing to discover anomalies in traffic matrices. [28] develops a framework to capture a range of detectors. [10] proposes a multi-scale robust subspace algorithm to identify changes in performance even when the baseline is contaminated. [21] uses ridge regression to learn Quality of Experience (QoE) in large-scale IPTV systems. Ridge regression does not work well in our context due to possible under-constraint issues and varying customers' response time. NICE [13] uses statistical correlation to detect chronic network problems. Mercury [12] detects persistent behavior changes using the time-alignment for distributed triggers. [11] combines different anomaly detection methods, such as EWMA, FFT, Holt-Winters, and Wavelets, to boost the performance.

The statistical and multi-scale analysis used in the previous works are complementary to our work. We complement the previous works by using L_1 minimization to select important metrics, leveraging temporal stability and low rank to enhance

Event	Location	Event summary by TF-IDF
3G network outage	New York, NY	service, outage, nyc, calls, ny, morning, service
outage due to an earthquake	East Coast	#earthquake, working, wireless, service, nyc, apparently, new, york
3G network outage	Miami, FL	outage, south, service, issue, broward (a county in FL), key, west, equipment, Florida
Internet service outage	Bay Area	serviceU, bay, outage, service, Internet, area, support, #fail
New device release	Nationwide	iphone, sprint, verizon, apple, 4s, android, accessibility
New device release	Nationwide	4s, apple, iphone, #iphone4s, pre-order, order, site, #apple, store

TABLE II
EXAMPLES OF DETECTED ANOMALIES WITH THE SUMMARY. THEY ARE CONFIRMED BY THE NCCO REPORTS (THE DATE AND TIME OF EVENTS ARE NOT SHOWN DUE TO PROPRIETARY ISSUE.)

the accuracy of regression, and applying multiple classifiers to enhance robustness. These techniques can be potentially useful to other network diagnosis problems.

Event detection using Twitter: Twitter has been extensively studied to detect various events such as service issues [16], [14], earthquakes [20], stock markets [19], elections [23], and public health issues [15]. [16] shows the feasibility of social media to understand user experiences and finds correlation between tweets and customer tickets. [14] detects outages of popular services (*e.g.*, Gmail, Bing, PayPal) by tracking the volumes of tweets with the phrase "X is down" or the hashtag "Xfail". Our work takes a step further to summarize the events and localize the impacted regions using detailed information from tweets.

VII. CONCLUSION

We develop a systematic method to automatically detect anomalies in a cellular network using the customer care call data. Our approach scales to a large number of features in the data and is robust to noise. Using evaluation based on the call records collected from a large cellular provider in US, we show that our method can achieve 68% recall and 86% accuracy, much better than the existing schemes. Moreover, we show that social media can be used as a complementary source to get higher confidence on the detected anomalies and to summarize the user feedbacks to anomalies with text and location information.

Acknowledgements: This work is supported in part by NSF Grants CNS-0916309 and CNS-1117009.

REFERENCES

[1] P. Barford, J. Kline, D. Plonka, and A. Ron. A signal analysis of network traffic anomalies. In *Proc. of IMW*, 2002.
[2] D. Donoho. For most large underdetermined systems of linear equations, the minimal L1-norm near-solution approximates the sparsest near-solution. http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0approx.pdf.
[3] Foursquare API. http://developer.foursquare.com/docs/.
[4] Y. Huang, N. Feamster, A. Lakhina, and J. J. Xu. Diagnosing network disruptions with network-wide analysis. In *Proc. of ACM SIGMETRICS*, 2007.
[5] R. Kwitt and U. Hofmann. Unsupervised anomaly detection in network traffic by means of robust PCA. In *Proc. of IEEE Computing in the Global Information Technology*, Los Alamitos, CA, USA, 2007.
[6] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *Proc. of ACM SIGCOMM*, 2004.
[7] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. In *Proc. of ACM SIGCOMM*, 2005.
[8] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural analysis of network traffic flows. In *Proc. of ACM SIGMETRICS*, 2004.
[9] M. ling Shyu, S. ching Chen, K. Sarinnapakorn, and L. Chang. A novel anomaly detection scheme based on principal component classifier. In *Proc. of IEEE Foundations and New Directions of Data Mining Workshop (ICDM03)*, pages 172–179, 2003.
[10] A. Mahimkar, Z. Ge, J. Wang, J. Yates, Y. Zhang, J. Emmons, B. Huntley, and M. Stockert. Rapid detection of maintenance induced changes in service performance. In *Proc. of ACM CoNEXT*, 2011.
[11] A. Mahimkar, A. Lall, J. Wang, J. Xu, J. Yates, and Q. Zhao. Anomaly detection in large operational networks. In *Proc. of ICDCS*, 2011.
[12] A. Mahimkar, H. H. Song, Z. Ge, A. Shaikh, J.Wang, J. Yates, Y. Zhang, and J. Emmons. Detecting the performance impact of upgrades in large operational networks. In *Proc. of ACM SIGCOMM*, 2010.
[13] A. Mahimkar, J. Yates, Y. Zhang, A. Shaikh, J. Wang, Z. Ge, and C. T. Ee. Troubleshooting chronic conditions in large IP networks. In *Proc. of ACM CoNEXT*, 2008.
[14] M. Motoyama, B. Meeder, K. Levchenko, S. Savage, and G. M. Voelker. Measuring online service availability using twitter. In *Proc. of ACM WOSN*, 2010.
[15] M. J. Paul and M. Dredze. You are what you tweet: Analyzing twitter for public health. In *Proc. of AAAI ICWSM*, 2011.
[16] T. Qiu, J. Feng, Z. Ge, J. Wang, J. Xu, and J. Yates. Listen to me if you can: tracking user experience of mobile network on social media. In *Proc. of ACM IMC*, 2010.
[17] H. Ringberg, A. Soule, J. Rexford, and C. Diot. Sensitivity of PCA for traffic anomaly detection. In *Proc. of ACM SIGMETRICS*, 2007.
[18] B. I. Rubinstein, B. Nelson, L. Huang, A. D. Joseph, S. h. Lau, S. Rao, N. Taft, and J. D. Tygar. Antidote: understanding and defending against poisoning of anomaly detectors. In *Proc. of ACM IMC*, 2009.
[19] E. Ruiz, V. Hristidis, C. Castillo, A. Gionis, and A. Jaimes. Correlating financial time series with micro-blogging activity. In *Proc. of ACM WSDM*, 2012.
[20] T. Sakaki, M. Okazaki, and Y. Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proc. of WWW*, 2010.
[21] 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 *Proc. of ACM IMC*, Nov. 2011.
[22] TF-IDF. http://en.wikipedia.org/wiki/Tf∗idf.
[23] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. Predicting elections with twitter: What 140 characters reveal about political sentiment. In *Proc. of AAAI ICWSM*, 2010.
[24] Twitter to surpass 500 million registered users on Wednesday. http://www.mediabistro.com/alltwitter/500-million-registered-users_b188%42.
[25] Wikipedia. Bigram. http://en.wikipedia.org/wiki/Bigram.
[26] Wikipedia. Dice's coefficient. http://en.wikipedia.org/wiki/Dice's_coefficient.
[27] Wikipedia. F1 score. http://en.wikipedia.org/wiki/F1_score.
[28] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. of ACM IMC*, 2005.
[29] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and Internet traffic matrices. In *Proc. of ACM SIGCOMM*, Aug. 2009.