Early Network Failure Detection System by Analyzing Twitter Data

Kei Takeshita, Masahiro Yokota, Ken Nishimatsu

NTT Network Technology Laboratories
Midori-Cho 3-9-11, Musashino-Shi, Tokyo, Japan
Email: takeshita.kei,yokota.masahiro,nishimatsu.ken@lab.ntt.co.jp

***Abstract*—Mobile network failures have occurred many times in recent years. Some network failures become ”silent” failures that mobile carriers cannot detect because of incomplete rules concerning failure detection by the network operating system. However, the increasing number of services and devices, and the increasing complexity of the network make it hard to generate rules that cover all network failures. Therefore, monitoring the network performance from a subscriber's perspective is very important. The traditional way to obtain feedback from subscribers is to use call centers and email. However, it is difficult to detect problems early or in their entirety through those channels because subscribers typically do not call a call center until they are certain the problem was caused by a network.**

In this paper, we discuss a way to monitor a social networking service (SNS) (Twitter in particular) to find out about problems that affect subscribers. A previous study showed the possibility of early detection of network problems by monitoring Twitter. However, since Twitter includes many conversation topics, it is difficult to find tweets that relate to network problems. Searching by a particular keyword is insufficient since it produces a lot of false positive results that contain the keyword but not the topic of the network problem.

We solved this problem by using machine learning to suppress the false positive results. We implemented and evaluated a system to detect network failures from Twitter. As a result, we were able to identify 6 out of 6 large network problems and to suppress the number of false positives to only 6 events, whereas keyword matching detected 94 false positive events. Some of the problems were detected faster than through a call center. Furthermore, we conducted research in order to determine the appropriate machine learning algorithm, parameters, and volume of training data. We also propose a method to estimate the location where the tweeters were located.

I. Introduction

The current cell-phone penetration rate in Japan is over 100%. Despite the increase in the importance of the mobile phone infrastructure, the number of mobile network failures is increasing. In recent years, more than 4,000 mobile network failures have been occurring per year in Japan, according to the government, and some of these failures become critical failures that affect 30,000+ people and continue for more than two hours.

Network operators can monitor network equipment by using monitoring technology such as SNMP (Simple Network Management Protocol) or Ethernet-OAM (Operations, Administration, and Management). Although they can detect hardware failures and rule-based anomalies (e.g., when traffic volume exceeds a certain threshold), it is hard for network operators to detect software bugs and failures that are not included in the rules. Consequently, some failures become silent failures, which cannot be detected by network operators. For example, the problem with an email service [1] was not able to be detected since system equipment and database functions were working normally, but the data were inconsistent. Another difficulty for network operators is understanding how such failures affect subscribers. For example, an equipment failure that interrupts the use of LTE (Long Term Evolution) services will affect 3G subscribers because of the congestion that results from many LTE subscribers switching to 3G services.

To detect these kinds of failures early, network carriers need to have a way to monitor network performance from a subscriber's perspective. The traditional way to get a subscriber's feedback is through call centers or email. However, those channels are not effective for detecting problems early or for understanding a problem in its entirety because subscribers generally do not call a call center until they are certain that the problem is due to the network, and usually only a few people will actually call.

We have studied a way to monitor a social networking service (SNS) (namely, Twitter) to discover problems affecting subscribers. For example, if we see a surge in certain kinds of tweets such as the one below, we suspect a network failure, especially in data communication services. We call these tweets *network-failure tweets*

Why cant i send text messages?

We conducted an online questionnaire with over 1,500 Twitter users in Japan asking them when and how they tweet about network failures, and we found that over one-third of Twitter users had tweeted about a network failure within the past year. In addition, over half of those users tweeted about the network failure during the failure. This is because they wanted to know what the current situation was—whether it was because of a network failure or was only affecting that user. The carrier cannot announce a network failure in the early phase of a network failure since they need to investigate the situation and provide correct information, so Twitter becomes the main source of information about the failure. Even though users had no network access due to the failure, they tried to tweet about it by changing the network, for example, changing from 3G to Wi-Fi, or changing carriers by using a friend's terminal.

A previous study showed the potential effectiveness of monitoring Twitter to discover such problems [2]. That study revealed that there tends to be a surge in tweets that match a certain combination of keywords related to a network failure when a network failure occurs. However, we found in our investigation that keyword matching, a traditional way to search tweets, was not sufficient for automated monitoring because it resulted in many false positives, which contained the keywords but not the topic of the network problem. For example, if we search using the keywords "call" and "drop," we may get the following kinds of tweets. We call these tweets *false tweets*.

I **drop***ped my phone in toilet when i* **call***ed my friend.*

This is because the number of tweets about network problems is very small among all Twitter topics. Consequently, some tweets happen to match the keywords. It is therefore difficult to use automated monitoring or visual monitoring.

We solved this problem by using machine learning to suppress the number of false positives. We conducted experiments using 10,000 tweets in order to find the appropriate machine learning algorithm and to determine the appropriate parameters and volume of training data. We implemented and evaluated a network failure detection system using Twitter. As a result, we were able to find 6 out of 6 large network problems and to suppress the number of false positives to 6 events, whereas keyword matching detected 94 false positive events.

Furthermore, we studied a method to extract location information from Twitter. While location information is important for network operation, it is rarely included in tweets. We propose here a method to estimate the location information and show the accuracy of the estimation method.

Our main contribution is the proposal of a framework that extracts network failure information and its locations from Twitter. It can detect a network failure quickly and automatically. Furthermore, we investigated a method to classify tweets into network-failure tweets and false tweets. As a result, the tweets are simple enough to classify and linearly separable. Therefore, we examined appropriate parameters of algorithms and do not propose further machine learning algorithms.

II. Related Works

The study most closely related to this work is that of Qui et al. [2]. Qui et al. compared Twitter messages with incident reports of carriers and customer care tickets. They extracted network-performance related tweets by searching keywords related to mobile carriers and performance. They found that the correlation between tweets and incidents was low but that tweets were faster than customer care tickets for finding out about incidents when such incidents were mentioned in both. Although Qui's research showed the possibility of using Twitter to monitor network performance, they also found that only 1% of tweets searched by keywords were related to actual network performance.

We discuss some other studies that are not specifically related to networking but that nevertheless show the value of social network content where each Twitter user is considered as a "sensor." To the best of our knowledge, the first such study is that of Sakaki et al. [3]. Sakaki et al. extracted information on a typhoon and an earthquake from Twitter by searching for keywords such as *"typhoon," "shake," and "earthquake"*. They visualized maps by finding location-related tweets that were among the tweets mentioning those words. They also used machine learning to classify the current or past event.

Michael et al. extracted certain public health topics (e.g., cancer, flu) by using a topic model, which is a method of clustering for documents [4]. They showed that the curves of the time series of tweets about the flu and reports to the government were quite similar. However, these studies did not conduct quantitative evaluations from the view of the whole system of event detection.

There are also many commercial services that monitor Twitter to see whether the number of tweets that are matched to keywords exceeds a certain threshold. We have omitted a lot of studies about Twitter itself, such as those concerning information propagation or community evolution.

We explain another challenge to mining the information about the location from twitter. To the best of our knowledge, the first research to attempt this is that of Cheng et al. [5]. The problem setting involved calculating the likelihood of each location $l_j \in L$ for a given tweet set T_i, where L is a location set and i is a user. We found that most of these kinds of studies use supervised learning, in which the distribution of each word for each location is learned from the tweet data of tweeters whose locations are known. This research shows that the accuracy of location estimation is over 160 km at the 50th percentile.

III. Proposed method

A. Monitoring requirements

We tried to construct a system that automatically alerts network operators about network failures by monitoring Twitter. We explain the requirements of network monitoring and give an example of how our system might be utilized before we go on to the details.

We set two requirements for our system to use network monitoring. First, the system should reduce the number of false positives (alerted but nothing happened) as much as possible. Second, the system must add the information about what and where the incident happens. While "what" is written in each tweet, "where" is rarely included. Third, the system should process the tweets in real time to detect the problem as soon as possible.

One example of how our system could be utilized is to show the information extracted from an SNS on the video wall at the Network Operation Center (NOC). The video wall at the NOC mainly shows current network stats: alarms, ongoing incidents, and network performance. Additionally, some displays are used to show a TV news or weather channel so that the operators can find out about current events that may have an impact on the network. Our system can display additional information about the current network performance from the viewpoint of users. Therefore, the operators can learn about a potential network failure at an early stage or can monitor the impact of network failures on users.

B. Tweet categories

We explain the proposed method here. First, we make a distinction between the following three types of tweets.

1) **Tweets indicating that the Twitter user has a network problem**
This is the kind of tweet we want to extract from Twitter. We note that sometimes we cannot determine whether the problem is due to a network failure or the subscribers' device, but we include both of them in this tweet category when Twitter users say that they cannot use the network. Examples of tweets of this type are as follows.

My [Phone Name] won't send any of my texts and all it says is "message send failure."

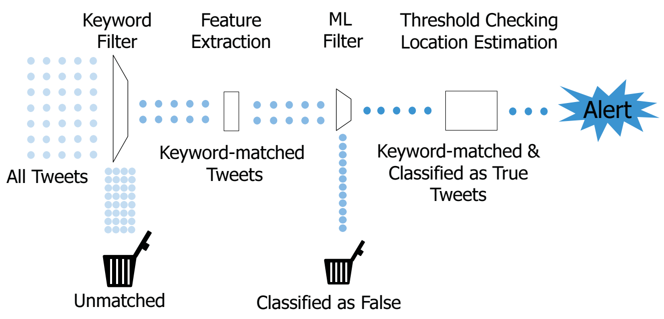


Fig. 1: Architecture of our system

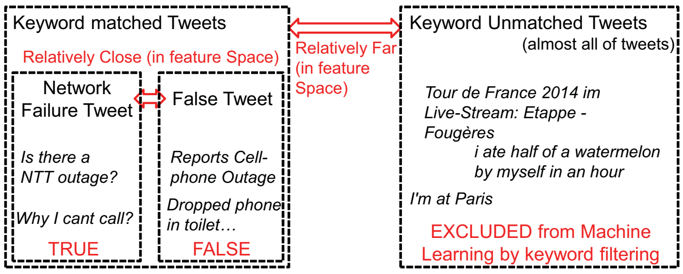

Keyword matched Tweets
Relatively Close (in feature Space)
Network Failure Tweet
Is there a NTT outage?
Why I cant call?
TRUE
False Tweet
Reports Cell-phone Outage
Dropped phone in toilet…
FALSE
Relatively Far (in feature Space)
Keyword Unmatched Tweets (almost all of tweets)
Tour de France 2014 im Live-Stream: Etappe - Fougères
i ate half of a watermelon by myself in an hour
I'm at Paris
EXCLUDED from Machine Learning by keyword filtering

Fig. 2: Image of the distance in relationship between three tweets

My phone won't let me make or receive calls. Is there a [Telecom Name] outage?

We call those tweets *network-failure tweets*

2) **Tweets that match keywords even though the Twitter user does not have a network problem**
These tweets are somehow matched to keywords, for example, when Twitter users have a conversation about phones, or they provide feedback on the news of a network failure (even though they are not affected by it), or when a random URL string contains keywords. Examples of tweets of this type are as follows.

I dropped my [phoneName] in toilet so i cant call or text.
RT: [Telecom Name] Reports Cellphone Outage, Affecting NY. http://bit.ly/xxxxxxxx

We call these tweets *false tweets*

3) **Completely unrelated tweets**
These tweets are not related to a network failure and do not match the keywords.

C. Overview of the proposed method

We show the architecture of the network failure detection in Fig. 1. Our method consists of three phases: keyword filtering, machine learning filtering, and alerting.

First, the keyword filter collects tweets that are possibly associated with a network failure by using a wide range of keywords about failures. In this part, the filtered tweets are a mix of *network failure tweets* and *false tweets*.

Second, the machine learning filter classifies each tweet into true (*network failure tweets*) and false (*false tweets*).

Finally, the alerting element alerts the operators that the network may have some problems when the number of *network failure tweets* exceeds a certain threshold. The operators check the tweets and the location information for each tweet.

There are two reasons we did not classify all tweets by using machine learning.

- An imbalance between *network failure tweets* and the remaining tweets.
The ratio of *network failure tweets* to the remaining tweets is more than 1:10,000. It is said to be hard to classify unbalanced data using machine learning approaches [6]. If 99.99% of the data are from one class, labeling everything with the majority class, which achieves 99.99% accuracy, is usually better than trying to classify the data into the minority class. To classify the minority class, it is preferable to alter the balance of training data so that the classifier can classify the minority class. However, such learning increases the probability of falsely classifying data into the minority class of a large number of *false tweets*. Therefore, we block tweets that do not contain any keywords.
- Closeness of *network failure tweets* and *false tweets*
The machine learning method for natural language processing usually deals with the text based on a value such as the frequency or distribution of words. Therefore, the *network failure tweets* and *false tweets* tend to have a similar value since they pass through the same keyword filter. This is depicted in Fig. 2. Therefore, if we put many completely unrelated tweets into the false class, a lot of *false tweets* will be classified as true.

D. Keyword filter

We collected the keywords to use in the keyword filter by analyzing actual *network failure tweets*. We manually classified all tweets occurring during certain network failures. We used two network failure cases; the first case affected the use of data communication, and the second case affected both data and voice communication. We checked about 2500 tweets and found there were 443 and 482 true tweets, respectively. We double-checked these tweets, which took about two person-days. We note that since we used 10% of the sampled tweets due to the limits of the Twitter application programming interface (API), the actual tweets that existed were about ten times these figures.

We ranked the words that occurred in tweets that were determined to be *network failure tweets* in Table I. The tweets we analyzed were in Japanese, although they have been translated into English for this paper. There is no point in using keywords themselves in other languages; we found that both rankings were similar. The keyword "No service" was especially common in failure B. This is because the "no service" was exclusive to failure B. However, almost all of the keywords appearing in these failures were the same. Therefore, we combined these keywords with the carrier keywords to make the keyword filter.

E. Machine learning filter

In the machine learning phase, tweets are classified into *network failure tweets* and *false tweets*. We use supervised learning that uses the data set of training examples. Each

TABLE I: Ranking of keywords in network failure tweets

Keywords	appeared in first case	appeared in second case
failure	152	238
can't connect	97	56
can't	49	22
dead	42	10
outage	25	15
trouble	23	19
bad	19	22
something is wrong	12	11
restore	9	19
hard to	11	2
down	8	1
can't send	6	3
slow	3	3
no service	0	44
unusual	0	2
disconnection	0	4

training example consists of a pair of a tweet text and a label indicating whether the tweet is a *network failure tweet* or not. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples.

We explain the feature transformation first. The feature transformation converts raw text into values so that an algorithm can calculate the inferred function. We use bag-of-words, which is commonly used in feature transformation in natural language processing. Bag-of-words uses the occurrence of each word in a tweet as a feature. The vector length is the number of words used in this classification. We use the top X frequent words appearing in the training data. In this example, each tweet T_i is translated into a vector of $(0, 1)$ in the dimension of X, and the i-th vector is 1 when T_i contains the i-th word. Additionally, we used a morphological analyzer to determine words since the Japanese language lacks delimiters between words.

We evaluated four learning algorithms known to be effective for binary classification: support vector machine (SVM: w/o kernel and w/ Gaussian kernel) [7], NaiveBayes [8], and Adaptive Regularization of Weights (AROW) [9].

F. Location Estimation Unit

We explain the location estimation method in this section. Because we intend to use location information in the network failure detection system, our accuracy requirement is city-level (within 25 km). Therefore, the related studies (over 300 km) do not meet this requirement. However, when we checked some users' past tweets, we found that we were able to estimate most of the users' locations to at least the prefecture level (within 50 km) by checking the tweets manually. We use geographical names such as those of a station, city, or landmark as hints to estimate the tweeter's location. Therefore, the use of a gazetteer to estimate the location is a very powerful method.

A gazetteer was not used in other previous studies since it presents some problems; for example, there are multiple locations with the same name, and it is necessary to distinguish whether the user was actually in a certain location, or was simply referring to the location by name. Therefore, we used a stochastic method to resolve these problems. We superimposed the score for each tweet matched using a gazetteer in coordinate space. The maximum score in coordinate space is the estimated tweeter's location. We rely on this method because of the overall trend with Twitter users to post information about "what they are doing." Even if some information is wrong because of the problems stated above, the superimposed possibility will be able to estimate the area where the tweeter lives.

Therefore, we propose a method to estimate the location by applying kernel density estimation as follows.

$$Score(i,j) = \sum_{k=0}^{n} \frac{1}{{D_k}^{\alpha}} \exp\Big\{-\frac{(i-lat_k)^2 + (j-long_k)^2}{\sigma^2}\Big\}$$

where (i, j) corresponds to certain coordinates and D_k is elapsed time [day] of the k_{th} tweet. $\Sigma_{k=0}^{n}$ means that this equation uses the tweeter's past n tweets that matches the gazetteer's location names. $(lat_k, long_k)$ indicates the coordinates (latitude, longitude) of the k_{th} tweet that matched the gazetteer entry. $\frac{1}{{D_k}^{\alpha}}$ is a decaying term in accordance with the elapsed time. Since a larger α indicates a faster decay, the newer tweets have a large effect on this estimation. The term $\exp(-\frac{(i-lat_k)^2+(j-long_k)^2}{\sigma^2})$ gives a score based on the closeness to $(lat_k, long_k)$. Since a larger σ indicates a slower decay, the score distribution in coordinate space becomes smooth so that it can cancel noise because of the problems above. However, if the score distribution becomes too smooth, the estimation itself cannot be done. Therefore, there is an optimal σ to estimate the tweeter's location.

TABLE II: Number of tweets for all types of failures

Dataset	date	failures	true data	false data
Failure A	2011/12	Data	443	433
Failure B	2012/08	Data and Voice	482	1209
Failure C	2012/01	Data	1257	1386
Usual[1]	–	–	2216	3896
Total	–	–	4398	6924

IV. Experiments of tweet classification

A. Datasets

We used the randomly sampled 10% of tweet data in our experiments due to the limits of the Twitter API. The data were collected for a period of one year beginning in November 2011. Network failures occurred 6 times with a certain carrier. We extracted the tweets by using the keywords in sec.III-D and manually labeled the true (*network failure tweets*) and false (*false tweets*) for three of the six network failures occurring during that period. We note that there were also *network failure tweets* in periods when no major incidents happened, which were due to small network failures or network construction. The number of all tweets is given in Table II.

There were 4398 *network failure tweets* and 6924 *false tweets*. We note that we removed retweets when we were labeling the data. Retweets mostly appeared in *false tweets*, and the ratio of *network failure tweets* to *false tweets* was about 1:9 in the usual period and 1:3 in network failure periods when we considered the retweets.

B. Machine learning algorithms

We used and compared four machine learning algorithms

1) SVM (w/o kernel, w/ Gaussian kernel): SVM is one of the most effective algorithms for binary classification. SVM constructs a hyperplane that maximizes the margin between two classes. SVM separates the classes linearly but it is often combined with kernel methods. Kernel methods map the data into a high dimensional feature space so that the data are separable linearly in high dimension; as a result, the hyperplane is no longer linear in the original feature dimension. In our study, we found the best $Cost, \gamma$ by using a grid search [10]. We set $Cost$ = 0.08 for SVM (w/o kernel) and ($Cost$, γ)=(1000, 0.001) for SVM (w/ Gaussian kernel)

2) NaiveBayes: NaiveBayes is a simple classifier based on the application of Bayes' theorem with strong (naive) independence assumptions. Despite the fact that assumptions are often inaccurate, the NaiveBayes classifier performs well in many classification tasks such as spam filtering.

3) AROW: AROW is an online learning algorithm that makes predictions based on real-time streaming data. Online learning systems usually update the model with the data predicted by the algorithm itself. This method is powerful when the data change over time. We obtained $weight = 0.05$ by searching for the best parameter.

C. Parameter tuning of algorithms

We tuned the parameters of SVM and AROW. We determined which parameters achieved the highest F-measure by conducting a grid search using the maximum number of training data and features.

D. Parameter setting

We evaluated the accuracy by changing the algorithm, the number of training data, and the number of features as follows.

- Number of training data
  Adding more training data means the classifier can handle more variations of tweets, which improves the accuracy. However, the complexity of the inferred function increases, which results in a longer calculation time. We set the number of training data as 200, 500, 1000, 2000, 4000, 6000, 8000, and 10,000.
- Number of features
  As with the training data, adding more features (words) improves the accuracy and also adds to the complexity. We set the number of features as 50, 100, 200, 500, 1000, 2000, and 4000.

We used 10-fold cross validation in our evaluations. We note that the test data use 1000 tweets for each fold to avoid having a small number of test data. We used an implementation of SVM and NaiveBayes included in Weka [11] and AROW included in Jubatus [12]. The central processing unit (CPU) used for classification was an Intel(R) Xeon(R) CPU E5620 @ 2.40 GHz.

E. Metric

Here, we explain the accuracy metric of the experiment. A single prediction has four different possible outcomes, as indicated in Table III, because True or False was possible for the actual data and the predicted data.

Our system should not miss any network failures (false negatives), and it should also achieve high purity of the classified tweets (false positives). Therefore, we use an F-measure, which considers both falsely classified cases. We show the average F-measure of 10-fold cross validation and its 95% confidence interval using a t-distribution.

TABLE III: Possible outcomes

		Predicted class	
		True	False
actual class	True	True positive	False negative
	False	False positive	True negative

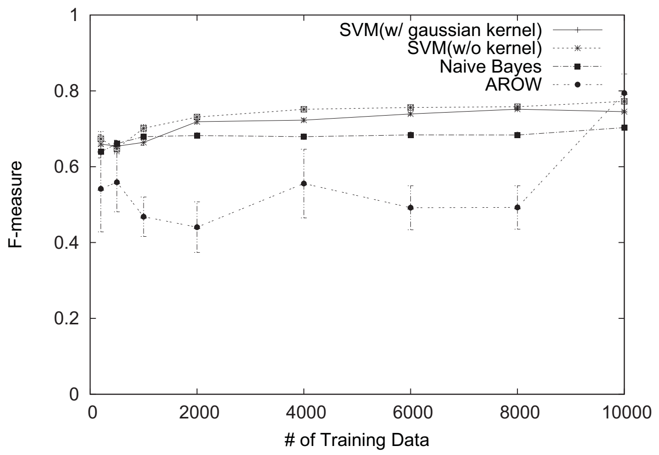


Fig. 3: Number of training data vs. F-measure

TABLE IV: F-measure for different uses of training data

		Training Data		
		Failure A	Failure B	Failure C
Test data	Failure A	-	0.665	0.611
	Failure B	0.687	-	0.655
	Failure C	0.641	0.674	-

F. Evaluation 1: Accuracy of classifying one tweet

We show the F-measure for each algorithm for varying numbers of training data in Fig. 3. The training data were randomly generated from the dataset described in Sec. IV-A, and test data were generated from the remainder of the dataset.

We set the number of features as 4000. We found that the F-measure was about 70% to 80% for each algorithm when there were 10,000 training data. There are few differences in the results for SVM with and without a kernel. AROW degrades slightly with less training data. This is because online algorithms need more data because they can use each datum only once.

Fig. 4 plots the F-measure when the number of features was changed. We set the number of training data at 10,000. The F-measure starts degrading at about 200 feature data. We can see that the F-measures plateaued at 1000 feature data. This suggests that a training data set of only 1000 words canclassify a network failure, which is smaller than in other natural language processing studies.

G. Evaluation 2: Using the datasets in different periods.

We also evaluated classifications when using the three datasets gathered in different periods. We used the datasets described in Table II. We used SVM (w/ Gaussian kernel) as the algorithm and 4000 features.

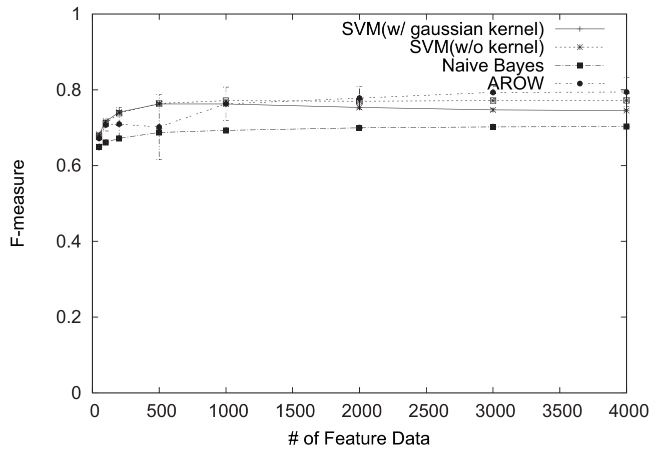


Fig. 4: Number of feature data vs. F-measure

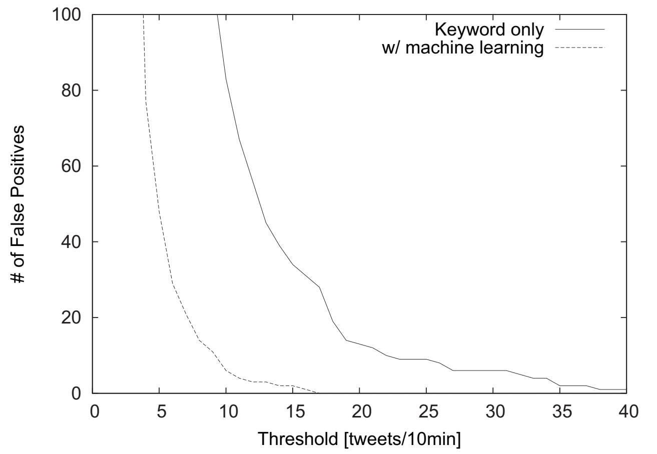


Fig. 6: Threshold vs. accuracy

TABLE V: Time vs. threshold

	Machine learning (10)	Keyword (30)
Failure 1	20	30
Failure 2	30	50
Failure 3	10	20
Failure 4	30	70
Failure 5	60	70
Failure 6	70	70

The results are listed in Table IV. The F-measure was lower than the previous result, which is due to the small number of training data. We found that there were few differences in the dates of the training data. Therefore, we did not think it was necessary to change the data according to time.

We concluded that the tweet data of network failures: 1) are linearly separable, 2) are not very wide and can be classified into about the top 500 words, and 3) do not change as time passes.

H. Evaluation 3: Evaluation of network failure detection

We evaluated a network failure detection system by counting the number of true tweets; we consider that we have detected a network failure when the count exceeds a certain threshold. Fig. 6 plots the results using a machine learning method (lower) and a keyword-only method (upper) over the course of a year.

The training data were used for the period from November 2011 to January 2012, which includes the first 2 failures of the 6 total failures. The number of training data was about 2500 tweets. We set the threshold as 10 [tweets/10 min][2]. There were 100 events detected by keyword only and 12 by machine learning.

Both methods detected the 6 actual network failures. We checked the remaining 94 events visually and confirmed that none of those events were related to the network failure. Therefore, our system drastically suppressed the number of false positive cases. The threshold is the tradeoff parameter between the speed of detecting an event and the false positive ratio. A smaller threshold achieves faster detection but produces more false positive events. The relation between them is plotted in Fig. 6. In this case, the keyword-only method with a threshold of 30 achieved the same number of false positive cases as the machine learning method with a threshold of 10. However, that comes with a price of slow detection. Table V lists the speed of detecting the 6 network failures with machine learning with a threshold of 10 and with the keyword-only method with a threshold of 30. The machine learning method can detect the network failure 15 minute faster on average.

V. Location Estimation Experiment

A. Datasets

We used evaluation data from tweeters whose locations were known. We collected tweets with messages such as "come home," and we added location information obtained from the Global Positioning System (GPS). We obtained 11,792 users' past tweets from the Twitter API. We note that we conducted our evaluation of the classification of network failure tweets and the estimation of the user's location separately. This is because there were very few users who tweet about network failure related topics and include GPS data.

We collected gazetteer data of city names and station names from a database published by a geospatial information authority. The number of entries was 9755. We set optimal parameters D and σ by using a grid-search method based on the 10-fold cross validation of the collected data.

B. Evaluation

Because (1) has to be calculated for each coordinate, we gridiron Japan into a 2000 * 2000 lattice; each divided region is 1 km * 1 km. We show the cumulative distribution of error distance in Fig. 7. The error distance can be calculated by the estimated coordinate by calculating (1) and the actual coordinate. We used the Hubeny formula to calculate the distance from two coordinates. We found that we were able to estimate the location of half of the users within 20 km, and the locations of over two-thirds of user were estimated within 50 km. This result is quite good compared to the result of a

[2]To avoid the case where the same events were detected more than once because the fluctuation dipped below the threshold, we consider whole tweets to be the same events when the count was lower than the threshold for three hours.

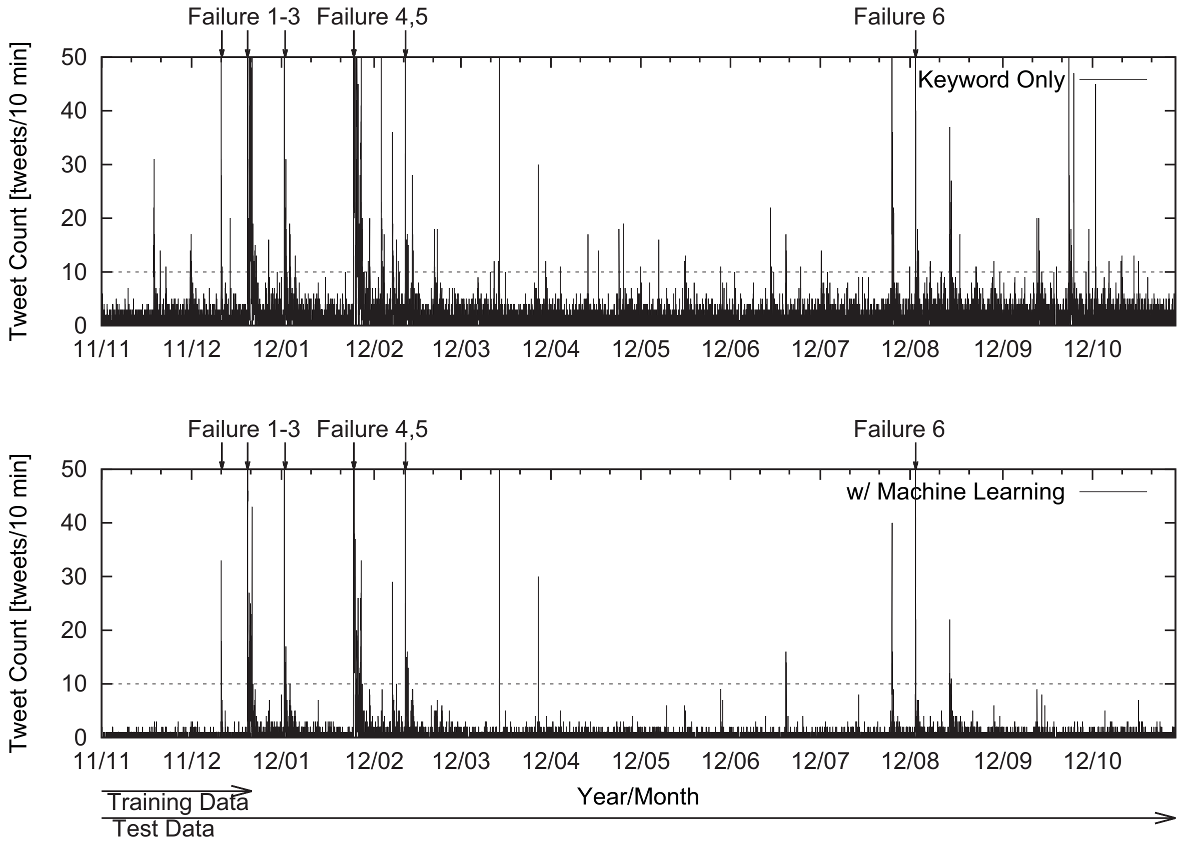


Fig. 5: Number of detected tweets

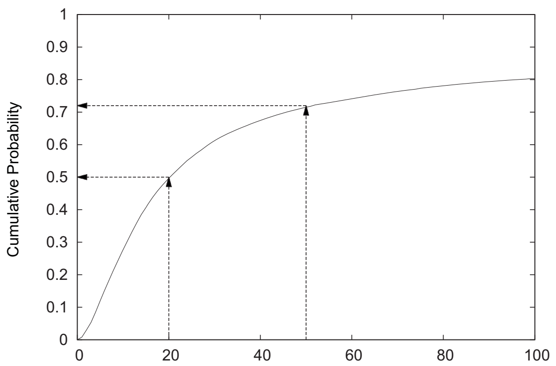


Fig. 7: Error distance

related study in which the error distance of the 50th percentile is 160 km or more.

VI. Processing Speed

We explain here the system's capability to process all tweets in real time.

A. *Keyword filtering*

First, we show the number of tweet data to be filtered by keyword matching for 40 keywords consisting of network failure words and carrier words. On Twitter, about 400 million tweets/day are posted. The Japanese tweets account for about 15% to 20% of this total. Therefore, the average number of Japanese tweets is determined as follows: 400 [million tweets/day] / 86,400 [sec/day] / 5 $\doteqdot$ 1000 [tweets/sec]. The variance in the tweets per minute[3] is shown in Fig. 8. Since the ratio of the maximum to the average number of tweets is as high as 3:1, the system needs to process at least 3000 tweets/sec. We note that the spikes that occur at every 0 minute are caused by tweet bots, which are tweets that are mechanically tweeted by a program. Our system can process about 10,000 tweets/sec for 40 keywords, which is sufficient performance for keyword matching.

B. *Machine learning filtering*

The tweets filtered by keyword matching were about 1% of the number of raw tweets when overestimating significantly. That means the number of tweets that need to be processed by machine learning is about 30 tweets/second. The machine learning method can classify 250 to 1000 tweets/second, which is also sufficient performance.

The time required to construct an inferring function also showed sufficient performance. Even when SVM (w/ Gaussian kernel) and the most time-consuming parameter setting, it took

[3]This system counts tweets in the order of minutes. Therefore, the keyword matching is not strictly done in real time. We plan to process the keyword matching within 1 minute so that the variance in the order of seconds can be ignored

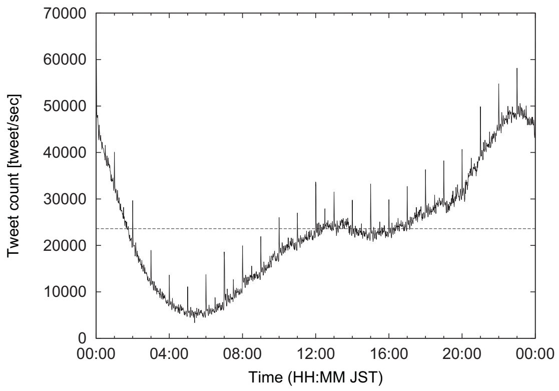


Fig. 8: Variation in tweet speed throughout the day

less than 1 hour to construct a model. We consider this to be sufficient for the system because model construction is not a frequent task, as we found that the temporal change in tweets was small.

C. Location estimation unit

The location estimation unit requires estimating each user who tweets *network failure tweets*. Therefore, the requirement for processing tweets by the location estimation unit is about 3 tweets/second. However, the location estimation unit requires a heavy processing load since (1) has to be calculated for each coordinate point for each tweet. It can process only 0.2 users/second. Therefore, we use parallel computing to calculate this for each user.

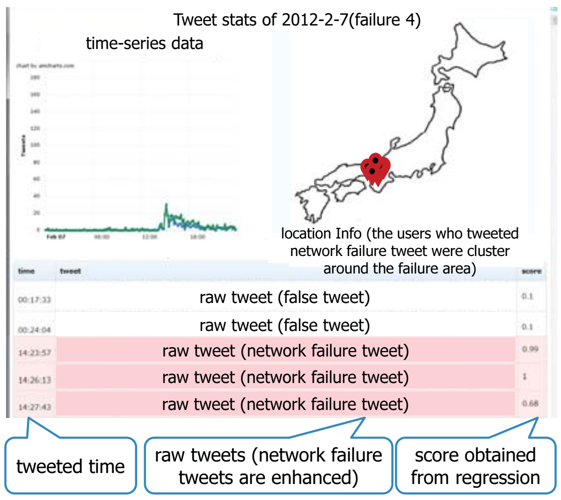
D. System image

We show an image of our system results in Fig. 9. The system shows timelines of tweets, raw tweets, and location information of the network failure tweets. We show a snapshot of the system when network failure 4 occurred. We found that the number of network failure tweets surged, and locations were clustered around the failure area.

VII. Conclusion

We proposed a method to detect network failures using Twitter. We showed that keyword matching is not sufficient because Twitter includes multiple topics. Therefore, we carried out a study in which we combined keyword matching with a machine learning method. Our main findings in tweet classification of network failures are as follows. First, the tweets are simple to separate linearly, so a many machine learning method such as SVM (w/ or w/o kernel), NaiveBayes, or AROW is suitable for this task. Second, the accuracy plateaued when there were about 2000 training data and about 500 features. Finally, the temporal change of tweets is small, which means that frequent reconstruction of the classification model is unnecessary. We showed the efficiency of our system by evaluating whether actual tweets detected network failures. Our method suppressed false positive tweets and detected the network failures substantially faster than the keyword matching technique. Furthermore, we proposed a highly accurate location estimation method that involves using gazetteer information.

In the future, we will evaluate our system to detect small network failures to confirm the feasibility of our system. In



Tweet stats of 2012-2-7(failure 4)
time-series data
location Info (the users who tweeted network failure tweet were cluster around the failure area)
tweeted time
raw tweets (network failure tweets are enhanced)
score obtained from regression

time	tweet	score
00:17:33	raw tweet (false tweet)	0.1
00:24:04	raw tweet (false tweet)	0.1
14:23:57	raw tweet (network failure tweet)	0.99
14:26:13	raw tweet (network failure tweet)	1
14:27:43	raw tweet (network failure tweet)	0.68

Fig. 9: Image of results obtained by our system

addition, we will investigate a method to understand what is happening without reading all classified tweets by combining a summarizing method and location-estimation methods. We will also try to apply this framework to messaging, game, and video services.

References

[1] NTT docomo, "anounce of the network outage," available at http://www.nttdocomo.co.jp/info/network/.

[2] 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 *Proceedings of the 10th annual conference on Internet measurement*, ser. IMC '10, 2010, pp. 288–293.

[3] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: real-time event detection by social sensors." in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 851–860.

[4] M. J. Paul and M. Dredze, "You are what you tweet: Analyzing twitter for public health," in *Proceeding of the 5th International AAAI Conference on Weblogs and Social Media*, 2011, pp. 265–272.

[5] Z. Cheng, J. Caverlee, and K. Lee, "You are where you tweet: a content-based approach to geo-locating twitter users," in *Proceedings of the CIKM'10: 19th ACM international conference on Information and knowledge management*, 2010, pp. 759–768.

[6] F. Provost, "Machine learning from imbalanced data sets 101," in *In Proceedings of the AAAIf2000 Workshop on Imbalanced Data Sets*, 2000.

[7] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[8] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, 1995, pp. 338–345.

[9] K. Crammer, A. Kulesza, and M. Dredze, "Adaptive regularization of weight vectors," *Machine Learning*, vol. 91, pp. 155–187, 2013.

[10] C. wei Hsu, C. chung Chang, and C. jen Lin, "A practical guide to support vector classification," available at http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, 2010.

[11] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, Nov. 2009.

[12] "Jubatus : Distributed online machine learning framework," available at http://jubat.us/.