소셜미디어 수집과 분석을 위한 재난 빅 데이터 플랫폼의 설계

반퀴엣뉘엔, 신응억뉘엔, 양쯔엉뉘엔, 김경백
전자컴퓨터공학부, 전남대학교
e-mail: quyetict@utehy.edu.vn, sinhngoc.nguyen@gmail.com, truongnguyengiang.bk@gmail.com, kyungbaekkim@jnu.ac.kr

Design of a Disaster Big Data Platform for Collecting and Analyzing Social Media

Van-Quyet Nguyen, Sinh-Ngoc Nguyen, Giang-Truong Nguyen, Kyungbaek Kim
Dept. of Electronics and Computer Engineering, Chonnam National University

Abstract

Recently, during disasters occurrence, dealing with emergencies has been handled well by the early transmission of disaster relating notifications on social media networks (e.g., Twitter or Facebook). Intuitively, with their characteristics (e.g., real-time, mobility) and big communities whose users could be regarded as volunteers, social networks are proved to be a crucial role for disasters response. However, the amount of data transmitted during disasters is an obstacle for filtering informative messages; because the messages are diversity, large and very noise. This large volume of data could be seen as Social Big Data (SBD). In this paper, we proposed a big data platform for collecting and analyzing disasters' data from SBD. Firstly, we designed a collecting module; which could rapidly extract disasters' information from the Twitter; by big data frameworks supporting streaming data on distributed system; such as Kafka and Spark. Secondly, we developed an analyzing module which learned from SBD to distinguish the useful information from the irrelevant one. Finally, we also designed a real-time visualization on the web interface for displaying the results of analysis phase. To show the viability of our platform, we conducted experiments of the collecting and analyzing phases in 10 days for both real-time and historical tweets, which were about disasters happened in South Korea. The results prove that our big data platform could be applied to disaster information based systems, by providing a huge relevant data; which can be used for inferring affected regions and victims in disaster situations, from 21.000 collected tweets.

1. Introduction

In the past few years, a number of studies have focused on collecting and analyzing information of social media data for detecting disasters using machine learning algorithms. Those researches utilized Twitter data, which accompanies with well-known limitations such as demographic bias, is a particular interest. Sakaki et al. [1] investigated the real-time nature of Twitter for earthquake event detection by applying Kalman filtering and particle filtering to estimate the center of the burst earthquake. However, users are required to specify explicitly the detected events. And a new classifier needs to be trained to detect new events, which makes it difficult to be extended. Imran et al. [2] employed machine learning for successfully extracting structured information from unstructured, text-based Twitter messages, and compared their results with manual classifications based on crowdsourcing. Vieweg et al. [3] analyzed Twitter messages during the flooding of the Red River Valley in the US and Canada in 2009 seeking to discern activity patterns and extract useful information. Starbird et al. [4] did not only test the hypothesis, in which crowd behaviors could be served as a collaborative filter for identifying people tweeting, but also found that machine learning techniques could be effective in identifying those who are likely 'on the ground'. Although some systems have been proposed for collecting and analyzing disaster information, they are still restricted either in the type of data (e.g., only historical data), the type of storage (e.g., only one local disk on a single computer), or the size of data they could handle. In this paper, we proposed a big data platform for collecting and analyzing disaster data from SBD.

Our work makes the following contributions:

- Firstly, we designed and implemented a collecting module which rapidly extracts disaster information from the Twitter by big data frameworks that support streaming data on distributed system such as Kafka and Spark.
- Secondly, we implemented algorithms for analyzing tweets to distinguish the useful information from irrelevant one. We adapt keyword-based and topic-based filtering which are the common approaches for analyzing Twitter messages.
- Finally, we designed and implemented a real-time visualization on web interface for illustrating the results of analysis phase.

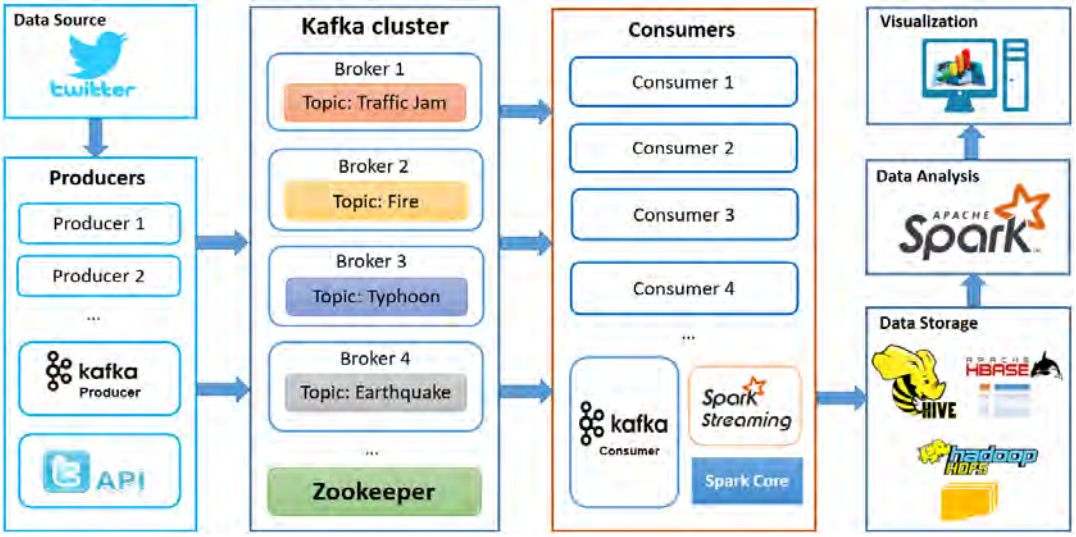

Figure 1. The Architecture of Disaster Big Data Platform

2. The Architecture of Disaster Big Data Platform

We propose a disaster big data platform to support collecting, real-time processing, and visualization as Figure 1 depicts. For data collecting, we use Kafka [5] which is a distributed publish-subscribe messaging system and a robust queue that could handle a high volume of data. Kafka messages are persisted on the disk and replicated within the cluster to prevent data losses. It is built on top of the Zookeeper [6] which allows distributed processes to coordinate with each other through a shared hierarchal namespace which is organized similarly to a standard file system. For data analysis, we use Spark [7] which is a new cluster computing framework designed for fast computation. It utilizes the concept of RDD (Resilient Distributed Datasets), letting users to store data in memory across queries, which makes RDDs to be read and written up faster than typical distributed file systems. After the output data of analysis phase is stored successfully in the database, they would be visualized on the web interface. The specific functions are illustrated as follows.

(1) Data Acquisition System. As a big data disaster analysis platform, it is needed to collect the different types of disasters, such as building fire, typhoon, earthquake, and other natural disasters. Those data are available to collect tweets from Twitter data source. In order to do this, we implement and deploy *Producers* using Twitter APIs along with Kafka Producer. The tweets relating to disasters would be collected by specified keywords, then be stored into corresponding topics which are managed by Kafka Brokers in the Kafka cluster.

(2) Data Warehouse. In our platform, disaster tweets data generated by Kafka Producer as semi-structured data in JSON format. These data are stored in local files system with data replicated mechanism to prevent data losses. For this files, Kafka employs its own storage format that is based on partitioned append-only log abstraction. After the data is consumed by *Kafka Consumers* with some preprocessing tasks by *Spark Streaming*, it is stored in Hadoop distributed file systems (HDFS) [8][9] as text file format. Simultaneously, the data are also stored in both Hive [10] and HBase [11] for the purpose of querying data in subsequently. HBase is a data model providing quick random accesses to huge amount of structured data in distributed system, while Hive is a data warehouse infrastructure built on top of Hadoop for providing data summarization, query, and analysis. In addition, MySQL database is used to store the final output after data is analyzed by Spark. The output data would be visualized by a web application.

(3) *Data Analysis.* In this module, the disaster tweets will be handled by Spark with mathematical statistics and data mining, machine learning techniques. Especially, Spark MLlib is a scalable machine learning library which contains a lot of algorithms such as classification, regression, and clustering, while SparkSQL is Apache Spark's module designed for working with structured data such as Hive or Hbase. SparkSQL includes a cost-based optimizer, columnar storage, and code generation to make queries fast. Therefore, Spark-based analysis module in our platform will provide a fast computation for real-time processing of disaster big data.

(4) Real-Time Data Visualization. We develop visualizations that take advantage of human perceptive to enhance users' disasters situational awareness and support decision-making for disaster management center. First, the display shows a map which uses Google Map API to create markers for the locations where people posted the tweets relating to disasters. Secondly, there are various of statistics of disaster tweets reported in periods by using charts.

3. The Implementation of Disaster Big Data Platform

3.1. Core Algorithms of Data Collecting

The collecting module is implemented to crawl tweets which contain the information relating to disaster from Twitter source. In order to do this, we have implemented two sub-modules: the first one is used to collect data posted in the past by a given specific duration, and the other is used to collect real-time tweets.

We implemented Producer component to look for tweets from Twitter relating to disasters with the given keywords corresponding to the topics and send tweets to Kafka topics. For this purpose, we utilized Twitter APIs and Kafka Producer APIs. We implemented two algorithms for collecting two kinds of tweet data (historical and real-time) that are illustrated as in Algorithm 1 and Algorithm 2.

Algorithm 1. Procedure **TwitterHistoricalProducer** is used to collect tweets
Input: *oAuth, topicName, keywords, since, until, conditions*
Procedure: Saving tweets satisfied conditions in Kafka topics.

1: Initialize *TwitterFactory* by *oAuth* and *KafkaProducer*
2: *tweetSequence* = 0
3: *query* = makeQuery(*keywords, since, until, conditions)*
4: *results* = *TwitterFactory*(*query, tweetSequence*)
5: While *results* != NULL
  5.1: while(*results*.length > 0):
    5.1.1: *KafkaProducer.save(results[0])*; unset(*results[0]*);
    5.1.2: *tweetSequence* = *tweetSequence* + 1.
  5.2: *results* = *TwitterFactory*(*query, tweetSequence*)
  5.3: if results.length == 0 then *results=NULL*

Algorithm 2. Procedure **TwtiterRealTimeProducer** is used to collect tweets in real-time manner
Input: *oAuth, topicName, keywords, filterPrams*
Procedure: Saving tweets satisfied conditions in Kafka topics.

1: Initialize *statusesFilterEndpoint* with *filterPrams; keywords* and Initialize *blockingQueue*
2: *clientBuilder = makeClientBuilder(oAuth, blockingQueue, statusesFilterEndpoint)*
3: *clientBuilder.connect()*
4: *clientBuilder.getTweets()*
5: Initialize *kafkaProducer*
6: while(true)
  6.1: for (int msgRead = 0; msgRead < 1000; msgRead++)
    6.1.1: *value* = queue.take()
    6.1.2: *kafkaProducer*.send(*topicName*, *value*)
  6.2: *clientBuilder.getTweets ()*

Algorithm 3. Procedure **TwitterDataConsumer** is used to aggregate tweets from Kafka topics and save them to HBase
Input: *topics* in Kafka cluster
Procedure: Import tweets data from Kafka to Hbase.

1: Initialize *sparkConf* and *javaStreamingContext, consumer*
2: *consumer.subscribe(topics)*
3: *javaInputDStream = makeDirectStream* (*javaStreamingContext, sparkConf, consumer*)
4: *record = javaInputDStream.getRecord()*
5: Hbase.saveRecord(*record*)
6: streamContext.start();

To obtain tweets sent by Producer, we implemented Consumer component. It is implemented for streaming data whose source is stored in Kafka topics. To speed up extracting data, we employed Spark Streaming with Kafka Consumer APIs for aggregating and storing data. This component will save data in Hive and HBase for subsequently analyzing. The process of Consumer component is illustrated in Algorithm 3.

3.2. Analysis Tweets Data

As mentioned before, after having some preprocessing tasks in Consumers, the data is stored in Hive and HBase for analysis phase. In this phase, we use Spark SQL to read data from Hive and HBase, then implement some mathematical statistics and disasters classification algorithms. In the scope of this paper, for testing our platform, we have done several implementations for tweets statistic as follows.

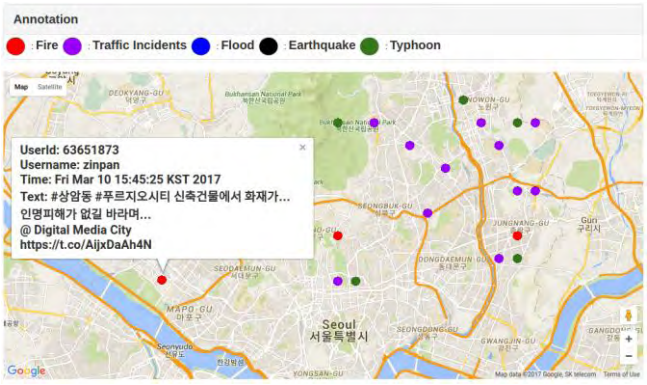

Figure 2. Real-time visualization for tweets about disaster in South Korea

- *Statistic tweets by type of disaster:* The implementation counts the number of tweets relating to each type of disaster including: traffic incidents, earthquake, typhoon, flood, and fire.
- *Statistic tweets rate of disasters type:* This implementation is similar to the previous implementation, but it computes the rate of each type of disaster.
- *Statistic disaster tweets by time:* This implementation is used to count the number of disaster tweets for every hour. The result will indicate when disasters often occur.
- *Statistic disaster tweets by cities:* This implementation counts the number of disaster tweets for every city in a given country.

In the next phase, we implement a PHP web application that read the results of the implementations above and visualize them with various of charts (see the next section).

4. Evaluation

We deployed our platform on five server machines, and each of them has 4 CPUs and 16GB of RAM. To evaluate our platform, we deployed Kafka cluster on four machines, and these machines are used to run Kafka consumers and Spark computations. Another machine is used to run Kafka producer and run Spark master, HBase master, and Hive. In our experiments, for testing our platform, we collected and analyzed the tweets relating disaster information in Korea from Mar 07, 2017 to Mar 16, 2017.

Figure 2 illustrated a road map along with markers which showed the location and type of disaster tweets. The users can interact with this map to see the details of each disaster tweets as shown in the map.

We collected approximately 21.000 tweets during the testing time; which 800 of them has their locations known, while the rest's one is unknown. There are four statistics of disaster tweets shown in Figure 3. In the first one called "Statistics by disaster type", the "traffic incidents" relating tweets; accompanied by "earthquake" and "typhoon" ones; were the most popular topics people often shared about. Their leading rates could be observed from the statistic called "Statistic by rate of disasters type". Also, information from the one called "Statistic of tweets by time" illustrated that people shared their information most frequently from daytime until night; which the highest time was in the afternoon. The last statistic depicted that people from metropolises often shared more tweets about disasters than the ones from other cities.

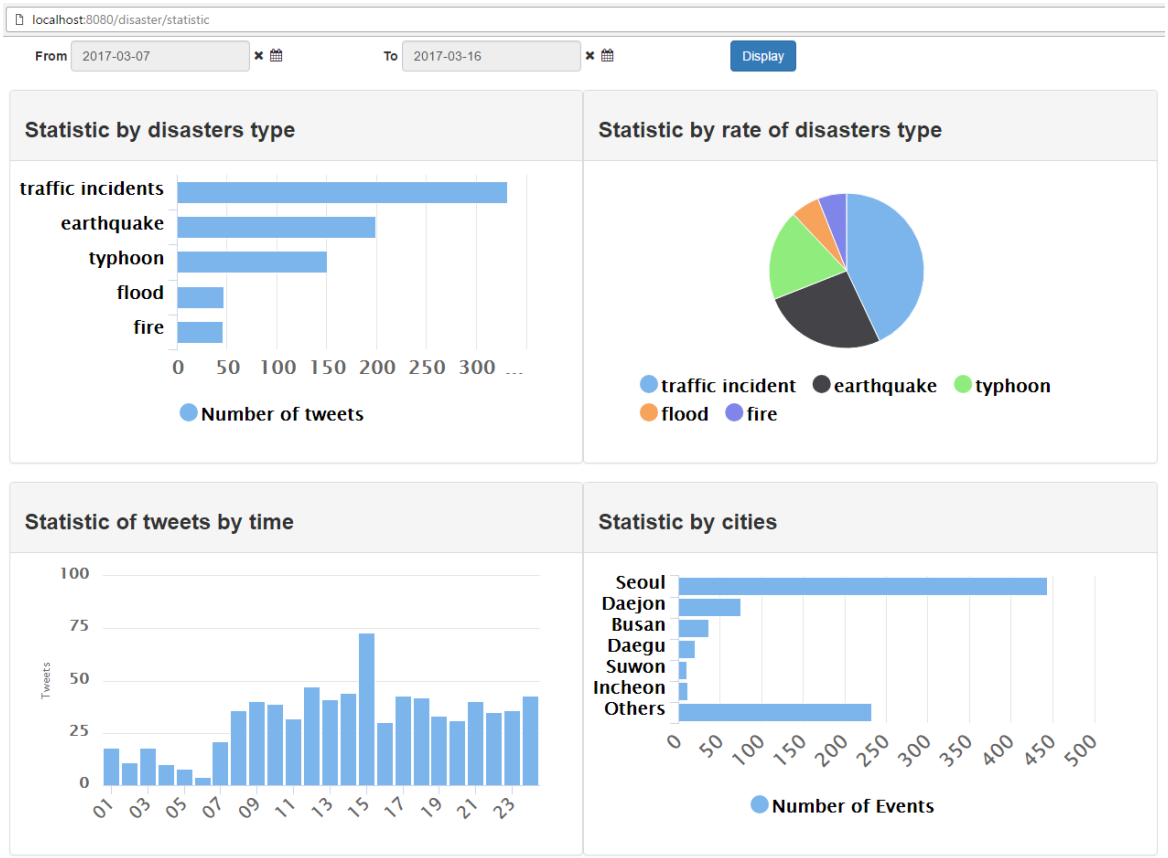

Figure 3. The web application for statistics of disasters from tweets in South Korea

5. Conclusion

This paper proposed a big data platform for collecting and analyzing disaster data from Twitter. We designed and implemented a collecting module to aggregate tweets relating to disaster information. The platform supports streaming data on distributed system and is enabled to collect both real-time and historical Twitter data. The Spark-based module is proposed for analyzing tweets data to support real-time useful information extraction from irrelevant data. We also designed and implemented a real-time visualization on the web interface for showing the results of the analysis phase. The experimental results proved our proposed platform could be applied to disaster information based systems. In the future work, we would focus on applying deep learning for analyzing tweets data to provide accuracy of predicting disasters.

Acknowledgment

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2014R1A1A1007734). This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2016-R2718-16-0011) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] 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, pages 851{860. ACM, 2010.

[2] M. Imran, S. M. Elbassuoni, C. Castillo, F. Diaz, and P. Meier. Extracting information nuggets from disaster-related messages in social media. Proc. of ISCRAM, Baden-Baden, Germany, 2013..

[3] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. Microblogging during two natural hazards events: what twitter may contribute to situational awareness. In Proceedings of the SIGCHI conference on human factors in computing systems, pages 1079-1088. ACM, 2010.

[4] K. Starbird, G. Muzny, and L. Palen. Learning from the crowd: Collaborative filtering techniques for identifying on-the-ground twitterers during mass disruptions. Proc. ISCRAM, 2012.

[5] Apache Kafka, “https://kafka.apache.org/”

[6] Apache Zookepper, “https://zookepper.apache.org/”

[7] Zaharia, Matei, et al. "Spark: Cluster Computing with Working Sets." HotCloud 10 (2010): 10-10.

[8] Borthakur, Dhruba. "HDFS architecture guide." HADOOP APACHE PROJECT http://hadoop.apache.org/common/docs/current/hdfs design. pdf (2008): 39.

[9] Apache Hadoop, “http://hadoop.apache.org” (2009)

[10] Apache Hive, “https://hive.apache.org/”

[11] Apache HBase, “https://hbase.apache.org