IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)
An International Open Access, Peer-reviewed, Refereed Journal

ENHANCED MODEL FOR PREDICTING AND CLASSIFYING THE DISASTER INFORMATION FROM TWITTER USING MACHINE LEARNING

Nivetha. R, Meena. K, Keziah. M, Kiruthika. R, Kannaki @ Vasanthaazhagu. A

Student, Student, Student, Student, Professor & Head of the Department

Department of Computer Science and Engineering

Achariya College of Engineering Technology, Puducherry, India

Abstract: Each day, vast quantities of social media data is being produced and consumed in a constantly increasing price. A user's digital footprint coming from internet sites or mobile products, such as comments, check-ins and Location traces contains valuable details about her behavior under normal and also emergency conditions. The collection and analysis of cellular and social media data before, during and after a tragedy opens new perspectives in areas such as for example real-time event detection, Emergency administration and personalization and valuable insights about the degree of the disaster, it is effect on the affected population and the rate of disaster recovery. Traditional storage space and processing systems cannot cope with how big is the collected data and the complexity of the applied analysis, thus distributed approaches are generally employed. In this function, we propose an open-source distributed platform that may serve while a backend for applications and solutions related to Emergency detection and administration by combining spatio-textual consumer generated data. The machine targets scalability and uses combination of state-of-the artwork Big Data frameworks. It presently supports the most popular internet sites, being easily extensible to any social system. The experimental evaluation of our prototype attestFFs its overall performance and scalability even under large load, using different query types more than various cluster sizes.

Index Terms - Disaster, Twitter, Emergency, Relief Workers

INTRODUCTION

In the modern times we've witnessed an unprecedented data explosion on the web. The wide adoption of internet sites offers concluded in terabytes of produced data each day. In March 2017 for instance, Facebook had normally 1.28 billion daily active users [5], while a lot more than 500 million tweets are produced every day [7]. The proliferation of mobile, smart products has contributed decisively to the trend: With an increase of than half the world right now using a smartphone, active mobile interpersonal median users take into account 34% the full total population [3]. As this data is really a product of human conversation, it reveals valuable information covering all areas of life, even emergency circumstances. Indeed, social media have grown to be a prevalent information resource and communication medium in instances of crises, generating high throughput data just seconds after an emergency occurs, while attested by the 500k tweets stated in the initial hours after the tsunami in Philippines and the 20k tweets/day time registered through the Sandy storm in NY in 2012[16]. Therefore, the processing and linkage of such mobile social press information, which include heterogeneous data varying from text to Location traces, provide huge opportunities for the detection of emergency circumstances [15], for the provision of solutions for immediate Emergency administration (e.g., getting existence signs from people affected [4], communicating with responders, etc.) and for data analytics that may assist in a nutshell and long-term decision making by evaluating the extent of the disaster, its effect on the affected populace and the price of disaster recovery [10]. When the quantity, velocity and selection of the collected data or the complexity of the applied strategies increase, traditional storage and digesting systems cannot cope and distributed methods are employed. To the end, we propose, design and put into action a distributed, Big Data system that can power Emergency detection and administration applications and providers by combining heterogeneous data from various data resources, such as user Location traces from cell phones, profile info and feedback from existing close friends in various social networks linked with the platform. Presently our prototype helps Facebook, Twitter and Foursquare, nonetheless it could be extended to more systems with the correct plug-in implementation. It's been tested with actual data but simulated workloads (we.e., synthetic user foundation). Through distributed spatio-temporal and textual analysis, our bodies provides the following functionalities:

- Socially improved search of Emergency-related details predicated on criteria such as location, period, sentiment or a mixture of the above.

- Automatic discovery of fresh Sights (POIs) and occasions that could show an emerging Emergency of any kind of extent, little or big, which range from traffic jams and spontaneous gatherings such as for example protests, to organic disasters or terrorist attacks.

- Inference of the user's semantic trajectory after and during the emergency through the mixture of her LOCATION traces with background information such as maps, check-ins, consumer comments, etc.

- Semi-Automatic extraction of an user's activity through the Emergency by means ofa blog.

A significant feature of our bodies is the automated POIs detection. A distributed edition [8] of DBSCAN, a well-known clustering algorithm, is applied to the LOCATION traces of our system's users. A dense focus of traces signifies a POI presence. Furthermore, the correlation of spatio-temporal information provided by the Location traces with POI related texts instantly produces a blog page with the user's activity. Moreover, an user may seek out Emergency-related social media info posing both simple and also more complex criteria. Simple requirements include commonly utilized features such as for example keywords (e.g., "flood", "terrorism", "traffic", etc.), location (e.g., a bounding package on a map) or a period frame of interest. Advanced criteria make reference to data annotations, that are based on the processing of data, such mainly because the sentiment of a tweet or a Facebook post. Along with these, each search could be socially charged, considering one's sociable graph when providing responses (e.g., rank content predicated on the sentiment of one's close friends of it). Thus, the proposed platform is with the capacity of supporting queries such as for example "Which are the locations in Greece where protests are occurring and my Facebook close friends (or a subset of these) take part in" or "Inform me personally of the experience and sentiment of my Facebook close friends which were near Lesvos island on June 12, 2017 (when the earthquake of 6.1 Richter level occurred)". The contribution of the work is many-fold:

- We design an extremely scalable architecture that effectively handles data from heterogeneous sources and can cope with big data scenarios.

- We adapt and fine-tune well-known classification and clustering algorithms in a Hadoop-based environment.

- We test out datasets in the region of tens of GB, from Tripadvisor, Facebook, Twitter and Foursquare.

- We validate the effectiveness, precision and scalability of the proposed architecture and algorithms.

ARCHITECTURE

Our system follows a layered architecture that's illustrated in Figure 1. It comprises the frontend and backend layers, both which follow a completely modular design to favor versatility and simple maintenance. The frontend layer constitutes the idea of interaction with an individual and includes all applications linked to Emergency recognition and management that may be supported simply by the platform. This application could be a web, a mobile phone (e.g., Google android, iOS) or an indigenous application. To check the platform's basic functionality, a web software has been implemented. The applications talk to the backend coating through an escape API. The requests to the backend along with the responses of the backend back again to the user follow a particular JSON file format. This feature allows the seamless integration of any client applications with the system. The backend is divided in two subsystems, the processing subsystem and storage space subsystem. For the processing subsystem, a Hadoop cluster and a web server farm have already been deployed, to be able to appeal to the special requirements of social networking data processing.

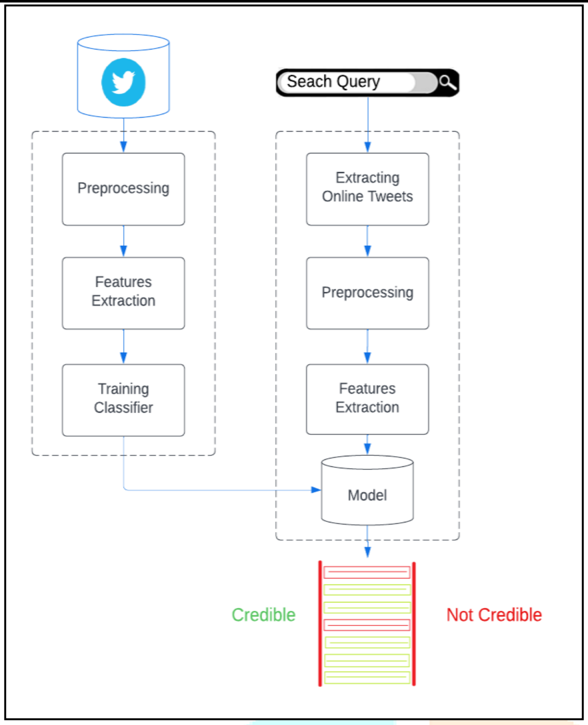

Fig 1. Architecture

Indeed, the quantity and velocity of the info made by social networks demand a distributed approach. Because the Hadoop framework offers emerged as the utmost prevalent platform of preference for large-level analytics, we design and deploy the next Hadoop-based digesting modules:

(a) the info Collection module,

(b) the Sentiment Evaluation module,

(c) the written text Processing module and

(d) the function detection module.

The net server farm hosts an individual Administration and the Query Answering modules, which become gateways to the system. Both modules are implemented as lightweight web solutions which place load to the datastore without stressing the web servers. The storage subsystem is accountable for storing all of the data utilized by our platform, both raw and processed. We make reference to the parts of the storage subsystem as repositories. Repositories are conceptually categorized to primitive and non-primitive data repositories. Primitive repositories shop raw, unprocessed data, known as primitive data. Primitive data are gathered from external data resources such as internet sites(Facebook, Foursquare, Twitter) and LOCATION traces and so are directly kept to the system. Non-primitive data repositories are the types serving answers to queries and keep info extracted from the evaluation of primitive data through the usage of spatio-textual algorithms. To take care of multiple concurrent users that issue queries to the system, the program follows a scalable strategy. To the end, the Apache Base Cluster [1] NoSQL datastore is used. Nevertheless, there are queries which need either complex indexing schemes or extended random usage of the underlying data. These queries cannot be effectively executed in Base Cluster. Because of this, we devise a hybrid architecture that uses Base Cluster for batch queries which can be efficiently executed in parallel and Server [6] for online random-gain access to queries that cannot. In the next, we describe in greater detail the repositories and the processing modules of the proposed system.

STORAGE SUBSYSTEM REPOSITORIES

EMERGENCY POI REPOSITORY:

It really is a non-primitive data repository which has all the details our platform must find out about POIs where a Emergency offers occurred. The name of a POI, its geographical area, the keywords characterizing the Emergency and the sentiment-related metrics are stored in this repository. A fresh access to the repository could be inserted either explicitly by an individual through the net GUI or automatically by the function Detection module. While POI repository must deal with low place/update rates, it must be able to handle weighty, random access read loads. Thus, indexing features are required. Server gives such features and has consequently been chosen as the perfect infrastructure for hosting the emergency POI repository.

SOCIABLE INFORMATION REPOSITORY:

This primitive data repository is usually a Base Cluster-resident table where interpersonal graph information is kept. For every platform user and for each connected social networking, the set of friends is persisted. Even more specifically, we shop a compressed list with the initial social networking id, the name and the profile picture of every friend. This list is usually periodically up to date through the Data Collection module to fully capture possible adjustments in an user's sociable graph.

TEXT MESSAGE REPOSITORY:

The textual data gathered from sociable media an processed through the written text Processing module are kept in this non-primitive data repository. Since the anticipated level of this data type is high, the written text repository is most demanding in conditions of disk space requirements. Because of this, it is stored in Base Cluster and pass on across all obtainable cluster nodes. THE WRITTEN TEXT repository holds all the collected feedback and reviews which contain Emergency related keywords, such as “flood”, “hurricane”, “visitors”, “protest”, etc. with their geo-location. Texts are indexed by consumer, geo-location and period. For just about any given Emergency-related keyword and any given rectangle on the map, we're able to retrieve the comments that any user produced at any moment interval, containing the keywords and geo-located within the search area.

FRIEND ACTIVITY REPOSITORY:

To be able to give information predicated on social friends' activity, we have to keep an eye on all places (Emergency POIs) and social media content material of an user's friends. These details is maintained in the Friend Activity repository, a non-primitive data repository persisted as an Base Cluster desk. Each activity is definitely represented by a task data structure with the total Emergency POI information (name, latitude, longitude, etc.) and also the wording of possible posts. Furthermore, this framework is enriched with sentiment metrics (positive/bad) through the Sentiment Evaluation module. Every time a consumer or an user's public friend is tracked near a Emergency POI through her LOCATION trace or geo-tagged posts, a task struct indexed by user and time is put into the repository. Therefore, for any given period interval, we realize the places that of an user's close friends have already been and a rating indicating each friend's sentiment. An obvious remark is definitely that the experience struct introduces high data redundancy, since every time someone visits a Emergency POI, the complete POI information is registered within the struct. The choice schema design strategy will be joining POI information with activity information at query time. However, our experiments recommend data replication to become more efficient. Our schema in mixture with Base Cluster coprocessors and a completely parallel query mechanism presents higher scalability and achieves decrease latency in case of many concurrent users, which may be the case in emergency circumstances. Thus, we sacrifice cheap space for storage for efficiency.

LOCATION TRACES REPOSITORY:

Cellular devices with appropriate geo-location applications supported by our system installed, can drive their LOCATION traces to the platform. These traces are kept in the primitive data repository of LOCATION Traces. Because the platform may constantly receive Location traces, this repository is expected to cope with a high update price. Furthermore, as Location traces are not queried directly simply by the users but are periodically prepared in bulk, there is absolutely no have to build indices upon them. The quantity of data, the possibilities for parallel mass processing and the lack of indices are the primary explanations why we choose Base Cluster as the storage space substrate for Location data.

SITES REPOSITORY:

We define a semantic trajectory to become a timestamped sequence of Emergency POIs summarizing user's activity during Emergency. these details is kept in a non-primitive data repository, the Websites repository. As POIs, blogs are generally queried by users however they don't need to deal with heavy updates and therefore are kept as a Server resident desk.

PROCESSING SUBSYSTEM MODULES

USER MANAGEMENT MODULE:

AN INDIVIDUAL Management module is in charge of an individual authentication to the platform. An individual is registered either through the cellular applications or the web site. The signing-in procedure is completed only with the usage of the social networking credentials. The sign up workflow follows the ZAuth protocol. The ZAuth authorization framework allows a third-party application to obtain usage of an HTTP service with respect to a resource owner. When the authentication is prosperous, an individual logs in and an access token is returned to the system. With this token, the system can connect to the connected social networks with respect to the end consumer. It could monitor user's activity, user's friends activity, posts etc. When multiple internet sites are connected to the platform, the platform joins the acquired data and enriches the info that's indexed and stored.

DATA COLLECTION MODULE:

The features of the module is to get data from exterior data resources. Periodically, the info Collection module scans in parallel all of the authorized users of the system; each employee scans a different collection of users. For every consumer and for all linked internet sites, it downloads all the interesting updates from the user's social profile. Because the platform provides interpersonal geo location services, interesting improvements are believed to be user checkins and the accompanying comments, position updates and geo-located tweets. Out of this information, the system will be able to gain understanding of the existence of Emergency events and people's sentiment about them. Once data are streamed to the platform, part of these are indexed and kept in the primitive data repositories, as the rest are prepared in-memory and indexed and kept to the appropriate non-primitive data repositories. Text message Processing module: This module indexes all textual information collected by the info Collection module relating to predefined Emergency related keywords (e.g., flood, earthquake, protest, etc.). To take action, it employs standard Natural Vocabulary Processing (NLP) techniques (lemmatization, stemming, etc.) to monitor the predefined keywords and shops the leads to the written text repository.

SENTIMENT EVALUATION MODULE:

The Sentiment Evaluation module performs sentiment evaluation to all or any textual information the system collects through the Data Collection module. Feedback are classified, real-period and in-memory, as positive or unfavorable. The score which outcomes from the sentiment evaluation is persisted to the datastore combined with the text itself. As a classification algorithm, we select the Naive Bayes classifier that the Apache Mahout [2] framework provides. Naive Bayes is usually a supervised learning algorithm and as a result it requires a pre-annotated dataset because of its training. For the teaching, data from Tripadvisor, made up of reviews for POIs can be used. The reason for choosing this schooling collection is that Tripadvisor feedback are annotated with a rank from 1 to 5 that can be utilized as a classification rating. After an extensive experimental study and a fine-tuning of the algorithm parameters, we managed to make a highly accurate classifier that achieves an precision ratio of 94% towards unseen data.

EVENT RECOGNITION MODULE:

The recognition of new occasions and emergency POIs takes its core efficiency of our system. A distributed, Hadoop based execution of the DBSCAN clustering algorithm [8] is utilized for this reason. The module is named periodically and procedures in parallel the updates of the LOCATION Traces repository and discover traces of high density; high density traces imply the presence of a fresh POI. To avoid detecting already known Emergency POIs, traces dropping close to existing Emergency POIs in the repository are filtered out and so are not taken into account for clustering. Query Answering module: The Query Answering may be the module that executes search queries. A search query may take as input the next parameters:

- a bounding package of coordinates (i.e., on a map)
- a listing of keywords
- a list of social networking friends
- a time window
- results sorting criteria

- the number of leads to be returned Queries are distinguished in customized and non-personalized ones. Personalized will be the queries that specifically concern the sociable media close friends (or a subset of these) of an user. Therefore, if a listing of friends is offered, the query is regarded as personalized. For individualized queries, the determined friends' activities ought to be taken into accounts. The repository that maintains such customized information is the Friend Activity repository which resides in Base Cluster. Hence, as Physique 1 depicts, individualized queries are directed toward the Base Cluster cluster. To be able to efficiently resolve customized queries, Base Cluster coprocessors are utilized. Each coprocessor is responsible for an area of the Friend Activity repository and performs Base Cluster obtain requests to the users under it is authority. Since different close friends are located with high probability in various regions, a different coprocessor manages serving their actions and multiple obtain requests are released in parallel. Increasing the amount of regions prospects to a rise in the amount of coprocessors and thus a higher amount of parallelism is achieved within an individual query. However, non-personalized queries involve repositories that handle no-personalized information. The Emergency POI repository, which contains all the required info, resides in Server. Therefore all non-personalized queries are translated to choose SQL queries and directed towards Server.

EXPERIMENTS

We experimentally evaluate our bodies in conditions of performance, scalability and accuracy of it is modules. In this manner we validate both our architectural style and the selected optimizations.

SCALABILITY AND OVERALL PERFORMANCE EVALUATION

We 1st present some experiments for the scalability and overall performance of the query answering module. Utilizing a man made dataset, we check the power of the platform to react to customized queries issued by it is users for various loads and various cluster sizes. Each individualized query involves a couple of friends and returns the actions, i.e., text message, geo-location and sentiment linked to a Emergency scenario designated by a keyword (e.g., earthquake). The platform user can set several other parameters aswell: the geographical location, the period of time of the actions, etc. Inside our experiments, we recognized that the dominant factor in the execution period of the query may be the number of social networking friends that the system users define. For the man made dataset era, we collected information from OpenStreetMap about 8500 POIs (which we considered Emergency POIs for out experiment) located in Greece. Predicated on those POIs, we emulated the experience of 150k different social networking users, each of whom offers performed activities in a number of Emergency POIs and assigned a sentiment to it. The amount of actions for each social networking friend follows the standard Distribution with $\mu = 170$ and $\sigma = 101$. The dataset is usually deployed into an Base Cluster cluster comprising 16 dual-primary VMs with 2 GB of RAM each, operating Linux (Ubuntu 14.04). The VMs are hosted in an exclusive Openstack cluster. At first, we will study the effect of the amount of social network friends in to the execution period of an individual query. At this time we will also examine the way the cluster size impacts the execution period of a query. Secondarily, we will extend our research to multiple concurrent queries where we will also examine the behavior of the platform for different amounts of concurrent queries and various cluster configurations. For the first stage, we measure the execution time of a query for different amounts of friends. In Physique 2 we offer our results. In this experiment, we executed 1 query at the same time involving from 500 to 3500 social networking friends for three different cluster setups comprising 4, 8 and 16 nodes. The friends for each query are picked randomly in an uniform way. We repeated each query 10 times and we offer the average of these runs. The amount of friends impacts the execution amount of time in an almost linear way. Furthermore, a rise in the cluster size prospects to a latency lower, since the execution is going on in parallel to multiple nodes. Through the use of Base Cluster coprocessors, we were able to exploit the vicinity of the computations into specific portions of the info: each coprocessor operates right into a specific Base Cluster region (holding a specific part of the info), eliminates the actions that do not fulfill the user defined requirements, aggregates multiple activities discussing the same Emergency POI and annotates them with aggregated sentiment scores. Finally, each coprocessor returns to the net Server the set of emergency POIs, the related activities and sentiment and the net Server, subsequently, merges the outcomes and returns the ultimate list to the finish user.

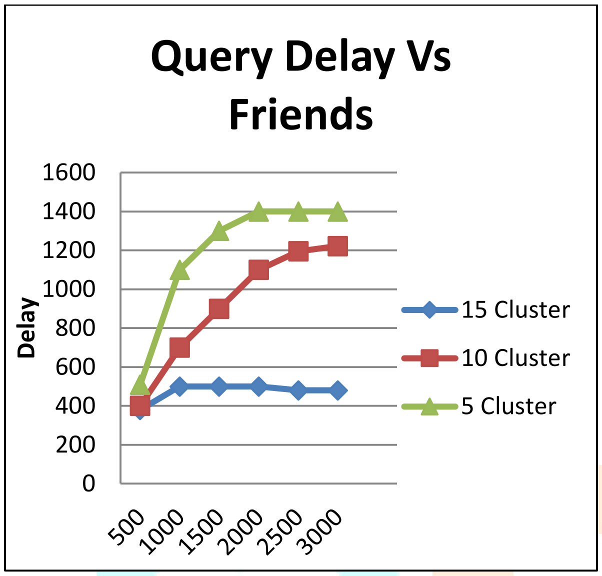

Fig. 2. Query Delay vs number of Friends

Using the previously explained technique, we accomplished latencies less than 1 second for a lot more than 5000 users. Bearing under consideration that social networks like Facebook, retain a limit on the utmost number of connections (5000 friends per consumer), we can assurance that the latency for every query remains acceptable for a genuine time application. We have now extend our evaluation for the instances where multiple queries are issued concurrently to the platform. For our experiments we create several concurrent queries involving 6000 social networking close friends each and we measure their execution period for different cluster sizes. In Figure 3, we offer our results. The execution amount of time in the vertical axis represents the common execution time for each case.

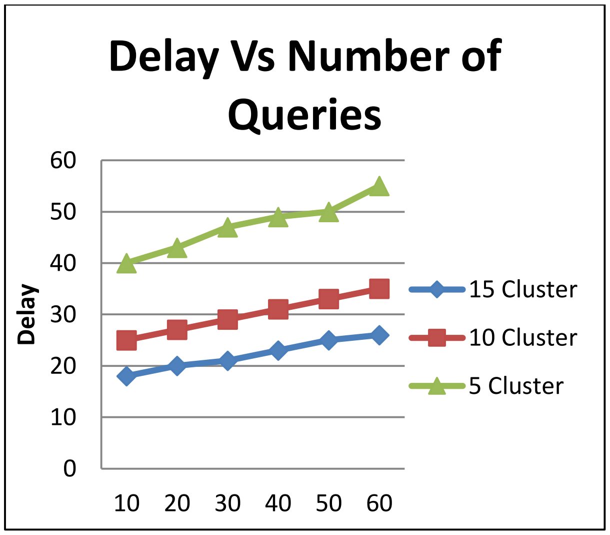

Fig. 3. Average execution time for queries

As Number 3 demonstrates, a rise in the amount of concurrent queries leads to worse efficiency (larger execution time). Nevertheless, for larger cluster sizes we are able to make the next observations: (a) even for the cheapest number of concurrent queries the 16 cluster case is approximately 2.5 times much better than the 4 cluster case, indicative of the correct utilization of even more resources and (b) larger cluster sizes don't allow execution time to go up fast as the amount of concurrent queries increases. Particularly, when the cluster includes 4 nodes, the execution time is high even for the cheapest number of concurrent queries and it continues to go up rapidly while the quantity of queries is increased. In the 8 nodes case, although initially the achieved execution occasions are relatively low, the increase becomes quick for more concurrent queries, whereas in the 16 nodes case we see that the boost is in a minimum amount level. That is indicative of the scalability of the platform, since even more resources are properly utilized and the platform becomes resistant to concurrency. Finally, since greater number of concurrent queries leads to even more threads in the net Server which, subsequently, hits the cluster, we are able to avoid any potential bottlenecks by replicating the net Servers even though simultaneously, we use lots balancer to path the visitors to the net servers accordingly. Inside our experimental setup, we determined that two 4-cores web servers with 4 GB of RAM each are more than enough in order to avoid such bottlenecks.

CONCLUSIONS

In this paper we presented a storage space and processing system that will be able to support applications and providers that leverage the energy of Big Data made by mobile and social networking users to identify and manage emergencies. Such data include spatio-temporal and textual info, which may be combined to automatically discover POIs and events that could indicate an emerging Emergency of any level, provide Emergency-related information predicated on criteria such as for example location, period, sentiment or a mixture of the over and infer an user's semantic trajectory after and during the Emergency. Our prototype, which currently supports Facebook, Twitter and Foursquare, can provide query latencies of a few seconds even under large load, falling in to the sub-second scale when executing over a 16-node cluster. Releasing an online general public version of our bodies and testing it below real life conditions is portion of our future programs.

REFERENCE

1. Cisco: Cisco visual networking index: global mobile data traffic forecast update 2015–2020, White Paper (2016) 7. Cooper, G., Yeager,V., Burkle, F., Subbarao, I.: Twitter as a potential disaster risk reduction tool. part 1: introduction, terminology, research and operational applications. PLoS Curr. Disast. (2015)

2. Cheng, Y., Qin, C., Rusu, F.: GLADE: big data analytics made easy. In: Proc. of the 28th International Conference on Management of Data, pp. 697–700 (2012)

3. I. Mytilinis, I. Giannakopoulos, I. Konstantinou, K. Doka, and N. Koziris. MoDisSENSE: A distributed platform for social networking services over mobile devices. In Big Data (Big Data), 2014 IEEE International Conference on, pages 49–51. IEEE, 2014.

4. Ashton, K.: That 'Internet of Things' Thing. RFiD Journal 22, 97–114 (2009)

5. Qin, X.-P., Wang, S.: Big Data Analysis—Competition and Symbiosis of RDBMS and MapReduce. Journal of Software 23(1), 32–45 (2012)

6. Velev, D., Zlateva, P.: Principles of Cloud Computing Application in Emergency Management. In: Proc. of the International Conference on E-business, Management and Economics, pp. 119–123 (2011)

7. Facebook Stats.https://newsroom.fb.com/company-info/.

8. Postgresql. http://www.postgresql.org/. 7. Twitter Usage Statistics. http://www.internetlivestats.com/twitter-statistics/.

9. Y. He, H. Tan, W. Luo, H. Mao, D. Ma, S. Feng, and J. Fan. Mr-dbscan: An efficient parallel density-based clustering algorithm using mapreduce. ICPADS, 2011.

10. Atzori, L., Iera, A., Morabito, G.: The internet of things: A survey. Computer Networks 54(15), 2787–2805 (2010)

11. M. Imran, C. Castillo, J. Lucas, P. Meier, and S. Vieweg. Aidr: Artificial intelligence for disaster response. In Proceedings of the 23rd International Conference on World Wide Web, pages 159–162. ACM, 2014.

12. M. B. Lazreg, M. Goodwin, and O.-C. Granmo. Deep learning for social media analysis in crises situations. In The 29th Annual Workshop of the Swedish Artificial Intelligence Society (SAIS) 2–3 June 2016, Malm¨o, Sweden, page 31, 2016.

13. Cheng, S., Z, Q.Q.Q. : Big data analytic with swarm intelligence. Ind. Manag. Data Syst. (2016)

14. Chatzimilioudis, G., Konstantinidis, A., Laoudias, C., Zeinalipour-Yazti, D.: Crowdsourcing with smartphones. IEEE Int. Comput. 16(5), 36–44 (2012)

15. Chen, Y., Miao, D., Wang, R.: A rough set approach to feature selection based on ant colony optimization. Pattern Recogn. Lett. 31, 226–233 (2010)

16. Cheng, S., Liu, B., Ting, T.O., Qin, Q., Shi, Y., Huang, K.: Survey on data science with population-based algorithms. Big Data Anal. 1(1), 3 (2016)