Exploiting Social Networking and Mobile Data for Crisis Detection and Management

Katerina Doka[1]([✉]), Ioannis Mytilinis[1], Ioannis Giannakopoulos[1],
Ioannis Konstantinou[1], Dimitrios Tsitsigkos[2], Manolis Terrovitis[2],
and Nectarios Koziris[1]

[1] CSLab, NTUA, Athens, Greece
{katerina,gmytil,ggian,ikons,nkoziris}@cslab.ece.ntua.gr
[2] IMIS, RC Athena, Athens, Greece
{tsitsigkosdim,mter}@imis.athena-innovation.gr

Abstract. Every day, vast amounts of social networking data is being produced and consumed at a constantly increasing rate. A user's digital footprint coming from social networks or mobile devices, such as comments, check-ins and GPS traces contains valuable information about her behavior under normal as well as emergency conditions. The collection and analysis of mobile and social networking data before, during and after a disaster opens new perspectives in areas such as real-time event detection, crisis management and personalization and provides valuable insights about the extent of the disaster, its impact on the affected population and the rate of disaster recovery. Traditional storage and processing systems are unable to cope with the size of the collected data and the complexity of the applied analysis, thus distributed approaches are usually employed. In this work, we propose an open-source distributed platform that can serve as a backend for applications and services related to crisis detection and management by combining spatio-textual user generated data. The system focuses on scalability and relies on a combination of state-of-the art Big Data frameworks. It currently supports the most popular social networks, being easily extensible to any social platform. The experimental evaluation of our prototype attests its performance and scalability even under heavy load, using different query types over various cluster sizes.

1 Introduction

In the recent years we have witnessed an unprecedented data explosion on the web. The wide adoption of social networks has concluded in terabytes of produced data every day. In March 2017 for example, Facebook had on average 1.28 billion daily active users [5], while more than 500 million tweets are produced on a daily basis [7]. The proliferation of mobile, smart devices has contributed decisively to this trend: With more than half the world now using a smartphone, active mobile social media users account for 34% the total population [3].

As this data is mostly a product of human communication, it reveals valuable information covering all aspects of life, even emergency situations. Indeed, social

media have become a prevalent information source and communication medium in cases of crises, producing high throughput data only seconds after a crisis occurs, as attested by the 500k tweets produced in the first hours after the tsunami in Philippines and the 20k tweets/day registered during the Sandy storm in New York in 2012 [16].

Thus, the processing and linkage of such mobile social media information, which includes heterogeneous data varying from text to GPS traces, provide tremendous opportunities for the detection of emergency situations [15], for the provision of services for immediate crisis management (e.g., getting life signs from people affected [4], communicating with responders, etc.) and for data analytics that can assist in short and long-term decision making by evaluating the extent of the disaster, its impact on the affected population and the rate of disaster recovery [10].

When the volume, velocity and variety of the collected data or the complexity of the applied methods increase, traditional storage and processing systems are unable to cope and distributed approaches are employed. To this end, we propose, design and implement a distributed, Big Data platform that can power crisis detection and management applications and services by combining heterogeneous data from various data sources, such as user GPS traces from cell phones, profile information and comments from existing friends in various social networks connected with the platform. Currently our prototype supports Facebook, Twitter and Foursquare, but it can be extended to more platforms with the appropriate plug-in implementation. It has been tested with real data but simulated workloads (i.e., synthetic user base).

Through distributed spatio-temporal and textual analysis, our system provides the following functionalities:

– Socially enhanced search of crisis-related information based on criteria such as location, time, sentiment or a combination of the above.
– Automatic discovery of new Points of Interest (*POIs*) and events that could indicate an emerging crisis of any extent, small or big, ranging from traffic jams and spontaneous gatherings such as protests, to natural disasters or terrorist attacks.
– Inference of the user's semantic trajectory during and after the emergency through the combination of her GPS traces with background information such as maps, check-ins, user comments, etc.
– Semi-Automatic extraction of a user's activity during the crisis in the form of a blog.

An important feature of our system is the automatic POIs detection. A distributed version [8] of DBSCAN, a well-known clustering algorithm, is applied to the GPS traces of our system's users. A dense concentration of traces signifies a POI existence. Furthermore, the correlation of spatio-temporal information provided by the GPS traces with POI related texts automatically produces a blog with the user's activity.

Moreover, a user can search for crisis-related social media information posing both simple as well as more advanced criteria. Simple criteria include commonly

used features such as keywords (e.g., "flood", "terrorism", "traffic", etc.), location (e.g., a bounding box on a map) or a time frame of interest. Advanced criteria refer to data annotations, that derive from the processing of data, such as the sentiment of a tweet or a Facebook post. On top of these, each search can be socially charged, taking into account one's social graph when providing responses (e.g., rank content based on the sentiment of one's friends of it). Thus, the proposed platform is capable of supporting queries such as "Which are the places in Greece where protests are currently taking place and my Facebook friends (or a subset of them) participate in" or "Inform me of the activity and sentiment of my Facebook friends that were near Lesvos island on June 12, 2017 (when the earthquake of 6.1 Richter scale occurred)".

The contribution of this work is many-fold:

– We design a highly scalable architecture that efficiently handles data from heterogeneous sources and is able to deal with big data scenarios.
– We adapt and fine-tune well-known classification and clustering algorithms in a Hadoop-based environment.
– We experiment with datasets in the order of tens of GB, from Tripadvisor, Facebook, Foursquare and Twitter.
– We validate the efficiency, accuracy and scalability of the proposed architecture and algorithms.

The remainder of this paper is organized as follows: Sect. 2 presents the system architecture, Sect. 3 provides an experimental evaluation of the platform, Sect. 4 presents the related work and Sect. 5 concludes our work.

2   Architecture

Our platform follows a layered architecture that is illustrated in Fig. 1. It comprises the *frontend* and *backend* layers, both of which follow a completely modular design to favor flexibility and ease of maintenance.

The frontend layer constitutes the point of interaction with the user and includes all applications related to crisis detection and management that can be supported by the platform. Such an application can be a web, a mobile phone (e.g., Android, iOS) or a native application. To test the platform's basic functionality, a web application has been implemented.

The applications communicate with the backend layer through a REST API. The requests to the backend as well as the responses of the backend back to the user follow a specific JSON format. This feature enables the seamless integration of any client applications with the platform.

The backend is divided in two subsystems, the *processing subsystem* and *storage subsystem*. For the processing subsystem, a Hadoop cluster and a web server farm have been deployed, in order to cater for the special requirements of social network data processing.

Indeed, the volume and velocity of the data produced by social networks demand a distributed approach. Since the Hadoop framework has emerged as
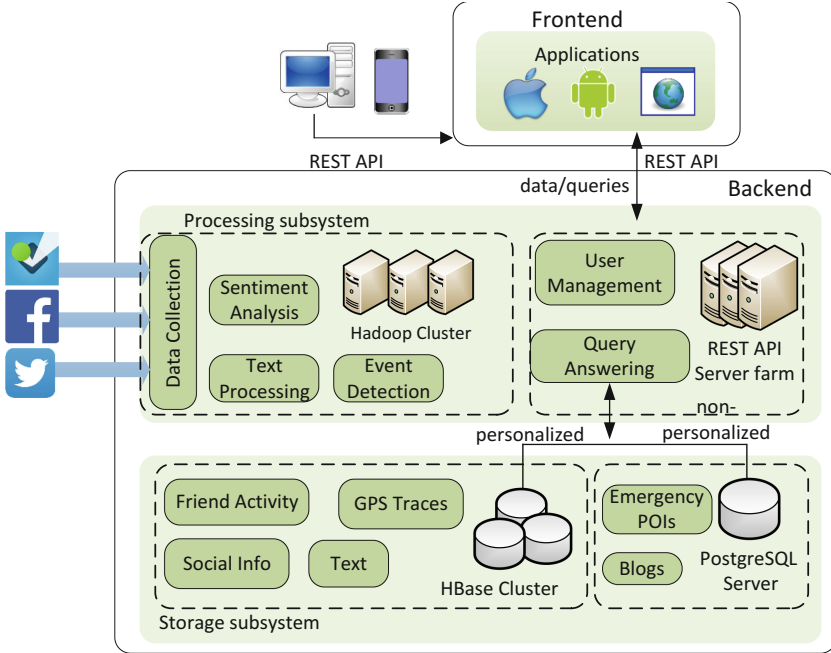
Fig. 1. The proposed platform architecture.

the most prevalent platform of choice for large-scale analytics, we design and deploy the following Hadoop-based processing modules: (a) the *Data Collection* module, (b) the *Sentiment Analysis* module, (c) the *Text Processing* module and (d) the *Event detection* module.

The web server farm hosts the *User Management* and the *Query Answering* modules, which act as gateways to the platform. Both modules are implemented as lightweight web services which put load to the datastore without stressing the web servers.

The storage subsystem is responsible for storing all the data used by our platform, both raw and processed. We refer to the components of the storage subsystem as *repositories*. Repositories are conceptually classified to *primitive* and *non-primitive* data repositories. Primitive repositories store raw, unprocessed data, referred to as *primitive data*. Primitive data are collected from external data sources such as social networks (Facebook, Foursquare, Twitter) and GPS traces and are directly stored to the platform. Non-primitive data repositories are the ones serving answers to queries and hold information extracted from the analysis of primitive data through the use of spatio-textual algorithms.

To handle multiple concurrent users that issue queries to the platform, the system follows a scalable approach. To this end, the Apache HBase [1] NoSQL datastore is used. However, there are queries which require either complex indexing schemes or extended random access to the underlying data. These queries

cannot be efficiently executed in HBase. For this reason, we devise a hybrid architecture that uses HBase for batch queries that can be efficiently executed in parallel and PostgreSQL [6] for online random-access queries that cannot.

In the following, we describe in more detail the repositories and the processing modules of the proposed platform.

2.1   Storage Subsystem Repositories

Emergency POI repository: It is a non-primitive data repository that contains all the information our platform needs to know about POIs where an emergency has occurred. The name of a POI, its geographical location, the keywords characterizing the emergency and the sentiment-related metrics are all stored in this repository. A new entry to the repository can be inserted either explicitly by the user through the web GUI or automatically by the Event Detection module. While POI repository has to deal with low insert/update rates, it should be able to handle heavy, random access read loads. Thus, indexing capabilities are required. PostgreSQL offers such capabilities and has therefore been chosen as the ideal infrastructure for hosting the emergency POI repository.

Social Info repository: This primitive data repository is a HBase-resident table where social graph information is held. For each platform user and for each connected social network, the list of friends is persisted. More specifically, we store a compressed list with the unique social network id, the name and the profile picture of each friend. This list is periodically updated through the Data Collection module to capture possible changes in a user's social graph.

Text repository: The textual data collected from social media an processed through the Text Processing module are stored in this non-primitive data repository. Since the anticipated volume of this data type is high, the Text repository is most demanding in terms of disk space requirements. For this reason, it is stored in HBase and spread across all available cluster nodes. The Text repository holds all the collected comments and reviews that contain crisis related keywords, such as "flood", "hurricane", "traffic", "protest", etc. along with their geo-location. Texts are indexed by user, geo-location and time. For any given crisis-related keyword and any given rectangle on the map, we are able to retrieve the comments that any user made at any given time interval, containing the keywords and geo-located within the search area.

Friend Activity repository: In order to give information based on social friends' activity, we need to keep track of all locations (Emergency POIs) and social media content of a user's friends. This information is maintained in the Friend Activity repository, a non-primitive data repository persisted as an HBase table. Each activity is represented by an *activity* data structure with the complete Emergency POI information (name, latitude, longitude, etc.) as well as the textual content of possible posts. Moreover, this structure is enriched with sentiment metrics (positive/negative) through the Sentiment Analysis module. Every time a user or a user's social friend is tracked near an Emergency POI through her GPS trace or geo-tagged posts, an activity struct indexed by user and time is added to the repository. Thus, for any given time interval, we know

the places that all of a user's friends have been and a score indicating each friend's sentiment.

An obvious remark is that the activity struct introduces high data redundancy, since each time someone visits an Emergency POI, the whole POI information is registered within the struct. The alternative schema design strategy would be joining POI information with activity information at query time. However, our experiments suggest data replication to be more efficient. Our schema in combination with HBase coprocessors and a fully parallel query mechanism offers higher scalability and achieves lower latency in the event of many concurrent users, which is the case in emergency situations. Thus, we sacrifice cheap storage space for efficiency.

GPS Traces repository: Mobile devices with appropriate geo-location applications supported by our platform installed, can push their GPS traces to the platform. These traces are stored in the primitive data repository of GPS Traces. Since the platform may continuously receive GPS traces, this repository is expected to deal with a high update rate. Furthermore, as GPS traces are not queried directly by the users but are periodically processed in bulk, there is no need to build indices on them. The volume of data, the opportunities for parallel bulk processing and the absence of indices are the main reasons why we choose HBase as the storage substrate for GPS data.

Blogs repository: We define a semantic trajectory to be a timestamped sequence of Emergency POIs summarizing user's activity during crisis this information is stored in a non-primitive data repository, the Blogs repository. As POIs, blogs are frequently queried by users but they do not have to deal with heavy updates and thus are stored as a PostgreSQL resident table.

2.2   Processing Subsystem Modules

User Management module: The User Management module is responsible for the user authentication to the platform. The user is registered either through the mobile applications or the website. The signing-in process is carried out only with the use of the social network credentials. The registration workflow follows the OAuth protocol. The OAuth authorization framework enables a third-party application to obtain access to an HTTP service on behalf of a resource owner. When the authentication is successful, the user logs in and an access token is returned to the platform. With this token, the platform can interact with the connected social networks on behalf of the end user. It can monitor user's activity, user's friends activity, posts etc. When multiple social networks are connected to the platform, the platform joins the acquired data and enriches the information that is indexed and stored.

Data Collection module: The functionality of this module is to collect data from external data sources. Periodically, the Data Collection module scans in parallel all the authorized users of the platform; each worker scans a different set of users. For each user and for all connected social networks, it downloads all the interesting updates from the user's social profile. Since the platform provides

social geo-location services, interesting updates are considered to be user check-ins and the accompanying comments, status updates and geo-located tweets. From this information, the platform is able to gain knowledge about the existence of crisis events and people's sentiment about them. Once data are streamed to the platform, part of them are indexed and stored in the primitive data repositories, while the rest are processed in-memory and then indexed and stored to the appropriate non-primitive data repositories.

Text Processing module: This module indexes all textual information collected by the Data Collection module according to predefined crisis related keywords (e.g., flood, earthquake, protest, etc.). To do so, it employs standard Natural Language Processing (NLP) techniques (lemmatization, stemming, etc.) to track the predefined keywords and stores the results to the Text repository.

Sentiment Analysis module: The Sentiment Analysis module performs sentiment analysis to all textual information the platform collects through the Data Collection module. Comments are classified, real-time and in-memory, as positive or negative. The score which results from the sentiment analysis is persisted to the datastore along with the text itself.

As a classification algorithm, we choose the Naive Bayes classifier that the Apache Mahout [2] framework provides. Naive Bayes is a supervised learning algorithm and thus it needs a pre-annotated dataset for its training. For the training, data from Tripadvisor, containing reviews for POIs is used. The reason for choosing this training set is that Tripadvisor comments are annotated with a rank from 1 to 5 that can be used as a classification score. After an extensive experimental study and a fine-tuning of the algorithm parameters, we managed to create a highly accurate classifier that achieves an accuracy ratio of 94% towards unseen data.

Event Detection module: The detection of new events and emergency POIs constitutes a core functionality of our platform. A distributed, Hadoop-based implementation of the DBSCAN clustering algorithm [8] is employed for this reason. The module is called periodically and processes in parallel the updates of the GPS Traces repository in order to find traces of high density; high density traces imply the existence of a new POI. In order to avoid detecting already known Emergency POIs, traces falling near to existing Emergency POIs in the repository are filtered out and are not taken into consideration for clustering.

Query Answering module: The Query Answering is the module that executes search queries. A search query can take as input the following parameters:

– a bounding box of coordinates (i.e., on a map)
– a list of keywords
– a list of social network friends
– a time window
– results sorting criteria
– the number of results to be returned

Queries are distinguished in *personalized* and *non-personalized* ones. Personalized are the queries that exclusively concern the social media friends (or

a subset of them) of a user. Thus, if a list of friends is provided, the query is considered to be *personalized*.

For personalized queries, the selected friends' activities should be taken into account. The repository that maintains such personalized information is the Friend Activity repository which resides in HBase. Thus, as Fig. 1 depicts, personalized queries are directed towards the HBase cluster. In order to efficiently resolve personalized queries, HBase coprocessors are used. Each coprocessor is responsible for a region of the Friend Activity repository and performs HBase *get* requests to the users under its authority. Since different friends are located with high probability in different regions, a different coprocessor is in charge of serving their activities and multiple get requests are issued in parallel. Increasing the number of regions leads to an increase in the number of coprocessors and thus a higher degree of parallelism is achieved within a single query.

On the other hand, non-personalized queries involve repositories that handle no-personalized information. The Emergency POI repository, which contains all the required information, resides in PostgreSQL. Thus all non-personalized queries are translated to *select* SQL queries and directed towards PostgreSQL.

3 Experiments

We experimentally evaluate our system in terms of performance, scalability and accuracy of its modules. This way we validate both our architectural design and the selected optimizations.

3.1 Scalability and Performance Evaluation

We first present some experiments for the scalability and performance of the query answering module. Using a synthetic dataset, we test the ability of the platform to respond to personalized queries issued by its users for various loads and different cluster sizes. Each personalized query involves a set of friends and returns the activities, i.e., text, geo-location and sentiment related to a crisis situation designated by a keyword (e.g., earthquake). The platform user can set a number of other parameters as well: the geographical location, the time period of the activities, etc. In our experiments, we identified that the dominant factor in the execution time of the query is the number of social network friends that the platform users define.

For the synthetic dataset generation, we collected information from Open-StreetMap about 8500 POIs (which we considered Emergency POIs for out experiment) located in Greece. Based on those POIs, we emulated the activity of 150k different social network users, each of whom has performed activities in a number of Emergency POIs and assigned a sentiment to it. The number of activities for each social network friend follows the Normal Distribution with $\mu = 170$ and $\sigma = 10$[1]. The dataset is deployed into an HBase cluster consisting

---

[1] The vast majority of the users has performed between 140 and 200 visits in different POIs.

of 16 dual-core VMs with 2 GB of RAM each, running Linux (Ubuntu 14.04). The VMs are hosted in a private Openstack cluster.

At first, we are going to study the impact of the number of social network friends into the execution time of a single query. At this point we will also examine how the cluster size affects the execution time of a query. Secondarily, we are going to extend our study to multiple concurrent queries where we will also examine the behavior of the platform for different numbers of concurrent queries and different cluster configurations.

For the first point, we evaluate the execution time of a query for different numbers of friends. In Fig. 2 we provide our findings. In this experiment, we executed one query at a time involving from 500 to 10 k social network friends for three different cluster setups consisting of 4, 8 and 16 nodes. The friends for each query are picked randomly in a uniform manner. We repeated each query ten times and we provide the average of those runs.
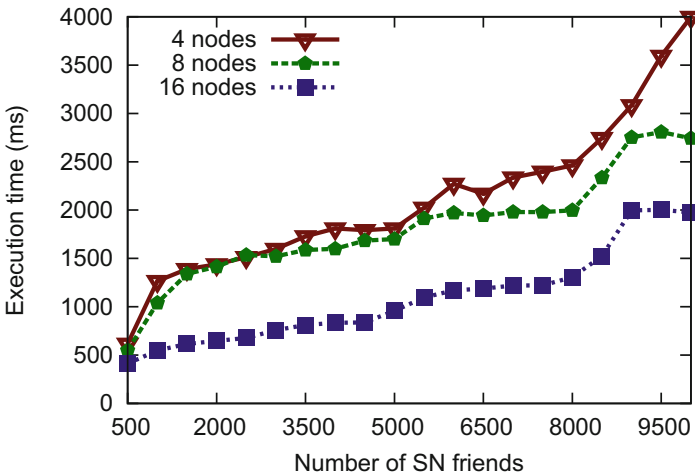

Fig. 2. Query latency vs number of users

The number of friends affects the execution time in an almost linear manner. Furthermore, an increase in the cluster size leads to a latency decrease, since the execution is happening in parallel to multiple nodes. By utilizing HBase coprocessors, we managed to exploit the locality of the computations into specific portions of the data: each coprocessor operates into a specific HBase region (holding a specific portion of the data), eliminates the activities that do not satisfy the user defined criteria, aggregates multiple activities referring to the same Emergency POI and annotates them with aggregated sentiment scores. Finally, each coprocessor returns to the Web Server the list of emergency POIs, the related activities and sentiment and the Web Server, in turn, merges the results and returns the final list to the end user.

Using the previously described technique, we achieved latencies lower than 1 second for more than 5000 users. Bearing into consideration that social networks like Facebook, retain a limit on the maximum number of connections (5000 friends per user), we can guarantee that the latency for each query remains acceptable for a real time application.

We now extend our analysis for the cases where multiple queries are issued concurrently to the platform. For our experiments we create a number of concurrent queries involving 6000 social network friends each and we measure their execution time for different cluster sizes. In Fig. 3, we provide our results. The execution time in the vertical axis represents the average execution time for each case.
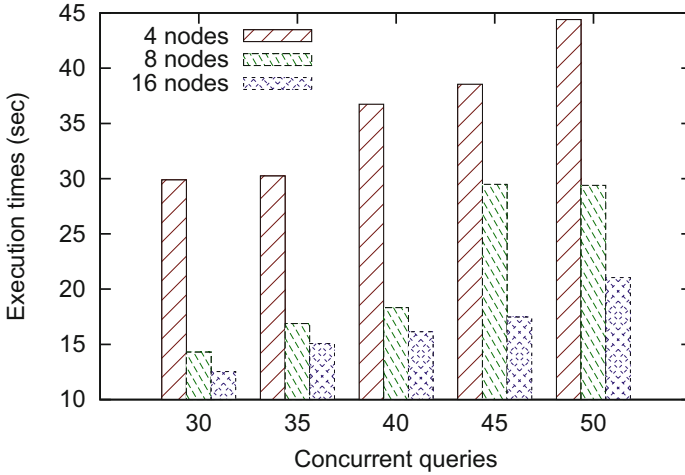

Fig. 3. Average execution time for concurrent queries

As Fig. 3 demonstrates, an increase in the number of concurrent queries leads to worse performance (larger execution time). However, for larger cluster sizes we can make the following observations: (a) even for the lowest number of concurrent queries the 16 cluster case is approximately 2.5 times better than the 4 cluster case, indicative of the proper utilization of more resources and (b) larger cluster sizes do not allow execution time to rise fast as the number of concurrent queries increases. Specifically, when the cluster consists of 4 nodes, the execution time is high even for the lowest number of concurrent queries and it continues to rise rapidly while the number of queries is increased. In the 8 nodes case, although at first the achieved execution times are relatively low, the increase becomes rapid for more concurrent queries, whereas in the 16 nodes case we see that the increase is held in a minimum level. This is indicative of the scalability of the platform, since more resources are properly utilized and the platform becomes resistant to concurrency.

Finally, since greater number of concurrent queries leads to more threads in the Web Server which, in turn, hits the cluster, we can avoid any potential bottlenecks by replicating the Web Servers while simultaneously, we use a load balancer to route the traffic to the web servers accordingly. In our experimental setup, we identified that two 4-cores web servers with 4 GB of RAM each are more than enough to avoid such bottlenecks.

3.2   Evaluation of Sentiment Analysis

In this section we evaluate the tuning of the Naive Bayes classifier we use for sentiment analysis. As we have already mentioned, for the training of the classifier, we crawl and use data from Tripadvisor. We consider a Tripadvisor review about a place to be a classification document. We divide training documents into two sets: positive and negative opinion documents. Both sets should have almost the same cardinality. Before feed the training set to the classifier, a preprocessing step is applied which involves stemming, turning all letters to lowercase and removing all words belonging to a list of stopwords. When the preprocessing step is finished, Naive Bayes is applied to the data. Let this procedure be the baseline training process. As a next step, we experiment with the following optimizations: use of the *tf* metric, *2-grams*, *Bi-Normal Separation* and deletion of words with less than x occurrences. These optimizations can be given as parameters to the classification algorithm. Experiments with different combinations for the algorithm parameters were also conducted but are omitted due to space constraints.

Figure 4 shows the classification accuracy for various training set sizes, when the baseline and the optimized classification are used. We observe that when optimizations are applied, classification results are more accurate for any training size. Especially, for a training set of 500 k documents, we achieve an accuracy of 93.8%. As we can see, the 500 k documents form a threshold for the classifier. For both versions of the algorithm after this point accuracy degrades. This is because there is an overfit of data, which is a classic problem in Machine Learning approaches.

4   Related Work

This work is not the first that proposes the use of social network [9,10,14,15] or mobile phone data [13] for crisis response and management. AIDR [9] is a platform that classifies tweets into user-defined, crisis-related categories in a semi-automatic way (classifiers are built using crowdsourced annotations). The work in [15] focuses on identifying timely and relevant responders for questions in social media in order to help people affected by natural calamities. The authors of [10] employ unsupervised learning techniques on textual data to extract knowledge and provide an overview of the crisis while the study in [13] correlates CDR
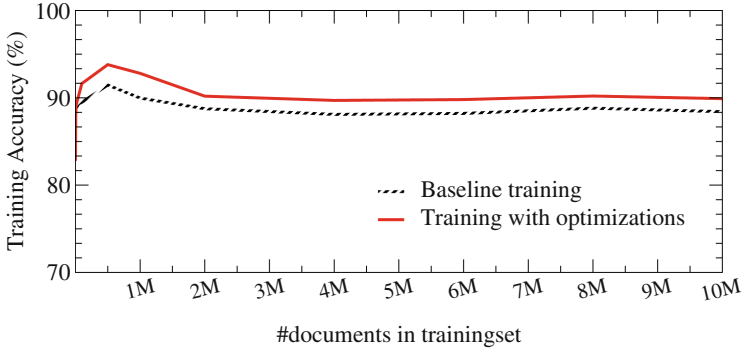
Fig. 4. Classification accuracy for different training sizes

data with rainfall levels, suggesting the potential in using cell tower activity information to improve early warning and emergency management mechanisms in case of floodings.

Contrarily, our platform focuses on combining textual as well a spatial data (GPS traces) from multiple social networks to (a) detect possible emergency POIs and to (b) provide personalized information about a crisis situation. Furthermore, our platform relies on distributed processing and storage solutions and thus proves to be scalable, while no implementation and performance information is provided for the other systems. Our platform is based on prior work of our team, which resulted in a socially enhanced recommendation system [11,12]. This has been adapted to cater the specific requirements of crisis detection and management.

5   Conclusions

In this paper we presented a storage and processing platform that is able to support applications and services that leverage the power of Big Data produced by mobile and social network users to detect and manage emergencies. Such data include spatio-temporal and textual information, which can be combined to automatically discover POIs and events that could indicate an emerging crisis of any extent, provide crisis-related information based on criteria such as location, time, sentiment or a combination of the above and infer a user's semantic trajectory during and after the emergency. Our prototype, which currently supports Facebook, Twitter and Foursquare, is able to provide query latencies of a few seconds even under heavy load, falling into the sub-second scale when executing over a 16-node cluster. Releasing an online public version of our system and testing it under real world conditions is part of our future plans.

References

1. Apache HBase. http://hbase.apache.org
2. Apache Mahout. https://mahout.apache.org/
3. Digital in 2017: Global Overview. https://wearesocial.com/special-reports/digital-in-2017-global-overview
4. Facebook Safety Check. https://www.facebook.com/about/safetycheck/
5. Facebook Stats. https://newsroom.fb.com/company-info/
6. Postgresql. http://www.postgresql.org/
7. Twitter Usage Statistics. http://www.internetlivestats.com/twitter-statistics/
8. He, Y., Tan, H., Luo, W., Mao, H., Ma, D., Feng, S., Fan, J.: MR-DBSCAN: an efficient parallel density-based clustering algorithm using mapreduce. In: ICPADS (2011)
9. Imran, M., Castillo, C., Lucas, J., Meier, P., Vieweg, S.: AIDR: artificial intelligence for disaster response. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 159–162. ACM (2014)
10. Lazreg, M.B., Goodwin, M., Granmo, O.-C.: 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, Malmö, Sweden, p. 31 (2016)
11. Mytilinis, I., Giannakopoulos, I., Konstantinou, I., Doka, K., Koziris, N.: MoDisSENSE: a distributed platform for social networking services over mobile devices. In: 2014 IEEE International Conference on Big Data (Big Data), pp. 49–51. IEEE (2014)
12. Mytilinis, I., Giannakopoulos, I., Konstantinou, I., Doka, K., Tsitsigkos, D., Terrovitis, M., Giampouras, L., Koziris, N.: MoDisSENSE: a distributed spatio-temporal and textual processing platform for social networking services. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 895–900. ACM (2015)
13. Pastor-Escuredo, D., Morales-Guzmán, A., Torres-Fernández, Y., Bauer, J.-M., Wadhwa, A., Castro-Correa, C., Romanoff, L., Lee, J.G., Rutherford, A., Frias-Martinez, V., et al.: Flooding through the lens of mobile phone activity. In: 2014 IEEE Global Humanitarian Technology Conference (GHTC), pp. 279–286. IEEE (2014)
14. Purohit, H., Sheth, A.P.: Twitris v3: from citizen sensing to analysis, coordination and action. In: ICWSM (2013)
15. Ranganath, S., Wang, S., Hu, X., Tang, J., Liu, H.: Facilitating time critical information seeking in social media. IEEE Trans. Knowl. Data Eng. **PP**(99), 1–14 (2017)
16. Zielinski, A., Middleton, S.E., Tokarchuk, L.N., Wang, X.: Social media text mining and network analysis for decision support in natural crisis management, pp. 840–845. Proc. ISCRAM. Baden-Baden, Germany (2013)