On the Support of a Similarity-enabled Relational Database Management System in Civilian Crisis Situations

Paulo H. Oliveira, Antonio C. Fraideinberze, Natan A. Laverde, Hugo Gualdron, Andre S. Gonzaga, Lucas D. Ferreira, Willian D. Oliveira, Jose F. Rodrigues-Jr., Robson L. F. Cordeiro, Caetano Traina Jr., Agma J. M. Traina and Elaine P. M. Sousa

Institute of Mathematics and Computer Sciences, University of Sao Paulo, Av. Trabalhador Sancarlense, 400, Sao Carlos, SP, Brazil

Keywords: Crisis Situation, Crisis Management, Relational Database Management System, Similarity Query.

Abstract: Crowdsourcing solutions can be helpful to extract information from disaster-related data during crisis management. However, certain information can only be obtained through similarity operations. Some of them also depend on additional data stored in a Relational Database Management System (RDBMS). In this context, several works focus on crisis management supported by data. Nevertheless, none of them provide a methodology for employing a similarity-enabled RDBMS in disaster-relief tasks. To fill this gap, we introduce a methodology together with the Data-Centric Crisis Management (DCCM) architecture, which employs our methods over a similarity-enabled RDBMS. We evaluate our proposal through three tasks: classification of incoming data regarding current events, identifying relevant information to guide rescue teams; filtering of incoming data, enhancing the decision support by removing near-duplicate data; and similarity retrieval of historical data, supporting analytical comprehension of the crisis context. To make it possible, similarity-based operations were implemented within one popular, open-source RDBMS. Results using real data from Flickr show that our proposal is feasible for real-time applications. In addition to high performance, accurate results were obtained with a proper combination of techniques for each task. Hence, we expect our work to provide a framework for further developments on crisis management solutions.

1 INTRODUCTION

Crisis situations, such as conflagrations, disasters in crowded events, and workplace accidents in industrial plants, may endanger human life and lead to financial losses. A fast response to this kind of situation is essential to reduce or prevent damage. In this context, software systems aimed at supporting experts in decision-making can be used to better understand and manage crises. A promising line of research is the use of social networks or crowdsourcing (Kudyba, 2014) to gather information from the crisis site.

Several desirable tasks can be performed by software systems designed for aiding in decision-making during crises. One of such tasks is to detect the evidences that best depict the crisis situation, so that rescue teams can be aware of it and prepare themselves properly. For instance, identifying fire or smoke on multimedia data, such as images, videos or textual reports, usually points to conflagration. Some relevant proposals in this direction comprehend fire and smoke detection based on image processing approaches (Celik et al., 2007) and techniques for fire detection designed over image descriptors that focus on detecting fire from social media images (Bedo et al., 2015).

Another important task is to filter the information received from crowdsourcing solutions dedicated to collecting data from crises. When reporting incidents, users might end up sending too much similar information, such as pictures from the same angle of the same object. Such excess of similar data demands a longer time to be processed. Moreover, it turns the decision-making process more time-consuming. Therefore, removing duplicates is an essential task in this context.

The task of searching for similar data in historical databases can support decision-making as well. Take for instance a database that contains images and textual descriptions regarding past crisis situations. If the crowdsourcing system gets, for instance, images depicting fire, a query might be posed on the database to

Oliveira, P., Fraideinberze, A., Laverde, N., Gualdron, H., Gonzaga, A., Ferreira, L., Oliveira, W., Rodrigues-Jr., J., Cordeiro, R., Jr., C., Traina, A. and Sousa, E.
On the Support of a Similarity-enabled Relational Database Management System in Civilian Crisis Situations.
In *Proceedings of the 18th International Conference on Enterprise Information Systems (ICEIS 2016) - Volume 1*, pages 119-126
ISBN: 978-989-758-187-8

retrieve similar images and the corresponding textual descriptions. Then, based on these results, specialists would potentially infer the kind of material burning in the crisis, by analyzing the color tone of the smoke in the retrieved images and their textual descriptions.

For all those tasks, it is desirable that a commodity system provide functionalities over existing software infrastructure. Commodity systems that can play this role are the Relational Database Management Systems (RDBMS). They are largely available in the current computing technology and are able to bring new functionalities without the need of redesigning the existing software. Moreover, RDBMS provide efficient data storage and retrieval. However, they do not readily support similarity operations, which are needed to address the aforementioned tasks.

Several works in the literature aim at embedding similarity support in RDBMS. Nevertheless, the literature lacks a methodology for employing a similarity-enabled RDBMS in the context of crisis management. This work aims at filling that gap. Our hypothesis is that providing similarity support on an RDBMS helps the decision support in crisis situations.

We contribute with a data-centric architecture for decision-making during crisis situations by means of a similarity-enabled RDBMS. Our proposal is evaluated using an image dataset of real crises from Flickr in performing three tasks:

- **Task 1.** Classification of incoming data regarding current events, detecting the most relevant information to guide rescue teams in the crisis site;
- **Task 2.** Filtering of incoming data, enhancing the decision support of rescue command centers by removing near-duplicate data;
- **Task 3.** Similarity retrieval from past crisis situations, supporting analytical comprehension of the crisis context.

This work has been conducted to cater to demands of the project *RESCUER: Reliable and Smart Crowdsourcing Solution for Emergency and Crisis Management*[1], supported by the European Union's Research and Innovation Funding Program FP7.

The results of our experimentation show that the proposed architecture is effective over crisis scenarios which rely on multimedia data. In addition to the high performance achieved, accurate results are obtained when using a proper combination of techniques.

The rest of the paper is structured as follows. Section 2 presents the related work and Section 3 presents the main concepts for similarity support on RDBMS. Section 4 describes the new Data-Centric Crisis Management architecture, on which the proposed methodology is based. Section 5 presents our methodology, describes the experiments and discusses the results. Finally, the conclusions are presented in Section 6.

[1] http://www.rescuer-project.org/

2 RELATED WORK

Existing research on crisis management highlights the importance of computer-assisted systems to support this task. The approaches may be categorized into different types according to their purpose.

One type refers to localization, whose purpose is to determine where victims are located during a disaster. There are works that accomplish this task by employing cell phone localization techniques, such as International Mobile Subscriber Identity (IMSI) catchers (Reznik et al., 2015).

Another type regards logistics. Examples comprehend an integer programming technique for modeling multiple-resource emergency responses (Zhang et al., 2012) and a methodology for routing rescue teams to multiple communities (Huang et al., 2013).

A different line of work refers to decision-making based on social media (Gibson et al., 2014). Most of the work focus on textual data, specially from services like Twitter (Ghahremanlou et al., 2015).

Although all the aforementioned approaches have been conceived to cater to different requirements, all of them share the characteristic of using Information Communication Technology (ICT) in response to crisis situations. The decision-making systems based on incoming data have one more characteristic: the participation of people somehow involved in the disaster.

Existing work have focused on the importance of crowdsourcing data for crisis management in the post-2015 world (Halder, 2014). Therefore, to describe our methodology, we assume the existence of crowdsourcing as a subsystem dedicated to gathering input data. Additionally, we assume the existence of a command center, where analysts evaluate the input data in order to guide the efforts of a rescue team at the crisis site.

The work of Mehrotra (Mehrotra et al., 2004) is the closest approach with respect to our methodology. That work presents an interesting approach, but it focuses mostly on textual data and spatial-temporal information, rather than on more kinds of complex data, such as images. Furthermore, it lacks a methodology for employing content-based operations. We fill those gaps by providing a methodology to perform such operations over disaster-related data and provide useful information to rescue teams.

3 BACKGROUND

3.1 Content-based Retrieval

Complex data is a common term associated with objects such as images, audio, time series, geographical data and large texts. Such data do not present *order relation* and, therefore, are unable to be compared by relational operators ($<$, $\leq$, $\geq$, $>$). Equality operators ($=$, $\neq$) could be used, but they have little or no meaning when employed on such data. Nevertheless, complex data can be compared according to their content by using *similarity* concepts (Barioni et al., 2011).

The interaction with a content-based retrieval system starts as the user enters a query, providing a complex object as the query example. This complex object is submitted to a feature extractor, which extracts representative characteristics from it and generates a feature vector. The feature vector is sent to an evaluation function, which compares another feature vector stored in the database and returns a value representing the dissimilarity degree (also known as the distance) between both feature vectors. This comparison is repeated over the database, generating the results at the end of the process and sending them to the user.

Two of the most common queries used in content-based retrieval are the Range Query and the k Nearest Neighbor (kNN) Query (Barioni et al., 2011). Range Query is defined by the function $Rq(s_q, \xi)$, where s_q represents an object from data domain $\mathbb{S}$ and ξ is the radius used as distance constraint. The query returns all objects within a distance ξ from s_q. kNN Query is defined by the function $kNNq(s_q, k)$, where s_q represents an object from the data domain $\mathbb{S}$ and k is the number of elements to be returned. The query returns the k most similar objects to s_q. The kNN Queries are employed in the context of Instance-Based Learning (IBL) algorithms, such as the kNN Classifier, which is used in our proposal and thus discussed in Section 3.2.

The feature extraction is usually required because the original representation of a given complex object is not prone to useful and efficient computation. Evaluation functions are able to compute the dissimilarity degree of a pair of feature vectors. These subjects are discussed in Sections 3.3 and 3.4.

The similarity retrieval process can be performed outside an RDBMS. However, enabling an RDBMS with similarity is a promising approach and there are several ways for doing so, as discussed in Section 3.5.

3.2 kNN Classifier

The concept of Instance-Based Learning (IBL) (Aha et al., 1991) comprehends supervised learning algorithms that make predictions based solely on the instances previously stored in the database. In these algorithms, no model is built. The knowledge is represented by the data instances already stored and classified. Then, new instances are classified in relation to the existing stored instances, according to their similarity. One of the main IBL algorithms is the well-known kNN Classifier (Fix and Hodges Jr., 1951).

For a given unlabeled instance, the kNN Classifier retrieves from the database the k most similar instances. Following, it predicts the label based on the retrieved instances, according to some predefined criterion. A simple one is to assign the label of the prevailing class in the k nearest neighbors. Another one is to weigh the retrieved instances by distance, so the closest ones have a higher influence.

3.3 Feature Extractors

One of the main tasks for retrieving complex data by content is the feature extraction process, which maps a high-dimensional input data into a low-dimensional feature space, extracting useful information from raw data while reducing their content. Using proper feature extractors for a complex data domain leads to results closer to what the users expect (Sikora, 2001).

Several feature extractors have been developed for different application domains. The main characteristics investigated in the context of images are color, texture and shape. There are several feature extractors for such characteristics, some of which are part of the MPEG-7 standard (MultiMedia, 2002). In this work, we employ a color-based and a hash-based extractors.

Color-based Extractors. Color-based extractors are commonly used as a basis for other extractors. For this reason, they are the most used visual descriptors in content-based image retrieval. The color-based feature extractors in the MPEG-7 standard are commonly employed in the literature. One of them is the Color Structure Descriptor, which builds a color histogram based on the local features of the image.

Perceptual Hash. An extractor suitable for near-duplicate detection is the Perceptual Hash[2]. It generates a "fingerprint" of a multimedia file derived from various features from its content. These "fingerprints" present the characteristic of being close to one another if the extracted features are similar.

3.4 Evaluation Functions

The dissimilarity between two objects is usually determined by a numerical value obtained from an eval-

[2]http://www.phash.org/

uation function. Objects with smaller values are considered to be more alike.

The Minkowski Family comprehends evaluation functions known as L_p metrics that are widely used in content-based retrieval (Wilson and Martinez, 1997). The L_1 metric corresponds to the Manhattan Distance, also noted as City-Block Distance. The L_2 metric is the well-known Euclidean Distance. Finally, there is the L_∞ metric, also noted as the Chebyshev Distance.

The Hamming Distance (Hamming, 1950), which is another well-known evaluation function, counts the substitutions needed to transform one of the input data into the other. It can be employed in near-duplicate detection tasks, since combining it with the Perceptual Hash leads to accurate results.

3.5 Similarity Support on RDBMS

SimDB (Silva et al., 2010) is a similarity-enabled RDBMS, based on PostgreSQL. The similarity operations and keywords were included in its core. Equivalence rules were also included, which allows alternative query plans. However, similarity queries are only available for numerical data, and traditional queries over other data types are not supported.

SIREN (Barioni et al., 2011) is a middleware between a client application and the RDBMS. The client sends SQL commands extended with similarity keywords, which are checked by SIREN to identify similarity predicates. First, SIREN evaluates the similarity predicates, accessing an index of feature vectors, then uses the RDBMS for traditional predicates.

FMI-SiR (Kaster et al., 2011) is a framework that operates over the RDBMS Oracle, employing user-defined functions to extract features and to index data. MedFMI-SiR is an extension of FMI-SiR for medical images in the Digital Imaging and Communications in Medicine (DICOM) format.

SimbA (Bedo et al., 2014) is a framework that extends the middleware SIREN. SimbA supports the inclusion and combination of feature extractors, evaluation functions and indexes on demand. The queries are processed just like on SIREN.

4 PROPOSED ARCHITECTURE

This section presents our architecture for crisis management, named as Data-Centric Crisis Management (DCCM). Section 4.1 describes the scenario of a typical crisis situation managed by DCCM. Then, Section 4.2 describes our architecture.

4.1 Crisis Management Scenario

Figure 1 shows the scenario of a crisis situation supported by DCCM.

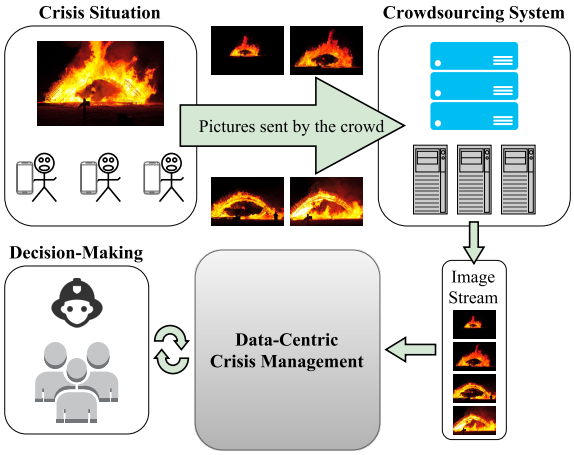

Figure 1: Scenario of a typical crisis situation considering our architecture for crisis management.

In a *Crisis Situation*, eyewitnesses can collect data regarding the event. For instance, they can take pictures, record videos and make textual reports, which are sent to the *Crowdsourcing System*. In Figure 1, the pictures taken by the eyewitnesses are redirected as an *Image Stream* to DCCM. Then, the command center can query DCCM for the *Decision-Making* process.

Additionally, the crowdsourcing system could receive other data, such as metadata (e.g. time and GPS localization) or other data types (e.g. video and text).

4.2 Data-centric Crisis Management

The Data-Centric Crisis Management (DCCM) architecture is represented in Figure 2. The whole mechanism has three processes, each of them depicted in the figure by arrows marked with the letters A, B and C, which represent the tasks introduced in Section 1.

In a crisis situation, we consider the existence of a crowdsourcing system that receives disaster-related complex objects (A1) and submits them to DCCM.

Each object of the data stream is placed in a *Buffer* and analyzed by the *Filtering Engine*. First, the engine checks whether the object is a near duplicate of some other object currently within the *Buffer*. For the near-duplicate checking, the *Filtering Engine* uses the *Similarity Engine* (A2) to extract a feature vector from the object and compare it to the feature vectors of the other objects within the *Buffer*. The object is marked as a near duplicate when its distance from at least another object is at most ξ, which is a threshold defined by specialists according to the application domain.

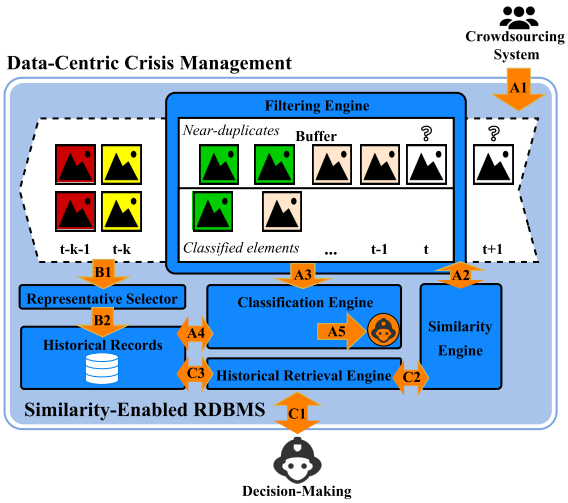

Figure 2: The DCCM architecture, consisting of the tasks: classification (A), filtering (B) and historical retrieval (C).

If the object is not a near duplicate, then it is submitted to the *Classification Engine* (A3). The classification process uses *Historical Records* in a database to train classifier methods (A4). Based on such training, the *Classification Engine* labels the object regarding the event it represents. For instance, it can be labeled as "fire" or "smoke". Finally, the *Classification Engine* notifies the specialists with the now-classified object for the *Decision-Making* process (A5).

If the object is a near duplicate, then it is not submitted to the *Classification Engine*. Instead, it stays in the *Buffer* to be compared to others that come later. Moreover, it is associated with the event of the object of which it is a near duplicate. In Figure 2, the objects from the same event have the same color. The white objects marked with "?" have not been analyzed yet.

The *Buffer* may be determined either by a physical size, such as the number of elements it holds, or by a time window. In Figure 2, it is delimited by a time window of length k, beginning at time t and ending at time $t-k$. The *Buffer* is flushed at every k-th time instant. Before flushing, the *Representative Selector* selects the object of each group that best represents its event (B1) according to a predefined criterion.

If a near-duplicate object is selected as the representative, then it gets the label of the classified object of its group. The already-classified object, in turn, is marked as near duplicate. On the other hand, if the selected representative is already the classified object of its group, then no changes are made. Lastly, the classified objects are stored in the database and the near duplicates are discarded, flushing the *Buffer* (B2).

There is another use case for DCCM, which refers to the historical analyses. The *Decision-Making* team may want to provide complex data samples to retrieve similar events from the past. For this purpose, DCCM provides the *Historical Retrieval Engine* (C1). First, the engine extracts the features from each provided sample (C2). Then, it compares the extracted features against the *Historical Records* and provides its findings to the *Decision-Making* team (C3).

5 CASE STUDY

In this section, we present the case study for evaluating the DCCM architecture over the three tasks discussed earlier. The experiments were carried out over a real crisis dataset known as Flickr-Fire (Bedo et al., 2015) containing 2,000 images extracted from Flickr, 1,000 labeled as "fire" and 1,000 as "not fire".

5.1 Implementation of DCCM

To implement DCCM, we extended the open-source RDBMS PostgreSQL. Our implementation, named as **Kiara**, supports an SQL extension for building similarity queries over complex data (Barioni et al., 2011). Also through the SQL extension, Kiara allows managing feature extractors and evaluation functions, which are dynamically (no recompilation) inserted and updated by user-defined functions written in C++. Kiara makes use of metatables to keep track of feature extractors and evaluation functions associated with the attributes of complex data that a user instantiates.

To support the SQL extension, we built a parser that works like a proxy in the core of Kiara. It receives a query and rewrites only the similarity predicates, expressing them through standard SQL operators. Then, it sends the rewritten queries to the core of Kiara.

After inserting a new extractor, the features are automatically extracted from the complex data (e.g. image, video or text) and then stored in user-defined attributes dedicated to representing such data. Similarity queries can be included through PL/pgSQL functions and new indexes can be included through the interface known as Generalized Search Tree (GiST), already present in PostgreSQL. Moreover, Kiara allows exploring alternative query plans involving traditional and similarity predicates.

5.2 Classification of Incoming Data

5.2.1 Methodology

Classifying disaster-related incoming data is helpful because of two reasons. One of them is to identify the characteristic that best depicts the crisis situation. The other is to store data properly labeled, which improves

further queries on a historical database. To do so, the DCCM architecture employs the *k*NN Classifier.

The parameter k can be selected arbitrarily. However, too small values can be noise-sensitive, whereas too large values allow including more instances from other classes, leading to misclassified instances.

5.2.2 Experimentation and Results

In this task, we classified the elements of the Flickr-Fire dataset. For a robust experimentation, we used the procedure *10-fold cross-validation* for a *k*NN classifier using $k = 10$. We used the Manhattan Distance and the extractor Color Structure Descriptor because existing work showed that they allow accurate results for fire detection (Bedo et al., 2015).

After 10 rounds of evaluation, we took the average accuracy and the average F1 score. The result was the same for both measures, which was 0.86. Considering a real event, this capability would be able to automatically group data according to their content, indicating the main characteristics of the crisis and thus saving the command center crucial time for fast response.

5.3 Filtering of Incoming Data

5.3.1 Methodology

In the task of filtering, we are interested in preventing duplicate information from being classified and subsequently sent to the command center.

To determine whether the incoming data is a near duplicate of existing data, they must be compared by their content. For this purpose, we must employ similarity queries. In this case, though, we are restricted to Range Queries. If the new object is a near duplicate of an object in the buffer, then the distance from each other is at most ξ, which is supposed to be a small threshold (range), since we want to detect pairs of objects that, in essence, represent the same information. Range Queries allow restricting results based on their similarity, differently from *k*NN Queries, which do it by the number of objects retrieved.

Hence, the DCCM architecture prevents near duplicates by using Range Queries. Each object that arrives in the buffer is submitted to a default feature extractor. Then, a Range Query is performed by using the extracted features as the s_q object. The range value ξ must be predefined as well, according to the application domain. If at least one object from the buffer is retrieved by the Range Query, then the s_q object is marked as a near duplicate.

5.3.2 Experimentation and Results

For this experiment, we employed the Hamming Distance with the Perceptual Hash extractor and assumed a buffer size of 80. We filled the buffer with 80 images from Flickr, of which 37 depict "small fire" events and 43 depict "big fire" events. Each of the 80 images was used as the s_q to a Range Query with $\xi = 10$.

The ξ parameter was set to retrieve around half of the images (40 images approximately), in order to be able to return an entire class ("big fire" or "small fire") of images. This allows evaluating the precision of the queries with the Precision-Recall method.

Figure 3 shows the Precision-Recall curve for this set of queries. The curve falls off only after 80% of retrieval. This late fall-off is characteristic of highly effective retrieval methods. In this result, one can notice a precision above 90% up to 50% of recall.

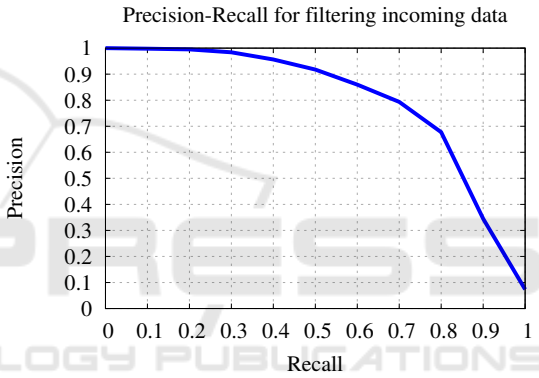

Figure 3: Precision-Recall in the process of filtering incoming data in the buffer.

The results show that DCCM is expected to filter out 90% of near-duplicate data. This is a strong indication that such capability can significantly improve both efficiency and efficacy of a command center. Filtering is the most desirable functionality considered in this work. This is because crowdsourcing is highly prone to produce redundant data. Right after a crisis is installed, if filtering is not possible, the flow of information streamed by eyewitnesses may be too high for the command center to make good use of them. However, a similarity-enabled RDBMS in DCCM is able to handle such situation with basic similarity queries.

5.4 Retrieval of Historical Data

5.4.1 Methodology

In the context of crisis management, the experts from the command center might be willing to analyze data from past events that are similar to the current ones.

Such data may lead to decisions about how to proceed with the current crisis. In these situations, similarity queries play an important role.

Considering the DCCM architecture, this task can be performed whenever the *Decision-Making* experts want information similar to the current data. For every notified data at point A5 of Figure 2, they might provide it as the s_q element to the *Historical Retrieval Engine* in order to get similar information.

5.4.2 Experimentation and Results

For this experiment, we combined the Color Structure Descriptor and the Manhattan Distance.

We performed Range and *k*NN Queries using each of the 2,000 elements from the Flickr-Fire dataset as the s_q element. We set the k parameter to 1,000 and the ξ parameter to 7.2, retrieving an entire class.

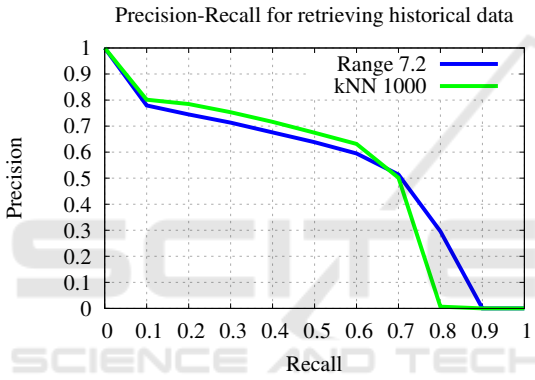

Figure 4: Precision-Recall for retrieving historical data.

We generated the Precision-Recall curve depicted in Figure 4. From these results, one can observe the high precision around 0.8 when fetching 10% of relevant data, roughly 100 images, and around 0.9 when fetching 5%, nearly 50 images — a more realistic scenario. These results point to an effective retrieval of images based on their class.

From the point of view of a command center user, there would be an ample knowledge bank from which initial considerations could be drawn from the current crisis. This initial knowledge has the potential of saving time of rescue actions by preventing past mistakes and fostering successful decisions.

5.5 Overall Performance

Concerning a computer system, it is important to receive the correct response in a timely manner. Therefore, we analyzed the overall performance of DCCM. For this purpose, we carried out one experiment regarding scalability and three regarding the tasks.

Overall Scalability. A solution based on DCCM spends most of its time receiving, storing and indexing data for the sake of similarity retrieval. Therefore, such processing must be efficient. We carried out an experiment to evaluate the time spent extracting features and inserting them into the database. The average time of five rounds is presented in Figure 5.

From the results presented in the figure, one can calculate that the solution is able to process up to 3 images per second — sufficient for most scenarios. These numbers refer to a machine with a hard disk of 5,400RPM; such results could be improved by using SSD disks or RAID subsystems.

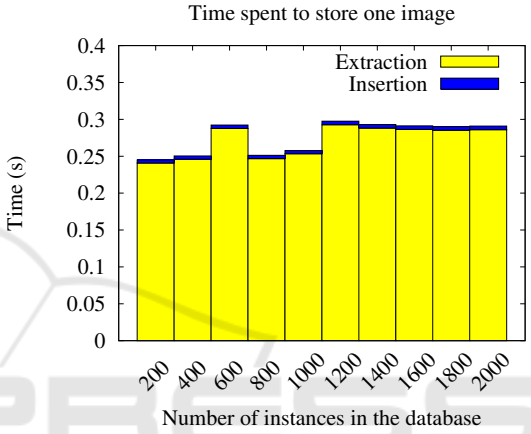

Figure 5: Time to extract features from one image and insert them into the database.

From Figure 5, one can also observe that the time spent inserting images is mostly taken by the feature extraction, while the time for inserting the features remains constant. The extraction time varies according to the image resolutions, which range from 300x214 to 3,240x4,290 pixels in the Flickr-Fire dataset.

Table 1: Overall performance of DCCM over Flickr-Fire.

Task	Average time (sec)
Classification	0.851
Filtering	0.057
Retrieval — Range Query	1.147
Retrieval — *k*NN Query	0.849

Table 1 presents the performance of DCCM to perform the three tasks of our methodology. We ran the classification and filtering tasks 10 times, whereas the retrieval tasks were performed 2,000 times, once for each element in the dataset. For the classification task, we used the distanced-weighted *k*NN classifier, with $k = 10$ and performing 10-fold cross-validation. For the filtering task, we used the 80 aforementioned near-duplicate images and the range value ξ was set to 10. Finally, in the retrieval tasks, ξ was set to 2.8 for the

Range Queries, retrieving around 50 tuples, and k was set to 50 for the kNN Queries. The results, which represent the average time to perform each task once, indicate that our proposal is feasible in a real-time crisis management application.

6 CONCLUSIONS

Fast and precise responses are essential characteristics of computational solutions. In this paper, we proposed the architecture of a solution that can achieve these characteristics in crisis management tasks. In the course of our work, we described the use of a similarity-enabled RDBMS in tasks that could assist a command center in guiding rescue missions. To make it possible, we implemented similarity-based operations within one popular, open-source RDBMS.

The core of our work is related to an innovation project led by the European Union; accordingly, we applied similarity retrieval concepts in an innovative manner, putting together relational and retrieval technologies. To demonstrate our claims, we carried out experiments to evaluate both the efficacy and the efficiency of our proposal. More specifically, we introduced the following functionalities:

- **Classification of Incoming Data.** We proposed to employ kNN classification to classify incoming data, aiming at identifying and characterizing crisis situations faster;
- **Filtering of Incoming Data.** We proposed to employ Range Queries to filter out redundant information, aiming at reducing the data load over the system and over a command center;
- **Retrieval of Historical Data.** We proposed to employ Range and kNN Queries to retrieve data from past crises that are similar to the current one.

The results we obtained for each of these tasks allowed us to claim that a similarity-enabled RDBMS is able to assist in the decision support of command centers when a crisis situation strikes. We conclude by stating that our work demonstrated the use of cutting-edge methods and technologies in a critical scenario, paving the way for similar systems to flourish based on the experiences that we reported.

ACKNOWLEDGEMENTS

This research has been supported, in part, by FAPESP, CAPES, CNPq and the RESCUER project, funded by the European Commission (Grant: 614154).

REFERENCES

Aha, D., Kibler, D., and Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*.

Barioni, M., Kaster, D., Razente, H., Traina, A., and Traina Jr., C. (2011). Querying Multimedia Data by Similarity in Relational DBMS. In *Advanced Database Query Systems*.

Bedo, M., Blanco, G., Oliveira, W., Cazzolato, M., Costa, A., Rodrigues Jr., J., Traina, A., and Traina Jr., C. (2015). Techniques for effective and efficient fire detection from social media images. ICEIS '15.

Bedo, M., Traina, A., and Traina Jr., C. (2014). Seamless integration of distance functions and feature vectors for similarity-queries processing. *JIDM*.

Celik, T., Ozkaramanli, H., and Demirel, H. (2007). Fire and smoke detection without sensors: Image processing based approach. EUSIPCO '07.

Fix, E. and Hodges Jr., J. (1951). Discriminatory analysis — Nonparametric discrimination: Consistency properties. Technical report, DTIC Document.

Ghahremanlou, L., Sherchan, W., and Thom, J. (2015). Geotagging Twitter messages in crisis management. *The Computer Journal*.

Gibson, H., Andrews, S., Domdouzis, K., Hirsch, L., and Akhgar, B. (2014). Combining big social media data and FCA for crisis response. UCC '14.

Halder, B. (2014). Crowdsourcing collection of data for crisis governance in the post-2015 world: Potential offers and crucial challenges. ICEGOV '14.

Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*.

Huang, M., Smilowitz, K., and Balcik, B. (2013). A continuous approximation approach for assessment routing in disaster relief. *Transportation Research Part B*.

Kaster, D., Bugatti, P., Ponciano-Silva, M., Traina, A., Marques, P., Santos, A., and Traina Jr., C. (2011). MedFMI-SiR: A powerful DBMS solution for large-scale medical image retrieval. ITBAM '11.

Kudyba, S. (2014). *Big Data, Mining, and Analytics: Components of Strategic Decision Making*.

Mehrotra, S., Butts, C., Kalashnikov, D., Venkatasubramanian, N., Rao, R., Chockalingam, G., Eguchi, R., Adams, B., and Huyck, C. (2004). Project Rescue: Challenges in responding to the unexpected. EI '04.

MultiMedia, I. (2002). MPEG-7: The generic multimedia content description standard, p. 1. *IEEE MultiMedia*.

Reznik, T., Horakova, B., and Szturc, R. (2015). Advanced methods of cell phone localization for crisis and emergency management applications. *IJDE*.

Sikora, T. (2001). The MPEG-7 visual standard for content description — An overview. *IEEE Trans. Cir. Sys. Vid.*

Silva, Y., Aly, A., Aref, W., and Larson, P. (2010). SimDB: A similarity-aware database system. SIGMOD '10.

Wilson, D. and Martinez, T. (1997). Improved heterogeneous distance functions. *J. Artif. Int. Res.*

Zhang, J., Li, J., and Liu, Z. (2012). Multiple-resource and multiple-depot emergency response problem considering secondary disasters. *Expert Syst. Appl.*