Open Access

Open Journal of Semantic Web (OJSW)
Volume 4, Issue 1, 2017

www.ronpub.com/ojsw
ISSN 2199-336X

A Semantic Safety Check System for Emergency Management

Yogesh Pandey, Srividya K. Bansal

School of Computing, Informatics, and Decision Systems Engineering, Arizona State University,
Mesa, AZ 85212, USA, {ypandey, srividya.bansal}@asu.edu

Abstract

There has been an exponential growth and availability of both structured and unstructured data that can be leveraged to provide better emergency management in case of natural disasters and humanitarian crises. This paper is an extension of a semantics-based web application for safety check, which uses of semantic web technologies to extract different kinds of relevant data about a natural disaster and alerts its users. The goal of this work is to design and develop a knowledge intensive application that identifies those people that may have been affected due to natural disasters or man-made disasters at any geographical location and notify them with safety instructions. This involves extraction of data from various sources for emergency alerts, weather alerts, and contacts data. The extracted data is integrated using a semantic data model and transformed into semantic data. Semantic reasoning is done through rules and queries. This system is built using front-end web development technologies and at the back-end using semantic web technologies such as RDF, OWL, SPARQL, Apache Jena, TDB, and Apache Fuseki server. We present the details of the overall approach, process of data collection and transformation and the system built. This extended version includes a detailed discussion of the semantic reasoning module, research challenges in building this software system, related work in this area, and future research directions including the incorporation of geospatial components and standards.

Type of Paper and Keywords

Regular research paper: *data integration, emergency management, earthquake alerts, weather alerts, linked data, semantic computing, semantic data.*

1 Introduction

You can never tell when a disaster is going to strike you. In times of crisis, web is one place where one can turn for help. The web provides an easy way to disseminate large amounts of information to large groups of people very quickly and efficiently [54]. In this work, we develop a semantic safety check system, which can be deployed a web application or a mobile application (as shown in Figure 1). With the real-time updates on what is going on, the safety check system can help people to stay safe and well-informed in the times of crisis. Our system for safety check identifies people that may get affected due to natural disasters such as earthquakes, floods, droughts, storms, cyclones, hurricanes, landslides, volcanic eruptions, disease outbreak and provides important information about the crisis, e.g., what areas are affected, what is the extent of the disaster, when is it safe to go back, and where to find shelter. The google public alerts is used in our system to get thus information before, during and after a disaster.

Google Crisis Response Public Alerts service [22] is an online notification service owned by Google.org, which publishes safety alerts, including weather watches, warnings, advisories, safety instructions. As Google also mentioned in a blog post [52]: “By providing useful, accurate, early-warning information,

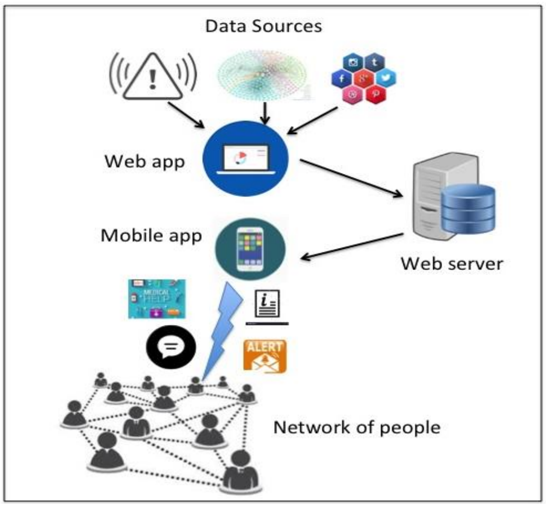

Figure 1: The semantic safety check system

we want to do our part to help people prepare. More information won't stop natural disasters from occurring, but it can go a long way to keeping people safe, and in some cases, could even save lives."

The second main aspect of this system is to utilize increasing role of social media in an emergency management tool. In the last few years, the popularity of social media has grown exponentially. As a significant number of people are using social media websites such as Facebook, Twitter, LinkedIn, we can utilize these mediums to gather the latest information about the people, issue emergency warnings, receive victim requests for assistance and conduct emergency communications [33]. Keeping these things in mind, we have developed our semantic-based system for safety check and targeted people on social media websites as shown in Figure 1.

The primary research question that this work attempts to answer is: How can we utilize the power of semantic computing and kinked data to develop an emergency management system? This paper is an extension of the work presented at SDB@SIGMOD 2017 [42]. It presents the design and implementation of a safety check system based on semantic technologies. This system uses an underlying semantic data model, which offers greater capabilities for data integration and extensibility over traditional approaches to detect and make information available about natural disasters. The extensions include a discussion of the data integration approach, details of the semantic reasoning module, research challenges in building this software system, related work that reviews the state-of-the-art and a comparison of this work with other tools and approaches for disaster management, and future research directions including the incorporation of geospatial components and standards. Feedback was obtained from users regarding the usability and functionality of this system. This feedback provided directions for future research, which are also presented in this paper.

The rest of this paper is structured as follows. Section 2 introduces the background of this work. Section 3 gives an overview of our semantic safety check system. Section 4 describes the large-scale data integration in the system, including a semantic data model, reference rules, and queries. Section 5 provides details of the client-side implementation of the system. Subsequently, we discuss related work and compare our semantic-based safety check system with the Facebook's safety check feature in Section 6. Challenges faced during the design and development of a semantic safety check system and future research directions are presented in Section 7. Finally, we concludes this work in Section 8.

2 BACKGROUND

Semantic web [4][9] is the next generation web, which allows much more advanced knowledge management by organizing knowledge into conceptual spaces according to its meaning. Semantic web uses automated tools and reasoners for supporting knowledge maintenance by checking inconsistencies and extracting new knowledge from existing knowledge bases. With the growing need of integration of Big Data, we need to find ways that computers can comprehend documents on the web. The semantic web aims to convert the current web, dominated by unstructured and semi-structured documents, into a "Web of Data", by encouraging the inclusion of semantic data on the internet. The ultimate goal is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network.

The foundation of the semantic web was laid by Sir Tim Berners-Lee (inventor of the World Wide Web and director of the W3C) along with James Hendler and Ora Lassila. He articulated it at the very first World Wide Web Conference in 1994 and later coined the term "Semantic Web" in 1998. To support this vision, the W3C has developed a set of standards and tools to enable human readable and computer-interpretable representation of the concepts, terms, and relationships within a given knowledge domain, which can be illustrated by semantic web technologies. Semantic web technologies are best suited to handle data with high volume, velocity and variety.

Big Data is transforming science, engineering, medicine, healthcare, finance, business, and ultimately society itself. Massive amounts of data are available to be harvested for competitive business advantages, government policies, and new insights into a broad variety of applications (including genomics, healthcare,

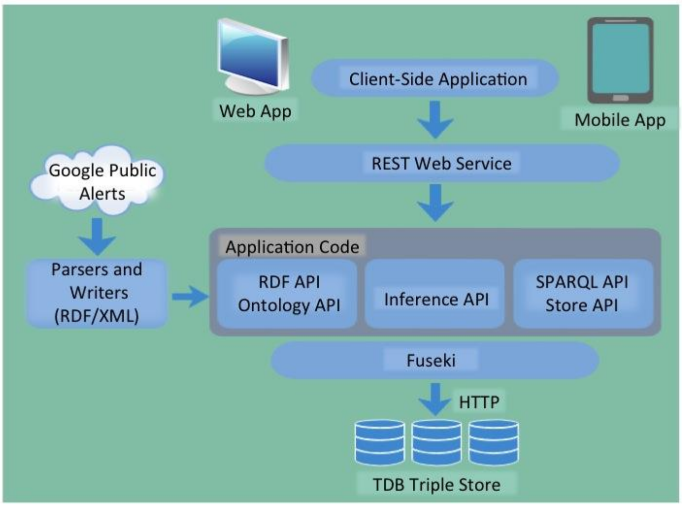

Figure 2: The semantic safety check system: High level architecture

biomedicine, energy, smart cities, transportation). However, most of this data is inaccessible to users, as users need technology and tools to find, transform, analyze, and visualize the data in order to make it consumable for decision-making [27]. Big data challenges are not only in storing and managing a variety of data but also extracting and analyzing consistent information from it [24]. Data management in a schema-less and complex big data web brings new challenges. The Linked Open Data (LOD) community effort has led to a huge data cloud with 31 billion RDF triples [24]. Some efficient approaches like [23] have been developed to manage large RDF datasets. LOD is the method of connecting and publishing structured data on the web, and can be used in a number of Web and mobile applications. A comprehensive survey on big data in the cloud is given in [38].

Various recent studies have focused on the use of semantic technologies to build emergency management systems. This is because disaster data are extremely heterogeneous both structurally and semantically. This creates a need for data integration and ingestion in order to identify and associate semantically corresponding concepts in the data [25]. The meaning of data must be fully comprehensible by machines so that the whole process can be automated. Through the use of ontologies, the semantics of data can be made explicit and therefore machine-processable. In the emergency management domain, the use of ontologies promotes data interoperability among systems and can assist the emergency management officials in rapid disaster recovery [44]. Based on these studies, we decide to build an emergency management system based on semantic web technologies and linked data. A high level architecture of the proposed system is shown in Figure 2.

3 System Overview

In this work we use semantic technologies for connecting, linking, and making data from different data sources and domains available through the linked open data cloud. The approach allows for large-scale data integration using a semantic Extract Transform Load process (shown in Figure 3) and involves the following phases:

Phase I (Big Data): This phase gathers data from multiple sources and in various formats.

Phase II (Data Ingestion): This phase extracts and cleans the data obtained in Phase I. It also processes and translates the data into RDF/XML format. To continuously add and update data, there is automate update cycle between Phase I and II.

Phase III (Semantic Graph): This phase builds semantic graphs from the data from Phase II. The semantic graphs semantically connect multiple datasets from different domains with Linked Open Data (LOD) cloud.

Phase IV (Semantic Reasoning): This phases discovers or infers new facts by using a semantic reasoner, which operates on the semantic graphs and inference rules. These inferred facts also get added into the semantic graphs.

Phase V (Safety Check Application): This phases uses the data obtained in Phase IV to predict, discover, or find impacted people and inform them accordingly.

Traditionally Data Integration has been defined as the problem of combining data residing at different sources, and providing the user with a unified view of

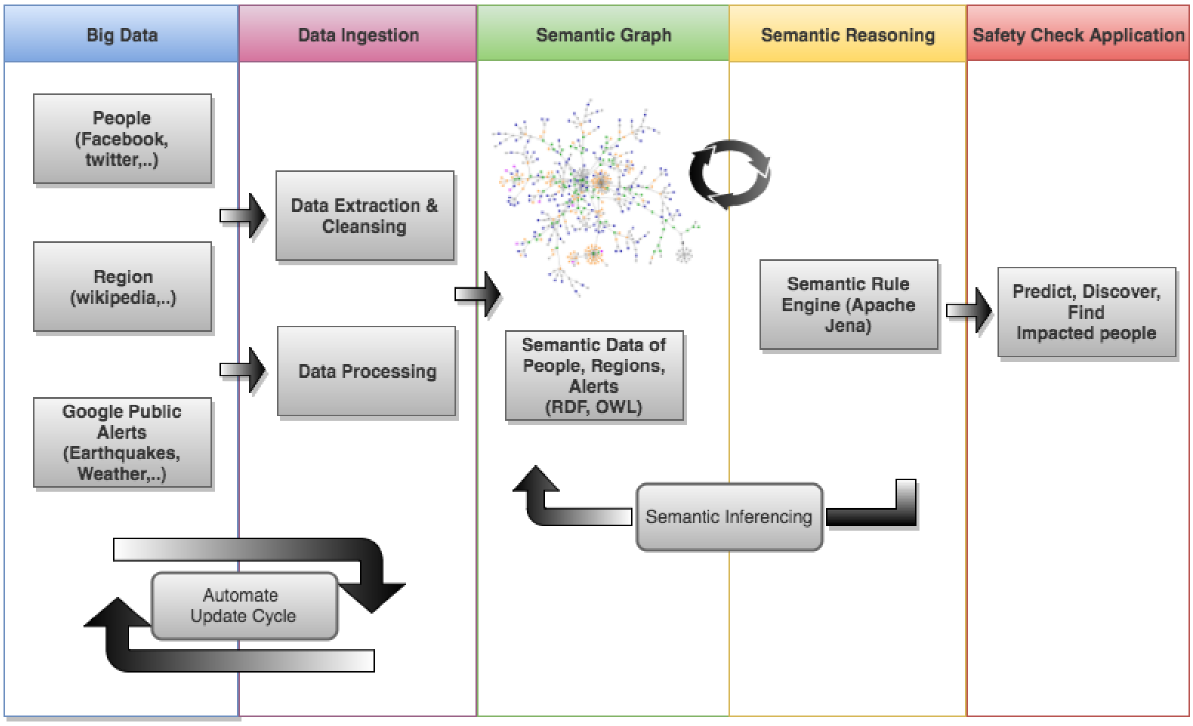

Figure 3: The semantic safety check system: Semantic-based large-scale data integration

these datasets [31]. The first step is to understand the terminology of the datasets and get familiarized with them by exploring in an unstructured fashion. The second step is to create a mediated or global schema that provides mapping between various source schemas. A data integration system exposes to its users a schema for posing queries. This schema is typically referred to as a mediated schema (or global schema). To answer queries using the various information sources the system needs mappings that describe the semantic relationships between the mediated schema and the schemas of the sources.

Definition 1 (Mediator Schema): Let $S_1, ..., S_n$ be the local schemas on n pre-existing data sources. Assume for brevity of presentation each local schema S_i is made of single relation also denoted as S_i. The relations $S_1, ..., S_n$ are called the local relations. Global schema G consists of global relations $G_1, ..., G_m$. Semantic mappings between local and global relations is of the form: $v(S_1, ..., S_n) \subseteq v'(G_1, ..., G_m)$, where v and v' are query expressions called views. Given an instance I of $\{S_1, ..., S_n\}$, $v(I(S_1), ..., I(S_n)) \subseteq v'(J(G_1), ..., J(G_m))$, where J is an instance of the global schema.

Schema mapping is done using an approach called *global-as-view* that requires the global schema to be expressed in terms of the data sources [31], [16], [17].

Definition 2 (Global-as-View): The semantic mappings are of the form: $V_i(S_1, ..., S_n) \subseteq G_i$, where each V_i is a view over the local schema, i.e., a query built on local relations.

The high-level architecture of the semantic safety check system has been shown in Figure 2. We have developed two client-side applications for different platforms: one is a Web Application and another one is an Android Mobile Application. Both make use of the restful-web services exposed by our system. We are using Apache Jena (or Jena in short), an open source Java framework for building semantic web and Linked Data applications [35], and serialize the triples into RDF/XML formats [45]. RDF (resource description framework) [55] is a framework for creating statements about Semantic Web resources in a form of 'subject-predicate-object' triples.

Jena also provides Ontology APIs to work with models, RDFS (RDF Schema) [46] and OWL (Web Ontology Language) [41] to add extra semantics to RDF data. RDFS is intended to structure RDF resources by providing a basic vocabulary for RDF. The OWL is a W3C standard for representing domain knowledge. It allows the representation of domain knowledge as a set of RDF and provides sematic-based integration of data. We use TDB [38], a high-performance triple store, to store our data, and use Jena's ARQ, a SPARQL [50] compliant engine, to

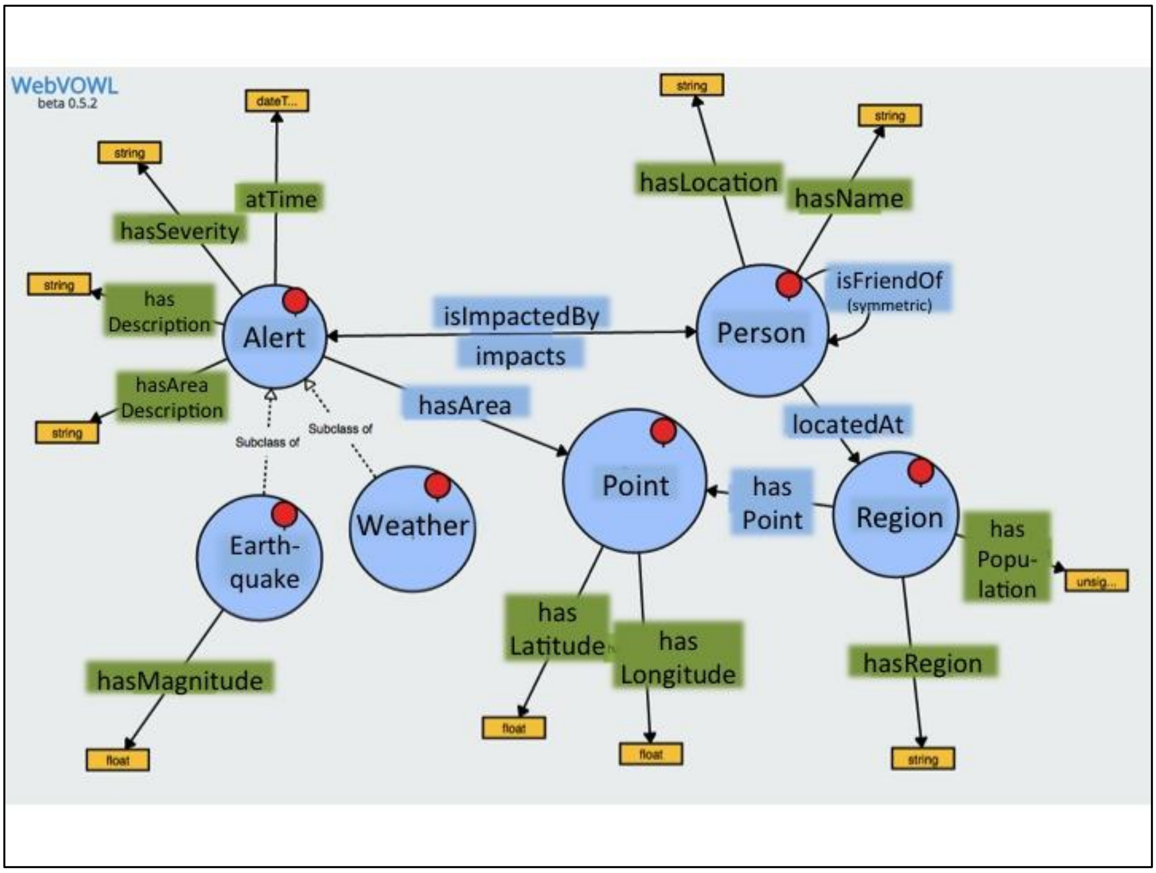

Figure 4: Ontology – the data model of the semantic safety check system

query RDF models. SPARQL is the standard Semantic Web query language to retrieve and manipulate RDF data. Jena also has a built-in support for many reasoners. We use Jena's inference APIs to reason over the data to expand the content of our triple store, and Fuseki Server [14] to query and serve RDF data over HTTP.

4 SEMANTIC-BASED LARGE-SCALE DATA INTEGRATION

The semantic safety check system utilizes information from various domains and sources like people, cities, coordinates, earthquakes and weather alerts. Different approaches were taken to extract data for each dataset:

- People data was collected using Facebook's Graph APIs. A client program was developed that uses our access tokens to get friends and family member information.
- To gather important information about major cities like latitude, longitude, area and population, web crawlers were implemented that gathered data in CSV format.
- To collect information on earthquakes and weather, we subscribed to Google Public Alerts. Google's Alert Hub implements PubSubHubbub [1], which is a simple, open, server-to-server protocol of publishers and subscribers. Publishers send their alert feeds to Alert Hub, which pushes those updates to our server.

The data collected had to be cleansed specifically for completeness, consistency, and uniformity. For data about contact records of persons with incomplete information or inaccurate location information had to be removed to ensure that the data fits our semantic data model. Location information was obtained in the form of latitude and longitude for all contact persons to ensure the uniformity of data. For data about different regions, web crawlers were written based on our data model. Rules were incorporated into the web crawler to maintain the completeness, consistency, and uniformity of data. A similar process was followed for alerts obtained from Google Alerts. An alert is associated with an impacted region as opposed to a location in case of the other two datasets. After all of the data is gathered, the data is cleansed and then translated into RDF instances.

4.1 Semantic Data Model - Ontology

In order to model the data obtained, we develop an ontology using OWL ontology language. The data model defines classes and properties of the data and the constraints and relations among them. For example, the impacted region is defined as a *region* class in our data model. Figure 4 visualizes the ontology data model. In the figure, the classes are shown as blue circles, properties are shown as green boxes and their domains are shown as yellow boxes. Each record represents an instance of these classes. This OWL ontology has been created using a software called Protégé [28], which is a free, open source ontology editor and a knowledge acquisition system.

In our ontology, we define four classes: Alert, Person, Region, and Point. The Alert class is a super class for all types of alerts. In our implementation, so far we have two subclasses of Alert class: Earthquake and Weather. In the future, as we extend the support for other types of alerts, we will implement more subclasses. When an alert is obtained from google public alerts service, based on the alert's topic URL the alert is either transformed into an instance of earthquake or weather alert. Information on major cities has also been translated as individuals of Region class. And we used the friends' information from Facebook to create instances of Person class. We have also kept "isFriendOf" property to maintain information about who is friend of whom. In the future, this information will be useful to notify a person if his/her friend(s) are impacted by a disaster. We have also defined a Point class that holds coordinates (Latitude, Longitude) of a location.

4.2 Inference Rules

Semantic Reasoners work on the ontology and reference rules to derive additional facts on the modeled concepts [3]. A reasoner creates new RDF graphs containing asserted and derived tuples. These inferred graphs can be queried in the same way as other RDF graphs. The reference rules enable advanced ontology-based inferences. The rules extend the expressivity of ontology with formal rule representation languages. We used the built-in OWL/RDFS reasoner of Jena and develop a set of logic inference rules. Using these rules, we programmed logic into our system and integrated data store, leading to significantly simpler software system with greater interoperability. The logic reference rules are written in the Jena's rule language and are presented here.

Rule 1: This rule links persons with their regions based on "locationName" attribute. The inferred knowledge (RDF graphs) provides the coordinate location of a person. The rule snippet is shown below:

```
[rule1:
   (?person rdf:type sc:Person)
   (?region rdf:type sc:Region)
   (?person sc:hasLocation ?ploc)
   (?region sc:hasRegionName ?rloc)
   regionMatch(?ploc, ?rloc) ->
      (?person sc:locatedAt ?region)
]
```

Rule 2: This rule is to identify all persons who may have been impacted due to an earthquake. It uses earthquake's information like magnitude, epicenter and people coordinate locations to infer who may have been impacted. The radial distance, over which the effects of an earthquake should be felt by people, has been estimated using McCue Radius of Perception Calculator [35]. The rule snippet is shown below:

```
[rule2:
   (?earthquake rdf:type sc:Earthquake)
   (?person sc:locatedAt ?region)
   (?earthquake sc:hasMagnitude ?mag)
   (?earthquake sc:hasArea ?epoint)
   (?epoint sc:hasLatitude ?elat)
   (?epoint sc:hasLongitude ?elong)
   (?region sc:hasPoint ?rpoint)
   (?rpoint sc:hasLatitude ?rlat)
   (?rpoint sc:hasLongitude ?rlong)
   eqImpactMatch(?elat, ?elong, ?rlat,
         ?rlong,?mag) ->
      (?person sc:isImpactedBy ?earthquake)
]
```

Rule3: This rule is to identify all persons who may have been impacted by a weather alert. This rule checks if person's location (or coordinates) lies inside polygon region of the weather alert and if it does, it adds an inferred fact that the person is impacted by the weather alert. The rule snippet is shown below:

```
[rule3:
   (?weather rdf:type sc:Weather)
   (?weather sc:hasPolygon ?wpoly)
   (?person sc:locatedAt ?region)
   (?region sc:hasPoint ?rpoint)
   (?rpoint sc:hasLatitude ?rlat)
   (?rpoint sc:hasLongitude ?rlong)
   weatherImpactMatch (?wpoly, ?rlat,
           ?rlong) ->
    (?person sc:isImpactedBy ?weather)
]
```

4.3 Custom Built-ins for Matching

Jena provides some builtin primitives. These primitives are called by rules to test if a rule matches or not. Jena also supports extending the set of procedural builtins [47]. A custom builtin should implement the *Builtin* interface. The way it has been done is by creating a subclass of *BaseBuiltin* and defining a name (getName), the number of arguments expected (getArgLength) and one or both of *bodyCall* and *headAction*. The *bodyCall* method is used when the builtin is invoked in the body of a rule clause and should return true or false according to whether the test passes. Once a builtin has been defined then an instance of it needs to be registered with *BuiltinRegistry* for the bulitin to be seen by the rule parser and interpreter. In our system, we have developed several custom builtins, “regionMatch”, “eqImpactMatch”, and “weatherImpactMatch”, to check if a person or a region is impacted by a disaster, e.g. earthquake. These custom built-ins are used as functions in the rules discussed in Section 4.2.

Function for Region Matching:
The *RegionMatch* function checks the location of a person against the impacted region of an alert. The location of a person is represented using the latitude and longitude, which map to the coordinates of Point class. Impacted region on the other hand is represented as a polygon, i.e., a collection of points. This function checks if the coordinates of a person’s location fall inside the polygon area of the impacted region and returns a true or false. The code snippet of the function is shown in Listing 1 in Appendix.

Function for Weather Impact Matching:
This *WeatherImpactMatch* function takes as input parameters a polygon representing the region and the coordinates of the region being matched against. A code snippet of the function is shown in Listing 2 in Appendix.

Function for Earthquake Impact Matching:
Earthquake impact region is obtained by computing the radius of the earthquake using its magnitude. McCue Earthquake Perception Radius Calculator [36] is used for this purpose. The code snippet of the function is shown in Listing 3 in Appendix.

$$\text{EarchquakeRadius} = \frac{(\text{Magnitude} - 0.13)}{1.01}$$

4.4 Semantic Querying

We develop several SPARQL queries. With these queries, our semantic safety check system can obtain needed data (like people, disasters, and geographical location) from various data sources, and find those people that may have been affected by a disaster occurring at a certain geographical location and notify them with safety instructions.

Query to get all earthquake instances:
This query retrieves all instances of earthquakes along with the information about the earthquake such as its magnitude, location, time of occurrence, and description. The results of this query can be filtered based on magnitude, time, or region. The SPARQL query is shown below:

```
select ?earthquake ?point ?lat ?lon ?mag
       ?time ?desc ?areaDesc
where {
   ?earthquake rdf:type sc:Earthquake.
   ?earthquake sc:hasMagnitude ?mag.
   ?earthquake sc:hasAreaDescription
            ?areaDesc.
   ?earthquake sc:hasArea ?point.
   ?point sc:hasLongitude ?lon.
   ?point sc:hasLatitude ?lat.
   ?earthquake sc:atTime ?time.
   ?earthquake sc:hasDescription ?desc
}
```

Query to get all weather alerts:
This query retrieves all instances of weather alerts along with the information about the alert such as its severity, location, time of occurrence, and description. The results of this query can be filtered based on severity, time, or region. The SPARQL query is shown below:

```
select ?weather ?areaDesc ?sev ?time ?desc
(GROUP_CONCAT(?lat) AS ?lats)
(GROUP_CONCAT(?lon) AS ?lons)
where {
   ?weather rdf:type sc:Weather.
   ?weather sc:hasSeverity ?sev.
   ?weather sc:hasAreaDescription
          ?areaDesc.
   ?weather sc:hasArea ?area.
   ?area rdfs:member ?point.
   ?point sc:hasLongitude ?lon.
   ?point sc:hasLatitude ?lat.
   ?weather sc:atTime ?time.
   ?weather sc:hasDescription ?desc
}
GROUP BY ?weather ?areaDesc ?sev
         ?time ?desc
```

Query to get all persons impacted by an alert:
This query retrieves all persons impacted by a specific alert based on the location of the person and region of the alert. Information about the person and his/her location in terms of latitude and longitude are retrieved. The SPARQL query is shown below:

```
select ?person ?name ?location ?region
      ?point ?lat ?lon
where {
   ?person sc:isImpactedBy
       <http://www.semanticweb.org/
        ontologies/2015/10/
        SafetyCheck#20005hxx>.
   ?person sc:hasName ?name.
   ?person sc:hasLocation ?location.
   ?person sc:locatedAt ?region.
   ?region sc:hasPoint ?point.
   ?point sc:hasLatitude ?lat.
   ?point sc:hasLongitude ?lon.
}
```

5 CLIENT-SIDE IMPLEMENTATION

Since the semantic safety check system is deployed as a RESTful web service, it is very easy to develop client-side software for different platforms. We have implemented two clients in our system: a web application and an android mobile application.

A Web Application:
This web application is hosted on the webpage (http://imod.poly.asu.edu:8080/SafetyCheckWeb/), and the source code is available on GitHub at https://github.com/yogeshpandey009/SafetyCheck.
This web app provides alerts about earthquakes and weather. Figure 5 and 6 are screenshots of the webpage, which list captured earthquake and weather alerts. If you double click a weather alert or an earthquake alert, the website navigates to another page that lists the persons that may have been impacted by that alert. Furthermore, there is two other webpages: one lists all persons who are currently being monitored by our system, and another lists all regions information that we have gathered for our system.

An Android Mobile Application:
The mobile application is hosted on the webpage (https://github.com/yogeshpandey009/SafetyCheckAndroidApp). This Android mobile application has some additional features in comparison with the web application. As shown in Figure 7, 8, 9 and 10, this application can list and even show earthquake alerts with its impacted region on the world map. It also has a background service to pull new alerts from our application and provides notification to the users. It also shows all people that may have been impacted by an earthquake on the map. We are currently working on adding similar functionality for weather alerts in this mobile application.

6 RELATED WORK

One of the popular approaches to data integration has been Extract-Transform-Load (ETL) [53]. This work described taxonomy of activities in ETL and a framework that uses a workflow approach to design ETL activities. It used a declarative database programming language called LDL to define the semantics of ETL activities. Similarly there are other research contributions that have used various other approaches such as UML and data mapping diagrams for representing ETL activities [51], quality metrics driven design for ETL [34], and scheduling of ETL activities [48]. The focus in all of these papers has been on the design of ETL workflow and not about generating meaningful or semantic data.

Data integration efforts have been towards fixed data sets. However, there are some applications that require temporal data, data that varies over time. A related work in this area uses a preference aware integration of temporal data [2]. The authors have developed an operator called PRAWN (preference aware union), which is typically a final step in an entity integration workflow. It is capable of consistently integrating and resolving temporal conflicts in data that may contain multiple dimensions of time based on a set of preference rules specified by a user.

Ontologies have been used as a formal tool for sophisticated querying and expressing the domain level knowledge at a high-level of abstraction [43]. A number of approaches have developed for automatic generation [7] and evolution [11] of ontology. [11] introduces the technique of ontology templates to automatically evolve ontologies. The automatic ontology generation could be classified as convertors or translators, mining-based techniques, and applications using external knowledge. Convertors or translators involved mapping data from specific format such as XML, XSD, UML into an ontology [12][20][21][32]. Text was annotated with formal ontologies using various mining-based techniques [10][15][26][37]. There were also frameworks built that used external knowledge or domain-knowledge to produce ontologies [28]. Clio [49] is an approach that uses SQL schemas alone to generate the ontological mappings.

[5] is a research contribution to building natural disaster data and information management systems that provide contingency in disaster situations [5]. But this work does not use linked data and ontology technology. Another related work involves emergency situation awareness using Twitter feeds and mining twitter messages [13]. This work does not involve data integration from different sources but instead relies on mining social media data. A case study of Haitian earthquake showed how US government used social media such as wikis and collaborative workspaces as the main knowledge sharing mechanism in providing assistance [56]. Our work is to provide integration of datasets relevant to emergency management in a way that is extensible in the future through the use of semantic web technologies, ontologies, and linked data.

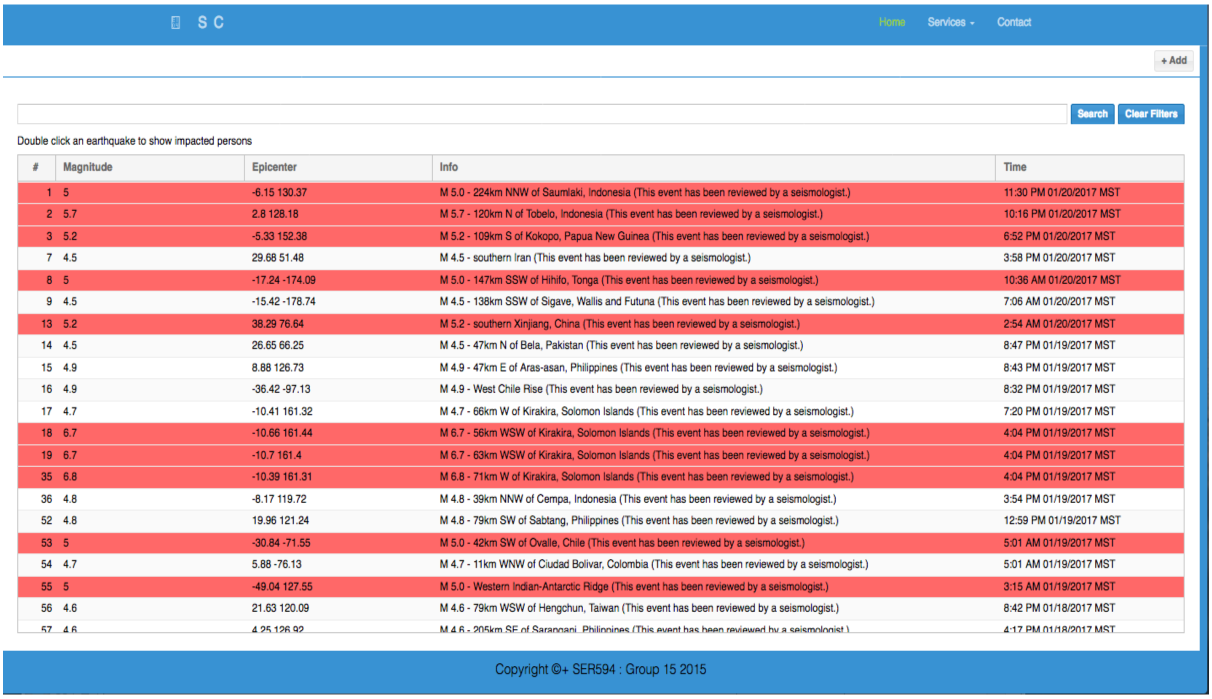

Figure 5: A screenshot of earthquake alerts of the web client in the semantic safety check system

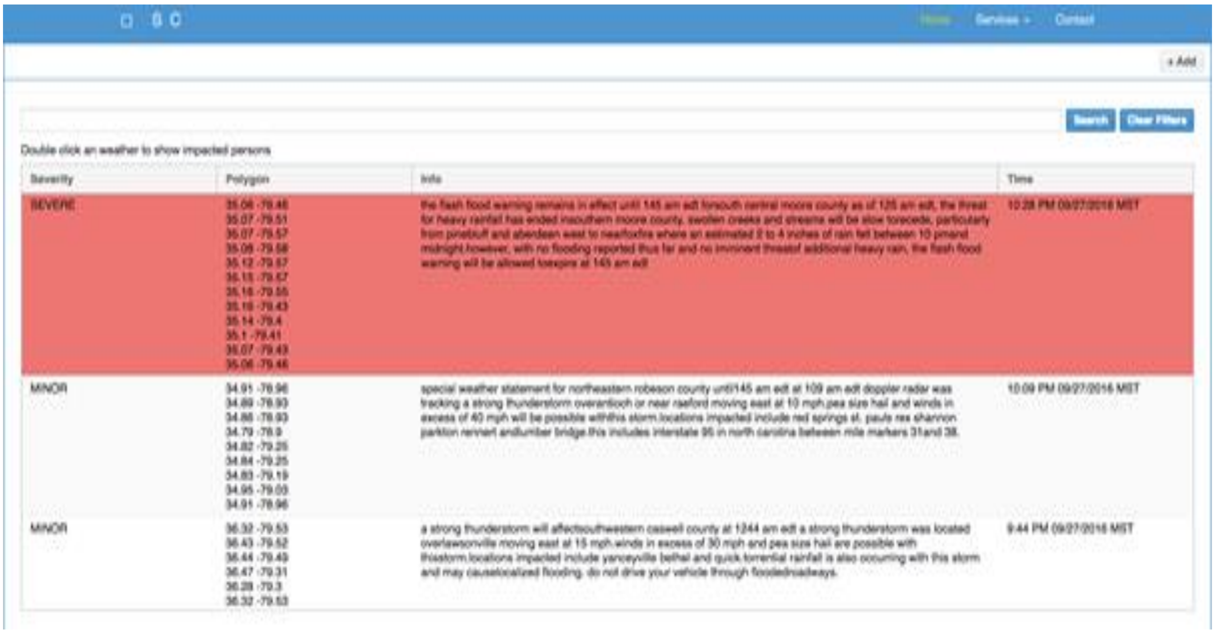

Figure 6: A screenshot of weather alerts of the web client in the semantic safety check system

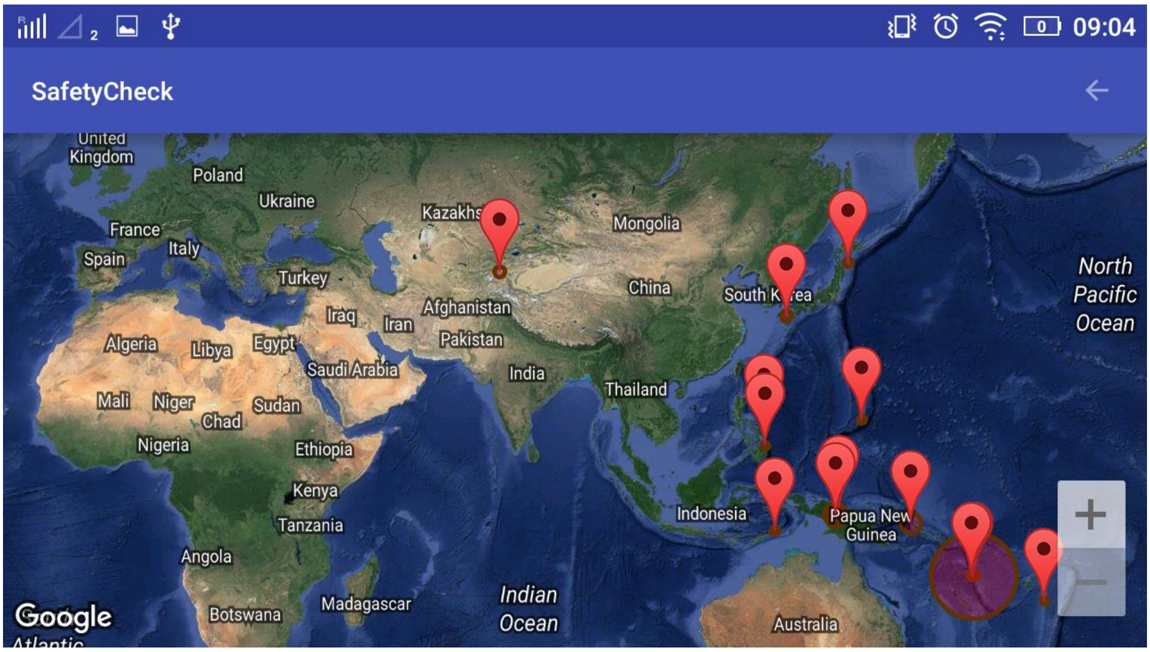

Figure 7: The map of earthquake alerts of the android mobile application with impacted regions

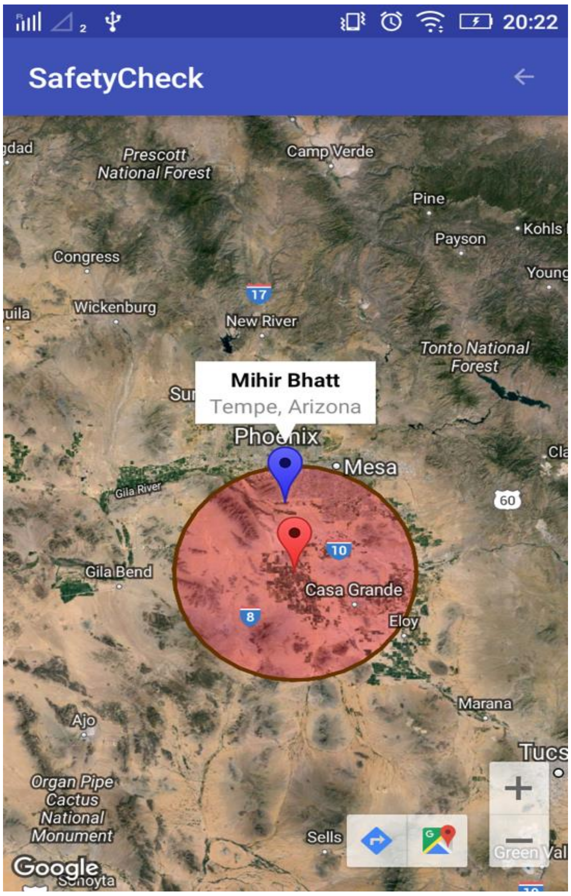

Figure 8: The map of person locations of the android mobile application impacted by an earthquake

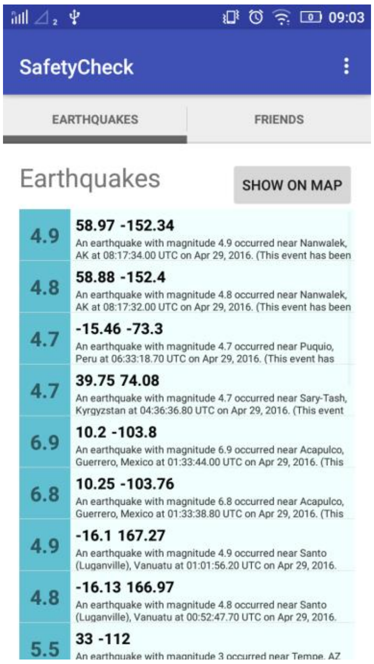

Figure 9: List of all earthquake alerts from the android mobile application

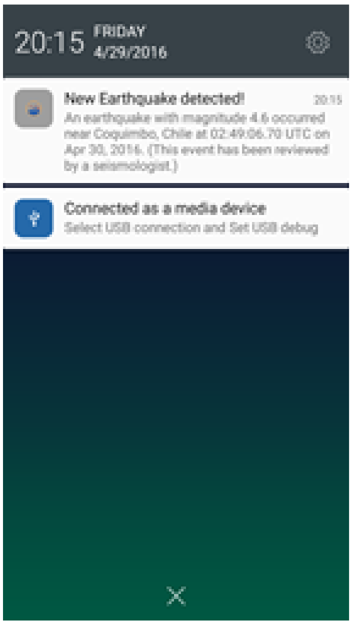

Figure 10: New earthquake alert notification from the android mobile application

6.1 Comparison with Facebook's Safety Check

In October 2014, Facebook introduced the Safety Check feature [18]. It allows people to quickly share with friends and families that they are safe and helps them connect with other people they care about. There are some key differences when compared to our semantic-based safety check system (shown in Table 1):

- Facebook's Safety Check feature relies solely on its own databases for people information, whereas our Semantic Safety Check system uses linked data, so it can easily be integrated to use data from other social media websites or government agencies to identify people who may have been affected by a crisis.
- Facebook's Safety Check feature is only activated for some major disasters. Facebook works with the local authority to determine what constitutes an emergency, whereas our system automatically responds to all received alerts. Moreover, using linked data our system can easily be extended to support new types and sources of alerts.
- The main feature of Facebook's Safety Check is for Emergency Check-In to notify friends and families that they are safe, whereas our semantic-based system supports more features. It provides early-warning information and other important safety instructions that can save lives of the people. This system is on the lookout for people who may be impacted in order to provide appropriate assistance.

7 Challenges and Future Work

7.1 Challenges

In the development of this system, we identify some challenges, which may provide insights for the development of other semantic-based safety check systems.

- As the datasets used for thus system comes from various sources, it has a mixed bag of structured, semi-structured and unstructured data. This requires a different approach or implementation to extract data from each data source.
- As major data of cities are crawled from multiple websites, there were some records, which had missing information (or fields) or sometimes are even improperly formatted. So this needs an extra step of data cleansing after extraction.
- Writing custom builtins for matching of rules requires an understanding of the internals of Apache Jena framework.
- Integrating datasets from the different sources requires a good understanding of each domain and source. Without the domain knowledge of the data, coming up with an integrated semantic data model is extremely challenging.

Table 1: Comparison of Facebook's safety check feature with our semantic safety check system

Feature	Facebook Safety Check	Semantic Safety Check
Semantic computing approach	Not used	Used. It allows an extension to other types of data sources
People Information DB	Uses Facebook's DB	Uses linked data from the web, and has possibilities to connect to other social media sites
Emergencies	Activated for major disasters determined by Facebook	Any emergency notified by the Google Alerts service
Notification	Allows affected persons to inform their friends and families	Notifies all registered persons when any of their contacts is in an emergency

7.2 Future Work

Our web application and android mobile application were presented to various users within our institution and at conferences. The received feedback provided insights into future directions.

The use of geospatial information that is available and integrating with GIS components and geo ontologies is one future research work. This will involve creation of geospatial Linked data that can be queried using GeoSPARQL [39] specifically designed for this purpose. In order to incorporate geospatial linked data, the semantic data model and transformation phase of the work will have to be revised to incorporate the classes and properties provided by GeoSPARL query language. For the reasoning part of the system the custom built-in functions to check if a person's location is within an impacted region of a disaster will be replaced with GeoSPARQL constructs that can check for this. The use of geospatial linked data and GeoSPARQL also allows integration of data from other GIS data sources as well. Smid's research team has presented integration and querying techniques over heterogeneous geospatial data sources [49]. As part of future research we will look into various heterogeneous data sources that can be used in our semantic safety check system.

Another useful feedback received was with respect to the temporal properties of an emergency event and how they change over time. For example, a natural disaster such as an earthquake occurs, and later on some aftershocks may occur that might change the emergency situation or the status of a person impacted by the earthquake. So it is important to include temporal properties and track how the event unfolds over time. We will look into semantic spatiotemporal RDF store such as Strabon [30], which uses a data model called stRDF and a query language called stSPARQL [8]. This data store is useful to store geospatial data that changes over time. This will involve tracking the alerts of emergencies and their change over time, and transforming temporal properties into the linked data. Querying the spatiotemporal data can then be done using stSPARQL.

8 CONCLUSIONS

In this work, we develop a semantic-based safety check system, which trace emergencies and help people well-informed in the times of crisis. Based on semantic technologies, our system offers greater compatibilities for data integration, extensibility and interoperability over traditional approaches. It provides accurate, reliable, and timely information, which is vital for public safety before, during, and after a crisis. Currently, the system monitors natural disasters like Earthquakes and Weather alerts. However, our system is a knowledge intensive system and it provides great opportunity to extend this support to humanitarian crises like major accidents, riots, conflict, wars, terrorist attacks and radiological accidents. The personal information used in our system is currently only from Facebook. We plan to gather more data from other social media websites like Twitter, Linkedin, and integrate with other emergency management organizations to reach more people. In the future, we also plan to support personalization based on user profiles and notify them (over email or SMS) instantly once a disaster or crisis is detected (or predicted) that may impact them or their friends and families. This will also provide possibilities for people to reply if they are safe or otherwise simply send a smoke signal when they need help.

REFERENCES

[1] P. Alexandre, and P. N. Mendes. “sparqlPuSH: proactive notification of data dpdates in RDF stores using PubSubHubbub”, in *SFSW*, 2010.

[2] B. Alexe, M. Roth, and W.-C. Tan, “Preference-aware integration of temporal data,” *VLDB Endow.*, 8(4), pp. 365–376, 2014.

[3] A. Ameen, R. K. Khaleel, and B. P. Rani. “Reasoning in semantic web using Jena”, *Computer Engineering and Intelligent Systems*, 5(4), pp. 39-47, 2017.

[4] G. Antoniou, F. vanHarmelen, “A semantic web primer”, *MIT Press*, 2004

[5] H. Assilzadeha, and S. B. Mansora. "Natural disaster data and information management system." University Putra Malaysia: Institute of Advanced Technology, 2004.

[6] A. Azzini and P. Ceravolo, “Consistent process mining over big data triple stores,” IEEE International Congress on Big Data, 2013.

[7] I. Bedini and B. Nguyen, “Automatic ontology generation: State of the art,” PRiSM Laboratory Technical Report. Univ. of Versailles, 2007.

[8] K. Bereta, P. Smeros, and M. Koubarakis, “Representation and querying of valid time of triples in linked geospatial data,” European Semantic Web Conference, 2013.

[9] T. Berners-Lee, J. Hendler, and O. Lassila. “The semantic web”, *Scientific American*, 84(5), pp. 28–37, 2001.

[10] B. Biebow, S. Szulman, and A. J. Clément, “TERMINAE: A linguistics-based tool for the building of a domain ontology,” *Knowledge Acquisition, Modeling and Management*, 1999.

[11] M. Blasko, P. Kremen, and Z. Kouba: "Ontology evolution using ontology templates", Open Journal of Semantic Web (OJSW), 2(1), pp. 16-29, 2015. [Online]: http://nbn-resolving.de/urn:nbn:de:101:1-201705194898

[12] H. Bohring and S. Auer, “Mapping XML to OWL ontologies.,” Leipziger Informatik-Tage, vol. 72, pp. 147–156, 2005.

[13] M. Cameron, R. Power, B. Robinson, and J. Yin. "Emergency situation awareness from twitter for crisis management", International Conference on World Wide Web, 2012.

[14] J.J. Carroll, I. Dickinson, C. Dollin, and D. Reynolds, A. Seaborne, and K. Wilkinson, K. “Jena: Implementing the semantic web recommendations”, International Conference on World Wide Web, 2004.

[15] Y. Ding, D. Lonsdale, D. W. Embley, M. Hepp, L. Xu: “Generating ontologies via language components and ontology reuse,” *Natural Language Processing and Information Systems*, 2007.

[16] A. Doan, A. Halevy, and Z. Ives, "*Principles of data integration*", Elsevier, 2012.

[17] X. L. Dong and D. Srivastava, “Big data integration,” *Synthesis Lectures on Data Management*, 7(1), pp. 1–198, 2015.

[18] Facebook Safety Check http://www.facebook.com/about/safetycheck, [Accessed: 20-Aug-2017].

[19] R. Fagin, L. M. Haas, M. Hernández, R. J. Miller, L. Popa, and Y. Velegrakis : “Clio: Schema mapping creation and data exchange,” *Conceptual Modeling: Fndns & Apps*, 2009.

[20] M. Ferdinand, C. Zirpins, and D. Trastour, “Lifting XML schema to OWL,” *Web Engineering*, 2004.

[21] R. Ghawi and N. Cullot, “Building ontologies from XML data sources.,” *DEXA*, 2009.

[22] Google, Inc. Google Crisis Response. https://www.google.org/publicalerts/, [Accessed: 20-Aug-2017].

[23] S. Groppe, D. Heinrich, C. Blochwitz, and T. Pionteck: "Constructing Large-Scale Semantic Web Indices for the Six RDF Collation Orders", Open Journal of Big Data (OJBD), 2(1), pp. 11-25, 2016. [Online]: http://nbn-resolving.de/urn:nbn:de:101:1-201705194418

[24] P. Hayes and B. McBride, “Resource description framework (RDF),” 2004. [Online]. Available: http://www.w3.org/TR/rdf-mt/, [Accessed: 20-Aug-2017].

[25] V. Hristidis: “Survey of data management and analysis in disaster situations*”, Journal of Systems and Software,* 83(10) , pp. 1701-1714, 2010.

[26] H. Hu and D.-Y. Liu, “Learning owl ontologies from free texts,” International Conference on Machine Learning & Cybernetics, 2004.

[27] E. Kandogan, M. Roth, C. Kieliszewski, F. Ozcan, B. Schloss: “Data for all: A systems approach to accelerate the path from data to insight,” Congress on Big Data, 2013.

[28] H. Knublauch, et al. “The Protégé OWL plugin: An open development environment for semantic web applications.” *International Semantic Web Conference,* 2004.

[29] H. Kong, M. Hwang, and P. Kim, “Design of the automatic ontology building system about the

specific domain knowledge," *Conference on Advanced Communication Technology*, 2006.

[30] K. Kyzirakos, M. Karpathiotakis and M. Koubarakis. Strabon, "A semantic geospatial DBMS," *International Semantic Web Conference*, 2012.

[31] M. Lenzerini, "Data integration: A theoretical perspective," *ACM SIGMOD symposium on Principles of database systems*, 2002.

[32] J. X. W. Li, "Using relational database to build OWL ontology from XML data sources," *Computational Intelligence*, 2007.

[33] B. R. Lindsay, "Social media and disasters: Current uses, future options, and policy considerations," *Media Telecommun*, 2011.

[34] S. Luján-Mora, P. Vassiliadis, and J. Trujillo, "Data mapping diagrams for data warehouse design with UML," *Conceptual Modeling–ER*, 2004.

[35] B. McBride, "Jena: A semantic Web toolkit," *IEEE Internet Computing*, 6(6), pp. 55-59, 2002.

[36] McCue Earthquake Perception Radius Calculator, http://www.cqsrg.org/tools/ perceptionradius/, [Accessed: 20-Aug-2017].

[37] D. I. Moldovan and R. Girju, "Domain-specific knowledge acquisition and classification using WordNet.," *Florida Artificial Intelligence Research Society Conference*, 2000.

[38] P. C. Neves, and J. Bernardino: "Big data in the cloud: A survey", *Open Journal of Big Data (OJBD)*, 1(2), Pages 1-18, 2015. [Online]: http://nbn-resolving.de/urn:nbn:de:101:1-201705194365

[39] Open Geospatial Consortium GeoSPARQL: A geographic query language for RDF Data, 2012. www.opengeospatial.org/standards/geosparql, [Accessed: 20-Aug-2017].

[40] A. Owens, A. Seaborne and N. Gibbins. "Clustered Tdb: A clustered triple store for Jena." *International World Wide Web Conference*, 2009.

[41] OWL Web Ontology Language Reference, World Wide Web Consortium, http://www.w3. org/TR/owl-ref, 31 Mar. 2003..

[42] Y. Pandey and S. K. Bansal. "Safety check – A semantic web application for emergency Management", *The Second International Workshop on Semantic Big Data (SBD 2017) in conjunction with 2017 ACM SIGMOD conference* in Chicargo, USA, 2017.

[43] A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, et al., "Linking data to ontologies," *Journal on data semantics*, 2008.

[44] K. R. Ratnam, and D. D. Karunaratne. "Application of ontologies in disaster management", *Conference on Advances in ICT for Emerging Regions*, 2016.

[45] RDF Primer. W3C Recommendation. http://www.w3.org/TR/rdf-primer, 2004, [Accessed: 20-Aug-2017].

[46] RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, http://www.w3.org/TR/rdf-schema, Feb 2004, [Accessed: 20-Aug-2017].

[47] D. Reynolds. "Jena 2 inference support." Online manual at https://jena.apache.org/ documentation/inference/, [Accessed: 20-Aug-2017].

[48] A. Simitsis, K. Wilkinson, M. Castellanos, and U. Dayal: "QoX-driven ETL design: Reducing the cost of ETL consulting engagements," *Conference on Management of Data*, 2009.

[49] M. Smid, P. Kremen. "OnGIS: Semantic query broker for heterogeneous geospatial data Sources", *Open Journal of Semantic Web (OJSW)*, 3(1), pp. 32-50, 2016. [Online]: http://nbn-resolving.de/urn:nbn:de:101:1-201705194936

[50] SPARQL Query Language for RDF. W3C Draft. www.w3.org/TR/rdf-sparql-query/, 15 January, 2008, [Accessed: 20-Aug-2017].

[51] J. Trujillo and S. Luján-Mora, "A UML-based approach for modeling ETL processes in data warehouses," *Conceptual Modeling-ER*, 2003.

[52] ValueWalk: "Google search now tracks local storms," Sends Public Alerts. 2015. Chatham: Newstex.

[53] P. Vassiliadis: "A generic and customizable framework for the design of ETL scenarios" in *Information Systems*, 30(7), pp. 492–525, 2005.

[54] C. M. White. "Social media, crisis communication, and emergency management: Leveraging Web 2.0 technologies". *CRC press*, 2011.

[55] W3C Consortium RDF (Resource Description Framework) http://www.w3.org/RDF. [Accessed: 20-Aug-2017].

[56] D. Yates and S. Paquette. "Emergency knowledge management and social media technologies: A case study of the 2010 Haitian earthquake." *International journal of information management* 31(1), pp. 6-13, 2011.

APPENDICES: MATCHING FUNCTIONS

```
boolean regionMatch(Node n1, Node n2) {
   if (n1.isLiteral() && n2.isLiteral()) {
      Object v1 = n1.getLiteralValue();
      Object v2 = n2.getLiteralValue();
      if (v1 instanceof String && v2 instanceof String) {
         String location = ((String) v1).toLowerCase();
         String[] locParts = location.split(",");
         String region = ((String) v2).toLowerCase();
         if(region.contains(locParts[0]))
            return true;
      }
   }
   return false;
}
```

Listing 1: The *RegionMatch* function for matching regions

```
boolean WeatherImpactMatch(Node wpoly, Node rlat, Node rlon) {
   Node wpoly = getArg(0, args, context);
   Node rLat = getArg(1, args, context);
   Node rLon = getArg(2, args, context);
   if (rLat.isLiteral() && rLon.isLiteral() && wpoly.isLiteral()) {
      String[] coordinates = ((String)wpoly.getLiteralValue()).split(",");
      Object rLatObj = rLat.getLiteralValue();
      Object rLonObj = rLon.getLiteralValue();
      if (rLatObj instanceof Float && rLonObj instanceof Float) {
         Float rLat = (Float) rLatObj;
         Float rLon = (Float) rLonObj;
         List<Point> points = new ArrayList<>();
         for(String coordinate: coordinates) {
            String[] vals = coordinate.split("_");
            points.add(new Point(Float.parseFloat(vals[0]),
Float.parseFloat(vals[1])));
         }
         return contains(points, new Point(rLat, rLon));
      }
   }
return false;
}

boolean contains(List<Point> points, Point test) {
   int i, j;
   boolean result = false;
   int size = points.size();
   for (i = 0, j = size - 1; i < size; j = i++) {
      if ((points.get(i).getLongitude() > test.getLongitude()) !=
          (points.get(j).getLongitude() > test.getLongitude()) &&
          (test.getLatitude() <
             (points.get(j).getLatitude() - points.get(i).getLatitude()) *
             (test.getLongitude() - points.get(i).getLongitude()) /
             (points.get(j).getLongitude() - points.get(i).getLongitude()) +
            points.get(i).getLatitude())) {
         result = !result;
      }
   }
   return result;
}
```

Listing 2: The function for matching weather impact

```
// Coordinates of the earthquake epicenter and the
// region are passed to this method.
boolean EarthQuakeImpactMatch(Node eLat, Node eLon, Node rLat, Node rLon,
                              Magnitude mag) {
   if (eLat.isLiteral() && eLon.isLiteral() && rLat.isLiteral() &&
       rLon.isLiteral() && mag.isLiteral()) {
      Object eLatObj = eLat.getLiteralValue();
      Object eLonObj = eLon.getLiteralValue();
      Object rLatObj = rLat.getLiteralValue();
      Object rLonObj = rLon.getLiteralValue();
      Object magObj = mag.getLiteralValue();
      if (eLatObj instanceof Float && eLonObj instanceof Float &&
          rLatObj instanceof Float && rLonObj instanceof Float &&
          magObj instanceof Float) {
         Float eLat = (Float) eLatObj;
         Float eLon = (Float) eLonObj;
         Float rLat = (Float) rLatObj;
         Float rLon = (Float) rLonObj;
         Float mag = (Float) magObj;
         float radiusInKm = computeEarthquakeRadius(mag);
         float radiusInDeg = radiusInKm / 110;
         return liesInsideEarthquake(eLat, eLon,  rLat, rLon, radiusInDeg);
      }
   }
   return false;
}

boolean liesInsideEarthquake(float cX, float cY, float x, float y, float r){
   float dx = x - cX;
   float dy = y - cY;

   return dx * dx + dy * dy <= r * r;
}
```

Listing 3: The function for matching earthquake impact

AUTHOR BIOGRAPHIES

Yogesh Pandey graduated with a M.S in Software Engineering from the School of Computing, Informatics, and Decision Systems Engineering in Arizona State University in December 2016. He received his Bachelor degree in Computer Science from Jaypee Institute of Information Technology in 2012. He is currently a Software Development Engineer II at Amazon Web Services. He has 3.5 years of prior industrial experience during which worked on a variety of technologies including web application development and distributed web services.

Srividya Bansal is an Associate Professor in Ira A. Fulton Schools of Engineering at Arizona State University. Her primary research focuses on semantic computing, semantics-based approaches for Big Data Integration, semantic querying of big data, web service description, discovery & composition, and tools for outcome-based instruction design in STEM education. She received a PhD in Computer Science from the University of Texas at Dallas in 2007 and Masters in Computer Science from Texas Tech University in 2002. She is a member of the IEEE Computer Society and Association of Computing Machinary (ACM). Contact her at srividya.bansal@asu.edu.