# Safe Neighborhood

## How safe is your neighborhood?

Author: Rajiv Prathapan: Software Engineering department,
Arizona State University
Tempe, AZ

Author: Sanjana Vasudevan: Software Engineering department,
Arizona State University
Tempe, AZ

Author: Prasanth Venugopal: Software Engineering department,
Arizona State University
Tempe, AZ

Author: Ejaz Saifudeen: Software Engineering department,
Arizona State University
Tempe, AZ

***Abstract*—**Generally, people tend to move to new cities and they must go about visiting different sites to know about the safety of the city in terms of fire, crimes, sanitation and natural calamities. A semantic web application which aims to be a one stop to know all that is needed to know about the safety of a neighborhood. Datasets are obtained from official US government sites and are linked in such a way that when the user searches for the safety of a neighborhood, all the relevant information is obtained.

To map this idea user is provided with the option to add multiple cities that he/she wants to compare. When the user selects the cities that he wants to compare, the dataset which contains all the data and mapped with the ontology is searched and a visual comparison is shown in the form of charts.

***Keywords—Semantic Web, Crime, Fire, Disease, Natural Disasters, Compare Cities, Neighborhood Safety***

## I. GOALS

The main goal is to create a data-driven semantic web application that using tools and technologies like XML, XSLT, RDF, RDF-S, OWL, SPARQL, Protégé, Apache Jena Semantic Web Framework. Four different data sources namely crime, fire, diseases and natural calamities for all the US cities were identified, collected, processed and integrated them using Semantic Computing concepts.

## II. RELATED WORK IN THE DOMAIN

Work in this direction is done previously by many sites like AreaVibes [4], NeighborhoodScout [5], FamilyWatchDog [3], CrimeReports [4], SpotCrime [5].

Enter a state, city, neighborhood, or address into the AreaVibes search box and the website instantly plots the location on a map and provides you with its livability score between one and 100. The site also maps and ranks nearby areas according to their scores and compares your area's livability score to state and national averages.

AreaVibes also assigns locations a letter grade from F to A+ for each of seven livability factors, one of which is crime.

Using NeighborhoodScout, you can compare neighborhood overall crime rate, violent crime rate, and property crime rate to the rates in other cities—as well as to the state and national average. Users can also outline a specific crime search area, whether by the distance around a city or by miles from an address. Plus, NeighborhoodScout can tell you what the chances are of becoming a crime victim in your neighborhood, and compare that with the city and the state odds.

Type a location or address into the website's search box, and FamilyWatchdog generates a map pinpointing the address of nearby registered sex offenders. Color-coded icons correspond to various sex crimes, including crimes against children, sexual battery, and rape. Click the icon and you'll see a picture of the offender, learn their aliases, and find out what sex crime they've been convicted of. If you're looking for a specific individual, you can search for them by name.

## III. UNIQUENESS OF PROJECT

There exists no such platform where the user can get all the data which included crime rate, fire accidents, diseases and natural calamities for all the US cities in one place.

The success of the application lies in the fact that the information from various sources merges into one semantic web model where a potential user can search, navigate, and query all the sources as if they are one model. The integrated data is made available as linked data in the Apache Fuseki server and the user could search and compare number of cities side by side, giving the customer combined and richer information than all the existing counterparts. The customer doesn't have to visit different sites to view all the different data. Our application will serve as a one stop to know all that is needed to know about the safety of a neighborhood and compare it with other cities so that they could decide.

## IV. DISCUSSION OF SEMANTIC DATA MODEL [1]

Semantic Web has five main components which help in accomplishing the required task and define the functioning of the web:

1. Uniform Resource Identifier:

A URI is simply a Web identifier: like the strings starting with "http:" or "ftp:" that you often find on the World Wide Web. Anyone can create a URI, and the ownership of them is clearly delegated, so they form an ideal base technology with which to build a global Web on top of. In fact, the World Wide Web is such a thing: anything that has a URI is considered to be "on the Web".

A URI may be classified as a locator (URL), or a name (URN), or both. A Uniform Resource Name (URN) functions like a person's name, while a Uniform Resource Locator (URL) resembles that person's street address. In other words: the URN defines an item's identity, while the URL provides a method for finding it.

The URI syntax consists of a URI scheme name followed by a colon character, and then by a scheme-specific part. The specifications that govern the schemes determine the syntax and semantics of the scheme-specific part, although the URI syntax does force all schemes to adhere to a certain generic syntax that, among other things, reserves certain characters for special purposes (without always identifying those purposes). The URI syntax also enforces restrictions on the scheme-specific part, in order to, for example, provide for a degree of consistency when the part has a hierarchical structure. *Percent encoding* can add extra information to a URI.

A *URI reference* is another type of string that represents a URI, and (in turn) represents the resource identified by that URI. Informal usage does not often maintain the distinction between a URI and a URI reference, but protocol documents should not allow for ambiguity.

A URI reference may take the form of a full URI, or just the scheme-specific portion of one, or even some trailing component thereof – even the empty string. An optional fragment identifier, preceded by #, may be present at the end of a URI reference. The part of the reference before the # indirectly identifies a resource, and the fragment identifier identifies some portion of that resource.

Web document markup languages frequently use URI references to point to other resources, such as external documents or specific portions of the same logical document.

**2. RDF:**

The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata data model. It has come to be used as a general method for conceptual description or modeling of information that is implemented in web resources, using a variety of syntax formats. [2]

The RDF data model is similar to classic conceptual modeling approaches such as Entity-Relationship or Class diagrams, as it is based upon the idea of making statements about resources (in particular Web resources) in the form of subject-predicate-object expressions. These expressions are known as *triples* in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. For example, one way to represent the notion "The sky has the color blue" in RDF is as the triple: a subject denoting "the sky", a predicate denoting "has the color", and an object denoting "blue". RDF is an abstract model with several serialization formats (i.e., file formats), and so the particular way in which a resource or triple is encoded varies from format to format.

A collection of RDF statements intrinsically represents a labeled, directed multi-graph. As such, an RDF- based data model is more naturally suited to certain kinds of knowledge representation than the relational model and other ontological models. However, in practice, RDF data is often persisted in relational database or native representations also called Triplestores, or Quad stores if context (i.e. the named graph) is also persisted for each RDF triple. As RDFS and OWL demonstrate, additional ontology languages can be built upon RDF.

The subject of an RDF statement is either a Uniform Resource Identifier (URI) or a blank node, both of which denote resources. Resources indicated by blank nodes are called anonymous resources. They are not directly identifiable from the RDF statement. The predicate is a URI which also indicates a resource, representing a relationship. The object is a URI, blank node or a Unicode string literal.

In our application RDFs were created by converting CSV data fetched from various government sources that are available under the domain data.gov to RDF triples with help of google refine. These RDF files help in mapping the ontology elements to the URIs and generating triples.

**3. OWL:**

The Web Ontology Language (OWL) is a family of knowledge representation languages for authoring ontologies endorsed by the World Wide Web Consortium. They are characterized by formal semantics and RDF/XML- based serializations for the Semantic Web.

The data described by an ontology in the OWL family is interpreted as a set of "individuals" and a set of "property assertions" which relate these individuals to each other. An ontology consists of a set of axioms which place constraints on sets of individuals (called "classes") and the types of relationships permitted between them. These axioms provide semantics by allowing systems to infer additional information based on the data explicitly provided. [3]

The classes in OWL were generated using Protégé. The concept about the data fetched and needed for executing user queries was generated as classes or properties to relate them. And the various relations were set up which integrated the various data fetched from various sources.

**4. Open Refine**

OpenRefine, formerly called *Google Refine*, is a standalone open source desktop application for data cleanup and transformation to other formats, the activity known as data wrangling. It works in 3 steps of exploration of data, cleaning and transformation of data and reconciliation and matching of data.
It takes up the data in CSV format extracted from the individual python codes and converts the data to RDF format by mapping it to the classes of the ontologies. This is performed by an adding an RDF extension.
Thus, the data in RDF triples is generated.

**5. Apache Jena and Fuseki Server**

Apache Jena is an open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL.
Fuseki is a SPARQL server. It provides REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP.
Thus, the data generated in different RDF is clubbed to a single RDF file and the triple are merged in fuseki and that data is queried using Jena+fuseki framework.

## V. CHALLENGES FACED IN INTEGRATING DATA

1. Vastness: The World Wide Web contains many different data sources which contain crime, fire, disease and natural disaster data and existing technology has not yet been able to eliminate all semantically duplicated terms. Any automated reasoning system will have to deal with truly huge inputs. Thus, combination of data from varied wide data sources can be tedious.

2. Vagueness: Imprecise concepts arises from the vagueness of data available on the web. Some of the data available were worldwide occurrence counts and it did not have occurrence instances which made giving any kind of semantic meaning to these data impossible.

3. Uncertainty: These are precise concepts with uncertain values. For example, a crime instance maybe given for a particular city but there would not be the type of crime associated with it which makes classifying it difficult. These data often lead to improper total count of the crimes and thus resulting in an inconsistent meaning to the data.

4. Inconsistency: These are logical contradictions which will inevitably arise during the development of large ontologies. Deductive reasoning fails catastrophically when faced with inconsistency, because "anything follows from a contradiction ". For example, the total count of crimes will be different from all the cities added up. We were not able to gather Fire data as only Phoenix data was at Zipcode Level while other cities had longitude and latitude level. The conversion of coordinates were paid service, hence we were not able to convert and use the dataset.

5. Deceit: This is when the producer of the information is intentionally misleading the consumer of the information. There can be instances of crimes which have not be tried yet and cannot be counted as a crime instance.

## VI. GENERATION OF INSTANCE DATA

The project contains 4 datasets which initially was in CSV format. Using the data from the given datasets, we were able to gain insights on the number of events which include Crime, Natural Disaster, Disease outbreaks, Fire outbreaks. We were able to capture the number of incidents for each subtype of an event in Zipcode level. For Natural disaster and Disease outbreak, the data is captured in state level. Data in CSV format is then converted into RDF triple format. This is performed by the use of OpenRefine (formerly called Google Refine). OpenRefine provides easy handling of large data sets and its cleaning. It also provides many data transformation functionalities. Using the RDF Extension 0.8.0 for exporting as RDF, raw data in the form of CSV files or spreadsheets are uploaded to this offline tool to transform the data in RDF/XML format. Some data transformations are then performed to get only the desired data from the csv files in the RDF files. After obtaining the data in the desired form, we design the following skeleton using "Edit RDF Skeleton..." command available under the RDF menu.

- Set the base URI for the RDF file
- *Add Prefix* option is used to add the ontology prefix and upload the ontology from the local machine
- Match the properties of the added ontology to the columns of the tabular data

RDF data is then extracted as RDF/XML from the Export menu. This way the RDF instance data are generated.

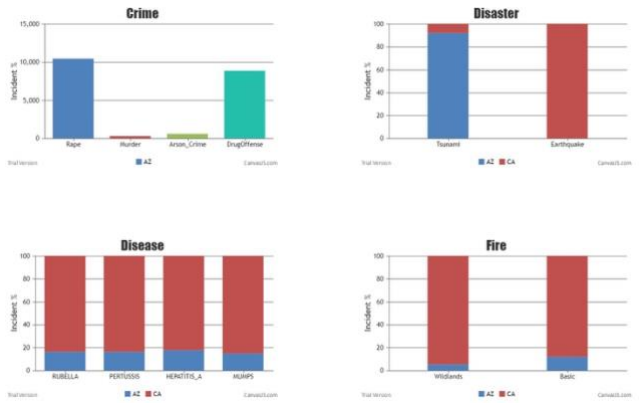
## VII. QUERYING THE LINKED DATA

The next step of the process is to query the aggregated RDF data. Like SQL queries the tables of a relational database, SPARQL query language is used in the Semantic data model that can query RDF triples. SPARQL is a W3C standard. Pattern matching is the basic idea behind the working of SPARQL. Similar to SQL, SPARQL chooses data from the RDF query data set by the use of a SELECT statement to figure out which subset of the selected data is returned. Additionally, SPARQL utilizes a WHERE clause to define graph pattern to discover a match for in the query data set. A graph pattern in a SPARQL WHERE clause consists of the subject, predicate and object triple to find a match for in the data. Following is two of the queries applied in the project for generating the chart and Heat map.

Chart SPARQL Query and Output:

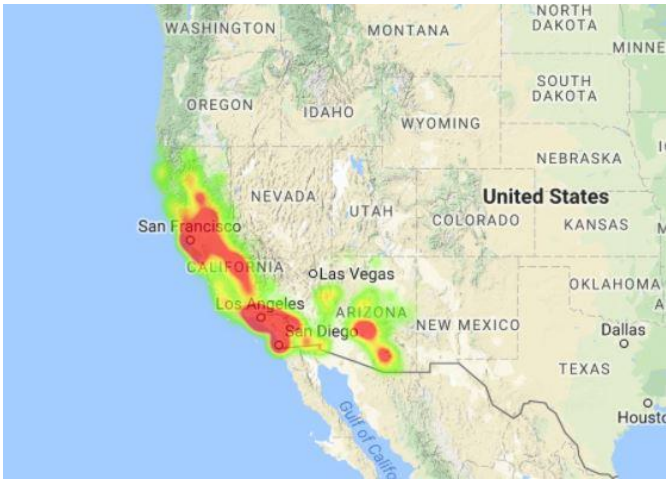
```
PREFIX sn:
<http://www.semanticweb.org/ejaz/ontologies/2017/9/SafeNeighborhood#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT * WHERE {{ SELECT (?state as ?location) ?type
```

```
        (count(?type) as ?count)
        WHERE {    ?r rdf:type ?type.
                   ?r sn:occured_at ?loc.
                   ?loc rdfs:subClassOf ?state.
         FILTER (?state IN (sn:AZ, sn:CA))
        FILTER (?type IN (sn:Arson_Crime, sn:DrugOffense,
sn:Murder, sn:Rape, sn:Theft, sn:Arson_Fire, sn:Basic,
sn:Wildlands, sn:Earthquake, sn:Tsunami, sn:Volcano,
sn:DIPHTHERIA, sn:HEPATITIS_A, sn:MEASELS,
sn:MUMPS, sn:PERTUSSIS, sn:POLIO, sn:RUBELLA,
sn:SMALLPOX)) }
        GROUP BY ?state  ?type}
        UNION {SELECT (?loc as ?location) ?type
(count(?type) as ?count)
        WHERE
         {    ?r rdf:type ?type.    ?r sn:occured_at ?loc
        .OPTIONAL  { ?r sn:incidents_per_1000000 ?i .
FILTER (?i > 0) }
FILTER (?loc IN (sn:AZ, sn:CA))
FILTER (?type IN (sn:Arson_Crime, sn:DrugOffense,
sn:Murder, sn:Rape, sn:Theft, sn:Arson_Fire, sn:Basic,
sn:Wildlands, sn:Earthquake, sn:Tsunami, sn:Volcano,
sn:DIPHTHERIA, sn:HEPATITIS_A, sn:MEASELS,
sn:MUMPS, sn:PERTUSSIS, sn:POLIO, sn:RUBELLA,
sn:SMALLPOX)) }
        GROUP BY ?loc ?type }}
        ORDER BY ?location
```




Heatmap SPARQL query and Output:

```
PREFIX sn:
<http://www.semanticweb.org/ejaz/ontologies/2017/9/SafeNeighborhood#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT * WHERE {{ SELECT
         (?loc as ?location)
         (count(?loc) as ?count)
        WHERE {    ?r rdf:type ?type.
         ?r sn:occured_at ?loc.
        ?loc rdfs:subClassOf ?state.
        FILTER (?state IN (sn:AZ, sn:CA))
        FILTER (?type IN (sn:Basic)) } GROUP BY ?loc }
UNION {SELECT (?loc as ?location)
        (count(?loc) as ?count)
        WHERE {    ?r rdf:type ?type.
         ?r sn:occured_at ?loc.
        OPTIONAL  { ?r sn:incidents_per_1000000 ?i .
        FILTER (?i > 0) }
         FILTER (?loc IN (sn:AZ, sn:CA))
        FILTER (?type IN (sn:Basic)) }GROUP BY ?loc }}
```




The SELECT statement requests the three variables to be returned namely ?mastertype, ?type, and ?sport_loc. In SPARQL query language, variable names are prefixed with the question mark ("?") symbol. In the above SPARQL query, ?mastertype returns the type of the incident, ?type returns the subtype of the instance, and ?loc returns the count of the particular subtype instances in the database.

SPARQL queries are executed using Apache Jena Fuseki server. We ran Fuseki from inside a Java program to provide SPARQL services for the application. The advantage of using Fuseki server is that it provides REST-style SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP.

## VIII. KEY FUNCTIONALITIES PROVIDED BY THE APPLICATION

This app provides information for people who are trying to move into a new neighborhood because of job transfer or starting a new school and other reasons. There is not a single web application which provides a consolidated information of all the important data needed for one to make an informed decision on whether to move to a particular city or not.

All this information is provided on a single platform. The platform can be used to search for a particular city's data over the years or it can also be used to compare many cities data so that they could choose between the options they have.

The application helps in providing insights to user with the help of Heatmap and Column/ Stacked Column charts. With these tools the user can compare the number of events that have occurred in the locations that the user wishes to compare. The Application uses column chart when a single location is checked and the application uses a stacked column chart when more than 1 location is chosen where the number of incidents are displayed in a 100% scale with a display showing the percentage of the incident that has occurred for each location.

## ACKNOWLEDGMENT

We would like to show our warm thank to Dr. Srividya Bansal who supported us at every bit and without whom it was impossible to accomplish the end task.

## REFERENCES

[1] Berners-Lee, Tim; James Hendler; Ora Lassila (May 17, 2001). "The Semantic Web". Scientific American Magazine. Retrieved March 26, 2008.

[2] Optimized Index Structures for Querying RDF from the Web Andreas Harth, Stefan Decker, 3rd Latin American Web Congress, Buenos Aires, Argentina, October 31 to November 2, 2005, pp. 71–80.

[3] "OWL 2 Web Ontology Language Document Overview". W3C. 2009-10-27.

[4] "AreaVibes" - http://www.areavibes.com/

[5] "NeighborhoodScout" - https://www.neighborhoodscout.com/

[6] "FamilyWatchDog"- http://www.familywatchdog.us/

[7] "Crime Reports"- https://www.crimereports.com/

[8] "SpotCrime"- https://spotcrime.com/