EPIC-OSM: A Software Framework for OpenStreetMap Data Analytics

Jennings Anderson Robert Soden Kenneth M. Anderson Marina Kogan Leysia Palen
University of Colorado Boulder
{jennings.anderson, robert.soden, ken.anderson, marina.kogan, leysia.palen}@colorado.edu

Abstract

An important area of work in big data software engineering involves the design and development of software frameworks for data-intensive systems that perform large-scale data collection and analysis. We report on our work to design and develop a software framework for analyzing the collaborative editing behavior of OpenStreetMap users when working on the task of crisis mapping. Crisis mapping occurs after a disaster or humanitarian crisis and involves the coordination of a distributed set of users who collaboratively work to improve the quality of the map for the impacted area in support of emergency response efforts. Our paper presents the challenges related to the analysis of OpenStreetMap and how our software framework tackles those challenges to enable the efficient processing of gigabytes of OpenStreetMap data. Our framework has already been deployed to analyze crisis mapping efforts in 2015 and has an active development community.

1. Introduction

We live at a time when organizations of all kinds increasingly have the means to generate, collect, and analyze large volumes of data via software systems. These systems—collectively known as *data-intensive software systems* or "big data" systems—are challenging to design, develop, and deploy [1]. One application area that requires the development of these systems is *crisis informatics* [12], which investigates how social computing can impact the practice of emergency management. Of particular interest is the use of digital maps to support disaster response, an activity known as *crisis mapping*.

However, the analytics of geospatial data are especially challenging to resolve. This is because a) map datasets tend to be extremely large—often consuming terabytes or petabytes of information—and b) map datasets are not good at conveying how they were created. That is, for any given version of a map, all one sees is the final aggregate map, not the individual edits that were performed to create it. This is what separates collaboratively-edited geospatial data from collaboratively-edited text documents—such as articles on Wikipedia—which can much more easily display editing history across users.

In the new world of crowdsourced data generation where information can be produced quickly for open use, understanding the *collaboration* that went into the construction of the map can be *as important as the map itself*. This is especially true for action-oriented communities, like the crisis mapping community, that are trying to understand their evolving work practices while they work to produce maps that can be used to aid crisis response. These communities seek to understand their work in situ to improve upon it. Social computing researchers desire the same understanding to both document what digital crowds can achieve and with an eye towards designing better tools to support that work in the future. For the big data community, this type of research is important, as it requires the novel use of data analysis techniques both for the batch processing of existing data sets as well as the real-time analysis of edits that stream in during a crisis event.

In this paper, we report on the design and development of a big data software framework that can be used to analyze the edit history of OpenStreetMap (OSM), making it possible to study the cooperative work that occurs there, including but not limited to the intensely collaborative periods of crisis mapping where much is at stake for humanitarian groups using these maps on the ground. At the time of this writing, there are no other frameworks that perform this type of analysis for OSM data; indeed, use of our software framework has been steadily increasing since its initial deployment for studying and monitoring the mapping activity surrounding the 2015 Nepal Earthquake event. This increased use is the direct result of the unique analysis capabilities our framework provides on top of OSM data.

OpenStreetMap is an open geographic data initiative that provides a map, and its associated geospatial data, that anyone can contribute to and access. Our software framework, known as *epic-osm*, can scale to process gigabytes of OSM data by employing a variety of techniques to both analyze

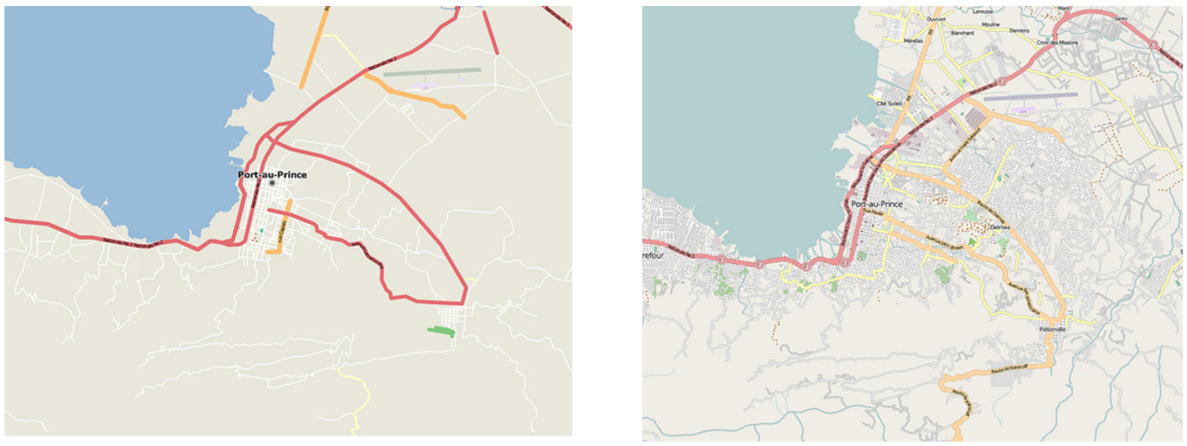


Figure 1: Port-Au-Prince, Haiti in OSM, before the 2010 earthquake (left) and 4 days after (right). Remotely-located volunteer mappers added all features by tracing aerial imagery [8].

data for desired metrics and visualize the results in ways that are meaningful to mappers themselves and the larger OSM organization and community. It also makes details of this enormous data-producing organization [13] available to researchers in the way that Wikipedia has been studied extensively for years as a notable site of collaborative data production. Our framework is more than just a design; our code is available on GitHub and the software tools that have been built on top of this framework are in active use.

Our experiences designing and implementing this framework can be of use to others. We demonstrate how to address the challenging data modeling issues that arise in the design of data-intensive software systems [16], as well as issues of extensibility, scalability, and interoperability.

1.1. Studying the OSM Community

With some notable exceptions [9], the majority of research on the social organization of the OSM community has been based upon qualitative research methods such as participant observation, interviews, and surveys. These studies have provided insights into participant motivation \cite{Budhathoki2012} and demographics \cite{Schmidt2013}. Our team saw that examination of the OSM database *itself*—which contains a complete record of every edit ever made—is critical to the advancement of the 2.1M OSM member organization, which needs to better understand its production functions to manage its growth [13], as well as for social computing researchers to characterize the nature of cooperative crisis mapping. Understanding the social processes governing the creation of OSM data is especially important for crisis informatics, since these behavioral phenomena can affect the quality of the geographic data produced. This can have real human consequences as OSM is frequently used as the primary base map in humanitarian response [18].

One likely reason that so little analytical research of socio-behavioral phenomena in OSM has been conducted (in comparison to the vastly-studied Wikipedia organization) is the challenges of manipulating OSM data. A complete download of the OSM history database is over a terabyte in size and is continuously growing as new edits are made. This difficulty affects not only scholars, but also the OSM community itself, which struggles to track its own activity, and hence its growth and impact [13]. To address this knowledge gap, we have identified a number of OSM members who have been willing to contribute to the development of the epic-osm framework as well as deploy and test it for a range of purposes. As will be discussed, this engagement has helped push the development of our framework and its surrounding toolset in new directions. Furthermore, this has catalyzed discussion within the OSM community about the need for new tools, as the existing community toolset, prior to the creation of our framework, is sparse and does not provide in-depth analytical capabilities.

1.2. Crisis Informatics and OpenStreetMap

When a major disaster occurs, a subset of the OSM community rapidly converges on the map around the impacted geographical area. The first well-documented case of this was after the 2010 Haiti Earthquake, where what few mapping products did exist were lost to the destruction of the office buildings of the national mapping agency. The international humanitarian responders converging onto the scene needed accurate maps to perform their work [18]. As depicted in Figure 1, hundreds of remote mappers from all over the world dramatically improved the digital map coverage of the affected areas in a matter of days by digitally tracing aerial imagery to build the map. This map then became the primary resource used in relief efforts [18].

Known as *high-tempo events*, these activations are of interest to the OSM community as a way to understand and communicate its impact. It is also of specific interest to crisis informatics researchers because of the rapid, large-scale convergence of "digital volunteers" from around the world, which demonstrates new forms of collective behavior [7, 13]. However, to begin asking questions of how this collaboration occurs, we must first create new tools to access and explore the "site of work"—the database supporting the map itself. This is the motivation behind the development of epic-osm—to create the first open framework for easily analyzing the large OSM dataset. Initially developed to support crisis informatics research, the use cases we will discuss are abundant and the framework provides great flexibility for all types of OSM research.

2. OpenStreetMap

Created in 2004 by students in the UK in response to restrictive licensing on geographic data \cite{Chilton2009a}, OSM has become the most widely used platform for "volunteered geographic information" \cite{ Elwood2008, Goodchild2007}. OSM is supported by a worldwide network of developers and volunteers committed to the open data values of the platform. Today, OSM has over 2.1M registered users, a small subset of whom are active editors [10], and 2.9B individual geographic points [11]. The website itself is a Ruby on Rails application on top of a PostgreSQL database. OSM incorporates an in-browser map editor and provides an API to interface with external tools.

2.1. OSM Data Structure

Six *domain-level* data types are found in the OSM database. Three of these primary objects construct the map itself: nodes, ways, and relations. *Nodes* are the most basic building blocks of the database and represent single geographic points. A *way* is composed of an ordered series of nodes, representing a line or polygon. A *relation* is a collection of nodes and/or ways, such as a country border or a noncontiguous set of polygons. When an object is first created, its version is set to "1." Any subsequent edit to that object will increment the version number; such edits also track the user who performed them and the *changeset* (discussed below) to which this edit belongs. Representations of nodes, ways, and relations are shown in Figure 2.

Beyond the primary map objects, the OSM database contains changesets, users, and notes. A *changeset* is the digital receipt associated with every edit to the map. Each time a user commits their edits to the database, a changeset is generated with information about the editing session. The changeset id is recorded with every map object it contains, allowing a user to view a complete grouping of all the objects edited within a single changeset.

A *note* object is a geographically-located comment that a user adds to the map. These notes are marked as either open or resolved and may contain a comment thread as users discuss the note. Notes document a discussion between users on how to represent a feature on the map, which can be another important element for understanding map creation.

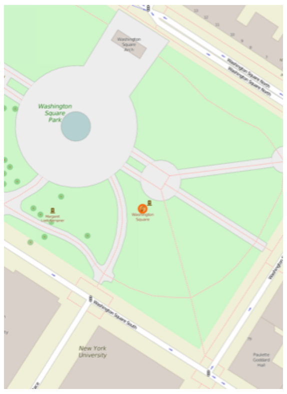
The OSM *user* database contains the user display name, a unique user id, and the date on which the user created an account on *openstreetmap.org*. epic-osm makes use of the date when a user creates an account to determine their experience level with OSM. This facilitates comparison of behavioral differences between novice and experienced editors.

2.1. Tags

The descriptive, non-spatial characteristic of each map object within OSM is a set of tags. These are unrestricted key-value pairs that can be added to any map object. An active wiki supports discussion about best tagging practices for consistency within the map, and editing tools offer default tag suggestions, but there are no database rules to enforce tagging schema or structure. For instance, Table 1 shows some of the top keys and common values for OSM objects in the map for New York City at the time of writing. From this table we can observe that information regarding the building footprints and heights for NYC is of major interest to the subset of the OSM community mapping in NYC, and is therefore not representative of all cities within OSM. This highlights the non-uniform characteristics of OSM contributions, calling for analysis tools that are capable of handling this dynamic nature.

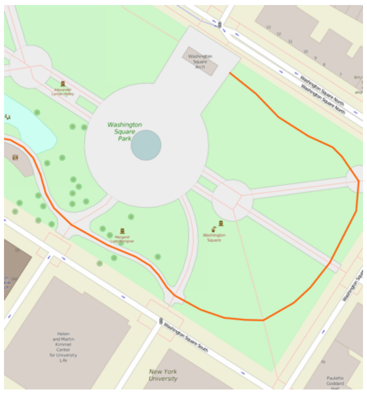
Table 1: Top Tags for OSM objects in NYC.

Objects w/ tag	*Key*	*Most-common values*
66%	building	garage, house, school
64%	height	8.2, 8.0
13%	highway	residential
11%	name	(various)
2%	amenity	parking, bicycle parking



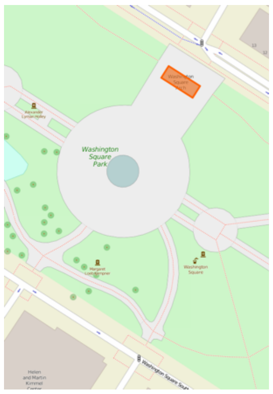
(a) Node
A drinking fountain as a single pair of coordinates.

```
lat: 40.7303993,
lon: -73.9970100,
version: 1,
tags: {
 amenity:
 drinking_water,
 name:
 Washington
Square}
```



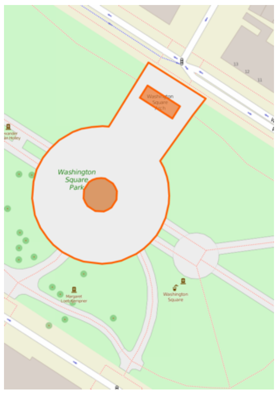
(b) Way: Path
A series of 41 nodes which create this footway

```
id: 197582876,
changeset: 31859815,
uid: 1306,
version: 2,
timestamp: 2015-06-
10T03:06:09Z,
tags: {
 highway: footway}
```



(c) Way: Building
Series of 4 nodes that outline the arch

```
id: 248166269,
tags: {
 building: yes,
 height: 20.5,
 name:
Washington
Square Arch
 tourism:
attraction}
```



(d) Relation: Path
A collection of 3 ways creating a footway

```
members: [
 {way: archId},
 {way: poolId},
 {way: parkId}
]
tags: {
 highway:
pedestrian}
```

Figure 2: OSM Objects as Rendered on openstreetmap.org. Each object shows various aspects of possible metadata (truncated) associated with OSM objects. Data © OpenStreetMap contributors.

Map rendering software then uses these tags to properly display an object. For example, a way tagged with `{"highway":"pedestrian"}` represents a path, while a way tagged as `{"building":"yes"}` represents a building. Examples can be seen in Fig. 2.

The importance of tags in OSM analysis cannot be overstated. However, given the open and dynamic nature of tags and tagging practices as the map evolves, an analysis tool must be robust to handle filtering by tags. For example, it is common for current OSM analyses to report summary statistics of OSM data by reporting on the number of new nodes added to the database. However, reporting that 956,725 nodes were added to the map in the month after the 2010 Haiti earthquake reveals very little about the manner in which the collaborative mapping was achieved. Filtering and sorting intelligently with tags instead can achieve results like this:

"308 users added 40,067 roads to the map and 162 users added 20,696 buildings to the map. 148 of these users were the same, adding buildings and roads."

Even this first-step expansion is a much richer summary of user contributions. The requirement, therefore, to develop a framework that is tag-aware is critical in understanding the creation of the map. As a result, epic-osm has advanced support for tags, and a mechanism for incorporating knowledge about the types of tags that the OSM community uses to create its maps (see Section 3.5). It can use this mechanism to find "all buildings" in a region even though different users tag buildings in different ways.

2.2. Planet Files

OSM provides its data in a common XML format via a RESTful API. Unfortunately for our analysis, this data represents the current state of the map, or the most recent version of the map objects, which, as we discussed above, is not of primary interest to those who study crisis mapping and the creation of the map itself. More useful are the "full-history planet files" that OSM strives to make available for download on a weekly basis. These files are bulk exports of the complete OSM database containing every edit to every object. Available in the Google protocol buffer format (PBF), these files are about 60gb in size, whereas the uncompressed history database in the OSM XML format is over a terabyte in size. While the PBF exports make obtaining the full history easier, working with the files requires specific knowledge of the file format and structure, and is computationally intensive to manipulate. This creates a requirement for an analysis framework: any OSM analytical framework must be able to handle the processing of full-history PBF files, which will continually grow in size as the OSM community continues to work.

3. epic-osm Framework

This section describes the current implementation of the framework and its features. epic-osm has supported crisis informatics research throughout its development. This iterative, domain-driven approach to development has been shown to be useful when creating data-intensive systems \cite{Barrenechea2015a}. As we refined our OSM research questions, the framework was adapted and refactored to support the processing of those questions. This agile development process has enhanced the usability and capabilities of the framework, thus supporting a main design goal which was to encourage the adoption and use of the framework among the many different communities interested in better understanding OSM data and mapping practices.

3.1. Features

The central object in our software framework is called an *analysis window (aw)*. This is a spatio-temporal bounding box for a researcher's given geographic area and time frame of interest. All data analyses operate within the scope of an analysis window. An analysis window is thus defined by specific start and end times and a set of polygonal geographic bounding boxes; in addition, an analysis window includes the queries to be performed on that subset of the database and other metadata such as the the contact person and associated data directories.

The framework does not limit the size or timeframe of an analysis window. However, we recommend working with a bounded analysis, especially during initial research. Since OSM is home to many different types of mappers with a great deal of variance around mapping practices, careful boundedness in space and time will yield results that are easier to interpret; one can then build on those results with progressively larger bounds, if desired.

3.2. Queries

Queries are associated with a specific analysis window and a specific temporal unit of analysis. Since every OSM object has a date and time associated with its creation, all queries return data sorted by these common features. A specific time unit for analysis can currently be set to *hour, day, month,* and *year.* These increments are then used to create time buckets for sorting the data returned. All queries return arrays of the form:

```
[{start_of_aw, bucket_end, results},
 {bucket_start, bucket_end, results},
 ...,
 {bucket_start, end_of_aw, results}]
```

The first bucket will always start at the beginning of the analysis window and will end on the first unit of analysis after that. For example, if the unit were specified as "month" and the analysis window started at 2014/06/15, then the first bucket would include results from this date up to 2014/07/01. The second bucket would include all data for the range 2014/07/01 to 2014/08/01. This design decision ensures that the colloquial units of analysis make sense. If a user is looking to perform an analysis on months, then their results are returned in time buckets of the common month, not a grouping of 28 days starting from the beginning of the analysis window. In the event no unit of analysis is specified, then a query will return an array with one item:

```
[ {start_of_aw, end_of_aw, results} ]
```

The framework is therefore designed to treat time as the default structure for analysis. This design decision supports the current practices in crisis informatics research and other observers of time- and safety-critical events. This makes our framework unique in comparison to other OSM data services that return the map data as it exists in real-time such as the official OSM API. These services are designed to deliver up-to-date geospatial data and map rendering, while epic-osm is designed for analysis of user contributions within a given period of time.

Furthermore, this ensures the results that are returned by queries represent individual edits, not necessarily distinct map objects. In other words, the same map objects with different versions may appear across multiple buckets of returned results. This allows users to explore the creation of the map by tracking changes to individual objects through time.

3.3. Conceptual Framework

In Figure 3, we show the semantic relationships between the various data objects in our framework.

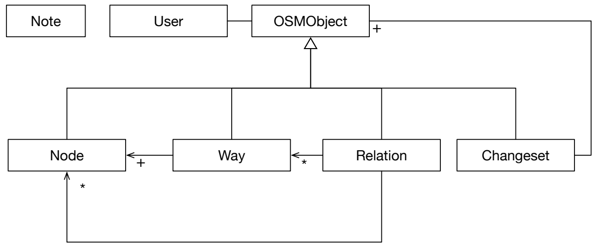


Figure 3: The domain objects of epic-osm.

The root class is OSMObject; it has attributes such as geometry, date created, user id, object id, and version number. Each OSMObject has an associated user who edited that particular version of that particular object. Nodes, ways, relations, and changesets are all subclasses of OSMObject.

The UML diagram shows that ways consist of one or more nodes and relations consist of some number of nodes and ways. While in practice this is true, our analysis framework performs extra work during import to ensure that each of these objects stands on its own. In particular, when importing a way, we traverse all of its associated nodes and embed the geographic information of those nodes in the way itself. We do the same thing for a relation, accessing all of its associated nodes and/or ways and embedding these objects into the relation itself. Therefore, when epic-osm performs a query on ways or relations, the query only has to access way or relation objects in epic-osm's persistence layer.

The decision to perform this extra work during import was twofold: a) improving run-time performance and b) reducing complexity during analysis. With respect to the former, we did not want to incur a run-time penalty during an analysis workflow spending time accessing a way or relation's constituent parts. With respect to the latter, users may edit attributes of either the way or relation itself, or the nodes and/or ways associated with it. In such cases, the associated objects may not be aware of these changes. To properly reconstruct the object requires resolving the geometries based on dates and changeset ids and "burning-in" the geometry as it existed in that specific version of a way or relation. We determined it was best to absorb this computational cost just once during import. This type of tradeoff is common in the design of big data software frameworks.

Changesets contain information about the editing session such as a geographical bounding box of the extents of the user's edits and the length of the editing session. Changesets themselves are unaware of the objects contained within the editing session, but the edited objects contain the changeset id of the changeset in which they were edited, allowing these relationships to be established after the fact. Note: although the semantics of our UML diagram allow changesets to include other changesets, this does not happen in practice: each changeset stands on its own and does not reference other changesets. Finally, our notes class contains attributes that allow OSM notes to be retrieved from the database and analyzed.

Figure 4 presents the framework classes that are used to perform an analysis at run-time. An instance of EpicOSM acts as a controller for the analysis session, creating the requested analysis window, asking it to connect to the database, and invoking its associated queries. The QuestionAsker acts as a proxy for the user who invoked epic-osm, and can influence where the results of the analysis are stored, provide other metadata about the invoking user, or further process the results of the invoked queries. The classes in Figures 3 and 4 are connected because query objects return instances of the domain objects. Thus, node queries will return instances of nodes that can then be further analyzed.

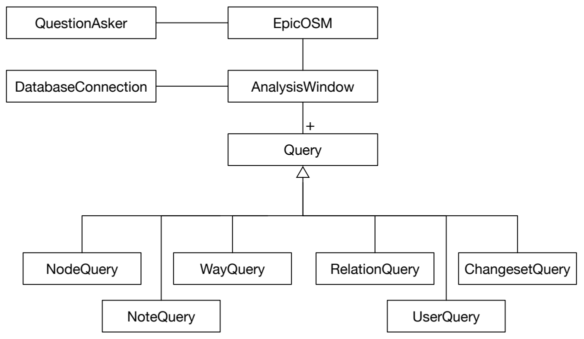


Figure 4: The run-time objects of epic-osm.

3.4. Current Technology Stack

In keeping with OSM's mission of open geospatial data, our framework is built on open source technologies. The logic of the framework is currently written in Ruby and is supported by a variety of open source libraries, developed by the greater OSM community and available on GitHub, for processing and importing OSM planet files.

Given the importance of OSM object tags and their key-value structure, we chose to use a NoSQL document database, MongoDB, with inherent key-value support for persistence. Mongo stores each domain-level OSM object in namesake collections (i.e., nodes, ways, relations, etc.). Common fields such as date created, user id, changeset id, and geometry are indexed by MongoDB to speed up most queries; specific tags such as "highway" or "building" are indexed as well to support queries against these objects of interest.

3.4. Flexible Query Language

To support the goal of *extensibility*, our framework makes use of metaprogramming techniques [14] to avoid binding clients of the framework to a particular set of metrics and query methods. Metaprogramming facilities have been a part of programming languages for many years and include techniques such as "monkey patching" in

Ruby, Python, and Javascript and key-value observing in Objective C. In epic-osm, we make use of a feature provided by the Ruby run-time system known as "method missing." This feature is invoked whenever a client calls a method on an object that does not have an implementation of that method either within itself, its included modules, or its superclasses. Though normally this situation would generate an exception that can crash a running program, Ruby's runtime instead calls the object again this time on a method called *method_missing*. It passes to this method a description of the method the client was trying to invoke. If that object has an implementation of method_missing and it can handle the processing of the failed call, the call will instead succeed. If it cannot handle the invocation, then, finally, an exception will be raised.

In epic-osm, almost all querying-related methods are handled by method_missing. This convention allows us to handle a wide range of possible queries that can be expressed using a domain-specific language that our method parses at run-time and allows for new queries to be added in an incremental fashion. For instance, a call to the method *nodes_x_year* will be interpreted by an analysis window as a request to return all edited nodes that fall within its constraints, grouped by year. That same functionality (retrieving all nodes) can be invoked but have the data grouped in a different way by simply calling the method with a different argument after the 'x', i.e. *nodes_x_month* or *nodes_x_day*.

Since the desired structure of the results is defined by the name of the function, arguments passed to the queries are for further filtering of the results and are passed through epic-osm to MongoDB unaltered. This allows users to take advantage of MongoDB query capabilities in their own epic-osm queries. For instance, the query: `ways_x_month( constraints: {"tags.highway" => "pedestrian" })` will return every version of a way which represents a pedestrian footpath which was edited or created within the analysis window, grouped into months. In this example, epic-osm handles grouping the results of the query into months while MongoDB finds all of the relevant ways while ensuring that all returned ways have a tag called "highway" with the value "pedestrian." For improved performance, users can externally index the underlying MongoDB collections to support common queries.

3.5. Question Modules

As shown in Figure 4, query objects target a specific type of domain object: Node queries return nodes while note queries return notes. This modular design allows analysts to focus their queries on just the domain objects they need. However, many questions require querying multiple types of objects. epic-osm provides this type of query via the use of Ruby's support for modules.

A specific module is created that contains all of the code that is needed to query across multiple types of domain objects; this module exports a single method that can then be invoked on an analysis window to execute the query at run-time. As an example, consider the need to ask an analysis window about the number of schools that were edited within its geo-temporal bounding box. For this particular query, it is important to check both nodes and ways to find all possible schools "hiding" in the map. According to OSM's community guidelines, the best practice for marking a school on the map is with the tag: `{"amenity": "school"}`. However, the actual OSM object that should contain this tag is not strictly defined. Mappers are encouraged to use an area (a polygon comprised of a closed way) that outlines the school's geographic footprint; however, the Wiki also states that mappers can "place a node in the middle of the site if [he or she is] in a hurry" (wiki.openstreetmap.org/wiki/Tag:amenity=school).

As a result, the question of "how many schools were mapped during the analysis window" becomes far more complicated than a simple query for objects with the school amenity tag. Instead, one must query both the ways and the nodes collection, identify distinct versions of interest and then resolve any geographic overlap in which both a node and a way mark the same school. To illustrate this, Table 2 shows the results of this query for the 2010 Haiti Earthquake across different types of OSMObjects and shows how the numbers change when accounting for geographic overlaps:

Table 2: Differences in use of "school" tag.

Query: "amenity": "school"	*Nodes*	*Ways*	*Geo-Unique*
Added	145	41	166
Edited	32	27	57
Unique Sum	146	52	**173**

Ultimately, one may conclude that 173 schools were edited in Haiti within OSM in the month following the 2010 Haiti Earthquake.

As mentioned, these more complex queries are isolated into Ruby modules—that epic-osm calls *question modules* since they contain all the code needed to ask a particular, complex question—that are then accessed via a single method with all support code cleanly hidden away from the main classes of the framework.

If OSM community guidelines change for a particular tag, just the code in the relevant module has to change in response. If one analyst has a broader (or more narrow) definition of what constitutes a particular entity, they can create their own module for finding instances of that entity. These modules can then be easily shared and plugged into any instance of the framework.

This is important because defining questions such as "how many schools were edited" as shown above are not immediately straightforward, so turning that question into a single method within a reusable module ensures that all users abide by the same rules when querying the data.

This modular design has also affected the development process by encouraging developers to write many questions in separate modules and then refactor common helper functions into the analysis window to make them available to all other question modules, thereby making the functionality provided by the core objects more powerful over time.

4. Implementation

Above, we presented the concepts and capabilities contained in the epic-osm framework. Here, we discuss how we have created a set of tools that use the framework and some of their implementation-related concerns. The advantage of creating a framework that can be incorporated into a wide range of tools is the large number of analysis use cases that can then be supported. Our initial set of tools handles the processing of a large amount of OSM data via the use of batch processing. First, command line tools are used to download and import OSM history data into MongoDB. Second, an input file is used to specify the parameters of a desired analysis window along with the desired queries. Third, a command line tool was created to read the input file, create instances of the objects shown in Figure 4, and kick off the processing of the specified queries. The output of that process is a directory of easily read JSON files. This straightforward set of tools and components can be used to process gigabytes of map data, ensuring *scalability*.

It is important to note that this same framework can be incorporated into a web application and be used to dynamically query MongoDB in response to user commands; indeed, we plan to develop such tools and, as we discuss later, we have already made changes to the framework to allow for more real-time processing of OSM data by analysis windows. Next, we discuss a few additional implementation-related concerns in more detail.

4.1. Persistence Layer

As mentioned above, MongoDB is used to store OSM history data and to perform the bulk of the work with respect to the queries that users specify. Storing the history data in this way allows users to have the flexibility to easily track changes to their queries over time. For example, a user may define an analysis window for their hometown over the past month. With each new month, they can create a new analysis window with the same geographical bounds, but with new start and end dates. As the user learns more about their data through defining new questions, persistence of previous analysis windows allows them to rerun those questions without having to re-import the underlying data. Furthermore, using a database ensures that the size of objects referenced by an analysis window can scale beyond the physical memory constraints of a user's machine. While MongoDB was selected for its ease of use and deployment, any key-value store or document store could be used as the persistence layer for epic-osm.

4.2. Output

In an effort to support *interoperability* via many types of analysis and by not forcing OSM researchers to use a single tool, epic-osm writes output to a pre-defined file structure: a series of JSON files. These files can then be easily parsed and visualized by a variety of libraries and analysis tools, leaving the visual inspection and analysis environment open to a user's preference. Currently, we build a static website from these JSON files that can be used to view and easily share the results of the analysis but many other options for how to make use of these files from more interactive web-based dashboards to network analysis toolsets are being pursued, both by our group and the OSM community. These multiple pursuits validate our design decision to create a common output directory of single JSON files.

5. Use of the Framework

At the time of this writing, our framework has supported academic research by our group as well as OSM community members. The initial release was in support of our post hoc research on the growth of the OSM organization between 2010 and 2014 in response to two distinct humanitarian events [13]. This required the processing of a month's worth of historical OSM data for each event, consisting of edits by nearly 500 users and 1500 users, respectively. Since then, the framework has been

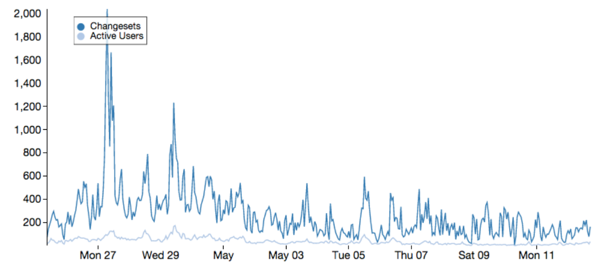


Figure 5: Count of OSM Changesets and Users. Graph shows the by-hour contributions to the map of Nepal after the April 25, 2015 earthquake.

available on GitHub and has been forked, contributed to, and adapted to support real time analysis and statistics of specific OSM mapping events.

For example, MapGive, a mapping initiative sponsored by the U.S. State Department, used epic-osm to visualize results of a competition between two universities to see which could create more data (mapgive.state.gov/events/mapoff). Additionally, it was deployed to monitor the first-ever mapping event at the White House (mapgive.state.gov/whmapathon). Another project, moabi.org, is also running an instance of the framework to monitor the mapping of logging roads in the Congo (loggingroads.org). The statistics are used to populate a "leaderboard" showing the highest-contributing users.

5.1. Nepal Earthquake Deployment & Improvements for Real Time Analysis

On April 25, 2015 a 7.8 magnitude earthquake struck central Nepal, killing over 8,500 people and destroying over 500,000 homes. Due to previous OSM work in the country [17], the city of Kathmandu was already mapped in detail. Yet many of the affected rural areas outside of Kathmandu were not well covered on the map. In what is believed to be the largest convergence of OSM mapping activity to date, over 7,000 contributors from all over the world mapped roads, buildings, and other features.

Our team deployed an instance of epic-osm immediately following the earthquake, which proved to be a valuable test case. A real-time import module developed by an epic-osm contributor that interfaces with a newly available OSM changeset streaming service (github.com/osmlab/osm-meta-util) supported this instance. Figure 5 illustrates this impressive convergence as tracked by epic-osm, showing the number of users editing and the number of changesets created per hour for the weeks following.

However, tracking this huge mapping activity in real-time exposed a problem. Designed to be a static snapshot in time that reads historical edits from a database, the analysis window could mimic near-real time results by running new queries every 10 minutes with bounds that spanned the time from the event to the current time. This solution worked well until the second day when the database had grown so large that the time it took to run the queries was longer than 10 minutes, creating a backlog.

To resolve this problem, we added a new feature: a rolling analysis window that would update the analysis window's constraints at each run to start at the top of the hour and end at the current time, thus never querying more than an hour's worth of data. These results were then output to separate directories, which could be iterated over to create the new totals. As a result, the framework was able to support a website providing visualizations of edits over the past hour. This site received over 1,700 unique visitors from 79 countries in the first week and was the OSM community's primary tool during the response for tracking its activity. This ad hoc solution worked in this particular use-case, but more importantly, exposed the weaknesses in the framework for similar use cases, which have since generated great interest in the OSM community.

6. Extensibility and Future Development

The desire to support both historical and real-time analysis of user contributions to OSM is strong across both industry and academia. At a June 2015 OSM conference (*The State of the Map US*) held in New York City, OSM users from the Red Cross, the US State Department, and three digital cartography-oriented start-up companies held a Birds-of-a-Feather discussion on the need for developing and supporting analysis tools such as epic-osm.

6.1. Stream Processing

The real time tracking of mapping activity in response to the Nepal earthquake identified a very powerful use-case for epic-osm that will significantly influence the next development iteration, specifically the ability to process the edits to the map as an incoming stream directly, instead of first importing to a database and extracting distinct time chunks. We will use contemporary big data solutions such as Apache Spark and its streaming capabilities to achieve better real time performance.

6.2. Database Improvements

With an emphasis on stream processing, the role of the persistence layer will also change in the next iteration. New user-level models will need to be developed to track mapping behavior, while the persistence of the individual object edits should also be preserved for later analysis, should users desire to perform new queries post-event. Alternative geo-spatial database technologies will be explored as well, which may improve query performance for geographic oriented analysis, such as "how many kilometers of a road did a particular user map?"

7. Conclusions

We have presented and discussed the design of epic-osm, the first full software framework to support the analysis of volunteered geographic information contributed to OSM. The framework was initially developed to support crisis informatics research surrounding the production of map data in two major crisis events, and has continued to grow and gain exposure to a larger community of developers and mappers alike, with hopes of allowing the entire OSM community to better reflect on its production of open geographical data. Our framework makes use of a number of techniques to efficiently handle large volumes of OSM data and serves as an example of how to design frameworks for data-intensive software systems. We believe that our framework, our lessons learned from initial deployments, and our iterative development approach, which is deeply grounded in empirical knowledge of a target domain—in this case, crisis mapping—will be of use to other designers and researchers of data-intensive software systems.

8. Acknowledgments

This material is based upon work sponsored by NSF Grants IIS-0910586 and IIS-1524806. We thank the OSM community for their involvement, particularly Mikel Maron of MapGive, Humanitarian OpenStreetMap Team and Kathmandu Living Labs.

9. References

[1] Anderson, K. Embrace the Challenges: Software Engineering in a Big Data World. In *Proc. 1st International Workshop on Big Data Software Engineering*, Part of 2015 Intl. Conf. on Software Engineering, pp. 19-25, IEEE.

[2] Barrenechea, M., Anderson, K., Palen, L., and White, J. Engineering Crowdwork for Disaster Events: The Human-Centered Development of a Lost-and-Found Tasking Environment. In Proc. *48th Hawaii International Conference on System Sciences*, pp. 182-191, IEEE 2015.

[3] Budhathoki, N., and Haythornthwaite, C. Motivation for Open Collaboration Crowd and Community Models and the Case of OpenStreetMap. American Behavioral Scientist. 2013. 54(5): 548-575.

[4] Chilton, S. Crowdsourcing is Radically Changing the Geodata Landscape: Case Study of OpenStreetMap. In *Proc. of UK Cartographic Conference,* 2009.

[5] Elwood, S. Volunteered Geographic Information: Future Research Directions Motivated by Critical, Participatory, and Feminist GIS. *GeoJournal*, 72(3&4): 173–183, 2008.

[6] Goodchild, M. Citizens as Sensors: The World of Volunteered Geography. *GeoJournal,* 69(4): 211–221, 2007.

[7] Keegan, B. Breaking News on Wikipedia: Dynamics, Structures, and Roles in High-Tempo Collaboration. In *Proc. of CSCW Companion,* pp. 315-318, 2012.

[8] Maron, M. Haiti OpenStreetMap Response. Blog. Jan. 14, 2010. Retrieved May 20, 2015 from http://brainoff.com/weblog/2010/01/14/1518

[9] Mooney, P., and Corcoran, P. Analysis of Interaction and Co-editing Patterns amongst OpenStreetMap Contributors. *Transactions in GIS*, *18*(5): 633–659, 2014.

[10] Neis, P., & Zipf, A. Analyzing the Contributor Activity of a Volunteered Geographic Information Project—The Case of OpenStreetMap. *ISPRS International Journal of Geo-Information*, 2012.

[11] OpenStreetMap Statistics. Accessed June 14, 2015. http://www.openstreetmap.org/stats/data_stats.html.

[12] Palen, L. and Liu, S. Citizen Communications in Crisis: Anticipating a Future of ICT-Supported Participation, In Proc. of *2007 Conference on Human Factors in Computing Systems, pp. 727-736.*

[13] Palen, L., Soden, R., Anderson, J. and Barrenechea, M. Success and Scale in a Data-Producing Organization: The Socio-Technical Evolution of OpenStreetMap in Response to Humanitarian Events In *Proc. of 2015 Conf. of Human Factors in Computing Systems, pp. 4113-4122.*

[14] Perrotta, P. Metaprogramming Ruby 2. The Pragmatic Programmers, LLC. 262 pages. 2014.

[15] Schmidt, M. and Klettner, S. Gender and Experience-related Motivators for Contributing to OpenStreetMap. In: Mooney, P. and Rehrl, K. (eds). International Workshop on Action and Interaction in Volunteered Geographic Information, pp. 13–18, 2013.

[16] Schram, A. and Anderson, K. MySQL to NoSQL: Data modeling challenges in supporting scalability. In Proc. of 3rd *Conf. on Systems, Programming, Languages and Applications: Software for Humanity*, pp. 191–202, 2012.

[17] Soden, R., Budhathoki, N., Palen, L. Resilience and the Crisis Informatics Agenda: Lessons Learned from Open Cities Kathmandu. In *Proc. of Conf. on Information Systems for Crisis Response and Management,* 2014.

[18] Soden, R., & Palen, L. From Crowdsourced Mapping to Community Mapping: The Post-Earthquake Work of OpenStreetMap Haiti. In *Proc of The 11th Intl. Conference of the Design of Cooperative Systems,* 2014.