Design Challenges/Solutions for Environments Supporting the Analysis of Social Media Data in Crisis Informatics Research

Kenneth M. Anderson, Ahmet Arif Aydin, Mario Barrenechea, Adam Cardenas, Mazin Hakeem, Sahar Jambi
University of Colorado Boulder
{ken.anderson,ahmet.aydin,mario.barrenechea,adam.cardenas,mazin.hakeem,sahar.jambi}@colorado.edu

Abstract

Crisis informatics investigates how society's pervasive access to technology is transforming how it responds to mass emergency events. To study this transformation, researchers require access to large sets of data that because of their volume and heterogeneous nature are difficult to collect and analyze. To address this concern, we have designed and implemented an environment—EPIC Analyze—that supports researchers with the collection and analysis of social media data. Our research has identified the types of components—such as NoSQL, MapReduce, caching, and search—needed to ensure that these services are reliable, scalable, extensible, and efficient. We describe the design challenges encountered—such as data modeling, time vs. space tradeoffs, and the need for a useful and usable system—when building EPIC Analyze and discuss its scalability, performance, and functionality.

1. Introduction

Crisis informatics is an area of research that studies how pervasive access to technology is transforming the way in which society responds to mass emergency events [14]. Crisis informatics has shown how human innovation around the use of social media has played a role in response to disasters such as floods [21], earthquakes [22], and hurricanes [29]. Two key challenges confront crisis informatics researchers: 1) the need to collect large amounts of social media data in real time as an event unfolds and 2) the need to filter the collected data set down to the size needed to answer common crisis informatics research questions.

The first challenge is critical since social media data—especially Twitter data—are ephemeral. That is, it is difficult to go back after an event and acquire a complete set of data. In the case of Twitter, the amount of information that is available via the Twitter Search API about an event in the past is several orders of magnitude smaller than what is available via the Twitter Streaming API as the event unfolds. Indeed, in most cases, past events are simply not available. The second challenge is also significant in that the size of a data set for an event collected from the Twitter Streaming API is typically in the tens of millions of tweets. When that is combined with post hoc collection of the tweets of "interesting users" via the Twitter REST API, the size of the data sets expand into the hundreds of millions. The challenge then is to reduce the size of these data sets by orders of magnitude to deliver a data set of thousands of *curated* tweets based on the research requirements of human analysts.

As part of Project EPIC [15], we have been working to address these challenges. With respect to the former, we designed and developed EPIC Collect, a software infrastructure that reliably collects large amounts of Twitter data [5, 18]. With respect to the latter, we aided Project EPIC analysts with a variety of manual tasks that involved extracting data from EPIC Collect to import it into third party analysis tools [6]. In this paper, we report on our work to automate these tasks by designing and developing EPIC Analyze, a comprehensive, scalable, and extensible data analysis environment for crisis informatics research.

The goal of EPIC Analyze is to provide an environment that addresses the second research challenge above: helping our analysts sift through enormous Twitter data sets to produce representative sets of tweets that they can use to answer socio-behavioral and socio-linguistic questions around mass emergency events. These questions include, e.g., "What types of information do people share in this event," "How do social networks form and do they persist after the event," and "How do people deal with stale information?" To answer these questions, a wide range of browsing, filtering, analysis, and annotation features are required. While we focus on analysts in this work, we believe our environment and its services will one day be useful for other audiences of social media data including researchers and practitioners in other domains, as well as everyday users. By studying the needs of analysts, we identify the features most needed for "advanced users" and can then determine how such features can be packaged for other audiences.

Since 2009, Project EPIC has developed a systematic methodology [10] for narrowing down a Twitter data set by multiple orders of magnitude so that the resulting set of tweets is either representative of the larger whole or a "found" subset of behaviors in the much larger set. In [10], we identified a comprehensive set of queries that analysts make in filtering down to users of interest. Prior to EPIC Analyze, we would *manually* engage in an iterative process with Project EPIC analysts that involved taking samples or filtering a given data set down to the exact types of behavior or information of interest for a particular event.

With EPIC Analyze, the need to conduct this process manually goes away. Analysts now can access large data sets in EPIC Analyze and perform multiple steps of browsing, filtering, and sampling to create the representative set of tweets on their own. Furthermore, analysts can do this on multiple data sets at once, helping to significantly increase the number of events that Project EPIC can study in parallel. In addition, EPIC Analyze is now the environment for integrating all of the analytical work that Project EPIC has generated, including our work on using natural language processing (NLP) to identify tweets that contain situational awareness information [28]. These services become additional tools that can be invoked by the analysts to help them with their work.

To ensure that EPIC Analyze is both scalable and extensible, we had to embrace heterogeneity at the software architectural level. EPIC Collect, for instance, makes use of Spring to manage a wide range of software services and Apache Cassandra, a NoSQL persistent data store, to store our Twitter data sets on a cluster of machines. In [18], we carefully documented the data modeling issues that needed to be solved to move to Cassandra, and how Cassandra met the requirements of reliable and robust data collection. In this paper, we discuss how we added new components (e.g. Hadoop, Pig, Redis, Ruby on Rails, Solr, and Splunk) to our existing environment to meet the analytical needs of Project EPIC in a scalable fashion.

2. The EPIC Analyze Platform

EPIC Analyze is a web application that acts as the front end to a software architecture that consists of a wide range of components and services. This front end presents a relatively simple facade that allows individual interactions with Project EPIC's data sets to be easily understandable while making use of the power provided by the backend services to browse, filter, sample, and annotate data sets consisting of millions of tweets. The web application's data model consists of a relatively simple set of concepts: analysts, affiliations, datasets, jobs, keywords, and tweets.

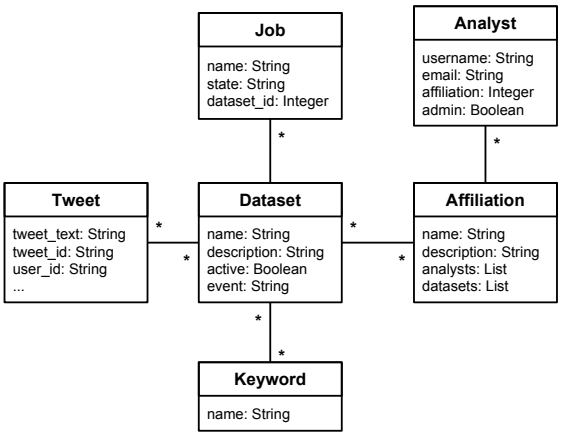

Figure 1. EPIC Analyze data model

Before we discuss the core functionality of EPIC Analyze, we first describe the data modeling issues we confronted when moving from a focus on data collection to data analysis. We then describe the heterogeneous software architecture adopted to meet our goals of producing a comprehensive, scalable, and extensible data analysis environment.

2.1. Data Modeling

To design and build EPIC Analyze, two data modeling issues had to be addressed. The first involves defining the data model of the front-end web application. The second involves identifying the primary unit of analysis within the environment and how to handle analysis results that do not conform to that format. *Data modeling is an extremely important issue to get right in the design of large-scale software infrastructure.* If one makes the wrong choices in the context of large data sets, one can end up with terabytes of data in the "wrong" format. In this context, "wrong" means that the format and/or structure of the data actively prevents useful work, perhaps by making it difficult to answer the type of questions that turned out to be the most important for analysis. As a result, we carefully considered each of the decisions that we made for the data modeling issues below and prototyped solutions that made use of them before relying on them to handle large data sets.

2.1.1. The EPIC Analyze Data Model. The web application for EPIC Analyze makes use of the data model shown in Fig. 1. This UML diagram presents a view of the data model showing the salient attributes needed by the core functions of the system. The primary concept of EPIC Analyze is the dataset. It tracks the name and description of a single dataset and maintains a pointer back to the event in EPIC Collect that was used to store the dataset's tweets in the first

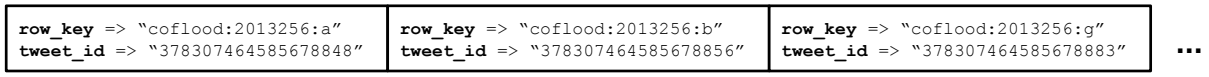

Figure 2. A tweet sequence for the 2013 Colorado floods dataset

place. In the future, we anticipate being able to associate a single dataset in EPIC Analyze with multiple events stored in EPIC Collect. For example, we often have generic terms like "earthquake" and "flood" collecting 24/7 in long-standing events, but we then create a new event with specific terms for a natural disaster when one occurs. The resulting dataset would then include tweets that contained both the generic and specific terms for the event.

Datasets consist of zero or more tweets, and tweets can be associated with zero or more datasets. The same tweet can be associated with more than one event by EPIC Collect. This can happen when, for instance, Project EPIC is collecting on more than one forest fire and both events are collecting on the term "wildfire." A dataset keeps track of the keywords used to collect its tweets, and these keywords can be associated with more than one dataset. Zero or more analysis jobs can be applied to a dataset. The dataset keeps track of those jobs to display a history of the jobs and the current status of any executing jobs (some jobs might take several hours to complete). Jobs in EPIC Analyze are described in more detail in Section 2.3.3.

The last two model objects—analyst and affiliation— allow EPIC Analyze to keep track of its users and their access rights to individual datasets. Each analyst belongs to a single affiliation, which is then associated with the datasets that they have permission to view and analyze. Project EPIC has several collaborations with external partners; the Twitter data sets collected for one partnership cannot be shared with the others. Affiliations allow Project EPIC analysts the ability to access all datasets while limiting access of our external partners to just the datasets relevant to that specific collaboration. We plan to add new concepts to this model—including project, annotation, and comment—to allow analysts to "mark up" a dataset and share that work with other analysts.

2.1.2. The Analysis Data Model. EPIC Analyze's primary purpose is to support the analysis of Twitter data sets consisting of millions of tweets. Thus, a key design goal was to ensure that the analysis and display of sequences of tweets is as efficient as possible. While Cassandra does an excellent job of storing our tweets in a reliable and robust fashion, it is not particularly suited for the task of supporting the random access and display of sequences of tweets. One cannot, for instance, ask Cassandra to provide a list of all its row keys in an efficient fashion. Instead, one must perform the equivalent of a "table scan" from the relational database world to ask for the first batch of row keys and then ask for the next batch and so on until all of the available keys have been retrieved. Likewise, one cannot ask a row for all of its column names in a single API call. One must, again, perform a scan to ask for the first batch of column names, and the next, and so on until a complete set is gathered. These scans take a significant amount of time (tens of minutes) which will only increase as we continue to add events and their associated tweets to EPIC Collect.

As a result, we needed a way to keep track of sequences of tweets related to a data set to enable the fast display and analysis of those tweets. When EPIC Analyze "imports" an event from EPIC Collect it creates a sequence of *tweet references* (see Fig. 2) that store for each tweet the row key where it can be found and its tweet id (which is the column name used to store the tweet's JSON object in that row). This initial sequence is sorted by tweet id, which means that it is also sorted by the time at which the tweet was created.

This sequence is stored in Redis, an in-memory database, that makes it efficient for EPIC Analyze to "paginate" a sequence of tweets for display, as well as to perform set operations on different tweet sequences that can be created as the result of filtering, sampling, and other analysis operations. For instance, an analysis job can create tweet sequences that track all of the tweets generated by highly active users, one sequence per user, while another job creates tweets sequences that track all tweets containing high frequency keywords, one per keyword. If we then want to display all tweets generated by an active user that contained one of the high frequency keywords, we do not have to run a new analysis job; instead we have Redis calculate the intersection of the two tweet sequences.

As these operations happen in memory, they are extremely fast. Indeed, via the use of tweet sequences and Redis, a dataset can have tens of thousands of pages of tweets and the display of the ten thousandth page is as fast as the display of the first. In both cases, our web application sends a request to Redis to retrieve the relevant page for the indicated tweet sequence. Redis quickly divides the sequence by the page size, locates the correct partition, and returns the tweet references for those tweets. The web application then asks Cassandra for those tweets, which efficiently returns them (in under a second) for display.

The benefit of this design is that there are many analysis tasks that have as output a sequence of tweets that match a given query or characteristic of interest. EPIC Analyze's browsing, filtering, and sampling

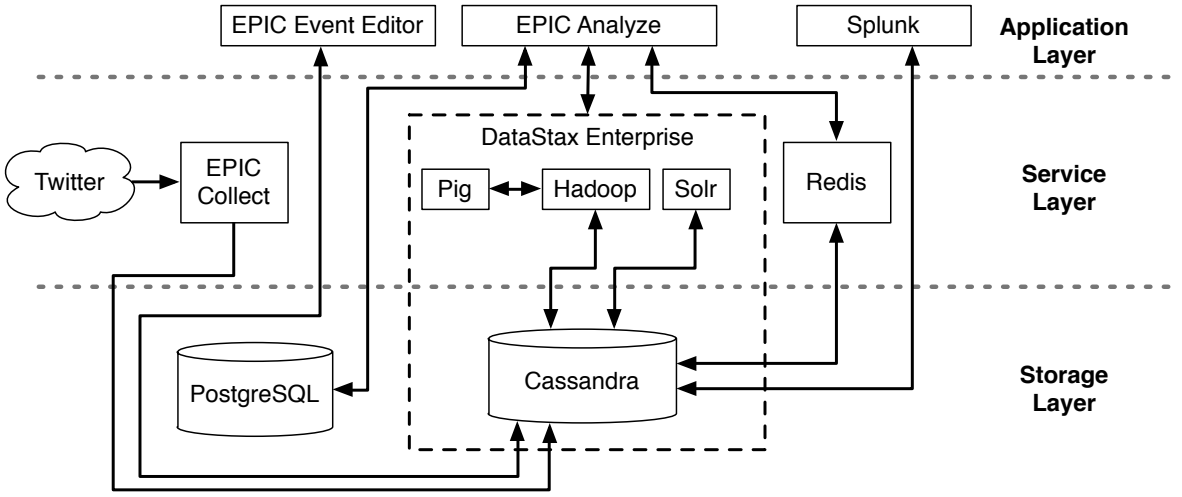

Figure 3. EPIC Analyze software architecture

capabilities natively support all such analysis tasks, as all of these functions are implemented to operate on tweet sequences. All that is required to integrate a new analysis operation that produces a tweet sequence is to ensure that the operation knows how to store the resulting sequence in Redis using the naming conventions we have put in place to associate tweet sequences with a dataset. The sequence then becomes available within EPIC Analyze and can be processed with all of EPIC Analyze's standard tools.

Of course, not all analysis operations produce sequences of tweets as output. For instance, a job might analyze a data set to identify the different languages used in the tweets stored in the data set. The output of that job would be a list of the languages found along with the number of tweets in each language and the percentage distribution of each language in the dataset. To accommodate these analysis operations, we require that their output be formatted in JSON. These objects are then stored in a column family in Cassandra that associates that particular object with the dataset and the job that produced it. For many of these operations, EPIC Analyze has built in support for displaying the results in a dashboard display associated with each dataset. For analysis operations that have not yet been integrated into the dashboard, EPIC Analyze provides a page that will display the raw JSON object produced by the analysis job. In this fashion, our analysis data model ensures the extensibility of EPIC Analyze. As new analysis operations are integrated into the environment, their developers can view the results as JSON objects. Once the analysis operation has been debugged, its developer can integrate the operation into the dataset dashboard, which will then display the results in a more accessible and intuitive fashion.

2.2. Architecture

The software architecture of EPIC Analyze (see Fig. 3) is organized into three layers, separating the roles of applications, services, and storage. The primary goal of this architecture is scalability and performance. With this architecture, we have designed EPIC Analyze to efficiently store, process, display, and visualize Twitter data sets that consist of hundreds of millions of tweets and consume terabytes of disk space. Project EPIC currently stores more than one hundred events in EPIC Collect; these events consume two terabytes of disk space. Each year, Project EPIC collects approximately fifty new events and those events, on average, will consume another terabyte of disk space. EPIC Analyze is scalable now, and we are confident that this architecture will continue to scale to meet Project EPIC's needs for the foreseeable future.

The remarkable aspect of this software architecture is its heterogeneity. Our experience indicates that architectural heterogeneity is inevitable when dealing with data-intensive systems, such as EPIC Analyze. For instance, in the storage layer of our architecture, we make use of Cassandra to store Twitter data because it met the requirements for EPIC Collect with respect to flexibility, availability, scalability, and reliability/robustness in a way that no other technology could match [18]. However, that does not mean that we abandon all of the work that has been invested in relational database technology. NoSQL has sometimes been defined as "No SQL" but its more commonly accepted definition is "Not Only SQL." As a result, we make use of a relational database system (currently PostgreSQL) to store information that EPIC Analyze

needs to track analysts, affiliations, and admin-specific information. We now discuss the other layers.

2.2.1. The Application Layer. As shown in Fig. 3, the application layer consists of three applications. The EPIC Event Editor is the user-facing application for EPIC Collect. It is a simple web application that edits the events/keywords submitted to the Twitter Streaming API by EPIC Collect. A thread in EPIC Collect monitors these events looking for changes every minute. If it detects a change (a new event or edits to an existing event), it disconnects from the Twitter Streaming API and reconnects submitting the new set of keywords for the active events. The EPIC Analyze application is a Ruby on Rails web application that will be discussed in more detail in Section 2.3.

Splunk is a third-party data analytics application designed for analyzing large amounts of time series data. Splunk is a general-purpose platform with a built-in query language that allows result sets to be displayed in a variety of formats. We integrated Splunk into EPIC Analyze via its Cassandra Connect software. Our goal with Splunk is to have it serve as a "crisis dashboard" for events that are actively being collected. Our analysts can start a collection on a new event and then switch to Splunk to see the tweets as they stream into EPIC Collect. Using Splunk's dashboard, they can get a sense for how people are talking about the event, what other keywords should be added to the collection, and whether any set of Twitter users stands out from the crowd and should be the focus of further study.

After collection on an event ends, the event will be imported into EPIC Analyze and our analysts will use its services to analyze the event in depth. This use of both general purpose and domain specific tools provides Project EPIC analysts with a range of options for analyzing the data sets stored in EPIC Collect.

2.2.2. The Service Layer. The service layer currently consists of EPIC Collect, the analysis components of DataStax Enterprise, and Redis. These components provide a set of service APIs that allow our applications to make requests for collecting, filtering, annotating, paginating, and retrieving data from the storage layer. Any technology that has the responsibility of offering new or enhanced services for applications is added to the service layer. EPIC Collect is responsible for all aspects of acquiring and storing data from Twitter. Its design is discussed in [5, 18].

DataStax Enterprise (DSE) is a collection of open source technologies that have been integrated to work with Cassandra. In particular, Apache Hadoop has been modified to read and write to Cassandra column families instead of HDFS. Apache Solr has also been modified to generate indexes based on information stored in Casssandra column families. Since EPIC Collect stores all of our Twitter data in Cassandra, DSE's integration of these technologies lets us build powerful analytical services within EPIC Analyze without having to migrate our data out of Cassandra.

To ease the task of writing MapReduce jobs in Hadoop, we make use of Apache Pig, a scripting language that auto-generates Hadoop MapReduce jobs without the need for directly configuring Java classpaths and jar files. Redis is a key-value in-memory database that organizes data into well-known data structures such as sets, lists, and dictionaries. As discussed above, EPIC Analyze uses Redis as a caching mechanism to achieve excellent performance when browsing and paginating large Twitter data sets.

2.3. Core Functionality

The core functionality of EPIC Analyze includes browsing, filtering, analyzing, and annotating Twitter data sets stored in EPIC Collect. The home page of EPIC Analyze displays all of the datasets available via a user's affiliation. Each dataset lists its name, description, status, keywords, number of tweets, and the date it was imported into EPIC Analyze. If the current user is also an admin, then a variety of CRUD-level operations are available, allowing datasets to be created, deleted, and edited. Otherwise, an analyst can "show" a dataset to start working with it.

2.3.1. Browsing and Pagination. EPIC Analyze provides an advanced browsing environment based on Shneiderman's visual information seeking mantra of "overview, zoom and filter, details on demand [19]." Fig. 4 displays three elements of this display: the dataset name, a list of tweets, and a detail view of a single tweet. The detail view is revealed when an analyst clicks on one of the tweets. The detail view allows an analyst to view the metadata associated with the tweet; it also contains links to the original tweet on Twitter and to the user who created it.

At the bottom of the page (not shown) is a pagination display that lists the number of pages contained in this dataset. As discussed in Section 2.1.2, the analyst can click on any page in the pagination display to immediately load that page. This form of rapid random access to any page in the dataset meets some of the requirements derived from previous interviews with Project EPIC analysts to understand their workflows and analysis needs [10].

2.3.2. Filtering. Filtering is an important feature for EPIC Analyze since human analysts engaging in qualitative research to elaborate quantitative analysis cannot manageably study Twitter data sets that are

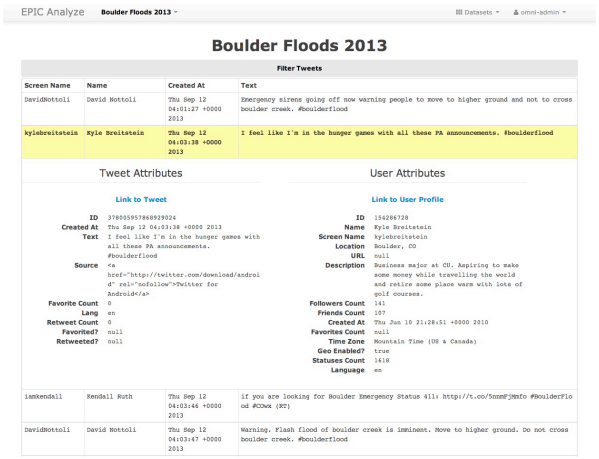

Figure 4. EPIC Analyze browsing interface

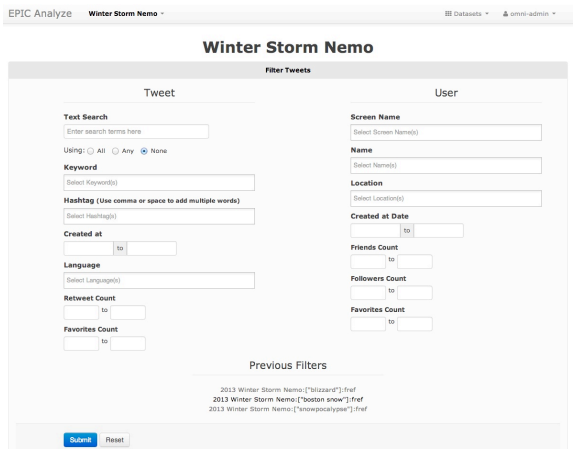

Figure 5. EPIC Analyze filter tweets form

larger than thousands of tweets. As mentioned in Section 1, before EPIC Analyze, Project EPIC developers would work with analysts to manually filter down data sets consisting of hundreds of millions of tweets five orders of magnitude to data sets consisting of thousands of tweets. With the filtering capabilities of EPIC Analyze, analysts can now perform this process themselves and save filtered results as "new datasets" that can be examined directly.

At the top of the browsing interface is a "Filter Tweets" button (see Fig. 4). When selected, a form is displayed that allows analysts to specify search queries over the current dataset (see Fig. 5). This form provides fields for specifying queries over the attributes of a tweet identified as the most useful by Project EPIC analysts [10]. These fields include the text of the tweet and its hashtags, creation date, language, retweet count, and favorite count. In addition, the form offers a keyword field in which a pre-populated menu of the dataset's keywords appears to allow keyword queries to be quickly specified.

A tweet's user object can also be searched via screen name, profile name, profile location, follower count, friends count, favorite count, and the date the user joined Twitter. As can be seen in Fig. 5, for those fields where it makes sense to specify ranges, the form contains controls for that (such as looking for tweets generated on a specific day or week). Currently, all of these fields combine to specify an AND query. That is, only those tweets that match *all* of the specified fields will be returned. We are working now to update this form to allow more complex queries to be specified that take advantage of all of the features that our underlying search engine provides (including OR queries, NOT queries, fuzzy search, etc.).

The filtering capabilities of EPIC Analyze are provided by DSE's integrated version of Apache Solr (as discussed in Section 2.2). Solr is a robust full-text search engine that handles a wide range of "document" types and provides advanced search features such as autocomplete for query terms, wildcard search, spell checking of queries, and stemming, as well as support for geospatial queries. Once Solr has indexed a large data set, it can retrieve results for queries within a tenth of a second. Indeed, we have performed benchmarks with Solr that demonstrate that it provides essentially O(1) lookup in response to search queries independent of the size of a column family. We tested this benchmark on column families containing 10K, 100K, and 1M tweets and search time was a tenth of a second for all queries. The only difference between these column families was the time it took to index them.

2.3.3. Job Framework. While browsing and filtering are essential features of EPIC Analyze, they provide only the most basic support for analysis. To meet our goal of providing an extensible and comprehensive analysis environment for crisis informatics research, EPIC Analyze must provide a mechanism for integrating a wide range of analytical tools. That mechanism is the EPIC Analyze job framework. This framework is supported by the analysis data model discussed in Section 2.1.2; that model dictates how the output of an analysis job is incorporated into the rest of the environment. If the output is a tweet sequence, then that sequence becomes available for browsing and filtering using the functionality described in the previous two sections. If the output is not a tweet sequence, then it is stored as a JSON object that can be displayed by EPIC Analyze's dataset dashboard.

An example of the former type of analysis job is sampling. Sampling involves creating a representative subset of a data set based on a specified constraint. Different approaches are utilized for creating samples such as simple random sampling, systematic sampling, cluster sampling, and so on. Moreover, rules for creating samples of a data set change based on the research domain and the type of data involved. The

desired result however is that the sample is representative of the larger data set in some way. In [10], McTaggart identified an algorithm that takes into account the long tail distribution of highly active Twitter users when producing a 10% sample of a large data set. We have implemented this sampling algorithm as a job in EPIC Analyze that produces a tweet sequence. This tweet sequence acts as "yet another dataset" within EPIC Analyze.

As an example of the latter type of analysis job, Project EPIC analysts ask questions concerning: the number of unique users who contributed to a data set; the most retweeted tweets; the most popular links; the most influential Twitter users; the percentage of tweets that were retweets; the number and percentage of geolocated tweets; the most common terms in the data set; the volume of tweets per day; and so on [10]. The EPIC Analyze job framework makes it simple and straightforward to integrate scripts that calculate these values. For instance, a job can wrap a Pig script that generates a MapReduce job that examines all of the tweets in a dataset for these values at once via DSE's Hadoop integration. The output of the final reduce phase is stored as JSON in the Cassandra column family for analysis results that are not tweet sequences.

Our job framework is not specific to running Pig scripts. It is a generic framework that can run code written in any language and invoke tools that sit outside the EPIC Analyze environment. In particular, we use Resque, a framework for queuing jobs that need to be processed in an asynchronous fashion. Resque queues are stored in Redis and consist of a sequence of Resque "workers." A worker is a Ruby class that inherits methods related to a job's life cycle and allow that job to be invoked, stopped, retried, etc. To aid with performance, EPIC Analyze requires that jobs be independent of one another and therefore makes use of a package called Foreman to ensure that all jobs in the EPIC Analyze queue are executed in parallel.

With this framework, Project EPIC is integrating a wide range of analysis tools that previously existed as stand-alone programs. Such tools include scripts to find all tweets that contain a link to an image or video, or the machine learning classifiers produced by Project EPIC's NLP researchers for finding tweets that contain situational awareness information [28]. Moving forward, all such scripts will be integrated with EPIC Analyze from the start and will quickly make their way into the hands of Project EPIC analysts, helping to make their work more effective and efficient.

3. Related Work

Our work on EPIC Analyze is consonant with research that attempts to build software environments that aid users in the performance of some task. Such research has a long history and includes anything from programming environments [25], design environments [30], writing environments [9], etc. In our own work, we have a history of designing and developing such environments [1, 4, 24] as well as performing basic research on integrating software infrastructure [2, 3].

As mentioned above, our EPIC Analyze work builds on the work we have invested in EPIC Collect [5, 6, 18] and the research Project EPIC performed to understand the needs of its own analysts [10]. In [10], McTaggart designed and implemented a web application for Twitter analytics and then conducted a usability study to elicit questions that social media analysts ask on Twitter data collected for disaster events. McTaggart mentions that the single largest issue with her prototype implementation is performance at scale. Preliminary numbers from her working prototype showed that queries performed against forty thousand tweets could take anywhere from 10 seconds to several minutes. This slow performance was problematic when computing queries on data sets consisting of 20 million tweets. It was these issues that sparked the work on EPIC Analyze.

In other work, socio-behavioral and NLP research on disasters call for robust and scalable software services that can provide tools for large-scale, collaborative social media analysis. To our knowledge, infrastructural work in this domain is limited. For a number of years, Purohit et al. have been maintaining Twitris [17], a citizen sensing platform for collecting and applying NLP techniques to millions of tweets to glean important information about a variety of events, from entertainment to disaster events. Unfortunately, the developers of Twitris do not reveal the technologies and systems they use to achieve their analysis at this level of scale. They do serve as an example of the types of analysis that we could apply to Project EPIC data sets in the future. EPIC Analyze, in comparison, is meant to be a more comprehensive data analysis environment for crisis informatics research supporting many aspects of the research life cycle and with support for queries that occur frequently when performing crisis informatics research.

Other systems have been designed for scalable Twitter analytics. Oussalah et al. describe a software architecture for collecting and analyzing geospatial and semantic information from Twitter data [13]. The tweets were consumed by a Twitter4j Java application and then transferred into a PostgreSQL database using the PostGIS spatial extension. They used Wordnet and Solr to relate tweets together if they share the same meaning. The data are then exposed by a Django web application, using GeoDjango for geospatial queries and the Haystack API for semantic queries. The goal of

this infrastructure was to search for tweets via semantic keywords as well as geo queries, and export the results via a Map or CSV file. EPIC Analyze faced similar challenges in designing for scalability, performance, and fault-tolerance as found in this work. However, the focus of Oussalah's work is on a particular domain-independent analysis technique and not with supporting the entire analysis life cycle for crisis informatics research. In general, this is how our work compares with other research that reports on the software infrastructure of data analysis environments, such as those found at large social media companies [26]. We make use of similar techniques but have focused our work on meeting the analysis needs of crisis informatics research in the context of large social media data sets and aim to provide a scalable, extensible, and comprehensive environment that can be used by crisis informatics teams for years to come.

4. Discussion

4.1. Design Tradeoffs and Challenges

The importance of making design decisions that ensure acceptable performance and scalability for EPIC Analyze is critical. However, those decisions impact factors such as disk and memory space, algorithmic complexity, and response time (latency). Crucially, deciding to preserve or optimize one may negatively affect the others, and thus it is important to understand the consequences of making these design decisions at the software architectural level.

4.1.1. Time vs. Space. As EPIC Collect matured, it became evident that data storage was a plentiful resource, and so it was natural to design solutions for retrieving tweets efficiently by investing in additional space. As discussed in Section 2.1.2, Cassandra's data model makes it difficult to quickly identify all of the row keys that exist in a column family. This difficulty led to our technique of generating tweet sequences and storing them in Redis where they can be efficiently manipulated. This is a classic example of a time vs. space tradeoff; it was clear that fast performance in response time was crucial and therefore the duplication of data (row keys and tweet ids stored in Cassandra AND Redis) was needed. It was therefore considered an acceptable cost to incur in our design.

4.1.2. Automation vs. Manual Control. Another design tradeoff that requires careful thought when designing an environment like EPIC Analyze concerns what services are automated and what "work" is left to the actions of our users. Should we automatically import a data set into EPIC Analyze when collection comes to a close on an event in EPIC Collect? Should EPIC Analyze automatically launch a suite of jobs to generate a set of standard metrics for a new data set? Saying, "yes" to these two questions saves on administration time but could "pollute" the analysis environment with data sets that the Project EPIC analysts are not yet ready to analyze. While it is technically possible to implement these services, it may cause needless stress in the system's surrounding socio-technical environment and, thus, best to make these steps wait until specifically invoked by a human user. For now, we have decided to err on the side of manual control, allowing actual use to dictate what services should be automated down the line.

4.2. EPIC Analyze as a Socio-Technical System

Systems that construct relationships between users and technological features of their workplace environment are referred to as socio-technical systems [8]. The reason for studying socio-technical systems is that it allows for a broader understanding of how users are oriented towards using technology to perform cooperative work in workplace environments. The relationship between the system and its environment is reciprocal; the environment dictates the tasks that need to be performed within the system but the system itself then changes the environment, causing a feedback loop that shapes both sides of the equation. Insights that highlight user motivations, organizational values, and the actions of users within a system are valuable in helping to design that system for "situated use" within its environment [11]. On the other hand, systems that fail to understand the behaviors of its users and its surrounding environment risk abandonment, non-adoption, and/or user churn.

Socio-technical theory is rooted in the pre-software working environments of the English coalmines in Haighmoor examined by social scientists and anthropologists at the end of the Second World War [27]. Since then, research communities like that of Computer-Supported Cooperative Work have conducted qualitative studies in workplace environments involving the use of computers as a way to understand human-computer relationships [12, 23]. In crisis informatics, it has been observed that socio-technical communities of volunteers on social media platforms help digitally route and verify information coming from the ground during disaster events [22]. Other communities orient themselves around open-source mapping platforms like OpenStreetMap [20]. These humanitarian-based workforces situate their work around the syntaxes and structures of the information being produced on these online platforms.

Here we argue that the direct intervention of a system like EPIC Analyze can positively influence the workflows of the analysts who perform social media analysis as a mode of daily work. Preliminary findings from a usability study conducted on the analysts in our research work environment [10] indicate that the tools used during analysis vary depending on the research questions being asked. Some analysts preferred using Excel, Google Docs, spreadsheets, or other specialized tools to browse and annotate a data set. Some used visualization software like Tableau; some even preferred programmatic access to the data sets. Based on this feedback, EPIC Analyze addresses the shortcomings that analysts experienced with using a heterogeneous set of tools.[1]

Our interviews with these analysts also showed that the current state of EPIC Analyze has already proven to be useful. During the interview, one analyst commented that her need to retain a subset of all tweets within a data set based on specified keywords has been fulfilled by EPIC Analyze. Another analyst commented that, with a few more UI-based modifications to the browser to glean more visual-based information (e.g. Twitter profile pictures), EPIC Analyze could replace other tools in her workspace. Their active participation in the iterative development process helps ensure that EPIC Analyze will continues to meets their needs.

5. Future Work

EPIC Analyze provides its users with a range of powerful services that support crisis informatics research. An instance of EPIC Analyze is being used by Project EPIC analysts on a daily basis on smaller data sets that allow the system to be used for active research while allowing the development team to respond to identified problems or requests for enhancement. Our future work on EPIC Analyze will focus on improving its ability to sort Twitter data sets on a combination of tweet/user attributes and making use of fast, scalable sorting methods known as non-recursive, most-significant digit radix sorts [7]. We intend to add the ability to annotate Twitter data sets, allowing tweets to be coded both manually and as the result of an EPIC Analyze job. We also intend to add support for "comments", allowing analysts to engage in conversations at the tweet level. Finally, support for visualization is an important feature that will be added to future versions of EPIC Analyze.

[1] The identified shortcomings included having data spread out across multiple tools hindering a comprehensive analysis, having different analysis capabilities in each tool, and not being able to easily share data and annotations with other analysts.

6. Conclusions

Designing and developing software infrastructure like EPIC Analyze while meeting complex functional and non-functional requirements is a challenging endeavor. Many software design and software architecture issues arise including data modeling challenges, architectural heterogeneity, and software integration problems alongside the more typical concerns of implementing core functionality. In this paper, we presented our solution to these challenges with unique contributions including 1) the identification of the open source frameworks and tools needed to achieve scalability and performance goals; and 2) our use of tweet sequences to achieve fast, random-access traversal of large data sets and mechanisms for incorporating new analysis techniques.

Also unique is the use of user-centered design techniques to drive our software architectural decisions such that EPIC Analyze supports the entire analysis life cycle of crisis informatics research. We worked closely with Project EPIC analysts to iterative design the environment, and its capabilities, ensuring that each feature met real-world needs. This approach, along with releasing prototypes to the analysts early and often, are helping to ensure that EPIC Analyze will be incorporated into the daily work of the analysts and support Project EPIC research for the foreseeable future. We see this focus on analysts as important as it will inform how this work can be expanded to include other audiences of social media data analysis—researchers, practitioners, and members of the public. Our goal is to one day deliver the services of EPIC Analyze to these audiences to support the natural inclination that people have to search for useful information in mass emergency events. This work thus supports our call for research that supports the "everyday analyst [16]."

Acknowledgments

This material is based upon work sponsored by the NSF under Grant IIS-0910586.

References

[1] Anderson, K., Taylor, R., Whitehead, E. Chimera: Hypermedia for Heterogeneous Software Development Environments In *ACM Trans. on Information Systems*, 18(3):211–245, July 2000.

[2] Anderson, K., Och, C., King, R., and Osborne, R. Integrating Infrastructure: Enabling Large-Scale Client Integration. In Proc. *ACM Conf. on Hypertext*, pp. 57–66, May 2000.

[3] Anderson, K., Sherba, S., Lepthien, W. Towards Large-Scale Information Integration. In Proc. *International Conference on Software Engineering*, pp. 524–534, May 2002.

[4] Anderson, K., Bradley, E., Zreda, M., Rassbach, L., Zweck, C., and Sheehan, E. ACE: Age Calculation Engine—A Design Environment for Cosmogenic Dating Techniques. In Proc. *Int. Conf. on Advanced Engineering Computing and Applications in Sciences*, pp. 39–48, Nov. 2007.

[5] Anderson, K. and Schram, A. Design and Implementation of a Data Analytics Infrastructure in Support of Crisis Informatics Research (NIER track). In Proc. *33rd Int. Conf. on Software Engineering*, pp. 844–847, May 2011.

[6] Anderson, K., Schram, A., Alzabarah, A., and Palen, L. Architectural Implications of Social Media Analytics in Support of Crisis Informatics Research. In *Bulletin of the Tech. Comm. on Data Engineering*, 36(3): 13–20, Sep. 2013.

[7] Arif Aydin, A., and Alaghband, G. Sequential and Parallel Hybrid Approach for Non-Recursive Most Significant Digit Radix Sort. In Proc. *Applied Computing*, pp 51–58, Oct. 2013.

[8] Baxter, G. Socio-technical Systems. In *LSCITS Socio-Technical Systems Engineering Handbook*. http://archive.cs.st-andrews.ac.uk/STSE-Handbook/. 2011.

[9] Bernstein, M. Storyspace 1. In Proc. *ACM Conf. on Hypertext and Hypermedia*, pp. 172–181, June 2002.

[10] McTaggart, C. Analysis and Implementation of Software Tools to Support Research in Crisis Informatics. MS Thesis. University of Colorado. 65 pages. May 2012.

[11] Mumford, E. Designing Human Systems for New Technology - The ETHICS Method. 1983.

[12] Orlikowski, W. Learning from Notes: Organizational Issues in Groupware Implementation. In *Computer-Supported Cooperative Work*, pp. 362-369. November 1992.

[13] Oussalah, M., Bhat, F., Challis, K., and Schnier, T. A Software Architecture for Twitter Collection, Search, and Geolocation Services. In *Knowledge-Based Systems*, 37: 105–120, January 2013.

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

[15] Palen, L., Martin, J., Anderson, K., and Sicker, D. Widescale Computer-Mediated Communication in Crisis Response: Roles, Trust & Accuracy in the Social Distribution of Information. http://www.nsf.gov/awardsearch/-showAward.do?AwardNumber=0910586, 2009.

[16] Palen, L., Vieweg, S., and Anderson, K. Supporting 'Everyday Analysts' in Safety- and Time-Critical Situations. In *The Information Society*, 27(1):52–62, January 2011.

[17] Purohit, H. and Sheth, A. Twitris v3: From Citizen Sensing to Analysis, Coordination, and Action. In Proc. *AAAI Conf. on Weblogs and Social Media*, pp. 746–747, July 2013.

[18] Schram, A. and Anderson, K. MySQL to NoSQL: Data Modeling Challenges in Supporting Scalability. In Proc. *ACM Conf. on Systems, Programming, Languages and Apps.: Software for Humanity*, pp. 191–202, Oct. 2012.

[19] Shneiderman, B. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In Proc. *IEEE Symp. on Visual Languages*, pp. 336–343, Sep. 1996.

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

[21] Starbird, K., Palen, L., Hughes, A. and Vieweg, S. Chatter on The Red: What Hazards Threat Reveals about the Social Life of Microblogged Information. In Proc. *ACM Conf. on Computer Supported Cooperative Work*, pp. 241–250, Feb. 2010.

[22] Starbird, K. and Palen, L. "Voluntweeters:" Self-Organizing by Digital Volunteers in Times of Crisis. In Proc. *ACM Conf. on Human Factors in Computing Systems*, pp 1071–1080, May 2011.

[23] Suchman, L. Office Procedure as Practical Action: Models of Work and System Design. *ACM Transactions on Information Systems*, Vol. 1, pp. 320-328. October 1983.

[24] Taylor, R., Nies, K., Bolcer, G., MacFarlane, C., Anderon, K., and Johnson, G. Chiron-1: A Software Architecture for User Interface Development, Maintenance, and Run-Time Support. In *ACM Trans. on Computer-Human Interaction*, 2(2):105–144, March, 1995.

[25] Teitelman, W. and Masinter, L. The Interlisp Prog. Environment. In *Computer*, 14(4):25–33, 1981.

[26] Thusoo, A., Shao, Z., Anthony, S., Borthakur, D., Jain, N., Sarma, J., Murthy, R., and Liu, H. Data Warehousing and Analytics Infrastructure at Facebook. In Proc. *ACM SIGMOD International Conference on Management of Data*, pp. 1013– 1020, June 2010.

[27] Trist, E. The Evolution of Socio-technical Systems. *Occasional Paper*, No. 2. June 1981.

[28] Verma, S., Corvey, W., Vieweg, S., Martin, J., Palen, L., Palmer, M., Schram, A., and Anderson, K. NLP to the Rescue?: Extracting "Situational Awareness" Tweets During Mass Emergency. In Proc. *Fifth Int. AAAI Conf. on Weblogs and Social Media*, pp. 385-392, Jul. 2011.

[29] White, J., Palen, L. and Anderson, K. Digital Mobilization in Disaster Response: The Work & Self-Organization of On-Line Pet Advocates in Response to Hurricane Sandy. In Proc. *ACM Conf. on Computer Supported Cooperative Work*, pp. 866–876, Feb. 2014.

[30] Winograd, T. From Programming Environments to Environments for Designing. In *CACM*, 38(6): 65–74, 1995.