Getting the Query Right: User Interface Design of Analysis Platforms for Crisis Research

Mario Barrenechea, Kenneth M. Anderson(✉), Ahmet Arif Aydin, Mazin Hakeem, and Sahar Jambi

Department of Computer Science, University of Colorado Boulder, Boulder, CO, USA
{mario.barrenechea,ken.anderson,ahmet.aydin,mazin.hakeem, sahar.jambi}@colorado.edu

Abstract. Web-based data analysis environments are powerful platforms for exploring large data sets. To ensure that these environments meet the needs of analysts, a human-centered perspective is needed. Interfaces to these platforms should provide flexible search, support user-generated content, and enable collaboration. We report on our efforts to design and develop a web interface for a custom analytics platform—EPIC Analyze—which provides interactive search over large Twitter data sets collected during crisis events. We performed seven think-aloud sessions with researchers who regularly analyze crisis data sets and compiled their feedback. They identified a need for a "big picture" view of an event, flexible exporting capabilities, and user-defined coding schemes. Adding these features allowed EPIC Analyze to meet the needs of these analysts and enable exploratory research on crisis data.

Keywords: User interfaces · Data-intensive systems · Crisis informatics

1 Introduction

We live in an era of big data. Our ability to generate and collect large amounts of data is having a transformative effect on the types of analysis we can perform. The term "big data" refers to a variety of techniques and technologies that enable this transformation and enable the creation of data-intensive software systems. These systems must collect, store, index, analyze, and annotate large sets of data and there are significant challenges in making these systems scalable, reliable, and efficient. Another class of challenges exist with respect to designing the user interface of these systems. These interfaces must provide users with a sense of scale, present details on demand, provide overviews, and provide a flexible set of operations that execute at interactive speeds.

We work in an area known as crisis informatics [15]; crisis informatics is a multidisciplinary research area that examines the socio-technical relationships among people, information, and technology during crisis events. It mainly examines the qualitative and quantitative aspects of social media data produced by members of the public during times of mass emergency. Our project—Project

P. Cimiano et al. (Eds.): ICWE 2015, LNCS 9114, pp. 547–564, 2015.
DOI: 10.1007/978-3-319-19890-3_35

EPIC—has been collecting crisis data sets from Twitter since Fall 2009; we have now amassed approximately 2.5B tweets across hundreds of events [2,3,18]. As a result, we have been designing and developing a data analysis environment—EPIC Analyze [1]—that provides a variety of services to help Project EPIC analysts explore and understand our large Twitter data sets.

As a result, we have been wrestling with a number of thorny design issues related to the design of data-intensive systems and their user interfaces [7]. In this paper, we report on the challenges we have encountered with designing user interfaces for services that enable the browsing, filtering, and annotation of large crisis data sets. For these services, our goals have been to a) provide interactive response times, to b) make it easy to query, filter, and explore a large data set, to c) ease the management and filtering of user-defined data, and to d) provide collaboration capabilities for our users. Drawing on techniques from human-centered computing, software engineering, and web engineering, our goal is to simplify the access to large crisis data sets and provide capabilities that allow EPIC Analyze to function as a vehicle for exploratory research on crisis data. To evaluate our efforts, we performed think-aloud sessions with seven researchers who regularly analyze crisis data sets and compiled their feedback. Their feedback drove the creation of the most recent version of EPIC Analyze.

This paper is organized as follows. In Section 2, we situate our work with respect to related research. In Section 3, we present the user interface and services of the current version of EPIC Analyze and then, in Section 4, we describe the evaluation we performed on a prior version of EPIC Analyze that led to the feedback that influenced the creation of the current version. In Section 5, we describe the data models and services in EPIC Analyze that make the user interface presented in Section 3 possible. Finally, we present avenues for future work and our conclusions in Section 6.

2 Related Work

We now present work related to our research on user interface design of data-intensive systems for crisis informatics. We start with a discussion on the challenges associated with interfaces for big data systems. We then present work in crisis informatics that provides insight into the needs of Project EPIC analysts. Finally, we discuss the importance of software architecture and design in producing data-intensive systems that are scalable, reliable, and efficient.

2.1 Interface Design for Big Data Systems

Software and web engineering researchers and practitioners face challenges with capturing, processing, integrating, analyzing, and archiving big data. To add to that burden, the goal for human-centered computing research in this domain is to put the power of big data systems into the hands of non-technical users [8]. Creating intuitive, flexible, and extensible user interfaces that allow users to pull from structured and unstructured data sources, query and analyze the

data, and make more informed claims about the data, are the objectives of big data interfaces. Users of such interfaces do not need to become familiar with big data frameworks—e.g. MongoDB, Redis, Cassandra, Spark, etc.—but need to have confidence that the systems built on top of them are reliable and efficient; otherwise they may choose to stay away from working with big data or seek to use other technologies that do not have the same capacity for scale and thus be forced out of taking advantage of the benefits that big data analysis can provide.

Systems like Wrangler [11] and Google Refine (http://openrefine.org/) decrease the amount of work required to transform data; this allows those not proficient in programming to work in this space, converting large data sets into the format they need. For large time series data sets, there are several advances in diverse domains. The LifeFlow system [24] aggregates event data from hospital visits and room transitions and visualizes them to identify problems in triaging or resource allocation. Splunk (http://www.splunk.com) is a commercial data analysis platform for working with time series data, providing a pipe-based textual language to manipulate data. There are shortcomings to these tools, however. Wrangler, Refine, and LifeFlow do not provide support for user-generated annotations and do not enable collaboration among multiple users. Splunk's programming language is flexible but has a significant learning curve, especially for analysts not familiar with the pipe-and-filter architectural style.

One must also appreciate the infrastructure that is built behind such interfaces; often the design of the software infrastructure itself shapes the look and feel of the interface (as we will discuss in Section 5). With data analysis platforms, it is not good enough to just display a web page with filters for querying a database. Oussalah et. al [14] presents a web-based analysis environment that ties in semantic and spatial analyses of tweets in addition to straightforward search capabilities. This work inspired aspects of the design of EPIC Analyze, especially with respect to providing a suite of integrated services to the end user.

2.2 Crisis Informatics

Crisis informatics is an emerging field of study that examines the socio-technical relationships among people, information, and technology during mass emergency [15]. During disaster, lots of data is generated on social media. Crisis informatics has thus quantitatively and qualitatively examined social media during mass emergency events to understand socio-behavioral phenomena of self-organization [21,23], policy change [9], information sharing between unofficial and official sources [17,22], and crowdwork [16,23]. While much has been written on these topics, research methods for collecting, storing, and making sense of these vast amounts of social media data is unwieldy, time-consuming, and expensive. As software and web engineering researchers in crisis informatics, we seek to study how to design systems that support the methods employed by research analysts in crisis informatics and how such systems may change those methods over time. We view EPIC Analyze as a system that supports both of these goals.

2.3 Software Architecture

With respect to software architecture and the design of data-intensive systems, it is important to identify useful software architectural patterns as well as the right combinations of middleware and persistence software to efficiently, scalably, and reliably support social media data collection and analysis [1–3,6,14]. It is important that these systems be reliable to ensure 24/7 operation; it is almost impossible to go "back in time" to collect Twitter data after an event has occurred. As such, these systems need to be running continuously to be ready to spring into action when an event of interest occurs.

3 EPIC Analyze

EPIC Analyze [1] is a data analysis platform that builds on top of our previous work on EPIC Collect [2,18], a system designed for reliable and scalable social media collection. EPIC Analyze extends EPIC Collect with an architecture designed to support social media analytics (see Fig. 1). These systems support an analysis workflow that starts when an event of interest has been detected. Project EPIC analysts monitor Twitter for keywords of interest and use the EPIC Event Editor (a simple web application) to associate those keywords with a new event. EPIC Collect detects the presence of this new event and submits its keywords (along with the keywords of all other active events) to Twitter's Streaming API. It collects tweets containing those keywords and stores them in Cassandra. Our four-node Cassandra cluster can store terabytes of information and serves as the foundation for the work performed by EPIC Analyze.

3.1 The EPIC Architecture

The architecture for EPIC Analyze shown in Fig. 1 builds on top of EPIC Collect's storage mechanism (Cassandra) via the use of Datastax Enterprise

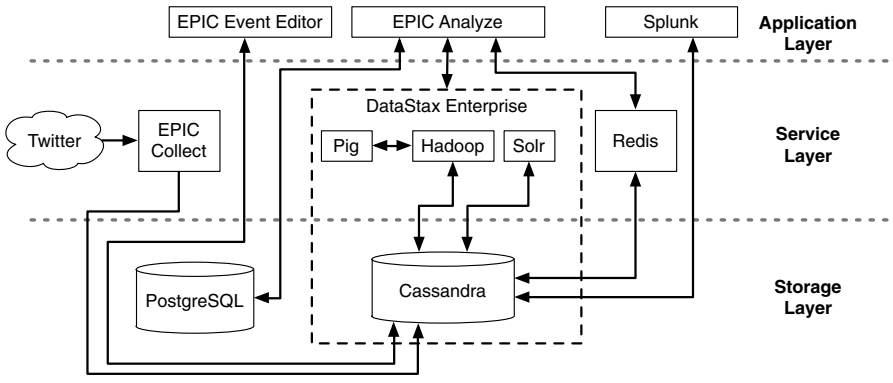

Fig. 1. EPIC Analyze Software Architecture

(http://www.datastax.com/) and its integrated versions of Solr, Pig, and Hadoop. Each of these components can be used to help index, search, or process our large Twitter data sets. We make use of PostgreSQL to store comments and annotations made by analysts while working with EPIC Analyze. EPIC Analyze is itself implemented as a Ruby on Rails web application that knows how to access all of the infrastructure provided by EPIC Collect and Datastax Enterprise. In addition, it makes use of Redis to cache the results of frequently accessed queries and data. Finally, a third-party data analysis tool—Splunk—is used as an alternative method for viewing and analyzing Twitter data, especially as it is streaming in during an active data collection and before it has been indexed by Solr and ready for use within EPIC Analyze.

3.2 The EPIC Analyze Application

EPIC Analyze is a web application that provides scalable and efficient filtering, analysis, and annotation capabilities on large Twitter data sets.

Browsing, Searching, and Visualizing. When an analyst logs in, she sees a list of data sets that have been indexed by EPIC Analyze (e.g. "2013 Boulder Flash Floods"). Once a data set has been selected, an analyst can view the tweets page-by-page in the EPIC Analyze browser (see Fig. 2.c). The browser provides an overview of the data set via a timeline that shows the volume of tweets over time at the top of the browser. On the right hand side, a detailed view of a single "page" of fifty tweets is displayed. On the left hand side, a form for querying the data set and its annotations is presented. If an analyst clicks a tweet, all of its relevant metadata is displayed in an in-line form for easy viewing; this form also contains links that take the analyst to see the original tweet on Twitter. On the timeline, analysts can click and drag (see Fig. 2.a) to specify a start and end date that will be used for all subsequent queries.

The filter form on the left allows analysts to search the data set by tweet or by annotation. The tab for tweet-based search presents a list of tweet attributes that can be used to filter an entire data set. The form supports standard boolean operations for advanced search; any number of tweet attributes can be used to specify a query. Backend services such as Solr and PostgreSQL are used to implement these queries and to provide facets. For instance, an analyst can click the Keyword filter field to see a dropdown list of all keywords associated with that data set (along with the number of times each keyword appears in the data set). This provides analysts with an idea of how popular (or unpopular) a certain keyword was and may guide or refine the questions they ask of the data set.

The tab for annotation search offers the analyst the ability to make queries against user-generated content. Annotations include both labels and comments and are visible to all analysts working on a data set to foster collaboration during the analysis process. These annotations can be queried using the same logical operations and faceting capabilities described above for tweet-based search. Submitted queries are processed quickly, often within a few seconds; we credit this performance to the design of EPIC Analyze's software infrastructure.

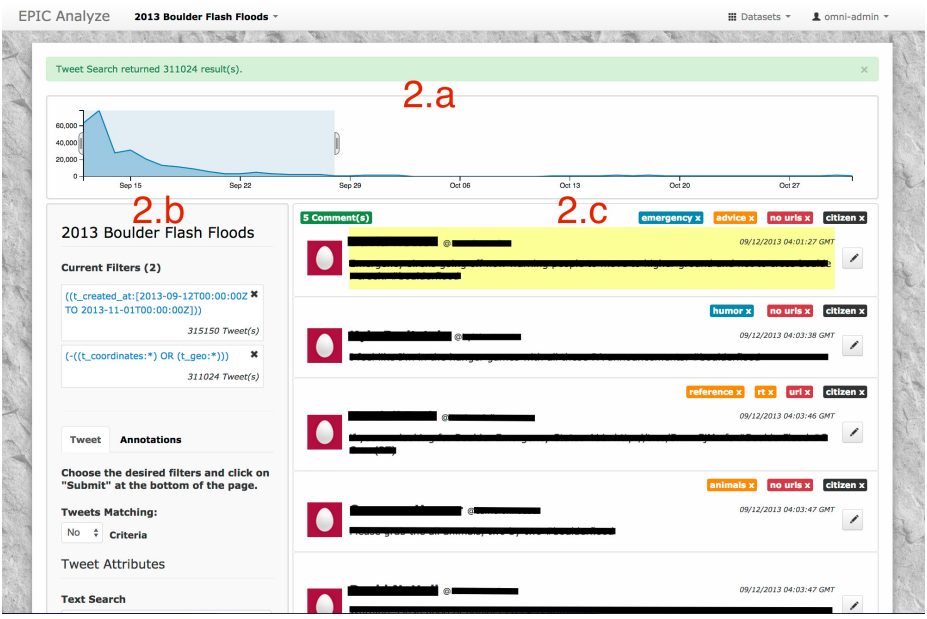

Fig. 2. The EPIC Analyze Browser. (Tweets are hidden for privacy reasons.)

After a query has been submitted and the result set has been filtered to match, the query appears in the "Current Filters" section of the user interface (see Fig. 2.b). This list provides a summary of the queries that were performed in the past and that are in effect as the analyst drills down further into a data set. Analysts can choose to delete a filter in this list or they can *jump to* a filter by clicking on its name. Such a click will render the result set for the filters up to that point. This first-class interface for queries provides a more tangible experience when analyzing data sets, and places the analyst in control over the analysis process.

Annotating Tweets. If an analyst wants to annotate a tweet, she can click on the pencil icon that appears on the right side of each tweet; this action causes an annotation form to appear next to the tweet. The analyst has the option to annotate the entire tweet or to annotate just portions of the text of the tweet. In this latter case, the browser updates the annotated text to appear in a color associated with the label making the annotation readily identifiable in future analysis sessions. Analysts can also comment on the tweet; multiple comments appear in a conversation thread (one per tweet) that appears in the annotation form. The browser indicates that comments exist for a particular tweet by displaying the number of comments for a tweet at the top of its display.

The annotation of tweets is a recognized and critical activity for crisis informatics research. Methods for doing so are documented in empirical studies

[9,17,22,23], but before our work with EPIC Analyze the process of annotating tweets in crisis data sets was laborious and error prone, tolerated only because it was the only way to conduct the research at the time. EPIC Analyze's support for annotations has been welcomed by Project EPIC analysts; they especially like its ability to allow search and filtering over the annotations.

However, future feature enhancements have been identified and include automating the tagging of tweets with labels—especially for large data sets—and allowing analysts to create and/or load their own labels (i.e. coding schemes) for a data set. In particular, the label becomes more than just a textual annotation but is instead a mapping scheme between values found in a tweet and a particular label. For example, analysts may want to tag all existing and future tweets that come from the user name "News6" as "local media." We intend to add this feature to EPIC Analyze in the near future.

Other Features. Research analysts can perform other useful functions with EPIC Analyze. For example, once they have applied a set of filters against a data set, and the resulting set of tweets is useful for future analysis, they can save the state of the result set as a new data set. With this feature, analysts can now engage in more localized analysis with other analysts without having to search a large data set from scratch each time. Secondly, as previously documented in [13], the ability to export these data sets in well-known formats is critical. For this reason, EPIC Analyze allows analysts to export result sets to CSV. Additionally, all annotation labels and comments that have been used to tag the tweets that appear in the result set also appear in the CSV. Finally, with the help of Redis and a job framework known as Resque, EPIC Analyze provides analysts and administrators with the ability to write Hadoop jobs or other scripts that process each tweet in the data set. This job framework is already used by EPIC Analyze to automatically sort large Twitter data sets by multiple sort dimensions, such as tweet id, screen name, and retweet count [4].

4 Evaluation of EPIC Analyze's User Interface

EPIC Analyze has been in constant design and development since Fall 2013. In that time, we have had multiple opportunities to respond to user feedback to make small improvements to the system here and there. However, we became interested in whether EPIC Analyze supports the analytical workflows that Project EPIC analysts employ to perform their research. We turned to task-oriented usability methods to evaluate whether these workflows can be performed with the EPIC Analyze user interface and whether it provides an enjoyable user experience [5]. In this section, we describe a user interface evaluation that we performed via think-aloud sessions on a previous version of the system interface (the one prior to the interface described in Section 3). Based on the results of these sessions, we developed the interface that EPIC Analyze now provides.

4.1 The Research Analysts

We interviewed seven crisis informatics researchers who self-identify as information scientists with specializations that draw from other disciplines, such as geography, journalism, and computational social science. These analysts all work for Project EPIC and work on events ranging from the 2014 Carlton Complex Wildfires in Washington (40,000 tweets), the 2013 Boulder Flash Floods (1 million tweets), the 2013 Japan Earthquake event (1.9 million tweets), and the 2012 Hurricane Sandy event (22 million tweets).

Previous work examined the work processes and tools used by research analysts in this domain [13]. In our study, we observed that they access data in diverse ways, from using commercial analytical tools such as Tableau and Splunk to more programmable interfaces like R and Python. Furthermore, their work processes are different based on their research: one analyst reports that she uses social network analysis methods in Python to model relationships of Twitterers within a data set, but she is often concerned about the representativeness of the data set. To address this concern, this particular analyst wrote her own scripts to filter the data sets such that she has a representative sample that can help answer her research questions.

Some analysts have stronger programming skills and are comfortable with programmatically retrieving data from EPIC Collect, storing it in their own databases, and scaffolding web pages of their analysis on top of that data. Still others use basic—yet powerful—tools, such as spreadsheets. Related work [10] shows that there is considerable overlap in functional power and usability with these tools since they are so well-known and understood; this poses challenges for adoption of other tools that can support these workflows, sometimes with greater analytical power. However, these productivity tools do fall short in some use cases, such as annotating tweets with labels and comments, which is necessary for qualitative and quantitative analysis. To understand how EPIC Analyze, therefore, may become useful to these analysts, we performed think-aloud sessions to see if EPIC Analyze could solve some of their current struggles and support their current and future analysis workflows.

4.2 Think Aloud Protocol Runs

We performed a series of think-aloud protocol runs to gather feedback on the progress we had made on the system interface, which included full search of most tweet-based metadata, support for creating and searching annotations on tweets, and creating new data sets out of previously-run filters (see Fig. 3). Think-aloud runs are well-known as a traditional usability method that captures feedback on a task that the user performs without any help or guidance from the interviewer [12]. We prepared several tasks that analysts should be able to perform without any prior experience with using EPIC Analyze. These tasks reflect the bulk of functionality that is available to use for any given data set in EPIC Analyze. In all seven interviews, we asked the research analyst to perform the following tasks or to answer the following questions:

1. Open an event that interests you
2. Investigate a tweet and look at all of its attributes.
3. How can you view the tweets from only the first or last day of this data set?
4. How can you get tweets written in Spanish? How about Spanish and French?
5. Can you find tweets that either have the word "bomb" in the text or are from a specific user? What about both?
6. How can you get back tweets with annotations?
7. How can you comment on a tweet that has already been commented on?

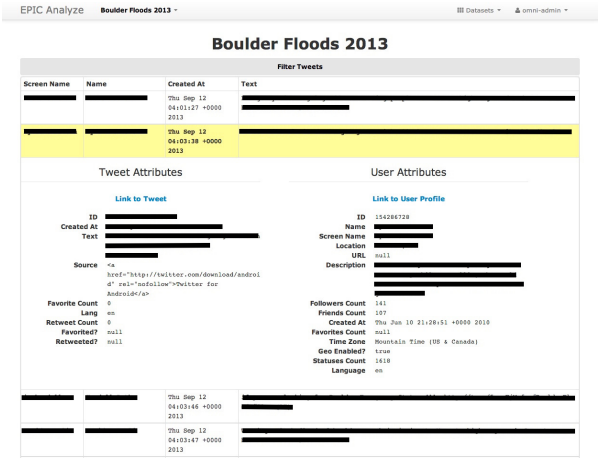

Fig. 3. Previous Version of EPIC Analyze

Each research analyst completed the tasks without any difficulty. All of them enjoyed using the interface and the general look-and-feel of the application. About half of them found it surprising that certain filters—such as issuing a query for both Spanish and French tweets—was supported. Almost all analysts complained that there was no "reset" button at the bottom of the query interface to clear text and other selections from the form. Some analysts complained that the filter result notification bar—which displays the results of the filtered result set when the query has finished—should stay on the screen rather than flashing away. This would create a reference point for the analyst in understanding how effective their query was. Also, they described the need for summative statistics, such as total dataset size, dataset date range, and keyword search terms (a feature already present but not very visible). One analyst suggested that all currently applied filters be displayed at the top of the search interface. As shown in Section 3, all of this feedback has been addressed.

After the tasks were performed, each interviewee was asked free-form questions about their experience with the interface. These semi-structured interviews at the end of the think-aloud session were critical to obtain a unique perspective on how and why these analysts would or would not use EPIC Analyze for future analysis. We now describe additional issues raised by the analysts.

4.3 The Big Picture

Analysts described the desire to see a "big picture view" of the event. The big picture refers to the extent to which analysts understand the context of their analysis through space, time, and control. They admitted that viewing thousands of pages was daunting and that they would prefer digging into the data after queries returned a more manageable set of data, on the order of hundreds of tweets. In order to achieve that, they suggested, it would be beneficial to have a way of displaying volume of tweets over time. That way, they could identify temporal areas that contained surges or "spikes" that might be interesting to investigate, rather than having to solely rely on their queries.

After the think-aloud runs, development on this feature started right away. As shown in Fig. 2, the current EPIC Analyze browser shows a timeline of tweet activity at the top of its display. This timeline is not just a static depiction of the volume of tweets for any given data set; it dynamically conveys volume of tweets for any query that is processed, following recommendations made by Schneiderman and his Visual Information-Seeking Mantra: "Overview first, zoom and filters, then details-on-demand" [19].

4.4 Filtering Data Out

The research analysts generally praised the ability to filter data based on conventional boolean operators such as AND and NOT, but they were also interested in *filtering data out* rather than *qualifying data in*. To do this, the unary operand NOT must be used to set up a query so that results that do not satisfy the filter values are returned. Analysts claim that this is useful because of the nature of their investigations into the data. Sometimes the most meaningful (and yet sparse) data comes from official sources, as in [9], and sometimes from "unofficial sources" who are tweeting about important local information, as in [22].

Of course, implementing this extra operand poses challenges for the interface. It is easy to imagine placing checkboxes at the beginning of each filter input option on the filter form, representing whether or not the analyst would like to filter this input in or out. But that leads to a lot more choices that, if improperly designed, may pose confusion and additional cognitive load on the analyst. In our most recent version of the EPIC Analyze interface, we have placed a drop-down list at the top of the filter form that includes three choices for filtering operators: AND, OR, and NOT. Analysts can choose exactly one option that will be applied to all filters in the interface at that time. Choosing the NOT operand will essentially group the filters with a locally implicit OR condition, and all filters are then prepended with a NOT clause in front of them. The reasoning behind this design decision is to abide by the observations we made during the think-aloud sessions and semi-structured interviews: analysts want to drill-down into data sets, not include as much data as possible. Specifically, when they are looking at a multi-million tweet data set, filtering out is a primary and important action that allows them to dig deeper into the data set for specific signals. Enabling the analyst with the NOT clause coupled with the OR condition over

the AND condition achieves this: they are able to select more specifically what data will be filtered out. In future user studies, we plan to evaluate whether this mode of filtering out is preferable over using exclusive joins (AND clauses) between the filters.

The user interface task is not yet done after implementing the NOT operand, however. What if the analyst wants to make compound queries using these operators, such as "(X OR Y) AND Z"? How can the interface and the system support this? As we discuss in Section 5, we turned to data-modeling techniques to solve this problem.

4.5 VOST Deployment and Work Process Integration

Finally, there was interest expressed by one research analyst about integrating EPIC Analyze into established virtual community organizational settings. Specifically, this analyst performs research in collaboration with several virtual operations support teams, or VOSTs, in the emergency management community. Since 2011, VOSTs have been monitoring social media streams during disaster events to glean situational and actionable information coming from both official and unofficial sources [20]. This information, they argue, comes at a faster rate, is more verifiable, and discovers events in a time frame that traditional models of emergency management do not have the resources to handle. These VOSTs are now increasingly using social media as communication channels with the public to maintain situational awareness of the disaster as it is unfolding.

In discussing how EPIC Analyze can provide an information workspace for VOSTs, we discovered that the functionality previously described for automatically annotating tweets with labels is essential. As such, we have not yet deployed EPIC Analyze in this way; however this discovery provided further validation that such a feature is needed. We will return to studying the ability of EPIC Analyze to support these teams once this feature is implemented.

5 Data Modeling in EPIC Analyze

Data modeling in data-intensive systems is key to making them reliable, scalable, and efficient [1–3,18]. With respect to EPIC Analyze and its support for searching and filtering large data sets, we have observed that the data modeling decisions made in the lower layers of the software architecture impact how well EPIC Analyze is able to respond to query requests.

In this section, we describe the server-side facilities that respond to the queries made by an analyst. We illustrate this infrastructure by stepping through a typical search request-response cycle; we then discuss two data modeling challenges we faced in providing a good user experience and the techniques we employed to solve them. We connect these lessons with the impact that they had on improving EPIC Analyze's browsing interface.

5.1 A Day in the Life of a Query

The application architecture of EPIC Analyze (see Fig. 4) is composed of various components to properly serve HTTP requests coming from the user interface. Some of these components are tied to the application framework (Ruby on Rails), such as the Datasets Controller, QueryDispatcher, and Model interfaces, and some of them are acting as independent entities. For example, the EPIC Gem, which is a Ruby gem that helps query data sources like Cassandra, Solr, and Redis, is a separate library that is integrated for use within the EPIC Analyze application and serves other external use cases. We now present a typical request-response cycle to properly serve a query from the web interface and explain how each component serves an important function in this process.

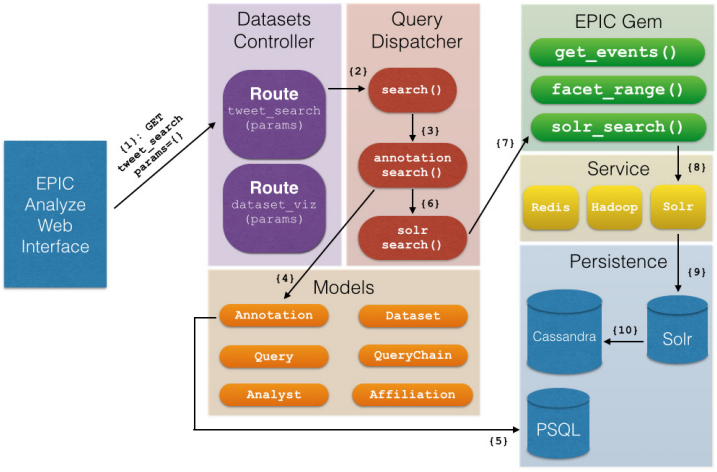

Fig. 4. EPIC Analyze Query Request Cycle

Important to this discussion are the objects that we have designed for this pipeline. The QueryChain object is a helper class designed for keeping track of Query objects, which are themselves encapsulations of query strings transformed from user input. More discussion of these two objects can be found in the next section. Additionally, it is important to highlight here that the data being computed and served are tweets, and so the common denominator between and among these data sources are the tweet id and its row key from Cassandra. Each secondary data source such as Solr and PostgreSQL may store additional information, such as enrichments to the data, but they are always bound to a tweet id and row key that are linked back to the primary data source, Cassandra. We refer to pairwise units of tweet id and row key as *tweet references*, because

they represent the address for every tweet in the Cassandra cluster. More information on the data modeling decisions in our persistence layer with respect to these tweet references can be found in [1,18].

A typical search request for annotations or tweets in EPIC Analyze is made on the web interface and is illustrated by the following request-response cycle:

1. A request to issue a query through the web interface begins with a set of filter parameters that are sent via an HTTP GET request. The request is received by the datasets controller route named `tweet_search`; it is responsible for unpacking the request and invoking the QueryDispatcher to properly route the request's filter parameters to the appropriate data sources.
2. The QueryDispatcher receives the request payload and the current QueryChain object, which keeps track of Query objects in the order they were created in the user interface. The filter payload is sent through the Query constructor and—based on its parameters—is transformed into a Query subclass, either AnnotationQuery or SolrQuery. The main objective of QueryDispatcher is to send the QueryChain through to neighboring functions that process the query against our persistent data sources (PostgreSQL and Solr). In this case, an AnnotationQuery was created, and so QueryDispatcher passes the QueryChain instance to the `annotation_search` function.
3. The `annotation_search` function searches the QueryChain instance for AnnotationQuery instances, passes them to the Annotation model within the Rails framework, and returns search results of matching tweet references. This function is designed to aggregate the results from the AnnotationQuery instances found in the QueryChain and honor the logical operators that were expressed for each query. Since the system imposes a global AND operation between all Query instances, if an AnnotationQuery returns no results, we can short circuit the entire search operation and return no results back to QueryDispatcher which will, in turn, return no results back to the browser.
4. The Annotation model interface offers functionality for searching Annotation instances based on tweet ID, label name, label type, or comment. In this case, its search function is called with the current AnnotationQuery parameters and searches for Annotation instances that satisfy them.
5. Rails provides object-relational mapping facilities for linking model interfaces with various database systems, in this case PostgreSQL. Annotation instances are returned from calls to the Rails models and control flow is returned back to the QueryDispatcher.
6. At this point within the QueryDispatcher, either there are tweet references that have been returned from the Annotation model or not. If no results are returned, then logically-speaking, no further querying can satisfy the current request, so the QueryDispatcher will return nothing back to the datasets controller. In the normal case, however, control flow will continue with annotation-derived tweet references to the `solr_search` function.
7. The `solr_search` function collates SolrQuery instances from the QueryChain object and uses the EPIC gem to construct a Solr-based query. The function combines the query strings from these Solr Query instances

with any annotation-derived tweet references so that the impending Solr query is further constrained by the ids of those tweets.
8. The corresponding `custom_search` call is made within the EPIC gem; it accepts the query from the QueryDispatcher and makes the call to Solr.
9. Solr executes the query according to the parameters and returns paginated tweet references back to the QueryDispatcher.
10. With a paginated payload of tweet references, the QueryDispatcher must now resolve the tweet references to retrieve the full Tweet JSON object from the Cassandra cluster. It achieves this by calling the EPIC gem to make a batch call to fetch page-sized JSON objects from Cassandra. Once the resolved tweets are returned, the QueryDispatcher wraps them within Tweet object instances and returns them to the datasets controller, whereupon they are formatted into a response payload that is returned back to the analyst.

5.2 Query Visibility

While interfaces from tools like Google Refine, Wrangler, and Splunk have powerful ways to query against large amounts of data, some of them struggle with providing visibility in their effectiveness against the data. Google Refine will display updated counts on facets and the total, but that does not describe when or where the query affected the data most. Splunk does have a visual timeline that updates itself after each query but falls short of providing usable facilities to perform previous queries, or to string them together without having to learn a query language based on pipe commands. Not enough of these tools promote the *tangible* aspects of the queries that are issued: how to create, jump between, and destroy them, as well as to view their history. That is why we implemented the interface described in Section 3; it directly supports these features. To make this possible, we had to implement two new domain models: Query and QueryChain.

In previous versions of EPIC Analyze, querying was relatively straightforward. A simple filter form was designed to accommodate filtering most fields, including tweet text, retweet count, date created range, the presence of URLs, the presence of geo-coordinates, and more. After the user submitted the filter, the set of filter parameters was transformed into parameters that could be used to submit to Solr. This achieved basic querying against large data sets, but it did not provide any visibility into the queries after the result set was displayed. Recall that most of the research analysts we interviewed felt as though the interface was missing the "big picture" of the event. They did not want to be scouring through thousands of tweets page after page. They wanted to see how their queries were making a difference in filtering out the data.

These issues of visibility and tangibility were solved with the Query and QueryChain objects. As mentioned before, the Query object is an encapsulation of data attributes coming from the filter parameters on the web interface. The QueryChain object keeps track of the Query instances created and groups them by instance type to run queries against the appropriate data source. With these querying facilities established, it becomes straightforward to display them on the

interface. A "Current Filters" section in the filter form displays a list of all of the queries that have previously been made.

Analysts now are able to create new queries, which will show up on the list after they have been processed. Analysts can also click on any one of these queries, and the datasets controller will identify an index that was linked to the clicked-on query. The QueryDispatcher is then able to re-arrange a subset of the queries with respect to the Query instance, whose name was clicked on, and execute them. The results reflect the query chosen, as well as the queries that came before it in the QueryChain. We refer to this user-action as *query-jumping* because the analyst is able to jump between previous queries to view the results that each one has made in drilling down into the data set. This provides visibility in the impact of each query, as well as control to the analyst in how to redirect analyst-made queries if (for example) the most recent query was not effective.

Additionally, analysts can delete any query along the chain, as shown in the filter form (see Fig. 2) with the X-mark next to each query name. Once deleted, the query will be removed from the QueryChain object, and the analyst will be able to continue making queries from the latest point while ignoring the effects of the deleted query. These query-based controls empower the analyst to make decisions without consequence: that is, the basic yet crucial operations that can be performed on queries is the basis for how analysts can drill down into the data set to identify the most important features for their research questions.

5.3 Query Expression

The ability to perform these operations on queries is critical for exploratory data analysis on large data sets, but not if the querying expression is not powerful enough. Based on our think-aloud sessions, we learned that the most basic querying operators—AND, OR, and NOT—reflected the kinds of queries that analysts wanted to make, such as: *Give me all of the tweets in the 2012 Hurricane Sandy data set that are geo-located, and then take away all the ones that have the screen name:"spammer123."* This kind of natural inquiry should be possible, assuming that the data fields are schematized under the data source and indexed for interactive response (in this case, Solr). However, what if the analysts want to make an additional query, such as, *Now, remove the tweets that have been previously tagged with:"social media" or "local/state government?"* How can a web interface provide the flexibility and expression for querying against multiple data sources? Current analytical tools either do not support this or struggle to provide timely responses for both dataset-specific and user-defined data.

Unbeknownst to the analyst, the nature of this problem requires insight into the complexity of algorithms needed to satisfy such heterogeneous queries. We observe here that there are naive ways to perform them, such as a brute force result set intersection: let the QueryDispatcher have the annotation search return tweet reference results based on the AnnotationQuery instances, and then let the Solr search return tweet references based on the SolrQuery instances. The resulting work now involves having to intersect these two result sets. However, the problem is that the result set coming from Solr is arbitrarily large—Solr

indexes tweets for datasets that have millions of records. Performing the intersection between annotations and tweets would be burdensome and take too long to return back to the user responsively.

Again, data modeling helped us find a solution. Recall the search request-response cycle in Fig. 4. The Annotations database will return result sets that are sufficiently smaller on average than those returned by the Solr index. This is evident because while EPIC Analyze supports annotations on tweets during exploratory analysis, most analysts will not manually annotate on the order of more than thousands of tweets. Therefore, annotation-based tweet references can be processed independently before searching the Solr Index, regardless of the ordering of the Query instances in the QueryChain.

Indeed, the QueryDispatcher will send the QueryChain object to its `annotation_search` function to collate all of the AnnotationQuery instances and aggregate any tweet references that satisfy the annotation-based filters. For each query—issued with the AND, OR, or NOT operator by the analyst—the results are stored in a data structure we refer to as the *tweet reference groupings*; each group contains tweet references and the chosen operator for that query. Once complete, the annotation-based tweet reference groupings are fed into the Solr query as an extra argument, which strings together the tweet ids and joins them with the logical AND operator. That way, local filter operators and the implicit global AND between all queries in the QueryChain are represented, and Solr will process a result set that is constrainted by the tweet reference groupings generated by the annotation search.

This approach works solely because tweet references are the lowest common denominator among these data sources. If we were to include additional sources that provide further enrichments to the user experience, the data modeling would then easily extend to those technologies if they also persisted tweet references. For our application infrastructure, we have seen no observable latency in Solr calls given that annotation-based tweet reference groupings are passed into the Solr query. Indeed, since Solr is supported by computational resources in our infrastructure that support indexes of more than 4 million tweets, query results are usually completed in a few seconds.

6 Conclusions and Future Work

In this paper, we have presented our work on designing user interfaces for large-scale data analysis environments. We reported on challenges we faced designing a user interface for the browser of EPIC Analyze, our custom-designed analysis platform for crisis informatics research. Our work—influenced by our strong human-centered computing perspective—took advantage of feedback provided by seven Project EPIC analysts who work with crisis data sets on a daily basis. Their feedback led to a number of improvements that allowed EPIC Analyze to become a tool they use on a daily basis. We focused on a challenging user interface design puzzle—drilling down into a large data set—and showed how the solution required not just user interface elements but objects on the server side that allowed the problem to be cleanly modeled and implemented effeciently.

Future work on EPIC Analyze will support 1) automatic tagging of tweets with labels that match user-supplied rules; 2) geo-located tweets, including a map-based timeline that shows how tweets have appeared in a specific area over time; 3) social graphs that exist among Twitter users; and 4) streaming services that enable analysts to filter and monitor dynamic data through interactive real-time visualizations. For real-time data, the querying facilities will be decoupled from data sources like Cassandra and Solr and instead be designed to handle *any* data source. For each of these tasks, the user interfaces that we develop to support them will be the result of a highly-iterative participatory design process that is conducted in tandem with any modifications to the server side data model. We believe our approach can be applied to data analysis platforms in general and not just to platforms that support crisis informatics research.

Acknowledgments. This material is based upon work sponsored by the NSF under Grant IIS-0910586. We would like to thank the seven analysts who participated in our think-aloud sessions; their feedback helped us to improve EPIC Analyze.

References

1. Anderson, K.M., Aydin, A.A., Barrenechea, M., Cardenas, A., Hakeem, M., Jambi, S.: Design challenges/solutions for environments supporting the analysis of social media data in crisis informatics research. In: 48th Hawaii International Conference on System Sciences, pp. 163–172. IEEE, January 2015
2. Anderson, K.M., Schram, A.: Design and implementation of a data analytics infrastructure in support of crisis informatics research (nier track). In: 33rd International Conference on Software Engineering, pp. 844–847. ACM, May 2011
3. Anderson, K.M., Schram, A., Alzabarah, A., Palen, L.: Architectural implications of social media analytics in support of crisis informatics research. IEEE Bulletin of the Technical Committee on Data Engineering **36**(3), 13–20 (2013)
4. Aydin, A.A., Anderson, K.M.: Incremental sorting for large dynamic data sets. In: First IEEE International Conference on Big Data Computing Service and Applications. IEEE, March + April 2015
5. Bargas-Avila, J., Hornbæk, K.: Foci and blind spots in user experience research. Interactions **19**(6), 24 (2012)
6. Cameron, M.A., Power, R., Robinson, B., Yin, J.: Emergency situation awareness from twitter for crisis management. In: International Conference Companion on World Wide Web, pp. 695–698. ACM (2012)
7. Fisher, D., DeLine, R., Czerwinski, M., Drucker, S.: Interactions with big data analytics. Interactions **19**(3), 50–59 (2012)
8. Heer, J., Kandel, S.: Interactive analysis of big data. Crossroads **19**(1), 50–54 (2012)
9. Hughes, A.L., St. Denis, L.A.A., Palen, L., Anderson, K.M.: Online public communications by police & fire services during the 2012 hurricane sandy. In: Human Factors in Computing Systems, pp. 1505–1514. ACM (2014)
10. Jara Laconich, J.J., Casati, F., Marchese, M.: Social spreadsheet. In: Daniel, F., Dolog, P., Li, Q. (eds.) ICWE 2013. LNCS, vol. 7977, pp. 156–170. Springer, Heidelberg (2013)

11. Kandel, S., Paepcke, A., Hellerstein, J., Heer, J.: Wrangler: Interactive visual specification of data transformation scripts. In: Human Factors in Computing Systems, pp. 3363–3372. ACM (2011)
12. Lewis, C., Rieman, J.: Task-centered User Interface Design: A Practical Introduction. Department of Computer Science, University of Colorado, Boulder (1993)
13. McTaggart, C.: Analysis and Implementation of Software Tools to Support Research in Crisis Informatics. Master's thesis, University of Colorado (2012)
14. Oussalah, M., Bhat, F., Challis, K., Schnier, T.: A software architecture for Twitter collection, search and geolocation services. Knowledge-Based Systems **37**, 105–120 (2013)
15. Palen, L., Anderson, K.M., Mark, G., Martin, J., Sicker, D., Palmer, M., Grunwald, D.: A vision for technology-mediated support for public participation & assistance in mass emergencies & disasters. In: ACM-BCS Visions of Computer Science. British Computer Society (2010)
16. Palen, L., Soden, R., Anderson, T.J., Barrenechea, M.: Success and scale in a data-producing organization: The socio-technical evolution of openstreetmap in response to humanitarian events. In: Human Factors in Computing Systems. ACM (2015)
17. Sarcevic, A., Palen, L., White, J., Starbird, K., Bagdouri, M., Anderson, K.M.: "beacons of hope" in decentralized coordination: Learning from on-the-ground medical twitterers during the 2010 haiti earthquake. In: Computer Supported Cooperative Work, pp. 47–56. ACM (2012)
18. Schram, A., Anderson, K.M.: MySQL to NoSQL: Data modeling challenges in supporting scalability. In: Systems, Programming, Languages and Applications: Software for Humanity, pp. 191–202. ACM (2012)
19. Shneiderman, B.: The eyes have it: A task by data type taxonomy for information visualizations. In: IEEE Symposium on Visual Languages, pp. 336–343. IEEE (1996)
20. St. Denis, L., Hughes, A., Palen, L.: Trial by Fire: The Deployment of Trusted Digital Volunteers in the 2011 Shadow Lake Fire. In: Rothkrantz, L., Ristvej, J., Franco, Z. (eds.) International Conference on Information Systems for Crisis Response and Management, pp. 1–10 (2012)
21. Starbird, K., Palen, L.: "voluntweeters:" self-organizing by digital volunteers in times of crisis. In: Human Factors in Computing Systems, pp. 1071–1080. ACM (2011)
22. Starbird, K., Palen, L., Hughes, A., Vieweg, S.: Chatter on the red: What hazards threat reveals about the social life of microblogged information. In: Computer Supported Cooperative Work and Social Computing, pp. 241–250. ACM, February 2010
23. White, J., Palen, L., Anderson, K.M.: Digital mobilization in disaster response: The work & self-organization of on-line pet advocates in response to hurricane sandy. In: Computer Supported Cooperative Work and Social Computing, pp. 866–876. ACM (2014)
24. Wongsuphasawat, K., Guerra Gómez, J.A., Plaisant, C., Wang, T.D., Taieb-Maimon, M., Shneiderman, B.: Lifeflow: Visualizing an overview of event sequences. In: Human Factors in Computing Systems, pp. 1747–1756. ACM (2011)