Final Report – CS 6604 Spring 2017

Global Events Team

CS 6604: TS: Digital Libraries

Instructor: Dr. Edward A. Fox

Liuqing Li, Islam Harb, Andrej Galad
{liuqing, iharb, agalad} @vt.edu

Virginia Polytechnic Institute and State University
Blacksburg, VA, 24061

May 3, 2017

Abstract

Researchers make full use of webpage data in various research areas, such as topic modeling, natural language processing (NLP), text mining, social networks, etc. With the Internet Archive's help, people are able to retrieve webpages covering different topics across more than 20 years, but currently there are many challenges in processing such longitudinal data.

Leveraging the Hadoop Cluster in the Digital Library and Research Laboratory (DLRL) and previous work on the IDEAL project, our team will enrich the general infrastructure supporting the GETAR project, funded by NSF. There are three sub-goals. First, we need to gather webpages about multiple global events and convert them into standard webpage format. Those collections are the data sources for further processing. Second, by leveraging the existing tools and techniques, our team is aiming at entity extraction from those webpages to help analyze those collections. Based on those entities, we are able to describe those events over time. Finally, to accomplish the whole pipeline, an interface is required to display entities and trends for users.

In order to accomplish our first goal, our team focused on the school shooting collections, part of the global event collections. We prepared good seeds for the school shooting events that took place in the past 10 years and enhanced the Event Focused Crawler (EFC, developed by Dr. Mohamed Magdy Farag, a collaborator and alumnus from DLRL) to download the webpages and merge them in the Web ARChive (WARC) format. To achieve the second goal, we deployed ArchiveSpark on a stand-alone server, and made full use of the previous reports on the IDEAL project and prototypes of different modules created by the former teams, including noise reduction and named entity recognition. Because a huge number of entities were extracted from the WARC files, some statistical methods have been introduced to create the connections among those entities and provide better descriptions of various events. For the third goal, we implemented a D3-based self-contained Web application by using Gradle, and HBase was selected to be a connecter between the collection processing module and Gradle.

Our project focuses on the GETAR project and ArchiveSpark, as well as other important components such as NLP, named entity recognition, statistical approaches. Beyond these, there are also some relevant components including Spark, Scala and D3. Our Global Event team will mention these components later in this report.

Table of Contents

Table of Tables

Table of Figures

1 Overview

1.1 Management

Intra-team collaboration is an essential step for completing the project. Our entire workflow pipeline depends on clear separation of concerns and distribution of responsibilities. In order not to needlessly waste time we established a plan of regular weekly meetings to discuss progress, plans, blockers, and status updates. To better organize our work, we decided to create a shared Google Drive folder as well as open-source our work using public GitHub repositories.

In terms of technology we opted to leave the choice to actual implementers. Our pipeline process only requires clear definition of input and output formats of data as each phase essentially depends on another, but in terms of inner workings of separate modules the choice is essentially up to the developer. The only constraint is that selected technologies must be supported by the DLRL Hadoop cluster for seamless deployment and dependency resolution.

1.2 Challenges

Our team has already identified several challenges, but we hope to solve all of them during this semester. Since we are likely to have new challenges later, which will also be varied over time, we will update this section as challenges come up or have been solved.

One of the biggest challenges in our project is the data that we need to collect, analyze, and visualize. As we are looking for trends of events that span over multiple years (e.g., ~10 years), Internet Archive (IA) sounds like a strong candidate as a source for such data collections. However, we realize that we don't have access to the interesting and relevant collections, in IA, for the purpose of our research project. Although we have credentials on IA, access permissions need to be granted on a collection-by-collection basis (not per credential). This process may take a very long time before we can have access to required data. For example, we were interested in two shooting collections: Tucson Shooting Anniversary 2012 (ArchiveIt-Collection-2998) and Norway Shooting July 23, 2011 (ArchiveIt-Collection-2772). Unfortunately, we don't have access to them. It would be helpful if we can establish a method of communication with IA to facilitate permission grants to collections of interest. One possible way is by preparing a list of collection IDs regularly (e.g., on a monthly or semester basis), then provide IA with this list. In return, they grant us the permission to these collections, so that we can use their tools to obtain them. Meanwhile, due to this access limitation, we excluded IA from our candidate data collection approaches. However, we have investigated the methodology and tools required to download any data collection from IA, if we have the requisite permission. You may refer to the Developer Manual (Section 7.1) for more details on the steps to download IA data collections.

Archive-It is another candidate for getting our data as we already have about 66 data collections that were created/archived by the Digital Library Research Laboratory (DLRL) in specific and

Virginia Tech (VT) in general. However, these collections suffer from two issues (1) they are quite small such that some of them comprise just a few webpages, (2) they have been collected over 1-2 years at most. We are able to obtain them by running the following simple command or an alternative approach that will be described in Section 7.1.

```
wget --http-user=<user> --http-password=<password> --recursive --accept gz,txt
https://warcs.archive-it.org/cgi-bin/getarcs.pl?c=<collection #>
```

However, they are insufficient for our study that needs data collections that span over ~10 years to look for trends and correlations. Nevertheless, with the help of Archive-It, if we can retrieve and store more relevant event data from now on, it will be very helpful for future analysis.

In order to analyze the trends over a long period of time, a huge number of techniques could be leveraged in our project, such as natural language processing [9], named entity recognition, topic modeling, document classification, and clustering. Unfortunately, there are still some challenges for our team to solve. After leveraging the name entity recognizer, we are able to extract a huge number of entities from the school shooting collections, but it is difficult to leverage them to describe those events. Moreover, for some features, such as shooter's name and weapon list, specific filters should be built to fit for the specific cases.

Moreover, another challenge we faced is an incompatible Cloudera version with some of our tools - namely ArchiveSpark. Unfortunately, in its current state the DLRL CDH Hadoop cluster hosts one of the older versions of Spark (Version 1.5.0) whereas the ArchiveSpark library leverages some of the API only present in Spark 1.6.1 onwards. We hope this situation can be resolved soon and proper versions of tools can be installed. For the time being, we are using a dedicated cluster node with the latest version of Spark installed.

Finally, while developing our collection visualization tool some of the problems we encountered had to do with broken JQuery UI libraries for rendering interactive components such as term count spinner or D3.js interactive line chart. We ended up fixing the issue by relying on HTML5 elements instead and by using Bootstrap for responsive fonts and elements based on the user viewport (screen) size.

1.3 Solution Developed

For the entity extraction, we designed and implemented multiple techniques, including basic parsing, Stanford NER, and regular expressions. Some techniques have been merged together to identify the event features. For instance, we leveraged basic parsing and regular expressions to extract the event date. Stanford NER and regular expressions have been used for identifying the shooter's name. After extracting those entities from one collection, we create a score function to rank them and take the most frequent item as the final output. The score function is based on both term frequency and document frequency, which gets rid of noisy data and performs well in the big collections.

For our data visualization, we developed a Java Web application - Global Events Viewer - allowing efficient profiling of a Web archive collection based on the frequency of terms (using word cloud visualization), as well as providing a general overview of a particular collection based on the extracted trends - victims count and shooter's age. As mentioned, our solution is developed in Java, leveraging the Gradle [18] build system, for fast and efficient deployment across various environments, together with the Spring Boot framework [26], allowing simple and quick deployment. The Global Events Viewer can be configured with either in-memory backend/database (static data) or HBase (containing results of data preprocessing). The UI of our application relies on HTML5/CSS3 constructs as well as several JavaScript for Document Object Model (DOM) manipulations - JQuery, JQuery UI, Bootstrap.js, and D3.js.

2 Literature Review

Web crawling is a huge challenge with the existence of billions of Web resources scattered and distributed on the Internet. Hence, there has been a need for a better and more focused crawler to improve the search engines' operation. Authors in [1] present a new approach, on focused crawling for events, to increase the retrieved webpages' relevance to an event of interest. Their work considers topics included in that event (e.g., shooting, hurricane, etc.), locations at which the event takes place, and the date of an event's occurrence, to ensure the freshness of the retrieved relevant webpages. The authors adopted machine learning techniques (i.e., SVM) to build their relevant vs. non-relevant binary classifier/model. They used manually selected/curated seed URLs to build their classifier/model in which they followed the 70% (training set) - 30% (testing set) methodology. They used their model to find the optimal values for their parameters, which are the number of keywords and the decision-making threshold cut-off. They show better results compared to the traditional baseline focused crawler (i.e., only topic(s) is/are considered).

Another effort [2] towards building a focused crawler is proposed as an extension to the unfocused crawler Apache Nutch. It considers both the topic(s) and date (through freshness concept via incorporating URLs coming from tweets) to improve the retrieved webpages' relevance to a specific topic/event. It integrates the Twitter streaming API into the crawling flow by continuously generating/enriching the priority queue with URLs to ensure the freshness of the retrieved relevant webpages. During the crawling process, the priority queue is being filled up with URLs coming from both webpages and tweets. The webpages' relevance is measured by applying cosine similarity on the topic vector. However, for the tweets, other metrics, such as popularity and user profile, are used as the content is very short (i.e., 140 characters only) and insufficient for content-based relevance check.

In [3] authors proposed "anthelion", a focused crawler, that combines a bandit-based selection strategy with online classification to direct a Web crawler towards relevant webpages. It targets webpages with structured data that are based on semantic annotations with markup standards

Microdata, Microformats, and RDFa. Semantic annotations of webpages make it easier to extract and reuse data. It has shown increased usage over the past years, by both search engines and social media sites, to provide better search experiences. Their approach shows that the percentage of relevant webpages may increase up to ~26% relative to a pure online classification-based approach. Their results also show that anthelion can gather ~66% more relevant pages within the first million than a pure online classification based approach.

Currently, there are various techniques and tools developed to analyze webpages or other relevant collections. Beautiful Soup [11] is a Python library which provides multiple functions for navigating, searching, and modifying a parse tree. The tool is able to automatically convert incoming documents to Unicode and outgoing documents to UTF-8. Moreover, it sits on top of popular Python parsers like lxml and html5lib, allowing us to try out different parsing strategies or trade speed for flexibility. For natural language processing, NLTK [10] is a powerful tool for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. The Stanford Named Entity Recognizer (NER) [25] is a Java implementation developed by the Stanford NLP Group. It labels sequences of words in a text which are the names of things, such as person and company names, or gene and protein names. For event identification and extraction, association rule learning [12] may be used in our project, which includes some traditional algorithms like Apriori [13], Eclat [14], and FP-growth [15]. Our team also read the relevant reports from the previous teams in CS5604, including Named Entity Recognition for IDEAL [4], Reducing Noise for IDEAL [5], and Document Clustering for IDEAL [6], which could be very helpful for our current work.

In terms of visualization techniques, the majority of them come as library functionality provided in d3.js [16]. One notable exception might be a special case of stacked graphs that we intend to use in order to visualize trend data over the years - ThemeRiver. This technique, proposed by Lee Byron and Martin Wattenberg in their paper “Stacked Graphs - Geometry & Aesthetics” [17], essentially optimizes presentation space by arranging stacked graph data around the x axis (from both sides) either based on symmetry or based on minimization of the sum of squares of each slope.

3 Requirements

This semester, our team is focusing on global events over a long period of time. To achieve this goal, we will develop a novel approach to enrich the relevant collections. Then, some existing techniques will be leveraged or improved to generate the results for trends. Finally, a Web front-end will be designed and implemented to visualize the data and provide better user experiences. Some of the requirements are explained as follows.

1. Crawl School Shooting Webpages. In order to get data collections on school shooting for the past 10 years, we need to prepare good seeds for the school shooting events that took place. This process will be done manually to assure the high quality of the seeds. Another requirement for our crawling process is to instrument the Event Focused Crawler to archive the webpages in WARC format. This serves our long-term plan to send this collection, after cleaning, out to IA for broader benefits to the public. Also, WARC format seems to work very well with ArchiveSpark, our tool that we use for webpage processing and analysis. Finally, we expect that this crawling process may take considerable/long time before we have enough data for our study. Therefore, we have another requirement to get access on several machines so that each machine will take a subset of the seed URLs to crawl the corresponding webpages.

2. Improve the current methods for trends analysis. Based on the literature review, our team will make full use of the previous reports on the IDEAL project. NLP will be carried out in this part. Some teams have already created prototypes of different modules, such as noise reduction, named entity recognition, and document clustering. Based on the school shooting collection, it will be very helpful to leverage those functionalities to process the relevant data. Moreover, we will build the connections among a couple of entities to represent a specific event. For a certain event, multiple dimensions will be taken into consideration, as shown in Table 1.

Table 1 Multiple Dimensions for Trends Analysis

Dimension	Example	Description
Date	April 16, 2007	Shooting date. It could be grouped with different time bins.
Location	Blacksburg, Virginia	Shooting location. It is able to generate the longitude and latitude by using the Google Geocoding API.
Shooter	Seung-Hui Cho	The name of the shooter.
Age	23	The age of the shooter.
Weapon	handgun	From multiple types of weapons, such as handgun, rifle.
Nationality	South Korean	The race of the shooter.
Start Time	7:15am	The start time of the shooting.
Victims	32	The number of people killed in the shooting.
Ending	suicide	The ending of the shooter (e.g., suicide, killed, arrested).

Our team will parse and process the webpage collections into the above dimensions, and then store the results into the database for the collection visualization.

3. Visualize the results with locations and trends. This is essentially the very last step in the pipeline. The main idea is to take structured, preprocessed, aggregated data and efficiently convey the information that they hold to the user. In addition, we plan for our solution to be applicable to any given collection with processed data. As such some specific requirements in terms of data structuring and storage must be upheld. Overall, the visualization project will be a self-contained Web application implemented using the Gradle [18] build tool to facilitate deployment and eliminate issues connected with dependency installation and permission. The database layer acts as a connection interface between the collection processing module and the viz module. For the most part the schema should be dynamic and easily extensible - as such HBase presents itself as the most suitable candidate. The backend layer will be written in Java and deployed using an embedded application container. The frontend will heavily rely on d3.js as well as Bootstrap [19] for responsive design.

4 Design

An overview of the architecture of the GETAR [8] system was presented as shown in Figure 1. As extension of the IDEAL project [7], some initial work has been completed towards handling global events. Based on the previous work, we are aiming at improving the current pipeline for trend analysis. Specifically, our team chose the webpage collection on school shooting as the source. By leveraging the relevant techniques (e.g., NLP, info extraction, clustering), we are able to analyze the trends. For the front-end, a Web server will be established to visualize the results. By doing this, our team hopes to provide one possible solution to trend analysis. After that, members in the GETAR project will be able to improve the pipeline and enrich the whole architecture, or choose one specific branch for further research.

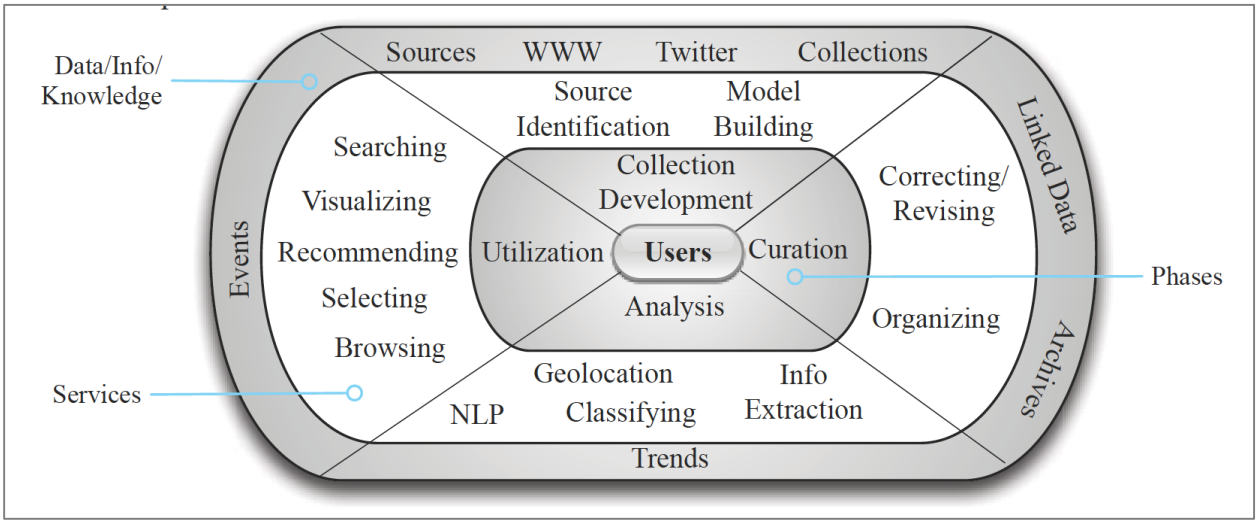

Figure 1 Architecture of GETAR [8]

Based on our specific requirements, we created a pipeline for our project. Figure 2 shows the architecture of our project. There are mainly three stages. First, we leverage EFC to crawl relevant webpages and merge them together into WARC files. Next, CDX Writer can help us generate the index files of those WARC files. Then, both WARC files and CDX files are imported into ArchiveSpark for further analysis. During this procedure, multiple techniques can be used to process the data, extract the entities, and create an entity-based output. Finally, a Web application has been designed to read those results from HBase and show them through a user interface.

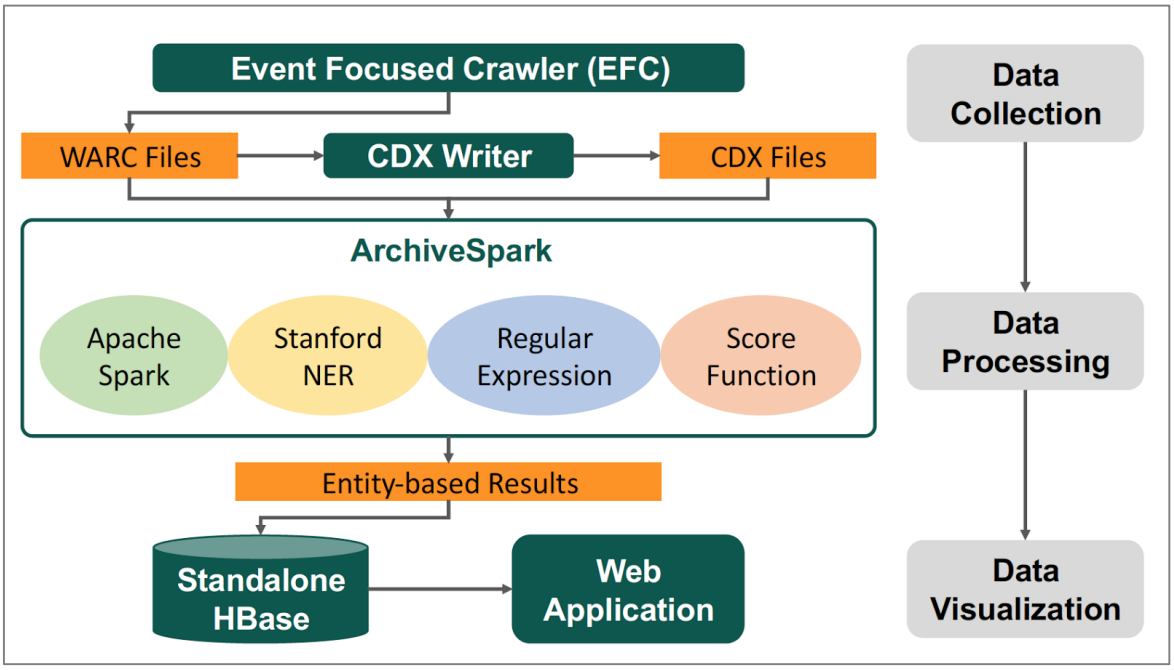

Figure 2 Architecture of CS6604 Project

4.1 Events Crawling Design

In Section 3, data collections on school shootings for the past 10 years are required for our further analysis. Thus, in order to meet our requirements and build the whole pipeline in an efficient way, we manually selected 10 representative school shooting events during the past 10 years. Based on those events, we are able to generate the seeds and then leverage EFC to crawl and produce more WARC files. The 10 school shooting events are shown in Table 2.

Table 2 School Shooting Event List

Event 1	**Virginia Tech Shooting**		
Date	April 16, 2007	**Deaths**	32
Location	Blacksburg, Virginia	**Injuries**	23
Description			
A 23-year-old student, Seung-Hui Cho, killed thirty-two students and faculty members at Virginia Tech, and wounded another seventeen students and faculty members in two separate attacks before committing suicide			

Event 2	Northern Illinois University Shooting		
Date	February 14, 2008	**Deaths**	6
Location	DeKalb, Illinois	**Injuries**	21
Description			
A 27-year-old Steven Kazmierczak, shot multiple people with a shotgun in a classroom of Northern Illinois University, killing five and injuring 21, before taking his own life.			

Event 3	Dunbar High School Shooting		
Date	January 9, 2009	**Deaths**	0
Location	Chicago, Illinois	**Injuries**	5
Description			
After attendees were leaving a basketball game at Dunbar High School, a truck pulled over and someone inside fired shots at the crowd.			

Event 4	University of Alabama Shooting		
Date	February 12, 2010	**Deaths**	3
Location	Huntsville, Alabama	**Injuries**	3
Description			
A 44-year-old biology professor, Amy Bishop, killed the chairman of the biology department, 52-year-old Gopi K. Podila, and biology professors, 50-year-old Maria Ragland Davis and 52-year-old Adriel D. Johnson.			

Event 5	Worthing High School Shooting		
Date	March 31, 2011	**Deaths**	1
Location	Houston, Texas	**Injuries**	5
Description			
Multiple gunmen opened fire during a powder puff football game at Worthing High School. One man, an 18-year-old former student named Tremaine De Ante' Paul, died. Five other people were injured.			

Event 6	Sandy Hook Elementary School Shooting		
Date	December 14, 2012	**Deaths**	27
Location	Newtown, Connecticut	**Injuries**	2
Description			
20-year-old Adam Lanza, killed twenty-six people, his mother and himself. He killed twenty first-grade children aged six and six adult staff members during the attack at school (i.e., four teachers, the principal, and the school psychologist). Two other persons were injured. Lanza then killed himself as police arrived at the school.			

Event 7	Sparks Middle School Shooting		
Date	October 21, 2013	**Deaths**	2
Location	Sparks, Nevada	**Injuries**	2
Description			
A 12-year-old seventh-grade student Jose Reyes opened fire with a handgun at the basketball courts of Sparks Middle School.			

Event 8	Reynolds High School Shooting		
Date	June 10, 2014	**Deaths**	2

Location	Troutdale, Oregon	Injuries	1
Description			
15-year-old Jared Padgett, exchanged gunfire with police officers and then committed suicide in a restroom stall.			

Event 9	Umpqua Community College Shooting		
Date	October 1, 2015	**Deaths**	10
Location	Roseburg, Oregon	**Injuries**	9
Description			
A gunman, identified as 26-year-old student Christopher Harper-Mercer, opened fire in a hall on the Umpqua Community College campus, killing eight students and one teacher, and injuring nine others. Mercer then committed suicide after engaging responding police officers in a brief gunfight.			

Event 10	Townville Elementary School Shooting		
Date	September 28, 2016	**Deaths**	2
Location	Townville, SC	**Injuries**	2
Description			
A teen opened fire at Townville Elementary School. The suspect's father was found dead at his home soon after the shooting.			

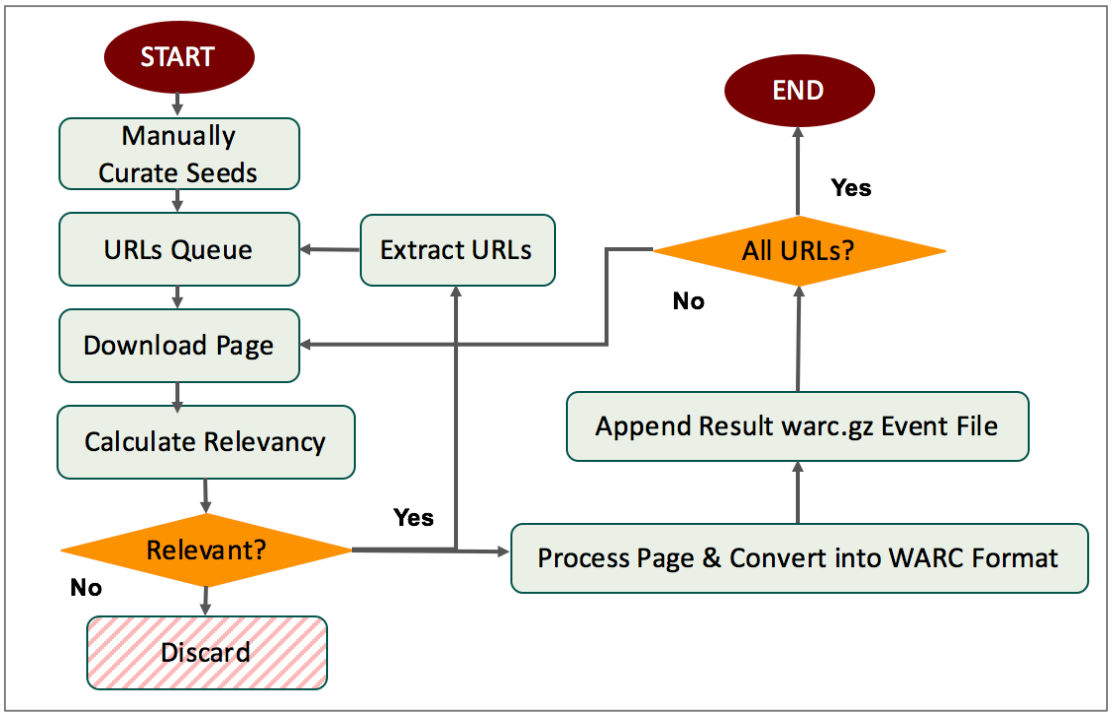

Figure 3 Modified Focused Crawler Flow

We modified the original EFC [1] to add the archiving capability. Figure 3 shows the flowchart of our modified EFC with two extra stages. Once the webpage is marked as relevant, the EFC will process and convert it into the WARC format. Later, the WARC_Writer component will append the webpage to its corresponding main event WARC-based archive.

```
islam 788109490 Apr 18 13:45 townville_2016.warc.gz
islam 758426598 Apr 18 13:43 umpqua_2015.warc.gz
islam 696778729 Apr 18 13:38 omaha_NE_2011.warc.gz
islam 661944603 Apr 18 13:40 Sandy_2012.warc.gz
islam 454081122 Apr 18 13:29 VT_2007.warc.gz
islam 279509497 Apr 18 13:41 Sparks_2013.warc.gz
islam  15635547 Apr 18 13:36 dunbar_chicago_2009.warc.gz
islam  14708587 Apr 18 13:41 reynolds_2014.warc.gz
islam   5705504 Apr 18 13:36 huntsville_2010.warc.gz
islam    531513 Apr 18 13:35 NIU_2008.warc.gz
```

Figure 4 Events Archives in WARC Format

We targeted to collect 25K webpages for each of the 10 events. Figure 4 shows the 10 archives with their size in bytes in the 2nd column. One of the challenges is that the relatively old events suffer from significant amount of broken links (removed/deleted webpages). This adversely impacts the size of their corresponding archives as shown in Figure 4 (e.g., NIU_2008).

4.2 HBase Schema Design

Based on the multiple dimensions for trends analysis, we use HBase for data storage and management. Our team designed the HBase schema to make a better user interface. Table 3 shows the HBase schema.

Table 3 HBase Schema

Row_Key	Event_Date + Event_Hash
Column Family 1: event	
Columns	
event: name	event: entities
event: date	event: entities_count
event: shooter_age	event: entities_url
event: shooting_victims	
Column Family 2: rich	
Columns	
rich: shooter_name	rich: weapon_list
rich: sner_people	rich: sner_locations
rich: sner_people_count	rich: sner_locations_count

rich: sner_people_links	rich: sner_locations_links
rich: sner_dates	rich: sner_organizations
rich: sner_dates_count	rich: sner_organizations_count
rich: sner_dates_links	rich: sner_organizations_links

5 Implementation

5.1 Overview

We have had several discussions with Prof. Fox on the whole procedure to get a better understanding of our tasks. Based on the current framework and previous work, our team is aiming at analyzing the trends of the school shooting collection over more than 10 years.

Based on the reviews of the previous work, we followed the tutorials and reports to get familiar with the existing technologies. We discussed the dataflow in the current infrastructure, which helped our team to get familiar with the pipeline of the school shooting data and make a clear division of labor and cooperation. Up to now, we are able to build a model based on the topic(s), date, and location of each of the shooting events. We will crawl the Internet to retrieve relevant webpages and download them into WARC files. ArchiveSpark has already been set up on a stand-alone machine to filter the WARC files efficiently.

5.2 Timeline

The timeline of our Global Events team is shown below, including each task, duration, current status, and responsible member for accomplishment. Moreover, deliverables are also listed in Table 4, that may be beneficial for future work.

Table 4 Timeline of Team Activities

#	Week		Task	Status	Assigned To
1	1	01/17 - 01/22	Set up Global Events team	Done	All
2	2	01/23 - 01/29	Set up share folder using Google Drive and hangouts for instant messaging	Done	All
3	3, 4, 5	01/30 - 02/19	Review the relevant literature	Done	All

4	5	02/13 - 02/19	Deploy ArchiveSpark on local machine	Done	Liuqing
5	6	02/20 - 02/26	Discuss interim report 1 with the instructor	Done	Islam, Andrej
6	6	02/20 - 02/26	Instrument the event focused crawler to download the crawled relevant webpages in WARC format	Done	Islam
7	6	02/20 - 02/26	Test ArchiveSpark with fake WARC files	Done	Liuqing
8	7	02/27 - 03/05	Design the features for visualization	Done	Andrej, Liuqing
9	7	02/27 - 03/05	Look for interesting school shooting events in the past 10 years	Done	Islam
10D	7	03/02	Interim report 1	Done	All
11	8	03/06 - 03/12	Manually curate/prepare high quality seed URLs	Done	Islam
12	8	03/06 - 03/12	Reduce noise for fake WARC files	Done	Liuqing
13	9	03/13 - 03/19	Recognize named entities from fake WARC files	Done	Liuqing
14	9	03/13 - 03/19	Backend implementation	Done	Andrej
15	9, 10	03/13 - 03/26	Run event focused crawler distributedly to crawl relevant webpages.	Done	Islam
16	10	03/20 - 03/26	Word Cloud	Done	Andrej
17	10, 11	03/20 - 04/02	Improve the quality of named entities	Done	Liuqing
18	11	03/27 - 04/02	Keyword Searching	Future	Andrej
19D	11	03/31	Interim report 2	Done	All
20	12	04/03 - 04/09	Integrate all the data into one big collection for further work	Done	Islam
21	12, 13	04/03 - 04/16	Implement association rule learning	Done	Liuqing
22	13	04/10 - 04/16	Events Location map	Future	Andrej
23	14	04/17 - 04/23	Improve the quality of the event features	Done	Liuqing
24	14	04/17 - 04/23	Create trends with time series	Done	Andrej
25	15	04/24 - 04/30	Process the school shooting collection	Done	Liuqing

26	15	04/24 - 04/30	Data integration	Done	All
27D	16	04/27	Final presentation	Done	All
28D	16	05/03	Final report	Done	All
D – Deliverables					

5.3 Tools

In this section, we introduce the relevant tools, including ArchiveSpark [21] and D3.js [16]. These modules and components help us understand the workflow of our system. We will add more in this section.

5.3.1 ArchiveSpark

One of the most important frameworks our system relies on is ArchiveSpark. This tool was first introduced in the same-named paper [22] authored by the researchers from Internet Archive [23], a San Francisco-based nonprofit digital library, in collaboration with L3S Research Center [24]. ArchiveSpark is essentially a Spark-based library/framework enabling efficient data access, extraction, and derivation, with Web archive data. Its main advantage over the existing techniques is in its superior performance (granted by numerous optimizations) as well as its extensibility.

One of the main reasons for ArchiveSpark's efficient computation is its utilization of CDX index files - essentially metadata descriptors extracted from an underlying collection's WARC files. These files are greatly reduced in size when compared to their counterparts allowing users to efficiently filter out all the noisy data before using another one of ArchiveSpark's main concepts - enrichments. Overall, enrichments really only describe transformation functions that lazily/on-demand fetch relevant information from associated WARC files in order to allow further processing/filtering. This workflow of incremental filtering allows to efficiently process vast collections, which would be otherwise unprocessable.

As far as the framework's extensibility is concerned the authors designed the tool in a way that allows easy integration of 3rd party libraries in overall processing. This feature really plays to our advantage as it essentially allows us to reuse some of the topic-specific code created by our predecessors as well as popular 3rd party libraries such as Stanford CoreNLP [25].

5.3.2 D3.js

Another very important library that powers most of our visualization is d3.js. This tiny yet powerful JavaScript framework is held in high esteem [32] both by the scientific and professional communities. The main purpose of d3.js is to provide automatic bindings between data and the DOM as well as to apply data-driven transformation. Additionally, one of the huge advantages of d3.js is its flourishing development community and its modularity - d3.js can be arbitrarily

extended using many open-sourced 3rd-party components and plugins. In the context of our project we are essentially looking at 3 different visualization techniques.

1. Word Cloud (D3-cloud [20])

Figure 5 An Example of Word Cloud [20]

We choose word cloud as our main technique for visualizing terms and their frequencies. The user should be able to select a date range and then receive top K terms describing a particular collection – the size of the terms should be influenced by their frequency.

2. Geo-Location Maps (Datamaps [21])

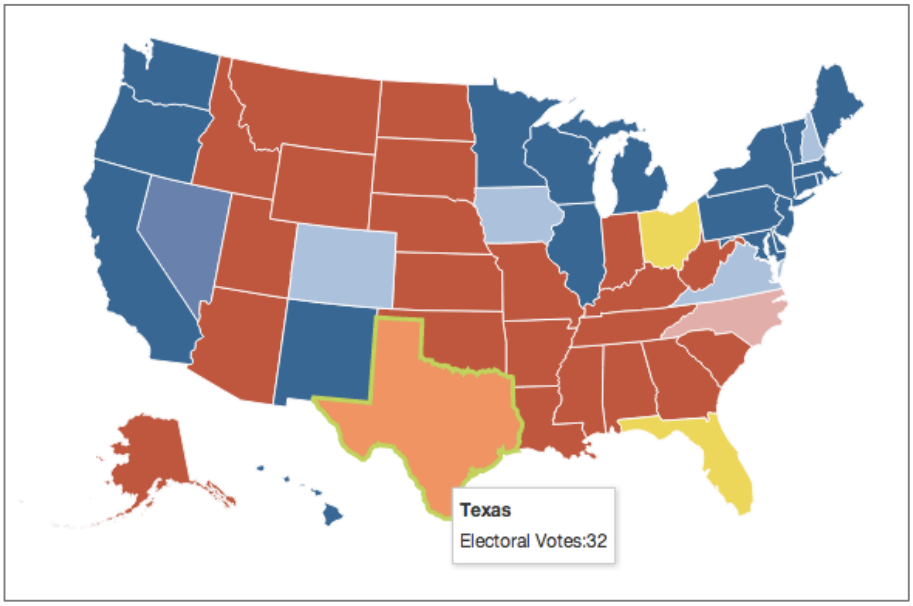

Figure 6 An Example of Geo-Location Based Events [21]

We intend to visualize events with regard to their geographical location. Leveraging external services such as the Google Geocoding API [31], we wish to extract event coordinates and plot them using the Datamaps library, as shown in Figure 6. For each event, we wish to visualize using localized circles on the map.

3. Timeseries/ThemeRivers

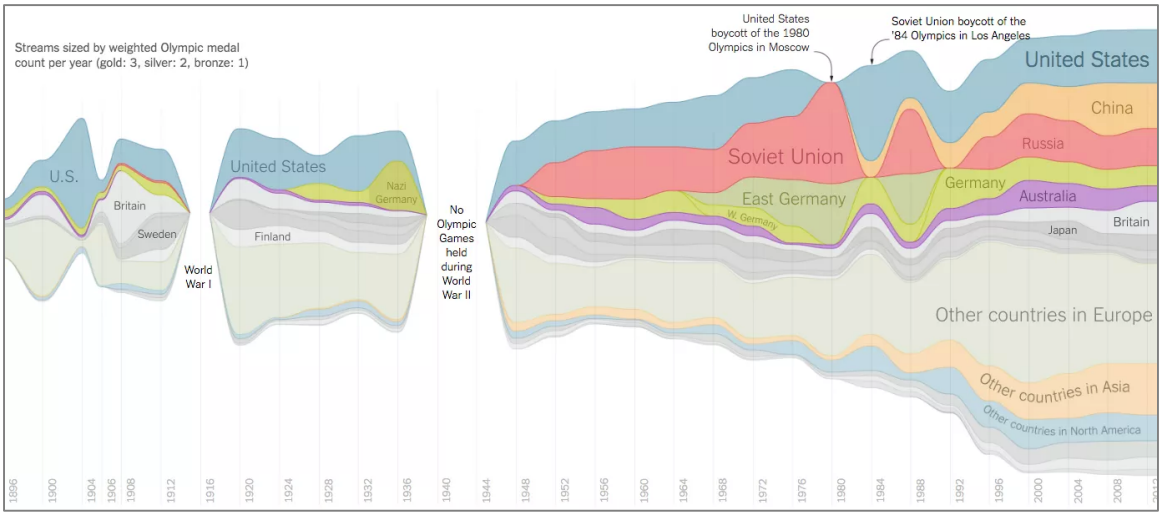

Figure 7 An Example of Trends in Twitter [33]

To efficiently describe trends - such as shooter's age or victims count over time - we hope to take advantage of several time-based chart visualizations. Figure 7 allows us to represent numerical data of several events as stacked areas over time.

6 User Manual

In order to help non-CS users to get familiar with our project, we designed and implemented a friendly interface – Global Events Viewer. Currently, the main components include word cloud and trend analysis. For each part, we also created some filters and expanders for users to do further analysis. With the help of the interface, users are able to get more details or trends from multiple collections. More work will be done in the future, which is discussed in Section 8.

Our current implementation of the Global Events Viewer provides full support for term frequency visualization, effectively displaying words classifying a particular collection. Frequency of the words is encoded in the size of words; color doesn't bear any meaning at this point. The term cloud also features temporal filtering. Users can specify a range describing events over years to get an aggregate of words identifying documents pertaining to events from that period. In addition, once rendered, a list of events gets displayed right next to the main visualization, allowing the user to unselect (everything is selected by default) shootings which he doesn't care

for. The intuition is to be able to disable specific events and only research correlations on shootings of interest. The main screen also features an input field specifying term count - how many terms should be fetched from the backend - allowing identification on more precise or coarser granularity.

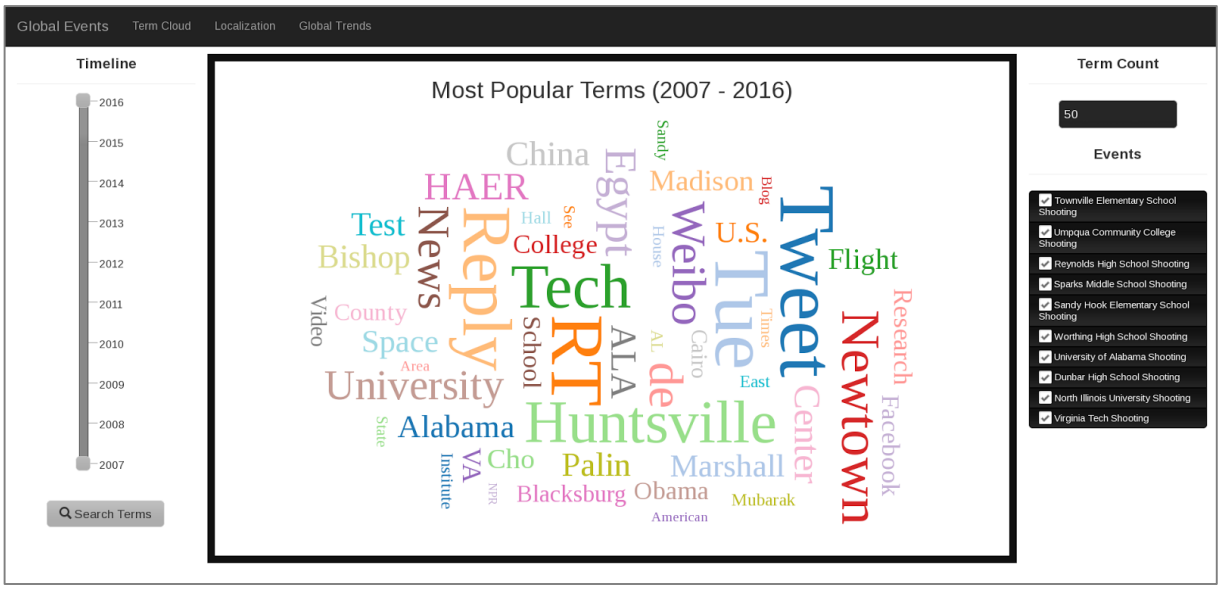

Figure 8 Global Events Viewer - Default Term Cloud

The term cloud screen also enables interaction with specific entities. When hovering over a particular entity with a mouse, its count would be displayed. It is also important to note that the selected size scale of entities is logarithmic – we found out that the disproportions among entity counts would prevent the user from seeing certain terms if a linear scale were to be used.

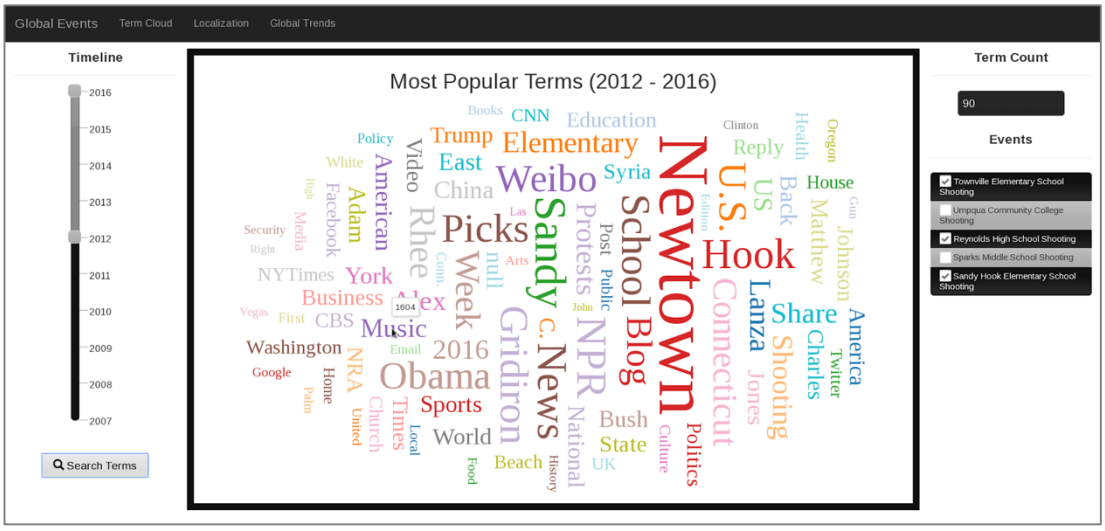

Figure 9 Global Events Viewer - Filtered Term Cloud

Clicking on a specific word takes the user to a new screen - term mentions. Here the user is able to see, for his particular timeline+event selection, which URLs have referenced selected term. The URLs are grouped by their events to provide smoother interaction.

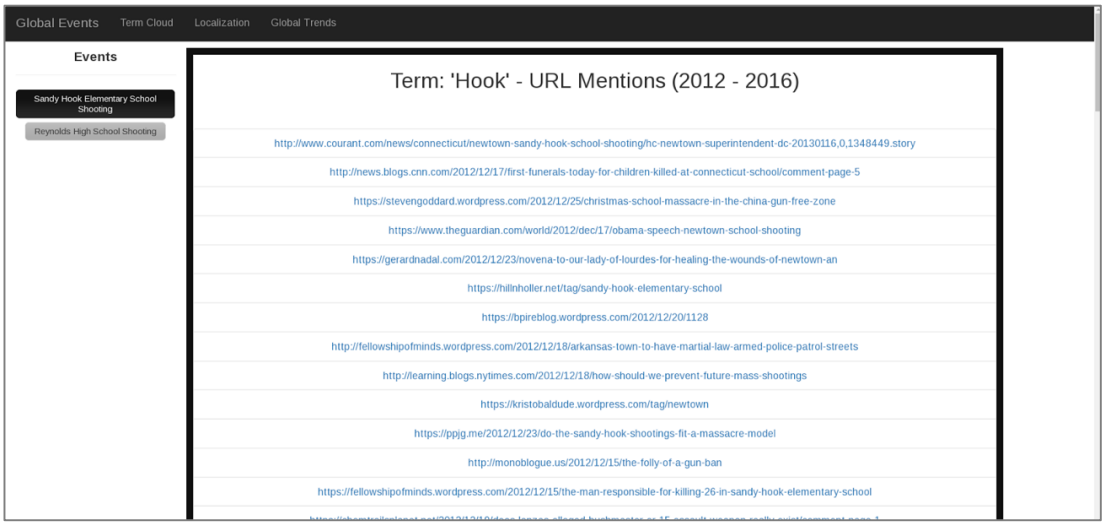

Figure 10 Global Events Viewer - URL Mentions

The last visualization that the current implementation of Global Events Viewer contains is labeled Global Trends. On this screen, the user can observe the evolution of common features over time - e.g., time series. Our original idea was to visualize trends using ThemeRiver, which might look nicer for multiple events in the same year but given the fact we only had a 1-event-1-year mapping, we settled for simpler, yet more descriptive dynamic line charts. As such, not only can the user observe the evolution of the y value but also the distance between x values, to get an idea how close specific events were to each other, over time.

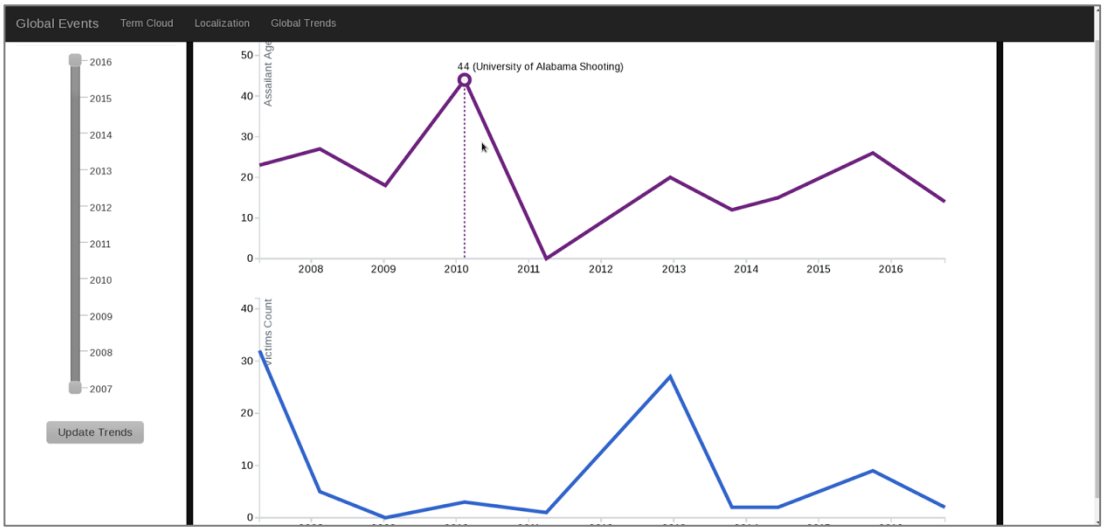

Figure 11 Global Events Viewer - Full-range Trends

Much like in the case of the term cloud, the user can once again limit the time interval of events to observe the evolution of trends on much finer granularity.

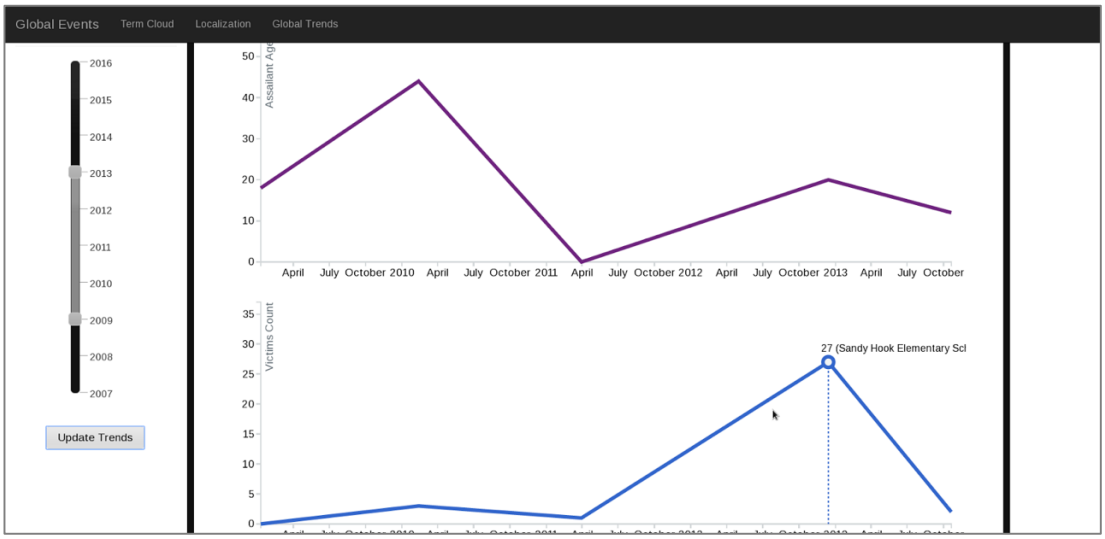

Figure 12 Global Events Viewer - Filtered Trends

7 Developer Manual

Our code has been packaged into GlobalEvents_Code.zip, which can be downloaded and unzipped for further use. Table 5 shows the key files in the code inventory.

Table 5 Key Files in Code Inventory

Folder	Sub-folder	File Path	Description
GlobalEventsCode	Data Collection	/FocusedCrawler.py	Main
		/crawler.py	Crawl Webpages
		/eventModel.py	Model Generation
		/VSM_Centroid.py	Vector Space Model
	Data Processing	/src/main/scala-2.12/globalevent.scala	Main
	Data Visualization	/settings.gradle	Configuration
		/build.gradle	Main

7.1 Internet Archive Tool

The following shows the steps to download a specific data collection from IA:
1- Download the Internet Archive tool (https://github.com/jjjake/internetarchive)
2- curl -LO https://archive.org/download/ia-pex/ia && chmod +x ia

3- Configure the IA tool (you will be asked for the credentials)
>> ./ia configure
4- Create a data collection directory (i.e., Destination Directory)
>> mkdir Collection_XYZ
5- Run the IA tool to download that target collection
>> ia download --search 'collection:Collection_ID" --destdir Collection_XYZ
Example: ia download --search 'ArchiveIt-Collection-2950' --destdir /home/harb/Collection_2950

7.2 Tutorials for Deploying EFC

The Event Focused Crawler needs very high quality URL seeds to train and build the topic model. This seeds list is usually curated manually to ensure relevance and quality relative to the topic of interest. Then you pass a list of the URLs that will be used for the crawling (including the high-quality seed URLs). The focused crawler requires few libraries to be installed. Below are the steps towards setting up the proper environment for the focused crawler.

7.2.1 Install Dependencies

EFC requires multiple dependencies. Follow the steps below to install all the relevant dependencies.

```
# You may want to update your "pip" ==> pip install --upgrade pip
sudo pip install requests_cache
sudo pip install bs4
sudo pip install nltk
sudo pip install python-dateutil    # Extensions to the standard Python datetime module
sudo pip install pytz
sudo pip install ner

# Download NLTK data
sudo python -m nltk.downloader -d /usr/local/share/nltk_data all
```

7.2.2 Run EFC

EFC includes two modes, which are Baseline Mode and Event Mode. After installing the above dependencies, developers could choose one of the two commands below and run EFC with different modes.

```
# Baseline Mode
python ./FocusedCrawler.py b input/<high_quality_seed_input_file> <URLs_input_file>
```

```
# Event Mode
# Start NER Server
java -mx1000m -cp stanford-ner.jar edu.stanford.nlp.ie.NERServer –loadClassifier
classifiers/english.muc.7class.distsim.crf.ser.gz -port 8000 -outputFormat inlineXML
python ./FocusedCrawler.py b input/<high_quality_seed_input_file> <URLs_input_file>
```

7.3 Tutorials for Deploying ArchiveSpark in Jupyter

ArchiveSpark is a bit sensitive to the versions of its dependencies. After multiple tests, we finally deployed it successfully on a stand-alone machine. As a result, for the following subsections, the developers should be very careful about those versions. However, if there is an “Out of Memory” error while running Scala scripts (e.g., load, filter, enrich) with Jupyter notebook, developers need to uncomment the spark.driver.memory in spark-defaults.conf and set a proper value.

```
# Default system properties included when running spark-submit.
# This is useful for setting default environmental settings.

# Example:
#spark.master                     spark://master:7077
#spark.eventLog.enabled           true
#spark.eventLog.dir               hdfs://namenode:8021/directory
#spark.serializer                 org.apache.spark.serializer.KryoSerializer
spark.driver.memory              4g
#spark.executor.extraJavaOptions  -XX:+PrintGCDetails -Dkey=value -Dnumbers="one
 two three"
```

Because it is a bit complicated, we suggest the developers run the scripts through the Spark terminal or use other IDEs like Eclipse or IntelliJ.

7.3.1 Install JDK 8

First, JDK 8 should be installed in the machine. The default JAVA version in our machine is 1.7.0; we need to run the following commands to upgrade it into 1.8.0.

```
sudo yum -y update
sudo yum list *jdk*
sudo yum install java-1.8.0-openjdk
```

7.3.2 Install Python 3.5 and Pip

We suggest the developers install Python 3.5 for convenience. After installing Pip, it is easy to install other packages or libraries. The commands for the installation are shown as below.

```
sudo yum -y groupinstall "Development tools"
sudo yum -y install zlib-devel bzip2-devel openssl-devel ncurses-devel sqlite-devel readline-
devel tk-devel gdbm-devel db4-devel libpcap-devel xz-devel

wget https://www.python.org/ftp/python/3.5.2/Python-3.5.2.tgz
tar xvfz Python-3.5.*.tgz
cd Python-3.5.*
./configure --prefix=/usr/local --enable-shared LDFLAGS="-Wl,-rpath /usr/local/lib"
su -
make && make altinstall
ln -s /usr/local/bin/python3.5 /usr/bin/python3.5
```

```
exit

wget https://bootstrap.pypa.io/get-pip.py
sudo python3.5 get-pip.py
sudo ln -s /usr/local/bin/pip /usr/bin/pip
```

7.3.3 Install Jupyter

Follow the easy step to install Jupyter. Then, run the command “jupyter notebook” and type “http://localhost:8888” to test the status of the Jupyter notebook.

```
sudo pip install jupyter
jupyter notebook
```

For a remote connection, the remote user needs to input a specific token number to get the access. Here, our team suggests the developers modify the configuration file of Jupyter to skip the token access.

```
jupyter notebook --generate-config
```

7.3.4 Install Spark 2.1.0

Please read the official documents of Spark and know more about it. Then, run the following commands to install Spark 2.1.0 on the machine.

1. Download and extract the Spark file

```
spark-2.1.0-bin-hadoop2.7.tgz
tar -xzf spark-2.1.0-bin-hadoop2.7.tgz
```

2. Move Spark software files

```
sudo mv spark-2.1.0-bin-hadoop2.7 /usr/local/share/spark
```

3. Set PATH for Spark

```
export SPARK_HOME=/usr/local/share/spark
export PATH=$PATH:$SPARK_HOME/bin

vi /etc/hosts
IP_ADDRESS archivespark.dlrl
```

4. Run Spark

```
$spark-shell
...
Welcome to
```

```
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.0.0
      /_/

Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.7.0_99)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```

7.3.5 Install ArchiveSpark

1. Create a kernels directory if it does not exist yet

```
mkdir -p ~/.ipython/kernels
```

2. Download and unpack the ArchiveSpark/Toree to your kernels directory

```
tar -zxf archivespark-kernel.tar.gz -C ~/.ipython/kernels
```

3. Edit the kernel configuration file to customize it according to your environment. For example, replace USERNAME in line 5 after "argv" with your local username, set SPARK_HOME to the path of your Spark 2.1.0 installation, and change HADOOP_CONF_DIR/SPARK_OPTS if needed.

```
vim ~/.ipython/kernels/archivespark/kernel.json
```

4. Run Jupyter to test the kernel

```
jupyter notebook
```

Here, it is likely to get an error, which is shown below. The reason is the current version of Scala is incompatible with ArchiveSpark.

```
Kernel version: 0.1.0.dev6-incubating-SNAPSHOT
Scala version: 2.10.4
Exception in thread "main" java.lang.NoSuchMethodError:
scala.collection.immutable.HashSet$.empty()Lscala/collection/immutable/HashSet;
```

7.3.6 Replace the Original Scala

Here, we need to build a correct version of Scala to replace the old one. To achieve this goal, Docker and sbt should be installed to do the compiling job.

1. Install Docker 1.7.1

```
sudo rpm -iUvh http://dl.fedoraproject.org/pub/epel/6/x86_64/epel-release-6-8.noarch.rpm
sudo yum update -y
sudo yum -y install docker-io
sudo service docker start
```

2. Install sbt

```
sudo curl https://bintray.com/sbt/rpm/rpm | sudo tee /etc/yum.repos.d/bintray-sbt-rpm.repo
sudo yum install sbt
```

3. Download and extract incubator-toree

4. Build the correct version and replace the old one

```
cd
sudo make dev
sudo cp target/scala-2.11/toree-assembly-0.2.0.dev1-incubating-SNAPSHOT.jar
~/.ipython/kernels/archivespark/toree/lib/toree-assembly-0.1.0.dev6-incubating-SNAPSHOT.jar
```

After completing the above steps, developers should be able to run ArchiveSpark with Jupyter. We will add more details for the installation by using other IDEs later.

7.4 Tutorials for Deploying ArchiveSpark in IntelliJ

It is easy to deploy ArchiveSpark in IntelliJ. Before that, we need to install JDK 8, and then install Spark and Scala plugins in IntelliJ. For installing JDK 8, please follow the steps in Section 7.3.1.

7.4.1 Install Spark and Scala

Developers can read the installation document by clicking the following link:
https://www.ibm.com/developerworks/community/files/basic/anonymous/api/library/e5c0146d-f723-446b-9151-c31d4c56ed01/document/b41505ac-141b-45a2-84cd-1b6a8d5ae653/media

After that, you are able to create a Scala project in IntelliJ. Figure 13 shows the HelloWorld example in IntelliJ.

7.4.2 Deploy ArchiveSpark

In IntelliJ, it is easy to repeat the above steps and import two ArchiveSpark jar files into the current project.

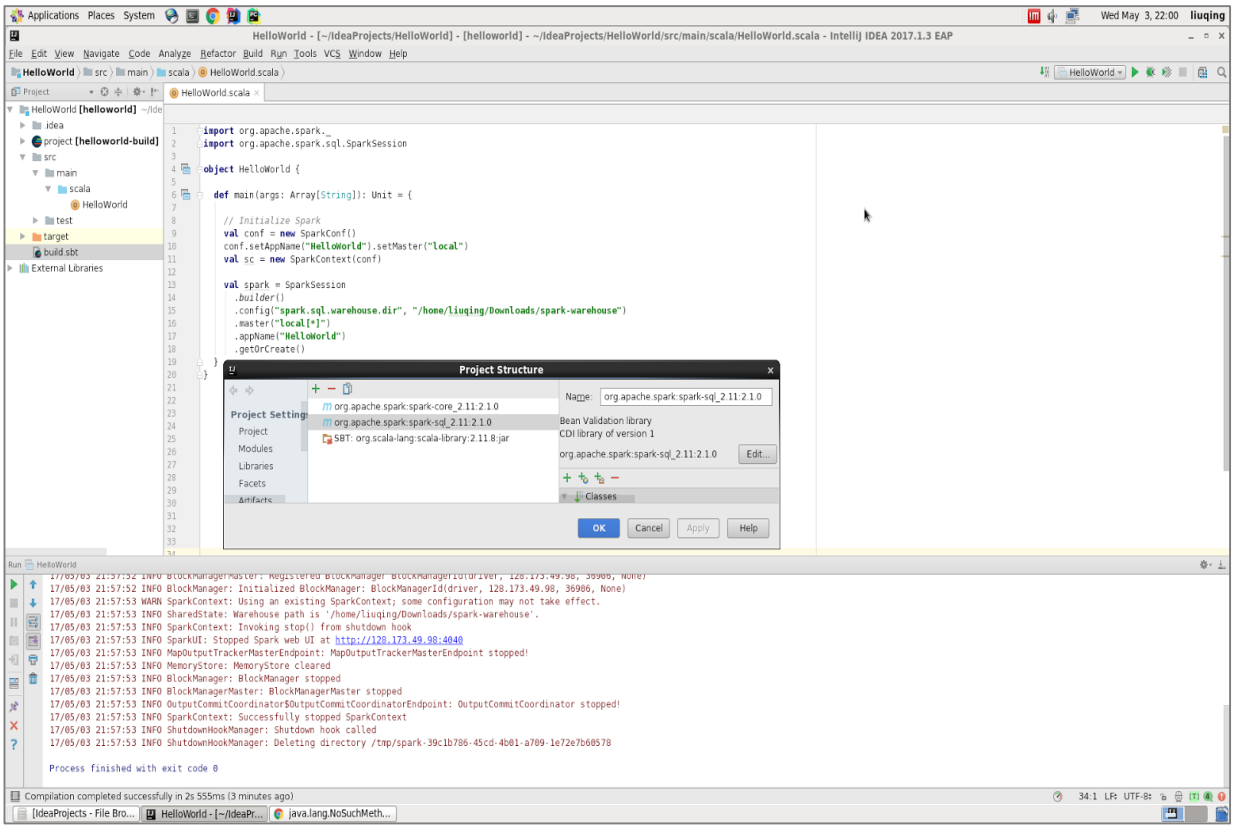

Figure 13 Run Helloworld.scala in IntelliJ

7.5 Tutorials for Building CDX Files

In Section 7.2, we can crawl a huge number of WARC files. However, ArchiveSpark needs both WARC files and CDX files as the input. Therefore, we made use of CDX-Writer, a Python script to create CDX index files of WARC data, to generate the CDX files. Please notice that CDX-Writer can only work with Python 2.7.

7.5.1 Install CDX-Writer

This script is not properly packaged and cannot be installed via pip. Run the following commands to install CDX-Writer.

```
pip install git+git://github.com/rajbot/surt#egg=surt
pip install tldextract==1.0 --use-mirrors
pip install chardet --use-mirrors
```

7.5.2 Link a Third-party WARC Tool

Before running the kernel script, we need to link CDX-Writer with a third-party WARC tool in order to apply all its functionalities.

```
cd tests
wget https://bitbucket.org/rajbot/warc-tools/get/e8266e15f7b6.zip
unzip e8266e15f7b6.zip
ln -s rajbot-warc-tools-e8266e15f7b6/hanzo/warctools .
```

7.5.3 Generate CDX File

Run the command below to create a CDX file.

```
python2.7 cdx_writer.py [Input – WARC file] [Output – CDX file]
```

7.6 Tutorials for Data Processing

Data processing is the key component in our project. The following sections show the main stages of our work. Developers can find detailed steps in globalevent.scala in GlobalEvents_Code.zip.

7.6.1 Preparation

Import Dependencies. Multiple libraries need to be imported into the project, including the Java, Scala, and Spark libraries. The most important libraries, which are Stanford NLP and ArchiveSpark, should also be imported.

```
// Java, Scala, Spark Libraries
import java.io.PrintWriter
import java.util.Properties
import collection.mutable._

import scala.util.Try
import scala.collection.JavaConverters._
import scala.collection.immutable.ListMap

import org.apache.spark._
import org.apache.spark.sql.DataFrame
import org.apache.spark.sql.SparkSession

// Stanford NLP Libraries
import edu.stanford.nlp.ling.CoreAnnotations.{NamedEntityTagAnnotation,
SentencesAnnotation, TextAnnotation, TokensAnnotation}
import edu.stanford.nlp.ling.CoreLabel
import edu.stanford.nlp.pipeline.{Annotation, StanfordCoreNLP}
import edu.stanford.nlp.util.CoreMap

// ArchiveSpark Libraries
import de.l3s.archivespark._
import de.l3s.archivespark.implicits._
import de.l3s.archivespark.enrich.functions._
import de.l3s.archivespark.specific.warc.specs._
```

Global Variables. We use HashMap structure to store the entities, their frequencies, and URL lists.

```
val shooter_list = new ListBuffer[String]()
val shooter_count = new HashMap[String, Int]
val shooter_link = new HashMap[String, Set[String]] with MultiMap[String, String]

val weapon_list = new ListBuffer[String]()
val weapon_count = new HashMap[String, Int]
val weapon_link = new HashMap[String, Set[String]] with MultiMap[String, String]

val age_list = new ListBuffer[String]()
val age_count = new HashMap[String, Int]
val age_link = new HashMap[String, Set[String]] with MultiMap[String, String]

val victim_list = new ListBuffer[String]()
val victim_count = new HashMap[String, Int]
val victim_link = new HashMap[String, Set[String]] with MultiMap[String, String]

val date_list = new ListBuffer[String]()
val date_count = new HashMap[String,Int]
val date_link = new HashMap[String, Set[String]] with MultiMap[String, String]
```

Import Stanford NER. With the help of Stanford NER, we can easily get the entities and split them into persons, locations, organizations, and dates. We tried different types of classifiers and finally created an integrated model with three different classifiers. Furthermore, by setting the combination mode as HIGH_RECALL, we are able to get better results.

```
val props = new Properties()
props.setProperty("annotators", "tokenize, ssplit, pos, lemma, ner")
props.setProperty("ner.model","/home/liuqing/Downloads/stanford-ner-2016-10-
31/classifiers/english.muc.7class.distsim.crf.ser.gz,/home/liuqing/Downloads/stanford-ner-2016-
10-
31/classifiers/english.all.3class.caseless.distsim.crf.ser.gz,/home/liuqing/Downloads/stanford-
ner-2016-10-31/classifiers/english.conll.4class.distsim.crf.ser.gz")
props.setProperty("ner.useSUTime", "false")
props.setProperty("ner.combinationMode", "HIGH_RECALL")
props.setProperty("serializeTo", "/home/liuqing/Downloads/stanford-ner-2016-10-
31/classifiers/example.serialized.ncc.ncc.ser.gz")

val pipeline = new StanfordCoreNLP(props)
```

7.6.2 Import WARC Files

Read WARC and CDX Files into RDD. ArchiveSpark makes use of two types of input files, which are WARC and CDX files.

```
val root_path = "rawdata/"
val cdx_file_path = collection_name + ".cdx"
val warc_file_path = collection_name + ".warc.gz"
val rdd = ArchiveSpark.load(sc, WarcCdxHdfsSpec(root_path + cdx_file_path, root_path + warc_file_path))
```

Basic Parsing. We leverage the basic parsing method to extract the event name and date. After that, a hash function is used to calculate a unique code of event name, which follows the event date to generate the row key for HBase.

```
// Extract Event Name
val event_name = collection_name.split("_").head.replace("-", " ")

// Extract Event Date
val event_date = collection_name.split("_").tail(0)

// Generate HBase Row_Key
val row_key = event_date + (event_name.hashCode & 0xFFFFFFF).toString
```

Filter out Webpages. ArchiveSpark provides various functions to handle the WARC files.

```
// Filter out Webpages
val rdd_raw_webpages = rdd.filter(r => r.status == 200 && r.mime == "text/html")

// Remove Duplicated Webpages
val rdd_webpages = rdd_raw_webpages.distinctValue(_.originalUrl) {(a, b) => a}

// Total Number of Webpages
val webpage_count = rdd_webpages.count.toInt

// Extract Webpage Title into RDD
val title = HtmlText.of(Html.first("title"))
val rdd_titles = rdd_webpages.enrich(title)

// Extract Webpage Body into RDD
val body = HtmlText.of(Html.first("body"))
val rdd_body = rdd_webpages.enrich(body)
```

7.6.3 Custom Functions

We created multiple custom functions to help process the data, including isAllDigits, countSubstring, hasColumn, and similarity functions. Two more help functions have also been used to store the entities into HashMap and sort them by the tf-df score. These custom functions can be found in globalevent.scala in GlobalEvents_Code.zip.

7.6.4 Data Processing

Webpage Cleaning. After extracting the raw text from the webpages, we need to clean the data to improve our results.

```
// Extract original url from webpage head
val df_url = r.originalUrl
// Extract raw text from webpage body
val df_body = df.select("payload.string.html.body.text")
// Remove jquery & java scripts
val df_body_clean_1 = df_body.first().toString().replaceAll("\\{.*\\}", "")
// Remove tags
val df_body_clean_2 = df_body_clean_1.replaceAll("\\<.*\\>", "")
// Remove markers
val df_body_clean_3 = df_body_clean_2.replaceAll("[+*,|]", " ")
// Extract specific patterns from webpage body with stopwords
val df_body_rich = df_body_clean_3.split(" ").filter(x => x.matches("[A-Za-z0-9\\.\\-]+")).toList

// Get stopwords from file
val stopWords = sc.textFile("rawdata/stopwords_en.txt")
val stopWordSet = stopWords.collect.toSet
```

Entities Extraction. The entities can be extracted from the webpage payload by using multiple approaches. Here, we show the entities by leveraging Stanford NER.

```
// Stanford NLP process
  val document = new Annotation(df_body_basic.mkString(" "))
  pipeline.annotate(document)
  val sentences = document.get(classOf[SentencesAnnotation]).asScala.toList

  // Generate name entities
  val nlp_tokens = for {
    sentence: CoreMap <- sentences
    token: CoreLabel <- sentence.get(classOf[TokensAnnotation]).asScala.toList
    word: String = token.get(classOf[TextAnnotation])
    ner: String = token.get(classOf[NamedEntityTagAnnotation])
  } yield (token, word, ner)

  insert_to_map_basic(nlp_tokens, "ALL", df_body_basic, df_url, sner_all_count, sner_all_link)
  insert_to_map_basic(nlp_tokens, "PERSON", df_body_basic, df_url, sner_person_count,
sner_person_link)
  insert_to_map_basic(nlp_tokens, "ORGANIZATION", df_body_basic, df_url, sner_org_count,
sner_org_link)
  insert_to_map_basic(nlp_tokens, "LOCATION", df_body_basic, df_url, sner_loc_count,
sner_loc_link)
```

Sort and print results. We leverage the custom function to sort the entities and then print them out.

```
val sner_all_sort = entity_sort(sner_all_count, sner_all_link)
val sner_person_sort = entity_sort(sner_person_count, sner_person_link)
val sner_org_sort = entity_sort(sner_org_count, sner_org_link)
val sner_loc_sort = entity_sort(sner_loc_count, sner_loc_link)

val date_sort = entity_sort(date_count, date_link)
val shooter_sort = entity_sort(shooter_count, shooter_link)
val age_sort = entity_sort(age_count, age_link)
val weapon_sort = entity_sort(weapon_count, weapon_link)
val victim_sort = entity_sort(victim_count, victim_link)
```

7.7 Global Events Viewer

Our entire visualization effort is captured in a Java project we named Global Events Viewer. This is an open-source standalone application that leverages the Spring Boot framework [26] to provide seamless, powerful UI powering our trend visualization. The project is versioned on GitHub and available for download behind the following URL:

https://github.com/dedocibula/global-events-viewer

Internally, to make sure that the backend can be deployed in the ecosystem of our cluster the Global Events Viewer heavily relies on the Gradle build system. This arrangement, together with startup scripts, allows the user to quickly download all the necessary dependencies and build the project regardless of underlying environment or access control restrictions imposed by administrators. The choice of Spring Boot was mostly influenced by the simplicity of its configuration as well as by the fact that it doesn't rely on any external Web server. Flexibility of Spring Boot allows the application to essentially bootstrap itself to an embedded application container (most of the time Tomcat [27] although Jetty [28] is also an option).

The data originating from WARC archive preprocessing is stored in the HBase database. As such the Global Events Viewer takes advantage of the default Apache HBase client to access the information. In order not to constrain the Global Events Viewer to one particular database we abstract the database logic using the Data Access Object (DAO) pattern [29]. This way, each db-specific code simply implements a contract, which the remainder of the application relies on. Further, HBase configuration is extracted in an accompanying property file. As such, different HBase configurations can be supplied, allowing users to deploy the app in various environments with different HBase setup (e.g., local 1 instance HBase, clustered multi-node HBase).

As already mentioned in our requirements section (Section 3) the bulk of the project constitutes visualizations - i.e., frontend leveraging JavaScript client libraries. In the scope of our project we primarily rely on 2 - jQuery [30] and D3.js [16]. jQuery is primarily used to retrieve the data from the backend using AJAX requests. D3.js powers all the visual elements via the use of mentioned plugins.

8 Further Discussion

Based on the requirements of our project, each team member built an individual module for each stage and shared the inputs and outputs. We will establish an integrated pipeline to process the data automatically soon, from the beginning to the end.

For the data collection, currently we discard those broken links. The Wayback Machine can be used to retrieve those webpages and we will be able to add those resources into EFC for further process. In addition, we would like to consider events that happened in the past (e.g., 1995) and those that occurred outside the United States. Finally, we may want to look to extend our data archives to include other data sources such as tweets (not only webpages).

At present, we have deployed ArchiveSpark in a stand-alone machine due to the version conflict of Spark. The version of Spark for running ArchiveSpark is 1.6.0 or 2.1.0. Unfortunately, the Spark version is 1.5.0 in our Hadoop Cluster. Therefore, we need to upgrade the cluster and then deploy our framework to process big collections.

For the Stanford NER, we have found the results seem not so good while processing the parsing tree. For instance, Virginia can be considered as both an organization (part) and a location, which can affect our final results. We will improve the current methods to increase the accuracy of those entities. One possible solution is to use its context with a sliding window. For instance, if Virginia is close to Tech, it should be considered as an organization. If Virginia comes after Blacksburg, it ought to be a location.

For our visualization project, we want to make sure to also incorporate event localization as described in Section 5. Unfortunately, due to the project time constraints as well as the limited man-power we simply didn't find enough time to add the integration with the Google Geocoding API [31] to our data processing, and as this part constitutes a prerequisite for the Global Events Viewer's Localization visualization, the corresponding code was also omitted.

In addition, we also want to make sure to enrich our Global Trends visualization with the evolution of weapons used in the shootings. As this feature cannot be represented as numerical data it is impossible for us to simply reuse our existing line charts. A dynamic (time-evolving) pie chart would be a much better way of representing this categorical data.

9 References

[1] Farag, M. et al., "Focused Crawler for Events", International Journal on Digital Libraries (IJDL'17), DOI: 10.1007/s00799-016-0207-1, 2017. http://hdl.handle.net/10919/73035.
[2] Gossen, G. et al., "iCrawl: Improving the Freshness of Web Collections by Integrating Social Web and Focused Web Crawling", Proceedings of the 15th ACM/IEEE-CS Joint Conference on Digital Libraries, Pages 75-84, Knoxville, Tennessee, USA, June, 2015.
[3] Meusel, F. et al., "Focused Crawling for Structured Data", Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14), Pages 1039-1048, Shanghai, China, November, 2014.
[4] Du Qianzhou, Zhang Xuan, Named Entity Recognition for IDEAL, Virginia Tech, Blacksburg. May 10, 2015. http://hdl.handle.net/10919/52254.
[5] Wang Xiangwen, Chandrasekar Prashant, Reducing Noise for IDEAL, Virginia Tech, Blacksburg. May 12, 2015. http://hdl.handle.net/10919/52340.
[6] Thumma Sujit Reddy, Kalidas Rubasri, Torkey Hanaa, Document Clustering for IDEAL, Virginia Tech, Blacksburg. May 13, 2015. http://hdl.handle.net/10919/52341.
[7] Edward A Fox, Kristine Hanna, Andrea L Kavanaugh, Steven D Sheetz, Donald J Shoemaker, III: Small: Integrated Digital Event Archiving and Library (IDEAL), NSF grant IIS - 1319578, 2013-2016. http://eventsarchive.org
[8] Edward A Fox, Donald Shoemaker, Chandan Reddy, Andrea Kavanaugh, III: Small: Collaborative Research: Global Event and Trend Archive Research (GETAR), NSF grant IIS - 1619028, 2017-2019. http://eventsarchive.org
[9] Steven Bird, Ewan Klein, and Edward Loper, Natural language processing with Python. O'Reilly Media, 2009.
[10] NLTK project, NLTK 3.0 documentation. http://www.nltk.org, accessed on 03/02/2017.
[11] Leonard Richardson, Beautiful Soup Documentation. https://www.crummy.com/software/BeautifulSoup/, accessed on 03/02/2017.
[12] Rakesh Agrawal and Ramakrishnan Srikant, Fast algorithms for mining association rules in large databases. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September 1994.
[13] Zaki, M. J. Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering. 12 (3): 372–390, 2000.
[14] Han, Mining Frequent Patterns Without Candidate Generation, Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data. SIGMOD '00: 1–12, 2000.
[15] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-Local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), 363-370
[16] D3 - Data-Driven Documents, https://d3js.org. Accessed on March 2, 2017.
[17] Lee Byron and Martin Wattenberg. "Stacked Graphs – Geometry & Aesthetics". IEEE Transactions on Visualization and Computer Graphics, Vol. 14, Issue: 6, 2008.
[18] Gradle Build Tool, https://gradle.org/. Gradle, Inc, CA. Accessed on March 2, 2017.
[19] Mark Otto, Jacob Thornton et al., Bootstrap, http://getbootstrap.com/. Accessed on March 2, 2017.

[20] Jason Davies, https://github.com/jasondavies/d3-cloud, London, UK. Accessed on March 2, 2017.
[21] Mark DiMarco, https://github.com/markmarkoh/datamaps, Austin, TX. Accessed on March 2, 2017.
[22] Helge Holzmann, Vinay Goel, Avishek Anand. "ArchiveSpark: Efficient Web Archive Access, Extraction and Derivation". Proceedings of the 16th ACM/IEEE-CS on Joint Conference on Digital Libraries, pages 83-92, 2016.
[23] Grotke, A, Web Archiving at the Library of Congress, Computers in Libraries, v.31 n.10, pp. 15–19. Information Today, 2011.
[24] L3S Research Center, https://www.l3s.de/home, Hannover, Germany. Accessed on March 2, 2017.
[25] Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky, The Stanford CoreNLP Natural Language Processing Toolkit, Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60, 2014.
[26] Phillip Webb, Dave Syer et al., Spring Boot Reference Guide, http://docs.spring.io/spring-boot/docs/current-SNAPSHOT/reference/htmlsingle/. Accessed on March 31, 2017.
[27] Apache Tomcat, http://tomcat.apache.org/. Accessed on March 31, 2017.
[28] Eclipse Jetty, http://www.eclipse.org/jetty/, The Eclipse Foundation, Ottawa, Canada. Accessed on March 31, 2017.
[29] Hodgson Kyle, Reid Darren, ServiceStack 4 Cookbook, Packt Publishing Ltd, ISBN 9781783986576, January, 2015.
[30] jQuery API, http://api.jquery.com/, The jQuery Foundation. Accessed on March 31, 2017.
[31] Google Geocoding API, https://developers.google.com/maps/documentation/geocoding/start. Accessed on May 3, 2017.
[32] Wappalyzer, https://wappalyzer.com/applications/d3. Accessed on May 3, 2017.
[33] Gregor Aisch and Larry Buchanan, A Visual History of Which Countries Have Dominated the Summer Olympics, New York times, 2016. https://www.nytimes.com/interactive/2016/08/08/sports/olympics/history-olympic-dominance-charts.html?_r=0. Accessed on May 3, 2017.