CS 5604: Informational Storage & Retrieval

Spring 2015

Social Networks & Importance

05/06/2015

Instructor: Dr. Edward A. Fox

Team Members

Bhaskara Srinivasa Bharadwaj Bulusu	bbsb08@vt.edu
Vanessa Cedeno	vcedeno@vt.edu
Islam harb	iharb@vt.edu
Yilong Jin	jin28@vt.edu
Sai Ravi Kiran Mallampati	sairavi5@vt.edu

Virginia Polytechnic and State University
Blacksburg, VA

Table of Contents

List of Figures:

List of Tables:

Acknowledgments

- We would like to thank Dr. Edward A. Fox (instructor of the course) for guiding us through the development of this project. In addition, we would also like to thank the US National Science Foundation for support of grant IIS - 1319578.

- We would also like to thank Mohamed Magdy and Sunshin Lee for their unconditional support.

- We would also like to thank the Reducing Noise team for providing the cleaned web pages and the Hadoop team in helping out with the HBase schema designed for our team.

- We would also like to thank other teams for providing valuable comments to every report that we submitted, which only made us work better in achieving the goals of the project.

1. Abstract

The IDEAL (Integrated Digital Event Archiving and Library) project involves VT faculty, staff, and students, along with collaborators around the world, in archiving important events and integrating the digital library and archiving approaches to support R&D related to important events. An objective of the CS5604 (Information Retrieval), Spring 2015 course, was to build a state-of-the-art information retrieval system, in support of the IDEAL project. Students were divided into eight groups to become experts in a specific theme of high importance in the development of the tool. The identified themes were Classifying Types, Extraction & Feature Selection, Clustering, Hadoop, LDA, NER, Reducing Noise, Social Networks & Importance and Solr & Lucene.

Our goal as a class was to provide documents that were relevant to an arbitrary user query from within a collection of tweets and their referenced web pages. The goal of the Social Network and Importance group was to develop a query independent importance methodology for these tweets and web pages based on social network type considerations.

This report proposes a method to provide importance to the tweets and web pages by using non-content features. We define two features for the ranking, Twitter specific features and Account authority features. These features include Favorite count, Retweet count, list count and the number of followers. To determine the best set of features, the analysis of their individual effect in the output importance is also included. At the end, an "importance" value is associated with each document, to aid searching and browsing using Solr.

For computing importance of web pages too, we do not consider content. Instead, we consider the links from/to web pages. The PageRank algorithm is used to compute the importance values of the web pages.

GraphX is used to compute the importance values of web pages using the PageRank algorithm.

The importance values for tweets and web pages only from the Police department data collection are available. The other data collections do not have non-content information such as Followers count, Retweet count, list count and the Favorite count. Thus, the importance values could not be computed.

Tweets with high importance are from the users who have high follower counts. The top 10 tweets are from CNN Breaking News.

2. Project Overview

2.1. Introduction

Twitter is an online social networking service that allows its users to share any type of information in messages of 140 characters or less called “tweets”. These messages may include news, conversations, pointless babble, advertisement or spam. The user has various options to format a tweet. The message, besides words, could include links to web pages. In the same way, “hashtags” to group posts together by a topic, including the character “#” before a word. If the character “@” is included it means a user is being referenced. The user also has the option to retweet a message he is interested in to share with his followers or the world.

Tweets including the same webpage in their content might denote author perceived relevance. This can help us to determine the importance for an specific webpage.

Importance values to tweets and web pages are computed using the social network graph without taking any type of content similarity features into consideration. The social network graph is constructed using some of the tweet features such as retweets, favorites, followers etc., and is independent of the user query. The influence of a tweet is computed by using these features.

The web pages that the tweets refer to are considered in this project. The short URLs in the tweets are first expanded. The web pages are then crawled using Nutch. As mentioned earlier, the content of web pages is not considered. The importance of a web page is computed the number of web pages that refer to it.

2.2. Literature Review

We received a few papers suggested by Dr. Naren Ramakrishnan and Xuan Zhang from the NER team.

The increasing popularity and the rapid emergence of Twitter and other microblogs makes improved trustworthiness and relevance assessment of microblogs very important [5]. As the content-based relevance is out of our scope, so we will consider only the popularity and the trustworthiness in our calculations. This metric “Trustworthiness and popularity” will be reflected in the “User Importance Value (UIV)” that is discussed later in the upcoming sections.

Duan et al. [6] proposed a tweet ranking strategy with a learning to rank algorithm. They take into consideration content relevance features, Twitter specific features and account authority features. They define these as the top three effective features for tweet ranking. They found out that containing a URL is the most effective feature. The URLs shared in tweets provide more detailed information beyond the tweet’s 140 characters, and may be relevant to the query at a high probability. Another useful feature is the number of lists that the author of the tweet has been listed in. Features such as Hash tag Score and Retweet Count were not as effective as expected. They used Normalized Discount Cumulative Gain

(NDCG). This research is based on the search results returned from Twitter containing the input query. Also, they include relevance in the algorithm, which we will not do and they did not consider spam issues in the ranking process.

Three measures of influence in a large amount of data from Twitter were compared and presented: indegree, retweets, and mentions [19]. Retweets are driven by the content value of a tweet, while mentions are driven by the name value of the user. Popular users who have high indegree are not necessarily influential in terms of spawning retweets or mentions. Most influential users can hold significant influence over a variety of topics. Influence is not gained spontaneously or accidentally, but is gained through efforts such as tweeting in the same topic. In addition, it is also mentioned that there are two modes in Twitter, one of them based on information consumption and the other based on social ties. A hybrid structure of Twitter as both information network and social network seems to be plausible [4]. Potential influencers are identified to seed a word-of-mouth campaign so that marketers can have better impact on their businesses. The distribution of cascade sizes is approximately power law, which means a majority of posted URLs do not spread at all [21].

On-line page importance computation (OPIC) is implemented to find the importance of web pages connected by hyperlinks [7]. The World Wide Web is viewed as a directed graph G, where the web pages are vertices and links from one page to another are edges. The importance of a web page is defined in an inductive way and computed using a fixpoint. If a page is decided to be important then all the web pages that point to this page or pointed by this page are important. More formally, two values are associated with each node in the graph of web pages: cash and history. Initially, each node is assigned with a cash value (1/n) where n is the number of nodes in the graph. A node v is selected at random to start. The crawling along the directed graph is done by selecting the next page randomly. The cash value of each node stores the recent information discovered about the page and the history value of each node is the sum of the cash obtained by the page since the start of the algorithm.

Haewoon et al. [20] discuss about ranking different Twitter users by Page Rank and also by the retweets. They give a brief comparison among the rankings. They further discuss about the impact of retweets in Twitter and also give a temporal analysis of a retweet. Overall, they present a quantitative study of Twitter and the information present in it.

2.3. Project Requirements

We associate each tweet and web page with one or more measures of document importance. The tasks for the team were identified as follows:

i. Preparing a graph representation of the tweet-web page collection based on social network type considerations.
ii. Computing some measure to assign importance values to each of the web pages/tweets.
iii. Collaborating with the Solr team to associate each document with the importance measures to aid searching and browsing using Solr.

2.4. Proposed Solution

We describe a ranking function, that gives an “importance” to each tweet and web page in order to aid the Solr team with an efficient query search output.

- **Users:** Each user is associated with another user if it’s mentioned by it.
- **Tweets:** Each tweet is associated with another tweet if it’s retweeted. Each tweet is associated with a user if it’s tweeted by it.
- **Web pages:** Each web page is associated with a tweet if it’s mentioned by it. Each web page is associated with another webpage if it’s mentioned by it.

2.4.1. Non content features for computing importance

For the selection of the feature set, we describe:

- **Twitter specific features:** The characteristics of a tweet, like retweet count and favorites count.
- **Account authority features:** The features that represent the influence of authors of the tweets in Twitter, like followers count and listed count.

2.4.2 Tweet Specific Features

Retweet count: The number of times the tweet has been retweeted.

Favorites count: The number of times the tweet has been favorited.

2.4.3 Account authority features

Followers count: The number of followers a user has.

List score: The number of lists a user belongs to.

2.4.4 Social Network Graph

We construct a graph G(V, E). G is built from a tweet and web page collection. V represents the set of vertices that represents users (Circles), tweets (Squares) and web pages (Triangles). E represents edges between vertices. An edge between two users represents the “Mention (@)” action in the Twitter. For example, if “U1” mentions “U2” in a tweet or vice versa, then there will be an edge representing that “mention” in the graph and Figure 1 shows this. This user-to-user edge (U_Edge) is a weighted directed edge. The weight of the U_Edge represents the popularity of these users connected to that edge as shown in the Equation (1). An edge between a user and a tweet indicates that this user posted this tweet. An edge between a tweet and a web page indicates that this tweet includes the URL for this webpage. Figure 2 shows an example of a simplified graph that shows the relationships between all the different nodes as described above. The dashed line between tweet nodes tells that these two tweets share the same URL for the same webpage.

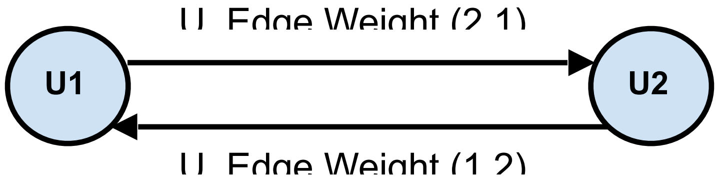


Figure 1: User nodes

U_Edge Weight(1,2) = (#Mentions of U1 made by U2) / #Mentions[U2] .... Equation (1)

U_Edge Weight(2,1) = (#Mentions of U2 made by U1) / #Mentions[U1] .... Equation (2)

where:

- #Mentions[U1] : Total number of tweets posted by U1 that mention other users.
- #Mentions[U2] : Total number of tweets posted by U2 that mention other users.

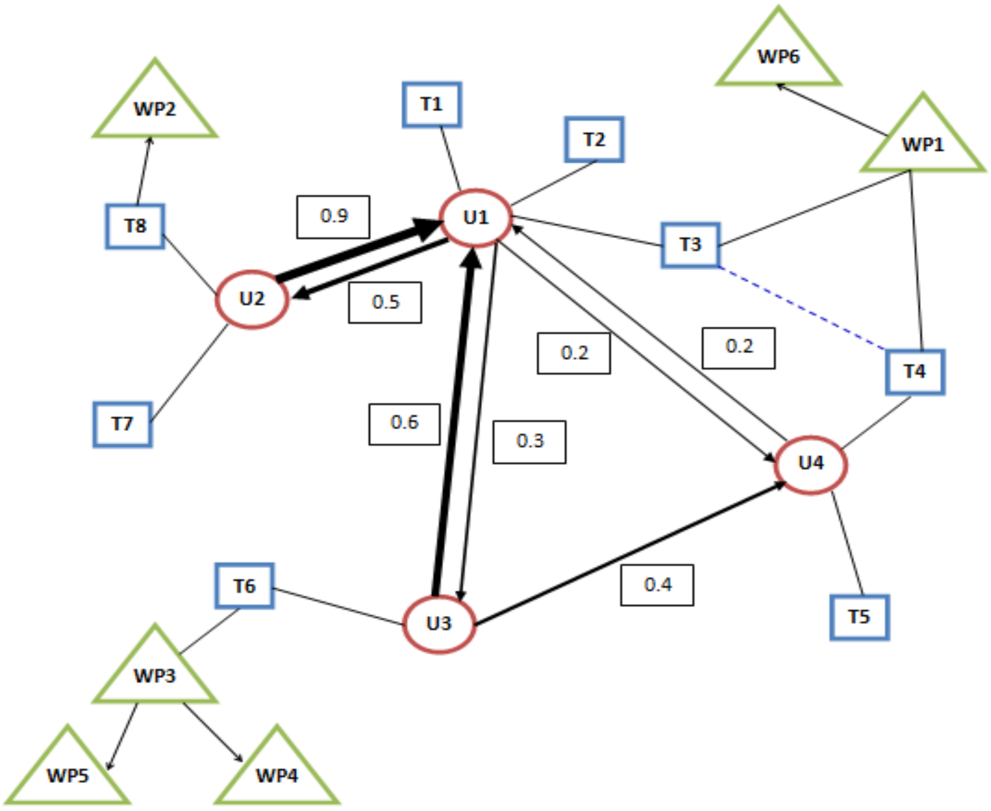


Figure 2: Social Network graph structure

2.4.5. The Tweet Importance Value

Each tweet has attributes that indicate its importance compared to other tweets. We selected five attributes which are used to calculate the importance value for each tweet (square node in the graph in Figure 2). The five attributes are as follows.

i. **The Favorite Count**: It indicates the number of favorite clicks the tweet received from users.
ii. **The Retweet Count**: It represents how many times this tweet has been retweeted by users.
iii. **The List Count**: It represents the number of groups/lists that this tweet is posted at.
iv. **The Number of Followers**: It represents the number of followers for the user/owner who posted this tweet.

v. **The User** (Owner, the user who posted it) importance value. The UIV (User Importance Value) is calculated by the summation of all the weighted U_Edges connected to that user.

$$UIV = \begin{cases} \Sigma\,(U\_Edge\ Weight) \text{ / number of inlink edges} \\ \\ \mathbf{0} \quad \textbf{, if there are no inlink edges} \end{cases}$$

The Tweet Importance Value (TIV) is calculated by the summation of all the attributes multiplied by arbitrary weights. These weights are chosen to be uniform for simplicity. But, we believe that there should be a heuristic methodology to calculate the optimal values for these weights.

where,

- W1, W2, W3, W4 and W5 are the arbitrary weights that should give a priority for one attribute compared to other. For now, it is uniform. Therefore, W1 = W2 = W3 = W4 = W5 = 0.2. Using regression analysis, we will compute the values of the weights.
- Number of attributes = 5
- A1: Favorite Count -- (# Fav(i) - Fav(min)) / (Fav(max) - Fav(min))
- A2: Retweet Count -- (# RT(i) - RT(min)) / (RT(max) - RT(min))
- A3: List Count -- (# List(i) - List(min)) / (List(max) - List(min))
- A4: Number of Followers -- (# Followers(i) - Followers(min)) / (Followers(max) - Followers(min))
- A5: UIV

2.4.6. The Web Page Importance Value

The webpage importance value is calculated with the PageRank algorithm using GraphX on all the webpages.

Earlier approach to compute the webpage importance values:
Using the earlier approach, the web page importance values were calculated based on their respective neighbors in the graph. Two aspects were considered to compute the web page importance values:

i. The web pages might be related to tweets. This also means that the URL of the web page is in the tweet. In other words, in the social network graph that we construct, a web page is related to a tweet if and only if there is an edge between the tweet and the web page. This edge will be unweighted. The webpage importance value is computed from the importance values of the tweets to which the web pages are related. The importance value of a web page was computed as follows:

where,

- *TIV(i)* is the tweet importance value of tweet *i*,
- *t* is the number of tweets to which the web page is related.

It can be observed that the above value computed is the average of the tweet importance values of the tweets that the web page is related to. For example, if both web pages A and B have the same importance value of 0.4; but A is related to *a* tweets and B is related to *b* tweets, where *a* > *b*. The web page A would have a higher importance than B because the reach of web page A is higher than that of B.

If two web pages have the same importance value, the web page with a higher *t* is given more importance. We did not consider the weighted average since the Tweet importance value is already a weighted one from which the web page importance value is computed.

ii. The web pages might be related to other web pages. This happens when a web page has a hyperlink to another web page. A web page is related to other web pages if and only if there is an edge from one web page to the other. The web page importance value was computed in Apache GraphX using the PageRank algorithm. PageRank values are computed using transition probability matrix.

Using the earlier approach, we compared the webpage importance values computed by the PageRank algorithm and that computed by our approach using Tweet Importance values. The final webpage importance value was decided to be the higher of the two importance values computed by both of these approaches. We discarded this approach since we cannot compare the results from both these approaches as they were computed by two different methods.

2.5. Project Management

This project was developed as a part of the Virginia Tech graduate course CS5604: Information Storage and Retrieval, during the Spring Semester, 2015. The project was supervised by Dr. Edward Fox who was assisted by Mohammed and Sunshin.

We followed the incremental model by initially following the naive method to find the importance values in the social network. We improved the quality of this graph and thus enhanced the importance values by following the approaches mentioned in the proposed solution.

Google Docs were used in creating the reports so that every team member can contribute to the report and modify it in a collaborative way.

2.6. Challenges

In the development of the solution, because of the inter-dependencies among the teams, we encountered the challenge of identifying tasks that might overlap with other teams. We were trying to construct a graph using content features, but the content features were already being considered by the Clustering team. Dr. Fox advised us to use non-content features and non-similarity approaches to construct a query independent graph.

We were also having trouble understanding the fact that the social network graph is independent of the user query.

3. Design

3.1. Project Model

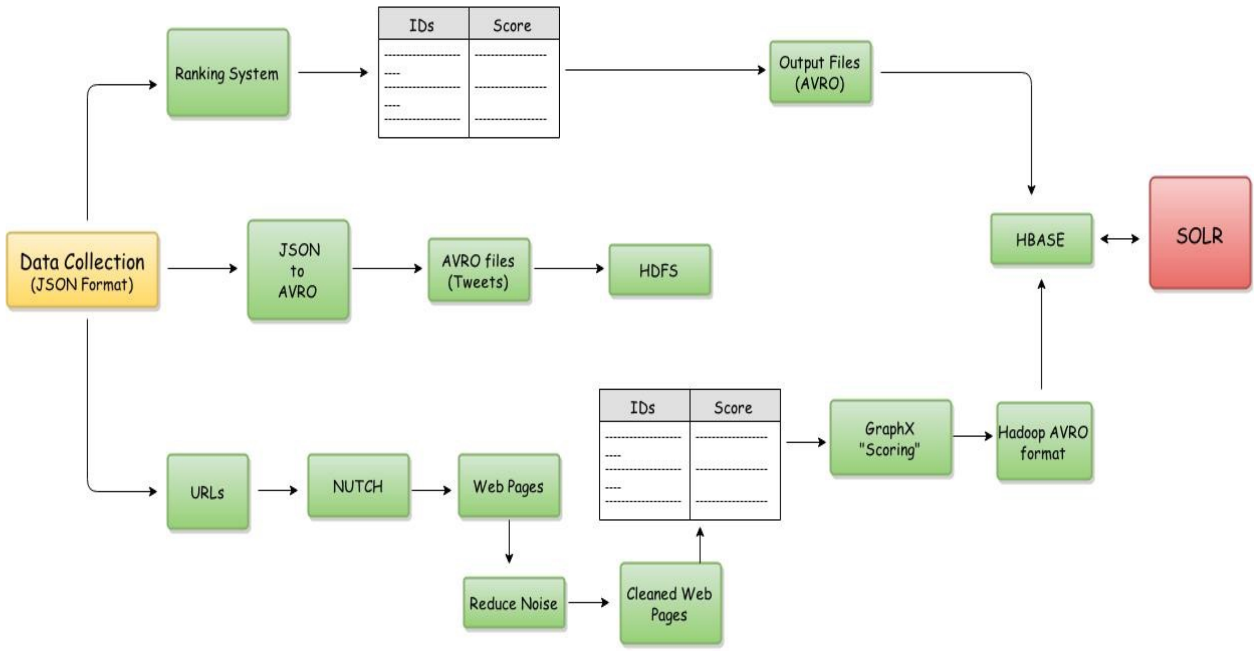


Figure 3: Project Model

Figure 3 describes the Social Networks project model. There is a data collection which is in JSON format. There are three different “stages” in the model.

First, The data collection is fed into the Ranking System, which produces the Tweet IDs and Tweet Importance Scores. These together are converted to AVRO format and stored in an output file. This output file is then stored in the HBASE for the SOLR team to use.

In the next stage, the data collection in the JSON format is converted to AVRO format and is fed to the HDFS.

In the final stage, the URLs are extracted from the data collection, crawled using NUTCH and then sent to the Reducing Noise team for cleaning of the webpages. Once the webpages are cleaned, the webpage IDs are sent to GraphX to get the webpage scores. This is then sent to the Hadoop team in AVRO format which then fed into the HBASE.

3.2. Tools

Programming language: Python

JSON [8]

The input data we worked with is JSON packages returned from Twitter API calls. In order to extract useful information, we use a generic Python JSON library to parse said packages.

NetworkX [9]

NetworkX is a network graph construction library that helps us to create useful data structures for visualization.

matplotlib [10]

We visualize the graph with matplotlib to understand the data in a better way.

Graphviz [11]

Graphviz is a generic data visualization tool, used to visualize the call graph of our workflow.

Solrpy [25]

Solrpy is a Python library used to import data to Solr. Solrpy is replaced by AVRO in the later part of the project.

Graphviz, matplotlib and NetworkX are visualization tools.

3.2.1 GraphX

GraphX is the new (alpha) Spark API for graphs (e.g., Web-Graphs and Social Networks) and graph-parallel computation (e.g., PageRank and Collaborative Filtering). At a high-level, GraphX extends the Spark RDD (Resilient Distributed Dataset) abstraction by introducing the Resilient Distributed Property Graph: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and mapReduceTriplets) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks. The GraphX API is currently only available in Scala but they plan to provide Java and Python bindings in the future. [15]

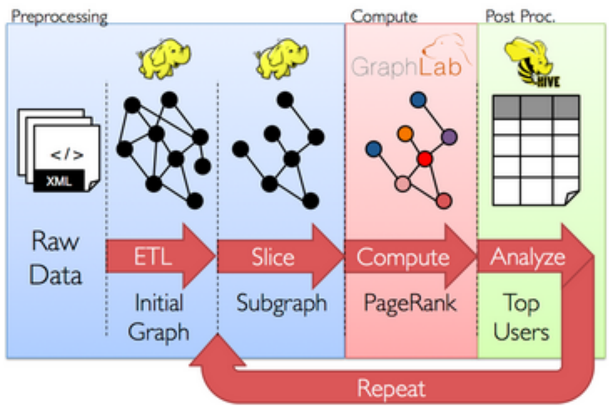


Figure 4: PageRank Process

The goal of the GraphX project is to unify graph-parallel and data-parallel computation in one system with a single composable API. The GraphX API enables users to view data both as graphs and as collections (i.e., RDDs) without data movement or duplication. By incorporating recent advances in graph-parallel systems, GraphX is able to optimize the execution of graph operations. We run our solution using GraphX in section 4.2 and give instructions to install GraphX in section 6.3.

In general, Spark can run well with anywhere from 8GB to hundreds of gigabytes of memory per machine. It is recommended to allocate only at most 75% of the memory for Spark; leave the rest for the operating system and buffer cache. [29] The class project cluster has a total memory of 31 GB.

3.2.2 Giraph

Apache Giraph is an iterative graph processing framework, built on top of Apache Hadoop. The input to a Giraph computation is a graph composed of vertices and directed edges. [18] For example vertices can represent people, and edges friend requests. Each vertex stores a value and so does each edge. The input, thus, not only determines the graph topology, but also the initial values of vertices and edges. Computation proceeds as a sequence of iterations, called supersteps in a bulk synchronous (BSP) fashion. Initially, every vertex is active. In each superstep each active vertex invokes a Compute method provided by the user. The Compute method: (1) receives messages sent to the vertex in the previous superstep, (2) computes using the messages, and the vertex and outgoing edge values, which may result in modifications to the values, and (3) may send messages to other vertices. The Compute method does not have direct access to the values of other vertices and their outgoing edges. Inter-vertex communication occurs by sending messages. Computation halts if all vertices have voted to halt and there are no messages in flight.
Giraph is an open source implementation of Google's Pregel. Giraph from the ground up is built for graph processing. This is different from Spark's GraphX which contains an implementation of the Pregel API built on the Spark DAG engine. [16]

There are three main stages of a Giraph program:

- Read and partition the data.
- Batch process the graph with BSP.

- Write the graph back to disk.

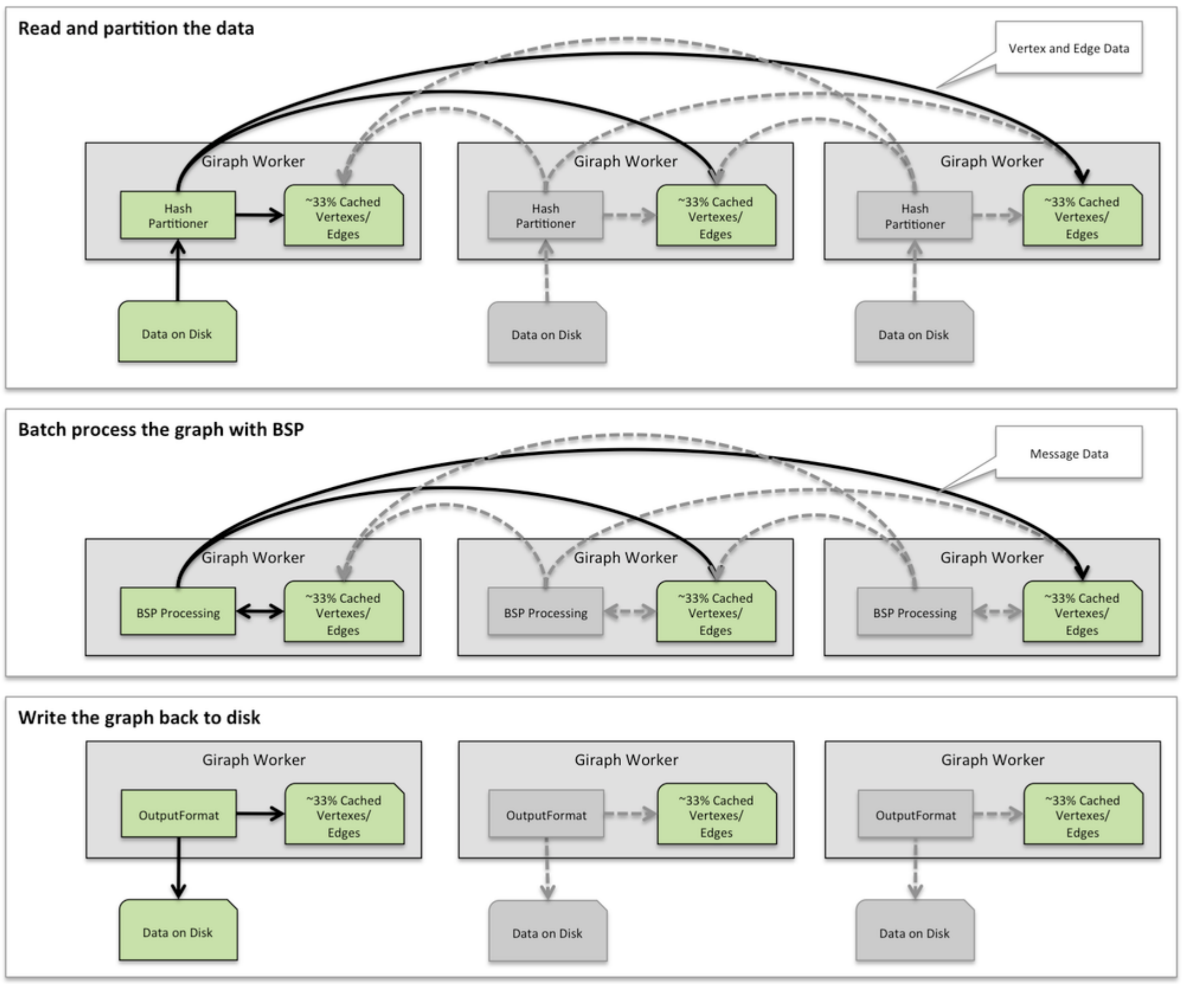

Read and partition the data
Vertex and Edge Data
Giraph Worker
Hash Partitioner
~33% Cached Vertexes/ Edges
Data on Disk
Batch process the graph with BSP
Message Data
BSP Processing
Write the graph back to disk
OutputFormat

Figure 5: Three main stages of a Giraph program

Like the Hadoop framework, Giraph is an efficient, scalable, and fault-tolerant implementation on clusters, with the distribution-related details hidden behind an abstraction. On a machine that performs computation, it keeps vertices and edges in memory and uses network transfers only for messages. During program execution, graph vertices are partitioned and assigned to workers. The default partition mechanism is hash-partitioning, but custom partition is also supported.

4. Implementation

4.1 Python Implementation

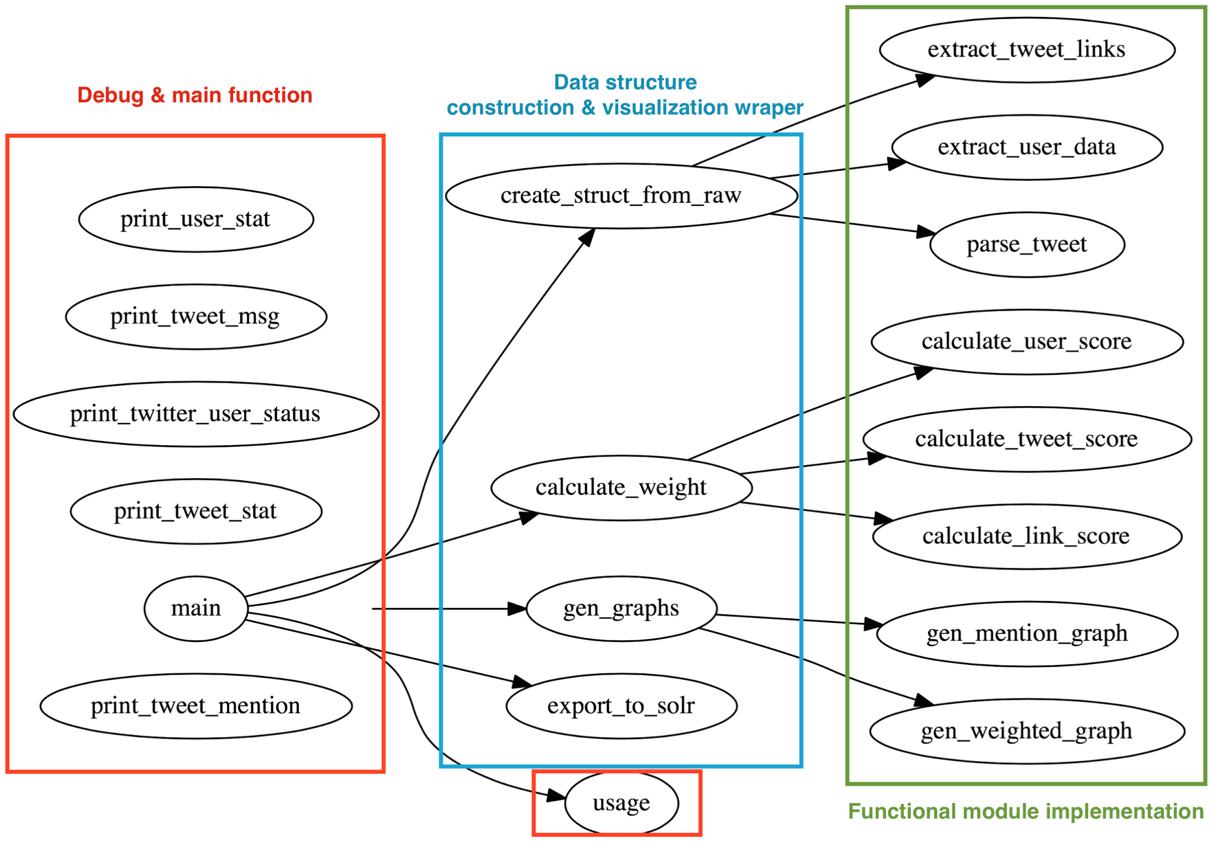


Figure 6: Call graph for current implementation of parsing/graph constructing script

The data collection we considered is related to “Police department” and the data is in the form of JSON files. Unlike other data collections, this data collection has metadata of tweets such as listed count, followers count, tweet count and so on.

Figure 6 shows the call graph of our current naive implementation of a parsing script. It reads input from this data collection and generates data structures for weight/importance computation. In addition, we also have components that generate diagrams that allow us to visualize the connection between certain relationships.

For example, Figure 6 shows a “re-tweet” connection among users constructed from a single JSON package file. This graph is an initial graph developed and is not similar to the graph structure described in Figure 2. Nodes in the graph represent users. An edge is formed when one user retweets another user’s tweet. The diagram has 206 nodes, and more than 70 edges. One thing worth noticing is that not all 206 users (nodes) are in the Police department data collection, therefore, we plotted a second graph (Figure 7) that contains only the users in Police department data collection. Since the User mention graph (full) is not clear in Figure 7, we provided it again in Appendix A so it is clearly visible. Similarly, Figure 7 is also present in Appendix B with an enlarged and clear visibility.

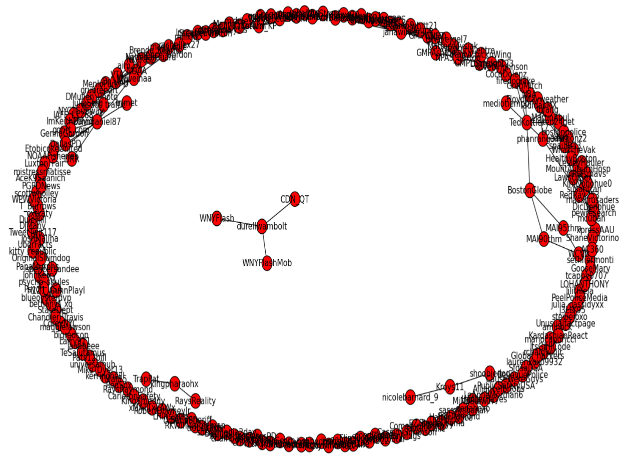


Figure 7: User Mention graph (full)

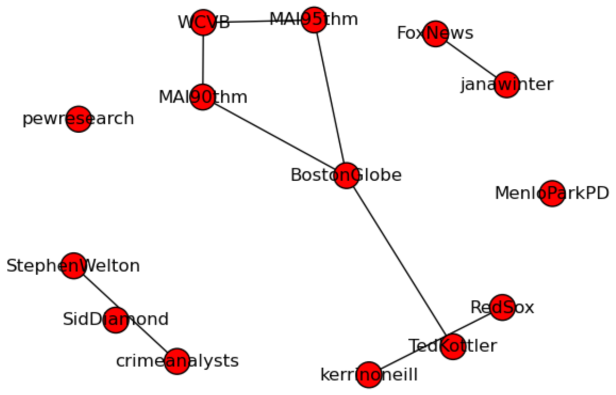


Figure 8: User Mention graph (Users in the Police department collection only)

As one can observe from Figures 6 and 7, after eliminating users that are not directly included in the data set, the graph size is greatly reduced. The number of nodes was reduced from 206 to 18, and the number of edges was reduced from 70 to less than 20. The singular nodes in the second diagram represents users that retweeted themselves.

In addition, we have obtained useful information from the Police department data collection. For example:

- The total number of tweets in the collection was 242 but only 75 of them were retweeted (either by other users or the user him/herself)
- 94 of 242 tweets in the data collection contained links to webpages
- As mentioned above, only 18 of 206 users in the data collection have communicated via tweet once or more

There were a few challenges we have discovered while implementing the parsing script:

- The data collection does not contain the list of friends for a user. Searching for friends of a user in the data collection includes a lot of computation.
- Hashtags are very useful to determine the theme/topic a tweet belongs to. But later, we approach the problem using the non-content features and thus, hashtags were not considered.

4.1.1 Parallel Python Implementation

The amount of computation our algorithm deals with is huge, therefore we have improved upon on our original implementation with multi-process support from native Python multiprocessing library.

We have found that with a small data collection that contains 100k tweets, the performance of multi-process version, running on quad core machine with 16GB memory, has improved almost 5 times. Most of the tasks were supposed to be parallelized. For example, reading from different parts of the input file and construct data structure.

Importance calculation for tweets has also sped up, but since it includes multiple synchronizations for each calculation, it is one of the bottlenecks in the script.

4.2. Graph-Parallel Computation

From social networks to language modeling, the growing scale and importance of graph data has driven the development of numerous new*graph-parallel* systems [15]. We have selected GraphX to be the tool compatible with Hadoop to test our importance equation. GraphX expresses graph computation within the Spark framework. Spark uses the Hadoop core library to talk to HDFS and other Hadoop-supported storage systems. Giraph is an iterative graph processing system built for high scalability and deployed in Hadoop [16], but while it allows to define a graph with several weighted edges between vertices it doesn't permit to define many weights for a given vertex.

The following table contains the tweets.

```
        "link":"http://twitter.com/aminock/statuses/462381268001652736","body":"Fastest way to legitimate an opinion
besides an PHD."
        "link":"http://twitter.com/aminock/statuses/462381150984765441","body":"Oh stop it Wilson, who in hell wanna
be a reporter, lie to people with a official seal."
        "link":"http://twitter.com/aminock/statuses/462382029729193985","body":"Se a palestina é minha banana é
porque sou cristão reformado."
        "link":"http://twitter.com/aminock/statuses/462382309954818048","body":"http://t.co/JnHNmA0ibb"
        "link":"http://twitter.com/aminock/statuses/462382670409109504","body":"RT @mariocaporicci: Se eu pudesse
dar outra dica a vocês, seria: Usem filtro de pano, o café fica mais gostoso."
        "link":"http://twitter.com/aminock/statuses/462382704018092032","body":"@mariocaporicci vou cobrar a
gorjeta."
        "link":"http://twitter.com/aminock/statuses/462382591220645888","body":"Fui no show dos mamonas co meu tio
que morreu e câncer e meu primo."
        "link":"http://twitter.com/aminock/statuses/462383002728017921","body":"RT @Global_hackers: RT @lilithlela:
No Jail Time for Stanley Cohen https://t.co/4Ez9agBxpx #UpTheRebels http://t.co/oyAgy1Nt2K"
        "link":"http://twitter.com/aminock/statuses/462383275106140160","body":"Algo entre vincerotrix julio césar a
mobília inglesa migrações e mentiras bíblicas."
        "link":"http://twitter.com/aminock/statuses/462383044272590848","body":"Carpintaria."
        "link":"http://twitter.com/aminock/statuses/462383440827269120","body":"A questão é como foi que convenceu
```

Figure 9 shows the relation between the tweets $\{t_n\}$, users $\{u_n\}$ and web pages $\{w_n\}$.

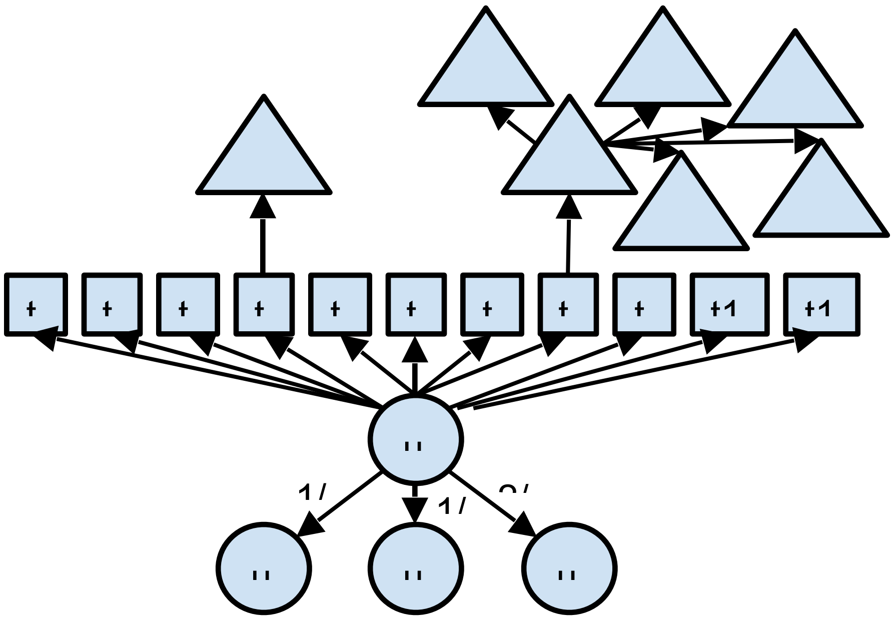


Figure 9: Graph example

4.2.1. Running our solution on GraphX

The graph from Figure 9 is created with the following code to describe the vertices.

- **User:** (ID, listed count, followers count, sum of weighted edges values to and from users/number of inlinks)
- **Tweet:** (ID, favorites count, retweets count, sum of weighted edge values/number of inlinks)
- **Web page:** (ID, favorites count, retweets count, sum of weighted edge values/number of inlinks)

The values have already been normalized using the formulas described in section 2.4.5.

The edges have source ID, destination ID, edge weight. The edge weight is the user Importance Value described in section 2.4.4 and it is calculated for every user vertex. For a tweet vertex, since it has an inlink from one user, the value is 1 by default. For a web page vertex, the value will be 2 by default. This helped us with our implementation when we needed to verify if a node is a tweet or web page.

```
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD
// Creates an RDD for the vertices
val vertexRDD: RDD[(Long, (Int, Int, Int))] = sc.parallelize(Array(
(296306236L, (100,100,0)),
(970845552L, (0,0,33)),
(263053919L, (0,0,33)),
(137739287L, (0,0,67)),
(462381268001652736L, (0,0, 100)),
(462381150984765441L, (0,0, 100)),
(462382029729193985L, (0,0, 100)),
(462382309954818048L, (0,0, 100)),
(462382670409109504L, (68,33, 100)),
(462382704018092032L, (0,0, 100)),
(462382591220645888L, (0,0, 100)),
(462383002728017921L, (100,100, 100)),
(462383275106140160L, (0,0, 100)),
(462383044272590848L, (0,0, 100)),
(462383440827269120L, (0,0, 100)),
(1L,(0,0,40)),
(2L,(0,0,80)),
(3L,(0,0,0)),
(4L,(0,0,0)),
(5L,(0,0,0)),
(6L,(0,0,0)),
(7L,(0,0,0))))
// Creates an RDD for edges
val edgeRDD: RDD[Edge[Int]] = sc.parallelize(Array(
Edge(296306236L, 970845552L, 33),
Edge(296306236L, 263053919L, 33),
Edge(296306236L, 137739287L, 67),
Edge(296306236L, 462381268001652736L, 100),
Edge(296306236L, 462381150984765441L, 100),
Edge(296306236L, 462382029729193985L, 100),
Edge(296306236L, 462382309954818048L, 100),
Edge(296306236L, 462382670409109504L, 100),
Edge(296306236L, 462382704018092032L, 100),
Edge(296306236L, 462382591220645888L, 100),
Edge(296306236L, 462383002728017921L, 100),
Edge(296306236L, 462383275106140160L, 100),
Edge(296306236L, 462383044272590848L, 100),
```

In addition to the vertex and edge views of the property graph, GraphX also exposes a triplet view. The triplet view logically joins the vertex and edge properties yielding an RDD[EdgeTriplet[VD, ED]] containing instances of the EdgeTriplet class. This join can be expressed using the following SQL query:

```
SELECT src.id, dst.id, src.attr, e.attr, dst.attr
FROM edges AS e LEFT JOIN vertices AS src, vertices AS
dst
```

The EdgeTriplet class extends the Edge class by adding the srcAttr and dstAttr members which contain the source and destination properties respectively. We use the triplet view of our graph to generate the score values for the tweets.

```
def printToFile(f: java.io.File)(op: java.io.PrintWriter => Unit) {
  val p = new java.io.PrintWriter(f)
  try { op(p) } finally { p.close() }
}

import java.io._
//tweet score
printToFile(new File("tweets.txt")) { p =>
for (triplet <- graph.triplets.filter(t => t.attr ==100).collect) {
   val score=(0.2*triplet.dstAttr._1/100) + (0.2*triplet.dstAttr._2/100) +
(0.2*triplet.srcAttr._1/100)+(0.2*triplet.srcAttr._2/100)+(0.2*triplet.srcAttr._3);
  p.println(triplet.dstId+s" "+score)
}
}
```

We cannot obtain the web pages scores at the same time because we need the tweets values to obtain the average edge weighted values for some web pages. The following code snippet uses PageRank to obtain the web pages importance score for the web pages

```
//Restrict the graph to webpages
val subgraph = graph.subgraph(t => t.attr == 200)
val pagerankGraph = subgraph.pageRank(0.001)

//webpage score
printToFile(new File("webpages.txt")) { p =>
for (triplet <- pagerankGraph.triplets.collect) {
     p.println(triplet.dstId+s" "+triplet.srcAttr)
}
}
```

The importance score output will be the following for the tweets:

Table 1: Importance values of the tweets

Tweet ID	**Importance value**
462383002728017921	0.8
462382670409109504	0.602
462381150984765441	0.602
462381268001652736	0.4
462382029729193985	0.4
462382309954818048	0.4
462382591220645888	0.4
462382704018092032	0.4
462383044272590848	0.4
462383275106140160	0.4
462383440827269120	0.4

Our earlier approach for the web pages selected the higher value comparing the PageRank value with the value obtained from linked tweets. The values for the web pages with IDs 1 and 2 with PageRank was 0.15 and 0.15 respectively, but the output for the earlier approach was 0.4 and 0.8 because these were the values for the linked tweets referencing those web pages.

The importance score output will be the following for the web pages:

Table 2: Importance values of webpages

Web Page ID	**Importance value**
2	0.15 (Earlier approach: 0.8)
1	0.15 (Earlier approach: 0.4)
3	0.277
4	0.277
5	0.277
6	0.277
7	0.277

These are the top 5 tweets and web pages in the data collection of 500,000 tweets:

```
        "link":"http://twitter.com/aminock/statuses/462383002728017921","body":"RT @Global_hackers: RT @lilithlela: No Jail Time for Stanley Cohen https://t.co/4Ez9agBxpx #UpTheRebels http://t.co/oyAgy1Nt2K"
        "link":"http://twitter.com/aminock/statuses/462382670409109504","body":"RT @mariocaporicci: Se eu pudesse dar outra dica a vocês, seria: Usem filtro de pano, o café fica mais gostoso."
        "link":"http://twitter.com/aminock/statuses/462381150984765441","body":"Oh stop it Wilson, who in hell wanna be a reporter, lie to people with a official seal."
        https://www.change.org/en-GB/petitions/hon-norman-a-mordue-no-jail-time-for-stanley-cohen
        http://www.youtube.com/watch?v=0_3Qtw4Pze8
```

4.2.2. Running our solution on cleaned web pages

After the Reducing noise team processed the web pages, we received an AVRO file with 8794 web pages and their important attributes. From the AVRO file, with the script **graph.py**, we extracted the vertex and edges and produced two files with the GraphX format, **vertex.scala** and **edges.scala**. To check if there are connections betweens the outlinks and the initial webpages run the script **graphVertex.py** and **graphEdges.py**. For this particular collection there were no connections between the web pages with assigned IDs. Thus, we have the scores for all the web pages as 0.15. We talked with the Hadoop team and decided that because the outlinks web pages don't have IDs assigned by the noise reduction team, we will only provide them with scores for the initial web pages with IDs.

After this we ran three scripts in GraphX to create the graph and obtain the PageRank values

```
:load vertex.scala
:load edges.scala
:load rank.scala
```

Then we ran the Python script **avro.py** to produce our social importance score for webpages in AVRO files with the schema defined by the class groups.

```
{"namespace": "cs5604.tweet.social",
 "type": "record",
 "name": "PageSocial",
 "fields": [
     {"name": "doc_id", "type": "string"},
     {"doc": "analysis", "name": "social_importance", "type": 
["double", "null"]}
 ]
}
```

The files are described in section 6.6. "Inventory of Repository Files" and available in the VTechWorks repository.

4.3. Timeline

January 19 - January 22:

- Groups were formed

January 22 - January 29:

- Literature review
- Analysis of dependencies with other teams, Installing Solr

January 30 - February 5:

- Literature review
- Creation of a team-dependency graph
- Uploading sample data in Solr
- Discussion with the Clustering and NER teams.

February 6 - February 13:

- Literature review
- Loading the Massachusetts Police Departments tweet collection in Solr to begin the Social Network and Importance analysis.

February 14 - February 19:

- Discussion of the design of our model
- Definition and use of algorithms to gather the best features to obtain an importance value for tweets and web pages

February 20 - February 26:

- Uploading tweets and web pages from the class sample data
- Proposing new publications to read about social networks and asking for help from professors/ experts in the area
- Discussing the schema that describes the specific output our team will produce for each of the web pages and propose a solution
- Installing, running and using Hadoop, GraphX and Giraph
- Uploading the Massachusetts Police Departments collection of tweets as a graph in GraphX and Giraph.

February 27 - March 22:

- Definition of the social network directed graph G(V, E)
- Description of the relation our group has with the rest of the teams, specifying inputs and outputs
- Loading of our small collection (Massachusetts Police Department) into the Hadoop cluster
- Loading of our large collection (Massachusetts Police Department) onto the cluster

March 23 - March 29:

- Use of Nutch on our tweets to crawl web pages.
- Definition of the social network directed graph G(V, E) taking class discussion into consideration.

- Definition of our output schema in AVRO format.

- Building of the prototype that will output the importance score for tweets and web pages.

  March 30 - April 5:
- Revision of the social network directed graph G(V, E) taking class discussion into consideration.
- Building of the prototype that will output the importance score for tweets and web pages.
- Revision of the resources "Twitter research and Tools" shared by the class TA for the Social Network group.

  April 6 - April 12:
- Upload the AVRO Massachusetts Police Department collection to HDFS.
- Generate the Massachusetts Police Department web page collection using Nutch.

  April 13 - April 20:
- Web page importance value calculation.
- Testing and evaluation.

  April 21 - April 26:
- Testing and evaluation.
- Interdependencies between LDA and Clustering.

  April 27 - May 6:
- Presentation to the class.
- Final Report.
- Final implementation.

4.4. Collaboration with others teams

The following descriptions explain how the work of some of the teams relates to the Social Network & Importance team.

<u>Hadoop Team:</u> After briefly discussing with the Hadoop team, they would receive the output of our graph in either sequence file/AVRO format in HBase.

<u>Solr Team:</u> After in-class and Piazza discussion, the Solr team added a hashtag field for tweet schema and ownership, content type fields for the webpage schema as we requested. The output of the social network team is a vector of floating point values that represent the score, or importance, for tweet messages. The vector is going to be stored in the social_vector field in the tweet schema. The same goes for webpages.

<u>Reducing Noise Team:</u> The Reducing Noise group is conscious that the Social Network and all the other teams worked with the assumption that the data they'll receive, after the Reducing Noise processed it, is clean and relevant to the event. They provided the "clean" data in the exact same format as the initial input data.

4.5. Deliverables

We have a prototype using JSON files as data source that output vectors that represent importance for each tweet to Solr. The importance calculation is based on the "importance algorithm" we mention in Section 4.2.

The implementation will adapt to using HBase as data source with the confirmed schema.

Webpages Generation:
We have a “Python” script that walks through the tweet collection and extract the “URLs”. A small collection of the “URLs” is selected to be expanded and fed to the “Nutch” to generate the web page collection.

Uploading the Tweet Collection into the HDFS:
The tweet collection was handed to us in the JSON format. In order to upload it into the HDFS, we used a tool “json2avro” to convert all the tweets from the JSON to AVRO format. These files were uploaded into the HDFS.

Uploading the Web Page Scores into the HDFS:
After producing our social importance score for webpages in AVRO files with the schema defined by the class groups, this file was uploaded into the HDFS.

```
scp /Users/vcedeno/Desktop/webpages/pagesocial.avro
cs5604s15_social@hadoop.dlib.vt.edu:/home/cs5604s15_social/webpages

hadoop fs -copyFromLocal /home/cs5604s15_social/webpages/
pagesocial.avro /user/cs5604s15_social/webpages

hadoop fs -ls /user/cs5604s15_social/webpages
```

5. User’s Manual

5.1. Data Collection

5.1.1. Tweets Archives

The size of the Police department data collection is roughly 14GB in size (uncompressed), containing more than 13,000 JSON packages. Each JSON package contains about 250 tweet messages. Each tweet message is encoded in the following format:

```
{
"id":"tag:search.twitter.com,2005:462381267473170432",
"objectType":"activity",
"actor":
      {
            "objectType":"person",
            "id":"id:twitter.com:437092941",
            "link":"http://www.twitter.com/lapdwilshire",
            "displayName":"LAPD Wilshire",
            "postedTime":"2011-12-15T00:33:15.000Z",
            "image":"https://pbs.twimg.com/profile_images/2145475578/
wilshire2_normal.JPG",
            "summary":"Not monitored 24/7. 911 for emergencies.
Wilshire Community Police Station\r\n4861 West Venice\r\nLos Angeles,
CA 90019\r\n213-473-0476 Voice\r\n213-485-2112 TDD/TTY",
            "links":[{"href":"http://www.lapdonline.org/
wilshire_community_police_station","rel":"me"}],
            "friendsCount":655,
            "followersCount":4659,
            "listedCount":114,
            "statusesCount":1800,
            "twitterTimeZone":null,
            "verified":false,
            "utcOffset":null,
            "preferredUsername":"lapdwilshire",
            "languages":["en"],
            "location":{"objectType":"place","displayName":"Los
Angeles, CA"},
            "favoritesCount":595
      },
"verb":"post",
"postedTime":"2014-05-03T00:00:59.000Z",
"generator":{"displayName":"Instagram","link":"http://
instagram.com"},
"provider":
{"objectType":"service","displayName":"Twitter","link":"http://
www.twitter.com"},
"link":"http://twitter.com/lapdwilshire/statuses/462381267473170432",
"body":"LAPD Air support-birds eye view. You can run but you can't
hide! #LAPD #LA #LosAngeles #CityOfAngels… http://t.co/VFvz1hLK6w",
"object":
      {
            "objectType":"note",
            "id":"object:search.twitter.com,2005:462381267473170432",
            "summary":"LAPD Air support-birds eye view. You can run
but you can't hide! #LAPD #LA #LosAngeles #CityOfAngels… http://t.co/
VFvz1hLK6w",
```

5.1.2 Web page Archives

We received cleaned web pages from the Noise Reduction team in the following format:

```
{"namespace": "cs5604.webpage.NoiseReduction",
 "type": "record",
 "name": "WebpageNoiseReduction",
 "fields": [
     {"name": "doc_id", "type": "string"},
     {"doc": "original", "name": "text_clean", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "text_original", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "created_at",  "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "accessed_at",  "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "author", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "subtitle", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "section", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "lang", "type": ["null", "string"],
"default": null},
     {"doc": "original", "name": "coordinates", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "urls", "type": ["null", "string"],
"default": null},
     {"doc": "original", "name": "content_type", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "text_clean2", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "collection", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "title", "type": ["null", "string"],
"default": null},
     {"doc": "original", "name": "domain", "type": ["null",
"string"], "default": null},
     {"doc": "original", "name": "url", "type": ["null", "string"],
"default": null},
     {"doc": "original", "name": "appears_in_tweet_ids", "type":
["null", "string"], "default": null}
 ]
}
```

5.2 Project results and statistics

The Police department data collection contains 2,848,618 tweets from 376,459 users, containing 582,186 web pages. The script gives as output the importance scores for each tweet in AVRO format.

A few interesting observations include:

i. The data collection does not contain many closely related user groups.

ii. High score (Importance value) Tweets are from the users who have high follower counts. We observed that all the tweets in the top 10 tweets in the Police department data collection are from CNN Breaking News.

Here are some results regarding the performance of the script:

```
Environment:
    Data size: 2848618 tweets, 376459 users
    Hardware: Intel Core i7  2.4GHz processor with 6MB shared L3 cache
    RAM: 16GB
    All read/write operations are done on a RAMDisk.

Process number = 1
Time taken for constructing edges and calculating weights: 888.279s
Time taken for calculating User Importance Values (UIV):  4.028s
Time taken for calculating Tweet Importance Values (TIV): 187.757s
Total time: 1080.064s

Process number = 4
Time taken for Reading tweets: 769.504s
Time taken for calculating User Importance Values (UIV): 6.554s
Time taken for calculating Tweet Importance Values (TIV): 125.788s
Total time: 901.846s
```

Compared to our previous implementation, which divides the data collection into small pieces and runs the entire script in parallel on all processors, our final script has to read from and write to single sources. This creates an I/O bottleneck as can be seen from the running time statistics above.

Top 10 Tweets in the Police department data collection

'RT @BleacherReport: Belgium defeats the U.S. 2-1 and they advance to the quarterfinals. 2014 was an incredible #WorldCup run, USMNT http:/\u2026'
cnnbrk
score 0.236169
UIV: 0.3333333333333333
# of followers: 17842891
list count: 155404
mentioned by users in the collection 1 times
retweet count: 1773
fav_count: 0

'RT @BleacherReport: The Cleveland Cavaliers select Andrew Wiggins with the first pick in the 2014 NBA Draft! http://t.co/fhgIICzdOn'
cnnbrk
score 0.236164
UIV: 0.3333333333333333
# of followers: 17842891
list count: 155404
mentioned by users in the collection 1 times
retweet count: 1673
fav_count: 0

'RT @BleacherReport: The #Spurs are the 2014 NBA Champions! http://t.co/jLbgRqVD9R'
cnnbrk
score 0.236163
UIV: 0.3333333333333333
# of followers: 17842891
list count: 155404
mentioned by users in the collection 1 times
retweet count: 1651
fav_count: 0

'RT @BleacherReport: The Browns trade the No.4 pick to the Bills. Buffalo selects WR Sammy Watkins and he takes a selfie with Goodell http:/\u2026'
cnnbrk
score 0.236130
UIV: 0.3333333333333333
# of followers: 17842891
list count: 155404
mentioned by users in the collection 1 times
retweet count: 1004
fav_count: 0

'RT @cnni: Why should the abduction of Nigerian schoolgirls matter to you? http://t.co/FaKqzmEpjf #BringBackOurGirls http://t.co/u7mU1SiQcQ'

These are the top 5 tweets and web pages in the data collection of 500,000 tweets:

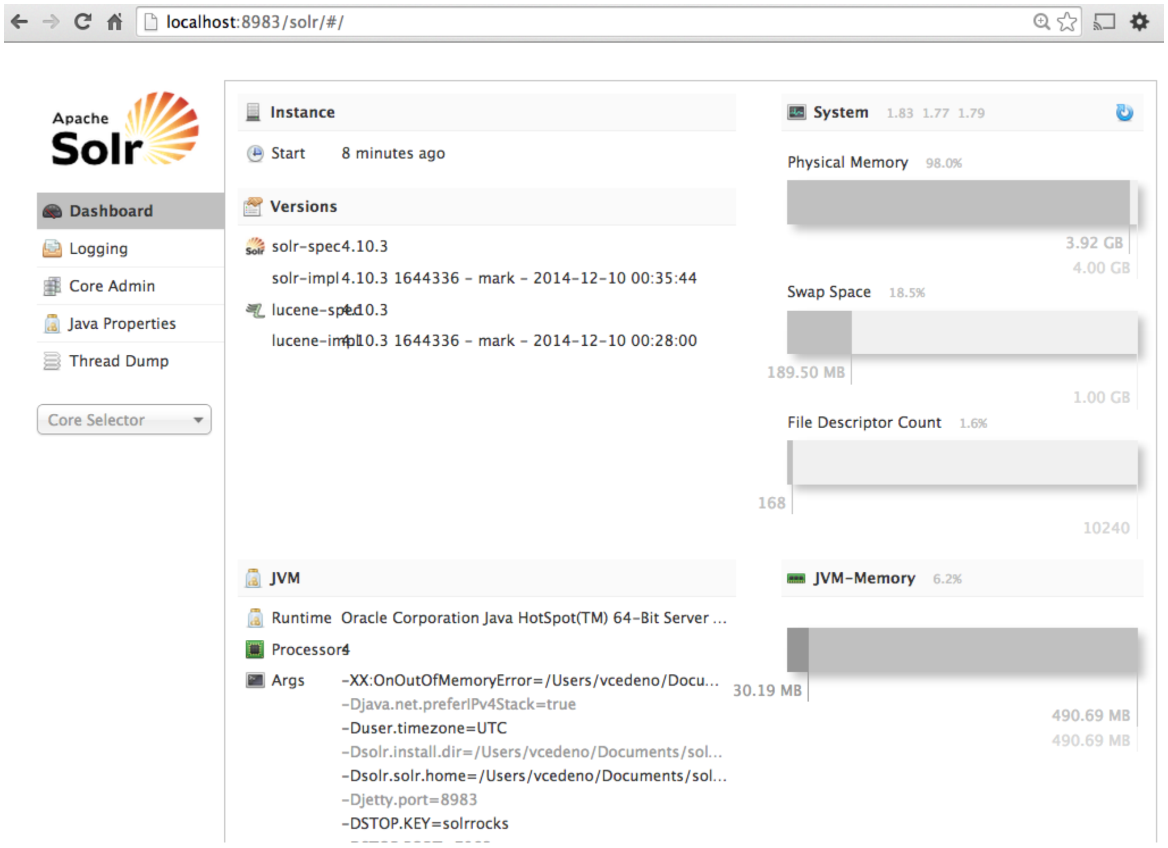
```
        "link":"http://twitter.com/aminock/statuses/462383002728017921","body":"RT @Global_hackers: RT @lilithlela:
No Jail Time for Stanley Cohen https://t.co/4Ez9agBxpx #UpTheRebels http://t.co/oyAgy1Nt2K"
        "link":"http://twitter.com/aminock/statuses/462382670409109504","body":"RT @mariocaporicci: Se eu pudesse
dar outra dica a vocês, seria: Usem filtro de pano, o café fica mais gostoso."
        "link":"http://twitter.com/aminock/statuses/462381150984765441","body":"Oh stop it Wilson, who in hell wanna
be a reporter, lie to people with a official seal."
        https://www.change.org/en-GB/petitions/hon-norman-a-mordue-no-jail-time-for-stanley-cohen
        http://www.youtube.com/watch?v=0_3Qtw4Pze8
```

6. Developer's Manual

6.1. Solr

6.1.1 Installing Solr

i. The latest version of Apache Solr was downloaded from: http://apache.cs.utah.edu/lucene/solr/4.10.3/
ii. To meet the requirements to run Solr the latest version of the Java Development Kit was downloaded from: http://www.oracle.com/technetwork/java/javase/downloads/jdk8downloads2133151.html
iii. In a terminal in the example folder from the Solr folder the following command has to be executed to run Solr.
`$ java jar start.jar`
iv. Solr is now running and the Solr Admin web application can be accessed by loading http://localhost:8983/solr/admin/



Instance
Start 8 minutes ago

Versions
solr-spec 4.10.3
solr-impl 4.10.3 1644336 - mark - 2014-12-10 00:35:44
lucene-spec 4.10.3
lucene-impl 4.10.3 1644336 - mark - 2014-12-10 00:28:00

System 1.83 1.77 1.79
Physical Memory 98.0%
3.92 GB
4.00 GB
Swap Space 18.5%
189.50 MB
1.00 GB
File Descriptor Count 1.6%
168
10240

JVM
Runtime Oracle Corporation Java HotSpot(TM) 64-Bit Server ...
Processors 4
Args -XX:OnOutOfMemoryError=/Users/vcedeno/Docu...
-Djava.net.preferIPv4Stack=true
-Duser.timezone=UTC
-Dsolr.install.dir=/Users/vcedeno/Documents/sol...
-Dsolr.solr.home=/Users/vcedeno/Documents/sol...
-Djetty.port=8983
-DSTOP.KEY=solrrocks

JVM-Memory 6.2%
30.19 MB
490.69 MB
490.69 MB

The following tutorial can be use to test Solr. [12] http://www.solrtutorial.com/solrin5-minutes.html

6.1.2 Uploading tweets

Class Data Sample

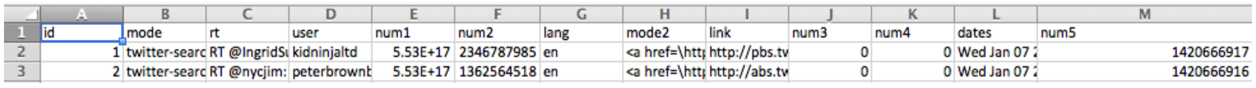
The following procedure was used to upload tweets and text files from the web pages in the Paris shooting Class data sample:

i. The Solr `schema.xml` file was modified to include the new data present in the CSV file having tweets:

```
<!--
class data schema
 -->
<field name="mode" type="text_general" indexed="true"
stored="true"/>
<field name="user" type="text_general" indexed="true"
stored="true"/>
<field name="rt" type="text_general" indexed="true"
stored="true" omitNorms="true"/>
<field name="num1" type="text_general" indexed="true"
stored="true"/>
<field name="num2" type="text_general" indexed="true"
stored="true"/>
<field name="num3" type="int" indexed="true" stored="true"
/>
<field name="num4" type="int" indexed="true" stored="true"
/>
<field name="num5" type="text_general" indexed="true"
stored="true"/>
<field name="lang" type="text_general" indexed="true"
stored="true"/>
<field name="mode2" type="text_general" indexed="true"
stored="true" omitNorms="true"/>
<field name="link" type="text_general" indexed="true"
stored="true" omitNorms="true"/>
<field name="dates" type="text_general" indexed="true"
stored="true"/>
```

ii. After this, Solr was started on the Solr directory:
```
$ bin/solr start
```

iii. In the CSV file, a header-row was added with the names of the columns. Also, for each tweet the ID column was created.

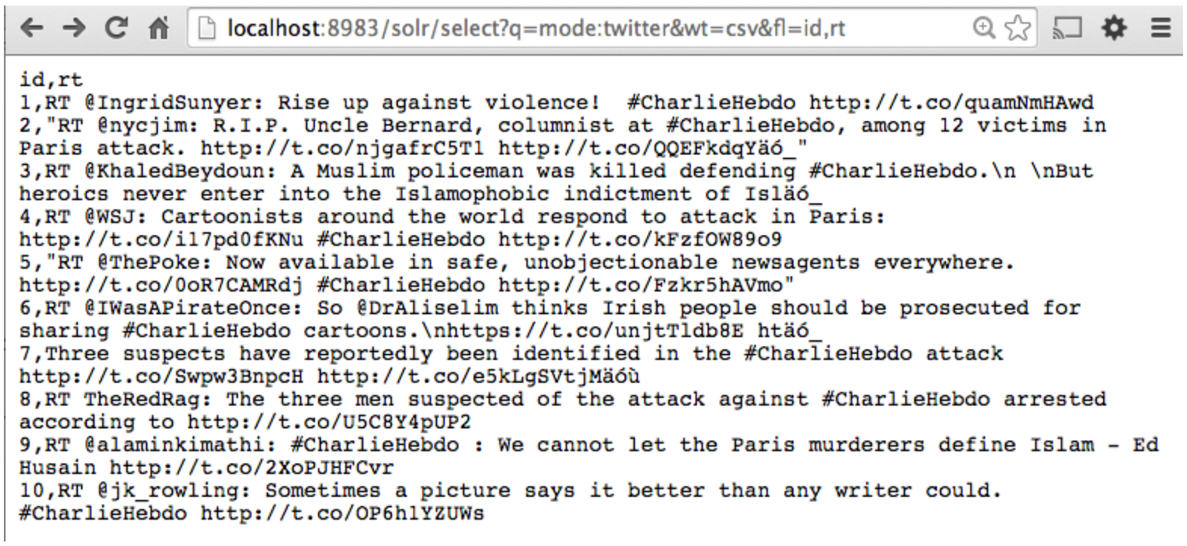


	A	B	C	D	E	F	G	H	I	J	K	L	M
1	id	mode	rt	user	num1	num2	lang	mode2	link	num3	num4	dates	num5
2	1	twitter-searc	RT @IngridSι	kidninjaltd	5.53E+17	2346787985	en	<a href=\http	http://pbs.tw	0	0	Wed Jan 07 2	1420666917
3	2	twitter-searc	RT @nycjim:	peterbrownt	5.53E+17	1362564518	en	<a href=\http	http://abs.tw	0	0	Wed Jan 07 2	1420666916

iv. After the file was ready, we ran the following command in Solr command line to upload the data:

```
$ curl http://localhost:8983/solr/update/csv?commit=true
--data-binary @paris.csv -H 'Content-type:text/plain;
charset=utf-8'
```

v. Next, we were able to query the data from Solr.



```
localhost:8983/solr/select?q=mode:twitter&wt=csv&fl=id,rt
```

```
id,rt
1,RT @IngridSunyer: Rise up against violence!  #CharlieHebdo http://t.co/quamNmHAwd
2,"RT @nycjim: R.I.P. Uncle Bernard, columnist at #CharlieHebdo, among 12 victims in
Paris attack. http://t.co/njgafrC5T1 http://t.co/QQEFkdqYäó_"
3,RT @KhaledBeydoun: A Muslim policeman was killed defending #CharlieHebdo.\n \nBut
heroics never enter into the Islamophobic indictment of Isläó_
4,RT @WSJ: Cartoonists around the world respond to attack in Paris:
http://t.co/il7pd0fKNu #CharlieHebdo http://t.co/kFzfOW89o9
5,"RT @ThePoke: Now available in safe, unobjectionable newsagents everywhere.
http://t.co/0oR7CAMRdj #CharlieHebdo http://t.co/Fzkr5hAVmo"
6,RT @IWasAPirateOnce: So @DrAliselim thinks Irish people should be prosecuted for
sharing #CharlieHebdo cartoons.\nhttps://t.co/unjtTldb8E htäó_
7,Three suspects have reportedly been identified in the #CharlieHebdo attack
http://t.co/Swpw3BnpcH http://t.co/e5kLgSVtjMäóù
8,RT TheRedRag: The three men suspected of the attack against #CharlieHebdo arrested
according to http://t.co/U5C8Y4pUP2
9,RT @alaminkimathi: #CharlieHebdo : We cannot let the Paris murderers define Islam - Ed
Husain http://t.co/2XoPJHFCvr
10,RT @jk_rowling: Sometimes a picture says it better than any writer could.
#CharlieHebdo http://t.co/OP6hlYZUWs
```

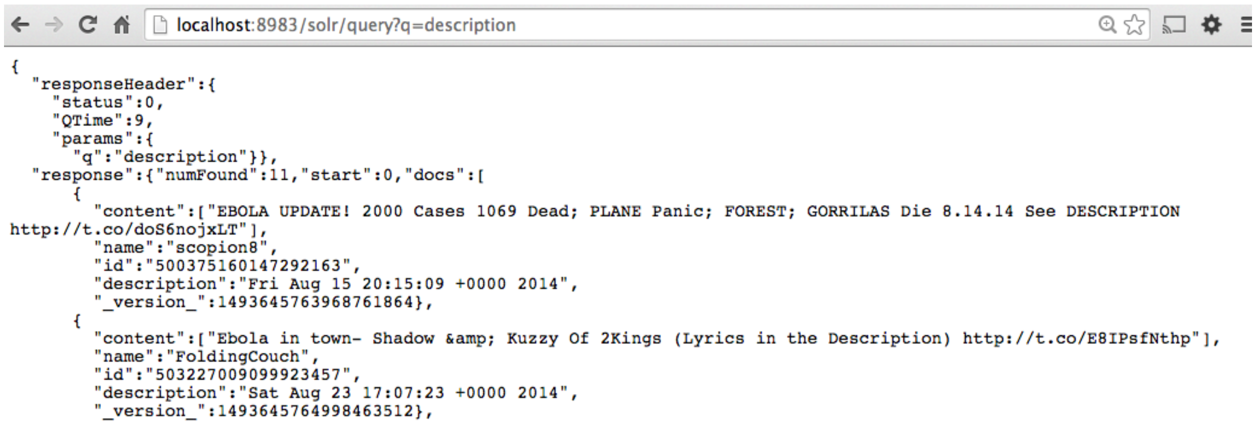
Ebola Dataset

We uploaded the ebola dataset into Solr to have the format of the tweets other groups are working with.

i. We ran the following command:

```
$curl 'http://localhost:8983/solr/update?commit=true&separator=
%09' -H 'Content-type:application/csv' --data-binary @z224t.csv
```

ii. Next, we were able to query the data from Solr.



localhost:8983/solr/query?q=description

```
{
  "responseHeader":{
    "status":0,
    "QTime":9,
    "params":{
      "q":"description"}},
  "response":{"numFound":11,"start":0,"docs":[
      {
        "content":["EBOLA UPDATE! 2000 Cases 1069 Dead; PLANE Panic; FOREST; GORRILAS Die 8.14.14 See DESCRIPTION
http://t.co/doS6nojxLT"],
        "name":"scopion8",
        "id":"500375160147292163",
        "description":"Fri Aug 15 20:15:09 +0000 2014",
        "_version_":1493645763968761864},
      {
        "content":["Ebola in town- Shadow &amp; Kuzzy Of 2Kings (Lyrics in the Description) http://t.co/E8IPsfNthp"],
        "name":"FoldingCouch",
        "id":"503227009099923457",
        "description":"Sat Aug 23 17:07:23 +0000 2014",
        "_version_":1493645764998463512},
```

6.1.3 Uploading web pages

Class Data Sample

To upload the web pages, Solr needs to be configured to index non-XML based files [2]:

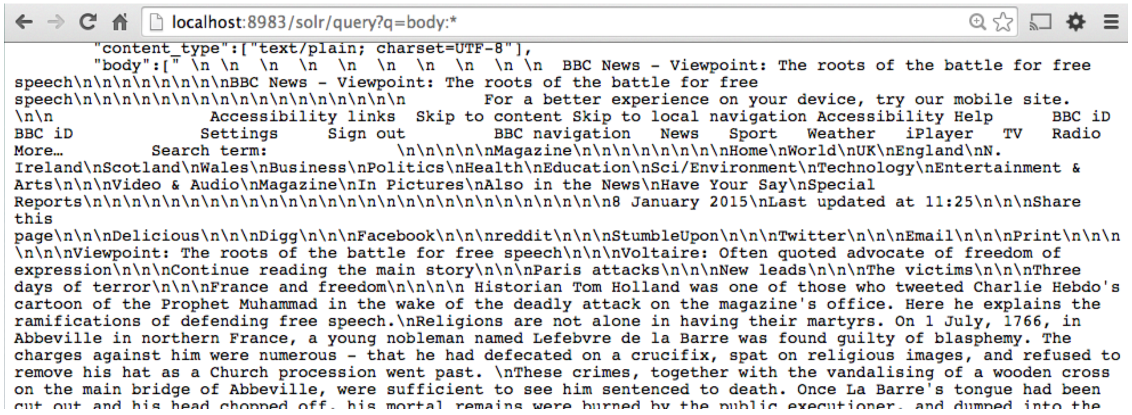
i. Edit `solr/conf/schema.xml`
ii. Add the following entries:

```
<field  name="body"  type="text_general"  indexed="true"
stored="true"  multiValued="true"/>
<copyField source="body" dest="text"/>
```

Then we used the following command to upload the files:

```
curl "http://localhost:8983/solr/update/extract?
literal.id=d0&uprefix=attr_&fmap.content=body&commit=true"
-F "myfile=@0.txt"
```

Then, we were able to query the web pages in Solr:



localhost:8983/solr/query?q=body:*

```
        "content_type":["text/plain; charset=UTF-8"],
        "body":[" \n \n  \n  \n  \n  \n  \n  \n  \n \n  BBC News - Viewpoint: The roots of the battle for free
speech\n\n\n\n\n\n\n\nBBC News - Viewpoint: The roots of the battle for free
speech\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n        For a better experience on your device, try our mobile site.
\n\n                Accessibility links  Skip to content Skip to local navigation Accessibility Help      BBC iD
BBC iD              Settings     Sign out          BBC navigation    News    Sport    Weather    iPlayer    TV    Radio
More…        Search term:            \n\n\n\n\nMagazine\n\n\n\n\n\n\n\nHome\nWorld\nUK\nEngland\nN.
Ireland\nScotland\nWales\nBusiness\nPolitics\nHealth\nEducation\nSci/Environment\nTechnology\nEntertainment &
Arts\n\n\nVideo & Audio\nMagazine\nIn Pictures\nAlso in the News\nHave Your Say\nSpecial
Reports\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n\n8 January 2015\nLast updated at 11:25\n\n\nShare
this
page\n\n\nDelicious\n\n\nDigg\n\n\nFacebook\n\n\nreddit\n\n\nStumbleUpon\n\n\nTwitter\n\n\nEmail\n\n\nPrint\n\n\n
\n\n\nViewpoint: The roots of the battle for free speech\n\n\nVoltaire: Often quoted advocate of freedom of
expression\n\n\nContinue reading the main story\n\n\nParis attacks\n\n\nNew leads\n\n\nThe victims\n\n\nThree
days of terror\n\n\nFrance and freedom\n\n\n\n Historian Tom Holland was one of those who tweeted Charlie Hebdo's
cartoon of the Prophet Muhammad in the wake of the deadly attack on the magazine's office. Here he explains the
ramifications of defending free speech.\nReligions are not alone in having their martyrs. On 1 July, 1766, in
Abbeville in northern France, a young nobleman named Lefebvre de la Barre was found guilty of blasphemy. The
charges against him were numerous - that he had defecated on a crucifix, spat on religious images, and refused to
remove his hat as a Church procession went past. \nThese crimes, together with the vandalising of a wooden cross
on the main bridge of Abbeville, were sufficient to see him sentenced to death. Once La Barre's tongue had been
cut out and his head chopped off, his mortal remains were burned by the public executioner, and dumped into the
```

Solr can now be stopped by using the following command:

```
$ bin/solr stop -all
```

We need to load JSON files in Solr since the "Community" tweet data collection has data in JSON format.

Ebola Dataset

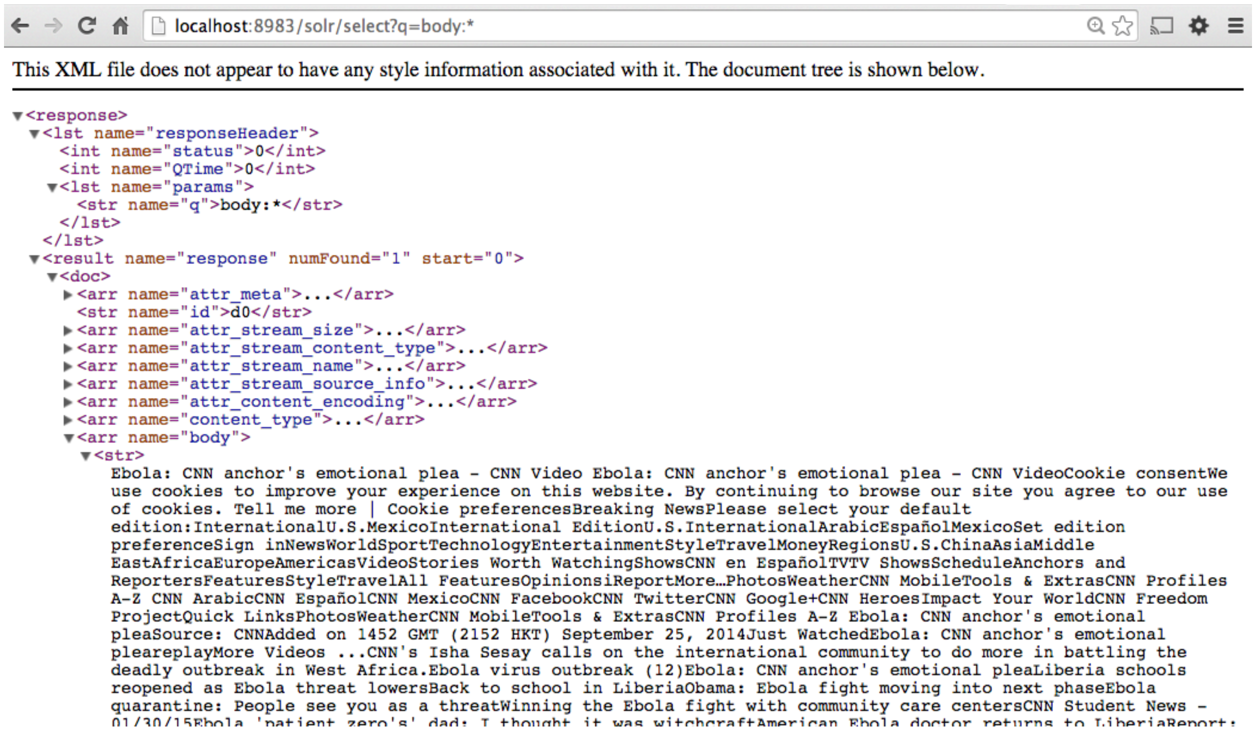
i. To generate the web pages from the Ebola tweets we ran the tweet_URL_archivingFile.py script to extract the content of the webpages present in the tweets. We obtained 74 web pages.

```
Vanessas-MacBook-Air-2:info vcedeno$ python tweet_URL_archivingFile.py z224t.csv

tweets is read from File
short Urls extracted:  169598
cleaned short URLs:  169523
Unique short URLs:  160900
Freq short URLs (>10):  133
/Library/Python/2.7/site-packages/requests/packages/urllib3/connectionpool.py:73
4: InsecureRequestWarning: Unverified HTTPS request is being made. Adding certif
icate verification is strongly advised. See: https://urllib3.readthedocs.org/en/
latest/security.html
```

ii. Then we used the following command to upload the files:

```
curl "http://localhost:8983/solr/update/extract?literal.id=d0&uprefix=attr_&fmap.content=body&commit=true" -F "myfile=@1.txt"
```

iii. Then, we were able to query the web pages in Solr:



localhost:8983/solr/select?q=body:*

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<response>
  ▼<lst name="responseHeader">
     <int name="status">0</int>
     <int name="QTime">0</int>
    ▼<lst name="params">
       <str name="q">body:*</str>
     </lst>
   </lst>
  ▼<result name="response" numFound="1" start="0">
    ▼<doc>
      ▶<arr name="attr_meta">...</arr>
       <str name="id">d0</str>
      ▶<arr name="attr_stream_size">...</arr>
      ▶<arr name="attr_stream_content_type">...</arr>
      ▶<arr name="attr_stream_name">...</arr>
      ▶<arr name="attr_stream_source_info">...</arr>
      ▶<arr name="attr_content_encoding">...</arr>
      ▶<arr name="content_type">...</arr>
      ▼<arr name="body">
        ▼<str>
           Ebola: CNN anchor's emotional plea - CNN Video Ebola: CNN anchor's emotional plea - CNN VideoCookie consentWe
           use cookies to improve your experience on this website. By continuing to browse our site you agree to our use
           of cookies. Tell me more | Cookie preferencesBreaking NewsPlease select your default
           edition:InternationalU.S.MexicoInternational EditionU.S.InternationalArabicEspañolMexicoSet edition
           preferenceSign inNewsWorldSportTechnologyEntertainmentStyleTravelMoneyRegionsU.S.ChinaAsiaMiddle
           EastAfricaEuropeAmericasVideoStories Worth WatchingShowsCNN en EspañolTVTV ShowsScheduleAnchors and
           ReportersFeaturesStyleTravelAll FeaturesOpinionsiReportMore…PhotosWeatherCNN MobileTools & ExtrasCNN Profiles
           A-Z CNN ArabicCNN EspañolCNN MexicoCNN FacebookCNN TwitterCNN Google+CNN HeroesImpact Your WorldCNN Freedom
           ProjectQuick LinksPhotosWeatherCNN MobileTools & ExtrasCNN Profiles A-Z Ebola: CNN anchor's emotional
           pleaSource: CNNAdded on 1452 GMT (2152 HKT) September 25, 2014Just WatchedEbola: CNN anchor's emotional
           pleareplayMore Videos ...CNN's Isha Sesay calls on the international community to do more in battling the
           deadly outbreak in West Africa.Ebola virus outbreak (12)Ebola: CNN anchor's emotional pleaLiberia schools
           reopened as Ebola threat lowersBack to school in LiberiaObama: Ebola fight moving into next phaseEbola
           quarantine: People see you as a threatWinning the Ebola fight with community care centersCNN Student News -
           01/30/15Ebola 'patient zero's' dad: I thought it was witchcraftAmerican Ebola doctor returns to LiberiaReport:
```

6.2. Hadoop

6.2.1 Installing Hadoop

i. Make sure Java 1.6+ is installed. [13]

ii. With Ruby installed, you can install Homebrew:

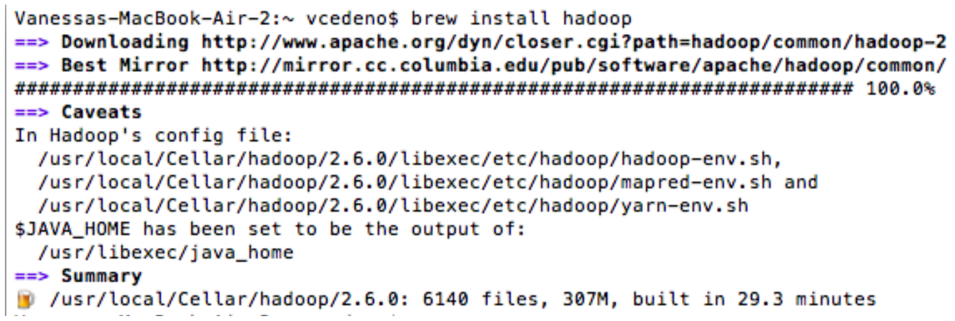
```
$ ruby -e "$(curl -fsSL https://raw.githubusercontent.com/
Homebrew/install/master/install)"
```

iii. Enable SSH , then generate and authorize SSH keys

```
$ ssh-keygen -t rsa -P ""
$ cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

iv. SSH into localhost: `ssh localhost`

v. Install Hadoop: `brew install hadoop`



```
Vanessas-MacBook-Air-2:~ vcedeno$ brew install hadoop
==> Downloading http://www.apache.org/dyn/closer.cgi?path=hadoop/common/hadoop-2
==> Best Mirror http://mirror.cc.columbia.edu/pub/software/apache/hadoop/common/
######################################################################## 100.0%
==> Caveats
In Hadoop's config file:
  /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop/hadoop-env.sh,
  /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop/mapred-env.sh and
  /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop/yarn-env.sh
$JAVA_HOME has been set to be the output of:
  /usr/libexec/java_home
==> Summary
🍺 /usr/local/Cellar/hadoop/2.6.0: 6140 files, 307M, built in 29.3 minutes
```

vi. Configure Hadoop files in /usr/local/Cellar/hadoop/2.6.0/libexec/etc/hadoop:

- hadoop-env.sh

  Add the following lines:

```
export JAVA_HOME="$(/usr/libexec/java_home)"
export HADOOP_OPTS="${HADOOP_OPTS} -
Djava.security.krb5.realm= -Djava.security.krb5.kdc="
export HADOOP_OPTS="${HADOOP_OPTS} -
Djava.security.krb5.conf=/dev/null"
```

- mapred-env.sh

  Add the following line:

```
export JAVA_HOME="$(/usr/libexec/java_home)"
```

Here's full configuration:

- yarn-env.sh

  Add the following line:

```
YARN_OPTS="$YARN_OPTS  -Djava.security.krb5.realm=OX.AC.UK
-Djava.security.krb5.kdc=kdc0.ox.ac.uk:kdc1.ox.ac.uk"
```

- core-site.xml
  The core-site.xml config file allows you to override any site-specific entries. Here's a sample file for development (uses localhost and pseudo-distributed mode):

```
<configuration>
     <property>
          <name>hadoop.tmp.dir</name>
          <value>/tmp/hadoop-${user.name}</value>
          <description>A base for other temporary
directories.</description>
     </property>
     <property>
          <name>fs.default.name</name>
          <value>hdfs://localhost:9000</value>
     </property>
</configuration>
```

- hdfs-site.xml
  By default this file is blank. The hdfs-site.xml config file allows you to override any Hadoop Distributed File System parameters. Here's a sample file:

```
<configuration>
          <property>
               <name>dfs.replication</name>
               <value>1 </value>
          </property>
</configuration>
```

- mapred-site.xml
  By default, this file is blank. The mapred-site.xml config file allows you to configure the MapReduce job tracker host & port. Here's a sample file:

```
<configuration>
          <property>
               <name>mapred.job.tracker</name>
               <value>localhost:9001</value>
          </property>
</configuration>
```

- yarn-site.xml

By default this file is blank. The yarn-site.xml config file allows you to customize YARN configuration. Full configuration is here. Here's a sample file:

```
<configuration>
    <property>
       <name>yarn.resourcemanager.resourcetracker.address</name>

       <value>$resourcemanager.full.hostname:8025</value>
       <description>Enter your ResourceManager hostname.</
description>
    </property>

    <property>

<name>yarn.resourcemanager.scheduler.address</name>
       <value>$resourcemanager.full.hostname:8030</value>
       <description>Enter your ResourceManager hostname.</
description>
    </property>

    <property>
       <name>yarn.resourcemanager.address</name>
       <value>$resourcemanager.full.hostname:8050</value>
       <description>Enter your ResourceManager hostname.</
description>
    </property>

    <property>
       <name>yarn.resourcemanager.admin.address</name>
       <value>$resourcemanager.full.hostname:8041</value>
       <description>Enter your ResourceManager hostname.</
description>
    </property>

    <property>
       <name>yarn.nodemanager.local-dirs</name>
       <value>/grid/hadoop/hdfs/yarn,/grid1/hadoop/hdfs/yarn</
value>
       <description>
             Comma separated list of paths. Use the list of
directories from $YARN_LOCAL_DIR.
             For example, /grid/hadoop/hdfs/yarn,/grid1/hadoop/
hdfs/yarn.
       </description>
    </property>

    <property>
       <name>yarn.nodemanager.log-dirs</name>
       <value>/var/log/hadoop/yarn</value>
       <description>
             Use the list of directories from $YARN_LOG_DIR.
             For example, /var/log/hadoop/yarn.
       </description>
    </property>
```

6.2.2 Running Hadoop

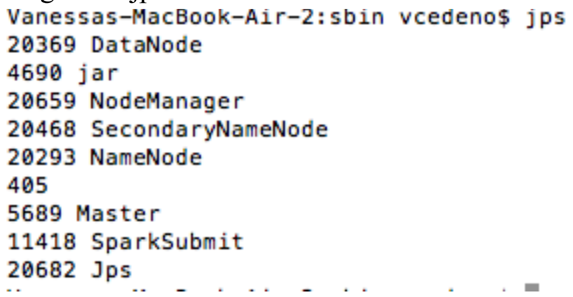
i. Prior to using Hadoop, you need to format the HDFS and initialize file & metadata storage:

```
$ hadoop namenode -format
```

ii. To run Hadoop execute:

```
$ cd /usr/local/Cellar/hadoop/2.6.0/libexec/sbin
$ ./start-dfs.sh
$ ./start-yarn.sh
```

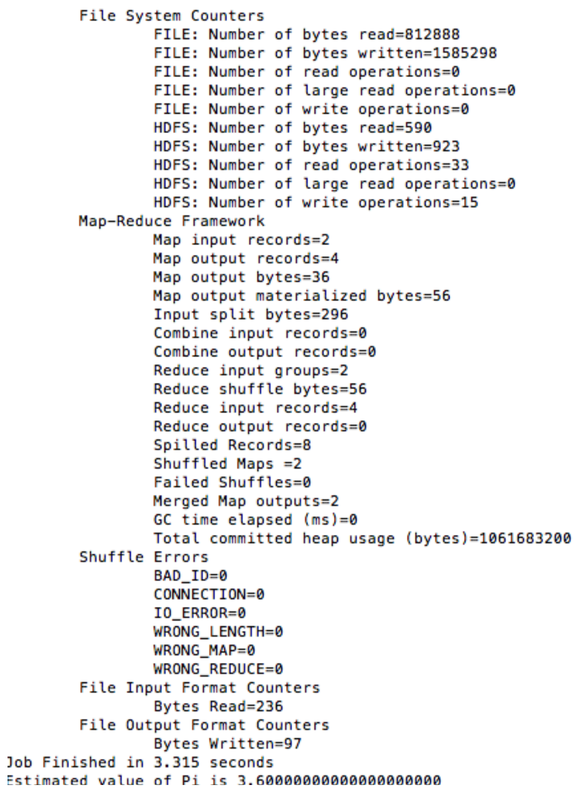
iii. To verify that Hadoop is running use: $jps



```
Vanessas-MacBook-Air-2:sbin vcedeno$ jps
20369 DataNode
4690 jar
20659 NodeManager
20468 SecondaryNameNode
20293 NameNode
405
5689 Master
11418 SparkSubmit
20682 Jps
```

iv. To test Hadoop there are a few sample MapReduce jobs that you can run to see whether your HDP instance is running and can accept jobs. Here's a sample MapReduce job to calculate the value of PI:

```
$ hadoop jar /usr/local/Cellar/hadoop/2.6.0/libexec/share/
hadoop/mapreduce/hadoop-mapreduce-examples-2.6.0.jar pi 2 5
```



```
                File System Counters
                        FILE: Number of bytes read=812888
                        FILE: Number of bytes written=1585298
                        FILE: Number of read operations=0
                        FILE: Number of large read operations=0
                        FILE: Number of write operations=0
                        HDFS: Number of bytes read=590
                        HDFS: Number of bytes written=923
                        HDFS: Number of read operations=33
                        HDFS: Number of large read operations=0
                        HDFS: Number of write operations=15
                Map-Reduce Framework
                        Map input records=2
                        Map output records=4
                        Map output bytes=36
                        Map output materialized bytes=56
                        Input split bytes=296
                        Combine input records=0
                        Combine output records=0
                        Reduce input groups=2
                        Reduce shuffle bytes=56
                        Reduce input records=4
                        Reduce output records=0
                        Spilled Records=8
                        Shuffled Maps =2
                        Failed Shuffles=0
                        Merged Map outputs=2
                        GC time elapsed (ms)=0
                        Total committed heap usage (bytes)=1061683200
                Shuffle Errors
                        BAD_ID=0
                        CONNECTION=0
                        IO_ERROR=0
                        WRONG_LENGTH=0
                        WRONG_MAP=0
                        WRONG_REDUCE=0
                File Input Format Counters
                        Bytes Read=236
                File Output Format Counters
                        Bytes Written=97
Job Finished in 3.315 seconds
Estimated value of Pi is 3.60000000000000000000
```

6.2.3 Monitoring Hadoop

To monitor HDFS, MapReduce, and Tasks in a pseudo-distributed single node mode:

- HDFS Administrator: http://localhost:50070
- ResourceManager Administrator: http://localhost:8088

6.2.4. Uploading the data into the DLRL Hadoop Cluster

Each team has their login and space on Centos in the head node of the DLRL Hadoop cluster. Our group account is cs5604_social. The following steps were followed to upload the Police department collection of tweets.

- `scp /Desktop/1.json cs5604s15_social@hadoop.dlib.vt.edu:/home/cs5604s15_social`
- Create a test folder under our directory: hadoop fs -mkdir /user/cs5604s15_social/test
- We made our test folder invisible to other groups with:
  `hadoop fs -chmod go+rx /user/cs5604s15_social/test`
- Upload the files to HDFS:
  `hadoop fs -copyFromLocal /home/cs5604s15_social/1.json /user/cs5604s15_social/test/1.json`
- To verify the file has been uploaded: hadoop fs -ls /user/cs5604s15_social/test

6.2.5 Apache Nutch

Apache Nutch is an open source Web crawler written in Java. It finds Web page hyperlinks in an automated manner, reducing maintenance work. It checks broken links, and creates a copy of all the visited pages for searching over. With Solr, visited pages from Nutch can be searched. [26]

The following tutorial describes how to use Nutch to crawl the URL's from the Police department data collection of tweets: [27]

i. We connected to our account in the Hadoop cluster:
```
ssh cs5604s15_social@hadoop.dlib.vt.edu
```
ii. We downloaded Nutch using the following command:
```
wget http://www.apache.org/dyn/closer.cgi/nutch/1.9/apache-nutch-1.9-bin.zip
```
iii. We edit the file conf/nutch-site.xml
iv. We extracted the URL's from the collection of tweets and created the file seed.txt
v. We ran the following command to start the crawling using the shared SOLR server:
```
bin/crawlurls/TestCrawl/http://preston.dlib.vt.edu:8980/solr 1 & disown
```

6.2.6 Loading AVRO files into HDFS

To load the AVRO files from local machine to HDFS we need to follow this procedure:

i. We connected to our account in the Hadoop cluster:
```
ssh cs5604s15_social@hadoop.dlib.vt.edu
```
ii. Secure copy all the AVRO files from the local machine to the cluster:
```
scp 1.avro cs5604s15_social@hadoop.dlib.vt.edu:1.avro
```

6.3 GraphX

6.3.1 Installing GraphX

i. Install Apache Spark from the official site: https://spark.apache.org/downloads.html. We are using spark-1.2.1-bin-hadoop2.4. [14]
ii. Run the following command in the Spark folder: bin/spark-shell
iii. To get started you first need to import Spark and GraphX into your project, as follows in your Spark shell:

```
import org.apache.spark._
import org.apache.spark.graphx._
import org.apache.spark.rdd.RDD
```

6.3.2 Uploading Tweets

The property graph is a directed multigraph (a directed graph with potentially multiple parallel edges sharing the same source and destination vertex) with properties attached to each vertex and edge. Each vertex is keyed by a unique 64-bit long identifier (VertexID). Similarly, edges have corresponding source and destination vertex identifiers. The properties are stored as Scala/Java objects with each edge and vertex in the graph. [15]

In this example we model a small social network with users and their followers modeled as vertices and "following" modeled as directed edges.

i. We begin by creating the property graph from arrays of vertices and edges. Paste the following code into the Spark shell:

```
val vertexArray = Array(
    (24546890L, ("burlingtonpd",9194)),
    (2670103226L, ("incident_wx",2)),
    (71357821L, ("chrisWBZ",1504)),
    (272985778L, ("makemeaSAMich12",291))
)

val edgeArray = Array(
    Edge(2670103226L, 24546890L,1),
    Edge(71357821L, 24546890L,1),
    Edge(272985778L, 24546890L,1)
)
```

ii. Using sc.parallelize construct the following RDDs from the vertexArray and edgeArray variables.

```
val vertexRDD: RDD[(Long, (String, Int))] =
sc.parallelize(vertexArray)
val edgeRDD: RDD[Edge[Int]] = sc.parallelize(edgeArray)
```

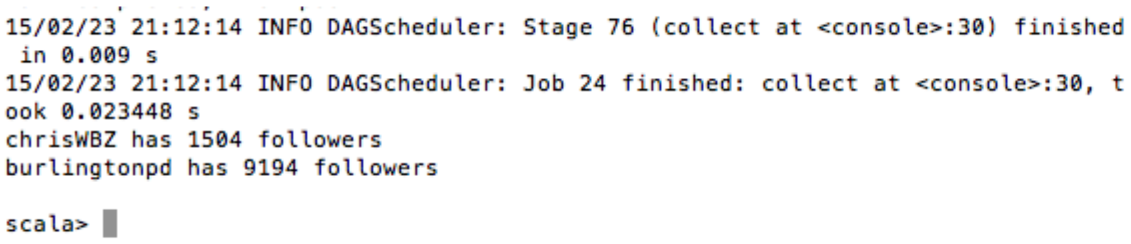
iii. Now to build a property graph, the basic property graph constructor takes an RDD of vertices (with type RDD[(VertexID, V)]) and an RDD of edges (with type RDD[Edge[E]]) and builds a graph (with typeGraph[V, E]). Run the following:

```
val graph: Graph[(String, Int), Int] = Graph(vertexRDD,
edgeRDD)
```

6.3.3 Viewing Graphs

i. Use graph.vertices to display the names of the users that have at least 1500 followers:

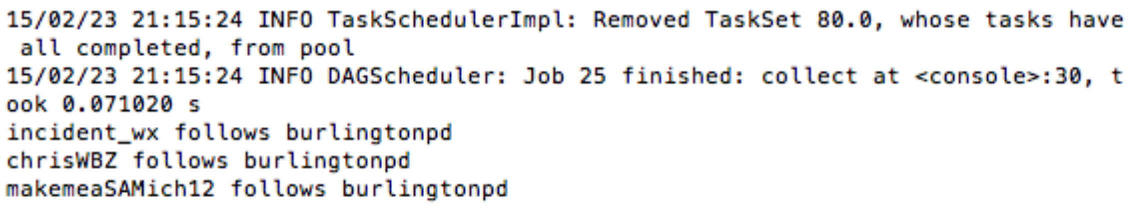
```
graph.vertices.filter { case (id, (name, followers)) =>
followers > 1500 }.collect.foreach
{
     case (id, (name, followers)) => println(s"$name has
     $followers followers")
}
```



```
15/02/23 21:12:14 INFO DAGScheduler: Stage 76 (collect at <console>:30) finished
 in 0.009 s
15/02/23 21:12:14 INFO DAGScheduler: Job 24 finished: collect at <console>:30, t
ook 0.023448 s
chrisWBZ has 1504 followers
burlingtonpd has 9194 followers

scala> 
```

ii. The EdgeTriplet class extends the Edge class by adding the srcAttr and dstAttr members which contain the source and destination properties respectively. The graph.triplets view can be used to display who follows who.

```
for (triplet <- graph.triplets.collect) {

println(s"${triplet.srcAttr._1} follows ${triplet.dstAttr._1}")

}
```



```
15/02/23 21:15:24 INFO TaskSchedulerImpl: Removed TaskSet 80.0, whose tasks have
 all completed, from pool
15/02/23 21:15:24 INFO DAGScheduler: Job 25 finished: collect at <console>:30, t
ook 0.071020 s
incident_wx follows burlingtonpd
chrisWBZ follows burlingtonpd
makemeaSAMich12 follows burlingtonpd
```

6.4 Apache Giraph

6.4.1 Installing Giraph

i. Choose the directory you wish to have Apache Giraph in and run:

```
git clone https://git-wip-us.apache.org/repos/asf/giraph.git

cd /System/Library/Frameworks/JavaVM.framework/Versions

sudo rm CurrentJDK

sudo ln -s /Library/Java/JavaVirtualMachines/jdk1.7.0_75.jdk/
Contents/ CurrentJDK
```

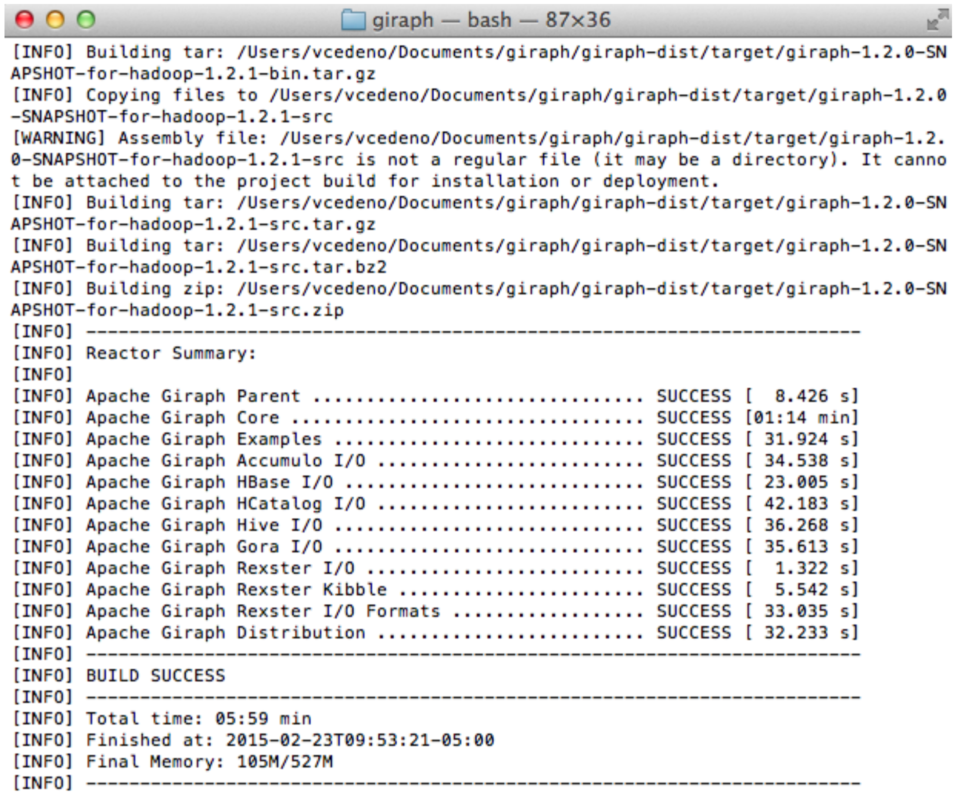
ii. Compile the code using Maven:

```
sudo mvn -Phadoop_2 -fae -DskipTests -Dhadoop=non_secure clean
install
```

iii. If you don´t have Maven you can install it from here: http://www.mkyong.com/maven/install-maven-on-mac-osx/

iv. Don't forget to set JAVA_HOME to the jdk7 home directory:

```
export JAVA_HOME=$(/usr/libexec/java_home -v 1.7)
```


giraph — bash — 87×36

```
[INFO] Building tar: /Users/vcedeno/Documents/giraph/giraph-dist/target/giraph-1.2.0-SN
APSHOT-for-hadoop-1.2.1-bin.tar.gz
[INFO] Copying files to /Users/vcedeno/Documents/giraph/giraph-dist/target/giraph-1.2.0
-SNAPSHOT-for-hadoop-1.2.1-src
[WARNING] Assembly file: /Users/vcedeno/Documents/giraph/giraph-dist/target/giraph-1.2.
0-SNAPSHOT-for-hadoop-1.2.1-src is not a regular file (it may be a directory). It canno
t be attached to the project build for installation or deployment.
[INFO] Building tar: /Users/vcedeno/Documents/giraph/giraph-dist/target/giraph-1.2.0-SN
APSHOT-for-hadoop-1.2.1-src.tar.gz
[INFO] Building tar: /Users/vcedeno/Documents/giraph/giraph-dist/target/giraph-1.2.0-SN
APSHOT-for-hadoop-1.2.1-src.tar.bz2
[INFO] Building zip: /Users/vcedeno/Documents/giraph/giraph-dist/target/giraph-1.2.0-SN
APSHOT-for-hadoop-1.2.1-src.zip
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary:
[INFO]
[INFO] Apache Giraph Parent .............................. SUCCESS [  8.426 s]
[INFO] Apache Giraph Core ................................ SUCCESS [01:14 min]
[INFO] Apache Giraph Examples ............................ SUCCESS [ 31.924 s]
[INFO] Apache Giraph Accumulo I/O ........................ SUCCESS [ 34.538 s]
[INFO] Apache Giraph HBase I/O ........................... SUCCESS [ 23.005 s]
[INFO] Apache Giraph HCatalog I/O ........................ SUCCESS [ 42.183 s]
[INFO] Apache Giraph Hive I/O ............................ SUCCESS [ 36.268 s]
[INFO] Apache Giraph Gora I/O ............................ SUCCESS [ 35.613 s]
[INFO] Apache Giraph Rexster I/O ......................... SUCCESS [  1.322 s]
[INFO] Apache Giraph Rexster Kibble ...................... SUCCESS [  5.542 s]
[INFO] Apache Giraph Rexster I/O Formats ................. SUCCESS [ 33.035 s]
[INFO] Apache Giraph Distribution ........................ SUCCESS [ 32.233 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 05:59 min
[INFO] Finished at: 2015-02-23T09:53:21-05:00
[INFO] Final Memory: 105M/527M
[INFO] ------------------------------------------------------------------------
```

6.4.2 Uploading Tweets and Web Pages

i. In the Hadoop directory, /usr/local/Cellar/hadoop/2.6.0/libexec/sbin:

```
./start-dfs.sh
./start-yarn.sh
```

ii. The following file represents the relationships between tweets and web pages. The file is uploaded to the HDFS to run the PageRank algorithm in Giraph. Each line has the format

```
[id,              favorites_value,[[dest_id, edge_value],...]]
[462381268001652736,    803,      []]
[462381267473170432,    595,      [[1,1]]]
[462381207058382848,    1029,     []]
[462381213303734272,    38,       [[2,1]]]
[462381224695062528,    0,        [[462029938401357824,1],
[3,1]]]
[1,                      0,       []]
[2,                      0,       []]
[3,                      0,       []]
```

Copy the input file to HDFS:

```
$HADOOP_HOME/bin/hadoop fs -mkdir /user

$HADOOP_HOME/bin/hadoop fs -mkdir /user/hduser

$HADOOP_HOME/bin/hadoop fs -copyFromLocal /Users/vcedeno/
Desktop/police_followers/graph.txt /user/hduser/tiny_graph.txt

$HADOOP_HOME/bin/hadoop fs -ls /user/hduser
```

6.4.3 Viewing graphs

The following command runs the Shortest paths algorithm:

```
bin/hadoop jar $GIRAPH_HOME/giraph-examples/target/giraph-
examples-1.2.0-SNAPSHOT-for-hadoop-2.5.1-jar-with-dependencies.jar
org.apache.giraph.GiraphRunner
org.apache.giraph.examples.SimpleShortestPathsComputation -vif
org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputForm
at -vip /user/hduser/tiny_graph.txt -vof
org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /user/
hduser/output/shortestpaths -w 1 -ca giraph.SplitMasterWorker=false
```

To print the output use the following command:

```
$HADOOP_HOME/bin/hadoop dfs -cat /user/hduser/output/shortestpaths/p*
| less
```

To run the PageRank algorithm the following command has to be executed:

```
bin/hadoop jar $GIRAPH_HOME/giraph-examples/target/giraph-
examples-1.2.0-SNAPSHOT-for-hadoop-2.5.1-jar-with-dependencies.jar
org.apache.giraph.GiraphRunner
org.apache.giraph.examples.SimplePageRankComputation -vif
org.apache.giraph.io.formats.JsonLongDoubleFloatDoubleVertexInputForm
at -vip /user/hduser/tiny_graph.txt -vof
org.apache.giraph.io.formats.IdWithValueTextOutputFormat -op /user/
hduser/output/outPageRank -w 1 -mc
org.apache.giraph.examples.SimplePageRankComputation\
$SimplePageRankMasterCompute -ca giraph.SplitMasterWorker=false
```

To print the output use the following command:

```
HADOOP_HOME/bin/hadoop dfs -cat /user/hduser/output/outPageRank/* |
less
```

PageRank output:

Table 3: PageRank output for tweets/webpages

Tweet/Webpage ID	Importance value
462381213303734272	0.017
462381207058382848	0.017
462381267473170432	0.017
2	0.031
462029938401357824	0.024
462381224695062528	0.017
1	0.031
3	0.024
462381268001652736	0.017

After defining the formula and algorithm that we will be using to calculate the scoring value, we will upload the collection in Giraph to evaluate performance.

6.5. Conversion of JSON to AVRO format

1. Download Avro-tools from the following URL:
http://supergsego.com/apache/avro/avro-1.7.7/java/avro-tools-1.7.7.jar
2. Copy the JAR file to your home directory.
3. Run the following command in the command line:
java -jar ~/avro-tools-1.7.4.jar
4. Run the following command to convert JSON to binary AVRO format:

```
java -jar ~/avro-tools-1.7.4.jar fromjson --schema-file isr.avsc
20140503-20140801_j2mr9y1xw7-2014_05_03_00_00_activities.json >
social_networks.avro
```

6.6. Inventory of Repository Files

ReportSN.pdf:
CS5604, Spring 2015 class project report.

PresentationSN.pdf:
CS5604 Spring 2015 class project presentation.

main.py:
driver script that takes tweet file, user file and output importance for each tweet message in AVRO format

multi_convert.py:
convert JSON objects to AVRO format. Notice that the JSON objects should contain user information as well

util.py:
fucntions that is called by main.py

basic_classes.py:
static mention class

Webpages Graphx Score Files

graph.py:
Extracts the vertex and edges from the cleaned web pages AVRO file
graphVertex.py:
The IDand url of the webpages from the AVRO file to be used by graphEdges.py
graphEdges.py:
Find connections between out links to initial webpages, to add to edges.scala
vertex.scala:
Produced by graph.py in GraphX format, contains the web page graph vertex
edges.scala:
Produced by graph.py in GraphX format, contains the web page graph edges
rank.scala:

Runs the GraphX PageRank algorithm in the web page graph

avro.py:

Produces the social importance score for webpages in AVRO files

Public Repo Address:
VTechWorks
Private Dev Repo Address:
git@bitbucket.org:0xffff/cs5604project.git

7. Evaluation

Most (if not all) of the tweet ranking approaches are considering the content-based metrics along with the social network metrics. As our work considers the social network concepts and metrics without content-based aspects, therefore it is not logical to compare our approach by itself against these approaches. It is difficult to locate the source code of other approaches to compare with. Most of them are not accessible online or not even implemented.

In order to evaluate our standalone approach, we need to consider the other teams work as well (the content-based ranking teams). We will use the TREC 2011 Microblog Dataset and gold standard (NIST). This dataset is coupled with 49 queries. Our evaluation will be as it follows.

- Modify the dataset to be compatible with the IDEAL project formatting (AVRO).
- Use the Whole IDEAL IR System (excluding the SN ranking), to retrieve the tweets for the 49 queries.
- Calculate the similarities between the retrieved ranked tweets with the golden solution for each query.
- Apply our SN approach/ranking system on the retrieved tweets.
    - If the SN approach is valuable, it should improve the similarity between the final retrieved tweets and the golden solution.
    - Otherwise, it will make no difference or in the worst case it will decrease the similarities.

For example, for Query 1, let's say

- IDEAL IR System retrieved Tweet IDs <1, 2, 5, 3, 10, 8, 9, 11, 12, 7>
- The Golden Solution is <1, 2, 3, 5, 7, 8, 9, 10, 11, 12>
- Therefore, the similarity is : 60% "3, 5, 7 and 10" are misplaced.
- Now, we will apply the SN ranking system on the 10 retrieved tweets to enhance the output.
- [After Applying SN] Final Tweets IDs < 1, 2, 3, 5, 7, 8, 9, 11, 12, 10>
- That means only "Tweet ID # 10" is misplaced. Therefore the new similarity is: 90%
- This indicates that the Social Network (SN) approach improves the IDEAL IR system ranking with ~2x fold.

Due to the time limitation, we didn't do the evaluation part. But, it is feasible and easily to be implemented. It just needs time and the formatting conversion, of the TREC dataset, might be tedious.

8. Summary, Conclusions, Future Work

In terms of design, our tweet importance rating algorithm is still very fundamental. From our observation, user's follower count has a significant impact on the final importance value. On the other hand, User Importance Value (UIV) does not affect the importance value as much. This is because most users in our data collection are not closely connected. Instead of using the same weights, regression analysis could be done for calculating weights.

Tweet content can be used to help calculating importance score. For example, our algorithm does not detect trending tweets/topics. However, tweets that are closed affiliated with trending topics or events could be favored over ones that have similar content but are not created during the event time frame.

In terms of implementation, our scripts are written in Python, an interpreter based language that has great library support. However, one may consider porting the script to C or C++ for better performance. In addition, the most computationally intensive part in the script is implemented with standard Python multiprocessing library. However, since the implementation of native Python data structures are made thread safe via locks, using multiple processes to update a shared data structure can cause high contention. Constructing edges, calculating edge weights and calculating UIV are not implemented with multiprocessing library. We made this decision because the frequent write operation for a shared memory location can cause contention and race condition. In the future work, one may consider using MapReduce framework, especially considering all data that are stored in HBase.

There are at least these two steps one can take advantage of MapReduce. To start, URL extraction and short URL to long URL expansion can take advantage of MapReduce since it is an embarrassingly parallel task. The mapper can be each individual tweets, intermediate value <K,V> pairs can be <short_URL, tweet_id> , reducer can take such intermediate value and output <tweet_id, long_url> as final output.

UIV calculation can also take advantage of MapReduce framework since UIV of a user is the summation of all edge weights. In this case, Once all edge weights are calculated, one can write a simple mapper that takes an edge object, which contains the user ids at both ends of the edge, and the value of the weight. Intermediate generated by mapper can be <user_id, edge_weight>. Since MapReduce sorts intermediate values according to key values, all the edge weights belongs to the same user are going to be processed by the same reducer. In this case, a reducer thread will simply calculate the summation of all the edge values, and emit <user_id, UIV> as output.

Our schema contains two extra attributes of Followers Count and List Count that are required by our SN ranking approach. In order to handle other data in different collections, the steps IDEAL can incorporate is to add these attributes. By doing this, our approach usage should be used straightforward with no difficulties. They would benefit from our work by building a graph and computing different importance scores.

References

1. Events Archive. Project History Overview. Retrieved 16:30, February 11, 2015, from http://www.eventsarchive.org/?q=node/70

2. Tianyi Wang et al., 2011. "Understanding Graph Sampling Algorithms for Social Network Analysis", 31st International Conference on Distributed Computing Systems Workshops (ICDCSW), Pages 123-128, Minneapolis, MN, USA.

3. How we analyzed Twitter social media networks with NodeXL, http://www.pewinternet.org/files/2014/02/How-we-analyzed-Twitter-social-media-networks.pdf

4. Seth A. Myers et al., 2014. Information Network or Social Network?:The Structure of the Twitter Follow Graph, Proceedings of the companion publication of the 23rd international conference on World Wide Web companion, Pages 493-498, ISBN: 978-1-4503-2745-9, Geneva, Switzerland.

5. Srijith Ravikumar, Raju Balakrishnan, and Subbarao Kambhampati. 2012. Ranking tweets considering trust and relevance. In Proceedings of the Ninth International Workshop on Information Integration on the Web (IIWeb '12). ACM, New York, NY, USA, pages 4. http://doi.acm.org/10.1145/2331801.2331805

6. Yajuan Duan, Long Jiang, Tao Qin, Ming Zhou, and HeungYeung Shum. 2010. An empirical study on learning to rank of tweets. In Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10). Association for Computational Linguistics, Stroudsburg, PA, USA, 295303. h ttp://dl.acm.org/citation.cfm?id=1873815

7. Serge Abiteboul, Milhai Preda, Gregory Cobena. 2003. Adaptive On-line Page Importance Computation, Proceedings of the companion publication of the 12th international conference on World Wide Web companion, Pages 280-290, Budapest, Hungary. http://dl.acm.org.ezproxy.lib.vt.edu/citation.cfm?id=775152

8. Patrick Dlogan, *python-json 3.4*, http://sourceforge.net/projects/json-py/. Accessed 2-10-2015.

9. John Hunter, *matplotlib*, http://matplotlib.org. Accessed 2-12-2015.

10. *NetworkX*, https://networkx.github.io. Accessed 2-12-2015.

11. *Graphviz - Graph Visualization Software*, retrieved from http://www.graphviz.org. Accessed 2-13-2015.

12. *Solr in 5 minutes*, retrieved from http://www.solrtutorial.com/solrin5minutes.html. Accessed 1-26-2015.

13. Big Data and Hadoop. April 21, 2014. Installing Hadoop on Mac OS X Mountain Lion 10.8.5 bit. R etrieved 11:00, February 20, 2015 from http://glebche.appspot.com/static/hadoop-ecosystem/hadoop-hive-tutorial.html#hadoop-installation-mac-osx

■

14. Spark. Quick Start. Retrieved 15:00, February 20, 2015 from https://spark.apache.org/docs/latest/quick-start.html

15. Ampcamp. GraphX. Retrieved 10:00, February 23, 2015 from http://ampcamp.berkeley.edu/big-data-mini-course/graph-analytics-with-graphx.html#constructing-an-end-to-end-graph-analytics-pipeline-on-real-data

16. Apache Giraph. Quickstart. Retrieved 15:00, February 23, 2015 from http://giraph.apache.org/quick_start.html

17. Hadoop application architectures. Chapter 4. Graph processing on Hadoop. Retrieved 17:00, February 23, 2015 from https://www.safaribooksonline.com/library/view/hadoop-application-architectures/9781491910313/ch04.html

18. Apache Giraph. Introduction. Retrieved 18:00, February 23, 2015 from http://giraph.apache.org/intro.html

19. Meeyoung Cha, Hamed Haddadi, Fabricio Benevenuto, Krishna P. Gummadi. 2010. Measuring User Influence in Twitter: The million follower fallacy. In Proceedings of the Association for the Advancement of Artificial Intelligence. USA.

20. Haewoon Kwak, Changhyun Lee, Hosung Park, and Sue Moon. 2010. What is Twitter, a Social Network or a News Media? Pages 591-600. In Proceedings of the 19th international conference on World Wide Web.

21. Eytan Bakshy, Winter A. Mason, Jake M. Hofman and Duncan J. Watts. Everyone's an Influencer: Quantifying Influence on Twitter. 2011. In Proceedings of the ACM WSDM 2011. Pages 65-74. Hong Kong.

22. The Engineering Behind Twitter's New Search Experience. By Twitter Search. Retrieved 10:00, March 10, 2015 from https://blog.twitter.com/2011/engineering-behind-twitter%E2%80%99s-new-search-experience

23. Hang Li. 2011. A Short Introduction to Learning to Rank. IEICE TRANS. INF. & SYST., VOL.E94–D, NO.10

24. Page Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford University.

25. Solrpy project Retrieved March 22, 2015 from https://code.google.com/p/solrpy/

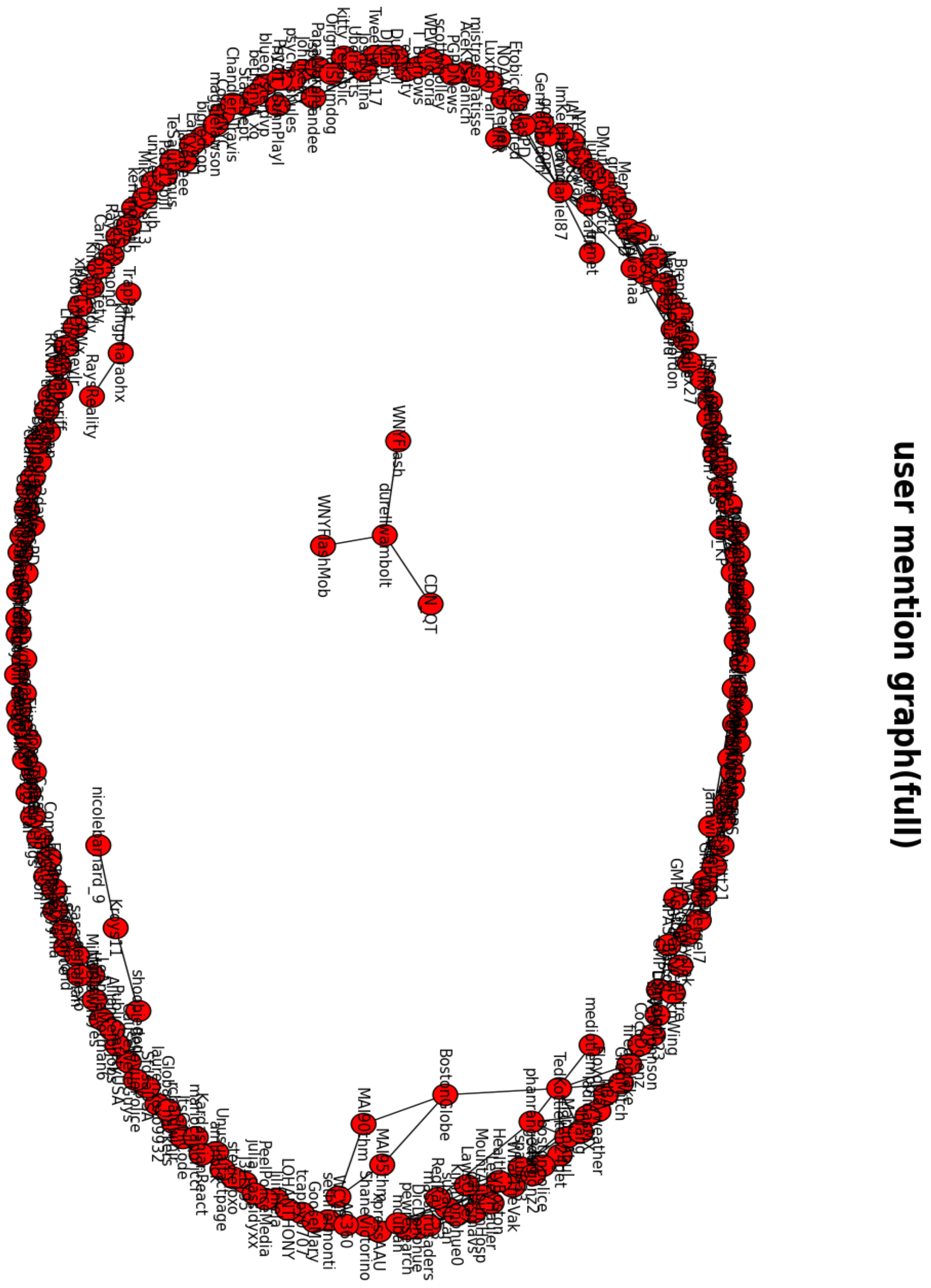
26. Nutch Tutorial. Introduction. Retrieved 20:00, March 27, 2015 from https://wiki.apache.org/nutch/NutchTutorial

27. Nutch Tutorial. Retrieved 14:00, March 23, 2015 from https://scholar.vt.edu/access/content/group/5508d3d6-c97d-437f-a09d-2cfd43828a9d/Tutorials/Nutch%20Tutorial.pdf

28. Reading and Writing Avro Files From the Command Line, Retrieved March 28, 2015 from http://www.michael-noll.com/blog/2013/03/17/reading-and-writing-avro-files-from-the-command-line/

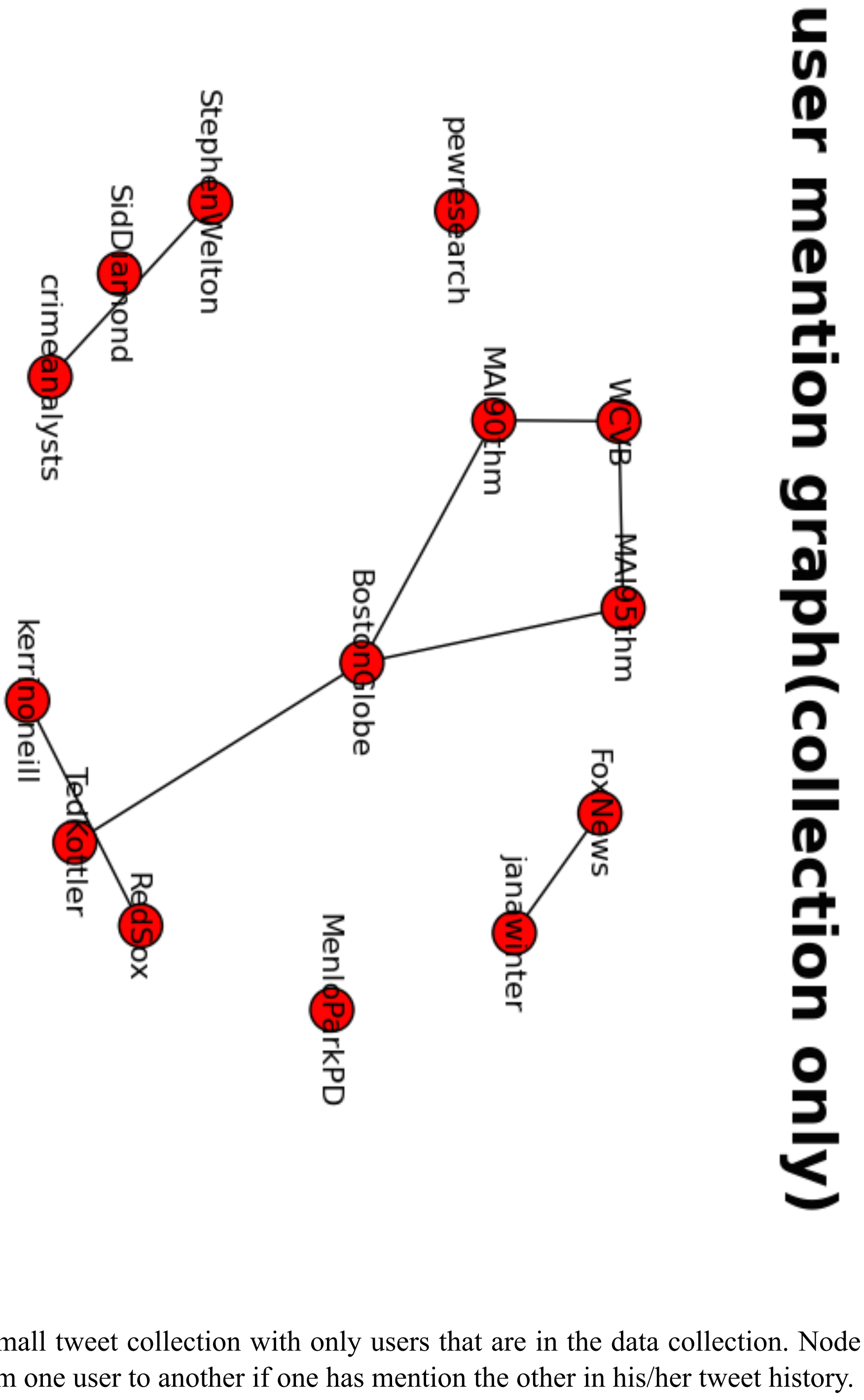
29. Spark. Hardware Provisioning. Retrieved May 12, 2015 from https://spark.apache.org/docs/latest/hardware-provisioning.html

Appendix A



User graph in a small tweet collection with any users. Node are users. An edge is drawn from one user to another if one has mention the other in his/her tweet history.

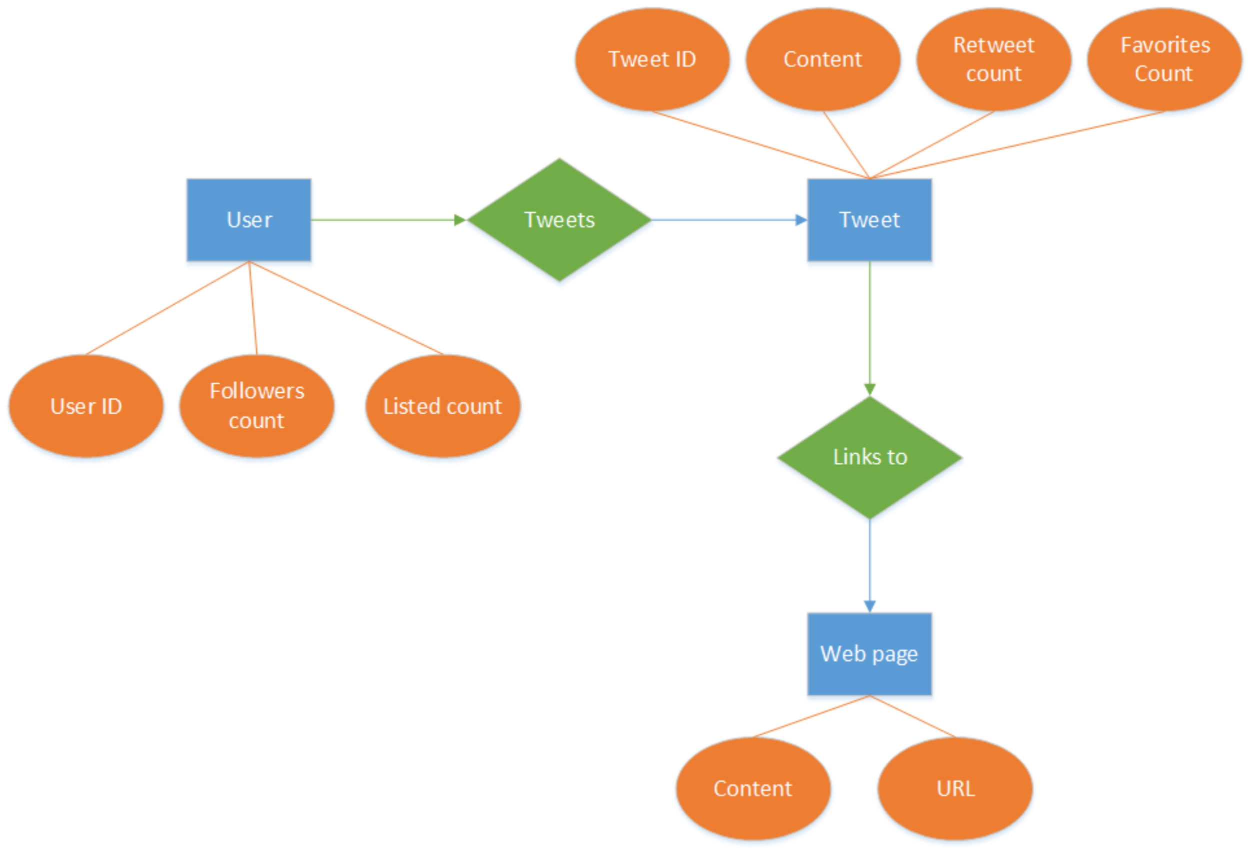
Appendix B



User graph in a small tweet collection with only users that are in the data collection. Node are users. An edge is drawn from one user to another if one has mention the other in his/her tweet history.

Appendix C

Entity Relationship diagram



The Entity Relationship diagram shows the relation in the Social Network graph. We have different entities as shown above. The attributes like User ID, Followers count and Listed count pertain to the User and similarly, each Tweet has its own attributes. The User tweets a tweet. A tweet has a link to the Web Page where the attribute of the webpage consist of the URL and the Content. Each tweet also has a Tweet ID, Content, Retweet count and Favorites count.

Appendix D

Social Network Team output AVRO schema:

```
{
    "namespace": "cs5604.tweet.social",
    "type": "record",
    "name": "TweetSocial",
    "fields": [
        {
            "name": "doc_id",
            "type": "string"
        },
        {
            "doc": "analysis",
            "name": "importance",
            "type": "double",
            "default": 0
        }
    ]
}
```

The output contains 2 values: tweet_id in string type and its corresponding importance value in double precision floating point value.