Community-Scale Online Network Analysis of Social Data Streams

Progress Report for

ENGR 497 Global Design Project I

Ammar Raşid

Prof. Ahmet Bulut

College of Engineering

İstanbul Şehir University

Fall-2017

Abstract:

The project is about collecting data pertaining to Şehir community and analyzing them in holistic manner. The main three aspects of the project are, first, that the data are collected in online streams and not dumped into offline snapshots for analysis. Secondly, network analysis is applied while pertaining the entirety of the data, that is, we do not aim to analyze different types of data separately but rather integrate them. Thirdly and lastly, processed data are visualized with interactive representations.

Table of Contents

Abstract 2
Introduction 4
Background and Literature Review 5
A BAD Demonstration: Towards Big Active Data 5
FlashView: An Interactive Visual Explorer for Raw Data 7
Adaptive online event detection in news streams 9
Report Body 11
Microblogs Data: Twitter 11
Generic Social Media: Facebook 12
Spark Streaming 13
System Workflow 13
Scrappers 14
Analyzers 15
Visualization 16
Social Sciences aspects 18
References 19

Introduction

The amount of high-throughput data made available in the past decade coinciding with an exponential increase in the computational power has provided researchers with the opportunity to make holistic analysis without taking data out of their contexts. The sheer visualization of community networks has the potential of revealing latent problems in early stages and predicting others before they take place. Operational Intelligence (OI) leverages fully contextual data and social networks to observe, analyze and make prudent decisions. Analyzing the common interest, demands, objections and other sentiments-related trends of the mass are manifested in the political application of OI. The motivation for our project is to crawl social media data of members of Sehir community (Students and staff), integrating them with structured data from university's database (e.g. field of study) and analyzing the community in holistic manner. One caveat though is that we do not brainstorm the tools available to us and look for problems to apply them too, but rather we seek advice and guidance from social science departments regarding the problems that they think are likely to happen or phenomena worth studying. We aim to have an online system, continuously collecting and integrating contextual data and keeping track of the evolution of the community network. A system with this momentum once ready is passed through an interactive visualization pipeline. Observation and analysis are a recurrent cycle forming the core of our system.

Background and Literature Review

A BAD Demonstration: Towards Big Active Data

(Jacobs, et al., 2017). summarized three key requirements for an effective 'Big Active Data' analysis system; First and foremost, significant importance of data could only be detected and analyzed in the entirety of the data context and their relationships to other data items. Second, data should be 'enrichable' using other relevant data to make up for important absent information. Third and Last, active data should both be processed on the fly, thus the name, and also in retrospective manner as the evolution of the data in a time-line per se has a significant value. Their proposed BAD system is mostly an extension to Event Condition Action (ECA) rules and Triggers in that it overcomes their two key limitations. First, in contrast to ECA where when event E happens, action A is performed, BAD provides optimizable way of detecting complex events of interests. Second, BAD scales to a degree required for Big Data that hasn't been achieved by previous implementations of Triggers or ECA rules . They extended Apache AstrexDB with a new feature they called Channels. Channels are versions of queries that are instantiated with parameters and executed continuously starting at their creation.

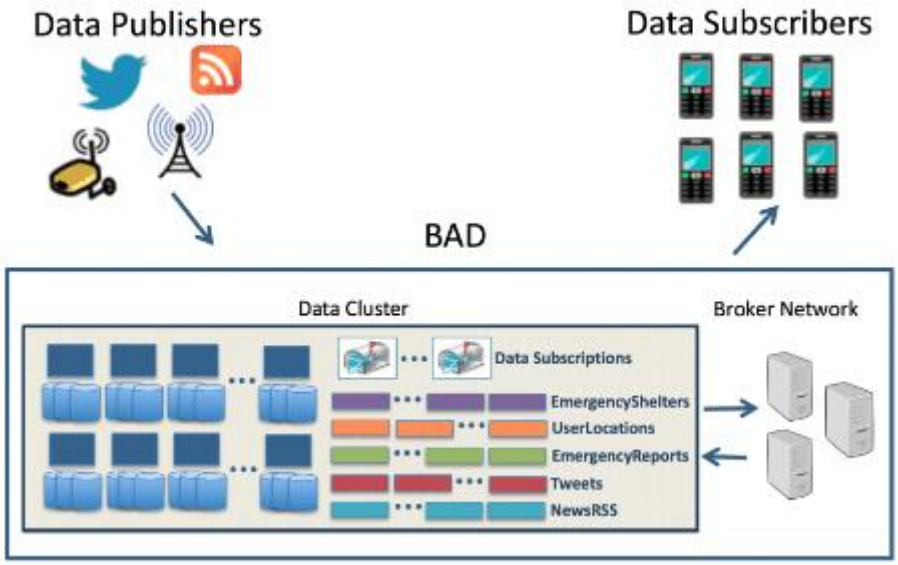

Figure 1 Big Active Data (BAD) System Overview

They also introduced a BAD Broker Network coordinate the data flow between the data cluster and the client. BAD Broker Network consists of Broker Coordination Service and BAD broker nodes. Brokers, as the name suggests, act as mediators between the client (handled by a client-facing part) ,for handling client registration, managing subscriptions and delivering results for those subscriptions, and the data cluster, for handling interactions with the Asterix backend (handled by Asterix-facing part). The common ground between our project and this work is the 'Big Active' data, as their work scales to the large number of subscribers, though the number is not mentioned, while maintaining the feasibility of the system and not compromising any of the three key requirements they summarized for effective Big Active Data systems. However, we are not really into data subscribers and notifications. What matters for us is crawling the published data. In contrast to their work, we focus only on data published by individuals in a specific community, i.e. Sehir Community.

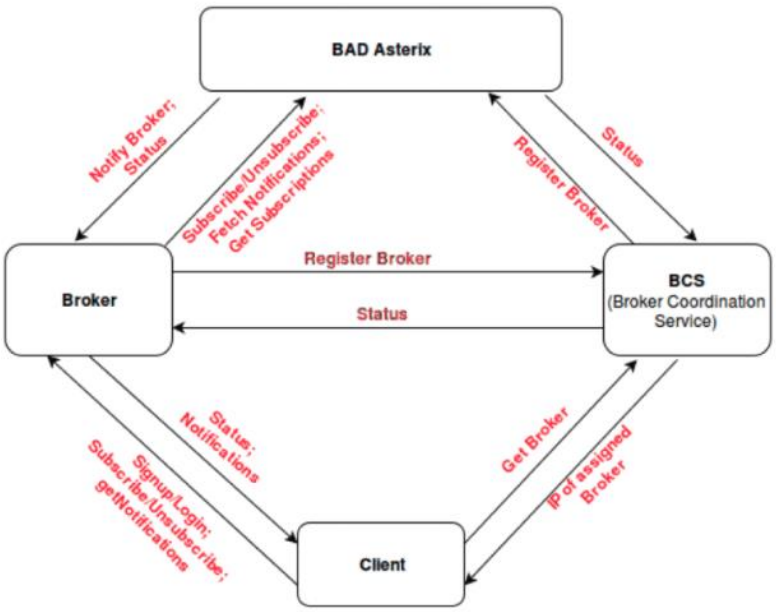

Figure 2 Broker Data Flow

FlashView: An Interactive Visual Explorer for Raw Data

In order to query big data in near-real-time speed in continuous manner, a trade off between loading speed or querying speed; processing and indexing raw data takes significantly long time, but once the data is loaded and indexed, querying is almost instantaneous. On the other hand, we could skip the data-loading step to save the time consumed in this bootstrapping step, but then queries would suffer from tediously slow performance. (Pang, Wu, Chen, Chen, & Shou, 2017). address this dilemma by introducing FlashView- an interactive visual explorer for raw data made to help analysts get initial acquaintance with the data they are given to analyze. The key point in providing a real-time response for data aggregation, without loading the data but rather manipulating raw data files directly, is leveraging approximate query processing techniques. Since FlashView provides fast but accuracy-compromised queries, an error metric and bound had to be specified. FlashView executes queries on randomly selected samples from the data and uses Hoeffding Inequality to estimate the bounds for the approximation.

$$P\{|\bar{Y} - A| < \varepsilon\} > 1 - 2e^{\frac{-2n\epsilon^2}{(b-\alpha)^2}}$$

Equation 1 Hoeffding Inequality

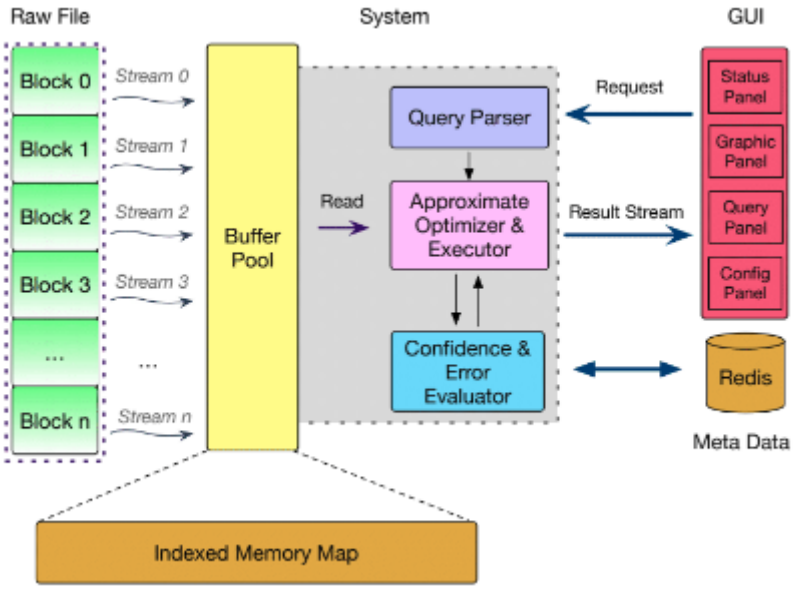

Figure 3 FlashView Architecture

The interesting component in their paper to our project is how they regulate the Data Flow in SQL trees. They use caching to keep track of active queries, so that if a subsequent query depends on one of the active queries, the later query could branch out from the active query directly and not from the root query.

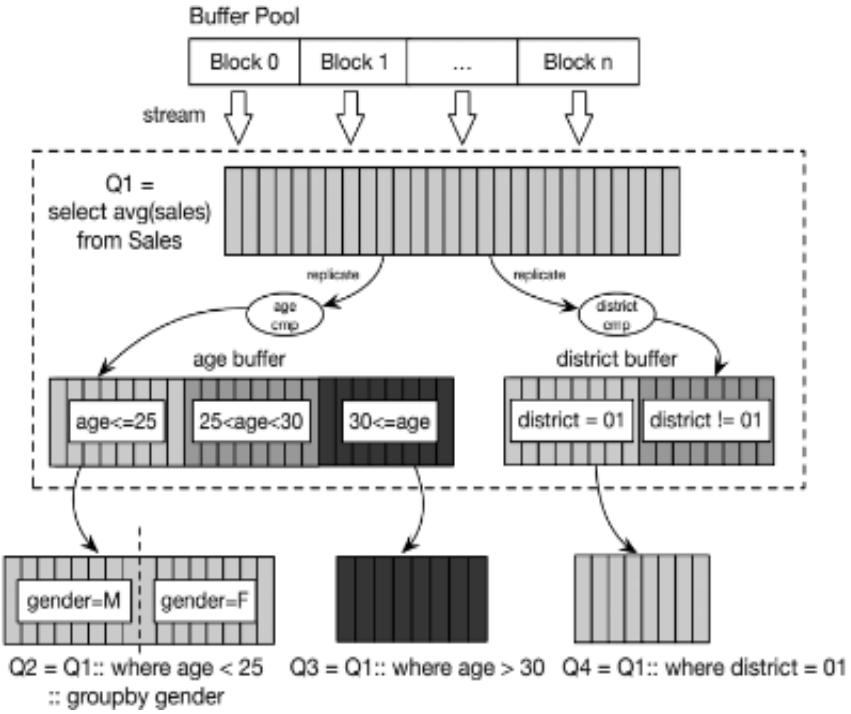

Figure 4 Data Flow of SQL Tree

Adaptive online event detection in news streams

Though the scarcity of big contextual data had been the main hindrance facing analysts couple of decades ago, the amount of these data is growing in an overwhelmingly rapid way. Big data are not just overwhelming for analysts but also to Data Management Systems (DMS); keeping record of big contextual data over long timelines and applying machine learning models on the top of that could be thwarted by poor performance in querying speed and full-patch variations of machine learning models. (Hu, Zhang, Hou, & Li, 2017) address the problem of event detection through capturing news stream data and feeding them to single-pass online clustering model, yet with some intermediary processing of the data, making their method significantly faster than standard single-pass online clustering and even with higher Normalized Mutual Information (NMI) and F1 scores. The method this paper provides is pertaining to our project in that it processes time-sliced textual (unstructured) data streams while maintaining feasible performance speed and accuracy. There are two main keys to the significant superiority of the method proposed in this paper over the standard single-pass online clustering algorithm. First, before applying single-pass online clustering , they use K-means clustering to cluster similar words based on their skip-gram word embeddings.

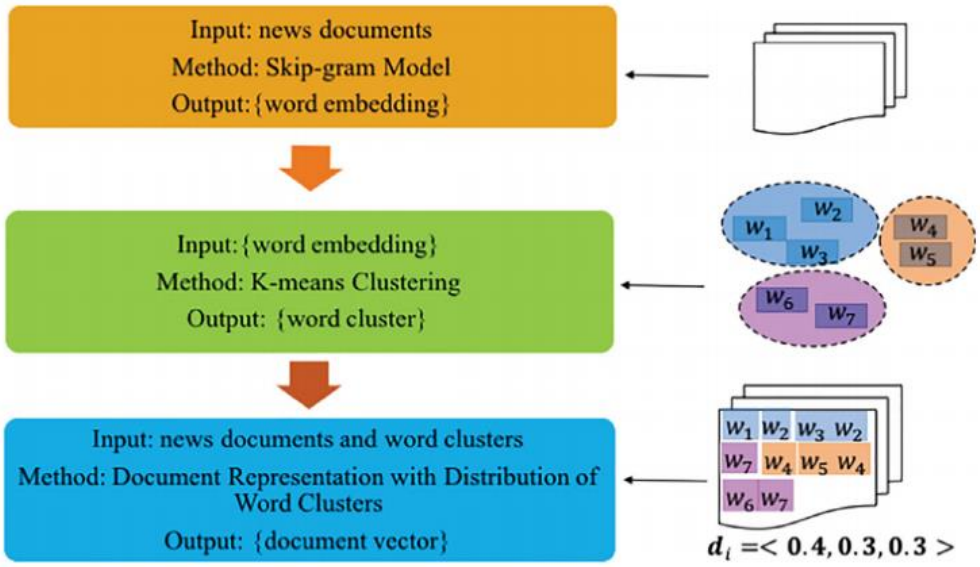

Figure 5 Process of document representation

Second, after obtaining document vectors from the distribution of word clusters, they slice the documents set to subsets of similar time period. Having relevant subsets of data, they parallelize the single-pass online clustering on each of the subsets (time periods).

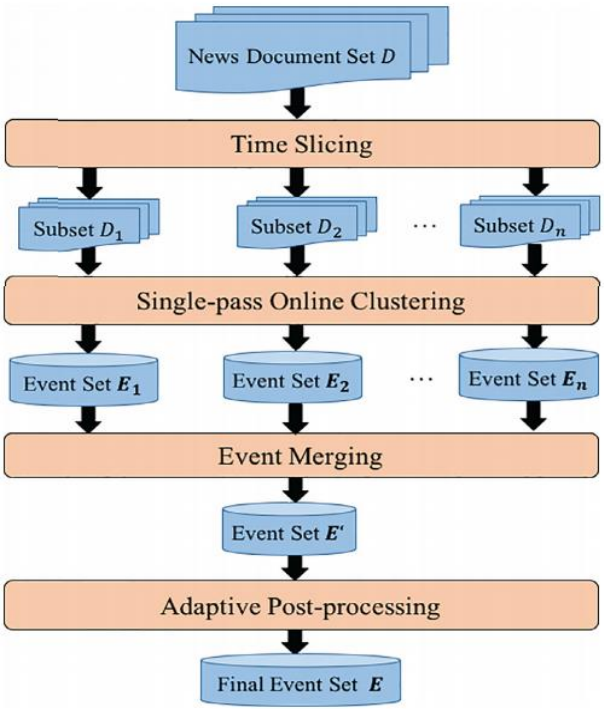

Figure 6 Process of adaptive online clustering method

In order to polish the precision of the clustering algorithm -eliminate the noise associated with similar news documents belonging to different events- they use variance to determine which clusters are 'high-quality' (i.e. with low variance) and which are not based on a threshold parameter δ. They use F1 and NMI scores as metrics for the recall-precision performance assessment. Adaptive online event detection has the momentum of capturing, not only events from news streams, but also trends and networks from micro blogs (e.g. Twitter) and flattening the evolution of these trends and networks over a time-line.

Report Body

Microblogs Data: Twitter

Microblogs such as Twitter provide invaluable data in significant amounts and rich contexts. Not only that, but also Twitter has an open-source API for developers to access and analyze these data. Such versatile data have been used for many types of analyses. Just to name a few, (Lekha R. Nair, 2005) have used data streams from Twitter for job search using machine learning categorization algorithms based on hashtags and Spark Streaming for real time data collection. (Marcus, et al., 2011) in MIT CS-AI Lab built two systems for programmers upon Twitter API- TweeQL and SQL-Like stream processor that provides streaming semantics and user-defined functions for extracting and aggregating tweet-embedded data. And for end-users, they build TwitInfo- a timeline-based visualization of events in the tweet stream linked to raw tweet text, sentiment analysis and even maps. (Jin, Zhu, Jin, & Arora, 2014) opted to visualize the 'SentimentRiver' where the balance between positive and negative sentiments is visualized as a 'current' of three layers, each representing a sentiment –positive, neutral and negative.

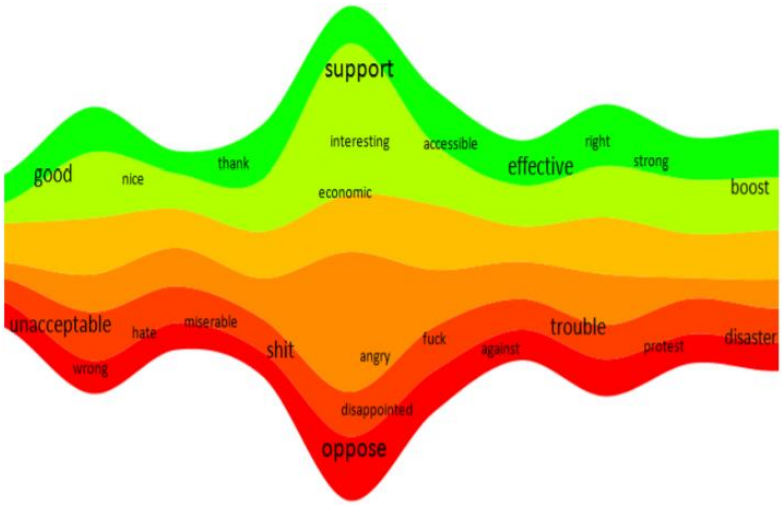

Figure 7 SentimentRiver with labels

One of the main challenges facing Twitter-based analyses is the sparsity of publicly available tweets in specific area and Twitter users mostly do not share information about their locations. (Wei, Sankaranarayanan, & Samet, 2017), utilize geotagging procedures to estimate the location for unknown-location users by examining the publicly-known locations of their friends on Twitter, and thus eliminating the aforementioned challenge. They use TwitterStand, a news tweet processing system that collects tweets and classifies them as

news or not that was developed by them, to aggregate news tweets into clusters, determine their geographical focus and display them on an accessible map-query interface.

Generic Social Media: Facebook

Although Facebook has the potential of providing even more valuable data than twitter, especially for friend-friend relationship networks. In fact, Twitter as a microblog is not used for casual social 'posts' as much as Facebook and, therefore, Facebook with features like, likes, comments, shares, events, pages and groups provides richer context for community-scale analysis. However, Twitter is far more 'opensourced' than Facebook. (Chen, et al., 2016), gave an overview of the system Facebook use in realtime data processing. They identified five design-related decisions regulating ease of use, performance, fault-tolerance, scalability and correctness.

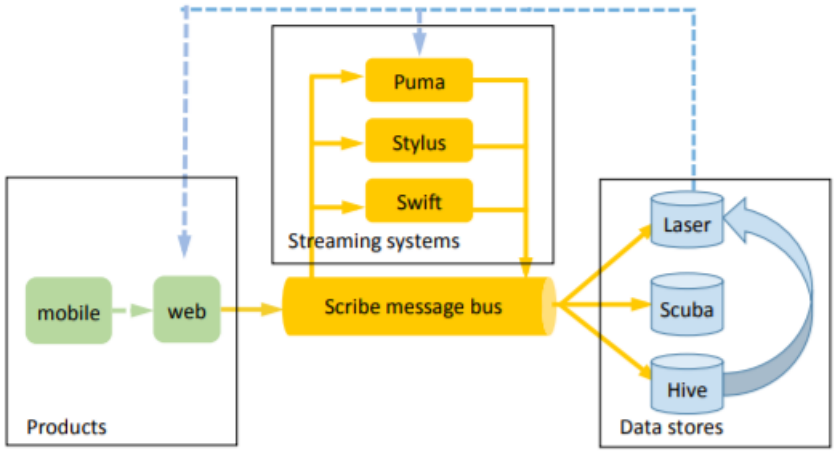

Figure 8 Realtime data processing system overview

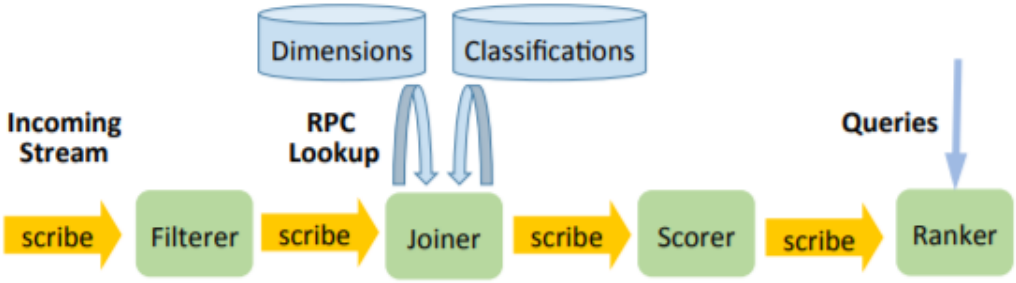

Figure 9 Computing "trending" events with a streaming application

Spark Streaming

Spark is an opensource computing-cluster that has a long list of applications including machine learning (MLlib), network analysis and visualization (GraphX) and realtime data collection and analysis (Kafka and Spark Streaming).

Figure 10 Overview of data flow in Spark Streaming

Figure 11 Overview of Spark Streaming application pipeline

System Workflow

EventOrient system consists of three key components; Scrappers, Analyzers and Visualizers. Scrappers collect high-throughput chronologically ordered streams of social data from micro bloggers (e.g. Twitter) and other social media platforms (e.g. Facebook). Analyzers are fed with batches of data collected by Scrappers. Analyzers leverage Machine Learning algorithms and Network Analysis techniques to thoroughly make insights of the collected data. Machine Learning analyzers are used to analyze document-level data. For example, tweets reporting same topics are clustered together and their sentiments are quantified. Document-level analysis are essential for understanding mass trends and dominating sentiments. Network Analyzers are used to analyze community-level relational data. For example, a network of Twitter's follower-followed is constructed. The network is augmented with data from other sources (e.g. Facebook groups). Centrality metrics are

computed for the network before it is saved into a JSON format for Visualizers to depict them into interactive graphs.

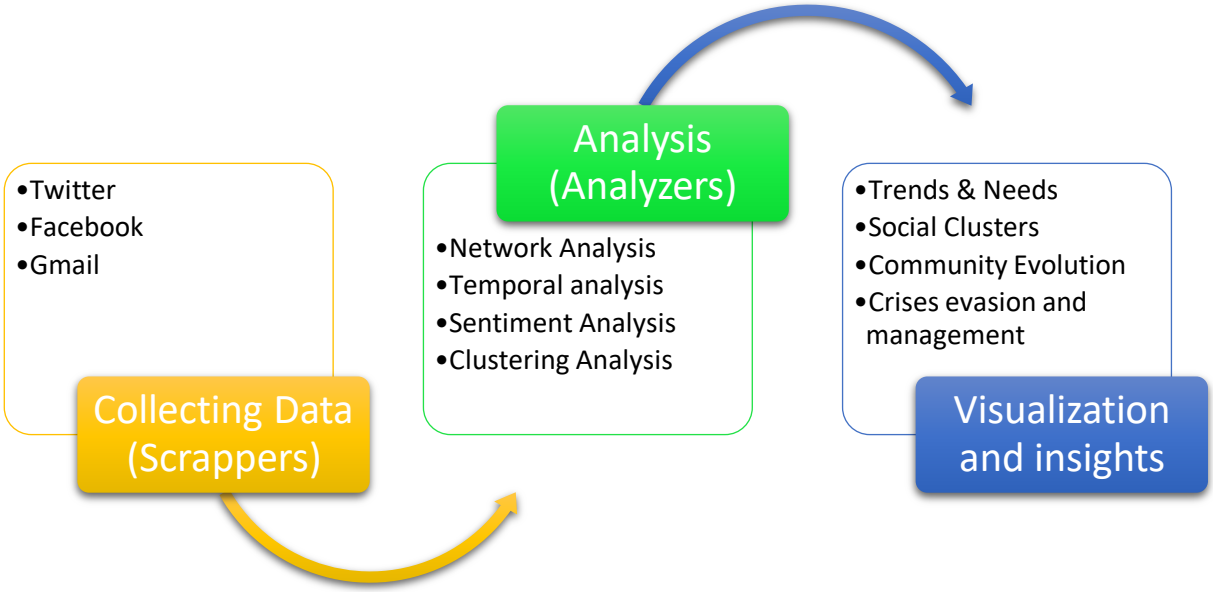

Figure 12 System Workflow

Scrappers

For Twitter, Şehir official Twitter accounts are used as root nodes for our crawler to get the accounts of their followers and the followers of their followers. The collected Twitter accounts are then classified as Şehir account if they match a Şehir Gmail contact with at least 85% similarity. Twitter Streaming and Spark Streaming APIs are used to continuously collect tweets posted by accounts classified as Şehir accounts.

For Facebook, Şehir's community groups are used as pools of Facebook accounts. Posts and shares are not allowed for API-based crawling.

Augmenting Twitter users with their groups membership in Facebook is done by merging Twitter users Data frame with Facebook users Data frame on their *sehir_matches* (the Şehir Gmail contact name of that account).

	full_name	sehir_matches	membership
fb_ID			
1719024661487920	aykut sahin	aykut sahin	Sehir Dersler&Hocalar
2054872011195704	ahmed el bhiri	ahmed bhiri	Sehir Dersler&Hocalar
10210680285591082	sevva deniz bulut	sevva deniz bulut	Sehir Dersler&Hocalar
10155572985146137	seval kavsut guleryuz	seval guleryuz	Sehir Dersler&Hocalar
10215188262249069	malik ekicim	malik ismail ekicim	Sehir Dersler&Hocalar

Table 1Filtering Facebook accounts

Analyzers

For document-level analysis, documents (e.g. tweets, Facebook posts) are represented with vectors in the way proposed in (Hu, Zhang, Hou, & Li, 2017). Online Clustering algorithm is leveraged to cluster tweets reporting similar issues. Online clustering takes one tweet at time incrementally and map it to a cluster of tweets with a similarity threshold, δ. If the tweet's similarity to the cluster is below the threshold, a new cluster is appended with that tweet's vector as its centroid. Our assumption is that tweets with high similarity, and thus, belonging to the same cluster, pertain to similar topics.

Algorithm 1: Single-Pass Online Clustering.

Input: News documents **D**=$\{d_1, d_2, \ldots d_M\}$, Similarity Threshold δ

Output: Event-centric clusters **E**=$\{E_1, E_2, \ldots, E_K\}$

for *the* $j-th$ *document* d_j **do**
  calculate the vector representation $\mathbf{d}_j$ of d_j
  if $\boldsymbol{E} = \emptyset$ **then**
    create E_1
    let $d_j \in E_1$
    represent E_1 by $\mathbf{d}_j$
  else
    foreach *event-centric cluster* E_k **do**
      calculate the similarity $sim(d_j, E_k)$
      let $maxS = \max_j sim(d_j, E_k)$
      let $maxE = E_k | sim(d_j, E_k) = maxS$
      if $maxS \geq \delta$ **then**
        let $d_j \in maxE$
        recalculate the representation of $maxE$ by the centroid
      else
        create a new event-centric cluster E_{new}
        let $d_j \in E_{new}$
        represent E_{new} by $\mathbf{d}_j$
  j++
return all event-centric clusters $E_1, E_2, \ldots, E_K$

For relational-data analysis, we construct networks of different relational contexts. For example, Facebook accounts belonging to same groups and Twitters' follower-followed network. Having the network constructed, we compute the network's metrics; Eigenvector centrality, Communicating Centrality, Betweenness Centrality, degree and parity. The network is saved as a JSON-formatted file for visualization.

```
(0,
 {'betweenness': 0.0039696955301306396,
  'closeness_centrality': 0.28871193488717206,
  'degree': 14,
  'eigenvector_centrality': 0.014345969031633458,
  'name': '[redacted]',
  'parity': 1})
```

Calculating the Network's Metrics and saving it into JSON format

Visualization

Having JSON-formatted networks, JavaScript's D3 library is used to plot an interactive graph of networks and their metrics.

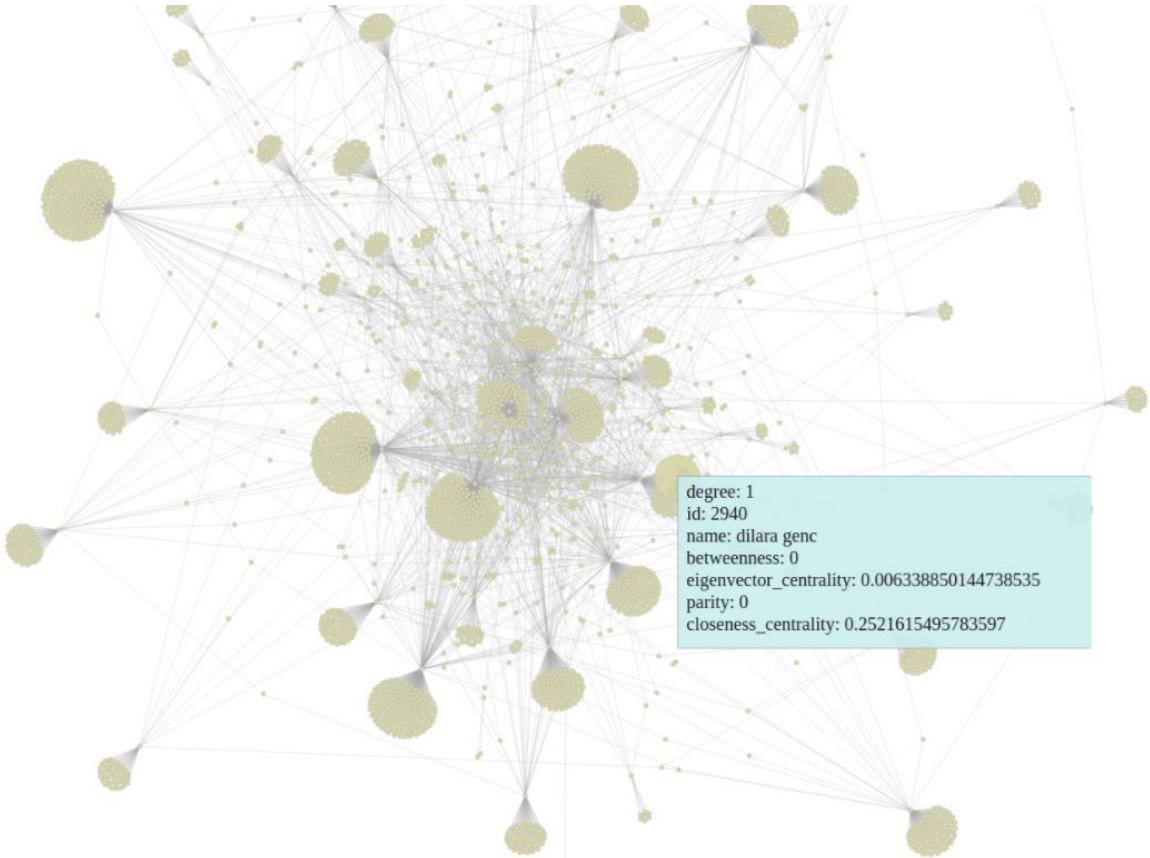

Figure 13 Twitter Followers-Followed Network

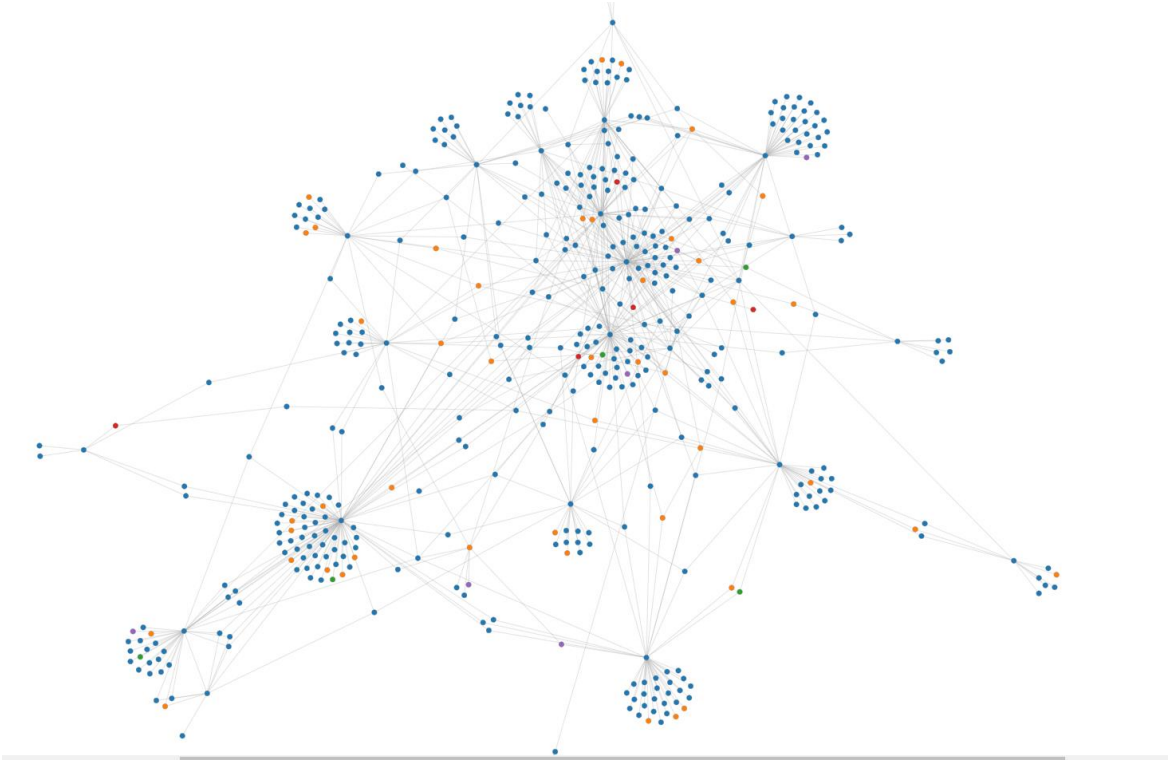

Figure 14 Twitter Networks Augmented with Facebook Group Membership. Colors correspond to Facebook Groups.

Figure 15 Showing Network metrics

Social Sciences aspects

Lest we should become a “hammer” with the well-known analysis computational tools and everything becomes “nails” for us, we sought cooperation from Social Sciences faculty. We believe that, though providing the right answers is important, asking the right questions is far more important. Therefore, it is not a reversed logic to directly look for applications for the analysis tools we have in hand, but rather we should look for the issues shouting to be addressed and then look for the right means of addressing them. With the help of a philosophy-major student, we have conducted meetings with professors from sociology and political science departments who in turn, including the philosophy student as well, have shared valuable insights about how a community-scale analysis should consist of, what information are important to collect, what theories and methods relate to our goal and what challenges are likely to face us.

References

Chen, G. J., L. Wiener, J., Iyer, S., Jaiswal, A., Lei, R., Simha, N., . . . Yilmaz, S. (2016). Realtime Data Processing at Facebook. *International Conference on Management of Data* (pp. 1087-1098). California, USA: Facebook Inc.

Hu, L., Zhang, B., Hou, L., & Li, J. (2017). Adaptive online event detection in news streams. *Knowledge-Based Systems*, 105-112.

Jacobs, S., Sarwar Uddin, M., Carey, M., Hristidis, V., J. Tsotras, V., Venkatasubramanian, N., . . . Li, Y. (2017). A BAD Demonstration: towards Big Active Data. *Proceedings of the VLDB Endowment*, 1941-1944.

Jin, H., Zhu, Y., Jin, Z., & Arora, S. (2014). Sentiment Visualization on Tweet Stream. *JSW*, 2348-2352.

Lekha R. Nair, D. S. (2005). Streaming Twitter Data Analysis Using Spark. *Journal of Theoretical and Applied Information Technology*, 349-353.

Marcus, A., S. Bernstein, M., Badar, O., R. Karger, D., Madden, S., & C. Miller, R. (2011). Processing and Visualizing the Data in Tweets. *ACM SIGMOD*, (pp. 21-27). NY, USA.

Pang, Z., Wu, S., Chen, G., Chen, K., & Shou, L. (2017). FlashView: An Interactive Visual Explorer for Raw Data. *Proceedings of the VLDP Endowment*, 1869-1872.

Wei, H., Sankaranarayanan, J., & Samet, H. (2017). Finding and Tracking Local Twitter Users for News Detection. *International Conference on Advances in Geographic Information Systems* (p. 4). NY, USA: DOI: 10.1145/3139958.3141797.