A Digital Library for Crowds on the Real-Time Social Web

Jeff A. McGee
Texas A&M University
College Station, TX 77843
jeffamcgee@gmail.com

Krishna Y. Kamath
Texas A&M University
College Station, TX 77843
kykamath@cs.tamu.edu

ABSTRACT

In this project, we want to build a digital library for transient crowds in highly-dynamic social messaging systems like Twitter and Facebook. A crowd is a short-lived ad-hoc collection of users, representing a "hot-spot" on the real-time web. For example, an event like super-bowl might result in formation of crowds that are discussing various happenings in the game. Successful detection of these hot-spots can positively impact related research directions in online event detection, content personalization, social information discovery, etc. We will build a framework that allows an analyst or curious user to find interesting crowds and see how they evolve. This framework will have three main parts: a database to store crowds, users, and their messages; a set of crowd detection algorithms and filters; and a tool for searching for crowds by topic, geography, or user-name.

1. INTRODUCTION

In much the same way as web search engines provide instant access to the *retrospective web* of previously crawled and indexed content, there is growing excitement over a new generation of applications for monitoring, analyzing, and distilling information from the *prospective web* of real-time content that reflects the current activity of the web's participants. As a step towards this vision of a prospective web information platform, this paper developing a digital library for "hotspots" on the real-time social web. In general, a "hotspot" could be defined by the posting and sharing actions of users in social systems, for example triggered by an offline event (e.g., Facebook posts and Tweets in response to a live Presidential debate or a chemical fire at a nearby refinery) or by an online phenomenon (e.g., reaction to Internet memes, online discussion). Detecting these hotspots as they arise in real-time and providing a framework for browsing and searching them, is an important and fundamental building block for enabling new real-time web applications, applications related to identification and dissemination of disaster and emergency-related information, among many other emerging social mining applications.

In Section 2, we describe some papers that discuss the problem we are interested in and in Section 3, we briefly describe our method to discover crowds. We finally present our architecture in Section 4.

2. RELATED WORK

Twitter[1] is a microblogging platform which is fast gaining popularity[7] among broad sections of society and has a global outreach spreading from developed, urban nations like the United States where it has a high adoption rate [3], to developing countries in parts of Asia and South America.

Along with working on understanding microblogging usage and communities [3], the main author - Akshay Java was one of the first few who dealt with the measurement of the usage and nature of communities in microblogging. In his latter study [2], he presented his observations of the microblogging phenomena and user intentions by studying the content, topological and geographical properties of such communities. He found that microblogging provides users with a more immediate form of communication to talk about their daily activities and to seek or share information.

Identifying highly dynamic ad-hoc collections of users what we refer to as crowds in massive social messaging systems like Twitter and Facebook is important [4]. Kamath et al. in the study suggest an efficient locality-based clustering approach for identifying crowds of users in near real-time compared to more heavyweight static clustering algorithm. The study consisted of 711,612 users and 61.3 million messages, and tells what approaches to follow to efficiently and effectively identify Twitter-based crowds.

In the other study based on the previous one, Kamath et al [5], they add another salient feature - a novel crowd tracking and evolution approach for linking crowds across time periods. Unlike the more static and long-lived group-based membership offered on many social networks their goal was to support the discovery of organic and highly-temporal group affiliation, which they refer to as "transient crowds". A transient crowd according to them is a potentially short-lived ad-hoc collection of users bound together by some common thread - which can be communication-based, location-based or interest based. We are inspired by this idea of formation and detection of crowds. We try to implement detection of crowd in Crowdy in a similar fashion.

3. CROWD DISCOVERY ALGORITHM

[1]http://www.twitter.com

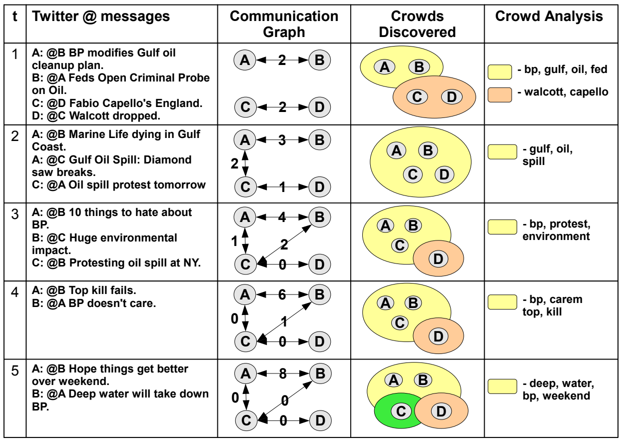

Figure 1: Example of crowd discovery and tracking in Twitter.

To illustrate the problem of crowd discovery, consider the simple example in Figure 1. At time t=1, users A and B send messages to each other, as do users C and D.[2] The associated communication graph shows an edge between the two pairs, where for simplicity the edge is annotated with the number of messages between the users (2, in both cases). Further, suppose we identify crowds based purely on graph connectivity. So for time t=1, we see there are two crowds discovered {A,B} and {C,D}. For each crowd, we can characterize the semantics of their communication with simple keywords extracted from the content of the tweets: ("oil", "gulf") and ("walcott", "capello"). At time t=2, the communication graph is updated with a new edge (connecting User A and User C), and the existing edges are decayed by one (again, a simplifying assumption for the purposes of this example). A single crowd is discovered since all users are connected via edges with non-zero edge weights. At time t=3, User D leaves the main crowd since no messages to or from User D have been observed since time t=1. This process continues until time t=5 when User C also leaves the main crowd due to inactivity. Note that crowds are discovered from communication graph only and not from the content of the messages. As an example of crowd tracking, we can track the evolution of the yellow crowd across time periods, observing the changes it goes through as it grows in size from t=1 to t=2 and then reduces to two users by t=5.

The grouper deals with the transient crowd discovery and tracking in systems like Facebook and Twitter. For practical crowd discovery and tracking in a large time-evolving communication network, however, we face four key challenges:

- First, systems like Facebook and Twitter are extremely large (on the order of 100s of millions of unique users), placing huge demands on the computational cost of traditional community detection approaches (which can be $O(n^3)$ in the number of users [1]).

- Second, these services support a high-rate of edge addition (new messages) so the discovered crowds may become stale quickly, resulting in the need to re-identify all crowds at regular intervals (again, incurring the high cost of community detection). The bursty nature of user communication demands a crowd discovery approach that can capture these highly-temporal based clusters.

- Third, the strength of association between two users may depend on many factors (e.g., recency of communication), meaning that a crowd discovery approach based on graph clustering should carefully consider edge weights. With no decay at all (meaning that edges are only inserted into the network but never removed), all users will tend towards a single trivial large crowd. Conversely, overly aggressive edge decay may inhibit any crowd formation at all (since edges between users may be removed nearly as soon as they are added).

- Fourth, crowds may evolve at different rates, with some evolving over several minutes, while others taking several days. Since crowds are inherently ad-hoc (without unique community identifiers – e.g., Fans of LA Lakers), the formation, growth and dispersal of crowds must be carefully managed for meaningful crowd analysis.

With these challenges in mind, we propose to discover and track transient crowds through a communication based clustering approach over time-evolving graphs that captures the natural conversational nature of social messaging systems. Two of the salient features of the proposed approach are (i) an efficient locality-based clustering approach for identifying crowds of users in near real-time compared to more heavyweight static clustering algorithms; and (ii) a novel crowd tracking and evolution approach for linking crowds across time periods.

To support transient crowd discovery in Twitter-like services with 100s of millions of participants, we propose to leverage the inherent locality in social messaging systems. Concretely, we identify two types of locality that are evident in Twitter-like messaging systems: (i) temporal locality and (ii) spatial locality.

Temporal Locality: Transient crowds are intuitively short-lived, since they correspond to actively communicating groups of users. Hence, the composition of a crowd at a point-in-time should be impacted by recent messages as opposed to older messages. As more users interact with the crowd, the crowd should grow reflecting this *temporal locality* and then shrink as users in the crowd become inactive (that is, their last communication with the crowd becomes more distant in time).

Spatial Locality: Intuitively, transient crowds are made up of a very small percentage of users compared to the entire population of the social network. Hence, new messages (corresponding to the addition of edges to the communication network) should have only a local influence on the crowds that exist at any given time. That is, changes in a small region of a graph should not affect the entire graph. In a dataset of 61 million Twitter messages described in [6], we have confirmed the existence of this *spatial locality* by finding that only about 1% of users are within two hops, meaning that an edge insertion has only a local effect.

[2]For simplicity, the example discretizes time so that all messages between users occur in steps. In practice, the proposed algorithm relaxes this assumption and can handle arbitrary message sending times.

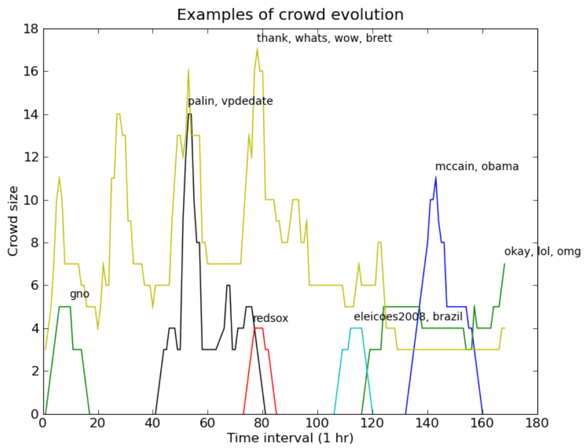

Figure 2: Examples of the crowds discovered in the dataset.

Hence, we can take advantage of both, local changes to the overall communication network (spatial locality) and recent changes to the network (temporal locality), for supporting efficient transient crowd discovery. The complete algorithm is described in [6]

3.1 Sample Crowds Discovered

We illustrate some of the discovered crowds and their lifespans in Figure 2, with an annotation next to the crowd peak showing the topic of discussion. We see a crowd (shown in black) discussing Sarah Palin and the Vice-Presidential debate from the 40^{th} hour to 80^{th} hour that peaks around the time of the actual debate. We observe that crowds that talk about general everyday things have a greater lifespan than crowds discussing specific events. For example in Figure 2, a crowd (annotated with *thank, whats, wow*) discussing everyday things lives through the entire week, while, during the same period we observe several event-specific crowds, like crowds discussing the Red Sox, Sarah Palin, and Girl's Night Out (gno) forming and dispersing. These event-specific crowds start forming just before the event and die a few intervals after the completion of that event. This distinction between the crowds discovered clearly indicates two types of Twitter usage: first, it is used as a platform to discuss and debate specific events, and second, it as also used a means of everyday communication.

4. ARCHITECTURE

An architecture for our solution is shown in Figure 1. Data flows from a crawler, through zero or more filters, to a grouper, and then through zero or more filters, and then it is finally stored in a MongoDB database. We went with an extremely modular architecture to make the system flexible. We can easily integrate new crawlers, filters, and groupers into the system.The various modules of this architecture are described in the remainder of this section.

4.1 Crawler

Crawlers interact with social media websites to collect information. All of the crawlers we use in this project acquire tweets and users from Twitter using their APIs.

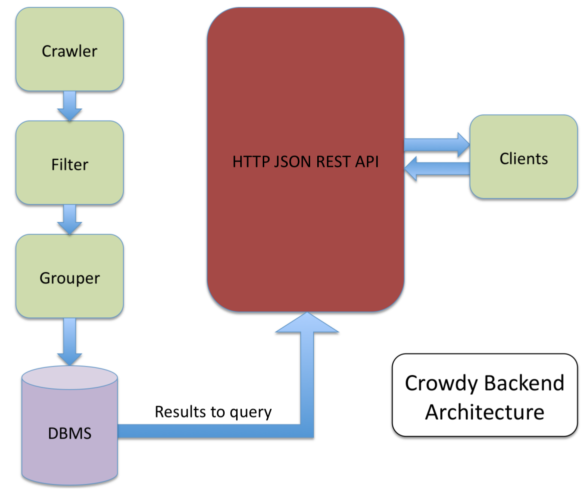

Figure 3: Examples for trending phrases discovery problem.

4.1.1 Localcrawl

This crawler collects data from users in a region using the standard Twitter API. We started with a tiny sample set of seed users. It uses snowball sampling to collect more users that are likely to live in an area. Once we have a list of users

We have used this crawler and the crowd detection algorithms on both Bryan/College Station and the Houston area, but the demo is currently exclusively running on the Bryan/College Station data.

4.1.2 GeoStream

This crawler listens to Twitter's streaming API to find geo-located tweets. It receives every geo-located tweet posted from anywhere in the world. In the future, this data may be used to show crowds on a map.

4.2 Grouper

Groupers detect crowds in the data collected by a crawler. The precise definition of a crowd depends on the implementation of the grouper. A grouper could create crowds based on communication patterns, geographical proximity, or textual similarity. Once crowds are discovered, this module formats crowds to a well defined crowds object and adds it to a DBMS.

4.2.1 Mentions

For this project, we exclusively discover crowds based on communication. The mentions grouper looks for sets of people who are communicating with each other. The algorithm to discover crowds is described in Section 3.

4.3 Filter

Filters are used to remove unnecessary data, such as spam tweets, and add automatically generated metadata to tweets, users, and crowds. A filter can appear before or after the grouper module as needed. In the following sections, we discuss two filters that we implemented and several that we plan to add in the future.

4.3.1 Lucene Filter

The Lucene module creates a Lucene index from the text of the tweets in a crowd. This index is used to speed up searching through the text of the tweets. This filter is designed so that a crowd can be updated as new data arrives from a crowd.

4.3.2 Network Filter

We also have a network filter that builds a graph that represents the communication between the users in a crowd. At the moment, it analyzes two things: degree centrality and clustering coefficient. The degree centrality is used to find the most important users in a given crowd. The clustering coefficient of the crowd is added to the crowd as automatically generated metadata

4.3.3 Planned Filters

There are several filters that we have not implemented, but may implement in the future. The ability to easily add new filters and create an automated workflow is an important part of our design.

- Remove users that appear to be spammers.
- Add metadata to each user that contains an estimate of the users location, and some information about the quality of the estimate.
- Find the key terms in a crowd and add that metadata to the crowd.
- Summarize the tweets from a crowd.

4.4 HTTP JSON RESTful APIs

: This module provides client an interface to the digital library. The module exposes JSON APIs that the clients can use to browse and search the crowds archive. Here is a brief description of the more important methods:

search/crowd Find crowds sorted by relevance or time.

crowd/id Retrieve all of the data for a crowd.

crowd/users Retrieve a list of all the users in the crowd.

crowd/tweets Retrieve a list of all the tweets in the crowd.

crowd/star Mark a crowd as significant.

user/edges Retrieve the friend and follower relationships for a given user.

4.5 Tools

The tool we used to develop this framework are as follows:

- **PyLucene**: To provide the search functionality we use Apache Lucene. It provides methods for text indexing and searching. To integrate Lucene with our framework we use Pylucene which is a Python wrapper for Lucene. This is used in the filter module of our architecture.
- **CherryPy**: This is an object-oriented web framework for Python. The JSON REST API that interface the digital library with clients is hosted using this web framework.
- **NetworkX**: This is a library for manipulating graphs in Python. We use this to determine quality of crowds generated by crowd detection framework and filter out crowds of poor quality. This sits in the filter module.
- **Beanstalk**: Beanstalk is a simple, fast work queue. We use it to connect the crawlers, filters, and groupers together.
- **MongoDB**: MongoDB is a scalable, high-performance, open source, document-oriented database. We store the crowds discovered by the framework in this DB. The API module interacts with this database to retrieve collections as required by the user query.

5. CONCLUSION

■< Updated upstream In this project, we presented a framework of a digital library for transient crowds in highly-dynamic social messaging systems like Twitter and Facebook. We built a framework that allows an analyst or curious user to find interesting crowds and see how they evolve. The framework enabled clients to interact with the library using JSON APIs. The APIs enabled searching and browsing of the digital library.

======= In this project, we presented a digital library for transient crowds in highly-dynamic social messaging systems like Twitter and Facebook. We described a framework that allows an analyst or curious user to find interesting crowds and see how they evolve.

■> Stashed changes

6. REFERENCES

[1] G. W. Flake, R. E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

[2] A. Java and T. Finin. Mining social media communities and content. In *Doctoral Dissertation in University of Maryland at Baltimore County Catonsville*. ACM, 2008.

[3] A. Java, X. Song, T. Finin, and B. Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, WebKDD/SNA-KDD '07. ACM, 2007.

[4] K. Kamath and J. Caverlee. Identifying hotspots on the real-time web. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10. ACM, 2010.

[5] K. Kamath and J. Caverlee. Transient crowd discovery on the real-time social web. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11. ACM, 2010.

[6] K. Y. Kamath and J. Caverlee. Transient crowd discovery on the real-time social web. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 585–594, New York, NY, USA, 2011. ACM.

[7] T. O'Reilly and S. Milstein. *The Twitter book*. O'Reilly, 2009.