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

2. RELATED WORK

3. ARCHITECTURE

An architecture for the proposed solution is shown in Figure 1. The various modules of this architecture are as follows:

- **Crawler**: This module deals with interacting with social media websites to collect information. For this project we will be collecting data from Twitter using Twitter Streaming APIs.
- **Filter**: This module removes unwanted data like spam messages, from data collected by the crawler.
- **Grouper**: The grouper module analyses the data collected, to determine crowds and adds them to the DBMS.

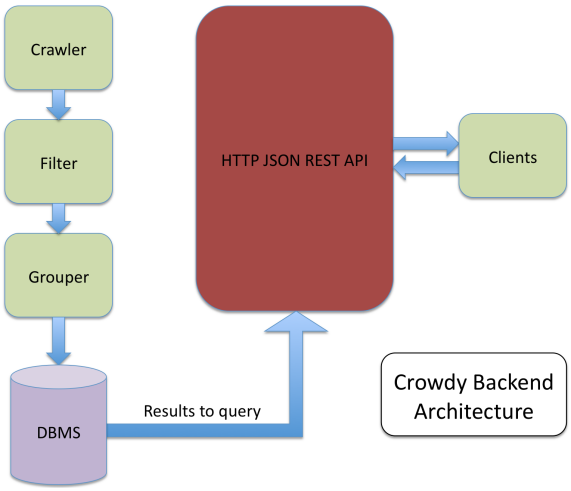

Figure 1: Examples for trending phrases discovery problem.

- **HTTP JSON RESTful APIs**: This module will expose a HTTP-based API that clients can query to get data from the collection. The API will return crowds, users, and their tweets.
- **Beanstalk**: Beanstalk is a simple, fast work queue. We use it to connect the crawlers, filters, and groupers together.
- **Tornado**: Tornado is an open source, non-blocking web server. We choose it because it is fast and scalable.
- **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.

4. CROWD DISCOVERY ALGORITHM

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 [**?**]).

- 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 [**?**], 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.

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 decsribed in [**?**]

5. CONCLUSION