Reconstruction and Analysis of Twitter Conversation Graphs

Peter Cogan∗
peter.cogan@alcatel-lucent.com

Matthew Andrews†
andrews@research.bell-labs.com

Milan Bradonjić†
milan.bradonjic@alcatel-lucent.com

Gabriel Tucci†
gabriel.tucci@alcatel-lucent.com

W. Sean Kennedy†
sean.kennedy@alcatel-lucent.com

Alessandra Sala∗
alessandra.sala@alcatel-lucent.com

ABSTRACT
Communication over social networks has been an emergent theme over the last several years. We wish to examine the underlying graph structure in social networks and to understand the dynamics of how information propagates in these networks. Gathering data from large social networks such as Facebook and Twitter has become increasingly more difficult due to external pressure to improve privacy and internal pressure to generate revenue. We examine the case of Twitter, the micro-blogging social network. Twitter provides APIs which allow users to sample tweets stating specific keywords - however the reconstruction of complete conversations (which is integral to the dynamics of how information propagates) is difficult. We focus on the particular case of a conversation formed by users replying to tweets. We present a robust approach for the reconstruction of complete conversations and compare the resultant graph measurements to those created using keyword searches.

Categories and Subject Descriptors
E.1 [Data Structures]: Graphs and Networks
H.3.5 [Information Storage and Retrieval]: Online Information Services - Data sharing
H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval Information filtering

Keywords
Social Network, Graph

1. INTRODUCTION
Communication over social networks has become an emergent research theme over the last several years followed by a large body of literature and several major conferences. We wish to examine the underlying graph structure in social networks and to understand how information propagates in these networks. In particular we would like to understand the conversation graphs that are formed when users in the network discuss a particular topic.

Performing such a study requires data collection from social networks which presents a number of challenges. Gathering data from large social networks such as Facebook [1] and Twitter [2] has become harder than it used to be several years ago due to external pressure to improve privacy and internal pressure to generate revenue. For example: (i) Facebook has significantly improved user privacy awareness and control, which has led to most Facebook users keeping their data private; (ii) Twitter employs third parties to help monetize the release of data which can be a hurdle to research. However, both Twitter and Facebook still provide APIs to support third party applications. The main difference is that as Twitter accounts are typically open these APIs enable examination of most accounts, whereas Facebook only allows access to accounts of users that have authenticated access, typically by ‘friending’.

Our current focus is on the social networking site Twitter due to its popularity, general open nature, and availability of limited APIs for extracting data. Twitter launched in 2006, has several hundred million users and is primarily categorized as a micro-blogging social network whose posts (tweets) can contain a maximum of 140 characters. Users write tweets on different topics that include their current activities and share links to interesting media. Once a user has posted a tweet, all her followers will receive the tweet. Tweets posted by a public profile user can be viewed online by anyone. Additionally, tweets can contain geographic information if posted from GPS-enabled devices (≈ 1% of tweets contain valid latitude and longitude coordinates). Whenever a user receives a tweet, she may retweet the tweet (send it on to their own set of followers), or alternatively may reply to the tweet (respond to it by using her own original tweet). In this work we present two data collection methods for Twitter in order to study the information propagation within conversations among its users.
Twitter makes data available via various streaming feeds that are freely available to anyone with a Twitter account. This includes the filter feed which enables the user to specify keywords they are interested in tracking. Other methods include the REST API and the Search API which we describe in Section 2.

Our particular focus is on conversation graphs that represent how users interact within the network. There are a number of ways in which a conversation graph can be defined. We will consider the following types of structures:

1. **Mention graphs**: The first type is a graph where vertices are users and (directed) edges represent any interaction between the users. Such interactions include replies to each other’s tweets, retweets, mentions (a mention is when a user refers to another user directly by username), etc.

2. **Reply trees**: The second type is a graph where vertices are tweets and a (directed) edge represents one tweet that is a reply to another. In this case a tweet can only reply to one other tweet, thus the connected components of such graphs are trees.

3. **User graphs**: The third type can be viewed as a projection of the second type. In this case vertices are users and a (directed) edge exists if one user replies to another.

The main difference between the first type and the second and third types is that the first type includes retweets and mentions in the definition of an edge whereas the second and third types only include replies in the definition. One reason to examine a graph with replies only is that intuitively a reply represents the most interactive communication between two users. Hence by studying graphs formed by replies we can gain insight into the most interactive relationships within the network.

**Our Contributions**

In this paper we focus on two main questions. First, how do we reconstruct conversation graphs without violating limits imposed by the Twitter APIs. Second, how can we characterize the structure of conversation graphs that we detect.

In particular we consider two methods to build conversation graphs. The first method utilizes keyword filtering. This allows the user to retrieve tweets matching a particular keyword, i.e., a user could retrieve all tweets which mention the keyword “Obama”. Results from the two methods are described in Section 2.1. One finding is that two types of conversation graph, namely paths and stars, are extremely common. Paths typically correspond to two users having a back-and-forth conversation. Stars correspond to multiple users replying to a single popular tweet. However, there are a number of conversations that fall between these two extremes and we investigate the structures that are formed.

**Previous Work**

Conversation graphs form a dynamic representation of how information is propagated throughout the social network over time. To the best of our knowledge, there has not been previous work on the structure of reply-based conversation graphs or on methods to collect these graphs without violating Twitter API limits.

However, the general topic of information propagation in social networks has been addressed previously. For example, in [8] the authors proposed a simple mathematical model that generates basic conversation structures and takes into account the identities of each member of the conversation in Usenet, Yahoo! Groups, and Twitter. Moreover, some recent studies looked at how information flow depends on the structural properties of user interactions [3, 9]. Concretely, social cascades (the information spread in social networks) in Flickr are the focus of [9]. The pictures in Flickr represent the information unit around which the interactions happen. Becoming a fan of a particular picture is interpreted as information flowing from one user to another. The 98% of the social cascades generated from pictures with less than five fans happen within the one hop neighborhood of the picture uploaders. For popular pictures, instead, more than 50% of their fans are outside the uploader’s one hop neighborhood.

This analysis suggested that the main feature leading to a broader information spread is the popularity of a picture. On the other hand, the study of [9] provides a preliminary analysis of the topological structure of retweets. The forest of retweet trees has a large number of one or two-hop chains. The majority of retweet trees have a height smaller than six, and no trees beyond eleven hops were found.

Moreover, modeling the interaction among users plays a fundamental role in understanding how the information is disseminated in the network [13, 9] and how to maximize its spread [4, 5]. In [9], Kwak et al. proposed the first large scale analysis of the topological and temporal structures of retweets to capture popularity of users, trending topics and temporal patterns to describe how the information propagates in the network. The temporal dimension of the information spread plays a significant role both to characterize users’ retweets and to identify influentials [10, 14]. Inspired by sociological and viral marketing studies, Cha et al in [9] propose an in-depth analysis to characterize user influence based on the investigation of several factors that may play an important role in identifying influentials in Twitter. Finally, social cascades have also been used to demonstrate that, differently from static properties, in dynamic interactions the users geographic locality is a key factor in characterizing how the information spreads in the network [12].

Complementary to the aforementioned, our work is mainly focused on constructing reply-based conversation graphs sub-
object to the API limits and understanding the distribution of their structural properties ranging from one common extreme form (paths) to another (stars). To the best of our knowledge, these topics have not been the focus of prior work.

2. RECONSTRUCTING CONVERSATIONS

In this section we describe two methods for constructing twitter conversation graphs. These are the filtered conversation method which produces mention graphs and the complete conversation method which produces reply trees and user graphs. Both of these methods employ the Twitter filter API where in addition the latter employs the Twitter search API. The goal of the former is to understand the structure of the conversation graphs surrounding specific topics, in particular, involving specific keywords. In contrast, the latter aims to reconstruct entire conversations without the restriction that each tweet of a conversation must contain specific keywords. In Section 2.1, we discuss our analysis of the graphs built by these two methods.

Filtered Conversations. Given a set of keywords to track, we use the Twitter filter API to collect all tweets containing any one of these keywords over a fixed period of time. We then create a mention graph from this set of tweets. We note that given a set of tweets it is trivial to construct any of the three types of conversation graphs in linear time.

Complete Conversations. The goal of the second algorithm is to be able to create reply trees and user graphs. The main reason this is difficult is that, although a tweet payload contains a large amount of meta-data, it does not include a list of tweets that reply to it. The inclusion of such a list would be impractical since it can be very long, and is of course dynamic.

Before describing the algorithm, the Twitter API is briefly described. The REST API allows access to specific tweets, follower lists, retweet lists, etc. Each tweet is assigned a unique unsigned 64 bit integer identifier - this is the ID. Usage of many of the APIs are limited to 350 calls per hour for authenticated applications, thus any conversation collection algorithm which relies on the REST API will be severely limited. In order to mitigate against this issue, this algorithm takes advantage of the search API which returns tweets matching a specified keyword. Although this API is also rate-limited, it offers the opportunity to recreate more conversations than would otherwise be possible.

The reconstruction algorithm consists of a series of algorithms which are run in parallel. First, the streaming API is used to filter tweets with specific keywords which are the subject of interest. The set of tweets retrieved from the filter operation is denoted D. Each tweet D_i in D is part of a conversation C - however it may not be the root of conversation C. A tweet can have three types, root, reply or retweet, however for the purpose of reconstructing reply trees, retweets are ignored. Let the type of tweet t be denoted type(t). The goal of the reconstruction algorithm is, given D_i, to determine the root of the conversation C, and then to determine the set of all tweets which are part of C.

Algorithm 1 Iterative Root Finder

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S_n is the tweet passed as an input</td>
</tr>
<tr>
<td>2</td>
<td>k=0</td>
</tr>
<tr>
<td>3</td>
<td>repeat</td>
</tr>
<tr>
<td>4</td>
<td>if type(S_n−k)==Reply then</td>
</tr>
<tr>
<td>5</td>
<td>if !Call Search API to search for tweets addressed to user then</td>
</tr>
<tr>
<td>6</td>
<td>RETURN false</td>
</tr>
<tr>
<td>7</td>
<td>end if</td>
</tr>
<tr>
<td>8</td>
<td>if ! Extract S_n−k−1 by matching field in_reply_to_status_id then</td>
</tr>
<tr>
<td>9</td>
<td>RETURN false</td>
</tr>
<tr>
<td>10</td>
<td>end if</td>
</tr>
<tr>
<td>11</td>
<td>end if</td>
</tr>
<tr>
<td>12</td>
<td>k=k+1</td>
</tr>
<tr>
<td>13</td>
<td>until (type(S_n−k−1)==Tweet)</td>
</tr>
<tr>
<td>14</td>
<td>RETURN true</td>
</tr>
</tbody>
</table>

Algorithm 2 Iterative Search

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i=1</td>
</tr>
<tr>
<td>2</td>
<td>repeat</td>
</tr>
<tr>
<td>3</td>
<td>for j = 1 to j ≤</td>
</tr>
<tr>
<td>4</td>
<td>Search for tweets addressed to author of T_ij</td>
</tr>
<tr>
<td>5</td>
<td>Extract replies to T_ij by matching field in_reply_to_status_id - add tweets to P. Add new members to Q</td>
</tr>
<tr>
<td>6</td>
<td>end for</td>
</tr>
<tr>
<td>7</td>
<td>Assign T_{i+1} = (T_i, P)</td>
</tr>
<tr>
<td>8</td>
<td>Assign M_{i+1} = (M_i, Q)</td>
</tr>
<tr>
<td>9</td>
<td>i=i+1</td>
</tr>
<tr>
<td>10</td>
<td>until (</td>
</tr>
</tbody>
</table>

If D_i is not a root tweet, the Iterative Root Finder algorithm is used to determine the root of C.

Let S be a subset of conversation C where S_0 is the root of the conversation C and S_n is a single tweet of the conversation retrieved using the filter (ie D_i in the above context). S is defined such that S_{n−k} is a reply to S_{n−k−1} for k=0,...,|S|-2, with this pattern repeating to S_0. The goal of the Iterative Root Finder algorithm is to identify S_0 given S_n. Note that when the algorithm starts, |S| is not known.

Once S, in particular the conversation root S_0 has been established, the remainder of C can be sought using the Iterative Search algorithm. Let T_i be the vector of all tweets in C at iteration i and M_i be the vector of all contributors to C at iteration i. The j^{th} tweet at iteration i is denoted T_{ij}.

Since new tweets and new conversation members can be continuously added, the Iterative Search algorithm is run repeatedly until some condition, indicating that the conversation has ended, is met.

One of the key components of this algorithm is the ability to find replies to tweets. The Twitter REST API does not provide a mechanism to directly find replies, thus we use the search API to find replies to users. This is possible because a reply to a user will always begin with @username, thus a search call will return all the replies to that user. In order to determine whether the returned tweet is an element of the conversation, we can check the in_reply_to_status_id field. This field was only added to the search output in December 2011. Previous efforts by the authors to create
reply trees were far more complex and involved use of the track and follow filter feeds, but gave significantly less reliable results. The search API returns up to a maximum of 1500 tweets from the previous week. Thus we cannot track old conversations, and we cannot track conversations whose members receive huge numbers of replies. In order to limit the number of search API calls we also emply the since_id field which ensures we do not search for tweets before the root. The algorithm was tested by injecting multi-user test conversations with known structure into Twitter.

Our analysis has shown that the vast majority of conversations are not continued if the oldest tweet in the conversation is more than 6 hours old. Thus in our algorithm if the oldest tweet in a conversation is more than 6 hours old we stop tracking it. There is no way to know for certain that a conversation will not be replied to at some indeterminate time in the future, however the algorithm must eventually stop following conversations since there are practical limits to how many search calls can be made while still aggregating active conversations.

There are some circumstances under which we are unable to correctly recreate a reply tree as it originally occurred. For example, the user has the option of deleting a tweet, or a user with a public profile could write a tweet in reply to a user with a private profile. Less than 5% of conversations are affected this way.

Twitter applies limits to the usage of the APIs which ensure Twitter servers do not get overloaded with DOS attacks, runaway scripts, etc. The algorithm handles these limits by inserting delays should the search API limit be reached. The frequency with which the search API can be called is not published. If the search rate limit is hit, the search API can not be used for a period of time. This period of time is provided to the user via the HTTP header and is typically close to 8 minutes. The authors are aware of some academic researchers who have been blacklisted by Twitter for not respecting these limits.

The algorithm was run on a single multi-core machine running OSX 10.7.2. Processor speed is largely irrelevant due to the large latency invoked by calling the search API, plus the rate limiting delays which must be inserted. The repeated calls to the search API result in the storage of a large number of tweets. For the few days of data collection, approximately 2 million tweets were collected which must be stored in memory as they could be added to conversations during any iteration. For this work, the data structures stored all the tweet meta information in memory, however most of this is redundant and could be backed up to disk while a leaner data structure is kept in memory.

### 2.1 Results

We now describe the results of the filtered conversation and complete conversation methods.

**Filtered Conversations Algorithm.** We used the filter conversation method with numerous keywords including “GOP” and “flu” between January 11 and February 11, 2012 to create both reply trees and mention graphs. These two topics alone resulted in the collection of over two million tweets. Figure 4 shows the 9 largest connected components of the reply tree constructed from the collection of tweets containing the keyword “GOP”. Figure 5 plots the diameter as a function of the number of vertices for each connected component. This figure immediately reveals a large number of components $C$ whose diameter is nearly $|V(C)| – 1$; clearly, a graph $H$ has diameter $|V(H)| – 1$ precisely when it is a path. On the other extreme are stars, that is, components whose diameter exactly 2 (see Figure 4).

In contrast, the structure of the mention graphs constructed from the same set of tweets is very different. For example, the largest connected component for the mention graph constructed from tweets containing the keyword “GOP” only on January 21 has 18132 nodes and 32815 edges; clearly, this component is not a tree. Furthermore, the next largest connected component has cardinality 14. Less popular topic create smaller scale examples which are easier to visualize. For instance, Figure 6 shows the 4 largest connected components of a mention graph constructed from tweets containing the keyword “flu” only on January 21. We note that unlike the giant component of the “GOP” mention graph, the “flu” mention graph has no single component whose cardinality dominates the remaining components. Interestingly, there are still a large number of components which are star-like, though, the largest two components have a much richer ge-
Complete Conversations Algorithm. Using the complete conversation algorithm, we collected a total of 3114 conversations containing 33k tweets from February 1st 2012 to February 5th 2012 and constructed the corresponding reply trees and user graphs. The cardinality and diameter of these graphs are shown in Figures 1 and 3 while Figure 8 shows a scatter plot of the cardinality and diameter. Similarly to Figure 5, Figure 8 immediately reveals a large number of conversations whose diameter is exactly one less than its cardinality, i.e., paths. This is caused by a back-and-forth conversation between two users. The set of conversation graphs is dominated by this structure, indeed, more than 60% of the reply trees are paths. A large cardinality example of a reply tree is shown together with its user graph in Figure 7(a). Since there are only two users, the user graph is trivial. On the other extreme, there are also conversation graphs whose diameter is 2, some having very large cardinality. These graphs correspond to instances whereby a single user generates a tweet to which a large number of people reply – however the users do not respond to each other’s replies. When plotted, such a graph looks like a star as in Figure 7(b). Since each tweet is from a unique user, the corresponding user graph would look identical.

The path and star graphs can be considered the trivial extremes of conversation graph behaviour. Graphs in between these two extremes reveal richer geometric detail. Figure 7(c) shows an example reply tree and user graph for such a case. It is especially interesting to note the extra complexity in the reply tree (Figure 7(d)). This highlights the potential for the geometric exploration of dynamic reply tree compared to older standard techniques which have looked at static user graphs.

The graph density is shown in Figure 2. The plot is dominated by path graphs wherein $|E| = |V| - 1$ where $E$ is the set of edges and $V$ is the set of vertices. Thus the peaks are at $2/|V|$ with the height of the peak determined by the number of such path graphs.

3. CONCLUSIONS AND FUTURE WORK

One natural feature of any conversation graph is the fact that it evolves over time. In future work we would like to extend our studies of the structural and spatial aspects of
the conversation graphs and address their temporal structure. One basic question here is what is the distribution of conversation length. As already mentioned for the vast majority of the reply trees that we examined there was no activity after 6 hours.

Another interesting topic to study is how fast a conversation spreads in both a temporal and spatial sense. As before, a conversation starts when one or several individuals tweet about a specific topic. For a temporal analysis let us assume that we count the number of retweets or replies in the conversation as a function of time. More specifically, let \( F(t) \) be the number of retweets or replies up to time \( t \) and let \( f(t) \) be the number of tweets between time \( t - 1 \) and \( t \) where the granularity could be 10 minutes, 30 minutes or an hour. What is the typical behavior of the previous functions? At what time does the maximum of \( f \) occur? In other words, after how long does the conversation become most active? How this depends on the topic of conversation? How often does it occur that the function \( f \) has several local maxima?

Other questions concern the relationship between conversation graph dynamics and geographic location. How are these dynamics affected by the geographic dispersion of the users? Are conversations likely to grow more rapidly if the users are themselves physically close together or does physical proximity make no difference?

Lastly we would like to use our studies of conversation graph structure as a predictive tool. In [11], Liben-Nowell and Kleinberg studied the general problem of link prediction in social networks. We are interested in the most effective way to do link prediction in conversation graphs. In particular, can we use our classification of conversation graph types and past history of users to predict whether or not a user will join a conversation? Moreover, can we predict how they will join the conversation? For example, are they more likely to join by replying to the root node or by replying to one of the more recent tweets that is a current leaf of the conversation?

4. REFERENCES