Finding Relevant Sources in Twitter Based on Content and Social Structure

Kevin R. Canini
Computer Science Division
University of California
Berkeley, CA 94720
kevin@cs.berkeley.edu

Bongwon Suh and Peter Pirolli
Palo Alto Research Center
3333 Coyote Hill Rd.
Palo Alto, CA 94304
suh,pirolli@parc.com

Abstract

As an increasingly large amount of knowledge is shared between users in Twitter, it is becoming a popular source of relevant information to many people. In Twitter, information is transferred primarily via a social relationship called following. Identifying users to follow who are highly relevant to a particular topic of interest can be a difficult task. To address this problem, we introduce a novel method of automatically identifying and ranking Twitter users according to their relevancy to a given topic. The algorithm combines the standard Twitter text search mechanism with information about the social relationships in the network, effectively leveraging the opinions of the crowd. We performed a study comparing the ranked lists generated by the algorithm with lists provided by a commercial website specifically designed for the same purpose. Our initial findings show a good potential for automatically identifying interesting and relevant users.

1 Introduction

Twitter is a social networking and microblogging service that allows users to send and receive short messages called tweets, enabling people to share and discover topics of interest in real-time. As the service grows in popularity, Twitter has become a tool for public discourse in many aspects of society, including pop culture, business, and politics [1, 2]. People are actively exploring its potential to serve many different purposes beyond personal conversations. One important and popular use of Twitter is as a source of relevant information related to a user’s topics of interest. In order to receive information on Twitter, a user must subscribe to (follow) another user’s tweets. Therefore, in order to receive high-quality information about a particular topic, it is important to be able to identify credible users whose tweets are relevant to a topic of interest. Identifying such users can be difficult because there are far too many to manually browse through (Twitter currently has over 100 million registered users), and the service does not provide any relevant tools beyond a simple text search mechanism which returns only a reverse-chronological list of the most recent tweets containing a search term.

Ideally, given a topic of interest, one would hope to find users who provide credible information about the topic. Credibility is often conceived as a function of expertise and trust [3], and expertise is commonly defined by the support and nomination of other professionals in the domain [4]. Therefore, to find credible sources of information, one should seek out people who not only frequently publish topically relevant tweets but also are trusted by their peers. Unfortunately, there is no simple way for a Twitter user to observe how trusted someone is in a particular field. Thus, one of the most important aspects of credibility is also of the hardest for a non-expert to gauge. Our approach measures this quality using the existing social structure in Twitter to provide users with a list of candidate experts for any given topic query.
2 Related work

Since its creation in 2006, Twitter has gained worldwide popularity, and various aspects of the network have been closely studied, including its social structure, usage patterns, and information diffusion properties [2, 5]. Due to the overwhelming availability of information on Twitter, systems designed to recommend and filter out relevant users and tweets have gained attention from researchers, commercial organizations, and regular users alike. Chen et al. [6] examined the personal recommendation of URL items in the Twitter stream. Bernstein et al. [7] proposed tools to organize tweets into topical groups to support fast browsing. Weng et al. [8] applied webpage ranking techniques to analyze the Twitter network, introducing an algorithm for identifying influential users in Twitter by extending the PageRank algorithm. Their algorithm requires access to the full social graph and the entire history of tweets, while our method proposed in this paper could be feasibly implemented without heavy batch processing.

Perhaps the most notable commercial systems in this category are Twitter directory services such as WeFollow, Listorious, and MyTwitterCloud. These websites provide rankings of influential users for many different topics, allowing people to find active, knowledgeable, and trustworthy users to follow. In WeFollow, once a user associates himself with a keyword of his choice, a proprietary algorithm ranks him against other users who have also opted-in with the same keyword. Recently, Twitter introduced a new feature, Twitter Lists\(^1\), which allows users to organize their followees into groups and give a name to each group. Services such as Listorious and MyTwitterCloud take advantage of the Twitter Lists generated by all Twitter users to index popular users based on their membership in other users’ lists. The aggregated list assignments are used to generate a ranking of users for a given query.

While these folksonomy-based systems can be helpful, they suffer from a number of disadvantages. First, the amount of data they can use is limited by what users will manually provide. For example, in WeFollow, a user must first opt-in to be included on a list, meaning that a large number of true experts will be missing from the list altogether. Similarly, the systems based on Twitter Lists rely on Twitter users utilizing this feature. Although growing in popularity, the Twitter Lists feature has by no means been universally adopted. Second, people use arbitrary vocabulary to associate users with topics, meaning that the topics an expert becomes associated with is determined by the particular word choices of a small number of people. For example, in these systems, a user could be associated strongly with the term *photography* but weakly with the term *photographer*. Ideally, users should be associated with topics, rather than particular typographical instantiations of those topics.

3 Ranking topically relevant users in Twitter

To help users discover potential experts to follow, we introduce a novel method of identifying and ranking users in Twitter according to their relevancy to any given topic. Our algorithm first performs a standard Twitter search (which returns a simple reverse-chronological list of results) to identify a small set of users who are associated with a query. It then applies a social filter, identifying users whose followers appear frequently in the search result. By combining a basic text search with a social ranking technique, the algorithm generates a ranked list of relevant and trusted Twitter users for any given topic.

3.1 Identifying candidates

The first step in our algorithm is to identify a set of candidates who are potentially relevant to the topic of interest. Given a topic expressed as a search term, a standard Twitter search is first executed using the Twitter API\(^2\). Taken alone, this search procedure is not particularly useful for identifying relevant users because the results are only a chronologically ordered list of the 1,500 most recent tweets containing the search term. However, those who published the tweets in the search result do form a small set of users, which we call Voters, who are associated with the topic.

The next step in our algorithm is to measure the opinions of the Voters by observing who they follow. If one user follows another in Twitter, it indicates that the first user values the information published

\(^1\)http://blog.twitter.com/2009/10/theres-list-for-that.html
\(^2\)http://dev.twitter.com/doc
by the second. Taking advantage of this fact, the algorithm next builds a set of users, which we call Candidates, by including anyone who is followed by at least one of the Voters. This process not only expands the set of potentially relevant candidates, it also provides a way to compute a relevancy score for each candidate, since a more influential, trustworthy Candidate will presumably be followed by more Voters.

For each user $u$ in the Candidates set, we retrieve the following two numbers:

- $f_u =$ the number of Voters who follow user $u$, and
- $F_u =$ the total number of Twitter users who follow user $u$.

The number $f_u$ can be explained by a process (depicted in Figure 3.1) where each of the Voters casts a vote for each of their followees, and $f_u$ is the number of votes received by user $u$. Using just the two numbers $f_u$ and $F_u$, we compute a relevancy score for each member in the Candidates set and rank them according to this score.

There are multiple benefits to basing our algorithm on a combination of the Twitter search mechanism and the social relationships in the network. First, any topic that can be expressed as a search query can be used. Second, the search mechanism returns a list of recent results (usually within the past day), so our algorithm can dynamically adapt to quickly-changing topics. Finally, since we don’t rely entirely on the search results, experts don’t need to have tweeted recently to be identified and highly ranked.

### 3.2 Ranking by relevancy

Once we have identified a set of Candidates and retrieved the relevant numbers $f_u$ and $F_u$ for each user $u$ in the set, we can compute the relevancy of each user to the topic. Before describing the formula used by our algorithm, we describe a series of alternative formulas of increasing complexity, building up to our own. For the remainder of this section, we will simplify the notation by writing $f$ and $F$ instead of $f_u$ and $F_u$, assuming the discussion is specific to a given user.

The first and most basic relevancy measure one could consider using is just the number $f$ itself. We call this measure NumVotes. This measure is appealing because it directly counts how many times a user’s followers have recently tweeted about the topic; however, in practice it tends to too heavily favor generally popular Twitter users who aren’t trusted or even relevant to the topic of interest. For example, a widely-followed user such as Barack Obama would rank very highly for virtually any search query.

Next, we consider the relevancy measure $f/F$, called DivF. This rationale behind this measure is that it counts the proportion (rather than the actual number) of one’s followers who showed up in the search results. Intuitively, the higher the proportion of a user’s followers who are associated with a topic, the more relevant that user should be considered. In practice, however, we found that this measure often overpenalizes generally popular users, underpenalizes unpopular users, and is overly sensitive to spuriously large values of $f$ when $F$ is small. Next, to strike a balance between the NumVotes and DivF measures, we consider the measure $f/\log F$, called DivLogF which takes its inspiration from TF-IDF methods from the information retrieval literature.

Finally, we introduce our preferred relevancy measure, called BetaBin($\alpha, \beta$). It is motivated from a Bayesian probability perspective. If we assume that each of the user’s $F$ followers is randomly
included in the Voters set independently and with probability $p$, then $f$ can be approximated by a Binomial($F, p$) probability distribution\(^3\). We use a Beta($\alpha, \beta$) prior distribution over $p$, so after observing $f$ of the user’s $F$ followers occurring in the Voters set, the posterior probability of $p$ follows a Beta($f + \alpha, F + \beta$) distribution. The expected value of this posterior distribution gives us an estimate of how probable each of the user’s followers is to show up in the Voters set, after observing the values of $f$ and $F$. The posterior expected value has a simple formula:

$$E[p|f, F] = \frac{f + \alpha}{F + \alpha + \beta},$$

which defines our relevancy measure BetaBin($\alpha, \beta$). Intuitively, this measure acts like NumVotes when $F \ll \alpha + \beta$, since $(f + \alpha)/(F + \alpha + \beta) \approx (f + \alpha)/(\alpha + \beta) \sim f$, and it acts like DivF when $F \gg \alpha + \beta$, since $(f + \alpha)/(F + \alpha + \beta) \approx f/F$. Thus, it has the benefits of DivF, measuring the proportion of one’s followers who are in the search results, while appropriately penalizing unpopular users like NumVotes does.

Since the proportion of a user’s followers who show up in the Voters set is expected to be quite low on average, it is generally a good idea to set $\alpha \ll \beta$. As such, in our evaluations, we compare three versions of the measure, all with $\alpha = 1$, with $\beta$ taking on the values $10^2$, $10^3$, and $10^4$.

### 3.3 Topic modeling

The algorithms described above take into account information about the link structure of the social network, restricting attention to sections of the graph highlighted by a simple text search over recent activity. Anticipating that useful information could also be found by a more thorough examination of user content (tweets), we also implemented a content-based algorithm. Based on the results of the previous algorithms, we compiled a list of about 28,000 Twitter users who were potentially relevant to a set of 10 queries: “biking”, “democrat”, “django”, “hadoop”, “medicine”, “photoshop”, “republican”, “startup”, “teaparty”, and “wine”. We collected the entire tweet histories of these users and ran the latent Dirichlet allocation (LDA) topic model on the corpus\(^4\). The LDA results provide a way of determining the topical similarity of any user to a search query based on the content of the user’s tweets. We used this method to re-rank the list of Twitter users found by the previous algorithms, using topical similarity to the search query as the ranking criterion.

### 4 Evaluation

To evaluate the various algorithms presented above, we first performed a modest case study on the search query “django” using two human volunteers, followed by a more thorough evaluation of five search queries using Mechanical Turk participants.

#### 4.1 Case study: “django”

As a preliminary investigation of the feasibility of the Bayesian ranking algorithm, we first compared the ranked list it generated for the query “django” (a Python web application framework) against the list of influential users provided by WeFollow for the same query and a list of expert Twitter users manually compiled by a Django expert.

Using Twitter’s API, we queried the term “django” on July 21, 2010, obtaining 1,500 tweets authored by 980 unique authors who formed the Voters set. Expanding to those users’ followees, we compiled 234,166 users who formed the Candidates set. The Candidates were ranked according to each of the relevancy measures defined above. We also collected the top 200 users for the same query from WeFollow\(^5\) on July 27, 2010.

We first measured the precision of each algorithm; that is, how many of each algorithm’s top-ranked users actually were relevant to the topic. We prepared the top 20 list for each relevancy measure

\(^3\)The Binomial approximation is not exact because it has support from 0 to $F$, while $f$ is actually bounded by the number of Voters returned by the search procedure, typically around 1500. The true distribution is hypergeometric.

\(^4\)We used $T = 500$ topics, with hyperparameters $\alpha = 0.5$ and $\beta = 0.1$, which gave the best perplexity scores out of 24 tested sets of hyperparameters.

\(^5\)http://wefollow.com/twitter/django
Table 1: Results from the case study for the search term “django”. The precision columns show the number of users in each top 20 list who were judged as relevant by two human raters. The recall column shows how many users from a list of 25 known experts were identified by each algorithm.

<table>
<thead>
<tr>
<th>Measure</th>
<th>Formula</th>
<th>Precision 1</th>
<th>Precision 2</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumVotes</td>
<td>( f )</td>
<td>7</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>DivF</td>
<td>( f / F )</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>DivLogF</td>
<td>( f / \log F )</td>
<td>13</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>BetaBin(1, 10^2)</td>
<td>((f + 1)/(F + 10^2 + 1))</td>
<td>15</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>BetaBin(1, 10^3)</td>
<td>((f + 1)/(F + 10^3 + 1))</td>
<td><strong>19</strong></td>
<td><strong>17</strong></td>
<td><strong>13</strong></td>
</tr>
<tr>
<td>BetaBin(1, 10^4)</td>
<td>((f + 1)/(F + 10^4 + 1))</td>
<td>17</td>
<td>15</td>
<td>11</td>
</tr>
<tr>
<td>WeFollow</td>
<td>N/A, proprietary</td>
<td><strong>19</strong></td>
<td><strong>14</strong></td>
<td><strong>10</strong></td>
</tr>
</tbody>
</table>

and merged them all together with the top 20 list from WeFollow, producing a list of 97 candidate experts. We recruited two Twitter users with Django experience and asked them to classify each of the 97 users as either relevant or irrelevant to Django. One participant identified 38 users as relevant, and the other chose 31 users. They agreed on 27 relevant users and 55 irrelevant users, disagreeing on 15 cases (Cohen’s kappa = 0.66, indicating substantial inter-rater agreement).

Next, we measured the recall of each algorithm; that is, given a list of known experts, how many of them were identified by each algorithm. We used a list of 25 recognized Django experts\(^6\) on Twitter compiled by one of the main developers of Django. We then counted how many of these users were present in the top 100 list of each algorithm and the top 100 list from WeFollow. We chose to use the top 100 lists because this is roughly the longest list one can be reasonably expected to look through when searching for relevant users.

The results of the evaluation are summarized in Table 1. The Measure and Formula columns give, respectively, the name of each measure and the formula it uses to calculate relevancy. The Precision 1 and Precision 2 columns give the number of users in each measure’s top 20 list who were rated as relevant to the topic by the first and second human raters, respectively. The Recall column gives the number of the 25 known experts who were found in each algorithm’s top 100 list. In the first precision evaluation, the BetaBin(1, 10^3) and WeFollow algorithms had the best performance, and in the second, the BetaBin(1, 10^3) algorithm alone had the best performance. In the recall evaluation, the BetaBin(1, 10^3) algorithm again performed the best. Interestingly, although the BetaBin(1, 10^3) measure is quite similar to the DivF measure, their performances on every evaluation were completely opposite. This suggests that while finding users whose followers are highly associated with the topic of interest is a good strategy, a major obstacle is being able to identify the users with only a few followers who received a relatively large number of votes by chance alone.

4.2 Mechanical Turk evaluation

Following the “django” case study, we performed a more thorough study of the performance of the various algorithms on five different search queries, using Amazon Mechanical Turk participants to rate the top-ranked Twitter users according to their relevance and expertise and whether they were worth following. The search queries we used are “biking”, “medicine”, “photoshop”, “teaparty”, and “wine”. For each query, we compiled each algorithm’s top-20 list and asked a number of participants on Amazon Mechanical Turk to rate each Twitter by agreeing or disagreeing with each of the following statements: “This Twitter user seems to be a source of relevant information relating to the search term.”, “This Twitter user seems to be an expert in an area relating to the search term.”, and “If I were interested in learning more things relating to the search term, I would follow this Twitter user.” Each Twitter user was evaluated a number of times, and a consensus was found among the participants.

The results are summarized in Table 4.2. In general, the WeFollow rankings and the LDA rankings performed very well. The Bayesian Beta-Binomial algorithms also performed well, achieving the highest scores in three cases and scores not far below the best in the other cases. These results

\(^6\)http://twitter.com/simonw/djangonaughts
Table 2: Results from the Mechanical Turk study. For each of the five search terms, the table lists the number of users from each algorithm’s top-20 list who were rated by Turk participants as having tweets relevant to the search term (r), being likely to be an expert in an area related to the search term (e), and being someone who the participant would follow if they were interested in the search term (f).

<table>
<thead>
<tr>
<th>Measure</th>
<th>“biking”</th>
<th>“medicine”</th>
<th>“photoshop”</th>
<th>“teaparty”</th>
<th>“wine”</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r e f</td>
<td>r e f</td>
<td>r e f</td>
<td>r e f</td>
<td>r e f</td>
</tr>
<tr>
<td>NumVotes</td>
<td>1 1 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>13 8 12</td>
<td>4 4 4</td>
</tr>
<tr>
<td>NumFollowers</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>0 0 0</td>
<td>1 1 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>DivF</td>
<td>1 0 0</td>
<td>0 0 0</td>
<td>1 1 1</td>
<td>0 0 0</td>
<td>0 0 0</td>
</tr>
<tr>
<td>DivLogF</td>
<td>1 1 1</td>
<td>1 1 0</td>
<td>1 0 0</td>
<td>16 10 14</td>
<td>12 9 10</td>
</tr>
<tr>
<td>DivSqrtF</td>
<td>2 1 1</td>
<td>2 1 1</td>
<td>3 3 3</td>
<td>12 8 10</td>
<td>4 3 3</td>
</tr>
<tr>
<td>BetaBin(1, 10^2)</td>
<td>7 6 6</td>
<td>8 6 6</td>
<td>4 4 4</td>
<td>13 6 12</td>
<td>19 13 11</td>
</tr>
<tr>
<td>BetaBin(1, 10^3)</td>
<td>18 14 14</td>
<td>16 11 11</td>
<td>9 8 8</td>
<td>16 8 13</td>
<td>16 11 11</td>
</tr>
<tr>
<td>BetaBin(1, 10^4)</td>
<td>18 17 17</td>
<td>15 6 8</td>
<td>16 10 10</td>
<td>11 8 7</td>
<td>18 12 11</td>
</tr>
<tr>
<td>BetaBin(1, 10^5)</td>
<td>17 13 13</td>
<td>15 11 10</td>
<td>15 8 7</td>
<td>15 10 12</td>
<td>18 14 12</td>
</tr>
<tr>
<td>BetaBin(1, 10^6)</td>
<td>4 3 4</td>
<td>2 2 1</td>
<td>6 2 2</td>
<td>15 10 14</td>
<td>18 13 13</td>
</tr>
<tr>
<td>LDA</td>
<td>20 16 16</td>
<td>14 10 11</td>
<td>20 20 20</td>
<td>11 5 9</td>
<td>20 20 20</td>
</tr>
<tr>
<td>WeFollow</td>
<td>19 18 17</td>
<td>17 14 14</td>
<td>18 16 14</td>
<td>16 11 11</td>
<td>19 16 14</td>
</tr>
</tbody>
</table>

suggest that a content-based algorithm such as LDA can be a powerful tool to augment a network-based analysis of the relevance of Twitter users.

5 Conclusion

This paper introduces a technique to enhance the ability of social network users to identify relevant sources of information for a given topic. By combining a basic text search with an analysis of the social structure of the network, the algorithm generates a ranked list of relevant users for any given topic. Because the search mechanism returns a list of recent results, our algorithm adapts to temporal trends of topics. Also, it requires no manual tagging or special mechanisms beyond the ability to search over the messages in the network and query the social relationships of a small set of users. We found that a content-based topic analysis of the social network proved especially useful in identifying relevant, expert, and useful users to follow.

To investigate the feasibility of the algorithm, we performed a case study and a more thorough evaluation, comparing rankings generated by the algorithm with rankings provided by a commercial website. The algorithm shows great potential to help users identify interesting users to follow in Twitter. We hope that this research will inform the design of recommendation systems for Twitter and other social networks.

References


