Seeder Finder - Identifying Additional Needles in the Twitter Haystack

Nick Gramsky[1] Hanan Samet
Center for Automation Research, Institute for Advanced Computer Studies,
Department of Computer Science, [1]MIND Lab, University of Maryland
College Park, MD 20742, USA
{ngramsky, hjs}@cs.umd.edu

ABSTRACT

TwitterStand is a novel way to track the news cycle by allowing people to view and browse the news with a map query interface. TF-IDF scores for each document that is linked to by a tweet (also termed *twanchor* [22] when the document is a news article) are calculated after they enter the system and pass initial classification filters. These scores are used to cluster similar tweets. Clusters must contain tweets from reputable sources in order for the clusters to form. These reputable sources are known as seeders as they essentially seed a cluster. Seeders have become an integral part of the TwitterStand architecture. An optimal system monitors the set of seeders in order to find newsworthy tweets quickly.

This paper proposes methods to improve the current list of seeders by augmenting the pool with previously undiscovered users while routinely eliminating those that do not bring any value. We consider a successful seeder one who is timely in the reporting of large newsworthy events. An analysis of the current seeders precedes a proposed approach and serves as the basis for quantifying future seeder churn. A qualitative analysis based on that approach is conducted in an effort to quantitatively evaluate the process.

Keywords

Social Networks, Big Data, Information Storage and Retrieval

1 Introduction

A recent study from March 2012 shows that Twitter and its users currently produce over 350 million tweets a day [4]. Despite studies [3] that show that news topics comprise less than 5% of all tweets, the sheer volume still allows for the medium to monitor, track and analyze the pulse of the world. Through all of the random thoughts, conversations and topics that traverse the Twitter medium, news and topics emerge through the clutter and people have found ways to make sense from the large amount of noise. Twitter users have been shown to diffuse information faster than seismic waves [9] and provide pre- and post-election analysis [21] and predictions. Even the Central Intelligence Agency (CIA) produces a daily report destined for the president [37] on the sentiment and happenings around the world. Due to the volume and velocity of this data, combing through it in real-time provides challenges both in the ability to simply handle the data, much less decipher the news from the noise.

The TwitterStand system [34] represents an approach to overcome the above challenges by providing a method to find and track the news cycle by making use of a map query interface. In particular, the goal is to determine the locations about which people are tweeting rather than the locations from which they are tweeting. It has also been used to identify future events [14] and is a continuation of our work on making spatial (e.g., [11, 26], and temporal (e.g., [31]) data first class citizens in database applications. TwitterStand is implemented as a database of tweets [18] and supports operations such as spatial joins (e.g., [12]) and nearest neighbor finding (e.g., [27, 33]). TwitterStand is related to the NewsStand stem for reading news with a map query interface (e.g., [18, 32, 28, 36]) and involves geotagging which is the process of identifying text which corresponds to geographic locations (e.g., [5, 16, 17, 20, 19, 25]).

TwitterStand was developed to filter newsworthy tweets from noisy tweets in real-time. The goal was to provide an aggregation of up-to-the-minute news from tweeters around the world via a single website while providing a unique level of context. Once filtered and clustered, the system provides a novel way of displaying and browsing the news cycle. Rather than simply exploring what is being talked about, users can explore the geographical locations that are being discussed in a manner similar to geographic browsers developed by us such as the SAND Browser and its predecessors [30, 29]. The key idea is to make use of the additional information available via the links that are used by Twitter users to both expand and amplify the amount of information that they are sending by virtue of the 140 character bound on the contents of the messages. These links often point to news articles (such a tweet is termed a *twanchor* [22]) and this is the type of additional information that TwitterStand uses to identify locations within the text of tweets and attached articles which are displayed on a map. Browsing methods allow users to both see what locations are being discussed as well as listing the discussed topics. Clusters are plotted on maps as well in addition to simply being listed.

Tweets that drive this system are gathered from one of two methods depending on whether the source is a select group of individuals determined to be worthy of following (termed *seeder feeds* or a more general subset of individuals known as the *Gardenhose feed*. The seeder feed utilizes the Twitter API method *filter* to monitor the tweets of a select group of Twitter users. These users (seeders) are considered to be newsworthy and prompt in the reporting of news via Twitter. The Gardenhose feed uses the Twitter API method *sample* to gather tweets from random users. Roughly a random 10% of the given Tweets at any time are gathered through the Gardenhose process. Both functions are real-time streaming methods that produce Tweets as they occur.

The following explains the mechanics behind the clustering. TF-IDF scores for each document are calculated after they enter the system and pass initial classification filters. Cosine similarities are computed for the document against all other existing clusters. The system searches for the closest cluster that falls within the minimum distance of a document and adds the tweet to that cluster. A tweet, or linked news article, can be assigned to at most one cluster. Tweets from seeders that do not find matching clusters will form new clusters. If the tweet originates from the Gardenhose feed and does not fall into an existing cluster, then it is discarded. It is important to note that a cluster will not form without a seeder.

The prior implementation utilized the existing pool of seeders selected as the system was initially developed. A seeder is a Twitter user who is chosen to seed the creation of clusters. The current implementation of TwitterStand includes 1801 seeders. This set has remained consistent as there was no mechanism to evaluate or replace seeders. Prototype modules were built within TwitterStand to find and monitor friends of seeders. The idea behind the use of these modules was to gather additional tweets from users that likely had similar interests and augment the Gardenhose feed. While the idea of promoting Twitter users to seeders was discussed, there was no mechanism to promote friends or demote seeders. The identification of seeder friends is a valid method to find other important tweeters and potential seeders, though it should not be the only method in place.

In particular, social media aggregation systems should search for additional sources that extend beyond the social network of existing users. Parisar [23] warns of the filter bubble, a phenomenon that exists within search engines and social media web tools. The idea is that algorithms and tools can limit future search results if they are based or filtered on a person's interests, friends, etc. This 'bubble' will shield you from possible search results that lie outside of your interests. In Parisar's TED talk [24] he cites an example of two different sets of search results for the same topic for two different users. By not reaching outside of the realm of the existing seeders and their followers, TwitterStand is at risk of containing itself within its own information bubble.

Among others, two of TwitterStand's goals are to identify news cycles as quickly a possible and find those topics which most of the Twitter universe is tweeting about. The time at which any one seeder or Twitter user tweets to deliver a URL about a particular topic is critical. Delivery of the news should be quick and one goal of seeder identification is that of identifying those users who deliver the news the fastest. As TwitterStand strives to find the most talked about news topics, users who tweet about the highly discussed items should be assigned a higher rating than those who do not. Both time and cluster size will be used to identify potential seeders as well as score both potential and existing seeders.

In this paper we propose a method to identify and monitor potential seeders with the intention of possibly promoting them into the seeder pool. We go beyond the social network of the existing seeders in an effort to find additional reputable Twitter users. The aim of the work is to enhance the quality of the seeder pool by finding users who report the news in a timely fashion and outline a system that will identify both potential seeders and seeders in order to evaluate them. The rest of the paper is organized as follows: Section 2 discusses related work. An analysis of the current seeders is presented in Section 3. The proposed changes for an on-line method are outlined in Section 4. Section 5 evaluates the results of potential seeders as they underwent a trial in the system. These results are used to refine the methods discussed in Section 4. Section 6 surveys future work, while Section 7 contains concluding remarks.

2 Related Work

TwitterStand, the system SeederFinder augments, presents news in a rather unique fashion. Newsworthy tweets are clustered and locations are extracted such that one can geographically browse the news based on the locations being discussed. Several other systems have attempted to merge news and social media with GIS platforms, though these other systems do not quite do so in the same way. BuzzTracker [1] is an online system that aggregates the news, and reports scores for cities based on how much news is being discussed about that city. BuzzTracker visualizes how much news is being reported about a city and links cities with common stories through link-node diagrams. Though this enables news discovery through GIS tools, the interface is limited and it's fixed map interface only allows one to just view the very top cities of the current day. TwitterStand, on the other hand, allows one to discover stories in any town or jurisdiction by simply browsing.

TrendsMap [2] is similar to Twitterstand as it allows a user to use a GIS interface to browse to the local level and discover the latest Twitter data. However, this system differs from TwitterStand in two ways. First TrendsMap tracks trends in the Twittersphere. Trends are simple keywords or phrases that Twitter produces through its API. Additionally TrendsMap locates trends on the maps based on the location of the users submitting the tweets and the not the locations within the Tweets.

Several attempts have been made to identify Twitter users that are beneficial in one form or another. The goal of such work typically involves assisting in the search for similar users by finding Twitter users that are like other existing users in some fashion. The aim of our approach is to find top-ranked Twitter users for seeding news of any topic. Hannon [10] set up a system to find Twitter users that individuals would most find interesting or useful in sharing topics they enjoy. Using TF-IDF scoring to find distinguishing terms, the system evaluates Twitter members in an effort to find Twitter users who tweet about similar topics. Upon searching for a particular term or topic, the system returns top tweeters. New users are suggested as ones to follow if both the user performing the search and resulting users are considered similar in scoring, suggesting that people should follow others due to similar interests.

General recommendation systems typically look like the one developed by Chen [8] to recommend URL's (websites, links to news stories) and content to users based on the topics that they write about. By monitoring a users Tweets, their followers tweets, common URL's amongst followers and other Twitters actions, the authors were able to recommend content (URLs). The work resulted in a website (now defunct) that made the recommendation for a user.

Other systems include methods to find influential Twitter users. Bakshy [6] and Sun [35] both look to find influential Twitter users. Both authors utilize the social network structure around the Twitter users in order to quantify the Twitter users. Bakshy uses diffusion and cascading models to identify who the best Twitter users would be. The better a user diffuses information the more influential they are considered. Sun uses standard graphing metrics such as in-degree and node centralilty as the basis for scoring a user. We, however, are looking for Twitter users of a certain characteristic - namely reporting the largest news stories in a timely manner. It's only natural, though not necessary, that such users are immediately influential.

Perhaps most similar to our approach is the work performed by Canini et al. [7]. Their goal was to identify credible Twitter sources. They first studied the criteria that caused users to trust other Twitter users and then applied these criteria to formulate a mechanism to score and rank Twitter users. Twitter users were first pulled from the website www.WeFollow.com. Mechanical Turk workers were then tasked to evaluate the set of users to help identify why some users trust other users. After applying those same methods, the authors produced a top-20 list for particular topics and again had Mechanical Turk users evaluate the rankings by determining if the individual Twitter users were considered a relevant source of information for the given topic. While this approach attempts to rank Twitter users across the entire medium, the motivation for the ranking mechanism is based on trust and not quick news delivery or content. Additionally, this method works to evaluate a pre-existing list of users and not find new ones from the Twitter landscape. We have yet to find any work that discusses how an online system can discover, evaluate and rank users for the use and benefit of the system.

3 Current Implementation

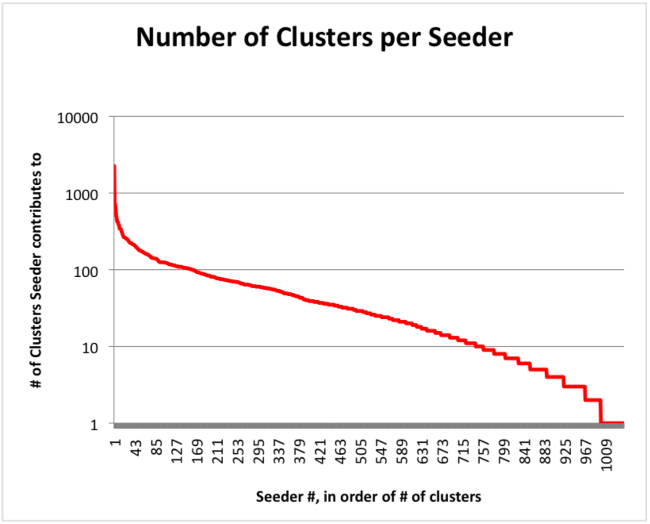


Figure 1: Distribution of cluster contributions across the seeders for data collected during one week in April 2012. One seeder (@HenryNews, ranked #1 in NumClusters) contributed to over 2249 clusters in the 5-day trial. His average Tweet rate often exceeds 45 tweets/hr. The y-axis is the number of clusters to which the seeders contribute and the x-axis are seeders, ranked from 1 - 1048 in order of their contributions. The 753 seeders that failed to contribute to any cluster during that week are omitted in order to yield a more legible graph.

TwitterStand has access to the Gardenhose feed from Twitter, giving it access to stream 10% of all Twitter tweets. This is roughly 35 million tweets a day. All of these tweets are subject to a classifier that helps separate the news tweets from the noise. An existing corpus of newsworthy and non-newsworthy tweets is used with a Bayes classifier to identify newsworthy tweets. Currently this classifier is allowing between 4-8% of the tweets from the Gardenhose feed to pass the filtering check and onto the clustering phase. This number is in line with previously mentioned studies [3] that suggest less than 5% of all tweets are news worthy. Of those 4-8% tweets , only 30% ever become members of clusters. This means only 1-3% of all tweets that we gather from the Gardenhose feed ever become members of a cluster.

In order to evaluate a seeder's contribution to TwitterStand, we have to first define the important traits of a user and provide a mechanism to quantify and monitor that behavior. There are three areas we quantify in order to evaluate a Twitter user: the number of clusters to which a user contributes; how many other users are also talking about the topics a user tweets about; and the timing of a user's tweets. We define **NumClusters** as the number of clusters a to which a user's tweets contribute. A tweet can only join one cluster, so at most this can be the number of tweets a user produces. We define **TotalTweets** as the total number of Tweets in all clusters to which a user contributes. For example, if a user produces 10 Tweets that contribute to 10 clusters and each cluster has an average of 200 Tweets, then the user's TotalTweets would be 2000.

We define two different metrics that quantify when a user tweets - an overall score **(RawTiming)** and an average score **(AvgTiming)**. RawTiming gives an indication of how large clusters are and how quickly users tweet about a topic. AvgTiming helps compare one Twitter user to another. This is similar to calculating the Earned Run Average (ERA) of a pitcher in baseball. The calculations of RawTiming and AvgTiming are done with the aid of the following constructs:

1. Let TW_C denote the set of tweets in cluster C, where we assume that each tweet belongs to just one cluster C.

2. Let $Rank(t, C)$ be the rank of tweet t within cluster C in terms of the order of its publication in C. Thus the first tweet in C has rank 1, the second tweet in C has rank 2, etc.

3. Let N_C denote the number of tweets that are contained in cluster C. Let $t_{u,C}$ be user u's first tweet that belongs in C. Let $Score(u, C)$ be the total number of tweets in cluster C minus the rank of $t_{u,C}$, user u's first tweet that belongs in C. In other words, $Score(u, C)= N_C$ - $Rank(t_{u,C}, C)$.

4. Calculate $Score(u, C))$ for each cluster C to which user u contributed. Now, sum up $Score(u, C))$ for all clusters C to which user u has contributed tweets and the result yields $RawTiming(u)$ which serves as an indicator of both how quickly the user tweets about events and how large these clusters are.

5. Calculate $AvgTiming(u)$which is the relative ranking of user u's first tweets for each cluster over all the tweets in these clusters. It is formed by summing up the values of $Rank(t_{u,C}, C)$ for tweets $t_{u,C}$ of user u for each cluster C and dividing by the sum of the tweets in each cluster C. In other words, $AvgTiming(u) = \frac{\sum_C Rank(t_{u,C},C)}{\sum_C N_C}$. This serves to give an indication of the timeliness of user u's tweets.

Table 1 provides a summary of the four metrics discussed above as well as an example of how each one is calculated. We cannot calculate these metrics in real-time as the values keep changing as a cluster grows. Thus these metrics are calculated each time a cluster is marked as inactive. A cluster is marked inactive and removed from the cache database after 3 days of no additional contributions; that is, no tweets for 3 days are scored so that the distance between the tweet and the cluster places the tweet within the cluster. In the event that a user contributes more than one tweet to a cluster, then we restrict the scoring using the above metrics to only the first tweet.

We gathered data on the existing seeders using the above metrics. For one week in April 2012 we ran TwitterStand and analyzed the data to get a sense of what the seeder activity looked like. We calculated the above metrics in an effort to baseline the seeder population. Cluster membership seems to adhere to a power-law distribution as expected. However a surprising discovery was that there exist a large number of seeders (753 out of 1801) who did not contribute to any

Metric	**Description**	**Example**
Number of Clusters (NumClusters)	The total number of clusters to which Twitter users contribute. Clusters are the items we post to the front-end. Each cluster is a news cycle item so that the number of clusters to which a seeder contributes is essentially the number of news stories about which the seeder tweets. The fact that users contribute to the same cluster a multiple number of times does not increase NumClusters. Each cluster is counted only once per user.	A seeder publishes 5 Tweets and each Tweet is a member of a different cluster. NumClusters = 5
Number of Tweets (TotalTweets)	The number of tweets that encompass a cluster. Simply creating clusters that contain just a few tweets does not add value to the system. High tweet-count clusters indicate both the attention a news item is receiving as well as provide a composite "pulse of the world". This is summed up over all clusters to which a user contributes in order to show the relative importance of these clusters.	The clusters from the above seeder example are sized 100, 200, 300, 400 and 500 respectively. TotalTweets = 1500
Tweet Timing (RawTiming and AvgTiming)	The timing of the clusters to which a user contributes. RawTiming indicates the relative importance of a user's first tweet about each cluster with respect to the user's other tweets about these clusters and corresponds to simply adding the Scores of the user's tweets about them. However, simply tweeting about a cluster c that many other users tweet about is important but not as important as being one of the first users who tweet about c. Therefore, we measure the importance of a user's tweets by summing up the rank of their first tweet about each cluster and dividing it by the total number of tweets in these clusters. This is a variant of an average where ideal AvgTiming values approach 0 as this provides an indication of how early the user's tweets enter their clusters.	Continuing the above example, suppose that the seeder contributed the 5th tweet in each cluster. RawTiming = (500-5) + (400 - 5) + (300 - 5) + (200 - 5) + (100 - 5) = 1475. AvgTiming = (25 / 1500) = 0.016667

Table 1: The four quantitative metrics used in seeder identification. Each metric is summarized and an example of each is provided. These metric are used to score seeders and potential seeders.

cluster at all during the week that we ran TwitterStand! Figure 1 shows the distribution of clusters across all seeders who contributed to clusters during that week.

We use this data in a multitude of ways. First it enables us to dig deeper into the seeders who contribute to zero clusters. Using the Twitter API we tabulated the total number of tweets for such seeders over their lifetime. We found that 28 of the seeders had yet to produce their first tweet as a Twitter user! Furthermore, 100 seeders (or 5%) had only managed 25 tweets or less, 155 seeders (8%) have only managed 100 tweets or less in their lifetime and 296 seeders (16%) had produced less than 725 tweets - i.e., one tweet a day for the past 2 years. While analysis and suggested measures will still follow, we take this time to note that seeders that have never tweeted should be removed from the seeder pool.

Once the non-contributing seeders have been purged, we can use the above statistics as a mechanism to differentiate between top contributors and those whose contributions leave more to be desired. We will later use the scores of seeders whose respective RawTiming, TotalTweets and NumClusters values fall in the bottom 10% as the starting point for seeder removal. For example, if we start to ingest a large number of potential seeders but cannot continue to ingest more seeders, we might trim the same number of seeders from the seeder pool. These methods are discussed further in the following section.

4 Proposed Changes

4.1 General Changes

As the need for seeder augmentation and a rough understanding of how to analyze our existing seeders exists, we now turn our attention to proposed changes to TwitterStand. These changes will facilitate the automatic promotion of Twitter users to the seeder level. As only 1-3% of all tweets that TwitterStand encounters are ultimately clustered and contribute to the voice of the news cycle, we will focus on finding the timely tweets in this set that were a part of large clusters.. Within our existing list of seeders we have found many of the proverbial 'needles in the haystack', those who create timely newsworthy Tweets, but have we found them all? The goal is to find other micro-bloggers who contribute to this minority of tweets. Furthermore, we are looking to identify those users who report the news quickly, often, and contain content that the world at large finds interesting, in that order.

We cannot, for various reasons, keep promoting new users to seeders as at some point we will become unable to monitor these users. First, the Twitter API stream method 'filter' limits the number of users one can monitor. Secondly, we do not want to waste time following users that never tweet about anything newsworthy or simply create clusters of sizes of 1.

Lastly, our scoring system can become costly, especially as it could be performed over the entire Twitter user base if not regulated. Thus we need to keep a manageable seeder and potential seeder pool in order to handle the extra load of ranking seeders and clusters. We propose to implement a decay function to allow new potential seeders to rate against existing seeders. As a seeder's membership may extend over years, simply accumulating stats will not allow for a fair comparison between existing seeders and potential seeders. New potential seeders will only have stats for a short period and may never overcome existing seeder scores. Decaying scores is one way of removing the bias that may exist from old clusters. Such a decay mechanism will force any seeder to maintain some consistent level of contribution to the system while providing an opportunity for additional worthy Twitter users to become seeders.

```
Algorithm 1: Demote N seeders from seeder pool.
N = number of seeders to demote.
NT_S = TotalTweets of all clusters to which user S contributes
NTT = Average Total Number of Tweets (TotalTweets metric) for seeders in bottom (4*N) (Seeder_Pool_Size) % of seeders
CA = Average number of clusters in the bottom (4*N) (Seeder_Pool_Size) % of seeders
S_i = Potential seeder, i, that may be removed from the seeder pool.
PS - S_1,S_2,..S_4N - Set (size 4*N) of potential seeders (S_i) to be removed, ranked by RawTiming.
SS_1, SS_2, SS_3, SS_4 - Above set (PS) broken up into 4 equal, continuous parts.
Resulting sets are the bottom 4 tiers of seeders, based on RawTiming.

i = 1
while N > | demoted seeders |
    Demote all S ∈ SS_i where NumClusters_S < CA
    for k = 1; k = k + 1; k ≤ i
        # Visit SS's based on k-index. First visit just 1, then 1 and 2
        # Start with a low threshold the first time we visit a group SS_k
        # Each time we increase the threshold for a group as we revisit it
        Demote all S ∈ SS_k where AvgTiming_S > 1 - (.25) · (i - k + 1)
        Demote all S ∈ SS_k where NT_S < NTT · (.25) · (i - k + 1)
    end for
    i++
end while
```

4.2 Specific Changes

Our first step is to identify potential seeders by finding quality users out in the wild. Clusters are automatically removed from the TwitterStand cache database once they fail to have been updated for 3 days. For all clusters whose tweet count is over 100 we look for non-seeders who contributed to the cluster very early on. We will rank all tweets based on how early the tweet entered the cluster and note any non-seeders who published one of the first five tweets in that cluster. The idea is that we have now found a Twitter user who was one of the first people to break/tweet about a story of relevant size. Users that meet these criteria are promoted to the potential seeder pool where their statuses are monitored using the filter API method and treated like seeders. Clusters can be created by the tweets of these users and their scores are augmented and decayed just as a seeder would be.

Scoring will take place by calculating the metrics in Table 1 above. Each time a cluster is removed from the cache database NumClusters, TotalTweets, RawTiming and AvgTiming are updated for both seeders and potential seeders who contributed to that cluster. The existing RawTiming, NumClusters and TotalTweets scores will increase by the amount for the corresponding cluster. The AvgTiming will be recalculated with these new values.

News cycles can, however, go on for quite long periods. As we only score the users whose tweets are contained in a cluster once the cluster becomes inactive, a cluster that never dies theoretically may never allow its contributing users to get a score. Leskovec's [15] performed a study of memes (ideas, style or topic that spreads throughout a culture or news) in the news cycles. In his study he provides a good illustration of how some news cycles can persist for quite some time. Recall that a cluster remains in the cache database and is considered active as long as a contribution is made every three days. Therefore, we will consider clusters that continue to have tweets contributing to the cluster for more than a month as inactive in order to score this cluster. This is done 1) to ensure we find potential seeders and 2) to enable scoring of existing seeders, both of which otherwise might not have happened if the cluster lives forever. Of course, the cluster remains in the cache database. This tracking/scoring mechanism requires a few changes to the TwitterStand database. In particular, the system will now need to track when the clusters were created in order to track the lifespan of a particular cluster.

All pre-existing scores will be decayed once a week in order to prevent existing seeders from distancing themselves from potential seeders. We will decay RawTiming, NumClusters and TotalTweets scores based on three decay rates, RTDecay (RawTiming Decay Rate), NCDecay (NumClusters Decay Rate) and TTDecay (TotalTweets Decay Rate). The RTDecay, NCDecay and TTDecay rates will be that of the RawTiming, NumClusters and TotalTweets values that fall in the bottom 10% of all scores respectively. Scores for all seeders and potential seeders will decrease according to these decay rates. In order to facilitate the incorporation of the decay, we will simply subtract the value of the decay rate from the corresponding metric for every user.

To illustrate the assignment of decay values let's suppose we have 100 RawTiming values from 100 seeders in a reverse-sorted array from 1 to 100 with the value in the first element of the array being the largest value and element 100 being the smallest. The values in array elements 91 through 100 are in the bottom 10% of all seeders. As all values of array elements 91 through 100 are in the bottom 10% or lower, RTDecay would be set to the value of the 91st element in the array of RawTiming values. All RawTiming values for all seeders would be reduced by this RTDecay value. A new RTDecay value is calculated each time scores are to be decayed. This same method is used to calculate the NCDecay and TTDecay values.

No scores will ever go below zero as the decay function is applied. As AvgTiming is an average, it will remain unchanged and thus we will not decay this quantity. Additionally as AvgTiming is an average of cumulative RawTiming and NumClusters throughout the lifetime of the seeder, copies of these scores (unchanged by decay) will need to kept. However, copies of these cumulative scores will not be used in the evaluation of seeders.

At the first of each month scores will be evaluated across seeders and potential seeders. Promotions/demotions occur each month in order to give any user a good chance to contribute to news cycles. Local news agencies and newspapers might not have a large amount of news in a short period of time. Thus a month time period is used to account for possibly slow news weeks. Potential seeders that have RawTiming values higher than the 10th percentile of seeders, have contributed to more than 2 clusters with a count of over 200 tweets and have AvgTiming $< .5$, will be promoted to the

Potential Seeder	**Raw Timing**	**Num Clusters**	**Total Tweets**	**Avg Timing**
1	4	2	7	0.428571
2	4	2	6	0.33333
3	4	2	6	0.33333
4	4	3	7	0.428571
~~5~~	~~3~~	~~1~~	~~14~~	~~0.785714~~
6	3	2	5	0.4
~~7~~	~~3~~	~~1~~	~~4~~	~~0.25~~
8	3	2	5	0.4

Table 2: Example seeder ranking to demonstrate demotion of 2 seeders.

seeder pool. Potential seeders that have not been promoted within 3 months are dropped from the potential seeder pool.

With promotions however come demotions. We cannot add users to the seeder pool forever without having to remove seeders. We establish high-water and low-water thresholds for the seeder-pool size, so that we can demote seeders if the seeder pool size exceeds the high-water threshold and bring the size back down to the low water threshold. Seeders will be removed based on finding the seeders with the lowest RawTiming, NumClusters, TotalTweets and highest AvgTiming scores. Recall that AvgTiming is a value in [0,1]. A value of 1 indicates that every tweet the seeder publishes is the last tweet in the respective cluster. The closer AvgTiming is to 0 the earlier all tweets are published within all clusters to which the seeder contributes.

As we monitor and score users across various metrics, there is no one score that defines the 'least valued seeder'. We let the RawTiming metric be the primary key. Therefore, our demotion process consists of cycling through the different metrics of all seeders who have the lowest RawTiming scores. Given that we need to remove a set of N seeders, we gather the bottom 4*N seeders, ranking them in ascending order by RawTiming. These 4*N seeders are broken into 4 continuous (based on RawTiming rankings) groups. These groups represent the bottom four RawTiming tiers of the seeders to be potentially removed. We cycle through each of these groups, removing the seeders with the lowest NumClusters, TotalTweets and highest AvgTiming scores.

Algorithm 1 shows one possible method of finding the users that satisfy the 'lowest score' across all four metrics. In this algorithm we use two variables i and k to cycle through the different sets and thresholds. Tables 2, 3 and 4 illustrate the workings of the algorithm for an example where the variable i is used to indicate the number of subsets that we take into consideration at each iteration and the threshold values for the metrics to indicate demotion. The variable k is used to distinguish between the subsets that are being considered at each iteration of the algorithm. Our example illustrates that the effect of varying i is to subject the lower-ranked seeders, based on *RawTiming*, to higher standards each time we revisit them in a subsequent iteration of our demotion algorithm.

Our example starts by having a set of seeders of which two are to be demoted (i.e., $N = 2$). This means that we will examine the $PS = 4N = 8$ lowest ranked seeders according to their RawTiming values. In the first iteration of the algorithm $i = 1$. As we cycle through the for loop for $i = 1$, we see that k can only take on the value of 1 which only allows us to check the scores of the bottom two seeders. In this example, the value of CA, the average number of clusters in the bottom 8 seeders is 1.875 and the value of NTT, the average number of *TotalTweets* in the bottom 8 seeders, is 6.75. Thus at this step, the algorithm removes any seeder in SS_1 (the bottom 2 seeders) that has a *NumClusters* value below 1.875, an *AvgTiming* value higher than .75 and *TotalTweets* < 1.6875. The *AvgTiming* threshold is .75 as $1 - (.25) \cdot (i - k + 1) = .75$. The *TotalTweets* threshold is 1.6875 as $NTT \cdot (.25) \cdot (i - k + 1) = 1.6875$. Observing the values of the bottom two seeders we see that SeederID 36496825 is removed as his *NumClusters* value of 1 is less than 1.85. SeederID 17411453 is not removed as his scores for *NumClusters*, *TotalTweets* and *AvgTiming* are within the permissible values.

Our next iteration sets $i = 2$. This allows k to take on values of 1 and 2 as we cycle through the for loop. Before entering into the inner loop we examine SS_2 and find that seeder 5 has a *NumClusters* value of 1, which is below the threshold of 1.875 and is removed. The first time through the for loop we revisit the same set of seeders we evaluated in the previous iteration of while loop. Seeder 7 has already been removed so we only observe seeder 8. This time, however, we are subjecting the retention of seeders in this set to higher standards. As $i = 2$ and $k = 1$ our *AvgTiming* threshold has changed to 0.5 as $1 - (.25) \cdot (i - k + 1) = .5$. Similarly the *TotalTweets* threshold has changed to 3.375 as $NTT \cdot (.25) \cdot (i - k + 1) = 3.375$. Seeder 8 is still within these bounds so we continue to retain him. Continuing through the while loop we now set $k = 2$ and visit the next highest set of seeders as ranked by *RawTiming*. Here the thresholds for these two seeders are the same values we subjected the first two seeders in the very first iteration of the algorithm, namely 1.875 for *NumClusters*, 1.6875 for *TotalTweets* and .75 for *AvgTiming*. Seeder 6 is well within the necessary thresholds and will remain through this iteration. We now conclude the algorithm as we have demoted the necessary two seeders.

4.3 Choice of Metrics for Demotion

Though we defined four separate metrics for quantifying a seeder's contribution to the system, we chose RawTiming as the primary key when selecting a seeder or potential seeder for removal. While other metrics could have been used, we feel that RawTiming is the best choice because it is a product of both the rank and size of the clusters to which a seeder or potential seeder contributes. This score, however, is not negatively impacted if the seeder is occasionally untimely. Our goal in demoting a seeder is to find one that routinely does not contribute in a valuable way rather than seeders that might occasionally be late to break a story. If a seeder is always one of the last to contribute to a cluster, the value of RawTiming will always be near one. This is true if the user contributes to very large clusters or clusters simply of size 1. Being the first to contribute to a large cluster increases a seeder's RawTiming score, but occasionally being the last one to an equally popular cluster will not decrease it.

Comparing RawTiming with AvgTiming, we find the opposite to be true. A seeder might, just one time, be the last to contribute to a very large cluster. This unfortunate timing might overshadow other contributions that are an indication of a valuable seeder. For example, consider the seeder who is first to produce a tweet for 5 clusters, each comprised of 100 tweets. This same Twitter user might happen to tweet about a very popular topic where he gives his opinion on the matter, but be the last to do so. If this last cluster is of size 10,000, this will result in an AvgTiming score of 0.975. This is a very high AvgTiming score and we might very well end up in removing this user despite the fact that regularly break news in a timely fashion. Table 3 shows the bottom 8 seeders from our example TwitterStand scenario as ranked by AvgTiming. Note that while several have very high RawTiming scores, there are a few that produce some tweets that contribute to sizable clusters. While the RawTiming shows they are not the earliest of users to break news, they are

Potential Seeder	Raw Timing	Num Clusters	Total Tweets	Avg Timing
A	4	1	39	0.897436
B	48	4	487	0.901437
C	42	3	661	0.93646
D	1	3	16	0.9375
E	16	2	290	0.944828
F	43	8	962	0.955301
G	57	11	1524	0.962598
H	7	2	3294	0.997875

Table 3: Bottom 8 seeders as ranked by AvgTiming. Note how the seeders with the worst scores, at times, have very high TotalTweets scores. A tweet that is grouped with a large cluster can unfairly weight AvgTiming.

still tweeting about relevant newsworthy items. Observing the TotalTweets values for these seeders we see high values for a couple of them, suggesting the clusters contain topics that many people across the general public are discussing. So while AvgTiming may produce users that are always untimely (though our hypothetical example produces a situation where that is not the case) we're not necessarily identifying users that do not discuss newsworthy, popular events. This is only half of the criteria we proposed to define a valuable seeder in the beginning of the paper, so we omit this as a possible primary key.

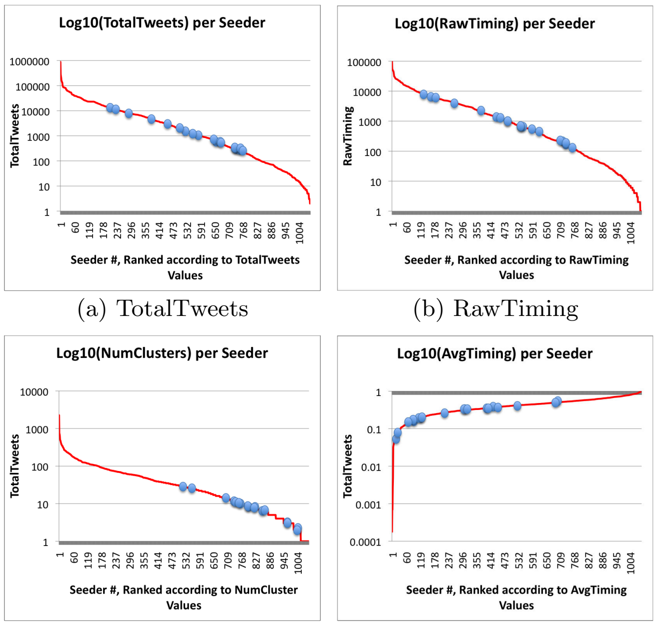
Observing NumClusters we find a similar situation which may result in the removal of niche seeders. Again consider a hypothetical situation, only this time several seeders only tweet once a week or so but break news or post a blog entry that many people tweet about. A collection of users who do not tweet very often but break very popular news could become grouped at the bottom of the seeder pool if NumClusters is the primary key. While we may eventually need to remove valuable seeders, our goal is to remove the least valuable. Using NumClusters to rank seeders for removal has the drawback that once again we only consider one dimension of their value, namely how often they tweet.

TotalTweets could have possibly been used as a primary key, though it is also a one-dimensional metric. Over time as the list of seeders is refined we may only subject ourselves to those who contribute to the smallest clusters.

5 Evaluation

During the same week in April during which we performed seeder analysis, we also analyzed a pool of potential seeders. Potential seeders were gathered by evaluating all clusters that contained at least 100 tweets and that terminated the three days prior to the study. We then ranked all users based on their publication time within the cluster. We first identified all clusters whose size was greater than 100. We identified all non-seeders that placed in the top 5 of these clusters and placed them in a pool of potential seeders. We then let these potential seeders run alongside the existing seeders for a week and scored appropriately. Our search yielded 91 potential seeders, 19 of whom satisfied the promotion criteria outlined in Section 4. Table 4 shows the first seven of the 91 potential seeders while Table 5 shows the bottom 7 of the potential seeders with respect to their RawTiming scores.

Comparing this list of 19 potential seeders to the original pool of 1048 seeders yields interesting results. Figures 2a-d are logarithmic plots of TotalTweets, RawTiming, NumClusters, and AvgTiming as they vary across the ranks of the seeders. Plotting the potential seeders on the same curve as the existing seeders reveals that, with the exception of NumClusters, the scores of the potential seeders lie within



(a) TotalTweets (b) RawTiming

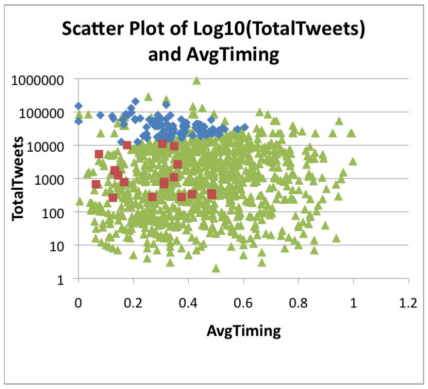
(c) NumClusters (d) AvgTiming

Figure 2: Distributions of (a) TotalTweets, (b) RawTiming, (c) NumClusters and (d) AvgTiming across the seeders and potential seeders. The red lines are the seeders and the blue dots are the potential seeders.

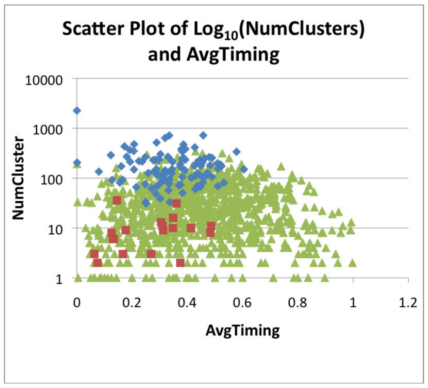
the middle 50% of the seeder population. Thus, according to the TotalTweets plot, these 19 potential seeders contributed to clusters whose cumulative sizes were similar in size as the total tweets in the clusters to which the seeder population contributed. Additionally, according to the AvgTiming plots, we observe that the potential seeders produced timely tweets. That is, the tweets that were produced were early relative to the life of the clusters to which they contributed. The NumClusters plot, however, shows that the 19 potential seeders, while timely and contributing to clusters large in size, are scattered across the bottom half of the original seeder population. The RawTiming plots show that the potential seeders contributed to large clusters in a timely manner consistent with the existing seeders. This low cluster count could be to due to the fact that many of the potential seeders are individuals and not news agencies as many of the existing seeders are.

Observing scatter plots we see a similar distribution of the potential seeders amongst the original seeders. Figures 3a and 3e are scatter plots of TotalTweets vs. AvgTiming and TotalTweets vs. RawTiming. The red icons are the potential seeders that were promoted to seeders, green icons are existing seeders, while blue icons are the top 10% of the seeders as ranked by RawTiming. Recall that lower AvgTiming and higher RawTiming values are desirable (marks of a good seeder) as these measures correspond to how timely a Twitter user contributes tweets to clusters. While the top 10% are concentrated on the upper left of 3a and upper right of 3e (signifying the quick generation of tweets that contribute to many large clusters) the potential seeders are vertically well-distributed within the center of the existing seeder population in Figure 3a, but concentrated above the median of the existing seeder population in Figure 3e.

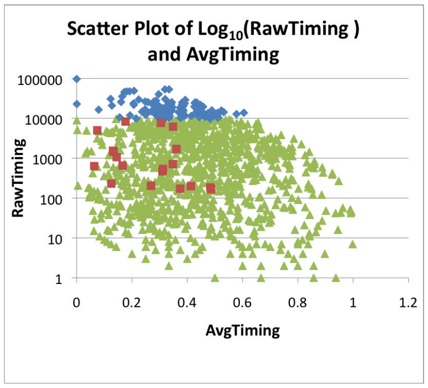
Figures 3b and 3d enable us to visualize a similar result but this time we analyze NumClusters. While the potential seeders are scattered amongst the left half of the existing seeders in 3b and the right half of the existing seeders in 3d, the range of the potential seeders over NumClusters ,in both 3b and 3d, varies across the bottom 60% of the existing seed-



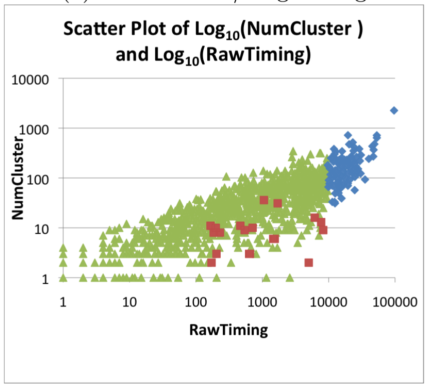
(a) TotalTweets/AvgTiming



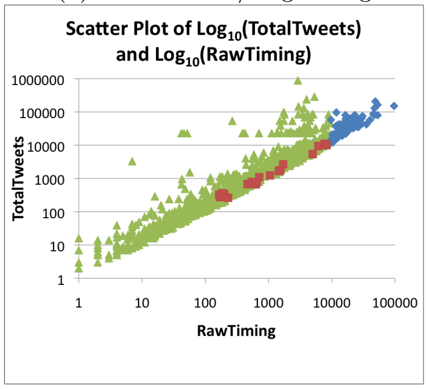
(b) NumClusters/AvgTiming



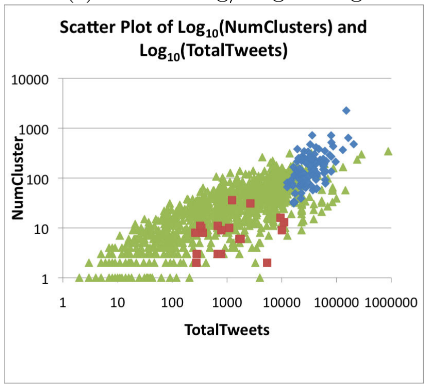
(c) RawTiming/AvgTiming



(d) NumClusters/RawTiming



(e) TotalTweets/RawTiming



(f) NumClusters/TotalTweets

Figure 3: Scatter plots of (a) TotalTweets vs AvgTiming, (b) NumClusters vs AvgTiming, (c) RawTiming vs. AvgTiming, (d) NumClusters vs. RawTiming, (e) TotalTweets vs. RawTiming, (f) NumClusters vs. TotalTweets. The seeders with RawTiming values in the top 10% are blue, the remainder are green. The 19 potential seeders that were promoted are plotted in red.

ers when ranked by NumClusters. This shows that although most of the potential seeders we found contribute to a large number of clusters in a timely fashion, the potential seeders tend to contribute to a smaller number of clusters than half of the existing seeders.

Figure 3c shows the relationship between RawTiming and AvgTiming as well as how the potential and existing seeders are distributed across these two temporal metrics. Here we see how timely the potential seeders are as compared to the existing seeders. Both of the RawTiming and AvgTiming values place the potential seeders in the top third of the existing seeders. Figure 3f shows the relationship between TotalTweets and NumClusters. We see a fairly linear relationship with these values with the potential seeder's NumClusters values again clustering near the bottom of the plot.

Although the potential seeders are not superior to the existing seeders using these statistical measures, in their totality they do prove to be evenly distributed across the middle sections of the existing seeders for most of the statistical measures. This suggests that while our measures might not necessarily find new seeders that are statistical leaders, the newly promoted potential seeders do provide value to the system that is comparable to at least the middle 50 percentile of the existing seeder base.

Applying our seeder identification methods yields mixed results when examining some of the potential seeders in greater detail. There were several promising Twitter users that arose from this initial survey and a few that provide insight into additional evaluation criteria. Manually reviewing the Twitter pages of these potential seeders that our quantitative methods identify for promotion enabled us to perform a qualitative analysis of this seeder identification method.

Among the potential seeders, ID 111824111 (@mkcholakkal) has good scores. The user is in the top 3 in RawTiming, has a high TotalTweets count and low NumClusters, a result from contributing to clusters that are large in size. This user often tweets and links articles about newsworthy events. Additionally this user discusses such events with other users through the Twitter medium. Potential seeders in this category both supply links to articles and provide a 'pulse of the world' with their opinions and thoughts about newsworthy events. They couple the 'pulse of the world' with facts through their tweets.

Viewing the pages of Twitter ID 467026458 (@CBB_FS) finds that this user is an avid sports Tweet'er, was the first to link an article regarding the transfer situation at the University of Wisconsin and did so over 12 hours before any other seeder tweeted about the same topic. Though this was a medium to small cluster with a modest tweet count of 80, it still shows that a fair amount of people were monitoring/discussing the topic. Without additional seeders of this caliber we not only fail to report the news item sooner, but lose accompanying tweets that otherwise would not bind to a cluster.

The seeder identification method outlined in this paper, however, is not without error and requires additional checks when considering potential seeders. @FixingNews (the first potential seeder in Table 4, ID 217842950) scored very well against all other potential seeders. This user was the clear RawTiming leader, has very low AvgTiming values and moderate NumClusters values. Upon visiting the Twitter page of this seeder we found that every single Tweet was a re-tweet from @BBCNews. @FixingNewsRjects (ID 264307945) was the second user in that chart and had similar behavior. Results such as these suggest the need to do a background check on potential seeders. Through Twitter's 'user_timeline' API method, we can obtain the last 3200 tweets for any user. Users that simply re-tweet another user will have an obvious pattern and can be discarded and/or potentially blacklisted from future seeder considerations.

Re-tweeting, however, is not always bad. The timely re-tweeting the content from diverse newsworthy sources would easily increase the metric values of a seeder. Such a seeder could very well be considered a news hub if they re-tweet enough content rather quickly. One should note that when considering re-tweeting as a method for seeder evaluation it should be done in a careful way.

Potential Seeder	**Raw Timing**	**Num Clusters**	**Total Tweets**	**Avg Timing**
217842950	8282	9	10059	0.176658
264307945	7700	13	11076	0.304803
111824111	6170	16	9467	0.348262
331180650	1705	31	2668	0.360945
436114131	1540	6	1773	0.131416
546694115	1460	6	1685	0.133531
467026458	1058	36	1237	0.144705

Table 4: Scores for the top-7 potential seeders after 5 days of monitoring. Users are ranked according to RawTiming. Potential seeders with less than 2 clusters and an AvgTiming value $>$ .5 are omitted.

Potential Seeder	**Raw Timing**	**Num Clusters**	**Total Tweets**	**Avg Timing**
227498479	1	2	10	0.9
548083055	4	2	42	0.904762
216608318	5	2	32	0.84375
187110093	9	1	10	0.1
62421098	10	2	21	0.52381
204046795	11	2	22	0.5
344796941	12	3	149	0.919463

Table 5: Scores for the bottom-7 potential seeders after 5 days of monitoring. Users are ranked according to RawTiming. Note how AvgTiming is, for the most part, very high, indicating that most of these potential seeders tweeted very late in the life of those clusters. One seeder in particular had a very low AvgTiming score, but only contributed to one small cluster. This single, small cluster yields a very small RawTiming score.

6 Future Work

The analysis performed in this paper was performed over a relatively short period of time. A longer-term analysis should be performed in order to gather a better indication of how seeders and potential seeders tweet over an extended period of time. The decay, demotion and promotion methods proposed here might benefit from a refinement of values that a longer-term study might offer. This refinement could focus on ensuring that the promotion/demotion process is stable and not a volatile/cyclical process.

The metrics involving time in our approach only take into account where a tweet lies in the rank of a timeline. They do not take into account the amount of time that has transpired since cluster formation or how much time passed until the cluster becomes inactive. Consider two clusters of size 200 with equal life spans. Each cluster has a 100th tweet that serves as the midpoint tweet. However if the 100th tweet in cluster one appears 10% into its life span and cluster two's 100th tweet appears 50% into that cluster's life span, then the former tweet was much more prompt in its delivery. Scoring algorithms that incorporate this consideration might serve to better score potential seeders.

While the timing of a tweet or the size of a cluster can be used to quantify how good a seeder is at reporting the news, these are not necessarily the only way to judge a seeder. A seeders ability to spread the news as well as the quality of what they discuss could also be used in determining how good a seeder is. Retweet counts could evaluate how well a seeder can spread a news story and not just report it as we were so interested in. We could also look outside of our system or the Twittersphere as well. As many of the tweets contain URL's to stories or blogs, we could also evaluate those documents and try to qualify how good the content is. Calculating the pagerank of any linked document could provide a metric from an external system and be used in the calculation of the quality of a seeder .

The use of RawTiming as the primary key for seeder removal could result in the biased retention of seeders who contribute to clusters of a large size. Seeders who have more timely tweets to smaller, yet still large clusters, could end up getting removed from the system in certain ideal conditions. A modification to RawTiming might resolve this issue. Rather than simply subtracting the absolute rank of the user from the cluster size we could use exponential decay to compute the rank. Given two seeders who are the first to contribute to two clusters of different sizes, this decay would favour the seeder who contributes to the larger cluster. In contrast, this decay would not provide an overwhelming RawTiming score to a seeder who is in the top 10% of a large cluster when compared to a seeder who contributes to a cluster a tenth of the size but is the first to do so.

Through our qualitative analysis of the potential seeders that qualify for a promotion into the seeder pool, we discovered a set of users that, despite meeting the quantitative requirements needed to become a seeder, would not provide any added benefit to the system. Users that simply re-tweet tweets from a dedicated Twitter user will not deliver additional newsworthy tweets to the system. As their tweets are simply re-tweets, they merely echo previously recorded tweets. However, the mere fact that all tweets from any user are only re-tweets is not necessarily an indication that he does not add value to the system. Twitter users who re-tweet tweets from hundreds of users might provide an aggregation of news, opinion or sentiment of a population as a whole. Thus two re-tweet metrics might be used to further quantify seeders. One such metric might track the percentage of tweets that are re-tweets, while another might track the diversity of the original authors of the re-tweeted tweets.

As noted in the related work section, Canini's [7] use of Mechanical Turk workers might be employed as a mechanism to evaluate Twitter users. Perhaps rather than letting the system automatically promote/demote seeders, the system could first identify possible candidates to swap and allow for Turkers to vote on the potential swaps. If nothing else this could server as a mechanism to help validate the algorithm.

The scoring mechanisms we put in place also track RawTiming values for each of the classification of clusters (general, sports, SciTech, business, and health). When evaluating users for promotion into the seeders pool, extra weighting might take place for those users who might not breach the 10% low-water mark mentioned above for their RawScore but do so for one particular category. For example a technical blogger might not crack the overall 10% but provide a niche set of news tweets in SciTech such that he is in the top 5% of all SciTech contributors.

Earlier we discussed the need to recognize the filter bubble defined by Pariser [23] and ensure we take appropriate steps to try and avoid such a phenomenon from happening when providing a service such as TwitterStand. While finding Twitter users outside of the pre-existing social network that feeds the system is a critical first step, there are other ways to continue this work. Hourcade [13] explains the need for systems to bring together people with differing opinions and thoughts. While we proposed a method for identifying and classifying the different topics about which seeders tweet, we fail to have a method to determine the sentiment between any two twitter users. The use of sentiment analysis between tweets could further score potential seeders. By classifying tweets as 'pro', 'con', or 'indifferent' about any topic, we could identify contrasting points of view for any one cluster. Armed with such data, an additional goal of any seeder promotion system might include the ability to promote seeders with differing views, providing an even broader view of what the world is thinking. This too could lead to the formation of an additional cluster in the future.

Lastly, a combination of the last two ideas could greatly extend this work. We could once again find contrasting contributors but now extend this search into the different domains. Now, we will not simply look for users that contribute to various domains but we can work to ensure that we have a balanced amount of contrasting views within any one domain. For example, this might lend the system to look for a fair number of Democrats and Republicans within the realm of business clusters.

7 Conclusion

TwitterStand is an evolving system, tracking the news cycle through tweets and offering a unique method to browse, discover and explore the current news cycle. News topics, or clusters, depend on seeders to form new clusters. In this paper we outlined the need to identify new seeders from the noisy Twitter landscape. We defined metrics that quantify both existing and potential new seeders and algorithms to score and place Twitter users within the system based on those values. A system architecture was outlined that facilitates the discovery and promotion process as well as includes necessary modifications needed to support the process. We performed a short-term implementation of this process, a quantitative analysis comparing the various metrics between the existing and potential seeders as well as a qualitative analysis through manual inspection of potential seeders with moderate success.

ACKNOWLEDGEMENTS. This work was supported in part by the National Science Foundation under Grants IIS-09-48548, IIS-10-18475, IIS-12-19023, and IIS-13-20791. We are also grateful to Jagan Sankaranarayanan for his comments and to Marco Adelfio for his help in getting the manuscript into its final shape.

8 References

[1] Buzztracker, `http://buzztracker.org`.

[2] Trendsmap, `http://trendsmap.com`.

[3] Pear analytics - twitter study. `http://www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf`, Aug. 2009.

[4] A high flying bird: Twitter breaks 140m users, 340m tweets per day. `http://www.redherring.com/internet/a-high-flying-bird-twitter-breaks-140m-users-340m-tweets-per-day/`, 2012.

[5] E. Amitay, N. Har'El, R. Sivan, and A. Soffer. Web-a-Where: geotagging web content. In *SIGIR*, pp. 273–280, 2004.

[6] E. Bakshy, J. M. Hofman, W. A. Mason, and D. J. Watts. Everyone's an influencer: quantifying influence on twitter. In *WSDM*, pp. 65–74, 2011.

[7] K. R. Canini, B. Suh, and P. L. Pirolli. Finding credible information sources in social networks based on content and social structure. In *SocialCom*, pp. 1–8, 2011.

[8] J. Chen, R. Nairn, L. Nelson, M. Bernstein, and E. Chi. Short and tweet: experiments on recommending content from information streams. In *CHI*, pp. 1185–1194, 2010.

[9] R. Ford. Earthquake: Twitter users learned of tremors seconds before feeling them. `http://www.hollywoodreporter.com/news/earthquake-twitter-users-learned-tremors-226481`. Pub. Aug. 2011.

[10] J. Hannon, M. Bennett, and B. Smyth. Recommending twitter users to follow using content and collaborative filtering approaches. In *RecSys*, pp. 199–206, 2010.

[11] G. R. Hjaltason and H. Samet. Speeding up construction of PMR quadtree-based spatial indexes. *VLDBJ*, 11(2):109–137, 2002.

[12] E. G. Hoel and H. Samet. Benchmarking spatial join operations with spatial output. In *VLDB*, pp. 606–618, 1995.

[13] J. P. Hourcade and N. E. Bullock-Rest. HCI for peace: a call for constructive action. In *CHI*, pp. 443–452, 2011.

[14] A. Jackoway, H. Samet, and J. Sankaranarayanan. Identification of live news events using Twitter. In *LBSN*, pp. 25–32, 2011.

[15] J. Leskovec, L. Backstrom, and J. Kleinberg. Meme-tracking and the dynamics of the news cycle. In *KDD*, pp. 497–506, 2009.

[16] M. D. Lieberman and H. Samet. Multifaceted toponym recognition for streaming news. In *SIGIR*, pp. 843–852, 2011.

[17] M. D. Lieberman and H. Samet. Adaptive context features for toponym resolution in streaming news. In *(SIGIR*, pp. 731–740, 2012.

[18] M. D. Lieberman and H. Samet. Supporting rapid processing and interactive map-based exploration of streaming news. In *GIS*, pp. 179–188, 2012.

[19] M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging: Using proximity, sibling, and prominence clues to understand comma groups. In *GIR*, 2010.

[20] M. D. Lieberman, H. Samet, and J. Sankaranarayanan. Geotagging with local lexicons to build indexes for textually-specified spatial data. In *ICDE*, pp. 201–212, 2010.

[21] P. Metaxas, E. Mustafaraj, and D. Gayo-Avello. How (not) to predict elections. In *SocialCom*, pp. 165–171, 2011.

[22] G. Mishne and J. Lin. Twanchor text: a preliminary study of the value of tweets as anchor text. In *SIGIR*, pp. 1159–1160, 2012.

[23] E. Pariser. *The Filter Bubble: What The Internet Is Hiding From You.* Read more. Penguin Books Limited, 2011.

[24] E. Pariser. Ted talks: Eli pariser: Beware online "filter bubbles". `http://www.ted.com/talks/lang/en/`, 2011.

[25] G. Quercini, H. Samet, J. Sankaranarayanan, and M. D. Lieberman. Determining the spatial reader scopes of news sources using local lexicons. In *GIS*, pp. 43–52, 2010.

[26] H. Samet. A quadtree medial axis transform. *CACM*, 26(9):680–693, 1983.

[27] H. Samet. K-nearest neighbor finding using MaxNearestDist. *IEEE TPAMI*, 30(2):243–252, 2008.

[28] H. Samet, M. D. Adelfio, B. C. Fruin, M. D. Lieberman, and B. E. Teitler. Porting a web-based mapping application to a smartphone app. In *GIS*, pp. 525–528, 2011.

[29] H. Samet, H. Alborzi, F. Brabec, C. Esperança, G. R. Hjaltason, F. Morgan, and E. Tanin. Use of the SAND spatial browser for digital government applications. *CACM*, 46(1):63–66, 2003.

[30] H. Samet, A. Rosenfeld, C. A. Shaffer, and R. E. Webber. A geographic information system using quadtrees. *Pattern Recognition*, 17(6):647–656, 1984.

[31] H. Samet and M. Tamminen. Bintrees, CSG trees, and time. *Computer Graphics*, 19(3):121–130, 1985.

[32] H. Samet, B. E. Teitler, M. D. Adelfio, and M. D. Lieberman. Adapting a map query interface for a gesturing touch screen interface. In *WWW (Companion Volume)*, pp. 257–260, 2011.

[33] J. Sankaranarayanan, H. Alborzi, and H. Samet. Efficient query processing on spatial networks. In *GIS*, pp. 200–209, 2005.

[34] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. Twitterstand: news in tweets. In *GIS*, pp. 42–51, 2009.

[35] B. Sun and V. T. Ng. Identifying influential users by their postings in social networks. In *MSM*, pp. 1–8, 2012.

[36] B. Teitler, M. D. Lieberman, D. Panozzo, J. Sankaranarayanan, H. Samet, and J. Sperling. NewsStand: A new view on news. In *GIS*, pp. 144–153, 2008.

[37] J. Vijayan. CIA monitors up to 5 million tweets daily, report says. `http://www.computerworld.com/s/article/9221564/`, Nov. 2011.