ABSTRACT
TwitterStand is a novel way to track the news cycle by allowing people to view and browse the news geographically. The system uses TF-IDF scoring 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 looks to improve on the use of seeders by identifying ways to augment the current list, while routinely eliminating those that do not bring any value to the system. 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, 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 [3]. Despite studies [2] 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 [6] and provide pre- and post-election analysis [10] and predictions. Even the Central Intelligence Agency (CIA) produces a daily report destined for the president [14] on the sentiment and happenings around the world. Yet combing through the vast amount of noise and data provides challenges not only in the ability to handle the large amount of data that is created every second, but more so in the ability to distinguish the news from the noise. Despite these challenges TwitterStand has provided a method to find and track the news cycle.

TwitterStand [13] 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. Locations within the text of tweets and attached articles are identified and displayed on a map. Browsing methods allow users to both see what locations are being discussed as well as listing 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: A seeder feed and a 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 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. 1801 seeders were chosen to seed the creation of clusters and this set 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, such systems should search for additional sources that extend beyond the social network of existing users. Parisar [11] 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 [12] 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 as 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 carry 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 quantify 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

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 [7] 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 the system evaluates and quantifies Twitter members. 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.

Similar to the approach of finding top-ranked Twitter users for a particular topic is http://www.netbadges.com [1] which employs a gaming mechanism to score and award badges for Twitter users for a particular hashtag (#) or topic. The more times a particular user’s tweet includes the hashtag in question, the more points they are given for that hashtag. Leaders with the most points for any hashtag are awarded badges. Badges are meant to provide some level of recognition for their knowledge or involvement for a particular topic. Both Hannon’s [7] system and Netbadges allow individual users to identify top-ranked Twitter users based on a topic, but do not find top-ranked users independent of a topic.

General recommendation systems typically look like the one developed by Chen [5] 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 Twitter actions, the authors were able to recommend content (URLs). The work resulted in a website (now defunct) that made the recommendation for a user. This is an example of a system that finds important topics rather than valuable twitter users.

Perhaps most similar to our approach is the work performed by Canini et al. [4]. 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 trusts 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. 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 online systems discover, evaluate and rank users for the use and benefit of the system.

3. CURRENT IMPLEMENTATION

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 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 [2] 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 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
AvgTiming are done with the aid of the following constructs: a pitcher in baseball. The calculations of RawTiming is similar to calculating the Earned Run Average (ERA) of large clusters and how quickly users tweet about a topic. Score(AvgTiming)

RawTiming gives an indication of how quickly the user tweets about events and how large these clusters are.

We define Coordinated as the number of clusters a user’s tweets contribute to any cluster during that week are omitted in order to yield a more legible graph.

Number of Clusters per Seeder

Figure 1: Distribution of cluster contributions across the seeders for data collected during one week in April 2012. One seeder (@HenryNews, ranked #1 in NumCluster) 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.

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 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 from 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 logs in: 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 \( TWC \) denote the set of tweets in cluster \( C \), where we assume that each tweet belongs to just one cluster \( C \).

2. Let \( \text{Rank}(t, C) \) be the rank of tweet \( t \) within cluster \( C \) in 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 \( NC \) denote the number of tweets that are contained in cluster \( C \). Let \( tu,C \) be user \( u \)’s first tweet that belongs in \( C \). Let \( \text{Score}(u, C) \) be the total number of tweets in cluster \( C \) minus the rank of \( tu,C \), user \( u \)’s first tweet that belongs in \( C \). In other words, \( \text{Score}(u, C) = NC - \text{Rank}(tu,C,C) \).

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

5. Calculate \( \text{AvgTiming}(u) \) 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 \( \text{Rank}(tu,C,C) \) for tweets \( tu,C \) of user \( u \) for each cluster \( C \) and dividing by the sum of the tweets in each cluster \( C \). In other words, \( \text{AvgTiming}(u) = \frac{\sum_{C} \text{Rank}(tu,C,C)}{\sum_{C} NC} \). 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 contributions. In the event a user contributes more than one tweet to a cluster, we restrict scoring 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) who did not contribute to any 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.

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

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Clusters (NumClusters)</td>
<td>The total number of clusters to which Twitter users contribute. 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.</td>
<td>A seeder publishes 5 Tweets and each Tweet is a member of a different cluster. NumClusters = 5</td>
</tr>
<tr>
<td>Number of Tweets (TotalTweets)</td>
<td>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 providing a composite &quot;pulse of the world&quot;. This is summed up over all clusters to which a user contributes in order to show the relative importance of these clusters.</td>
<td>The clusters from the above seeder example are sized 100, 200, 300, 400 and 500 respectively. TotalTweets = 1500</td>
</tr>
<tr>
<td>Tweet Timing (RawTiming and AvgTiming)</td>
<td>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.</td>
<td>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</td>
</tr>
</tbody>
</table>

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 TwitterStand streams 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.
### 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 TotalTweet 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 tweet 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 [9] 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. Thus we will want to score a news cycle if it has persisted for more than a month. This is done 1) to ensure we find potential seeders and 2) to enable scoring of existing seeders.

<table>
<thead>
<tr>
<th>Algorithm 1: Demote $N$ seeders from seeder pool.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N =$ number of seeders to denote.</td>
</tr>
<tr>
<td>$NT_S = TotalTweets$ of all clusters to which user $S$ contributes</td>
</tr>
<tr>
<td>$NTT =$ Average Total Number of Tweets $(TotalTweets$ metric$)$ for seeders in bottom $(4*N)$ $(SeederPoolSize)$ % of seeders</td>
</tr>
<tr>
<td>$CA =$ Average number of clusters in the bottom $(4*N)$ $(SeederPoolSize)$ % of seeders</td>
</tr>
<tr>
<td>$S_i =$ Potential seeder, $i$, that may be removed from the seeder pool.</td>
</tr>
<tr>
<td>$PS_S =$ Set(size $4*N$) of potential seeders ($S_i$) to be removed, ranked by $RawTiming$.</td>
</tr>
<tr>
<td>$SS_1, SS_2, SS_3, SS_4 =$ Above set ($PS$) broken up into 4 equal, continuous parts.</td>
</tr>
<tr>
<td>Resulting sets are the bottom 4 tiers of seeders, based on $RawTiming$.</td>
</tr>
</tbody>
</table>

$$i = 1$$

while $N > | \text{denoted seeders} |$

Demote all $S \in SS_i$ where $\text{NumCluster}_S < CA$

for $k = 1; k = k + 1; k \leq i$

# 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 $S \in SS_k$ where $\text{AvgTiming}_S > 1 - (.25) \cdot (i - k + 1)$

Demote all $S \in SS_k$ where $NT_S < NTT \cdot (.25) \cdot (i - k + 1)$

end for

$i++$

end while

Both of which otherwise might not have happened if the cluster lives forever. 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, NumCluster and TotalTweets scores based on three decay rates, RTDecay (Raw Timing Decay Rate), NCDecay (NumCluster Decay Rate) and TCDecay (TotalTweet Decay Rate). The RTDecay, NCDecay and TCDecay rates will be that of the RawTiming, NumCluster 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 decay, we will simply subtract the value of the decay rate from the corresponding metric for every user. 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 RawTiming and NumCluster throughout the lifetime of the seeder, copies of these scores will need to be kept. However, these copies of the 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 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 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 for a count of over 200 tweets and have AvgTiming < .5, will be promoted to the seeder pool. Users that have
Table 2: This is a listing of the bottom 8 seeders, ranked by RawTiming within TwitterStand. In this example as we wish to demote 2 seeders, PS is equal to 8 as there are 4 sets of 2 seeders each. In our first iteration when i = 1, k is only ever set to 1. Thus we only observe the bottom 2 seeders within this set. Seeder 36496825 ends up being removed in this first iteration as his value of NumCluster is less than CA.

Table 3: Continuing the example from Table 2 we have the same set but we have demoted seeder 36496825. As i is now 2 we will iterate k from 1 to 2. In the first iteration of the for loop when k = 1, we revisit the same two seeders as above but with modified thresholds. 17411453 remains as his values are within the new thresholds. 17411453 remains as his values are within the new thresholds.

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 and highest AvgTiming 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 NumCluster, TotalTweet and AvgTiming scores.

Algorithm 1 shows one possible method of finding the users that satisfy the 'lowest score' across all three 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. The example illustrates that the effect of varying i is to subject the lower-ranked seeders, based on RawTiming, to higher standards each time.
Table 4: Concluding the example from above $k$ is now 2 as we examine the next two highest seeders as ranked by RawTiming. The thresholds for these two seeders are the same values to which we subjected the bottom two seeders during first iteration of the while loop. Here we find that the value of NumClusters for seeder 20138772 is below CA and is thus removed. Seeder 14677769 remains as his scores are within the thresholds. We now end the algorithm as we have removed the necessary number of seeders.

<table>
<thead>
<tr>
<th>Potential Seeder</th>
<th>Raw Timing</th>
<th>NumClusters</th>
<th>TotalTweets</th>
<th>AvgTiming</th>
</tr>
</thead>
<tbody>
<tr>
<td>39589561</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>0.428571</td>
</tr>
<tr>
<td>19165478</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>0.2</td>
</tr>
<tr>
<td>14428018</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>0.33333</td>
</tr>
<tr>
<td>19087067</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>0.428571</td>
</tr>
<tr>
<td>20138772</td>
<td>3</td>
<td>4</td>
<td>14</td>
<td>0.785714</td>
</tr>
<tr>
<td>14677769</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0.4</td>
</tr>
<tr>
<td>36496825</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>0.25</td>
</tr>
<tr>
<td>17411453</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>0.4</td>
</tr>
</tbody>
</table>

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 as indicated in the bottom rows of Table 2. In this example, the value of CA, the average number of clusters in the bottom 8 seeders is 1.75 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 that has a NumClusters value below 1.75, 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 1.75. 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. The first time through the for loop we revisit the same set of seeders we evaluated in the previous iteration of while loop. Seeder 36496825 has already been removed so we only observe seeder 17411453. 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 NumClusters threshold has changed to 3.375 as $NTT \cdot (.25) \cdot (i - k + 1) = 3.375$. 17411453 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 in the very first iteration of the algorithm, namely 1.75 for NumClusters, 1.6875 for TotalTweets and .75 for AvgTiming. Here we find seed 20138772 has a NumClusters value below the threshold and as a result that seeder is demoted. Seeder 14677769 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 find that RawTiming is the best choice. RawTiming 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, not 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 score, but occasionally being the last one to an equally popular cluster will not decrease it.

Contrasting this 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 5 shows the bottom 8 seeders from TwitterStand as ranked by AvgTiming. Note that while several have very high RawTiming
scores, there are a few that produce some tweets that contribute to sizeable clusters. While the RawTiming shows they are not the earliest of users to break news, they are still tweeting about relevant newsworthy items. Observing the TotalTweet 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 produce 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 result in the need to remove valuable seeders, our goal is to remove the least valuable. Using NumClusters to rank seeders for removal we once again only consider one dimension of their value, namely how often they tweet.

TotalTweets could have possibly been used as a primary key, though it too is 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. RawTiming is the only metric that considers timing and cluster size in the evaluation of its score. While it does not quantify how often a seeder tweets, it is the only metric that considers multiple traits we value. Thus we use this as our primary key for demoting seeders while still considering the other metrics before making the final decision.

5. EVALUATION

During the same week in April when 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 terminated the three days prior to the study. We then ranked all users based on the publication time within the cluster. We first identified all clusters whose size was greater than 100. Any non-seeders that placed in the top 5 of these clusters were pulled and placed in a pool of potential seeders. We then let these potential seeders run alongside the existing seeders for a week and score appropriately. Our search yielded 91 potential seeders, 14 of which satisfy the promotion criteria outlined in Section 4. Table 5 shows the first seven of these fourteen potential seeders that will be promoted to seeders. Table 6 shows the 7 potential seeders with the lowest RawTiming scores. This table is presented to allow for a comparison between these two groups.

Applying this identification method yields mixed results. There were several promising Twitter Ids that arose from this initial survey and a few that provide insight into additional evaluation criteria. By manually reviewing the Twitter pages of these potential seeders that our quantitative methods identify for promotion we can 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 NumCluster, 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 5, 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.

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 falls off. 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
<table>
<thead>
<tr>
<th>Potential Seeder</th>
<th>Raw Timing</th>
<th>NumClusters</th>
<th>TotalTweets</th>
<th>AvgTiming</th>
</tr>
</thead>
<tbody>
<tr>
<td>20738970</td>
<td>4</td>
<td>1</td>
<td>39</td>
<td>0.897436</td>
</tr>
<tr>
<td>28138931</td>
<td>48</td>
<td>4</td>
<td>487</td>
<td>0.901437</td>
</tr>
<tr>
<td>16173645</td>
<td>42</td>
<td>3</td>
<td>661</td>
<td>0.93646</td>
</tr>
<tr>
<td>19087287</td>
<td>1</td>
<td>3</td>
<td>16</td>
<td>0.9375</td>
</tr>
<tr>
<td>19940115</td>
<td>16</td>
<td>2</td>
<td>290</td>
<td>0.944828</td>
</tr>
<tr>
<td>28168181</td>
<td>43</td>
<td>8</td>
<td>962</td>
<td>0.955301</td>
</tr>
<tr>
<td>27867231</td>
<td>57</td>
<td>11</td>
<td>1524</td>
<td>0.962598</td>
</tr>
<tr>
<td>34985745</td>
<td>7</td>
<td>2</td>
<td>3294</td>
<td>0.997875</td>
</tr>
</tbody>
</table>

Table 5: Bottom 8 seeders as ranked by AvgTiming. Note how the seeders with the worst scores, at times, have very high TotalTweet scores. A tweet that is grouped with a large cluster can unfairly weight AvgTiming.

<table>
<thead>
<tr>
<th>Potential Seeder</th>
<th>Raw Timing</th>
<th>NumClusters</th>
<th>TotalTweets</th>
<th>AvgTiming</th>
</tr>
</thead>
<tbody>
<tr>
<td>217842950</td>
<td>8282</td>
<td>9</td>
<td>10059</td>
<td>0.176658</td>
</tr>
<tr>
<td>264307945</td>
<td>7700</td>
<td>13</td>
<td>11076</td>
<td>0.304803</td>
</tr>
<tr>
<td>111824111</td>
<td>6170</td>
<td>16</td>
<td>9467</td>
<td>0.348262</td>
</tr>
<tr>
<td>331180650</td>
<td>1705</td>
<td>31</td>
<td>2668</td>
<td>0.360945</td>
</tr>
<tr>
<td>436114131</td>
<td>1540</td>
<td>6</td>
<td>1773</td>
<td>0.131416</td>
</tr>
<tr>
<td>546694115</td>
<td>1460</td>
<td>6</td>
<td>1685</td>
<td>0.13531</td>
</tr>
<tr>
<td>467026458</td>
<td>1058</td>
<td>36</td>
<td>1237</td>
<td>0.144705</td>
</tr>
</tbody>
</table>

Table 6: 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 AvgTime value > .5 are omitted.

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.

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 are.

As noted in the related work section, Canini’s [4] 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 serve 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 tech. blogger might not crack the overall 10% but provide a niche set of news tweets in SciTech such that he is in the top 20% of all SciTech contributors.

Earlier we discussed the need to recognize the filter bubble defined by Pariser [11] 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 [8] 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 as well as a qualitative analysis through manual inspection of potential seeders with moderate success.

8. REFERENCES


