# Duplicate Removal for Overlapping Clusters: A Study Using Social Media Data

**Amit Paul and Animesh Dutta**
Department of Computer Science and Engineering, NIT Durgapur, India, amitpaul06@gmail.com
Department of Computer Science and Engineering, NIT Durgapur, India, animeshnit@gmail.com



## Abstract

The social media is a labyrinth of information which when uncovers, provides a deep insight into the real-world happenings. In this study, we use social media Twitter to create user groups or clusters using the retweet and reply directed links. The main idea behind creating the groups is to figure out a user's best suited place and to generate crisp clusters. Each user forms a group and thus numerous overlapping groups or clusters are created. To get crisp clusters, we present an algorithm for removing duplicates in cluster configurations that feature a significant amount of overlapping. The idea presented in this paper is that we consider numerous overlapping clusters in a cluster set and proceed in a manner where each cluster is compared with a set of users. The user set is created from these clusters. The proposed algorithm deletes all duplicates and is compared to a naive algorithm. Moreover, a modified algorithm is also proposed whereby selected duplicates are kept based on most significant position of the user among all clusters in the configuration. This does not guarantee that all duplicates will be removed. But, as shown in the study a majority of duplicates are removed. Both the proposed and modified algorithm are lot faster than the naive one. This domain was selected because its a domain where we wish to identify unique user communities (clusters) and where large amount of overlap typically exists. After duplicate elimination, we are left with few clusters which are much bigger in size than other clusters in the cluster set.

## Introduction

Social networking sites generate extensive amounts of data. It is generally acknowledged that embedded within this data there is a lot of useful, domain dependent, knowledge (Adedoyin-Olowe, Gaber, and Stahl 2014). The challenge is to identify and extract this knowledge in a manner whereby it can be meaningfully utilized. One popular mechanism for attempting to do this is to use data mining technology (Srivastava 2008; Jensen and Neville 2003; Barbier and Liu 2011). Examples of where data mining technology has been applied to social network data include: content analysis (Naaman, Boase, and Lai 2010; Wu et al. 2011), identification of influencers (Cha et al. 2010; Kiss and Bichler 2008), identification of communities (Lee et al. 2010; Mishra et al. 2007; Zhang and Yu 2015; Duan et al. 2014; Gregory 2008; Whang, Gleich, and Dhillon 2016), determination of the geographic location of users (by message contents) (Cheng, Caverlee, and Lee 2010; Chandra, Khan, and Muhaya 2011) and using user location in profile (Hecht et al. 2011), sentiment analysis and opinion mining (Kouloumpis, Wilson, and Moore 2011), determining who is "following" / "friends with" / "connected to" whom (Brzozowski and Romero 2011; Kwak et al. 2010), trend identification (Gloor et al. 2009), and "hot spot" detection (Li and Wu 2010) (indicating some natural disaster) (Kryvasheyeu et al. 2016).

Generally, there are three ways to analyse Twitter data: the social network analysis, content analysis and context analysis. Many works have been carried out using message content while valuable retweet information is neglected (Bild et al. 2015). In this paper, we are considering retweet and reply directed links to identify user groupings or clusters. A reweet is a forwarded message from a user to his followers. This is of interest because it tells us who is connected to whom, or in Twitter jargon who is "following" whom. Moreover, a user in the Twitter network can retweet any other user's tweet and this shows the topical interest of the user who retweets the tweet of another user. This allows us to group (cluster) users, according to whom they are "following", which in turn is of interest with respect to a variety of socio-economic applications such as recommending followers, recommending feeds for tweeting etc. However, unlike in the case of conventional clustering algorithms, grouping users in this way typically results in numerous overlapping clusters (groups of users). Individual Twitter users typically follow many others, and are typically followed by many others. On an average a Twitter user has 208 followers although the variance is considerable[1]. Since a user may be following numerous other users he may belong to different communities and thus the overlap. Furthermore, Twitter does not require a user to be a follower of someone to retweet their content and thus this also increases the chance of overlapping since a single user can retweet many tweets of other



[1]Twitter statistics and facts (August 2016), http://expandedramblings.com/index.php/.

users and vice-versa.

Overlapping clusters (user groupings) may not always be a bad thing; but for many applications, for example social media user segmentation, we wish to identify "crisp" clusters, clusters that have a unique membership. More generally, overlapping clusters are undesirable in that they "fade" the dissimilarity (distinctiveness) between clusters. The greater the cluster overlap, the more similar the clusters become, and the differentiation between clusters deteriorates. The problem is exacerbated when we have, not two or three overlapping clusters, but many hundreds with varying degrees of overlap (similarity) as in the case of Twitter communities.

To derive "crisp" clusters from a set of clusters where one or more of the clusters overlap it is necessary to remove duplicate members from individual clusters, using some criteria, so that each cluster becomes unique; a process known as *duplicate removal*. In this paper, we have proposed an algorithm to remove all duplicates from clusters. However by doing so we may be loosing important information. Ideally duplicate removal should be conducted in such a way that information is not lost, or at least the loss is minimized. In the case where we have many overlapping clusters there is also a computational overhead involved, thus we wish our duplicate removal to be conducted in such a way that the number of comparisons that need to be made is minimized. In this paper, we thus propose a simple, another algorithm for the effective derivation of crisp clusters from overlapping clusters derived from Twitter data using the medium of Retweets. In doing so, we are placing users in groups that is best suited by hierarchy using the retweet/reply links.

With respect to the work presented in this paper we conceptualize Twitter data in terms of a directed graph where the vertices represent users and the edges retweets or replies from one user to another. Generally, it is assumed that a user "retweets" another user if there is something interesting (topical) in a received tweet. Clusters representing communities can then be generated starting with an individual "target" user, vertex in the graph, and proceeding in a breadth first manner, level-by-level, up to some pre-specified maxim level (distance from start) $l$. At each level the vertices are added to the clustered representing the target user. In this manner a set of clusters, a cluster configuration, can be produced; one cluster for each target user in given set of tweets. However, the resulting set of clusters will feature significant overlap which makes interpretation difficult (as discussed above). Note that clustering users using retweets and replies is different from using Follow links; Follow links are historical in nature, whilst retweet and reply links are current. Hence clusters generated using retweet and reply links tend to be much more current (topical) than clusters generated using Follow links.

## Related Work

Distinguishing overlapping clusters is difficult due to much similarity between the clusters. Our work on overlapping clusters is based on retweet or reply network (Paul, Dutta, and Coenen 2016; Lussier and Chawla 2011) of social media, Twitter. In our case, majority of duplicates are removed to get unique groups or communities where overlapping is minimized. Our problem is for exact duplicate removal. In social media a user has followers and friends. Tweets generally flow from a user to the followers and friends. The social followers graph and other communities using followers and friends are well studied. But the retweet network where there is a directed edge between two users from source to destination, has received not much attention (Bild et al. 2015). Size, noise and dynamism are dominant research issues with social media (Adedoyin-Olowe, Gaber, and Stahl 2014). A user may be present in different social groups or communities, that makes overlapping clusters.

Many works have been carried out to detect community clusters in social media (Lee et al. 2010; Mishra et al. 2007; Zhang and Yu 2015; Duan et al. 2014; Gregory 2008; Whang, Gleich, and Dhillon 2016; Goldberg et al. 2010; Arora et al. 2012; Hou et al. 2015; Dreier et al. 2014; Lancichinetti and Fortunato 2009). Social networking communities are highly overlapped as a node is present in more than one community. The benchmark algorithms to detect communities work better when overlapping is minimized (Lee et al. 2010). In the paper (Zhang and Yu 2015) the authors detect community for emerging networks using a closeness measure "intimacy". In our case, we have cluster nodes through retweet or reply links. After duplicate removal we get some unique cluster communities. Unique in the sense is that it does not follow the complete community definition (Arora et al. 2012) in the social network. In the paper (Duan et al. 2014), author has used correlation analysis to connect to modularity based methods (Shiokawa, Fujiwara, and Onizuka 2013; Clauset, Newman, and Moore 2004) for community detection.

Although there are number of works using seed expansion(Lee et al. 2010; Whang, Gleich, and Dhillon 2016) for detecting overlapping communities but there is no clear understanding which technique is most suitable for a particular domain (Kloumann and Kleinberg 2014) and the performance of community assignment algorithms (Lee et al. 2010). The paper (Lee et al. 2010) introduced a greedy clique expansion algorithm removing near duplicate communities using distinct cliques as seed. In (Conover et al. 2011) the authors have used network of retweets and mention network to find political alignment. Cluster analysis of these networks reveal clear segregation. Our approach, focuses on exact duplicate removal in overlapping clusters to get "crisp" cluster communities by finding a suitable position of a user in the group.

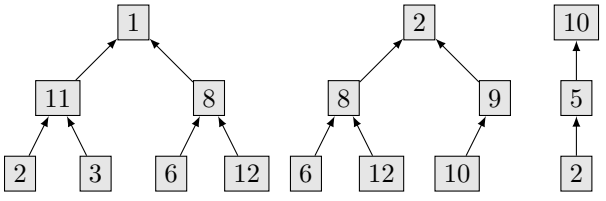
## Scope of The Work

The work presented in this paper is directed at deriving crisp clusters from overlapping clusters by finding a user's best suited position. Some or many clusters are overlapping depending upon the level of hierarchy. Number of overlapping clusters increases as well as the similarity between the clusters, by going up the level. At each level different cluster sizes are chosen by certain threshold. The problem addressed here is removal or elimination of duplicates among overlapping clusters. The first algorithm deletes all duplicate users among the clusters. The algorithm gradually creates a

set of unique users by comparing a user from this set with another user from the clusters and simultaneously removes user from the cluster. The second algorithm is the modification of the first algorithm where selected duplicates with certain criteria are not removed since removing all duplicates will eventually means loss of information. The algorithms are compared with the Naive algorithm with much improved time complexity.

## Problem Formulation

As noted above the overlapping clusters of interest, with respect to the work presented in this paper, are clusters of Twitter users. The clusters are formed using retweet and repliy links between users. The links are traversed in breadth first search manner. The sideway links within same layer or level are not taken. Given a retweet graph $G = \{V, E\}$ where $V$ is the vertex or user node and $E$ is the directional edge. A user $U_i$ is connected to another user $U_j$ if $U_j$ has retweeted or replied to user $U_i$, note that the relationship is unidirectional (as opposed to bidirectional). So, there is an edge $E$ between $U_j$ to $U_i$. Thus, starting from a given user we can place this user and all its immediate neighbours into a single cluster (where a neighbouring users is one connected directly to the current user by a retweet or reply). If two users are "connected" they are in the same cluster. We can then proceed to the immediate neighbours of the seed user plus one, and so on to some predefined maximum "level" $l$. If we assume a set of $m$ Twitter users $U = \{U_1, U_2, U_3, .., U_m\} = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$ and the following set of connections $\{11 \rightarrow 1, 8 \rightarrow 1, 8 \rightarrow 2, 9 \rightarrow 2, 2 \rightarrow 11, 3 \rightarrow 11, 6 \rightarrow 8, 12 \rightarrow 8, 10 \rightarrow 9, 2 \rightarrow 5, 5 \rightarrow 10\}$ (where $U_j \rightarrow U_i$ indicates a Retweet/Reply from user $U_j$ to user $U_i$); then we would get clusters of the form shown in Figure 1 assuming $l = 2$ (the root is at level 0, the "base level"). The figure shows three clusters with respect to users 1, 2 and 10. The clusters in this case are $C_1 = \{1, 2, 3, 6, 8, 11, 12\}$, $C_2 = \{2, 6, 8, 9, 10, 12\}$ and $C_{10} = \{2, 5, 10\}$. A user appears in a cluster only once; there is no duplicity within a cluster. Each user is allowed to form a cluster and is called "root user".




(a) $C_1$ (b) $C_2$ (c) $C_3$

Figure 1: Example clusters for users

From the above simple example we can see a substantial overlap. Note that each user in each cluster is marked with its "level of appearance" (neighbourhood level). Where a user has several level associated with it the level nearest the root is chosen (the closer to the root the more significant a user is deemed to be). Thus, given a real Twitter data set, we can expect extensive overlap. For the purpose of the proposed algorithm all the cluster formation in the cluster set is used. Moreover, experiments are also performed by selecting the largest clusters (in terms of number of members) defined using a threshold $\tau$. For a given maximum level $l$, the $\tau$ value is such adjusted that it gives top $0.25\%$, $0.5\%$, $1.0\%$, $2.0\%$, $4.0\%$ etc clusters of the total number of clusters. The selected value of $\tau$ thus dictates a minimum clusters size below which clusters are not considered for duplicate removal. The clusters are collection of "users" or "members". We will use these words interchangeably.

**Problem**: Given a set of $n$ overlapping clusters $C = \{C_1, C_2, C_3, \ldots, C_n\}$, with maximum level $l$, delete all duplicates in clusters.

In this paper we propose using an "empty bucket" cluster and set of $n$ overlapping clusters. Initially "Empty bucket" is empty. Starting from the first cluster in the set the members are compared to the "empty bucket" which is gradually populated by the members from the clusters. The common or duplicate users are deleted from the clusters and the non-duplicate members are added to the "empty bucket". The "empty bucket" will contain only unique members.

Given $C = \{C_1, C_2, C_3, C_4, \ldots, C_n\}$ and $E = \{\}$ where $C_i$=$\{U_1, U_2, U_3, U_4, \ldots, U_m\}$. If $C_i \cap E = C_s$. The duplicates in $C_s$ are deleted from the cluster. If $C_i$ - $E = C_u$. The uncommon members $C_u$ are added to the "empty bucket". If $C_i \cap E = \phi$. The cluster members are added to the empty bucket.

**Problem**: Given a set of $n$ overlapping clusters $C = \{C_1, C_2, C_3, C_4, \ldots, C_n\}$, with maximum level $l$, delete least significant duplicate users.

Select a level $l$ and $\tau$ to adjust the number of top clusters. Given an empty bucket $E = \{\}$ and $C = \{C_1, C_2, C_3, C_4, \ldots, C_n\}$. The first step is to populate the bucket with most significant user $U_k$. By doing so the algorithm reads all the clusters once. Suppose users in clusters is given by $U_i$ and users in empty bucket is $U_e$. The empty bucket is filled in the following fashion.

1. If $U_i = U_e$ and $U_i$ (level) $< U_e$(level). Replace $U_e$ by $U_i$.
2. If $U_i \neq U_e$. Put $U_i$ in $E$.
3. If $E = \{\}$. Put $U_i$ in $E$

In the second step, the bucket with most significant users are compared with all the clusters once. The duplicates are deleted from the clusters in the following manner.

1. If $U_i = U_e$ and $U_i$ (level) $> U_e$(level) and $U_i$ (level) $\neq 0$. Delete $U_i$ from the cluster.
2. If $U_i = U_e$ and $U_i$ (level) $= U_e$(level). Set $U_e$ (level) = $-1$. To make sure that all the duplicates with this condition is deleted except one.

## The Proposed Algorithm

In this section the proposed duplicate removal algorithm is presented. Recall, with respect to the forging, that using some maximum level $l$ we generate a cluster set $C$ describing social media (Twitter) users. The set $C$ will include one cluster per user and thus feature numerous overlapping clusters. Only those users who have got even a single retweet or

reply message are selected to create cluster. Others are ignored. Each cluster member (user) is associated with a level of appearance. If a user has several levels associated with it the level nearest to root (target user) will be used. We have experimented with all the selected users that form clusters. Yet, we have shown the use of threshold $\tau$. In case we have even larger data, $\tau$ can be used. In that case only the clusters who's size (in terms of number of members) as defined by the threshold $\tau$ are selected. Most of the clusters are overlapping.

The pseudo for the first algorithm is given in Algorithm 1. The input is a set of clusters $C$, generated using some max level $l$ and pruned using the threshold $\tau$. The output is the cluster set $C'$ with all duplicates removed.

**Algorithm 1** Delete User Without Condition

**INPUT:** A Cluster set $C$, generated using max level $l$, and pruned using $\tau$ and an empty bucket set
**OUTPUT:** The Cluster set $C'$ with all duplicates removed

```
1: for each user U_i in cluster C_i do
2:     for each user U_e in Bucket do
3:         if Bucket is Empty then
4:             put U_i in Bucket
5:         else
6:             if U_i == U_e then
7:                 Delete user U_i in Cluster C_i
8:             else
9:                 Put U_i in Bucket
10:            end if
11:        end if
12:    end for
13: end for
14: Return C'
```

The above algorithm given in Algorithm 1 deletes all the duplicates in the clusters by populating an empty bucket and comparing users in clusters with the bucket users. The bucket size is the total number of distinct users in the cluster set. Here, the bucket uses only the user and not the level of its appearance. In this algorithm the initial generated clusters will be bigger in size than the later ones because initially the bucket is empty. Nevertheless all the duplicates are removed from the cluster set but with a cost. The level of appearance of a user is not used and thus the information in the clusters will be less.

The next algorithm is the modification of the above Algorithm 1 which has two parts: These are discussed in further detail in the following two subsections, Sub-sections and .

## Generating Most Significant User Bucket

This sub-section generates a set of most significant user bucket $E'$. A user $U_i$ is more significant if it appears near to the root than the user further away from the root.

The above Algorithm 2 generates a set of users $E'$ which are most significant in nature. $E'$ contains users with its most significant position given by $level$. An user appearing closure to the root is considered more significant than a user

**Algorithm 2** Bucket with Most Significant Users

**INPUT:** A Cluster set $C$, generated using max level $l$, and pruned using $\tau$ and an empty bucket set $E$
**OUTPUT:** A Bucket set $E'$ with most significant users $U_e$

```
1: for each user U_i in cluster C_i do
2:     for each user U_e in Bucket do
3:         if Bucket is Empty then
4:             put U_i in E
5:         else
6:             if U_i == U_e and U_i ⟨level⟩ < U_e ⟨level⟩
   then
7:                 Replace U_e by U_i
8:             else
9:                 put U_i in E
10:            end if
11:        end if
12:    end for
13: end for
14: Return E'
```

appearing further away from the root. This, $E'$ is the set of distinct users with its most significant position. The level of a user is considered for comparing significance. The clusters are traversed only once. Initially the bucket set $E$ is empty. When the algorithm reads the first user in the first cluster, the bucket is populated. After that, one by one all the users in all the clusters are read. The user $U_i$ in clusters is compared to $U_e$ of the bucket set $E$ with their level (position of appearance from the root). The nearest user is the user which is closer to the root. In the algorithm 2 line 6:9 shows the comparison. The user $U_i$ with the less value i.e nearest to the root replaces user $U_e$ from the bucket. The algorithm continues till all the clusters are read.

## Duplicate Removal With Condition

The output from the Algorithm 2 is input for the third algorithm. Each user $U_i$ in the cluster set is compared to the bucket set $E'$ user $U_e$. All the users those are least significant and $level \neq 0$ are deleted from the clusters. If a user is present in both the cluster and the bucket set with same level then the user is kept in at least one cluster. All other duplicates are deleted.

# Evaluation

For the evaluation presented in this section the Geo-tagged Microblog data set[2] available from the ARK data repository held at the University of Washington was used. The dataset holds 377616 Tweets covering all US states and the District of Columbia (Eisenstein et al. 2010). The dataset feature 9477 users. From this data set four cluster sets were generated using a range of values for $l$, the maximum distance from the root, $\{6, 7, 8, 9\}$. The users who did not get any retweet or replied messages from any other user or

[2] http://www.ark.cs.cmu.edu/GeoTwitter.

**Algorithm 3** Duplicate Removal

**INPUT:** A Cluster set $C$, generated using max level $l$, and pruned using $\tau$ and Bucket set $E'$
**OUTPUT:**: The set $C'$ with most duplicates removed

```
1: for each user U_i in cluster C_i do
2:     for each user U_e in Bucket do
3:         if U_i == U_e and U_i ⟨level⟩ > U_e ⟨level⟩ then
4:             Delete U_i from cluster
5:             if U_i == U_e and U_i ⟨level⟩ == U_e ⟨level⟩ then
6:                 Set U_e⟨level⟩ == −1
7:             end if
8:         end if
9:     end for
10: end for
11: Return update C'
```

self within that time period are not considered for clustering.This produced cluster sets comprised of 7123 clusters ($|C| = 7123$) respectively. Since there are total 9477 users and the generation of clusters is around 7123, the remaining 2354 are single users with no retweet or reply messages from anyone or self. 7123 number of clusters also contains single user cluster like users who have retweeted themselves.These are 2262 single user clusters out of 7123 clusters. Further, in 7123 clusters total distinct users are 7576 in number. Thus, 1901 users neither received any retweet or reply message nor they have sent any in the same time period to other users. As $l$ increases the average number of members per cluster in the four different cluster sets also increases, and consequently the clusters become more diverse but feature greater numbers of duplicates.

To analyze the operation of the proposed algorithm we generated cluster sets with different values for $l$=$\{6, 7, 8, 9\}$. Total number of clusters generated is 7123 and the total number of distinct users for all levels is 7576. The results are presented in Table 1, Table 2, Table 3 and Table 4. In the tables the "Num. of Clusters" column indicates the number of clusters retained after application of the $\tau$ threshold value. Here $\tau$ is set to 100% to generate clusters with minimum size of one user. The "Num. of Distinct Users" column gives the number of distinct users in the retained cluster set. This is also the bucket set generated. The following two columns give the number of duplicates before and after the proposed duplicate removal process was applied, and the last column compares the run time of Naive algorithm with proposed and modified algorithm. From, the tables it can be seen that in all cases the proposed algorithm eliminates all the duplicates featured in each of the cluster sets. To highlight the advantages that can be gained using the proposed approach its operation was compared with a naive approach where we compare every cluster in the cluster set $C$ with every other cluster in $C$ and remove all duplicates. Moreover, the modified algorithm retains certain duplicates and its runtime is also compared. Number of duplicates retain in the modified algorithm is shown in the tables. Figure 2 shows the comparison of run time of the proposed and modified algorithm with the naive algorithm.

Adding further to the evaluation process, $\tau$ is used for different levels $\{6, 7, 8, 9\}$. To explore how the threshold $\tau$ effects the process we considered setting $\tau$ to a range of values in terms of the percentage of top clusters to be retained $\{0.25\%, 0.5\%, 0.75\%, 1.0\%, 2.0\%\}$ in the cluster set. Table 5 shows the result of different $\tau$ values for level 9 only. Figure 3 shows the run time of naive, proposed and modified algorithm by setting different $\tau$ values. In level 9 there are total 7123 clusters. $\tau$ is set such that we get certain percentage of top clusters. Thus in the column "Number of Clusters", is the top clusters, each with minimum size greater than the numbers mentioned in the column "Minimum Size of Clusters".

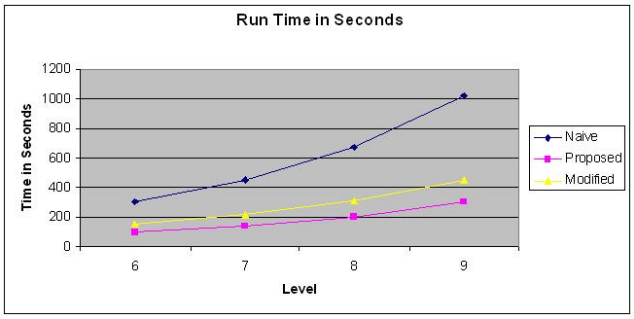



Figure 2: Comparison of runtime of naive algorithm with proposed and modified algorithm for different levels= $6, 7, 8, 9$. $\tau$ is set to 100%

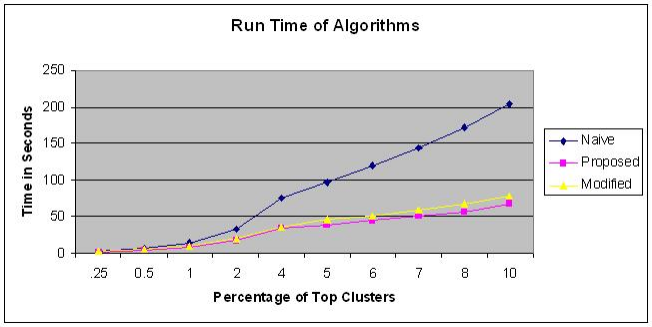



Figure 3: Comparison of runtime of naive algorithm with proposed and modified algorithm at level 9 with different $\tau$ values

## Analysis and Observation

The challenge of duplicate removal in large cluster configurations that feature a significant amount of overlap, as in the case of user communities extracted from social media networks (such as Twitter), is the resource required to remove all duplicates. Furthermore, it becomes more complicated if selected duplicates are to be retained due to its properties. In the proposed algorithm a cluster set is read only once and

| Algorithms | Num. of Clusters | Num of Distinct Users | Num. Duplicates Before Elimination | Num. Duplicates After Elimination | Percentage of Duplicates Retained | Run Time in Seconds |
|---|---|---|---|---|---|---|
| Naive | 7123 | 7576 | 165185 | 0 | 0.0 | 308.593 |
| Proposed | 7123 | 7576 | 165185 | 0 | 0.0 | 97.593 |
| Modified | 7123 | 7576 | 165185 | 8638 | 5.22 | 150.766 |

Table 1: Runtime compared between Naive, Proposed and Modified Algorithms where $l = 6$ and minimum size of each cluster is 1, $\tau = 100\%$

| Algorithms | Num. of Clusters | Num of Distinct Users | Num. Duplicates Before Elimination | Num. Duplicates After Elimination | Percentage of Duplicates Retained | Run Time in Seconds |
|---|---|---|---|---|---|---|
| Naive | 7123 | 7576 | 246524 | 0 | 0.0 | 451.328 |
| Proposed | 7123 | 7576 | 246524 | 0 | 0.0 | 139.297 |
| Modified | 7123 | 7576 | 246524 | 9152 | 3.71 | 217.938 |

Table 2: Runtime compared between Naive, Proposed and Modified Algorithms where $l = 7$ and minimum size of each cluster is 1, $\tau = 100\%$

| Algorithms | Num. of Clusters | Num of Distinct Users | Num. Duplicates Before Elimination | Num. Duplicates After Elimination | Percentage of Duplicates Retained | Run Time in Seconds |
|---|---|---|---|---|---|---|
| Naive | 7123 | 7576 | 359383 | 0 | 0.0 | 672.281 |
| Proposed | 7123 | 7576 | 359383 | 0 | 0.0 | 202.313 |
| Modified | 7123 | 7576 | 359383 | 9588 | 2.66 | 310.383 |

Table 3: Runtime compared between Naive, Proposed and Modified Algorithms where $l = 8$ and minimum size of each cluster is 1, $\tau = 100\%$

| Algorithms | Num. of Clusters | Num of Distinct Users | Num. Duplicates Before Elimination | Num. Duplicates After Elimination | Percentage of Duplicates Retained | Run Time in Seconds |
|---|---|---|---|---|---|---|
| Naive | 7123 | 7576 | 510768 | 0 | 0.0 | 1019.141 |
| Proposed | 7123 | 7576 | 510768 | 0 | 0.0 | 302.985 |
| Modified | 7123 | 7576 | 510768 | 9898 | 1.93 | 453.328 |

Table 4: Runtime compared between Naive, Proposed and Modified Algorithms where $l = 9$ and minimum size of each cluster is 1, $\tau = 100\%$

| Number of Clusters | Minimum Size of clusters | $\tau$ | Num of Distinct Users | Number of Duplicates before Elimination | Number of Duplicates After Elimination Naive, Proposed | Number of Duplicates After Elimination (Modified) | RunTime in Seconds Naive | RunTime in Seconds Proposed | RunTime in Seconds Modified |
|---|---|---|---|---|---|---|---|---|---|
| 18 | 750 | 0.25% | 1687 | 12847 | 0,0 | 719 | 2.938 | 2.015 | 2.578 |
| 36 | 700 | 0.5% | 1846 | 25751 | 0,0 | 1359 | 6.594 | 4.484 | 5.172 |
| 71 | 661 | 1.0% | 1900 | 49434 | 0,0 | 1504 | 15.063 | 8.281 | 10.109 |
| 142 | 560 | 2.0% | 2006 | 92265 | 0,0 | 3790 | 33.0 | 17.297 | 20.140 |
| 283 | 445 | 4.0% | 2196 | 162289 | 0,0 | 3496 | 75.563 | 34.000 | 35.828 |
| 355 | 400 | 5.0% | 2263 | 192506 | 0,0 | 3849 | 97.422 | 38.672 | 46.547 |
| 427 | 371 | 6.0% | 2336 | 220116 | 0,0 | 3967 | 119.328 | 45.265 | 49.859 |
| 499 | 342 | 7.0% | 2422 | 245718 | 0,0 | 4186 | 144.797 | 49.875 | 59.390 |
| 571 | 312 | 8.0% | 2472 | 269115 | 0,0 | 4292 | 171.89 | 56.907 | 67.344 |
| 714 | 268 | 10.0% | 2562 | 310420 | 0,0 | 4203 | 203.563 | 67.406 | 78.719 |

Table 5: Runtime compared between Naive, Proposed and Modified Algorithms where $l = 9$ and minimum size of each cluster is set using $\tau$

| Level | Cluster Size Less Than 50 | Cluster Size Between 50 and 100 | Cluster Size Between 100 and 200 | Cluster Size Above 200 |
|---|---|---|---|---|
| 6 | 7108 | 11 | 04 | 0 |
| 7 | 7103 | 13 | 06 | 01 |
| 8 | 7098 | 15 | 07 | 03 |
| 9 | 7094 | 19 | 07 | 03 |

Table 6: Cluster Size After Elimination

all the duplicates are removed. The proposed algorithm does not take care of the appearance of a user or member by level. Since, a cluster is generated for each user starting from the root, it is better to keep the root of the cluster. Moreover, our intuition say that certain duplicates will enrich the clusters. To accomplish this, we modified the proposed algorithm to keep selected duplicates. A user closer to the root user is more similar to it. The modified algorithm reads the cluster set two times. First, the algorithm reads the cluster set to generate a set of users by level and in the second pass, all the clusters are read and compared to the set of users by selection conditions. This set consists of most significant distinct user. Those members fail the conditions are deleted. From Table 1, Table 2, Table 3 and Table 4 it is observed that as the level $l$ increases the number of duplicates also increases but the percentage of duplicates deleted decreases drastically as shown in column "Percentage of Duplicates Retained". For level 6 the percentage of duplicates retained is $5.22\%$ where as for the level 9 the figure is $1.93\%$.

In the Table 5, top $10\%$ of the clusters gives us $33\%$ of total users in the cluster set which is around 7576. Moreover, after duplicate removal majority of clusters left is of size less than 50. For example, at level 6, out of 7123 clusters only 15 clusters are of size more than 50. If we bifurcate further then in this only 4 clusters are there which go beyond 100 in size. This is listed in the Table 6 for other levels. Thus, the root users in the big clusters are those who got retweet and reply messages from maximum other users directly or through chain of other users. We can see these users as the most prominent users in the cluster set. In general, an influencer spreads message to maximum members. But in our case, the root users are those who got maximum retweets or reply messages. Thus "crisp" clusters are generated for a cluster set where overlapping is minimized.

## Conclusion

Here, we demonstrated methods to eliminate duplicates among numerous overlapping clusters formed using retweet and reply message links among users using Twitter data. The retweet and reply network among users are highly overlapping. The overlapping clusters fade dissimilarity. To get some good clusters or dissimilar clusters to lessen the similarity, elimination of duplicates is necessitated. Our methods work much better than the naive algorithm. Moreover, using this method we can selectively delete duplicates much faster. This study also shows the generation of "crisp" clusters and among them a few prominent clusters, that is, the clusters which are much bigger than other clusters in the set after duplicate elimination. The retweet/reply network clusters represent active users in the cluster configuration if we do not consider the clusters with one member only.

In the future, there are few inroads that open up with this study. The bucket set can be converted into knowledge set of individual distinct users which can learn by reading different clusters. Thus, the properties of users will be enhanced and also the clusters. The use of $\tau$ can be used for much bigger data set for close approximation. Another future work of the study is to investigate how physical distance matters between users i.e users who are retweeting the post of another user. Furthermore, a more detailed study of most prominent clusters in the cluster set will open up new avenues.

## Acknowledgement

This research work is partially supported by the Visvesvaraya PhD scheme, Ministry of Electronics and Information Technology, Government of India.

## References

Adedoyin-Olowe, M.; Gaber, M. M.; and Stahl, F. 2014. A survey of data mining techniques for social media analysis. *Journal of Data Mining & Digital Humanities* 2014.

Arora, S.; Ge, R.; Sachdeva, S.; and Schoenebeck, G. 2012. Finding overlapping communities in social networks: Toward a rigorous approach. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, EC '12, 37–54. New York, NY, USA: ACM.

Barbier, G., and Liu, H. 2011. Data mining in social media. In *Social network data analytics*. Springer. 327–352.

Bild, D. R.; Liu, Y.; Dick, R. P.; Mao, Z. M.; and Wallach, D. S. 2015. Aggregate characterization of user behavior in twitter and analysis of the retweet graph. *ACM Trans. Internet Technol.* 15(1):4:1–4:24.

Brzozowski, M. J., and Romero, D. M. 2011. Who should i follow? recommending people in directed social networks. ICWSM.

Cha, M.; Haddadi, H.; Benevenuto, F.; and Gummadi, K. 2010. Measuring user influence in twitter: The million follower fallacy. In *4th International AAAI Conference on Weblogs and Social Media (ICWSM)*.

Chandra, S.; Khan, L.; and Muhaya, F. B. 2011. Estimating twitter user location using social interactions–a content based approach. In *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, 838–843.

Cheng, Z.; Caverlee, J.; and Lee, K. 2010. You are where you tweet: A content-based approach to geo-locating twitter users. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, number 10 in CIKM '10, 759–768. New York, NY, USA: ACM.

Clauset, A.; Newman, M. E.; and Moore, C. 2004. Finding community structure in very large networks. *Physical review E* 70(6):066111.

Conover, M.; Ratkiewicz, J.; Francisco, M.; Goncalves, B.; Menczer, F.; and Flammini, A. 2011. Political polarization on twitter. AAAI.

Dreier, J.; Kuinke, P.; Przybylski, R.; Reidl, F.; Rossmanith, P.; and Sikdar, S. 2014. Overlapping communities in social networks. *CoRR* abs/1412.4973.

Duan, L.; Street, W. N.; Liu, Y.; and Lu, H. 2014. Community detection in graphs through correlation. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, 1376–1385. New York, NY, USA: ACM.

Eisenstein, J.; O'Connor, B.; Smith, N. A.; and Xing, E. P. 2010. A latent variable model for geographic lexical variation. In *Proceedings of the 2010 Conference on EMNLP*, 1277–1287. Association for Computational Linguistics.

Gloor, P. A.; Krauss, J.; Nann, S.; Fischbach, K.; and Schoder, D. 2009. Web science 2.0: Identifying trends through semantic social network analysis. In *2009 International Conference on Computational Science and Engineering*, volume 4, 215–222.

Goldberg, M.; Kelley, S.; Magdon-Ismail, M.; Mertsalov, K.; and Wallace, A. 2010. Finding overlapping communities in social networks. In *2010 IEEE Second International Conference on Social Computing*, 104–113.

Gregory, S. 2008. A fast algorithm to find overlapping communities in networks. In *Machine learning and knowledge discovery in databases*. Springer. 408–423.

Hecht, B.; Hong, L.; Suh, B.; and Chi, E. H. 2011. Tweets from justin bieber's heart: The dynamics of the location field in user profiles. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, 237–246. New York, NY, USA: ACM.

Hou, Y.; Whang, J. J.; Gleich, D. F.; and Dhillon, I. S. 2015. Non-exhaustive, overlapping clustering via low-rank semidefinite programming. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, 427–436. New York, NY, USA: ACM.

Jensen, D., and Neville, J. 2003. Data mining in social networks. 287–302. In Dynamic Social Network Modeling and Analysis: Workshop Summary and Papers.

Kiss, C., and Bichler, M. 2008. Identification of influencers - measuring influence in customer networks. *Decis. Support Syst.* 46(1):233–253.

Kloumann, I. M., and Kleinberg, J. M. 2014. Community membership identification from small seed sets. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, 1366–1375. New York, NY, USA: ACM.

Kouloumpis, E.; Wilson, T.; and Moore, J. 2011. Twitter sentiment analysis: The good the bad and the omg! In *ICWSM*.

Kryvasheyeu, Y.; Chen, H.; Obradovich, N.; Moro, E.; Van Hentenryck, P.; Fowler, J.; and Cebrian, M. 2016. Rapid assessment of disaster damage using social media activity. *Science Advances* 2(3).

Kwak, H.; Lee, C.; Park, H.; and Moon, S. 2010. What is twitter, a social network or a news media? In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, 591–600. New York, NY, USA: ACM.

Lancichinetti, A., and Fortunato, S. 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E* 80(1):016118.

Lee, C.; Reid, F.; McDaid, A.; and Hurley, N. 2010. Detecting highly overlapping community structure by greedy clique expansion. *ArXiv e-prints*.

Li, N., and Wu, D. D. 2010. Using text mining and sentiment analysis for online forums hotspot detection and forecast. *Decis. Support Syst.* 48(2):354–368.

Lussier, J. T., and Chawla, N. V. 2011. Network effects on tweeting. In *Proceedings of the 14th International Conference on Discovery Science*, DS'11, 209–220. Berlin, Heidelberg: Springer-Verlag.

Mishra, N.; Schreiber, R.; Stanton, I.; and Tarjan, R. E. 2007. Clustering social networks. In *Proceedings of the 5th International Conference on Algorithms and Models for the Web-graph*, WAW'07, 56–67. Berlin, Heidelberg: Springer-Verlag.

Naaman, M.; Boase, J.; and Lai, C.-H. 2010. Is it really about me?: Message content in social awareness streams. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, 189–192. New York, NY, USA: ACM.

Paul, A.; Dutta, A.; and Coenen, F. 2016. Cluster of tweet users based on optimal set. In *2016 IEEE Region 10 Conference (TENCON)*, 286–290.

Shiokawa, H.; Fujiwara, Y.; and Onizuka, M. 2013. Fast algorithm for modularity-based graph clustering. In *AAAI*, 1170–1176.

Srivastava, J. 2008. Data mining for social network analysis. In *2008 IEEE International Conference on Intelligence and Security Informatics*, xxxiii–xxxiv.

Whang, J. J.; Gleich, D. F.; and Dhillon, I. S. 2016. Overlapping community detection using neighborhood-inflated seed expansion. *IEEE Transactions on Knowledge and Data Engineering* 28(5):1272–1284.

Wu, S.; Hofman, J. M.; Mason, W. A.; and Watts, D. J. 2011. Who says what to whom on twitter. WWW '11, 705–714. New York, NY, USA: ACM.

Zhang, J., and Yu, P. S. 2015. Community detection for emerging networks. In *Proceedings of the 2015 SIAM International Conference on Data Mining*, 127–135. SIAM.