Final Report

Construction of Twitter Community Expansion Algorithm

1006101825
Yuchen Zeng

Section 0: Summary

Users connect and exchange information on social platforms, and users that have a common interest forms a community. On Twitter, users provide information to a community by posting a tweet, and transferring and sharing information by retweeting others' tweets.

This report aims to find the best community expansion algorithm for Twitter users, by controlling utilities of candidate users. By testing different expansion algorithms, we found that the amount of information shared and transferred by a user is as important as the number of users a user actively interacts with, when expanding a community with the same interest.

Moreover, if we do not have information about similar communities, it is easy to introduce noises, which may cause the shift of topics. The expansion algorithms assume that the initial cluster is a random sample from the community.We use minimums regarding the initial cluster to restrict candidates, so the expansion algorithms may end too early or too soon when the assumption is violated.

Section 1: Introduction

This report focuses on finding an effective way to expand a community from a small group of users with the same interest on Twitter. The following utilities provide us the information about sorting information flow and user connections for a user u:

1. **Friends**
    - The number of users in the community that u follows.
2. **Followers**
    - The number of users following u in the community.
3. **Direct production**
    - The number of retweets made by u's followers in the community.
4. **Direct consumption**
    - The number of retweets u made from u's friends in the community.
5. **Indirect production**
    - The number of retweets made by users that are not following u in the community.
6. **Indirect consumption**

    - The number of retweets u made from users that u does not follow in the community.

We use the first four utilities for new user selection in a community expansion.

There is no rigid standard for a good community. To check whether a community is well defined, we want to assess "keywords" that users usually mention in the same communities, but not in other communities. Also, we need to consider that one user might be interested in multiple different topics. Hence, we want to add a user to only the community he is most active in.

The initial community is a small cluster of users from a user's local neighbourhood that shares a common interest. It can be generated from a user's local neighbourhood, which is the user's friend list. We start a community expansion from an initial community. For a community expansion to be successful, the expanded community should satisfy the following:

1. Users in the expanded community are still interested in the same topic as the original community.
    - We analyze the occurrence of "keywords" and check user interests manually.
2. The expanded community should not be reclustered into two or more clusters with different interests.

Hence, we contributed the following:

- We generated multiple initial clusters from the local neighbourhood of the user "timnitGebru", using a refining algorithm. We cleaned the data such that every user has direct production, direct consumption, followers, and followings in the cluster.
- We constructed several different expansion algorithms by changing the threshold for each utility.
- We evaluate the experimental results based on whether or not the expanded algorithm is well-defined, and how the pattern in the initial cluster might affect the result.
- We choose the most effective expansion algorithm.

The rest of the report is outlined as follows: Section 2 describes the algorithms we are using; Section 3 describes the generation of initial clusters and the data cleaning process; Section 4 presents our experimental results; Section 5 we interpret the results; Section 6 we conclude our result and next steps in our research. Section 7 includes our references and Section 8 is the appendix.

Section 2: Algorithms

Refining algorithm

This algorithm refines a local neighbourhood based on only the similarity between the friend lists of any pair of users. This algorithm calculates the similarity between friend lists using Jaccard similarity. If the Jaccard similarity between two users' friend lists is above a certain threshold, then an edge is created between the two users. We can generate multiple clusters of users with common interests using the refined local neighbourhood.

```
Input: Local Neighbourhood, threshold
Result: Refined Local Neighbourhood of Initial User
//Local neighborhood is represented by a map of users in the local
    neighborhood to each of their friends
Assign dictionary users_dict as the mapping of each user in the
local neighborhood with its friends
//Map each user to each of its friends who have a jaccard
    similarity between the user's friend list and the follower's
    friend list is higher than the threshold calculated
Initialize users_map as an empty dictionary
for user1 in users_dict do
    for user2 in users_dict do
        if user1 ≠ user2 then
            sim ← the jaccard similarity between the user1's friend
                list and the user2's friend list
            if sim ≥ threshold then
                    Append user2 to users_map[user1]
            end
        end
    end
end
return Local Neighbourhood constructed from the users_map
```

Note: Users without any connection with other users are removed from the social graph, because having no connection means they are inactive in all clusters.

Expansion Rules

The structure of community expansion process is the following:

1. Get the initial community we are interested in and other initial communities.
2. Find the potential candidates we want to consider in 1 iteration(less than 200 users)
    a. Select from users followed by users in the current community
    b. Rank them by number of followers they have in the current community
    c. The highest <200 users will be selected as potential candidates
    d. Extra requirement can be introduced(varies between algorithms)
3. *Filter candidates that do not satisfy certain requirements(varies between algorithms).*
4. Add candidates to the community.
5. When there are no candidates to be added to the community, end the expansion.

The expansion algorithm assumes that users in the initial community are random samples from the community of interest. Any user in the community must have direct production/consumption, friends, and followers greater than zero.

For each algorithm, we introduced requirements for candidates' utilities regarding the original community and current community. By adding rules we want to make sure candidates belong to the same community.

Algorithm 1
Select Potential Candidates
• Calculate minimum number of followers in the **current** community that a user from **initial** community has • Candidates cannot have minimum number of followers in the **current** community less than the minimum
Candidates Filtering
• Calculate minimum number of friends in the **current** community that a user from **initial** community has • Candidates cannot have minimum number of friends in the **current** community less than the minimum
• Calculate minimum number of direct production/consumption in the **current** community that a user from **initial** community has • Candidates cannot have minimum number of direct production/consumption in the **current** community less than the minimum
• Candidates cannot have highest direct production in other initial clusters • Rank candidates by direct production • Only add up to top 40 candidates to the community

Algorithm 2
Select Potential Candidates
• Candidates should not be in other **initial** communities • Same as Algorithm 1
Candidates Filtering

• Same as Algorithm 1
• Calculate the minimum number of **users** in the **current** community that a user from the **initial** community has direct production/consumption from.   ○ No bias when a user have a lot of production/consumption related to only few users • Candidates cannot have minimum number of **users** in the **current** community has direct production/consumption from less than the minimum
• Same as Algorithm 1

Algorithm 3
Select Potential Candidates
• Same as Algorithm 2
Candidates Filtering
• Same as Algorithm 1
• Calculate the minimum number of **users** in the **current** community that a user from the **initial** community has direct production/consumption from.   ○ No bias when a user have a lot of production/consumption related to only few users • Calculate minimum number of direct production/consumption in the **current** community that a user from **initial** community has • Candidates cannot have minimum number of **users** in the **current** community has direct production/consumption from less than the minimum • Candidates cannot have minimum number of direct production/consumption in the **current** community less than the minimum
• Same as Algorithm 1

Algorithm 4 does not have rules fixed to the initial cluster

Algorithm 4
Select Potential Candidates
• Calculate minimum number of followers a user in the community has • Candidates cannot have number of followers in the community less than the minimum
Candidates Filtering

- Calculate minimum number of friends a user has in the community
- Candidates cannot have number of friends in the community less than the minimum

- Calculate the minimum number of users in the community that a user in the community has direct production/consumption from.
- Calculate minimum number of direct production/consumption in the community that a user has
- Candidates cannot have minimum number of users in the community has direct production/consumption from less than the minimum
- Candidates cannot have minimum number of direct production/consumption in the community less than the minimum

- Rank candidates by direct production
- Only add up to top 40 candidates to the community

The following is the pseudo-code for Algorithm 1. The pseudo-codes for the rest of the algorithms can be found in Appendix 1.

Algorithm 1

Input:
community: The initial community

Result: The expanded Community

```
original = community
prevlength = 0
while prevlength != length of community do
    prevlength = length of community
    Download friend list of all users in community
    curr_candidate = find_potential_candidate(community, original)
    Try download all users and their friend list in curr_candidate
    Filter curr_candidate to only users that can be downloaded
    curr_candidate = clean_user_by_following(curr_candidate,
         community, original)
    Download tweets of all users in curr_candidate
    curr_candidate = clean_user_by_utility(curr_candidate,
         community, original)
    curr_candidate = rank_filter_by_utility(curr_candidate,
         community, others, 40)
    Add curr_candidate to community
end
return community
```

find_potential_candidate

Input:
community: The expanding community

original: The initial community

Result: A list of users that are followed by some users in *community*.
All users have followings no less than the minimum among users in *original*.
Only take the largest number of users less than the 200(multiple users can have the same number of common followers in *community*.)

```
following = list of friends in community of each user in
    original
x = minimum value in following
Initialize an empty list new_candidate
for candidate_user in candidate do
    count = following of candidate_user in community
    if count >= x do
        Add candidate_user to new_candidate
    end
end
return new_candidate

Initialize an empty dictionary user_map
for user in community do
    user_list = friend list of user
    for candidate in user_list do
        if candidate not in community do
            if candidate in user_map do
                user_map[candidate] += 1
            else
                user_map[candidate] = 1
            end
        end
    end
end

result = switch key and value of user_map
follower = list of followers in community of each user in
    original
warning_size = minimum value in follower
Initialize an empty list candidate
for i in from length of users to -1, decrement by -1 do
    if i in result do
        if length of result[i] + length of candidate <= 200 and
                i >= warning_size do
            Add result[i] to the end of candidate
        else if i < warning_size do
            break
        end
    end
end
return candidate
```

clean_user_by_following

Input:
candidate: A list of user to be added to the community
community: The expanding community
original: The initial community

Result: A list of users that is a subset of curr_candidate. All users have friends no less than the minimum among users in *original*.

```
following = list of friends in community of each user in
    original
x = minimum value in following
Initialize an empty list new_candidate
for candidate_user in candidate do
    count = following of candidate_user in community
    if count >= x do
        Add candidate_user to new_candidate
    end
end
return new_candidate
```

clean_user_by_utility

Input:
candidate: A list of user to be added to the community
community: The expanding community
original: The initial community

Result: A list of users that is a subset of curr_candidate. All users have direct production and consumption no less than the minimum among users in *original*.

```
production = list of direct production in community of each user in
    original
consumption = list of direct consumption in community of each user
    in original
x = minimum value in production
y = minimum value in consumption
Initialize an empty list new_candidate
for candidate_user in candidate do
    count_x = direct production of candidate_user in community
    count_y = direct consumption of candidate_user in community
    if count >= x and count_y >= y do
        Add candidate_user to new_candidate
    end
end
return new_candidate
```

rank_filter_by_utility

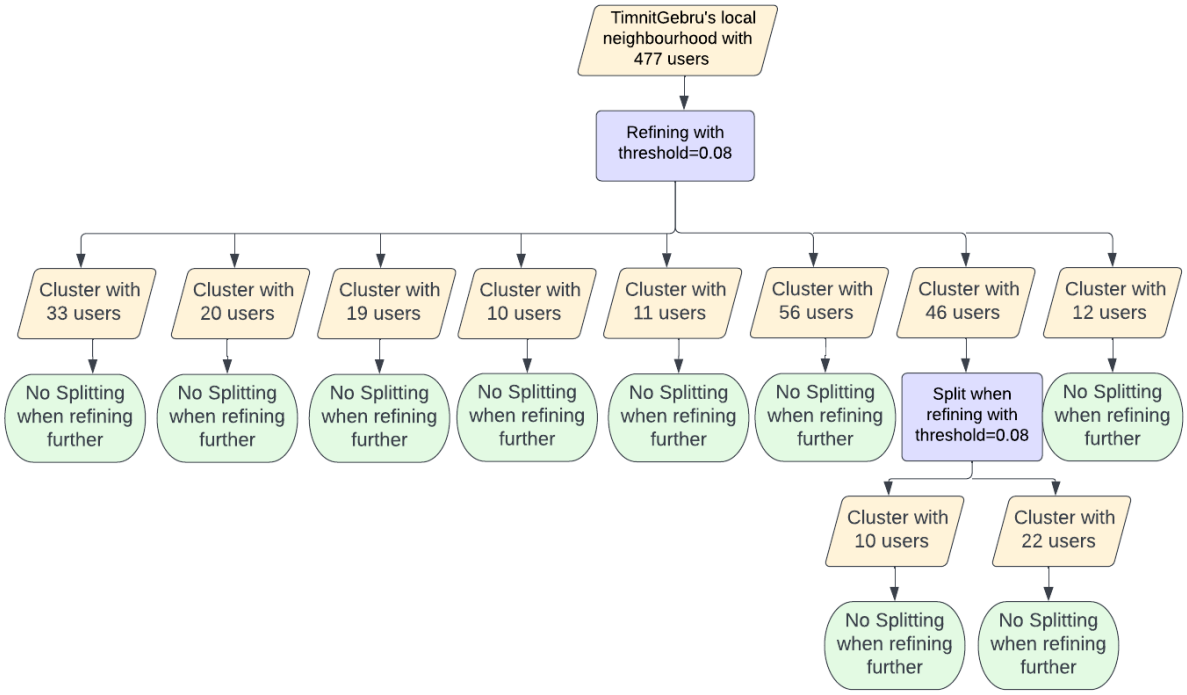
Input:
candidate: A list of user to be added to the community
community: The expanding community
others: other initial community with different topic
size: The maximum size of the filtered candidate list

Result: A list of less than *size* users that is a subset of *candidate*.
All users have the highest production and consumption in *community* compared to *others.*

```
Initialize an empty dictionary prod_map
Initialize an empty dictionary con_map
Initialize an empty list new_user_list
for user in user_list do
Initialize an empty list p_list
Initialize an empty list c_list
    Add direct production of user in community to p_list
    Add direct consumption of user in community to c_list
    for cluster in others do:
        Add direct production of user in cluster to p_list
        Add direct consumption of user in cluster to c_list
    end
    max_p = maximum value in p_list
    max_c = maximum value in c_list
    if max_p == 0 and max_c == 0 do
        prod_map[user] = direct production of user in community
        con_map[user] = direct consumption of user in community
        Add user to new_user_list
    end
    if size > new_user_list do
        return new_user_list
    end
end
Initialize an empty list prod
Initialize an empty list con
result_p = Switch key and value in prod_map
result_c = Switch key and value in con_map
key_p = key in result_p sorted in descending order
key_c = key in result_c sorted in descending order
for i in key_p do
    Add result_p[i] to the end of prod
end
for i in key_c do
    Add result_c[i] to the end of con
end
filtered = users that appears in both the top size of prod and con
return filtered
```

Section 3: Data Cleaning

We started with the local neighbourhood of the user "timnitGebru," who we expect to be a core user in the machine learning community on twitter. We generated initial communities from refining this local neighbourhood, and continuing refining until all clusters cannot split into two or more clusters size > 10. We refined timitGrebru using thresholds from 0.05 to 0.1 increment by 0.01. Finally, we chose threshold=0.08 because it provides the best result where there are 8 clusters with more than 10 users and less than 100 users.



Then we filtered users that have zero direct production, direct consumption, follower or following. The details about users utilities before and after cleaning can be found in Appendix 2.

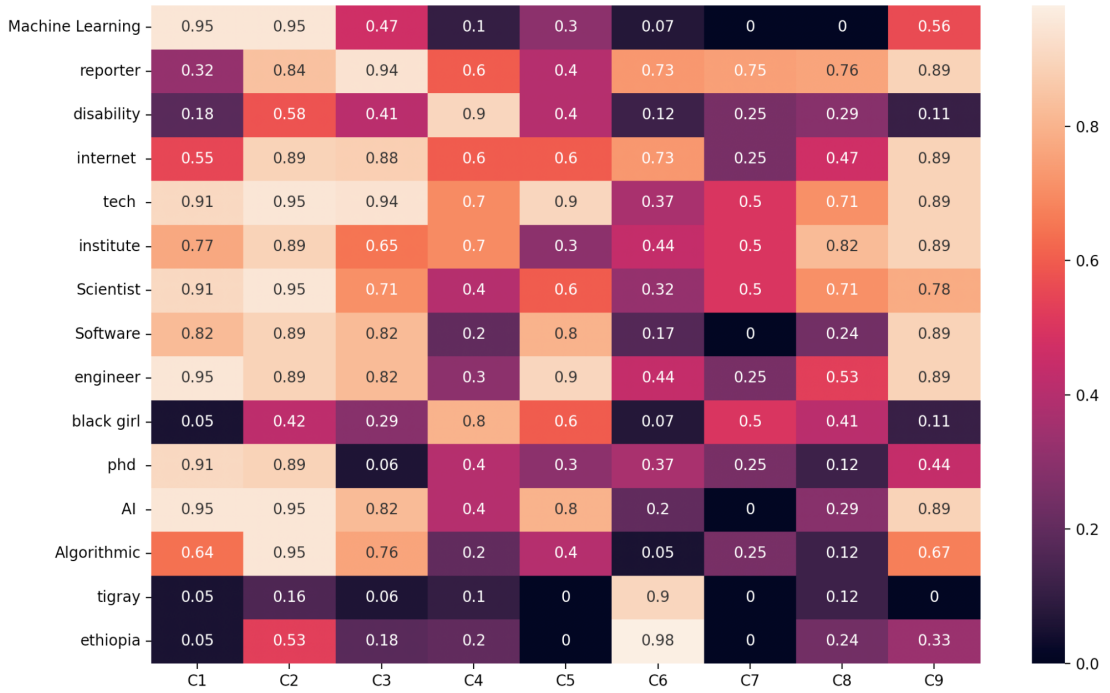
Cluster	Size before	Size after	Is 'timnitGebru' a top user?
1	33	21	Yes
2	20	19	Yes
3	19	17	No
4	10	10	No
5	11	10	No

6	56	38	No
7	10	4	No
8	22	16	No
9	12	9	No

Community interests are assessed by searching users manually. We found that the following keywords might be related to clusters:

"Machine Learning", "reporter", "disability", "internet ", " tech ", "institute", "Scientist", "Software", "engineer", "black girl", " phd ", " AI ", "Algorithmic", "tigray", "ethiopia".

The following is a heatmap showing the percentage of users in a community that have mentioned the keywords.

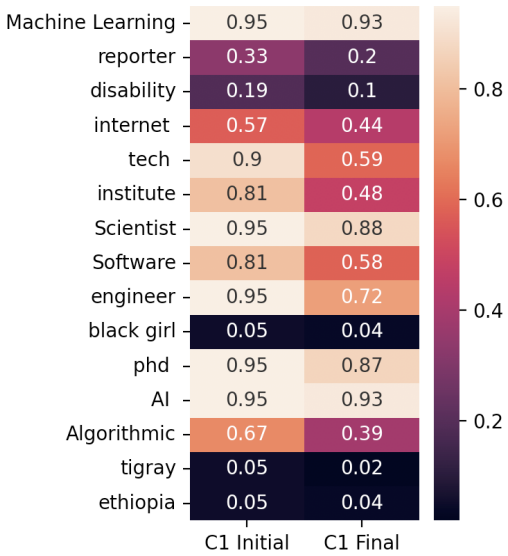
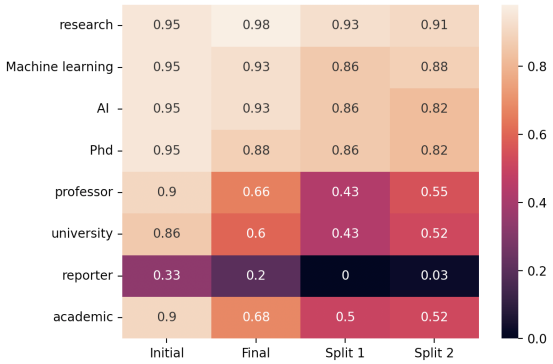
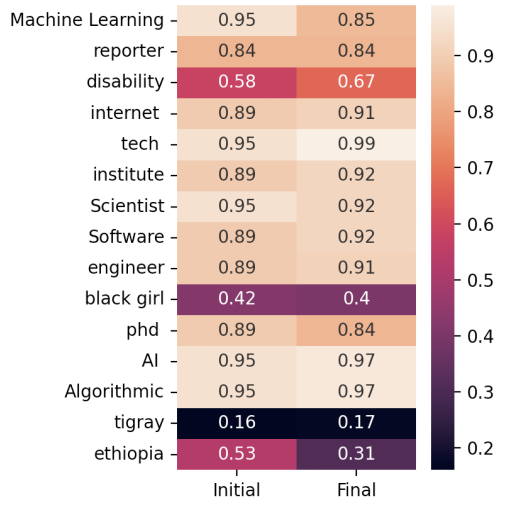


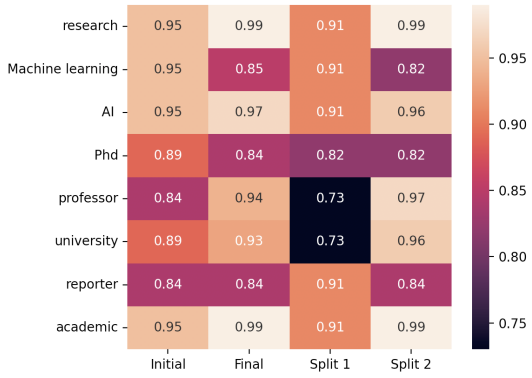
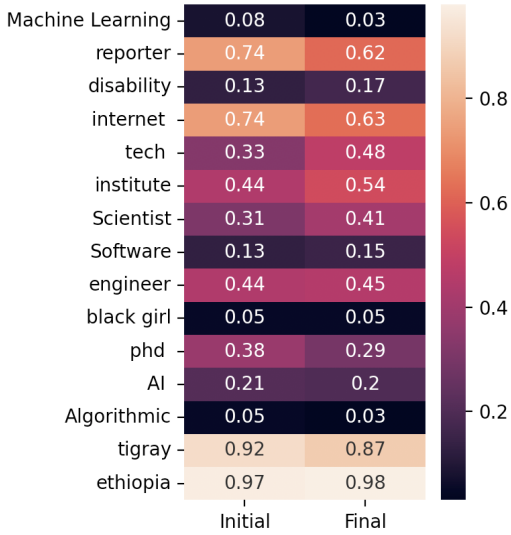
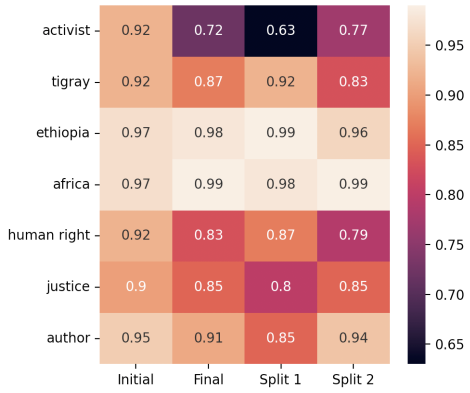
As shown in the graph, some clusters such as cluster 1, 4, 6 have a distinct interest while some clusters such as cluster 2 and cluster 3 are more noisy. Particularly, cluster 1 is about Machine Learning and cluster 2 is about technology reporters that discuss machine learning.

Section 3: Results and Interpretation

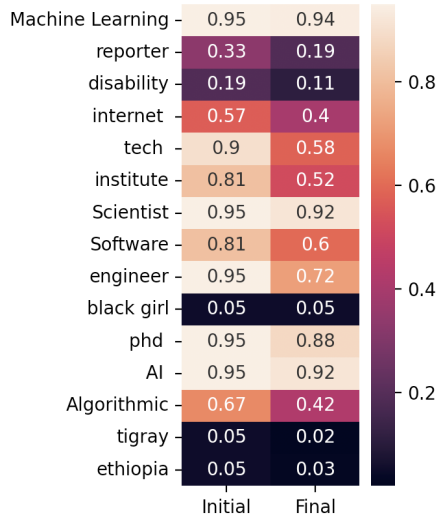
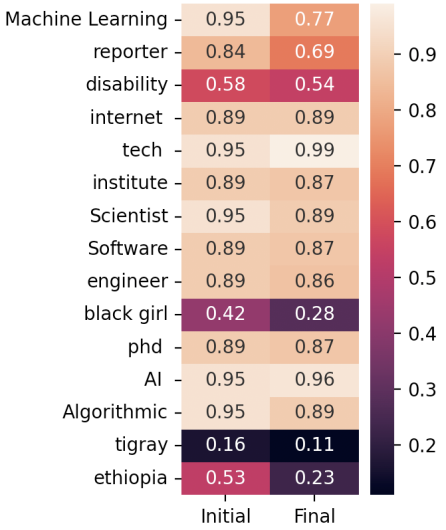
For simplicity, we only expanded cluster 1, 2, 6. The following are expanded communities using different algorithms.

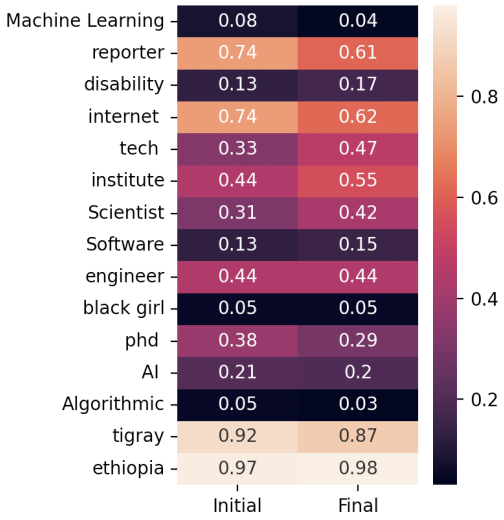
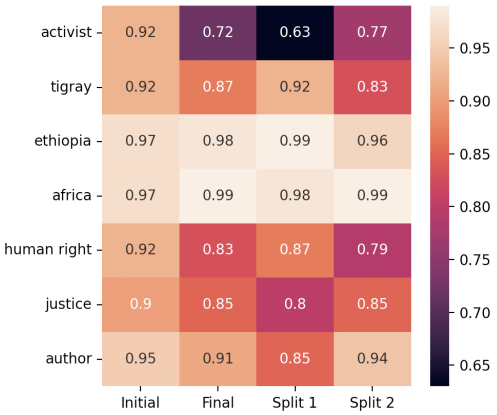
If the algorithm is stopped manually, there is a + behind the size.

<table>
<tr><th colspan="4">Algorithm 1</th></tr>
<tr><th>Cluster</th><th>Size</th><th>Keyword occurrence(% of users)</th><th>Clustering</th></tr>
<tr><td>1</td><td>137</td><td> 
</td><td>Split into 2 when threshold=0.16
● Size 33
○ 5 initial users are in this cluster other than timnitGebru
● Size 14
○ 0 initial user is in this cluster other than timnitGebru
● Both includes top, middle, bottom users
● About the same topic
 
</td></tr>
<tr><td>2</td><td>87</td><td> 
</td><td>Split into 2 when threshold=0.09
● Size 67
○ 12 initial users are in this cluster other than timnitGebru
● Size 11
○ 6 initial users are in this cluster other than timnitGebru
● Both includes top, middle, bottom users</td></tr>
</table>

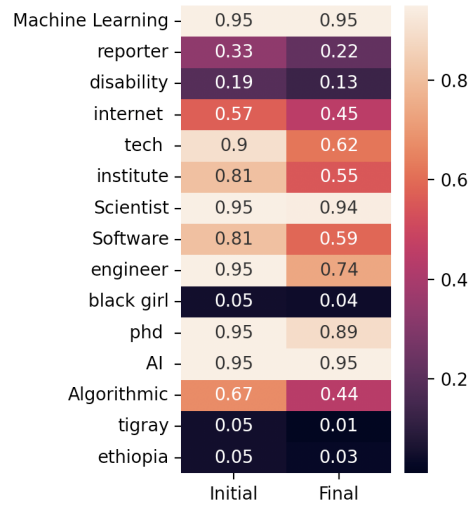
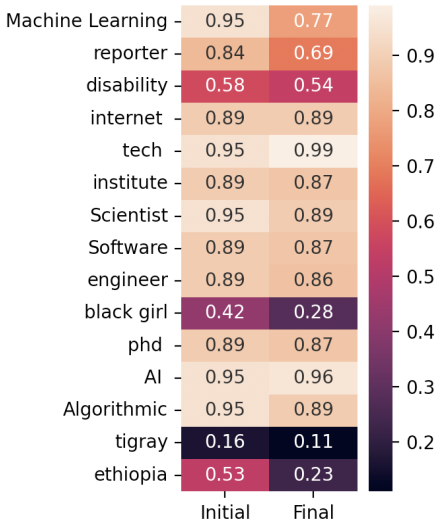
			
6	264		Split when threshold=0.08 ● Size 107 ○ 38 initial users are in this cluster other than timnitGebru ● Size 137 ○ 0 initial users are in this cluster other than timnitGebru 

Algorithm 2			
Cluster	**Size**	**Keyword occurrence(% of users)**	**Clustering**

1	119		Split into 2 when threshold=0.16 • Size 26, 12 • Same structure as result in A1
2	143		Split into 4 when threshold=0.09 • 31, 30, 13 • 0 initial users are in any of the clusters other than timnitGebru

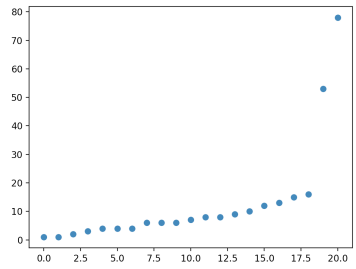
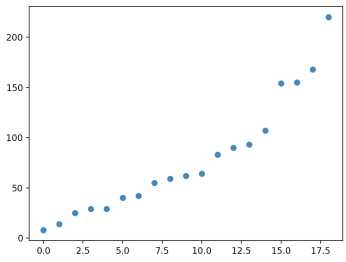
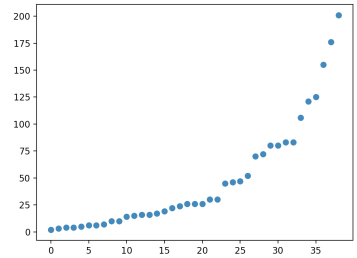
6	266		Split when threshold=0.09 ● Size 125 ○ 1 initial users are in this cluster other than timnitGebru ● Size 12 ○ 6 initial users are in this cluster other than timnitGebru ● Size 89 ○ 32 initial users are in this cluster other than timnitGebru ● timnitGebru does not belongs to any of the subclusters 

Algorithm 3			
Cluster	**Size**	**Keyword occurrence(% of users)**	**Clustering**

1	94		No splitting
2	31		No splitting
There are no common users between cluster 1 and cluster 2(except timnitGebru).			
6	192+	Skipped because we know from previous results that it will likely split into 2 clusters.	

Algorithm 4			
Cluster	**Size**	**Keyword occurrence(% of users)**	**Clustering**
1	Same as in Algorithm 3		
2	Same as in Algorithm 3		
6	188+	Skipped because we know from previous results that it will likely split into 2	

		clusters.

The plot of direct production of initial users in cluster 1, 2, and 6		
Cluster 1	**Cluster 2**	**Cluster 6**
		

According to the keywords, for all algorithms, the expanded communities do not visibly wander off. However, some communities can split into multiple clusters.

Cluster 6

It is worth noticing that Timitgrebru is a top user in clusters 1 and 2, but not in 6. In the results, all initial users are in one of the split clusters, although the two clusters have high similarities in their interests.

Since cluster 6 is the only cluster with a similar topic, we have no information about other communities related to cluster 6. We restricted that candidates cannot have the highest direct production in other initial clusters, but this rule is not effective due to a lack of information about other communities.

Cluster 1 and Cluster 2

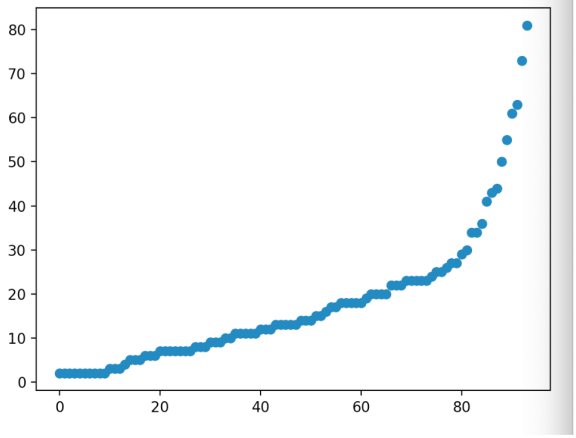
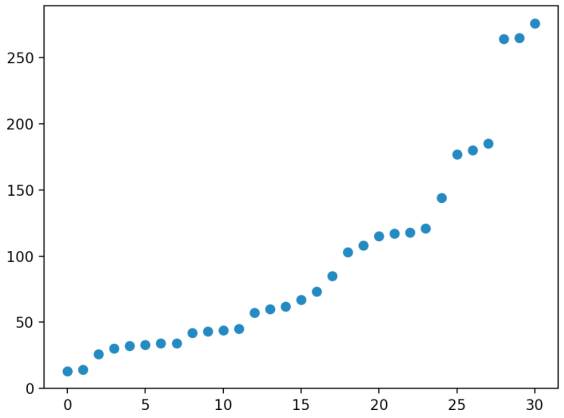
Cluster 1 and cluster 2 do not split when using algorithms 3 and 4. Although there is no difference between their results, algorithm 4 calculates the minimum based on the current community. It means that it always calculates a minimum no larger than the minimum calculated by algorithm 3. The minimum defined from users in the initial community increases quicker in cluster 2 compared to in cluster 1. It is also the global minimum, resulting in a small expanded community for cluster 2. Due to the randomness of how the initial clusters are generated, the utility of the least active user affects the performance of the algorithm.

In the beginning, the minimum production of an original user is 8 in cluster 2, while in cluster 1 it is only 1. The minimum quickly increases to 13 in the second run in cluster 2.

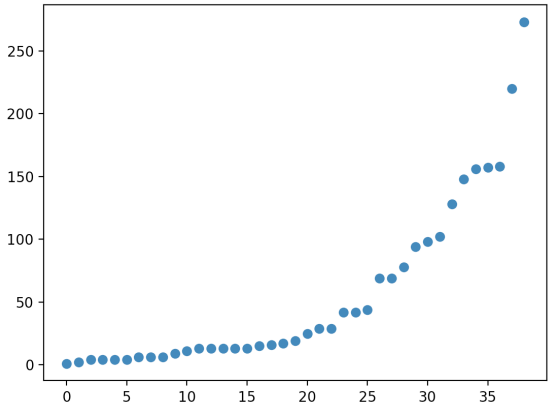
As shown in the plot of direct production, there is no clear drop in production compared to cluster 6, which may indicate cluster 2 does not have enough bottom users.

Also, the only two top users in cluster 1 are Timitgrebru and Hypervisible. Hypervisible has more common friends and higher production in Cluster 2. Timitgrebru has 2 more common friends in cluster 1, but higher production in Cluster 2.

However, the expanded communities from both cluster 1 and cluster 2 using algorithm 3 show the expected pattern of direct production.

Cluster 1	Cluster 2
	

When we add clusters 1 and 2 together, we have the expected pattern of direct productivity.



However, the expanded communities from both cluster 1 and cluster 2 using algorithm 3 show the expected pattern of direct production.

Section 5: Conclusion and Next Steps

In conclusion, algorithm 3 is the best community expansion among the four algorithms.

Select Potential Candidates
• Candidates should not be in other **initial communities** • Calculate the minimum number of followers in the **current** community that a user from the initial community has • Candidates cannot have a minimum number of followers in the **current** community less than the minimum
Candidates Filtering
• Calculate the minimum number of friends in the **current** community that a user from the initial community has • Candidates cannot have the minimum number of friends in the **current** community less than the minimum
• Calculate the minimum number of users in the **current** community that a user from the **initial** community has direct production/consumption from.   ○ No bias when a user has a lot of production/consumption related to only a few users • Calculate the minimum number of direct production/consumption in the **current** community that a user from the **initial** community has • Candidates cannot have the minimum number of users in the **current** community has direct production/consumption from less than the minimum • Candidates cannot have the minimum number of direct production/consumption in the **current** community less than the minimum
• Candidates cannot have highest direct production in **other initial clusters** • Rank candidates by direct production • Only add up to the top 40 candidates to the community

Compared to other algorithms, it conducts not only the number of direct production and consumption but also the spread of them - where production and consumption are from.

Besides the algorithm, the initial clusters can significantly affect the success of community expansion. Without the information about similar communities, it is easy to introduce noises in the community, which may cause the community to wander off or shift to more general topics. It is also important to note the assumption that the initial cluster is a random sample from the community since we use minimums regarding the initial cluster to restrict candidates in the algorithm. When an initial cluster is not nicely distributed, it may result in the expansion algorithm ending too early or too soon.

We can conduct the following experiments in the future:

- The relationship between Cluster 1 and Cluster 2. Are they belonging to the same community? Is there a hierarchy or overlap?
- We can look for a more effective way to generate initial clusters, which can increase the success rate of community expansion.
- We can analyze the difference in the result of community expansion between a cluster before and after data cleaning.
- We can analyze the core user in the community. Particularly, do the core users attract other users to follow them by their retweets(consumption) or tweets(production).

Section 6: Reference

Anonymous Author(s). *Localized Social Search to Detect Community Core Users.*

Section 7: Appendix

Appendix 1: Pseudo Codes for Expansion Algorithms 2, 3

Algorithm 1 and 2 are different in **find_potential_candidate** and **clean_user_by_utility**

find_potential_candidate

Input:
community: The expanding community
original: The initial community
other: other initial communities

Result: A list of users that are followed by some users in *community*.
All users have followings no less than the minimum among users in *original*.
All users must not be in other initial communities.
Only take the largest number of users less than the 200(multiple users can have the same number of common followers in *community*.)

```
# same as Algorithm 1

Initialize an empty dictionary user_map
for user in community do
    user_list = friend list of user
    for candidate in user_list do
        if candidate not in community do
            bool = True
            for cluster in others do
                if candidate in cluster do:
                    bool = False
                end
            end
            if candidate in user_map and bool do
                user_map[candidate] += 1
            else
                user_map[candidate] = 1
            end
        end
    end
end

# same as Algorithm 1
```

clean_user_by_utility

Input:

```
candidate: A list of user to be added to the community
community: The expanding community
original: The initial community
```

```
Result: A list of users that is a subset of curr_candidate.
All users have direct production and consumption no less than the
minimum among users in original.
```

```
production = list of number of users in community retweet and
    follows each user in original
consumption = list of number of users each user in original retweet
and follows in community
x = minimum value in production
y = minimum value in consumption
Initialize an empty list new_candidate
for candidate_user in candidate do
    count_x = number of users in community retweet and follows
        candidate_user in community
    count_y = number of users in community candidate_user retweet
         and follows
    if count >= x and count_y >= y do
        Add candidate_user to new_candidate
    end
end
return new_candidate
```

Algorithm 2 and 3 are different in the **find_potential_candidate** and **clean_user_by_utility**

```
clean_user_by_utility
```

```
Input:
candidate: A list of user to be added to the community
community: The expanding community
```

```
Result: A list of users that is a subset of curr_candidate.
All users have direct production and consumption no less than the
minimum among users in original.
```

```
production = list of direct production in community of each user in
    community
consumption = list of direct consumption in community of each user
    in community

production_dist = list of number of users in community retweet and
    follows each user in original
consumption_dist = list of number of users each user in original
    retweet and follows in community
x = minimum value in production
```

```
y = minimum value in consumption
z = minimum value in production_dist
h = minimum value in consumption_dist
Initialize an empty list new_candidate
for candidate_user in candidate do
    count_x = direct production of candidate_user in community
    count_y = direct consumption of candidate_user in community
    count_z = number of users in community retweet and follows
        candidate_user in community
    count_h = number of users in community candidate_user retweet
         and follows
    if count >= x and count_y >= y and
            count_z >= z and count_h >= h do
        Add candidate_user to new_candidate
    end
end
return new_candidate
```

Algorithm 3 and 4 are different in the **find_potential_candidate**, **clean_user_by_following** and **clean_user_by_utility**

Algorithm 4

Input:
community: The initial community

Result: The expanded Community

```
original = community
prevlength = 0
while prevlength != length of community do
    prevlength = length of community
    Download friend list of all users in community
    curr_candidate = find_potential_candidate(community)
    Try download all users and their friend list in curr_candidate
    Filter curr_candidate to only users that can be downloaded
    curr_candidate = clean_user_by_following(curr_candidate,
         community)
    Download tweets of all users in curr_candidate
    curr_candidate = clean_user_by_utility(curr_candidate,
         community)
    curr_candidate = rank_filter_by_utility(curr_candidate,
         community, others, 40)
    Add curr_candidate to community
end
return community
```

find_potential_candidate

Input:
community: The expanding community

Result: A list of users that are followed by some users in *community*.
All users have followings no less than the minimum among users in *community*.
Only take the largest number of users less than the 200(multiple users can have the same number of common followers in *community*.)

```
# same as Algorithm 1

result = switch key and value of user_map
follower = list of followers in community of each user in
    community
warning_size = minimum value in follower
Initialize an empty list candidate

# same as Algorithm 3
```

clean_user_by_following

Input:
candidate: A list of user to be added to the community
community: The expanding community

Result: A list of users that is a subset of curr_candidate.
All users have friends no less than the minimum among users in *community*.

```
following = list of friends in community of each user in
    community
# same as Algorithm 1
```

clean_user_by_utility

Input:
candidate: A list of user to be added to the community
community: The expanding community

Result: A list of users that is a subset of curr_candidate.
All users have direct production and consumption no less than the minimum among users in *original*.

```
production = list of direct production in community of each user in
    community
consumption = list of direct consumption in community of each user
    in community
# same as Algorithm 1
```

Appendix 2: Data Cleaning for Initial Clusters

The user with id '359831209' is timnitGebru.

Cluster 1	**Size: 33**
Direct Production	[53, 4, 6, 10, 2, 17, 1, 14, 11, 11, 1, 12, 12, 78, 4, 1, 7, 5, 6, 0, 16, 2, 9, 0, 0, 3, 8, 0, 1, 5, 8, 6, 1]
Direct Consumption	[85, 23, 6, 9, 3, 9, 0, 1, 7, 13, 0, 0, 6, 60, 1, 0, 0, 4, 14, 0, 7, 0, 5, 2, 11, 5, 7, 2, 7, 7, 13, 0, 7]
Indirect Production	[0, 1, 0, 0, 0, 2, 0, 0, 2, 0, 1, 0, 0, 13, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0, 0, 2, 0, 0]
Indirect Consumption	[0, 2, 0, 2, 0, 1, 0, 0, 0, 2, 0, 0, 0, 2, 1, 0, 0, 1, 3, 0, 1, 0, 0, 1, 1, 3, 1, 0, 3, 1, 0, 0, 1]
Follower	[31, 28, 28, 27, 26, 13, 3, 27, 21, 20, 16, 20, 26, 1, 25, 20, 22, 19, 17, 13, 25, 26, 22, 26, 19, 15, 28, 9, 19, 26, 16, 24, 23]
Following	[32, 19, 17, 13, 20, 18, 12, 16, 11, 23, 15, 25, 25, 14, 11, 15, 20, 18, 22, 23, 24, 29, 20, 20, 27, 30, 23, 22, 21, 16, 26, 27, 27]
Removed users with id: 6627032 15463062 39547749 15823641 68746721 22674817 15363432 1442906958 60944552 111060272 18850305	
Size: 22	
Direct Production	[53, 4, 6, 7, 2, 15, 13, 9, 10, 12, 78, 4, 4, 6, 16, 8, 3, 8, 1, 4, 6, 1]
Direct Consumption	[85, 23, 6, 8, 3, 9, 1, 7, 11, 6, 57, 0, 4, 4, 5, 3, 3, 7, 2, 7, 12, 7]
Indirect Production	[0, 1, 0, 0, 0, 2, 0, 1, 0, 0, 13, 0, 0, 1, 1, 0, 1, 0, 0, 0, 2, 0]
Indirect Consumption	[0, 2, 0, 2, 0, 0, 0, 0, 2, 0, 2, 1, 0, 3, 1, 0, 3, 1, 3, 1, 0, 1]
Removed users with id: 153196789	
Size: 21	
Direct Production	[53, 4, 6, 7, 2, 15, 13, 9, 10, 12, 78, 4, 4, 6, 16, 8, 3, 8, 1, 4, 6, 1]
Direct Consumption	[85, 23, 6, 8, 3, 9, 1, 7, 11, 6, 57, 0, 4, 4, 5, 3, 3, 7, 2, 7, 12, 7]
Indirect Production	[0, 1, 0, 0, 0, 2, 0, 1, 0, 0, 13, 0, 0, 1, 1, 0, 1, 0, 0, 0, 2, 0]

Indirect Consumption	[0, 2, 0, 2, 0, 0, 0, 0, 2, 0, 2, 1, 0, 3, 1, 0, 3, 1, 3, 1, 0, 1]
Follower	[20, 17, 18, 18, 17, 10, 17, 15, 11, 18, 1, 12, 12, 14, 13, 10, 20, 10, 18, 11, 17]
Following	[20, 14, 14, 9, 14, 14, 9, 10, 17, 16, 8, 13, 16, 13, 15, 19, 15, 14, 12, 18, 19]

Cluster 2	**Size: 20**
Direct Production	[220, 64, 14, 155, 25, 6, 93, 55, 83, 29, 107, 42, 170, 62, 40, 157, 8, 90, 29, 59]
Direct Consumption	[274, 19, 13, 257, 134, 0, 164, 55, 80, 2, 35, 27, 89, 52, 15, 107, 9, 80, 5, 91]
Indirect Production	[0, 0, 0, 1, 0, 0, 4, 0, 0, 2, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0]
Indirect Consumption	[0, 2, 7, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0]
Follower	[19, 16, 8, 17, 19, 16, 17, 17, 18, 16, 17, 16, 17, 19, 16, 18, 15, 17, 17, 18]
Following	[19, 11, 11, 18, 18, 13, 18, 18, 19, 18, 16, 17, 17, 19, 18, 17, 14, 18, 15, 19]
Removed users with id: 1161442118	
Size: 19	
Direct Production	[220, 64, 14, 155, 25, 93, 55, 83, 29, 107, 42, 170, 62, 40, 157, 8, 90, 29, 59]
Direct Consumption	[274, 19, 13, 257, 134, 164, 55, 80, 2, 35, 27, 89, 52, 15, 101, 9, 80, 5, 91]
Indirect Production	[0, 0, 0, 1, 0, 4, 0, 0, 2, 3, 0, 0, 0, 0, 0, 0, 0, 1, 0]
Indirect Consumption	[0, 2, 7, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0]
Follower	[18, 16, 8, 16, 18, 16, 17, 17, 15, 17, 15, 17, 18, 16, 17, 14, 16, 16, 17]
Following	[18, 11, 11, 17, 17, 17, 17, 18, 17, 16, 16, 16, 18, 17, 16, 13, 17, 14, 18]

Cluster 3	**Size: 19**
Direct Production	[20, 261, 83, 6, 0, 34, 0, 42, 44, 13, 84, 20, 8, 96, 35, 17, 12, 29, 8]
Direct Consumption	[30, 83, 24, 153, 11, 3, 0, 17, 272, 23, 6, 11, 29, 1, 24, 1, 45, 60, 19]
Indirect Production	[0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 9, 0, 0, 0, 1, 0, 0]
Indirect Consumption	[0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
Follower	[18, 16, 17, 15, 9, 17, 4, 12, 7, 15, 17, 14, 14, 17, 15, 11, 14, 12, 15]
Following	[18, 17, 15, 14, 9, 13, 5, 12, 14, 15, 17, 14, 13, 18, 16, 10, 14, 8, 17]
Removed users with id: 383466257 9500242	
Size: 17	
Direct Production	[20, 261, 83, 6, 34, 42, 44, 13, 81, 20, 8, 94, 35, 17, 6, 29, 8]
Direct Consumption	[30, 83, 24, 153, 3, 17, 272, 23, 6, 11, 29, 1, 24, 1, 45, 60, 19]
Indirect Production	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 9, 0, 0, 0, 1, 0, 0]
Indirect Consumption	[0, 0, 0, 10, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
Follower	[16, 15, 15, 15, 16, 12, 7, 15, 16, 14, 14, 15, 14, 11, 12, 12, 14]
Following	[16, 16, 14, 14, 13, 12, 13, 15, 16, 14, 13, 16, 15, 10, 12, 8, 16]

Cluster 4	**Size: 10**
Direct Production	[2, 48, 6, 41, 14, 45, 21, 351, 2, 37]
Direct Consumption	[4, 7, 31, 3, 24, 2, 7, 133, 19, 337]
Indirect Production	[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
Indirect Consumption	[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
Follower	[2, 8, 9, 9, 8, 8, 3, 9, 7, 9]
Following	[9, 7, 8, 6, 7, 6, 7, 8, 7, 7]

Removed None

Cluster 5	**Size: 11**
Direct Production	[13, 21, 7, 3, 32, 41, 6, 30, 0, 2, 16]
Direct Consumption	[10, 34, 9, 29, 13, 18, 17, 5, 16, 19, 1]
Indirect Production	[0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0]
Indirect Consumption	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2]
Follower	[9, 8, 9, 10, 9, 9, 9, 9, 5, 9, 8]
Following	[10, 7, 9, 10, 10, 10, 9, 10, 7, 8, 4]
Removed users with id: 9439062	
Size: 10	
Direct Production	[6, 21, 7, 3, 26, 41, 6, 27, 2, 16]
Direct Consumption	[6, 21, 7, 3, 26, 41, 6, 27, 2, 16]
Indirect Production	[0, 0, 0, 0, 2, 0, 0, 0, 0, 0]
Indirect Consumption	[0, 0, 0, 0, 0, 0, 0, 0, 0, 2]
Follower	[8, 8, 8, 9, 8, 8, 9, 8, 8, 8]
Following	[9, 7, 9, 9, 9, 9, 9, 9, 8, 4]

Cluster 6	**Size: 56**
Direct Production	[71, 0, 22, 4, 10, 2, 0, 9, 0, 14, 0, 56, 5, 98, 24, 3, 20, 5, 16, 170, 46, 0, 20, 127, 20, 214, 9, 0, 5, 0, 0, 2, 106, 0, 35, 2, 1, 27, 18, 72, 34, 87, 6, 6, 10, 54, 95, 82, 90, 3, 27, 53, 0, 130, 28, 195]
Direct Consumption	[41, 3, 3, 3, 38, 5, 0, 19, 5, 27, 151, 209, 0, 19, 45, 0, 37, 3, 12, 29, 125, 9, 5, 24, 10, 16, 179, 0, 17, 0, 1, 1, 68, 5, 1, 1, 0, 32, 78, 27, 7, 15, 0, 17, 326, 22, 197, 0, 151, 20, 40, 7, 0, 68, 11, 4]
Indirect Production	[4, 0, 3, 0, 1, 0, 0, 4, 0, 0, 6, 8, 0, 1, 0, 0, 42, 0, 0, 1, 10, 0, 9, 5, 2, 11, 28, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 2, 0, 0, 0, 0, 2, 25, 4, 12, 1, 0,

	6, 12, 0, 5, 0, 1]
Indirect Consumption	[0, 0, 6, 0, 0, 0, 0, 1, 0, 1, 5, 16, 0, 0, 3, 0, 1, 0, 0, 20, 0, 3, 1, 0, 0, 1, 14, 0, 4, 0, 0, 0, 0, 0, 0, 0, 0, 2, 5, 1, 0, 3, 0, 2, 65, 12, 33, 0, 5, 2, 1, 0, 0, 0, 0, 0]
Follower	[37, 3, 22, 21, 24, 22, 34, 11, 33, 21, 6, 16, 28, 43, 5, 24, 7, 29, 29, 20, 17, 18, 25, 32, 35, 51, 4, 35, 21, 11, 8, 28, 15, 27, 23, 6, 18, 28, 33, 36, 32, 46, 26, 28, 21, 27, 21, 31, 27, 38, 18, 38, 35, 34, 39, 52]
Following	[55, 18, 7, 26, 37, 11, 13, 16, 28, 21, 43, 19, 27, 29, 27, 17, 27, 21, 25, 15, 27, 21, 24, 27, 36, 42, 21, 22, 31, 10, 17, 34, 38, 26, 16, 10, 17, 25, 20, 34, 21, 31, 29, 27, 32, 28, 39, 21, 25, 27, 24, 24, 20, 23, 28, 40]
Removed users with id: 1037042773751296001 1531038577 200262214 817501429615259648 88017251 60690881 46122743 54189456 34098602 859784085383905282 244428162 254043964 592707731 278882092 31328392	
Size: 41	
Direct Production	[70, 22, 4, 10, 0, 7, 14, 47, 83, 24, 15, 5, 16, 155, 46, 17, 121, 19, 201, 6, 4, 2, 106, 30, 0, 27, 16, 72, 30, 80, 6, 10, 52, 83, 84, 3, 26, 45, 125, 26, 177]
Direct Consumption	[41, 3, 3, 38, 4, 18, 27, 201, 19, 44, 37, 3, 12, 29, 110, 5, 24, 10, 16, 169, 17, 1, 67, 1, 1, 31, 67, 23, 7, 15, 17, 322, 22, 194, 151, 20, 40, 6, 57, 10, 4]
Indirect Production	[4, 0, 0, 1, 0, 4, 0, 8, 1, 0, 39, 0, 0, 1, 10, 9, 5, 2, 11, 28, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 2, 25, 4, 1, 0, 6, 12, 5, 0, 1]
Indirect Consumption	[0, 6, 0, 0, 0, 1, 1, 16, 0, 3, 1, 0, 0, 20, 0, 0, 0, 0, 1, 14, 4, 0, 0, 0, 0, 2, 5, 1, 0, 0, 0, 62, 5, 30, 5, 2, 1, 0, 0, 0, 0]
Removed users with id: 3117868558 823788375513538560	
Size: 38	
Direct Production	[70, 22, 4, 10, 7, 14, 47, 80, 24, 15, 5, 16, 155, 46, 17, 121, 19, 201, 6, 4, 2, 106, 30, 26, 16, 72, 30, 80, 6, 10, 52, 83, 84, 3, 26, 45, 125, 26, 176]
Direct Consumption	[41, 3, 3, 38, 18, 27, 201, 19, 44, 37, 3, 12, 29, 110, 5, 24, 10, 16,

	169, 17, 1, 67, 1, 31, 67, 23, 7, 15, 17, 322, 22, 194, 151, 20, 40, 6, 57, 10, 4]
Indirect Production	[4, 0, 0, 1, 4, 0, 8, 1, 0, 39, 0, 0, 1, 10, 9, 5, 2, 11, 28, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 25, 4, 1, 0, 6, 12, 5, 0, 1]
Indirect Consumption	[0, 6, 0, 0, 1, 1, 16, 0, 3, 1, 0, 0, 20, 0, 0, 0, 0, 1, 14, 4, 0, 0, 0, 2, 5, 1, 0, 0, 0, 62, 5, 30, 5, 2, 1, 0, 0, 0, 0]
Follower	[27, 19, 14, 21, 8, 16, 13, 32, 4, 5, 21, 19, 13, 15, 20, 27, 29, 36, 3, 17, 22, 11, 18, 20, 24, 31, 28, 35, 19, 19, 23, 19, 17, 26, 13, 29, 25, 26, 36]
Following	[38, 4, 19, 26, 12, 16, 15, 24, 23, 23, 15, 17, 11, 22, 17, 23, 24, 33, 17, 25, 23, 28, 13, 19, 15, 26, 18, 25, 20, 26, 23, 28, 17, 19, 16, 18, 16, 18, 28]

Cluster 7	**Size: 10**
Direct Production	[0, 7, 3, 23, 0, 0, 0, 19, 3, 0]
Direct Consumption	[9, 20, 14, 6, 0, 1, 1, 4, 0, 0]
Indirect Production	[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
Indirect Consumption	[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
Follower	[3, 8, 7, 7, 5, 1, 2, 8, 9, 6]
Following	[9, 5, 6, 6, 5, 5, 6, 6, 5, 3]
Removed users with id: 359831209 427505253 18932031 171753658 16863340 256881576	
Size: 4	
Direct Production	[5, 3, 20, 13]
Direct Consumption	[17, 14, 6, 4]
Indirect Production	[0, 0, 0, 0]
Indirect Consumption	[0, 0, 0, 0]
Follower	[3, 3, 3, 3]

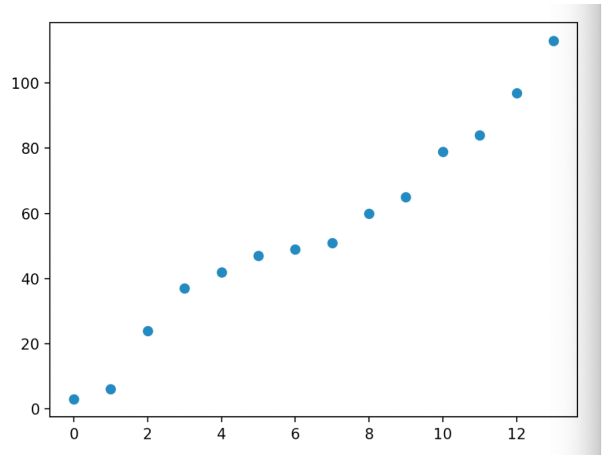
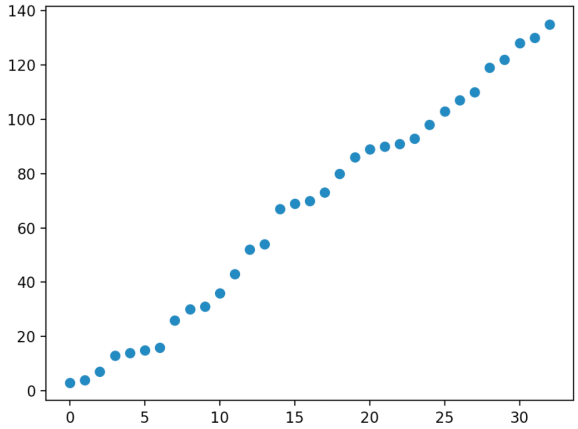
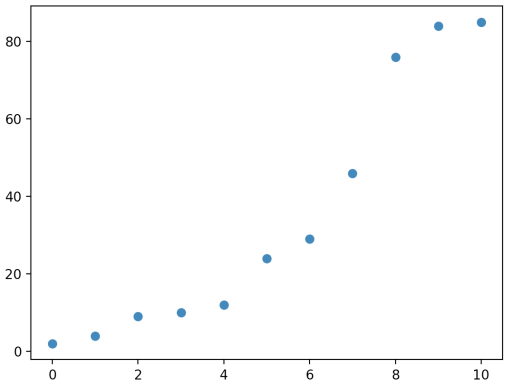
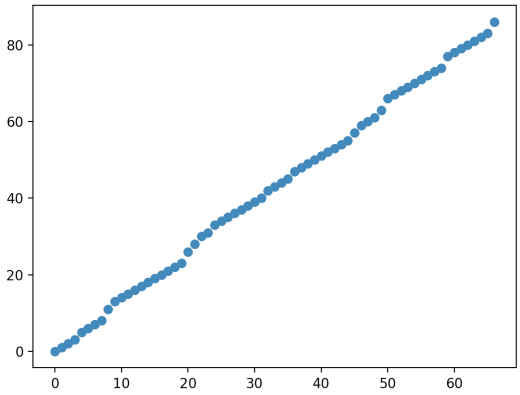
Following	[3, 3, 3, 3]

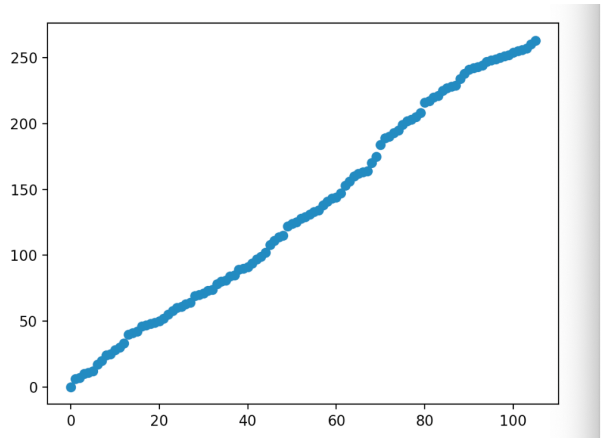
Cluster 8	**Size: 22**
Direct Production	[2, 1, 92, 4, 0, 5, 0, 6, 90, 15, 43, 76, 2, 1, 216, 41, 1, 1, 20, 9, 0, 7]
Direct Consumption	[33, 0, 232, 66, 0, 0, 7, 6, 29, 18, 1, 50, 19, 0, 10, 48, 51, 3, 15, 2, 1, 41]
Indirect Production	[0, 0, 0, 2, 0, 0, 0, 0, 1, 16, 0, 0, 0, 0, 0, 0, 0, 14, 0, 0, 0]
Indirect Consumption	[0, 0, 1, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 24, 5, 0, 0, 1, 0, 0]
Follower	[8, 6, 16, 15, 4, 18, 0, 8, 16, 7, 14, 18, 4, 4, 11, 15, 14, 18, 5, 5, 3, 14]
Following	[20, 8, 9, 15, 3, 2, 10, 12, 13, 13, 5, 15, 10, 7, 9, 9, 9, 10, 12, 11, 8, 13]
Removed users with id: 1713118033 63112528 36686040 965004924370079744 816298918468259841 190705200	
Size: 16	
Direct Production	[2, 1, 92, 4, 5, 90, 15, 42, 73, 2, 216, 41, 1, 1, 17, 9, 7]
Direct Consumption	[33, 0, 232, 66, 6, 29, 15, 1, 48, 19, 10, 48, 51, 3, 15, 1, 41]
Indirect Production	[0, 0, 0, 2, 0, 1, 16, 0, 0, 0, 0, 0, 0, 0, 14, 0, 0]
Indirect Consumption	[0, 0, 1, 0, 0, 2, 0, 0, 0, 0, 0, 24, 5, 0, 0, 1, 0]
Follower	[7, 6, 14, 13, 6, 14, 6, 11, 16, 3, 10, 13, 12, 15, 4, 4, 13]
Following	[16, 7, 9, 13, 11, 12, 11, 5, 13, 9, 7, 7, 7, 9, 11, 9, 11]

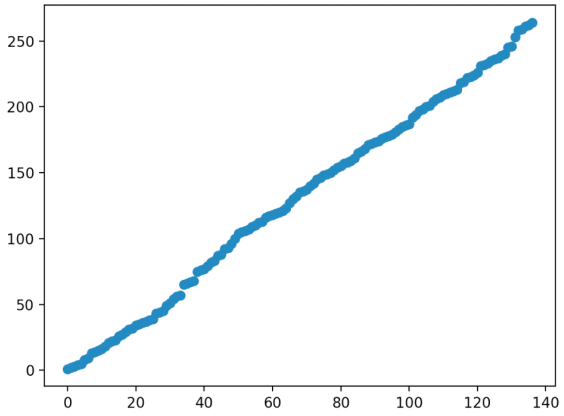
Cluster 9	**Size: 12**
Direct Production	[19, 10, 6, 10, 13, 0, 0, 1, 3, 0, 2, 21]
Direct Consumption	[19, 11, 1, 12, 7, 0, 1, 3, 18, 0, 1, 12]
Indirect Production	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
Indirect Consumption	[0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0]

Follower	[10, 9, 9, 4, 9, 5, 9, 8, 8, 9, 4, 8]
Following	[10, 9, 9, 4, 9, 6, 6, 9, 4, 9, 6, 11]
Removed users with id: 4579901 232294292 86351835	
Size: 9	
Direct Production	[19, 9, 6, 10, 13, 1, 3, 2, 21]
Direct Consumption	[19, 11, 1, 12, 7, 3, 18, 1, 12]
Indirect Production	[0, 0, 0, 0, 0, 0, 0, 0, 0]
Indirect Consumption	[0, 0, 0, 0, 0, 0, 0, 0, 0]
Follower	[7, 6, 6, 4, 7, 5, 7, 4, 7]
Following	[8, 6, 7, 3, 7, 6, 3, 5, 8]

Appendix 3: Expanded Communities

Algorithm 1
Cluster 1
Split users' rank in Cluster 1:  
Cluster 2
Split users' rank in Cluster 2:  
Split users' rank in Cluster 6:

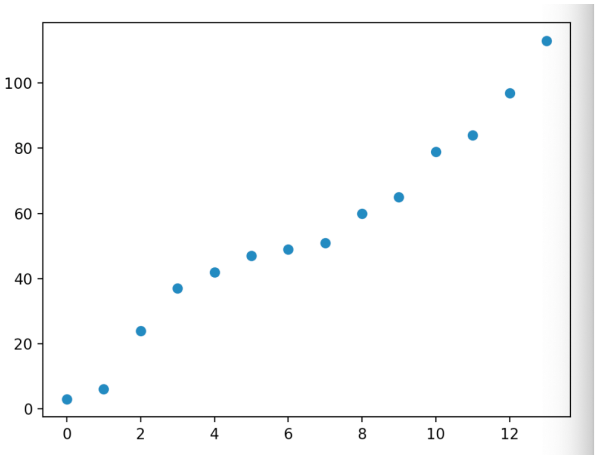


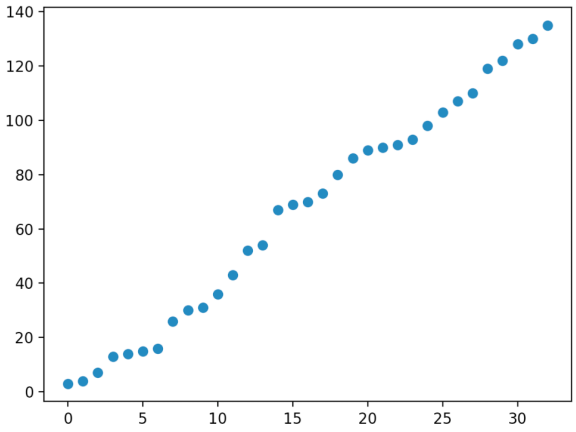


Algorithm 2

Cluster 1

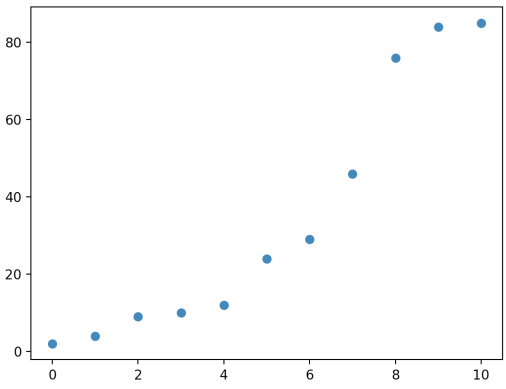
Split users' rank in Cluster 1:

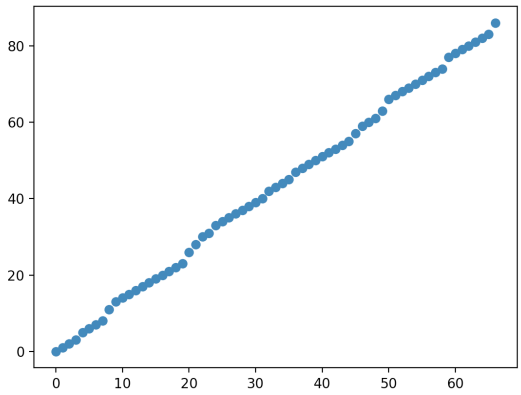




Cluster 2

Split users' rank in Cluster 2:





Cluster 6

Split users' rank in Cluster 6: