Final Report - Community Expansion Algorithms for Social Networks Using Twitter Data

Yuchen Zeng

December 24, 2022

Contents

0 Summary

This report focuses on finding an effective way to expand a community from a small group of users with the same interest on Twitter. We completed community expansion using four different methods. According to the experiment,

- **Refining the initial community before expansion** produces a better result.
- The more utilities we set restrictions on, the less noise we have in the expanded community. However, we increase the complexity of our algorithm.
- **A limit for the number of followers** is a must-have during community expansion.

About the community structure, we observed:

- Instead of having one core users, the community may have **a group of core users** and the size of core might vary based on the topic.
- It may be concerning that **top users have large difference between their number of followers followers**.
- Although we did not look for users with certain roles during the expansion, some new users have **a high score in one utilitiy and low in others**.

1 Introduction

1.1 Background

The following utilities provide us the information about sorting information flow and user connections for a user u:

- **Friends:** The number of users in the community that u follows.
- **Followers:** The number of users following u in the community.
- **Direct Production:** The number of retweets made by u's followers in the community.
- **Direct Consumption:** The number of retweets u made from u's friends in the community.
- **Indirect Production:** The number of retweets made by users that are not following u in the community.
- **Indirect Consumption:** The number of retweets u made from users that u does not follow in the community.

Previously, we used the first four utilities for new user selection in a community expansion. We found several **challenges** in building the community expansion algorithm:

1. There is no rigid standard for a good community. We tried assessing "keywords" that users usually mention in the same communities, but not in other communities. However, it is difficult to find the right group of keywords to start with.
2. The quality of community expansions largely depends on the quality of initial users. If initial users are not the "real" core users, we introduce noise into our expansion, and the community might be off-topic.

3. Since we are focusing on identifying memberships of users, we can introduce large accounts to the community due to their high utility scores. When a user has ten times more followers than the core user in the community, we know the community expansion went wrong.

1.2 New Ranking Functions

To have an more accurate membership identification, we introduced two new ranking functions that measures a users influence in the communuty.

1. **Influence 1:** The number of tweets from u retweeted by user u's direct followers + The number of tweets retweeted by u and later retweeted by u's direct followers. Score will be divied by number of tweets posted by u.

2. **Influence 2:** The sum of all **percentages** of all direct follower's retweets which can be roughly credited to u such as tweets retweeted later than u or tweets originated from u.

The advantage of influence 2 is that it reduce the effect of large number of retweets from few followers. A more detailed exploration of the two ranking functions are described in the report about community core detection.

1.3 Initial Users

To solve challenge 2, we need a more reliable initial cluster of users. Therefore, in this report, we used the top 10 users found by **community core detection**. The community core detection algorithm uses an **intersection ranking of influence1, influence2, production, and consumption** to find the core users of the community we are interested. For more information, please refer to the report about community core detection. Note that since we are only using the top 10 users, **the actual core of a community may be smaller or larger than 10**. Furthermore, since the initial users are core users, they should have higher utilities than other users. We can compare the expanded community utilities with initial users to analyze the performance of the expansion, which can be a solution to challenge 1.

1.4 Community Structure

Towards the end of the year, we discussed a new idea about community strcuture. We expect users in the community belongs in one of the three groups below:

- **Core Users:** Towards the end of the year, we discussed a new idea about community structure. We expect users in the community to belong to one of the three groups below:

- **Experts:** Users that are specialized in a certain action. They are users whose tweets and retweets are highly likely to be retweeted by the core. These are users with a high influence1 with respect to the core users only.

- **Consumers:** Users such that a large fraction of their retweets are tweets that are posted or retweeted by the core.

2 Algorithm

Initially, we took 200 candidates for each iteration. If we did not add any users to the community and $IsMore = True$, we would take 200 more candidates starting from the next iteration. The community expansion stopped when no users were added to the community and there was no user we did not go over. In each iteration, we added at most 40 users to the community.

Algorithm 1 Community Expansion

$Community \leftarrow$ Group of users
$PrevLength \leftarrow$ Number of users in $Community$
$P \leftarrow$ Number of Candidate we want ▷ Default 200
$IsMore \leftarrow True$ ▷ Whether there are more potential candidates
while $IsMore$ **or** PrevLength != $len(Community)$ **do**
    if PrevLength != $len(Community)$ **then**:
        Increment P by 200 ▷ If no one is added last time, but there are more user to check
    end if
    $PrevLength \leftarrow$ Number of users in $Community$
    $C\_Influence1 \leftarrow$ Influence 1 Ranking in $Community$
    $C\_Influence2 \leftarrow$ Influence 2 Ranking in $Community$
    $C\_Prod \leftarrow$ Production Ranking in $Community$
    $C\_Con \leftarrow$ Consumption Ranking in $Community$
    $Community \leftarrow Community$ ranked by intersection ranking

    $Candidates, IsMore \leftarrow$ **Select Potential Candidates**
    $NewCandidates \leftarrow$ **Filter Candidates**
    $Community = Community + NewCandidates$
end while
return $Community$

We found the potential candidates from users' friends list. We ranked each user by the number of followers they have in the current community, and then took less than P users.
The main difference between each community expansion algorithm is how we filter candidates. In this report, we explored the following four methods:

1. Filter Candidates using intersection of **4 Ranking Functions**. Users must have a minimum utility for all of production, consumption, influence 1 and influence 2 based on utilities of top users.

2. Filter Candidates using intersection of **4 Ranking Functions**. Then, filter candidate if they have **more than 150% followers of top users**

3. Filter Candidates using intersection of **3 Ranking Functions**, without influence 1. Then, filter candidate if they have **more than 150% followers of top users**

4. First, **expand the community to 20 core users**. Then, update the community using method 2 but **always take the top 20**. When the top 20 stop changing, start expanding community using method 2.

Algorithm 2 Select Potential Candidates

$Community \leftarrow$ Group of users
$T \leftarrow len(Community) * Threshold$
$P \leftarrow$ Number of Candidate we want ▷ Default 200
$IsMore \leftarrow True$ ▷ Whether there are more potential candidates
$Map \leftarrow$ Initialize an empty map of user with score
$Candidates \leftarrow$ Initialize an empty list of user
for each U in $Community$ **do**
    $Friends \leftarrow$ users that U follows
    for each F **in** $Friends$ **do**
        if F is **not in** $Community$ **then**
            Increment $Map[F]$ by 1
        end if
    end for
end for
$RevMap \leftarrow$Switch Map such that it becomes a map of score with users
for each $Score$ in $RevMap$ **do** ▷ Sorted in descending order
    if $Score \geq T$ **and** $len(Candidates) + len(RevMap[Score]) \leq P$ **then**
        Add $RevMap[Score]$ to $Candidates$
    else
        if $Score < T$ **then**
            $IsMore = False$ ▷ We have traversed through all candidates above Threshold
        end if
    end if
end for
return $Candidates$, $IsMore$

Algorithm 3 Filter Candidates(Intersection of 4 Ranking Functions and Follower Limit)

$Community \leftarrow$ Group of users
$Candidates \leftarrow$ Group of candidates
$T\_Influence1 \leftarrow$ Average Influence 1 of Top 5 users in $Community$
$T\_Influence2 \leftarrow$ Average Influence 2 of Top 5 users in $Community$
$T\_Prod \leftarrow$ Average Production of Top 5 users in $Community$
$T\_Con \leftarrow$ Average Consumption of Top 5 users in $Community$
$T\_Follower \leftarrow$ Average Number of Followers of Top 5 users in $Community$
$NewCandidates \leftarrow$Initialize an empty list for filtered candidates
for each U in $Candidates$ **do**
    $U\_Influence1 \leftarrow$ Influence 1 of U in $[U] + Community$
    $U\_Influence2 \leftarrow$ Influence 2 of U in $[U] + Community$
    $U\_Prod \leftarrow$ Production of U in $[U] + Community$
    $U\_Con \leftarrow$ Consumption of U in $[U] + Community$
    $U\_Follower \leftarrow$ Global Followers U
    if $U\_Influence1 \geq T\_Influence1$ **and** $U\_Prod \geq T\_Prod$ **and** $U\_Influence2 \geq T\_Influence2$ **and** $U\_Con \geq T\_Con$ **and** $U\_Follower < T\_Follower * 1.5$ **then**
        Add U to $NewCandidates$
    end if
end for
$C\_Influence1 \leftarrow$ Influence 1 Ranking in $NewCandidates + Community$
$C\_Influence2 \leftarrow$ Influence 2 Ranking in $NewCandidates + Community$
$C\_Prod \leftarrow$ Production Ranking in $NewCandidates + Community$
$C\_Con \leftarrow$ Consumption Ranking in $NewCandidates + Community$
$Ranking \leftarrow$ Initialize an empty map for new candidate ranking
$RevMap \leftarrow$Switch Map such that it becomes a map of score with users
for i in $range(len(C\_Prod))$ **do** ▷ Start from Rank 1 to the last rank
    $Intersection \leftarrow$ users that are higher than Rank i in all of $C\_Influence1, C\_Influence2, C\_Prod, C\_Con$
    for U in $Intersection$ **do**
        if U not in $Ranking$ and U not in $Community$ **then**:
            $Ranking[U] = i$
        end if
        if $len(Ranking) > 40$ **then**:
            break
        end if
    end for
end for
return List of users in $Ranking$

3 Initial Community

We used two initial communities, that are top 10 users from **community core detection**, mentioned in 1.3.

1. Topic: Machine Learning

rank	username	influence2	influence1	production	consumption	global follower
0	STLChessClub	0.946	0.060	217	326	36629
1	GrandChessTour	0.238	0.052	149	211	30546
2	WorldChessHOF	0.185	0.202	115	88	4168
3	FIDE_chess	0.221	0.035	247	54	207349
4	aicfchess	0.124	0.066	86	87	13291
5	TarjeiJS	0.074	0.015	33	425	25650
6	chesscom_in	0.157	0.015	42	224	25942
7	chess24com	0.246	0.046	622	47	146865
8	chesscom	0.214	0.071	286	8	397121
9	fionchetta	0.043	0.003	6	333	16167

2. Topic: Chess

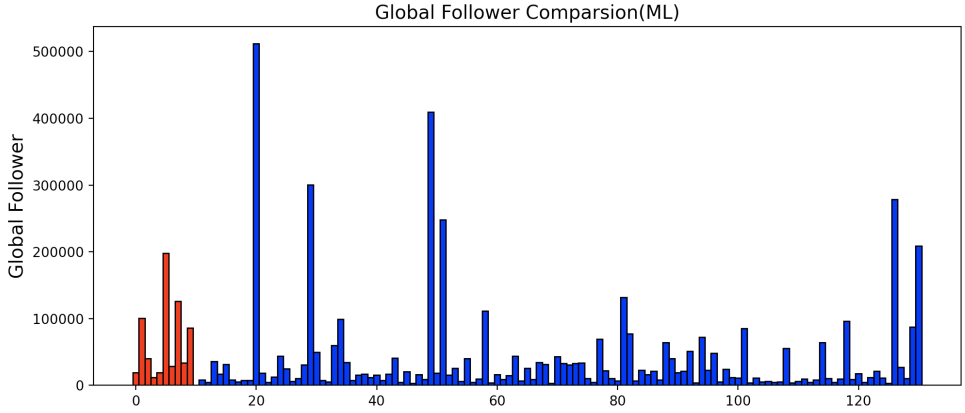
rank	username	influence2	influence1	production	consumption	global follower
0	egrefen	0.148	0.025	73	124	28600
1	_rockt	0.091	0.041	100	75	18825
2	kchonyc	0.077	0.014	41	20	33782
3	shakir_za	0.126	0.013	31	30	40132
4	hugo_larochelle	0.136	0.039	30	17	100941
5	sarahookr	0.085	0.013	27	48	19112
6	NandoDF	0.065	0.007	23	75	86375
7	JeffDean	0.128	0.037	48	9	197774
8	hardmaru	0.046	0.006	44	7	126226
9	ZoubinGhahrama1	0.044	0.03	20	32	12072

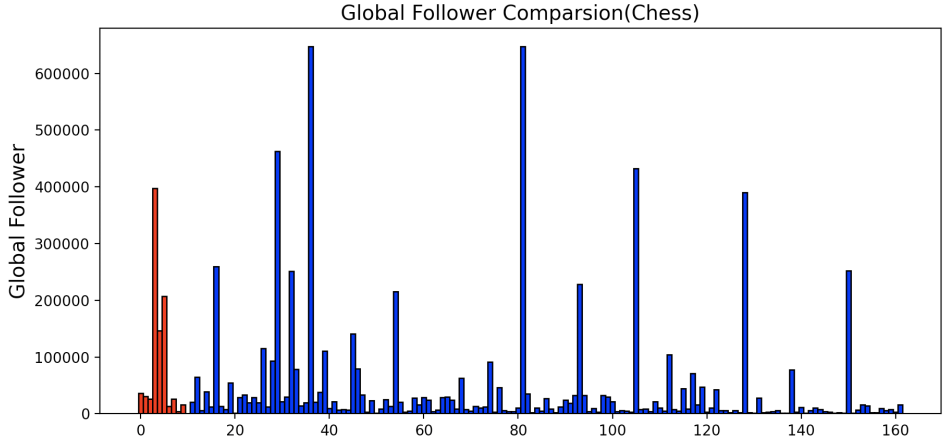
4 Results and Interpretation

For all the graphs, **red** represents initial users, and **blue** represents new users.

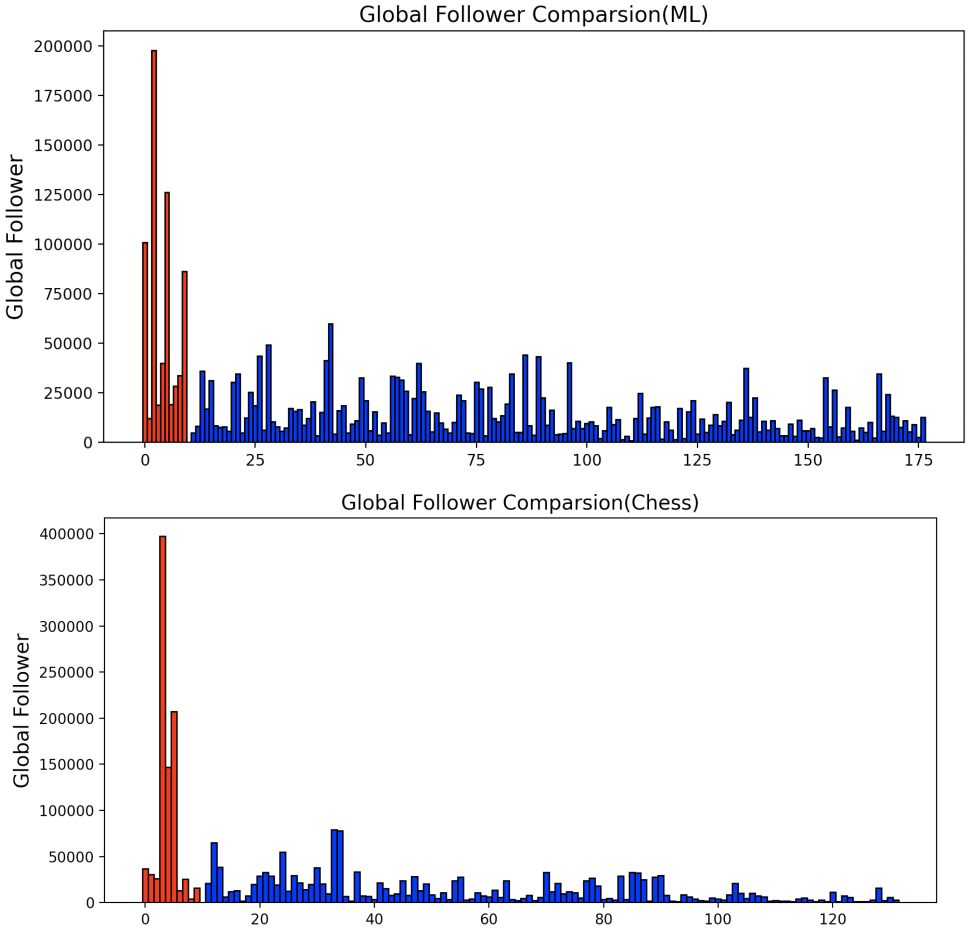
4.1 Filter Candidates using Intersection of 4 Ranking Functions

This method has issue because it introduces large accounts that have a much more higher number of followers than initial users.

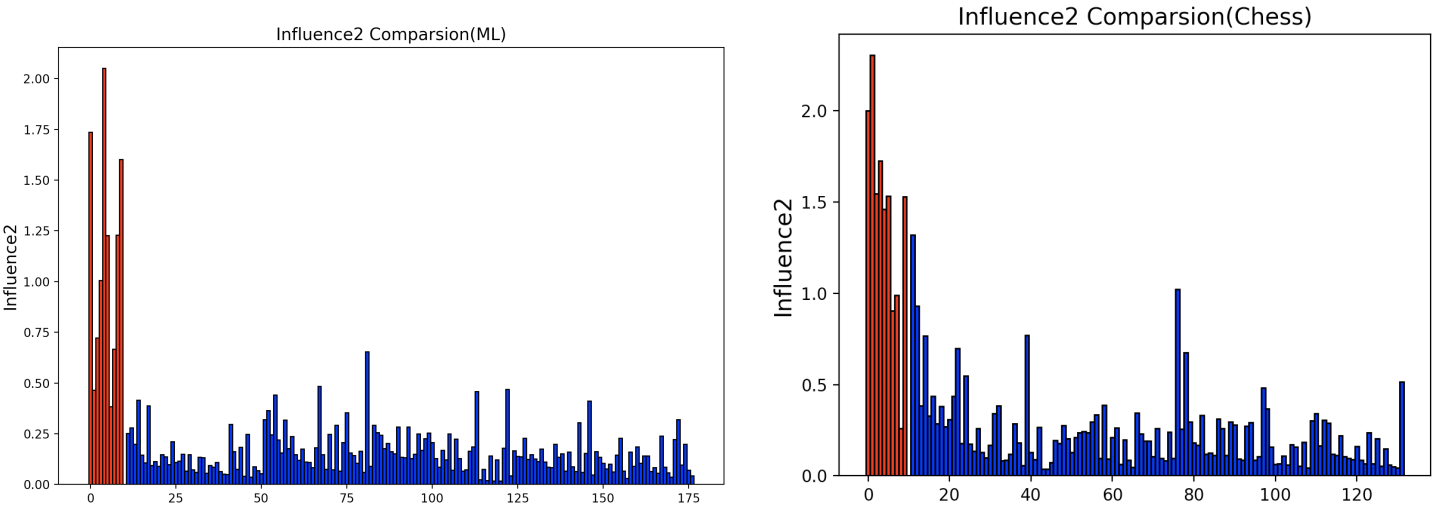




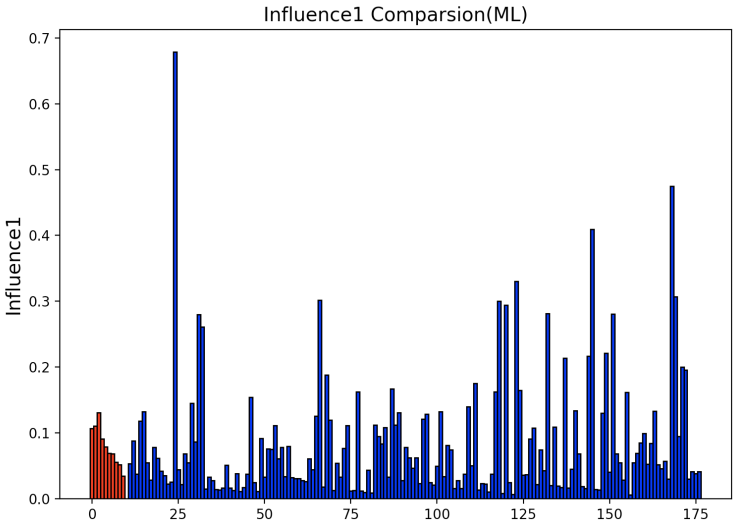
4.2 Filter Candidates using Intersection of 4 Ranking Functions and limit followers

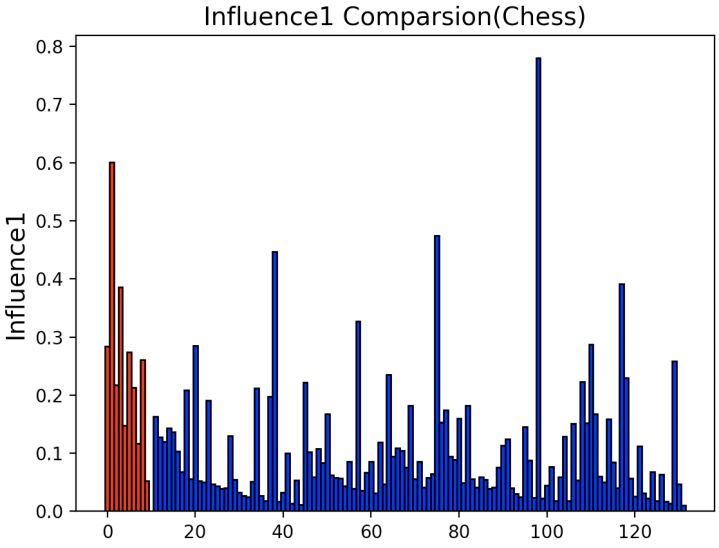


With the limitation on the number of followers, the graphs of global followers are more ideal. In general, initial users have a higher number of followers than new users. However, **there is a large gap between the number of followers of each core user** which can be concerning.

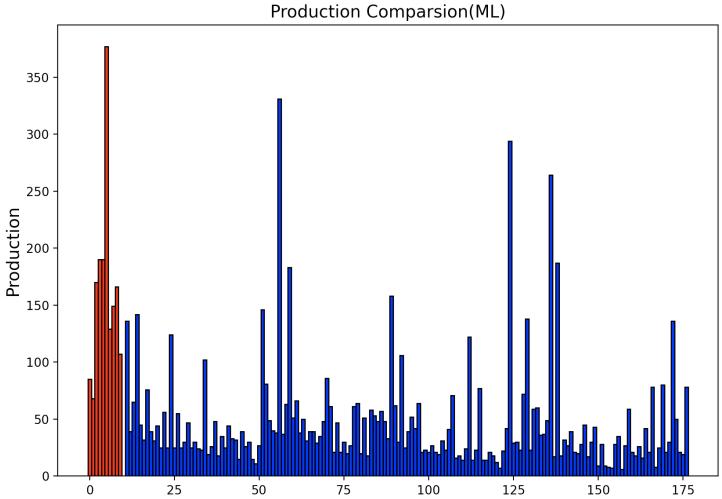


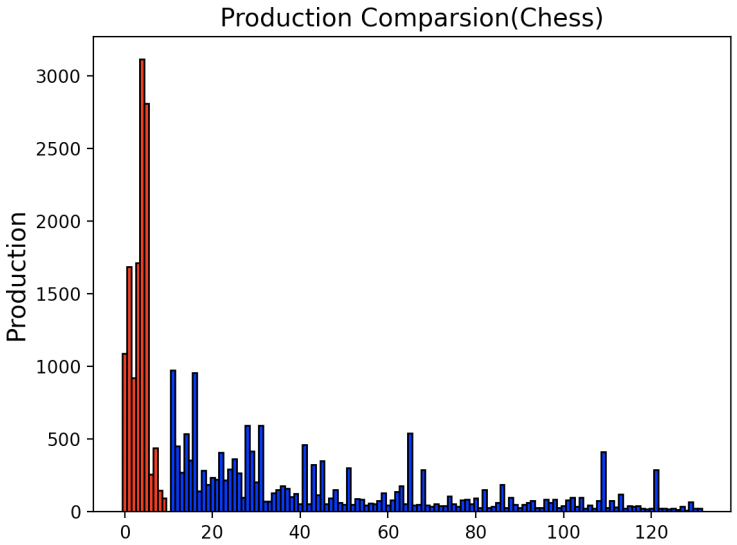
Influence 2 graphs look good since not much users have influence2 scores as high as core users.

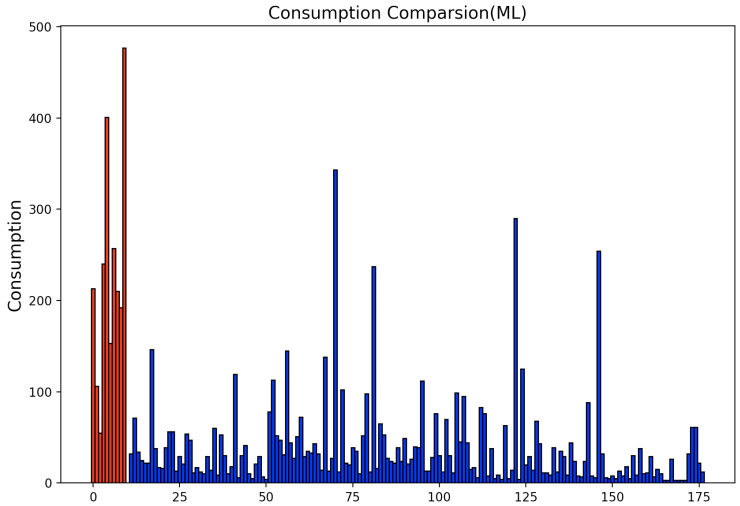


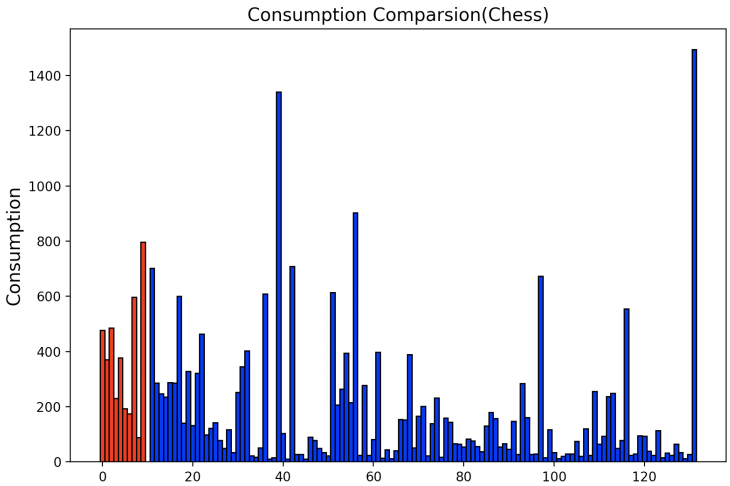


Influence 1 graphs look concerning, especially for ML community. There are many users that have higher Influence 1 scores than core user. However, they seems to have low scores in other utilities based on other graphs.

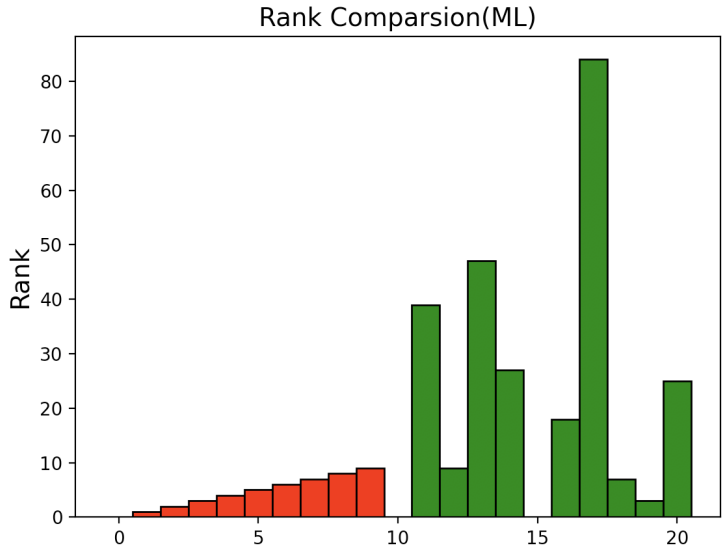


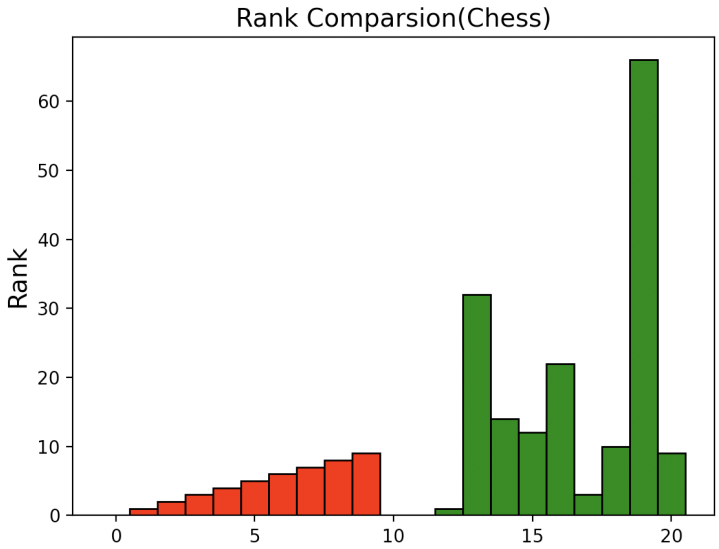






The graphs of production and consumption have similar concerns as Influence 1 graphs. Some users have high scores in a particular utility but not in others.

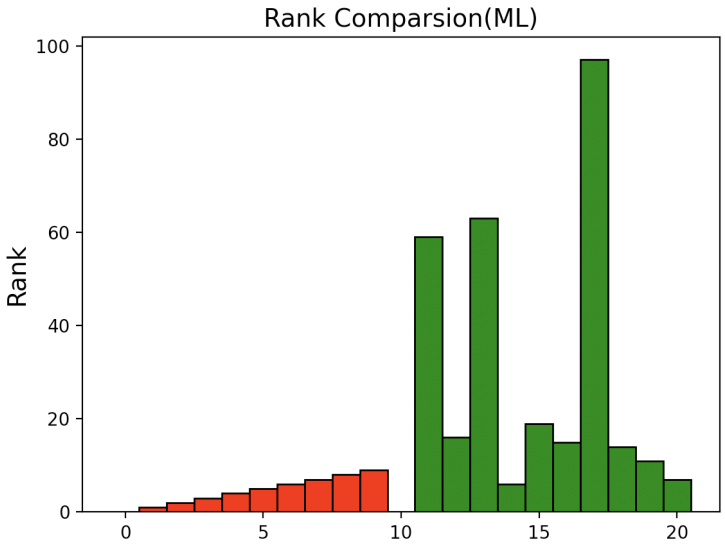


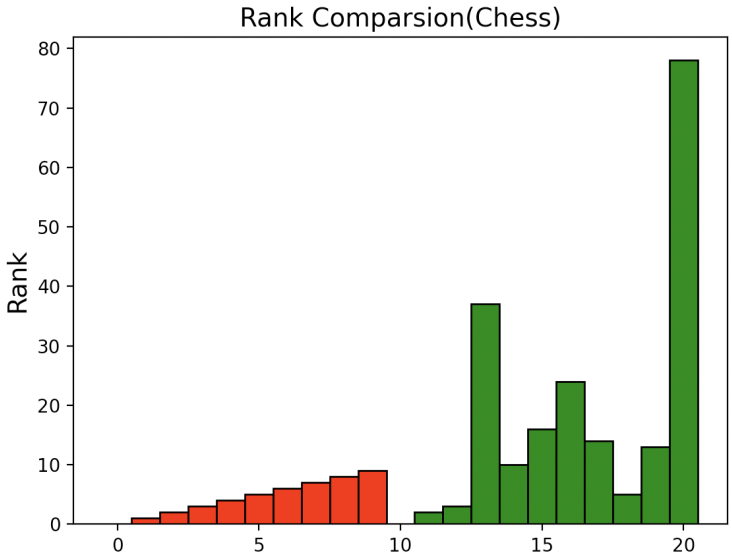


The Rank change graphs show the initial users' rank from the beginning to the end of the expansion. Some of the users might not be core users as their ranks dropped out of the top 50 and even 80 after the expansion.

4.3 Filter Candidates using Intersection of 3 Ranking Functions and limit followers

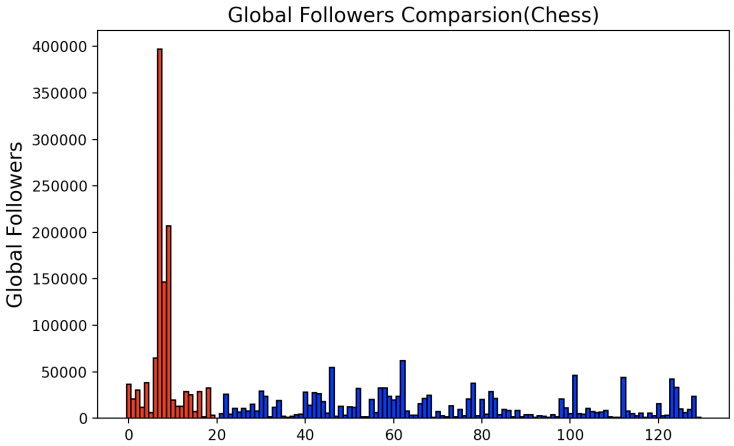
The utility graphs of Method 3 is similar to Method 2, avaliable in Appendix 6.1. However, The rank of core users dropped more after expansion than in Method 2.

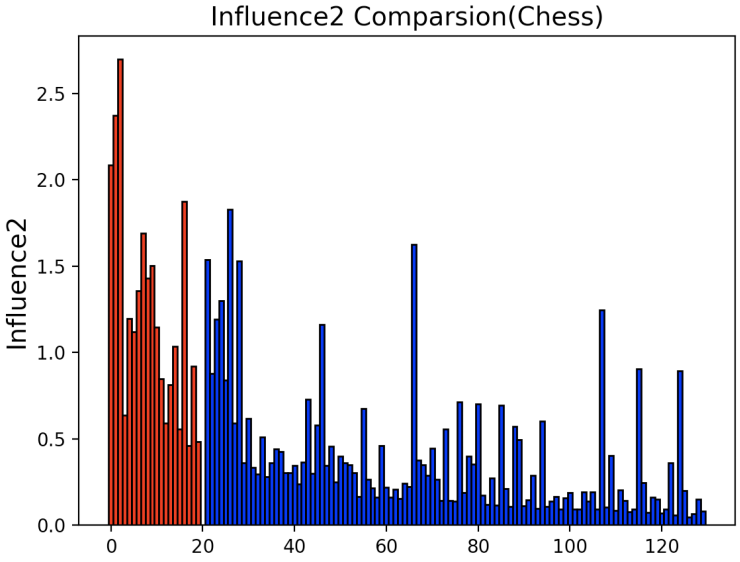


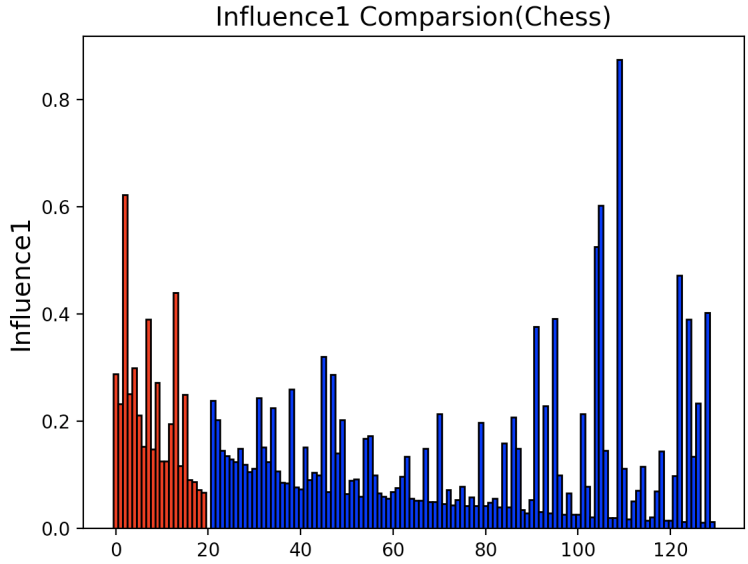


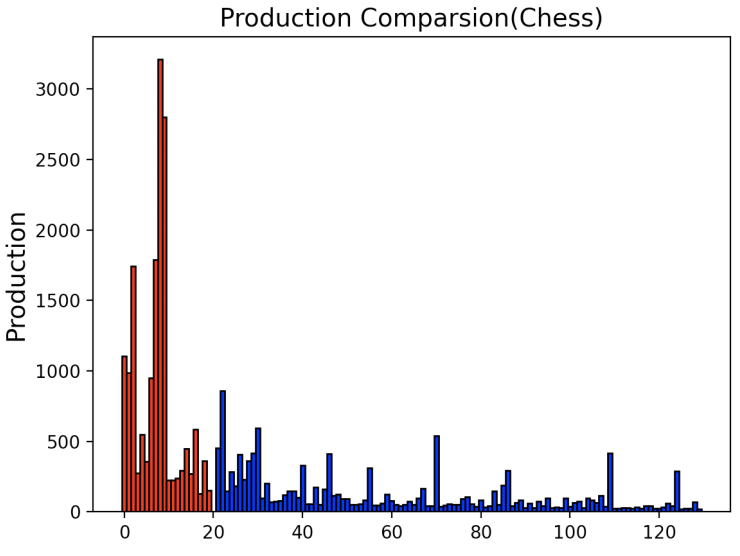
4.4 Find a Better Core Before Expansion

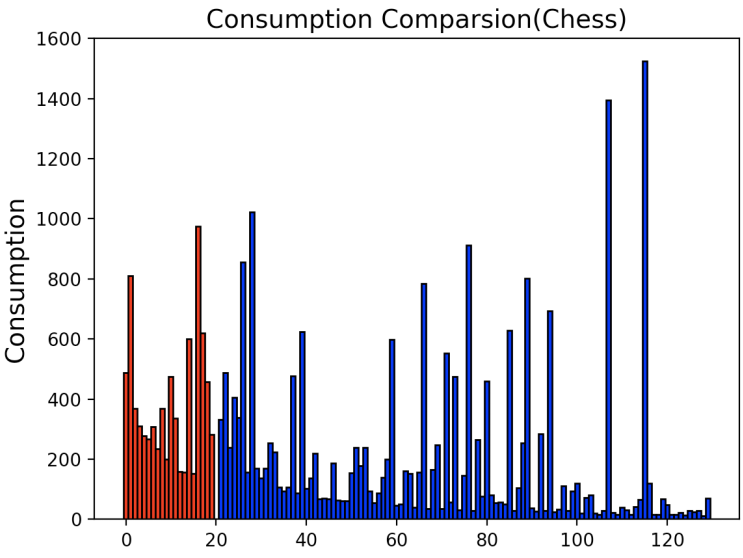
For Chess community, it took 4 iterations to find a 20 users core. 3 initial users are removed from the top 20. The downward trend of each utility scores are more visible using this method. The rank of core users dropped less than in Method 2 and 3. The users on average have a slight higher score than in Method 2 and 3.

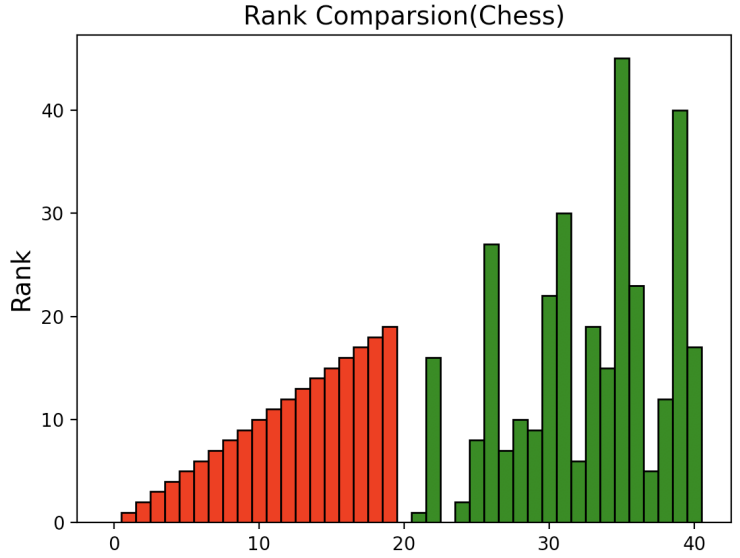

Global Followers Comparsion(Chess)
Global Followers
400000
350000
300000
250000
200000
150000
100000
50000
0
0
20
40
60
80
100
120


Influence2 Comparsion(Chess)
Influence2
2.5
2.0
1.5
1.0
0.5
0.0
0
20
40
60
80
100
120


Influence1 Comparsion(Chess)
Influence1
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
120


Production Comparsion(Chess)
Production
3000
2500
2000
1500
1000
500
0
0
20
40
60
80
100
120


Consumption Comparsion(Chess)
Consumption
1600
1400
1200
1000
800
600
400
200
0
0
20
40
60
80
100
120


Rank Comparsion(Chess)
Rank
40
30
20
10
0
0
10
20
30
40

5 Conclusion

So far, **Method 4** produces the best expanded community. The key difference between Method 4 and other algorithms is that it **refines the initial community(group of core users)** before expansion. Instead of having one core users, the community may have **a group of core users** and the size of core might vary based on the topic. Also, by comparing results from Method 2 and 3, setting restrictions on **more utilities reduces the noise slightly**. There is a trade-off between performance and complexity. Furthermore, **a limit for the number of followers** is a must-have during community expansion. Since we are looking for membership of users, none of the utility restrictions can stop introducing a user with "too high utility scores" into the community. However, it may be concerning that **top users have large difference between their number of followers followers**.

6 Next Step

6.1 Identifying Experts and Consumers

Recall the community structure contains **core users, experts, and consumers**. In the future, have a specific step to identify experts, and consumers can make community expansion more accurate. So far, we noticed that **there are users with extremely high influence 1 scores than core users but low scores in other utilities**, it is highly possible that they are the "experts" in the community. Similarly, there are users with high consumptions that may be the "consumers", but we need to **create a new ranking function that is sepcific to retweeting from core users**.

6.2 New Ranking Functions

As mentioned in 6.1, we can have more rankings functions that are speicifc to certains roles of a user in the community:

1. Percentage of tweets and retweets that are retweeted by the core. Similar to a high influence 1 with respect to **core users** only.

2. Percentage of tweets and retweets that are posted **from the core**, or after core users retweeted.

6.3 Tuning Parameters

So far, all the thresholds are not tuned to produced the best result. Here are the variables that we may want to modify:

1. Number of Potential Candidates: By default 200 and increase when running out of candidates.

2. Maximum Number of Candidates for Each Iteration: By default 40.

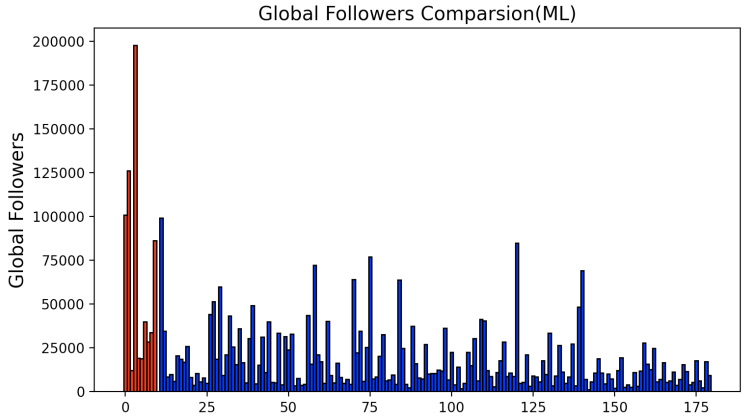
3. Threshold for utility scores: By default 20% for all the utilities functions.

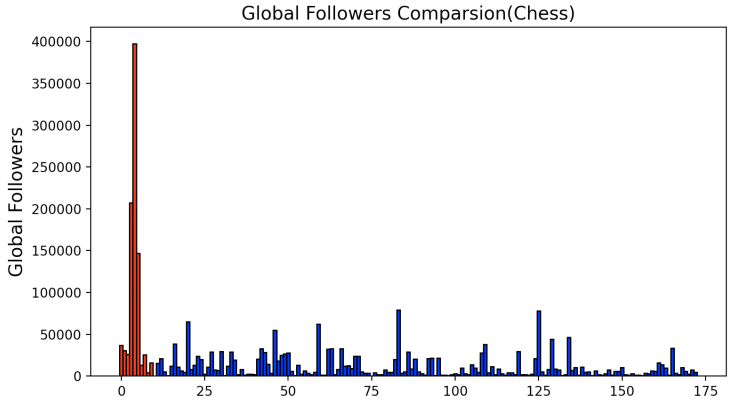
4. Calculate threshold value: By default using average of top 5 users. We may want to expand the average to include more or less users, depends on the size of community core.

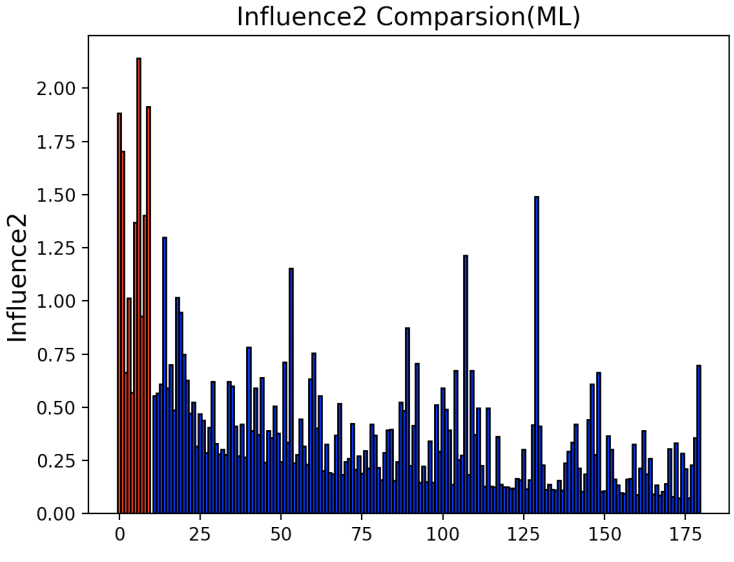
5. Threshold for Maximum Number of Followers: By default 150%. We may also try calculate using maximum of top X users, instead of the average.

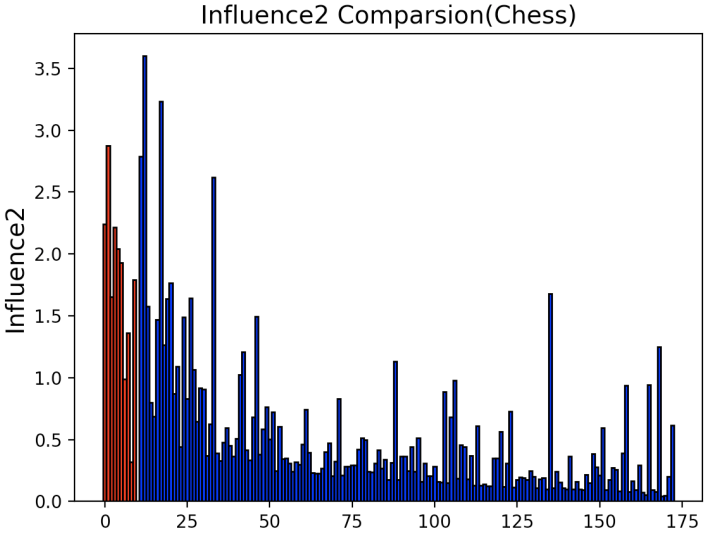
7 Appendix

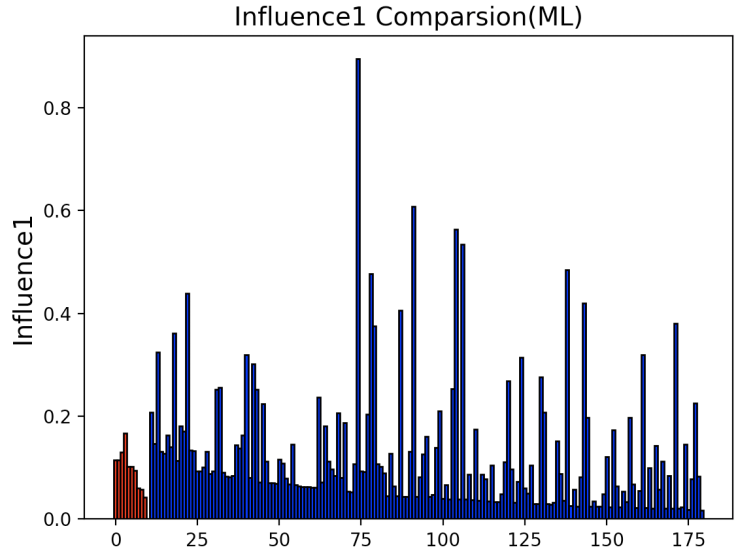
7.1 Results: Filter Candidates using Intersection of 3 Ranking Functions and limit followers

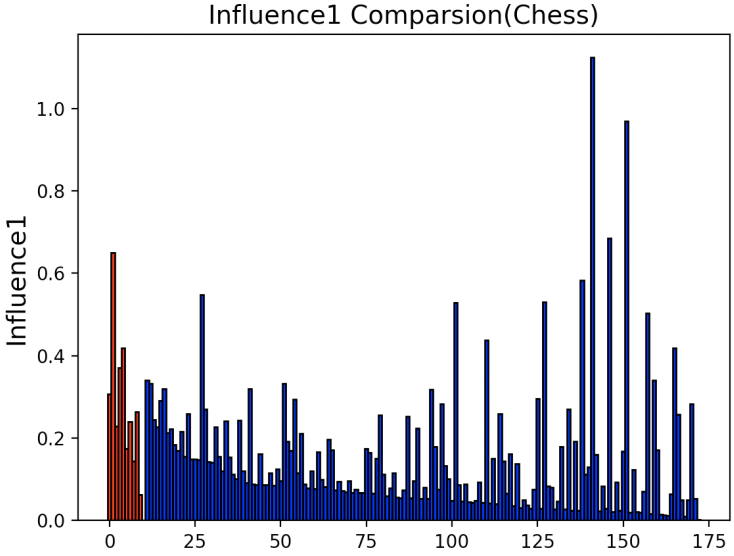


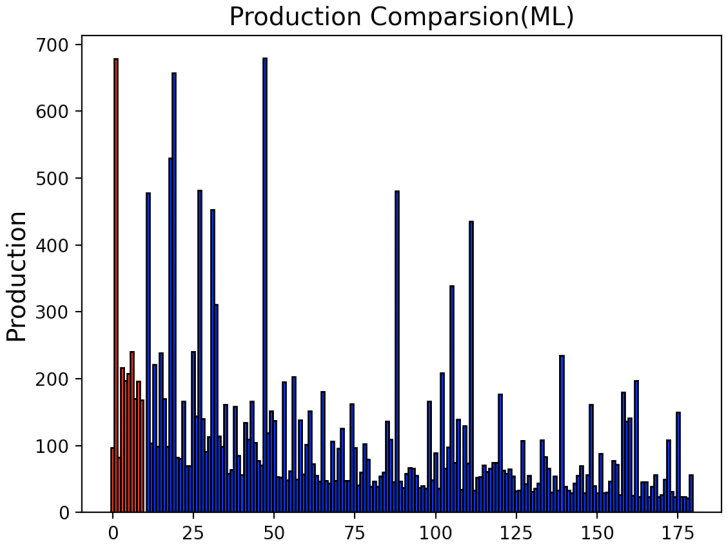


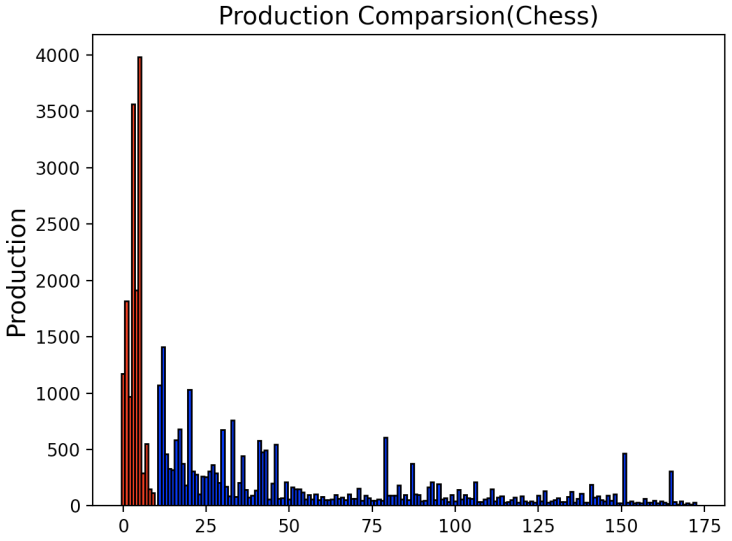


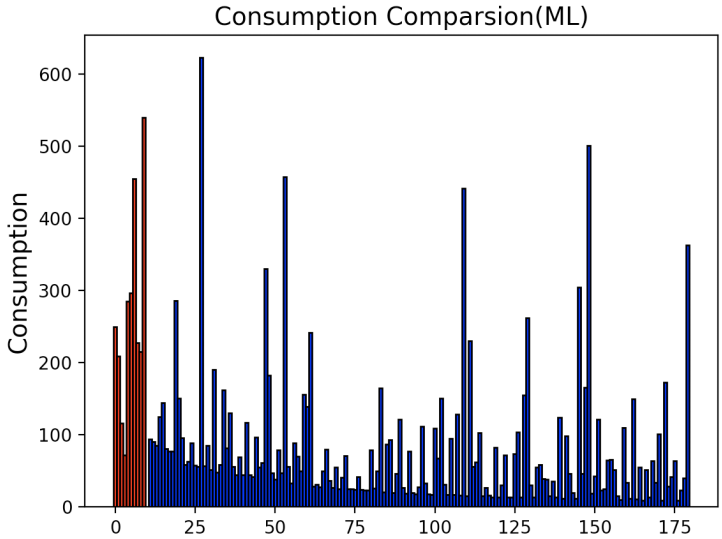


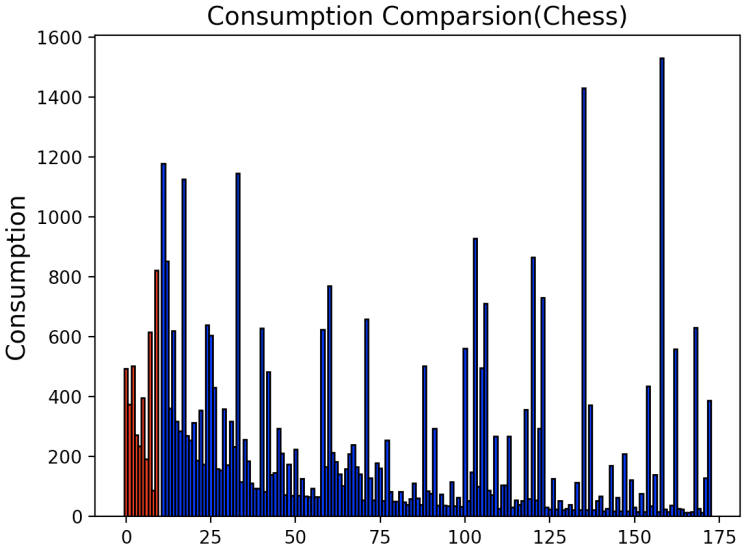





Production Comparsion(ML)
Production
0
100
200
300
400
500
600
700
0
25
50
75
100
125
150
175


Production Comparsion(Chess)
Production
0
500
1000
1500
2000
2500
3000
3500
4000
0
25
50
75
100
125
150
175


Consumption Comparsion(ML)
Consumption
0
100
200
300
400
500
600
0
25
50
75
100
125
150
175


Consumption Comparsion(Chess)
Consumption
0
200
400
600
800
1000
1200
1400
1600
0
25
50
75
100
125
150
175