Mining Mastodon for Silent Users

CS 483

Chan Lee, Sang Nguyen, Nancy Pitta*, Sidney Mei
University of Illinois at Chicago
cl84@uic.edu,snguye62@uic.edu,npitta2@uic.edu,sidney2@uic.edu

ABSTRACT

Knowing the mindset of silent users and their intentions is integral in social media, which grows and pushes recommendations in any aspect via data from all its users. However, silent users don't contribute much to social media. Analyzing such users would be beneficial in determining how to engage with them and make them active. They make up the majority of users on various social media, and despite their little to no contribution, there must be some level of inter- est towards the platforms they continue to log on to. Our project attempts to identify the silent users and discover any insightful behavior they may display. We examined 6 years of 4 different Mastodon servers data consisting of nearly 100k users in each server and dealt with 200k toots to analyse user patterns. We later show that determining the users' neighbors and associating a new user with them enables us to mark a user to be silent or active with high accuracy, also showing that we can leverage the little data available from silent users and identify similar new users.

ACM Reference format:
Chan Lee, Sang Nguyen, Nancy Pitta*, Sidney Mei. 1997. Mining Mastodon for Silent Users. In *Proceedings of CS 483, UIC, Apr 2023,* 6 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Social media has become an integral part in today's world via giving its users the ability to communicate across the world or to know people's opinions. There is a growing interest in a particular group of social media users, commonly referred to as "silent users", who tend to remain silent and consume content or engage in non-content-generating activities. These users are contrasted with "active users", who contribute the majority of social media content. Despite their significant numbers, silent users have received little attention in social media research, which has mostly focused on active users. This has led to biased representations of topical interests and sentiments since silent users are not considered in the analysis due to their low content generation. For instance, when mining user interests and sentiments, silent users are often overlooked, leading to skewed results. While they may not contribute content, silent users are still active individuals with unique interests and preferences. They pay attention to topics that interest them and actively seek out relevant content. Additionally, they can provide valuable feedback through ratings and reviews on consumer products and may also be a target audience for marketing efforts. It is important to consider that silent users may have different demographic and opinion distributions than active users, and neglecting to account for them could result in a skewed understanding of overall population-level interests. For example, Gayo-Avello (2012) noted that the low accuracy of election predictions using Twitter data is partially due to the neglect of the "silent majority," emphasizing the need for future research to investigate this group further.

2 DATA

2.1 Data Collection

Our data is collected from four different Mastodon servers namely:

- mastodon.social
- mstdn.social
- mas.to
- techhub.social

through the use of Mastodon.py, a Python wrapper for the Mastodon API.

We created bot accounts on each instance to access their public data. We used worker pools to collect data simultaneously on April 2nd and completed the collection on April 4th. We originally planned to collect more information, such as likes, shares, and messages, to determine the active status of a given user. However, the Mastodon API restricts bot accounts from collecting private user information.

Through the accounts endpoint, we collected the following user attributes:

- id - *numerical value identifying a user*
- bot - *boolean indicating whether an account is automated*
- silenced - *boolean indicating whether an account is suspended*
- locked - *boolean indicating whether an account can be followed without a request*
- created_at - *account creation time*
- last_status_at - *the last time a public status was created*
- statuses - *how many statuses they have*
- followers - *how many people follow them*
- following - *how many people they follow*

The first three instances have 100,000 accounts each (totaling 300,000). The latter has 11,004 accounts which will be used to test the two different clustering models.

2.2 Data Cleaning

We filtered all the bots, silenced and locked users as bots don't contribute to our goal. Furthermore, silenced and locked accounts were separated from "regular" users as they only skew our analysis. Since Mastodon is created in 2016, we replaced all null and wrong values, such as data created in year 1018, in the data for the dates columns with December 2015. Data collected from Mastodon.social has total 100k users and after filtering the bots, silenced users and users that are locked, we got 99.8k users. Similarly for mstdn.social, initially there were 100k users and after filtering, we got 94.9k users. For mas.to server, initially there were 100k users and after filtering, we got 93.4k users. We have decided to drop silenced, locked users and bots as they don't provide any insight into a regular user who uses the platform.

3 DATA ANALYSIS

3.1 Metrics

3.1.1 Toot per time. We define toot per time as the criteria to classify a given user as silent or active. We considered the time since the user last posted and the average status count of all users in a given server per time period (one day) to determine the activity status of a user.

$$Toot\ per\ time = \frac{Total\ toots\ since\ user\ joining}{Days\ since\ user\ joined\ Mastodon\ server} \quad (1)$$

where last_status_at is set as the final day of collection (Apr 5th) if null. This metric is essentially tootpertime with the current date replaced with last_status_at. This yields more consistent values so that our analyses remain the constant.

3.1.2 Engagement ratio. We also defined a new metric namely engagement ratio which condenses the information in status count, followers count and following count. We want to assign more weightage to the statuses posted and lesser weightage to their followers or follower count. We defined it as shown here in second equation.

$$Engagement\ ratio = \frac{2 * status\ count + 1}{follower + following + 1} \quad (2)$$

where following and follower can be seen as silent interactions, and statuses are active. Furthermore, a weight of 2 has been added to statuses to highlight their importance. A smoothing factor of 1 has been added to counter sparsity and numerical issues that may arise due to the skewed distribution of data and prevalence of zero values.

Based on the toot per time metric, we labeled users as silent if their toot per time ratio falls below the average measure or if their last post was more than a year ago. First metric is purely used for data analysis purposes and during training, we drop the label for the user, indicating whether they are silent or active. Second metric is used during training as it reduces the number of features by 3.

3.2 Data labeling

If a user falls below the average status count or when the last status that the user posted was a year ago then we label the user as 'silent' or else label it as 'active'. We append an extra column to store these labels.

In Mastodon.scl, 96k users are silent and 3.4k users are active with 574 toots per day on average. In Mstdn.social server, 74k users are silent and 20k users are active with 1955 toots per day. This server is the most active server among all other Mastodon servers. In Mas.to server, there are approximately 87k silent users and 6k active users with 2454 posts per day. This server has highest posts per day count among all other Mastodon servers. Based on these results, we can say that Mstdn.social is more active than Mastodon.scl. However Mas.to is not considered active as the thresholds for days since the user posted last was different for it since the majority of its users are silent and we excluded the posts per time period filter for its results.

Table 1: Popular vs unpopular

Server	Popular	Unpopular
Mastodon.scl	2.9	151
Mstdn.social	0.46	3.7
Mas.to	2.3	15.5

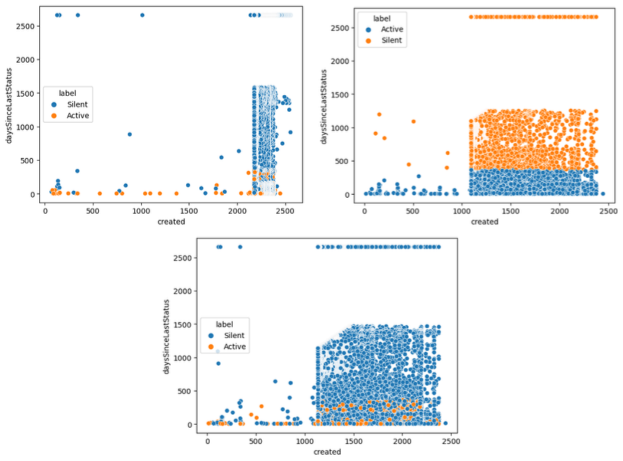

Figure 1: Proportion of silent and active users among old and new users

3.3 Popular user trends

We define a popular user as a user with a follower count greater than the average follower count on the given server. Our results are in Table 1.

We can say that there are more silent users among un- popular users than popular users. Hence we consider follower count to be a potential feature for our training

3.4 User retention trends

We define new users as the accounts created in the last 3 years and the rest as old users. We plotted the days since user last posted vs days since user created account to check the distribution of active vs silent user trends as in Figure 1.
We can say that the proportion of active users is higher among the new users than old users on Mastodon.social and mstdn.social servers but the proportion seemed to be similar in mas.to server.

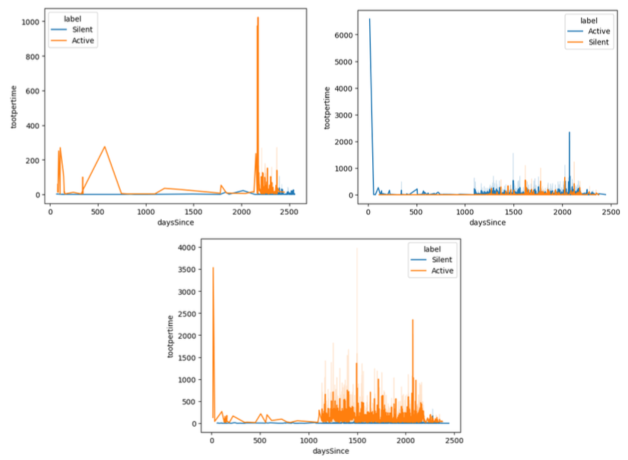

Figure 2: Average posts per day variation among old and new users

3.5 Activity status of old & new users

In Figure 2, we can see that there are more old users that post more often than recently joined new users.

4 FEATURE SELECTION

We now have user attributes like ID, whether the user is a bot or if user is silenced or if user is locked ot of the account. we also have two raw dates, denoting the date on which user is created at and the data since the user posted the last status on the server. Since we already filtered all the bots, silenced and locked users, we no longer need these features.
Apart from that, we want to keep track of only the number of days since the user joined Mastodon server and the number of days since the user last posted a status on Mastodon server. Hence, we extract the days values for both the columns. If the dates are missing for either we replace the creation date with December 2016 as that was the date Mastodon was created. If dates are missing for the last status post, we keep the users since those are the users that might not have posted a single post so far.

5 TRAINING

The aim of our project is to compare and contrast the efficacy of two clustering models, namely CURE, and Bisecting K-Means, to ascertain their relative performance. Each algorithm's effectiveness is measured using clustering evaluation metrics, specifically the Silhouette score and Calinski-Harabasz score.

Though we tried to use BFR (Bradley-Fayyad-Reina) algorithm initially, the data was not normally distributed along any dimension, so we decided to not use the BFR algorithm or K-means for our project.

The remaining features: created_at, last_status_at, statuses, followers, and following were combined into two new features. Limiting the features to 2 will allow for better interpretation, visualization, and performance of the data. The downside to this approach would be the loss of information. Nonetheless, they are determined by the metrics defined previously in equation 1 and 2.

5.1 Data normality

Most of the algorithms that were discussed in class like k-means or BFR assumes that the data is normally distributed along each dimension. Hence we checked the normality along each dimension for the data of all 4 servers. Since our data has many repeated values for each dimension, especially since we rounded off the custom metrics we have calculated, we chose Pearson's test to check if data is normally distributed. P-value for the hypothesis that data is normal came to be 0 for all the dimensions and hence we concluded that data is not normally distributed along any of the dimensions. So we did not use BFR or k-means on our collected data.

5.2 Evaluation metrics

To evaluate the clustering algorithms, we picked Silhouette and Calinski scores.

5.2.1 Sihouette score. The Silhouette is a measure for the validation of the consistency within clusters. It ranges between 1 and -1, where a value close to 1 means that the points in a cluster are close to the other points in the same cluster and far from the points of the other clusters.The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b).

5.2.2 Calinski Harabasz score. It is also known as the Variance Ratio Criterion. The score is defined as ratio of the sum of between-cluster dispersion and of within-cluster dispersion. While there is no fixed range for this metric, a higher value is typically indicative of better clustering.

5.3 Data description

Our initial direction was to compare the results of CURE and BFR along with other hierarchical clustering algorithms. In our case, the data is not normally distributed along any dimension and not normally distributed. Hence we did not use BFR algorithm for our problem.

Three further transformations were made at this stage, in addition to the optimizations made in the data cleaning tasks.

(1) Accounts that were less than 30 days old were removed because they were too new to be reliably determined as active or silent.
(2) Outliers that were below the first or above the third quartile (0.01 and 0.99, respectively) were removed.
(3) The MinMaxScaler in PySpark was utilized to normalize features between 0 and 1. This makes it easier for us to see the underlying patterns in the data.

The data was divided into two sets, a training set and a test set, using an 80-20 split. This rule is known as Parelo's Principle; by which, the training set was used to train the model, and the test set was used to evaluate the model's performance.

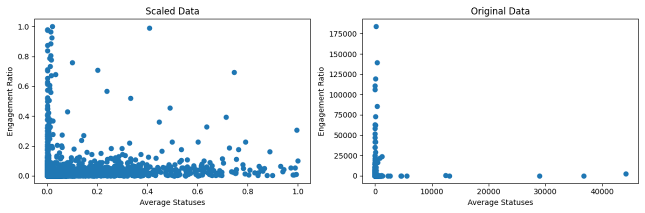

Figure 3: The features post- and pre-transformation

5.4 CURE

CURE (Clustering Using REpresentatives) is a clustering algorithm that works by performing hierarchical clustering on the random sample till the number of clusters passed to the algorithm were reached. It starts by merging two closest pair of points and repeats merging till the desired number of clusters is reached. Then it selects a predefined number of representative points from each cluster which we set to be the default value of 5. These points are compressed or moved closer toward the centroid of the cluster by a compression factor which we set to be 0.3. If representative points are closer then we merge their corresponding clusters.
In the second pass, we assign new points to the clusters that is the closest to the point. After each pass, we recalculate the centroid of the cluster, compress it and then repeat the same till we process all the data.

Since our data is not normally distributed along each dimension, we chose this for our project. The CURE algorithm was run on the techhub.social dataset with a varying number of clusters, starting with 2 and increasing up to 20. At least 2 clusters are required to obtain a Silhouette score. Silhouette scores and Calinski-Harabasz scores were computed to determine the efficacy of the clustering methodology.

Figure 4 shows that $k = 2$ is the optimal value for the algorithm to achieve a satisfactory level of clustering accuracy. The clustering accuracy for k=2 is greater than 0.9, which is close to the maximum value of 1. Whereas, Figure 5 shows that $k = 4$ is optimal to achieve the best clusters for this algorithm.

5.5 Hierarchical clustering (Bisecting K-means)

This is a modified bisecting k-means algorithm based on Steinbach, Karypis, and Kumar's paper "A comparison of document clustering techniques" that is tailored for Spark.
The algorithm starts with a single cluster containing all points and then iteratively divides clusters at the bottom level using k-means until there are k leaf clusters or no further division is possible. The bisecting steps of clusters at the same level are grouped together to increase efficiency. If bisecting all bottom-level clusters would result in more than k leaf clusters, the larger clusters are given priority.

Like CURE, Figure 6 shows that $k = 2$ is the optimal value to achieve a satisfactory level of clustering accuracy for the Silhouette scoring. However, Figure 7 shows that using a significantly higher amount of clusters, $k = 15$ is optimal to achieve the best clusters for this algorithm.

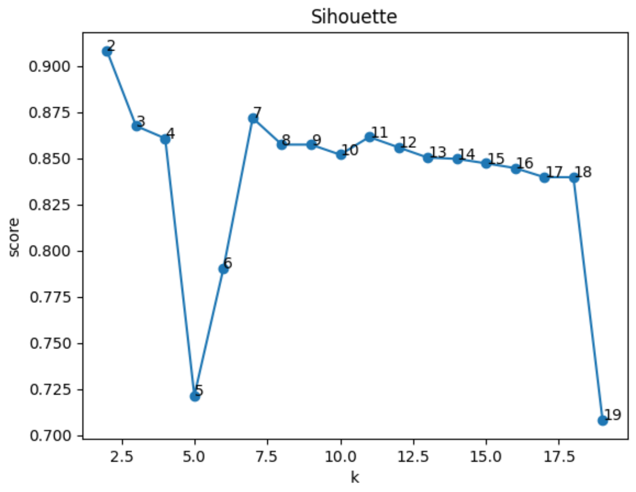

Figure 4: Scores peak at $k = 2$ for CURE

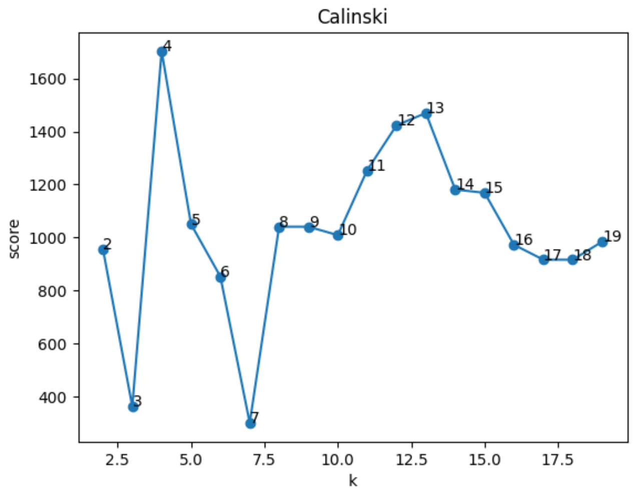

Figure 5: Score peak at $k = 4$ for CURE

Table 2: Evaluation metrics

Score	k	Algorithm
Silhouette	2	Bisecting k-means
Calinski	15	Bisecting k-means

5.6 Insights On The Models

With the scores for the two algorithms computed for the smaller data set, we can analyze the data for further findings in order to select the optimal model for the larger data set.

The scores (rounded to the nearest thousandth) for Silhouette and Calinski-Harabasz are 0.947 and 3197.373, respectively. These are the highest scores for all models, which indicates that Bisecting K-means is the most optimal model for the data set.

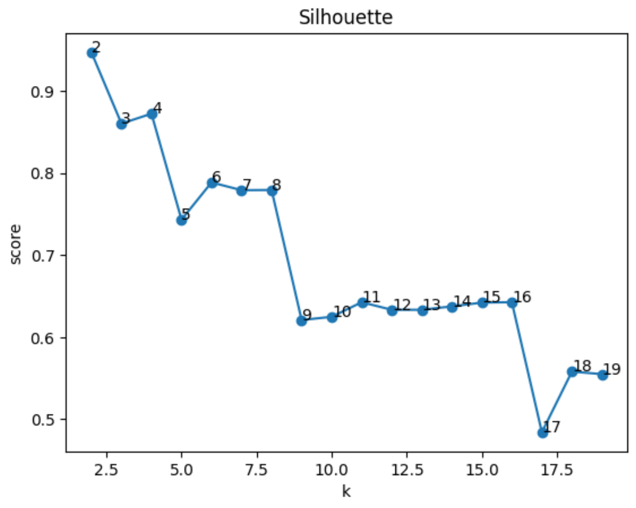

Figure 6: Scores peak at k=2 for Bisecting K-means

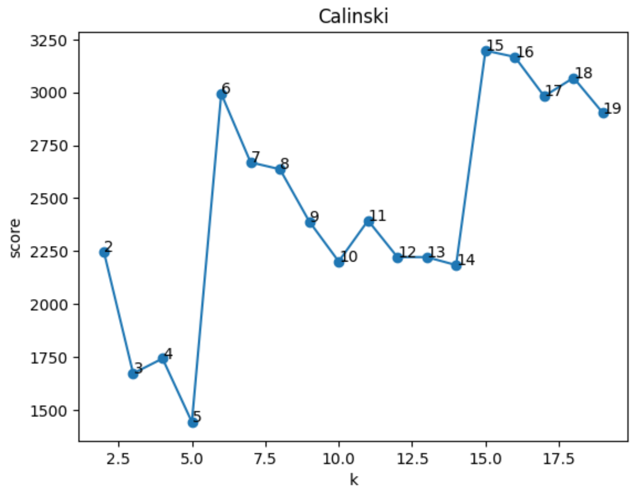

Figure 7: Scores peak at 15 for Bisecting K-means

The former suggests that the data points within the clusters have high intra-cluster similarity and low inter-cluster similarity. i.e. the data points within the clusters are closer to each other and farther from points in other clusters. The latter alone does not tell us much about the model's performance, but comparing the scores between the two models shows that Bisecting K-means is able to generate clusters that are better defined and more separated from each other.

6 RESULTS

The data collected from mastodon.social, mstdn.social, and mas.to are combined to form the final dataset. The evaluation of the data follows the same 80-20 split which was used in the smaller dataset.

The scores yielded were 0.942 and 56239.313, respectively, which is indicative of high intra-cluster similarity and low inter-cluster similarity.

The figure below shows that silent users average lower amounts of statuses and have little to no engagement, which is how we defined "lurkers" or silent users.

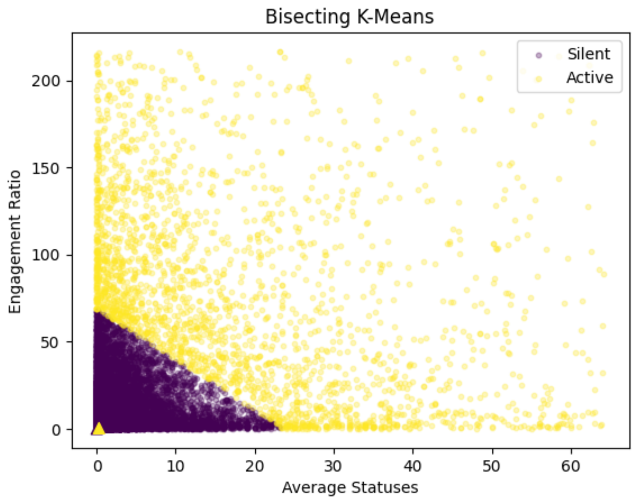

Figure 8: Scores peak at 15 for Bisecting K-means

The percentage of users in each cluster is 4.207% and 95.793%, respectively. The former refers to active users and the latter to silent users. This represents the fact that a majority of users on social platforms are silent. Some sources estimate between 60-80 percent of users are silent; whereas others estimate 90 percent of users as silent.

96% of users are silent users, which is over 90%. This is probably due to how silent users are defined as those who don't interact at all. This is within the 90-9-1 rule proposed by Jakob Nielsen, which states that 90% of users are lurkers, 9% of users contribute occasionally, and 1% of users contribute frequently.

7 CHALLENGES

- Mastodon's API is restrictive, which limited our access to critical attributes of user to see if they logged in or not to understand if the user is still interested in the platform.
- Setting thresholds for servers due to the difference in activity level was challenging and we went ahead with average metric.
- Data is not normally distributed. Hence our options to choose from for the algorithms were limited.

8 WORK DISTRIBUTION

- Data collection: Chan & Sang
- Data analysis: Nancy
- Training: Nancy & Sidney
- Fine tuning and results: Sidney

9 FUTURE WORK

- With more user attributes like logging in and out, and user interactions on the platform, we could get more insight into the reasons for users being silent.

- Model could be extended to generalize not only for Mastodon servers but also to other platforms.

REFERENCES

[1] Wei Gong, *Characterizing Silent Users in Social Media Communities.*
[2] Almond, D., Currie, J., 2011. *https://web.stanford.edu/class/cs246/slides/05-clustering.pdf*

ACKNOWLEDGMENTS

We would like to thank Dr. Lu Cheng for providing the guidance at every step of the project and for helping us to understand the different steps to completion.