Efficient Representation of Interaction Patterns with Hyperbolic Hierarchical Clustering for Classification of Users on Twitter

Tanvi Karandikar*
Avinash Prabhu*
International Institute of Information Technology, Hyderabad
{tanvi.karandikar,avinash.prabhu}@students.iiit.ac.in

Avinash Tulasi
Arun Balaji Buduru
Indraprastha Institute of Information Technology, Delhi
{avinasht,arunb}@iiitd.ac.in

Ponnurangam Kumaraguru
International Institute of Information Technology, Hyderabad
pk.guru@iiit.ac.in

Abstract

Social media platforms play an important role in democratic processes. During the 2019 General Elections of India, political parties and politicians widely used Twitter to share their ideals, advocate their agenda and gain popularity. Twitter served as a ground for journalists, politicians and voters to interact. The organic nature of these interactions can be upended by malicious accounts on Twitter, which end up being suspended or deleted from the platform. Such accounts aim to modify the reach of content by inorganically interacting with particular handles. These interactions are a threat to the integrity of the platform, as such activity has the potential to affect entire results of democratic processes. In this work, we design a feature extraction framework which compactly captures potentially insidious interaction patterns. Our proposed features are designed to bring out communities amongst the users that work to boost the content of particular accounts. We use Hyperbolic Hierarchical Clustering (HypHC) which represents the features in the hyperbolic manifold to further separate such communities. HypHC gives the added benefit of representing these features in a lower dimensional space – thus serving as a dimensionality reduction technique. We use these features to distinguish between different classes of users that emerged in the aftermath of the 2019 General Elections of India. Amongst the users active on Twitter during the elections, 2.8% of the users participating were suspended and 1% of the users were deleted from the platform. We demonstrate the effectiveness of our proposed features in differentiating between regular users (users who were neither suspended nor deleted), suspended users and deleted users. By leveraging HypHC in our pipeline, we obtain F1 scores of upto 93%.

CCS Concepts

• **Information systems** → *Clustering and classification*; **Social networks**; • **Computing methodologies** → **Dimensionality reduction and manifold learning**; *Supervised learning by classification.*

Keywords

Elections, User Interaction patterns, Twitter, Hyperbolic manifolds

ACM Reference Format:
Tanvi Karandikar, Avinash Prabhu, Avinash Tulasi, Arun Balaji Buduru, and Ponnurangam Kumaraguru. 2021. Efficient Representation of Interaction Patterns with Hyperbolic Hierarchical Clustering for Classification of Users on Twitter. In *IEEE/WIC/ACM International Conference on Web Intelligence (WI-IAT '21), December 14–17, 2021, ESSENDON, VIC, Australia.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3486622.3493953

1 Introduction

Discussions on Online Social Networks (OSNs) are a key player in huge democratic processes such as elections. Recently, the 2020 U.S. General Elections [9] and the subsequent Capitol Riots [31] have been heavily influenced by conversations on OSNs such as Twitter and Parler [26]. Politicians and political handles are very active on OSNs, especially during times of democratic elections [22]. The nature of their engagement on these platforms is often an important part of their campaign strategies [20]. Particularly, OSNs have played an important part in the 2014 and 2019 General Elections in India, where studies show the success of the winning party was closely associated with their use of Twitter to engage with voters [1, 19]. Users of these OSNs can interact with the content shared by political parties. The more interaction such content gets, the higher the popularity and higher the chance that such content reaches more people [3]. Higher engagement with posts can also lead to benefits such as a higher number of followers and more media coverage [21]. This engagement could lead to a domino effect that can turn the tide of election results [15]. Due to the effect of this engagement on the outcome of democratic processes, it is important for OSNs to be fair in the way content is boosted.

Twitter in particular aims to keep their algorithm fair by endorsing and boosting content that is of genuine interest to most users.[1] This is important to ensure its status as a reliable platform for all political parties as well as voters by preserving interests spanning across all users.[2] However, a group of users can manipulate the system by closely working together to artificially engage with the posts of a particular account.[3] During democratic processes like

*Authors contributed equally to this research.

WI-IAT '21, December 14–17, 2021, ESSENDON, VIC, Australia

ACM ISBN 978-1-4503-9115-3/21/12…$15.00
https://doi.org/10.1145/3486622.3493953

[1] https://help.twitter.com/en/rules-and-policies/platform-manipulation
[2] https://help.twitter.com/en/rules-and-policies/election-integrity-policy
[3] https://www.cigionline.org/articles/how-bjp-used-technology-secure-modis-second-win/

elections, collusive behaviour by users can be a serious threat to the integrity of a platform [27]. Identifying and moderating such malicious accounts thus becomes critical for these social media giants.

Accounts which are found to be participating in malicious engagement with posts with an intent to artificially boost their reach can be suspended from Twitter. The presence of such accounts could constitute large proportions of engagement with a politician's account [33]. Along with suspended accounts, accounts that are ultimately deleted from the platform can also pose a threat. Media has reported the spread of misinformative and deceptive content by fraudulent accounts that end up being deleted from the platform [38]. Accounts that end up being suspended or deleted can form significant portions of the aforementioned collusive groups [33], and thus identifying such accounts becomes very important, especially in the context of democratic processes like elections.

During the Indian General Elections of 2019, we observed three major classes of users. 1) Regular users, 2) Suspended users and 3) Deleted users. The class *suspended* consists of those users who were identified to be violating the Twitter policy and were suspended by the Twitter platform. The class *deleted* consists of users who were once active on Twitter but whose accounts were later deleted. *Regular* user accounts are Twitter users who were neither suspended nor deleted from the platform.

In the real world, any given OSN itself does not have information on the motivation behind a user's behaviour. The only information such platforms have regarding the users is the engagement of the users with the platform. To distinguish between the different classes of users, we make use of this same information available publicly. Given that Twitter is only able to observe user activity but not the ultimate outcome (here, suspension or deletion of the account), we design a set of features which are class independent. We then use these features to train models to distinguish between different classes of users.

User interactions in OSNs often emulate trees with users forming communities [6, 16, 17]. The communities of users which participate in artificial engagement can have deep hierarchies.[3] Capturing such extreme hierarchies in the traditional Euclidean space is hard and inefficient [17]. So, for our work we use hyperbolic manifolds [34] which are geometric structures with a negative curvature. This representation space can also be perceived as a continuous version of a tree with area increasing as we move away from the origin (point of observation) of the plane. This nature of geometry is useful to capture the tree-like user communities [5].

We use Hyperbolic Hierarchical Clustering - *HypHC* as the hyperbolic representation technique to capture communities and reduce dimensionality [5]. HypHC provides the inherent advantage of community detection with information encoding. The reduced dimensions in the hyperbolic manifold ensures a computationally efficient downstream task. Additionally, representations obtained using HypHC can be treated just like Euclidean embeddings. We reduce the dimensionality of our features by a factor of ten without compromising on performance by leveraging hyperbolic manifolds via HypHC, all the while keeping every other component in our classification pipeline unchanged from its original Euclidean implementation.

The contributions of our work are as follows:

(1) A feature engineering framework that can help OSNs engineer efficient representations which capture the interaction patterns with top handles.
(2) Demonstrating the application of hyperbolic manifolds on real world social media data to reduce the computational and space complexity while effectively separating the regular, suspended and deleted accounts.

2 Related Work

Our work covers two major domains: the study around classification of accounts on Twitter; and hyperbolic manifolds and their applications.

2.1 Classification of Users in OSNs

Earlier works that studied spam on Twitter [29] leveraged user characteristics such as number of followers, and tweet content to generate features. These features were used to train a random forest classifier to detect spamming accounts. Follow-up works [28] aimed to identify malicious accounts created in a short period of time by using account names. They compare algorithm-made names with man-made names by clustering accounts sharing similar name-based features. The work by Wei et al. [42] uses temporal sentiment analysis to differentiate suspended users from non-suspended users with the help of statistical techniques like Naive Bayes classifier and SVM.

With claims that Twitter had been influencing voter sentiment during the U.S. elections, there have been focused attempts at characterizing Twitter's part in democratic processes in many nations [13, 25, 35]. Notable studies on characterizing users based on Twitter's moderation decisions [27] show that the malicious communities suspended by Twitter exhibit a considerable difference from regular accounts. This class of works combines Twitter and elections to understand different user groups on the platform, spread of sentiment and news on the network and echo chambers and their effects, to minimize the effect Twitter has on democratic processes. In other works [9], authors group Twitter users into communities based on their retweet and mention networks and analyze different characteristics such as popular tweeters, domains, and hashtags. They found that malicious and regular accounts participate in communities which exhibit significant differences in terms of popular account and hashtag usage.

Work has also been done on identifying and characterizing deleted users on social media platforms. Authors in [2] found that a significant proportion of accounts involved in political discourse on Twitter revolving around the *Brexit* referendum campaign were deleted from the platform. Volkova et al. used profile, network and behavior clues, sentiment and emotion features, text embeddings and topics to detect accounts deleted from Twitter which were active in the context of the Russian-Ukranian crisis [38]. Twitter itself removed over 2 million accounts from the platform which were suspected to be fake. These accounts were allegedly giving misleading follower counts for some users [11].

2.2 Hyperbolic manifolds and applications

Hyperbolic Hierarchical Clustering (HypHC) [5] is a key component of our work. We use this technique to reduce the dimensionality

of our user representations for efficient computation. The following sub-section introduces previous research work on hyperbolic manifolds and their applications.

In the work [32] authors claim that the ability of embeddings to model complex patterns is bounded by the dimensionality of the embedding space, because of which it is impossible to extract embeddings of large graph-structured data without any loss of information. So, to increase the representation capacity of embedding methods, they used hyperbolic spaces. Representations in hyperbolic spaces are capable of effectively capturing the underlying hierarchical structures in data at lower dimensions [14].

Feng et al. [14] leveraged this property of hyperbolic spaces to build a hyperbolic metric emebedding model, which projects location-based check-in data of users from social media into the hyperbolic space to predict the next POI (point of interest) for a user. Wang et al. [40] proposed a hyperbolic geometry representation learning model to link user identities across different social network platforms.

Hyperbolic manifolds have also been used to embed knowledge graphs [41], images [23] and words [37] in far lower dimensions. Question-answering [36] and clustering [30] models have also leveraged hyperbolic manifolds to improve their performance.

3 Data description

In this section we introduce and describe the dataset used and the definitions of the different classes of users in our dataset, i.e. regular, suspended and deleted users.

For our work we use the 'Analysis of General Elections 2019 in India' (AGE2019) dataset [18]. This dataset consists of tweets that span from February 5th 2019 to June 25th 2019. This covers a time period starting from two months prior to the first polling in the elections upto one month after the results of the elections were declared. The tweets are collected by querying the Twitter data collection APIs to retrieve tweets having hashtags related to the 2019 Indian General Elections. A total of 45.6 million tweets made by 2.2 million unique users are collected in the dataset.

The AGE2019 dataset also consists of two lists of user ids - deleted users and suspended users. These are users who were found to be either deleted or suspended from the platform as of June 29th, 2019. A total of 56,927 of these users were identified to be suspended as they returned error code *63* on querying the Twitter API. These users form our *suspended* class. Additionally, 21,083 users returned error code *50*, signifying that these accounts had been deleted from the platform. These users form our *deleted* class. The AGE2019 dataset also shared a list of 100,000 users randomly sampled from the remaining users (neither suspended nor deleted). This set of users forms the *regular* class of users in our study. To generate features to separate the deleted, suspended and regular classes of users, we use the election-related tweets by each user, that are provided as a part of the AGE2019 dataset.

4 Feature Engineering

In this section, we describe our feature engineering process. Past research works have used features derived from the content of the tweets, extensive graphs of interactions between the users and more to extract features like sentiment, emotion, lexical features etc. to distinguish between classes on Twitter [38, 39]. However, these can become complicated to extract given the large amounts of data involved and complicated multi-lingual nature of the tweets. We use features that can easily be extracted based on information captured from the user's profile and tweet activity, and do not delve into the actual content of the tweets. Section 6.1.1 demonstrates the importance of our features by comparing the same with some *user-level features* that past works have used to model the account characteristics. We use this section to explain the motivation and design of our proposed *interaction features* to capture the nature of interactions of each user with top profiles that emerged during the General Elections.

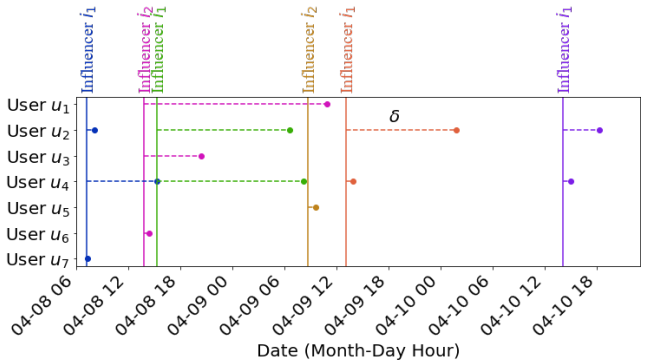


Figure 1: Interaction patterns between the top two Influencers and seven randomly sampled users over a two-day time period. X-axis (bottom) shows the time stamps of the tweets and retweets, Y-axis (left) shows the users. X-axis (top) shows the Influencers. Each vertical line represents a tweet by an Influencer. Each dot in the graph indicates a retweet of an Influencer's tweet by a user, joined by a dashed line to the original tweet. The length of each dashed line represents the time delay δ between the Influencer's original tweet and the user's retweet of that tweet.

4.1 Interaction features

During the 2019 General Elections in India, contesting parties have been known to maintain an *IT cell* [4, 7]. These IT cells, along with a dedicated team of supporters work to push the agenda of their parties as much as possible. As mentioned in [4], these IT cells are known to propagate particular agendas on OSNs by engaging with posts by a certain account. So, artificial engagement in the context of the General Elections revolves around boosting the popularity and pushing the ideologies of a particular leader or party.

To identify the top leaders, we curate a list of users whose content is widely shared across the platform. We call them *Influencers* in our work because of the effect these users have on the content shared on the platorm. We particularly look at the engagement on the platform as a result of their tweets. Influencers need not be political leaders, but in the context of the 2019 General Elections, we observed that most Influencers are political leaders or political party handles. We use these Influencers to generate our interaction features. Table 1 summarizes the notations used henceforth.

Retweets are a great way to quickly engage and amplify the reach of a tweet. Those retweets without any text of their own are just endorsements [10, 12, 24]. We take all retweets from the data and

Notation	Meaning
$I = \{i_1, i_2, \dots\}$	Set of all Influencers
p	Number of Influencers in I
rt_score	Number of users retweeting an Influencer
$T_y = \{t_{y1}, t_{y2}, \dots\}$	Set of all Tweets by Influencer i_y
$U = \{u_1, u_2, \dots\}$	Users engaging with the Influencers in I
R_{xy}	All retweets by user u_x of Influencer i_y
t_{yk}	kth Tweet by Influencer i_y
$r_{xt_{yk}}$	User u_x's retweet of tweet t_{yk}
$\delta_{xt_{yk}}$	Delay in retweeting t_{yk} by user u_x
D_{xy}	Set of all delays $\delta_{xt_{yk}}$
Δ_{xy}	Median of all delays in D_{xy}
n_{xy}	Number of times u_x retweeted i_y
$\vec{V}_x$	Interaction features for u_x

Table 1: Notations used in Sec. 4.1, Algo. 1 and Fig. 2, 3.

curate a list of user profiles I' whose tweets have been retweeted. For each such user in I', we keep track of how many users in the dataset retweeted their tweets (i.e. rt_score). We then consider the top p user profiles having the highest number of retweets as Influencers to form the set I.

The set of users interacting with Influencers in I is given by $U = \{u_1, u_2, \dots\}$ where each user in U has engaged with at least one Influencer in I by retweeting at least one of their tweets. For each Influencer, we define $T_y = \{t_{y1}, t_{y2} \dots\}$ to be the set of all tweets by Influencer i_y. For each pair (u_x, i_y) where user $u_x \in U$ and Influencer $i_y \in I$, we look at those tweets by user u_x which are retweets of any tweet by Influencer i_y, i.e. retweet of any tweet in T_y. We thus have a set of retweets $R_{xy} = \{\text{set of all } r_{xt_{yk}}\}$ where $r_{xt_{yk}}$ is a retweet of Influencer i_y's tweet $t_{yk} \in T_y$ by user u_x. If a user u_x has never retweeted a tweet by Influencer i_y, then $R_{xy} = \{\}$. In order to quantify a particular user's (u_x) interaction with any Influencer (i_y), we use two values:

(1) The first value we use is the delay, or time lag in retweeting. For each retweet $r_{xt_{yk}}$ in R_{xy}, we define the corresponding delay $\delta_{xt_{yk}}$ to be the absolute value of difference in seconds between time of the original tweet (t_{yk}) and the time of retweet of that tweet ($r_{xt_{yk}}$), i.e. $\delta_{xt_{yk}}$ = (time of retweet $r_{xt_{yk}}$ - time of original tweet t_{yk}). Figure 1 shows an illustrative demonstration of how each such δ is calculated. Thus, for each pair (u_x, i_y) we have a set D_{xy} which is the set of all $\delta_{xt_{yk}}$. D_{xy} forms the set of delays for each retweet by user u_x of Influencer i_y. We take the median of the delays in D_{xy} to get the final delay Δ_{xy}.

(2) The second value that we use is the number of times the user has retweeted that Influencer's tweets. We define n_{xy} to be the number of elements in R_{xy}, i.e. the number of times user u_x retweeted a tweet by Influencer i_y.

Thus for each pair (u_x, i_y), we have a two-element vector $\vec{v}_{xy}$ which has two values: delay Δ_{xy} and the number of retweets n_{xy}. We form the interaction feature vector $\vec{V}_x$ for each user u_x by concatenating all such vectors $\vec{v}_{xy}$ obtained for each corresponding Influencer i_y where $i_y \in I$. In case a user has never interacted with an Influencer (i.e. $R_{xy} = \{\}$), we set Δ_{xy} to a large negative value, and n_{xy} to 0. We choose a large negative Δ_{xy} in this case because such a value would never appear if R_{xy} was not empty, thus achieving a good

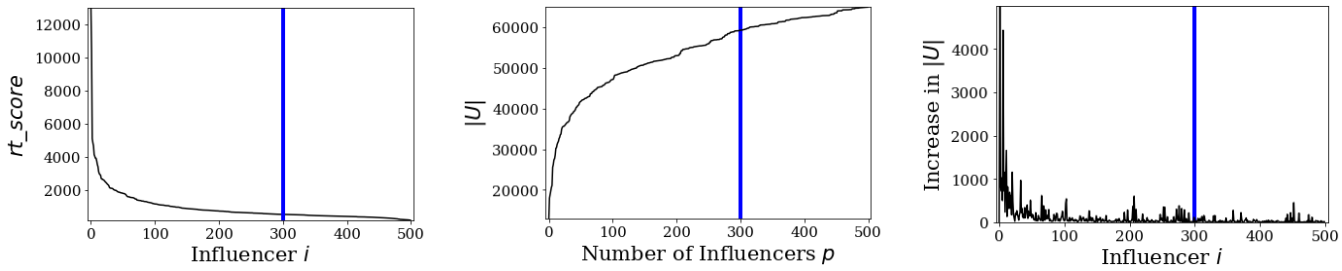
Data: AGE2019: $Tweets$ (all tweets) and $Users$ (all users)
Result: Interaction features of all users

potential Influencers $I' \leftarrow \{\}$;
foreach *tweet t in Tweets* **do**
  if *tweet t is a retweet* **then**
    if *original poster op of t* $\notin I'$ **then**
      $I' \leftarrow I' + op$;
    end
  end
end
foreach *potential Influencer i in* I' **do**
  i's $rt\_score \leftarrow 0$;
  foreach *user u in Users* **do**
    if *user u has retweeted any tweet by i* **then**
      i's $rt\_score \leftarrow i$'s $rt\_score + 1$;
    end
  end
end

sort Influencers I' by their rt_score;
take top p Influencers with highest rt_score as I;
$U \leftarrow \{\}$;
foreach *user u in Users* **do**
  if *user u has retweeted the tweet of anyone in I* **then**
    $U \leftarrow U + u$;
  end
end

foreach *user* u_x *in* U **do**
  Feature vector $\vec{V}_x \leftarrow \{\}$;
  foreach *Influencer* i_y *in Influencers I* **do**
    $R_{xy} \leftarrow \{\}$; $D_{xy} \leftarrow \{\}$;
    let T_y be the set of tweets by Influencer i_y;
    foreach *tweet* t_{yk} *in* T_y **do**
      if *user* u_x *has retweeted tweet* t_{yk} **then**
        let $r_{xt_{yk}}$ be user u_x's retweet of t_{yk};
        $\delta_{xt_{yk}}$ = time of $r_{xt_{yk}}$ - time of t_{yk};
        $R_{xy} \leftarrow R_{xy} + r_{xt_{yk}}$;
        $D_{xy} \leftarrow D_{xy} + \delta_{xt_{yk}}$;
      end
    end
    Δ_{xy} = median of elements in D_{xy};
    n_{xy} = number of elements in R_{xy};
    $\vec{v}_{xy} \leftarrow \{\Delta_{xy}, n_{xy}\}$;
    $\vec{V}_x \leftarrow \vec{V}_x + \vec{v}_{xy}$
  end
end

Algorithm 1: Interaction features engineering. We first identify the Influencers I, then obtain the set of users U who have interacted with them. We calculate the median delays and number of retweets of each user-Influencer pair to form the final feature vector.



(a) (b) (c)

Figure 2: Retweeting trends among users for the top 500 Influencers sorted in decreasing order of their rt_score. Figure (a) depicts the number of unique users retweeting a particular Influencer i, i.e. rt_score. Figure (b) is the number of unique users retweeting any of the top p Influencers, i.e. the size of U for a given size p of I. Figure (c) shows the increase in $|U|$ for each new Influencer i included in I. The vertical line in each figure represents our final choice of p which is 300.

separation in the feature space. Algorithm 1 describes the above explained feature engineering process in a pseudo-code format.

With the interaction feature vector $\vec{V}_x$, we quantify each user's interaction with the top handles. If there is a group of users colluding to interact with an Influencer account, their interaction feature vectors would look similar. Thus our features will help to capture such groups of collusive accounts.

In order to generate interaction features we have to choose the number of Influencers p to include in the set I. We observed the trends of number of unique users added to the set U for each new Influencer added to the set I. We wish to choose p high enough that a large number of users are incorporated. A larger set of users means different subgroups of these users interact with the tweets of different Influencers, thus capturing a larger number of collusive groups. However, at the same time we do not wish to have a large p because we only want the top tweeting Influencers, i.e. the most popular ones, who have a relatively large number of users retweeting their tweets. In order to achieve this balance, we study the relationship between the number of Influencers and number of users as depicted in Figure 2. In Figure 2a, each point on the x-axis represents a particular Influencer, and the y-axis depicts the number of unique users retweeting that particular Influencer (i.e. rt_score). This indicates how many unique users are interacting with each Influencer. Higher the value, more popular the Influencer.

To get a sense of the diversity of users incorporated by choosing a particular p, we plot Figure 2b. In Figure 2b, the x-axis represents the number of Influencers p, and the y-axis represents the size of U (or $|U|$) for a particular value of p. It is important to note that Figure 2b is not merely a cumulative plot of Figure 2a because there will be users who retweet the tweets of multiple Influencers. This graph is increasing, because as we add more Influencers to the set I, we incorporate more users in the set U. The amount of increase in the y-value of the graph as the x-value changes from p to $p+1$ is the number of users added in U when we increase the number of Influencers by one. To better visualise the effect of increase in p on $|U|$ we plot Figure 2c.

Each value in Figure 2.c indicates the *additional number of users* that are included for each new Influencer added to I. We reiterate that there will be users who retweet the tweets of multiple Influencers, and thus Figure 2c is different from Figure 2a. By increasing the value of our final chosen p, we include more spikes from Figure 2c, which means we incorporate a more diverse set of users in U. In Figure 2c, we observe that after around 300 Influencers, the *additional number of users* that are included for each new Influencer added reduces. We hypothesize this to be the optimal value of p. To verify the validity of this hypothesis, we chose values of p at intervals of 50 between 100 and 600 and found best results on our classifiers (the same classifiers described in Section 6.1) with p=300. This confirmed our hypothesis. By choosing $p = 300$ (as depicted with a vertical line in Figure 2), we end up with U containing 59,260 users. The class distribution of our final U is depicted in Table 2. Note that we refer to our proposed user interaction features as F henceforth.

Class	Number of Users
Deleted	8,078
Regular	32,386
Suspended	18,796
Total	59,260

Table 2: Class distribution of users when p=300 Influencers.

5 Dimensionality reduction using Hyperbolic Hierarchical Clustering

Hyperbolic Hierarchical Clustering (HypHC) is a similarity based clustering method [5]. First, a binary tree with n leaves is constructed. Each leaf node i denotes a Twitter user that needs encoding to a lower dimension. From these leaf nodes, intermediate nodes that connect closer nodes are formed. If two leaf nodes are found to potentially belong to the same cluster, they have a least common ancestor (LCA). Each sub-tree denotes a potential cluster. The goal of HypHC is to cluster nodes in such a way that the pair-wise similarity of the data is captured and used to form clusters. The binary tree is built such that the pair-wise similarity $sim_{i,j}$ between each pair of nodes i, j is preserved. Once the binary tree is created, the Dasgupta cost C_D is calculated on the tree. A good tree with distinct clusters is characterised by a low cost C_D. Minimizing the Dasgupta cost merges similar nodes in the hierarchy, resulting in a tree with nodes clustered into appropriate communities. If T

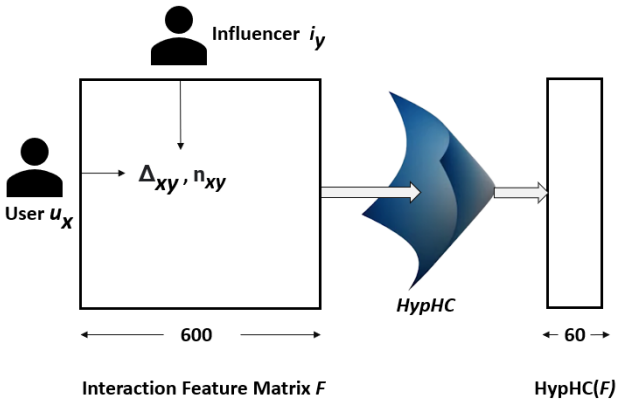


Figure 3: Overall framework: The x^{th} row in F represents the interaction features $\vec{V}_x$ for a user u_x. The columns represent the Influencers. Each user-Influencer pair (u_x, i_y) yields two features: delay Δ_{xy} and number of retweets n_{xy}. Thus, a total of 300 Influencers results in 600 features for each user. The dimensionality of F is reduced to 60 using $HypHC()$.

is a binary tree, and i, j is a pair of nodes with similarity $sim_{i,j}$ between the two nodes, HypHC minimizes the Dasgupta cost C_D amongst all possible binary trees as shown in the equation:

$$T^* = \min_{\forall T, (i,j) \in T} C_D(T, sim_{i,j})$$

With the objective function in place, HypHC minimizes the cost C_D via a continuous constrained optimization problem. A continuous tree representation is built with the help of leaf nodes which are initialized with random embeddings. The leaf nodes should ultimately contain enough information to recover the full tree. All the nodes are pushed towards the boundary of a Poincaré disk. A Poincaré is the hyperbolic geometric model HypHC uses. It has a negative curvature of -1. The curvature dictates how the geometry differs from a Euclidean plane. Negative curvature makes hyperbolic manifolds behave like continuous trees. In a hyperbolic manifold, under the Poincaré model, the distance between two points is defined by a *geodesic* [5].

The shortest path between any two nodes must pass through their least common ancestor (LCA), which in turn aids in constructing the whole binary tree from just the boundary nodes on the Poincaré disk. The HypHC algorithm gives us the binary tree with minimum Dasgupta cost. From this binary tree we extract embeddings of the leaf nodes, which are our final user embeddings. Figure 3 shows the application of HypHC in our work. We feed the interaction features F to HypHC. After reducing the dimensionality of the 600 dimensional interaction features with HypHC, we get a 60 dimensional vector for each user.

6 Results

In this section we present our experiments to evaluate the effectiveness of our features in segregating the three classes, and analysis of the same. We also evaluate the effectiveness of HypHC as a dimensionality reduction technique.

6.1 Classifier Results

While comparing the different classes of users, we trained each model to classify the users in a one vs two fashion by training for *suspended vs (regular + deleted), regular vs (deleted + suspended)* and *deleted vs (regular + suspended)* separations. We also trained each model to classify the users in a one vs one (i.e. binary) fashion by training for *suspended vs deleted*, *suspended vs regular* and *deleted vs regular* separations.

We use two types of classifier models: deep learning based and tree based. For the deep learning based classifiers, we use a deep neural network and an LSTM. The results are presented after appropriate hyperparameter tuning in the loss function, activation function, the optimizer used, number of layers and the number of epochs. For the tree based classifiers, we use lightGBM (LGBM), XGBoost (XGB), Gradient Boosting Classifier (GBC) and the Random Forest Classifier (RFC). We use Grid Search to find the best set of hyperparameters for the tree based models. To account for class imbalance, we balance our training dataset using SMOTE [8].

6.1.1 Comparison with standard feature engineering processes: To evaluate the performance of our proposed features, we calculate 13 additional features for each user which are total number of tweets, number of tweets that are retweets, number of friends, number of followers, total likes, friends to follower ratio, time since account creation, lengths of screen name and bio (in characters and words) and average length of the tweet (in characters and words). We refer to these as *user-level features*. These features have been used in previous works to distinguish between deleted, suspended and regular users on Twitter [38, 39]. We do not use psycholinguistic features for comparison for reasons discussed in Section 4.

To establish the benefit of our proposed features, we use various feature sets, two of which are:

(1) **U**: 13 dimensional user-level features as described above.
(2) **U+F**: 613 dimensional features formed by appending the user-level features (U) to the 600 dimensional interaction features (F).

Table 3 clearly shows that there is an increase in F1 scores in all the cases when the user-level features are appended with the interaction features (compare columns U and $U + F$), showing that our features help the various models to achieve better separation.

6.1.2 Comparison with other dimensionality reduction techniques: To establish our choice of HypHC as a dimensionality reduction technique, we compared its performance with popular unsupervised dimensionality reduction methods like Principal Component Analysis (PCA), t-distributed Stochastic Neighbor Embedding (t-SNE), Spectral Embedding (SE) and Feature Agglomeration (FA). Spectral Embedding and t-SNE are dimensionality reduction techniques based on manifold learning. Feature Agglomeration applies hierarchical clustering.

We reduced the 600 dimensional features (F) to 30, 60, 80 and 100 dimensions using HypHC, PCA, t-SNE, SE and FA, and appended the 13 dimensional user-level features (U). The results observed after reducing to the dimensions mentioned above followed the same pattern – which was that HypHC outperformed all the reduction techniques. For the sake of brevity, we only present and discuss the results obtained after reducing the dimensions to 60 (which gave

Model	Deleted vs (Suspended + Regular)					Suspended vs (Regular + Deleted)					Regular vs (Deleted + Suspended)				
	U	U+F	HypHC	SE	FA	U	U+F	HypHC	SE	FA	U	U+F	HypHC	SE	FA
RFC	90.03	92.25	**93.84**	90.29	87.40	85.82	**89.01**	87.50	84.23	84.16	77.31	**79.69**	79.03	76.34	74.53
LGBM	90.97	91.14	**92.73**	90.23	87.77	86.68	**88.49**	87.91	85.77	83.53	78.18	**79.43**	79.31	76.88	75.62
XGB	87.17	88.49	**89.16**	86.52	84.40	83.86	82.91	84.37	82.34	**83.40**	76.61	76.87	**78.50**	75.23	73.68
GBC	87.16	88.46	**89.04**	86.12	84.99	83.57	82.90	**83.88**	83.18	83.81	76.52	76.88	**78.58**	74.98	74.02
DNN	72.43	85.46	**88.42**	87.56	77.72	81.20	**88.03**	84.17	82.43	81.24	75.63	**81.57**	78.72	74.07	73.81
LSTM	80.33	89.40	**92.27**	87.13	76.39	67.69	**81.98**	77.81	76.94	74.97	61.44	68.21	71.38	**76.38**	74.85

(a) Results of one-vs-two class classifiers

Model	Deleted vs Suspended					Suspended vs Regular					Regular vs Deleted				
	U	U+F	HypHC	SE	FA	U	U+F	HypHC	SE	FA	U	U+F	HypHC	SE	FA
RFC	83.34	**86.42**	85.34	85.14	81.98	81.51	**87.29**	86.07	83.33	82.55	85.50	**88.52**	88.01	81.93	83.61
LGBM	83.64	**85.80**	84.89	84.20	81.87	85.78	**87.27**	87.25	82.34	83.33	87.20	88.01	**88.91**	83.25	84.45
XGB	81.50	**82.35**	82.27	80.03	79.11	82.84	82.37	**84.06**	81.39	79.45	84.77	85.71	**86.02**	80.89	81.93
GBC	81.22	82.33	**82.76**	81.85	80.87	83.21	82.80	**84.31**	81.43	79.85	84.56	**84.70**	84.37	82.15	81.64
DNN	78.15	**84.46**	83.81	81.39	80.23	81.70	**85.13**	83.70	82.42	81.05	73.36	81.67	**86.59**	72.58	74.29
LSTM	76.99	**82.73**	81.17	78.92	78.52	81.20	**85.87**	83.28	80.23	81.48	69.76	80.72	**86.38**	74.29	71.48

(b) Results of one-vs-one class classifiers

Table 3: F1 scores obtained on the classifiers and feature sets as described in Section 6.1. For each of the class separation configurations, we bold the best obtained F1 score for each classifier. Observe how across all classifiers and separations, appending our proposed features to the user-level features (*U+F*) leads to better results than just the user-level features (*U*). Also observe that in almost all cases HypHC performs better than SE and FA. The F1 scores are scaled out of 100.

the highest F1 scores overall) and with the Spectral Embedding and Feature Agglomeration techniques, which performed the best out of our comparison techniques.

We thus obtain our next set of features:

(1) **HypHC features**: 73 dimensional features formed by reducing the 600 dimensional features (F) to 60 dimensions using HypHC and appending the 13 dimensional user-level features (U).
(2) **SE features**: 73 dimensional features formed by reducing the 600 dimensional features (F) to 60 dimensions using SE and appending the 13 dimensional user-level features (U).
(3) **FA features**: 73 dimensional features formed by reducing the 600 dimensional features (F) to 60 dimensions using FA and appending the 13 dimensional user-level features (U).

HypHC, as mentioned in Section 5, takes advantage of hierarchical community detection to perform unsupervised dimensionality reduction to better separate classes. We evaluate the effectiveness of HypHC using two methods. First, we compare the performance of *HypHC* features with the original $U + F$ features. We find that in most cases, *HypHC* features perform at par with the $U + F$ features. Moreover, there are cases where *HypHC* features perform better than the $U + F$ features (compare columns $U + F$ and *HypHC*). This shows that HypHC is an effective dimensionality reduction technique that is able to preserve the separation between classes even at a much lower dimension.

Second, we compare the performance of HypHC with established unsupervised dimensionality reduction techniques like SE and FA. Barring a few cases in rows 3 and 6 in Table 3a, we find that the *HypHC* features outperform both *SE* featuresand *FA* features.

	HypHC	SE	FA
Deleted vs Suspended	10.9165	3.1636	2.9524
Suspended vs Regular	10.9197	3.4279	2.1796
Regular vs Deleted	10.9175	3.9857	3.2614

Table 4: Inter-class cosine distances obtained after reducing dimensions of the *F* features to 60 dimensions with HypHC, SE and FA.

This shows that HypHC is the superior dimensionality reduction technique.

6.2 Interclass Distances

To further evaluate the performance of HypHC, we compare the distances between the centers of the three classes after performing dimensionality reduction of the interaction features F using HypHC, SE and FA. We take each of these three representations, and standardize each feature by removing the mean and scaling to unit variance. Table 4 shows the cosine distances between the centers of each of the three classes in the 60 dimensional space that was generated by each of the dimensionality reduction methods. It is immediately obvious that HypHC is able to achieve the highest and most uniform separation between the classes. This ratifies our claim that HypHC is able to obtain the best separation between the classes.

Our observations from Sections 6.1 and 6.2 show the effectiveness of our interaction features, and the valuable advantage of using HypHC in our pipeline. On comparing the results of all the classifiers for the dimensionality reduced (*HypHC*) and high dimensional ($U + F$) features, we can see that the HypHC features often outperform the original features despite being at a much lower dimension. The added bonus of using lower dimensional data is decreased storage space and lower computation cost and time.

7 Conclusion

Ensuring that interactions between politicians and voters remain organic is critical to the fair functioning of any OSN, especially during democratic processes like elections. To do so, we capture these interaction patterns through our designed features. These interaction features are able to distinguish between the three classes effectively. To ensure that the model can run efficiently and take up as little space as possible, it is important to reduce the dimensionality of the features. To this end, we leverage HypHC, a novel unsupervised dimensionality reduction technique. We show that HypHC performs better than other established dimensionality reduction techniques at separating the classes. Since our interaction features are OSN-agnostic, we plan to carry out these same analyses on other platforms.

References

[1] Saifuddin Ahmed, Kokil Jaidka, and Jaeho Cho. 2016. The 2014 Indian elections on Twitter: A comparison of campaign strategies of political parties. *Telematics and Informatics* 33, 4 (Nov. 2016), 1071–1087. https://doi.org/10.1016/j.tele.2016.03.002

[2] Marco Bastos. 2021. This Account Doesn't Exist: Tweet Decay and the Politics of Deletion in the Brexit Debate. *American Behavioral Scientist* 65, 5 (Jan. 2021), 757–773. https://doi.org/10.1177/0002764221989772

[3] Shelley Boulianne and Anders Olof Larsson. 2021. Engagement with candidate posts on Twitter, Instagram, and Facebook during the 2019 election. *New Media & Society* (April 2021), 146144482110095. https://doi.org/10.1177/14614448211009504

[4] Ualan Campbell-Smith and Samantha Bradshaw. 2019. Global cyber troops country profile: India. Oxford Internet Institute.

[5] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. 2020. From Trees to Continuous Embeddings and Back: Hyperbolic Hierarchical Clustering. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 15065–15076. https://proceedings.neurips.cc/paper/2020/file/ac10ec1ace51b2d973cd87973a98d3ab-Paper.pdf

[6] Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic graph convolutional neural networks. *Advances in neural information processing systems* 32 (2019), 4869.

[7] Aditi Chattopadhyay. 2020. Killing Democracy Tweet By Tweet: How IT Cells Of Political Parties Wage Propaganda War. https://thelogicalindian.com/exclusive/bjp-it-cell-amit-malviya-congress-it-cell-seed-accounts-social-media-twitter-19712 Accessed: 21-03-2021.

[8] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. 2002. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of Artificial Intelligence Research* 16 (Jun 2002), 321–357. https://doi.org/10.1613/jair.953

[9] Farhan Asif Chowdhury et al. 2021. Examining Factors Associated with Twitter Account Suspension Following the 2020 U.S. Presidential Election. *CoRR* abs/2101.09575 (2021). https://dblp.org/rec/journals/corr/abs-2101-09575.bib

[10] B Colin Cork and Terry Eddy. 2017. The retweet as a function of electronic word-of-mouth marketing: A study of athlete endorsement activity on Twitter. *International Journal of Sport Communication* 10, 1 (2017), 1–16.

[11] Paresh Dave. 2018. Twitter cuts suspect users from follower counts again, blames bug. https://theacademicdesigner.com/2020/likes-and-retweets-are-endorsements/ Accessed: 01-11-2021.

[12] The Academic Designer. 2021. Likes and Retweets Are Endorsements on Social Media. https://theacademicdesigner.com/2020/likes-and-retweets-are-endorsements/ Accessed: 20-02-2021.

[13] Wilberforce S Dzisah. 2018. Social media and elections in Ghana: Enhancing democratic participation. *African Journalism Studies* 39, 1 (2018), 27–47.

[14] Shanshan Feng, Lucas Vinh Tran, Gao Cong, Lisi Chen, Jing Li, and Fan Li. 2020. Hme: A hyperbolic metric embedding approach for next-poi recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1429–1438.

[15] Claudia Flores-Saviaga, Brian Keegan, and Saiph Savage. 2018. Mobilizing the Trump Train: Understanding Collective Action in a Political Trolling Community. In *ICWSM*.

[16] Octavian Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic entailment cones for learning hierarchical embeddings. In *International Conference on Machine Learning*. PMLR, 1646–1655.

[17] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. 2018. Hyperbolic neural networks. (2018).

[18] Saurabh Gupta, Agarwal Anant, Suryatej Reddy Vyalla, Arun Balaji Buduru, and Ponnurangam Kumaraguru. 2020. #IVoted to #IGotPwned: Studying Voter Privacy Leaks in Indian Lok Sabha Elections on Twitter. (2020).

[19] Saurabh Gupta, Asmit Kumar Singh, Arun Balaji Buduru, and Ponnurangam Kumaraguru. 2020. Hashtags Are (Not) Judgemental: The Untold Story of Lok Sabha Elections 2019. 216–220. https://doi.org/10.1109/BigMM50055.2020.00038

[20] James Katz, Anshul Jain, and Michael Barris. 2013. *The Social Media President: Barack Obama and the Politics of Digital Engagement.*

[21] Tobias R. Keller and Katharina Kleinen von Königslöw. 2018. Followers, Spread the Message! Predicting the Success of Swiss Politicians on Facebook and Twitter. *Social Media + Society* 4, 1 (Jan. 2018), 205630511876573. https://doi.org/10.1177/2056305118765733

[22] Asif Khan et al. 2020. Predicting Politician's Supporters' Network on Twitter Using Social Network Analysis and Semantic Analysis. *Scientific Programming* 2020 (09 2020), 1–17. https://doi.org/10.1155/2020/9353120

[23] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. 2020. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6418–6428.

[24] Jihie Kim and Jaebong Yoo. 2012. Role of sentiment in message propagation: Reply vs. retweet behavior in political communication. In *2012 International Conference on Social Informatics*. IEEE, 131–136.

[25] Megan Knight. 2012. Journalism as usual: The use of social media as a newsgathering tool in the coverage of the Iranian elections in 2009. *Journal of Media Practice* 13, 1 (2012), 61–74.

[26] Ponnurangam Kumaraguru et al. 2021. Capitol (Pat) riots: A comparative study of Twitter and Parler. *arXiv preprint arXiv:2101.06914* (2021).

[27] Huyen Le, GR Boynton, Zubair Shafiq, and Padmini Srinivasan. 2019. A postmortem of suspended Twitter accounts in the 2016 US presidential electiong. In *2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, 258–265.

[28] Sangho Lee and Jong Kim. 2014. Early filtering of ephemeral malicious accounts on Twitter. *Computer Communications* 54 (2014), 48–57.

[29] Michael Mccord and M Chuah. 2011. Spam detection on twitter using traditional classifiers. In *international conference on Autonomic and trusted computing*. Springer, 175–186.

[30] Nicholas Monath, Manzil Zaheer, Daniel Silva, Andrew McCallum, and Amr Ahmed. 2019. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 714–722.

[31] BBC News. 2021. Capitol riots timeline: The evidence presented against Trump. https://www.bbc.com/news/world-us-canada-56004916 Accessed: 20-02-2021.

[32] Maximilian Nickel and Douwe Kiela. 2017. Poincaré Embeddings for Learning Hierarchical Representations. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 6341–6350. http://papers.nips.cc/paper/7213-poincare-embeddings-for-learning-hierarchical-representations.pdf

[33] A. Onuchowska, D. Berndt, and Sagar Samtani. 2019. Rocket ship or Blimp? - Implications of Malicious Accounts removal on Twitter. In *ECIS*.

[34] John G Ratcliffe, S Axler, and KA Ribet. 1994. *Foundations of hyperbolic manifolds*. Vol. 149. Springer.

[35] Kim Strandberg. 2013. A social media revolution or just a case of history repeating itself? The use of social media in the 2011 Finnish parliamentary elections. 15, 8 (Jan. 2013), 1329–1347. https://doi.org/10.1177/1461444812470612

[36] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic representation learning for fast and efficient neural question answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. 583–591.

[37] Alexandru Tifrea, Gary Becigneul, and Octavian-Eugen Ganea. 2019. Poincare Glove: Hyperbolic Word Embeddings. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Ske5r3AqK7

[38] Svitlana Volkova and Eric Bell. 2016. Account Deletion Prediction on RuNet: A Case Study of Suspicious Twitter Accounts Active During the Russian-Ukrainian Crisis. 1–6. https://doi.org/10.18653/v1/W16-0801

[39] Svitlana Volkova and Eric Bell. 2017. Identifying effective signals to predict deleted and suspended accounts on twitter across languages. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 11.

[40] Feiyang Wang, Li Sun, and Zhongbao Zhang. 2020. Hyperbolic User Identity Linkage across Social Networks. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322242

[41] Shen Wang et al. 2020. H2KGAT: Hierarchical Hyperbolic Knowledge Graph Attention Network. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 4952–4962.

[42] Wei Wei, Kenneth Joseph, Huan Liu, and Kathleen M Carley. 2016. Exploring characteristics of suspended users and network stability on Twitter. *Social network analysis and mining* 6, 1 (2016), 1–18.