Gossip-based Behavioral Group Identification in Decentralized OSNs

Naeimeh Laleh, Barbara Carminati, Elena Ferrari, and Sarunas Girdzijauskas

DiSTA, University of Insubria, Italy
Royal Institute of Technology KTH, Sweden
{nlaleh,elena.ferrari,barbara.carminati}@uninsubria.it
sarunasg@kth.se

Abstract. DOSNs are distributed systems providing social networking services that become extremely popular in recent years. In DOSNs, the aim is to give the users control over their data and keeping data locally to enhance privacy. Therefore, identifying behavioral groups of users that share the same behavioral patterns in decentralized OSNs is challenging. In the fully distributed social graph, each user has only one feature vector and these vectors can not move to any central storage or other users in a raw form duo to privacy issues. We use a gossip learning approach where all users are involved with their local estimation of the clustering model and improve their estimations and finally converge to a final clustering model available for all users. In order to evaluate our approach, we implement our algorithm and test it in a real Facebook graph.
Keywords: Decentralized Online Social Network (DOSN), Gossip Learning, Newscast EM, Behavioral Group Identification

1 Introduction

Recently, discovery of meaningful groups of users that share the same behavioral patterns in social networks has become an active research area. Behavioral group identification has many valuable applications. For instance, it can help in improving recommendation systems, it can be used for advertisement purposes, direct marketing, and for risk assessment in online social networks. The key idea in risk assessment is that the more the target user's behavior diverges from those of other similar users, the more the target user is risky [2]. Therefore, risk assessment approaches require to identify similar users that share the same behavioral patterns.

By considering a social network as a graph, each node is depicted as a user, and an edge connecting two users denotes the relationship between them. Here, the main problem is that all users are connected in friend to friend graph, but users that share the same behavioral patterns not necessarily have friendships in the graph. In grouping users we consider the profile and activity information such as age, gender, education, nationality, number of friends, activity level, etc. Furthermore, in investigating the discovery of behavioral groups, we have cast our attention to decentralized online social networks (DOSNs) [5]. In DOSN, there is no central infrastructure and discovery of behavioral groups is more challenging than in the centralized setup. In the fully distributed social graph,

each user can only communicate with his/her direct friends without sending all the private group identification feature values to his/her direct friends in a raw form. In more details, behavioral patterns can be classified into social and individual behavioral patterns. Social behavioral patterns rely on user interactions, while individual behavioral patterns are related to profile information [20]. In this paper, we propose a methodology for identifying both social and individual patterns to group users in DOSNs.

The problem of finding similar users in social networks has been widely studied in the context of community detection. Community detection approaches that are pure link-based, relying on topological structures [6], [7], fail to group users with the same behavioral patterns in that such users might belong to different communities based on their friendship links. Moreover, some of the community detection approaches are content-based that they rely on the analysis of the content generated by each user [21], [18]. The major problem of these approaches is the overhead for building the graph, based on similarity measures, that is not suitable for real-time applications. On the other hand, when each user feature vector includes both discrete and continuous features, the various behavioral patterns may not be obvious by similarity measures and then, this identification can not be made correctly. However, there are some stream-based community detection methods suitable for real-time applications [16], [22]. But, most of these approaches are link-based [17], and they do not consider the personal feature vector of users.

To alleviate the limitations of existing approaches, we propose a fully decentralized clustering algorithm which is capable of clustering distributed information without requiring central control. The selected clustering model requires specific aspects to be considered such as: the final clustering model should maintain a reasonable performance compared to a centralized clustering model and should be robust in that it should not easily fail when some of the users leave the network or do not answer to messages. Also, all users should be able to have the final clustering model at any time after convergence to assign a group for themselves and their direct friends in a local way. Finally, we need to minimize the communication cost by decreasing the number of messages and the size of them as well. These requirements bring us to exploit Newscast EM [13], a probabilistic gossip-based randomized communication clustering approach, originally developed for clustering users in peer-to-peer networks. In Newscast EM, each user initializes a local estimation of the parameters of the clustering model (mean, standard deviation, etc.) and then, contacts a random user from all users in the network, to exchange their parameters estimation and aggregate them by weighted averaging. The choice of random selection is crucial to the wide dissemination of the gossip [13], since, the probability of a user being sampled is proportional to his/her degree [19]. Gossip-based peer-sampling service [10] provides a user with a uniform random subset of all users in the peer-to-peer network. But, the main difference between peer-to-peer and social networks is that in peer-to-peer networks each user can directly communicate with any other user in the network to exchange information. On the contrary, in social networks

each user can just communicate directly with his/her direct friends. Therefore, we use the random-sampling implementation for social networks proposed in [11].

The main contribution of this paper is making Newscast EM to be applicable on top of DOSN and apply it to identify behavioral groups of users. Our goal is to achieve an accuracy comparable to a centralized scheme. The advantages of this distributed behavioral group identification are: 1. the usage of both social and individual patterns of users, 2. feature values of users are never send over the network in a raw form and 3. it has low computation and communication cost. The remainder of this paper is organized as follows. We first explain Newscast EM in Section 2. Then, we propose our gossip-based implementation for behavioral group identification in Section 3. In Section 4, we show the result of the clustering model. Section 5 introduces the related work. Finally, Section 6 concludes the paper.

2 Background

In selecting the clustering technique, we focused on soft clustering (i.e., probabilistic-based clustering). Hard clustering techniques, (e.g., k-means) are not proposing a solution to the problem of clustering discrete or categorical data [4] since they are based on distance metrics. We use EM (Expectation Maximization) algorithm, that is, a probabilistic based clustering method. In particular, EM defines K probability distributions to identify K clusters for all users based on their feature vectors, where each distribution represents the likelihood of those feature vectors to belong to a given cluster. In this way, EM first assigns a set of K *membership probabilities* to each user u based on his/her own feature vector. Then, EM maximizes these likelihoods by learning the parameters of the clustering model in order to assign to each user the cluster with the highest probability.

The main idea of distributed EM algorithm is that each user starts the Expectation-step with a local estimation of the parameters of the clustering model. Then, in the Maximization-step, the algorithm employs a gossip-based protocol to learn a final clustering model from these local estimations. Each user exchanges his/her own estimations with several other users by using a randomized communication protocol. By gathering these estimations from random users, the target user can update and re-estimate his/her own estimation.

In the following, we present a summary of Newscast EM. Interested readers are referred to [13] for more details. Let N be the set of users in the OSN, the probability of membership or weight of a target user $\boldsymbol{u}$, $\boldsymbol{u} \in N$, in cluster l is defined as [4]:

$$w_l(\boldsymbol{u}) = \frac{w_l . p_l(\boldsymbol{u}|\theta_l)}{\sum_{i=1}^{K} w_i . p_i(\boldsymbol{u}|\theta_i)} \tag{1}$$

where, w_l is a weight computed as $w_l = \frac{|N_l|}{|N|}$, with N_l denotes the set of users belonging to the l^{th} cluster, where $\sum_{l=1}^{K} w_l = 1$; function $p_l(\boldsymbol{u}|\theta_l)$ is the component density function modeling the feature vector of the l^{th} cluster, where $\theta_l = \{\boldsymbol{\mu_l}, \Sigma_l\}$ represents the parameters for l^{th} distribution, that is, the mean and the covariance.

Newscast EM uses a fully distributed averaging process for estimating the parameters $\Theta = \{w_l, \boldsymbol{\mu}_l, \Sigma_l\}$, $l = 1, \ldots, K$. Assuming that each user has just one feature vector, then the Expectation-step implies that each user locally estimates the parameters based on his/her own feature vector. In this manner, each user u_i, $i = 1, \ldots, N$ starts with a local estimation of $\Theta_i = \{w_{li}, \boldsymbol{\mu}_{li}, \Sigma_{li}\}$ for the parameters of the cluster l. However, in the Maximization-step of the algorithm user u_i needs information from all users in the network to recompute his/her parameters estimation. Therefore, this step is implemented as a sequence of gossip-based cycles. The details of the algorithm, which each user will run in parallel is as follows:

Initialization phase: We assume that all users agree on the number of clusters K and start the exchanging protocol. Each user $\boldsymbol{u}_i$, sets the membership probability for each cluster as $w_l(\boldsymbol{u_i})$ to some random positive value and then normalizes all to sum to 1 over all l. This phase is completely local for each user.

Maximization-step: In this step, user u_i initializes the local parameters estimation for each cluster l as follows: $w_{li} = w_l(\boldsymbol{u_i})$, $\boldsymbol{\mu}_{li} = \boldsymbol{u_i}$ and $\tilde{\Sigma}_{li} = \boldsymbol{u_i}.\boldsymbol{u_i}^T$, where T is the transpose of the feature vector of user u_i. Then, user u_i for $\Re$ cycles repeatedly initiates the information exchange with random users, i.e., u_j. Then, users u_i and u_j update their local parameters estimation for each cluster l as follows:

$$w_{li}^{'} = w_{lj}^{'} = \frac{w_{li} + w_{lj}}{2} \tag{2}$$

$$\boldsymbol{\mu}_{li}^{'} = \boldsymbol{\mu}_{lj}^{'} = \frac{w_{li}.\boldsymbol{\mu}_{li} + w_{lj}.\boldsymbol{\mu}_{lj}}{w_{li} + w_{lj}} \tag{3}$$

$$\tilde{\Sigma}_{li}^{'} = \tilde{\Sigma}_{lj}^{'} = \frac{w_{li}.\tilde{\Sigma}_{li} + w_{lj}.\tilde{\Sigma}_{lj}}{w_{li} + w_{lj}} \tag{4}$$

Expectation-step: User u_i, after waiting for $\Re$ cycles for the convergence of his/her local parameters estimation, computes new membership probabilities for each cluster l using the Maximization-step estimations w_{li}, $\boldsymbol{\mu}_{li}$ and $\Sigma_{li} = \tilde{\Sigma}_{li} - \boldsymbol{\mu}_{li}.\boldsymbol{\mu}_{li}^T$. We denote with Θ^t the parameter values set at iteration t and then $\Theta^{t+1} = \{(\boldsymbol{\mu}_{li}^{t+1}, \Sigma_{li}^{t+1}, w_{li}^{t+1}), l = 1, \ldots, K\}$. The sequence of Θ-values which is then the likelihood of Θ, $L(\Theta)$, is non-decreasing at each iteration. Then, user u_i checks the stopping tolerance by using the estimations from the previous EM-iteration to see if it is satisfied or not, until $|L(\Theta^t) - L(\Theta^{t+1})| \leq \varepsilon$, where $\varepsilon > 0$. If it is not satisfied, the Maximization-step is repeated, unless a stopping tolerance is satisfied. In the following, we will explain how to run newscast EM in decentralized social networks.

3 Newscast EM in DOSNs

In social networks, each user can just communicate directly with his/her direct friends. Therefore, we propose our gossip-based clustering framework on DOSNs that contains two main components: UserSelection and ClusteringModelUpdate. The same algorithm is run by each user in the network in parallel, as shown in Figure 1.

3.1 User Selection

In randomized user selection, the problem is that if users can be selected randomly with equal probability, the estimation will be unbiased. But, it is well known that the probability of a user being sampled is proportional to his/her degree [19]. Therefore, more popular users have a higher degree and tend to have a higher probability of being sampled. This will lead to overestimate the average value during the gossip process. There are plenty of approaches to select a uniform and unbiased random sampling of users in DOSNs such as: graph traversal techniques [15] and Random Walk [14]. But, most of these approaches are biased towards high degree users [19]. To have a uniform random sample of all users, each user needs to know every other user in the network. Since, accessing each user in the network to gossip with, is unrealistic in a large-scale dynamic networks, we apply a technique for DOSNs proposed [11] for randomized communication, to define a dynamically changing random graph topology over the network. This technique includes two methods *Initialization* and *SelectUser*.

The initialization procedure initializes the service for the new user when he/she joins the social network. First, each user u_i maintains a list of its direct friends and two hops friends in a small fixed size cache, called *Random Neighbors Cache* (RNC), including e entries. The set e of entries in the RNC contains the list of random users ID, their *longevity* field, and the path to reach them. The field *longevity* is the age of the entry since the moment it was created by the user. Then, in the *SelectRandomEntries()* procedure, a user selects S subset of neighbors from RNC and puts them in a cache, called Exchange Cache (ExC). After that, the target user u_i continuously selects one of his/her neighbors with the highest *longevity* from RNC, i.e., u_j, in *SelectRandomUser()* procedure, to exchange entries of ExC. Then, u_i increases by one the *longevity* of all the other entries in u_i's RNC.

In social networks, as shown in Figure 2, if user A wants to communicate with user D, he/she needs to reach first B and then C. Therefore, each user needs to maintain both a set of random users ID and also the path to reach them in order to exchange the entries of ExC. For instance, let us assume that in Figure 2.(a) user A has six entries in the RNC that include C, I, K, D, N and B. User A selects user I with the highest *longevity* among all neighbors in the RNC and wants to exchange S (i.e., 3 in this example) number of his/her neighbors in the ExC, such as C, D, N, with user I. On the other hand, user A needs to pass through (M, L) to reach user I. Therefore, user I, in order to reach all of the entries in the ExC of user A, needs to first reach user A and then he/she will be

able to reach all entries inside A 's ExC. The problem is that this path could be long. For instance, user I, in order to reach user D, first needs to reach user A by passing through (L, M) and then from A to user D via (B, C), i.e., (L, M, A, B, C). This path is long and it is not the shortest path from user I to D. But, user I can reach user D by passing through his/her mutual friends with D, like (E ,D) or (L, D).

More precisely, to decrease the communication cost, the protocol in [11] builds a new path for user I to reach all entries in user A's ExC during the exchanging process, illustrated in procedure *UpdateExC()* in Figure 1. In this way, the source user A asks all users within the direct path from A to I, i.e., (M, L, I) to build a new path for all the entries to be accessible for user I. This process in summary is as followings. First the source user (A) only adds his/her ID to the first part of the address of each entry in ExC and sends the ExC to the next user (M) on the path towards user I, as shown in Figure 2.(1) in red color. Then, every next user (for instance user M) within the path towards user I and also user I him/herself, after receiving the ExC, first reverses the path of each entry in ExC, as shown in Figure 2.(2).a and starts traversing all users inside the reverse path to check, if he/she has any direct friendship or mutual friends with those users. If yes, he/she removes the remaining part of the path and adds his/her friends or mutual friends ID to the path and adds the ID of his/herself to the first part of the path (except user I that does not need to add his/her ID to the first part of the path), as shows in Figure 2.(2).b (the first and second row of ExC). If not, he/she keeps the path and just adds his/her ID to the first part of the path as shown in Figure 2.(2).b (the third row of ExC).

Then, all other users within the path towards user I, i.e., user L, do the same and send the ExC to the next user towards user I. When user I receives the final ExC, checks all entries in the ExC and updates them in his/her own RNC. Then, user I replies by selecting a subset of his/her ExC entries, updates the entries path and, then, sends them to user A from the path (L, M, A).

After exchanging neighbors for a number of cycles, the service will converge to a random overlay where each user connects to a uniform random subset of all users currently in the network. But, in our framework, users in addition of exchanging neighbors, also exchange their parameters estimation of the clustering model. In this way, when the service converges to the random overlay, also converges to a final parameters estimation available for all users and then, users are able to update their local parameters estimation. After some iterations of the EM algorithm, they will converge to a final clustering model. In the next section, we will explain in more details the update of the clustering model.

3.2 Clutering Model Update

The second main component of our framework is the online clustering algorithm that updates the clustering model based on the local parameters estimation of each user. In our setting, in the set e entries of the user u_i's RNC, in addition to the list of random users ID, their *longevity* field, and the path to reach them, we maintain their parameters estimation of the clustering model. Therefore, the

Algorithm Gossip-based Clustering Protocol

```
Input: The local Θ_i for each user u_i, i=1,...,N
Output: The global Θ_i for each user u_i.
Initialization()
Loop:
  if Push then//if u_i has to push information
    Wait △T
    ExC ← RNC.SelectRandomEntries()
    ExC ← UpdateExC(ExC, u_i)
    u_j ← RNC.SelectRandomUser()
    Sends ExC to u_i 's neighbor toward u_j
    Sends Θ_i to u_i 's neighbor toward u_j
    Θ_j ← Receive(u_j)
    Θ_i ← UpdateModel(Θ_i, Θ_j)
  else if Pull then//If u_j has to reply u_i
    ExC ← RNC.SelectRandomEntries()
    Sends ExC to u_j 's neighbor toward u_i
    Sends Θ_j to u_j 's neighbor toward u_i
    Receives ExC from u_j 's neighbor
    ExC ← UpdateExC(ExC, u_j)
    Receives Θ_i from u_j 's neighbor
    Θ_j ← UpdateModel(Θ_i, Θ_j)
  else//Pull user u_k within the path
    ExC ← UpdateExC(ExC, u_k)
    Sends ExC to u_k 's neighbor toward u_j
End Loop
procedure Initialization()
  InitModel()
  InitRNC()
  return Θ, RNC
End procedure

procedure UpdateModel(Θ_i, Θ_j)
  for each cluster l do
    w'_l = (w_li + w_lj)/2
    μ'_l = (w_li.μ_li + w_lj.μ_lj)/(w_li + w_lj)
    Σ̃'_l = (w_li.Σ̃_li + w_lj.Σ̃_lj)/(w_li + w_lj)
  return Θ'
End procedure

procedure UpdateExC(RxC, u)
  UpdatedRxC= RxC
  if u = u_i then
    for all entries path ∈ UpdatedRxC do
      Path.AddFirstID()
  else//(u = u_j) or (u = u_k within the path to u_j)
    for all entries path ∈ UpdatedRxC do
      ReversedPath=path.Reverse()
      for all users ID ∈ ReversedPath do
        if ID ∈ Direct-Friends (u_k) then
          Path.AddFirstID(ID)
          Break
        else if ID ∈ Two-Hop-Friends (u_k) then
          Path.AddFirstID(ID)
          Path.AddFirstID(GetDirectFriend(ID))
          Break
        else
          Path.AddFirstID(ID)
  if u = u_k then
    for all entries ∈ UpdatedRxC do
      Path.AddFirstID(u)
  return UpdatedRxC
End procedure
```

Fig. 1: Gossip-based clustering protocol

gossip-based clustering algorithm shown in Figure 1 performs the following steps. In the initialization phase, each user u_i, in addition to filling *RNC* with direct friends and two hops friends, initializes the local parameters estimation for each cluster l. After the initialization phase, users initiate exchanging neighbors and the parameters estimation of the clustering model simultaneously and periodically at a fixed period $\triangle T$. We do assume that the length of the period $\triangle T$ is the same for all users. During a period $\triangle T$, each user initiates one exchanging cycle. There are two types of communication models for exchanging the information. In the Push based model, a target user u_i sends *ExC* and parameters estimation (Θ_i) to the selected user. In the Push-Pull based model, both the target user u_i and selected user u_j exchange their *ExC* and parameters estimation. Our communication model is based on Push-Pull, since the Push approach can easily lead to partitioning the set of users in the network [10], [11].

After initialization, the initiating user u_i increases by one the *longevity* of all neighbors in his/her *RNC*. After that, user u_i selects neighbor u_j with the highest *longevity* among all neighbors in *RNC*, and set the *longevity* of u_j to zero in his/her *RNC*. If the information has to be pushed, user u_i replaces u_j's entry in *RNC* with a new entry of *longevity* 0 and with u_i's ID and path to reach user u_i. Then, user u_i selects S subset of neighbors from u_i's own *RNC*, and save them in the temporary *ExC*. Next, user u_i updates the entries path in the *ExC* by building a new path as mentioned in procedure *UpdateExC()* in Figure 1, and sends it to the next user in the path towards user u_j. After that, user u_i sends his/her local parameters estimation to the next user u_k in the path towards user u_j. Later, all users in the path towards user u_j update the entries path in the

ExC and send the updated ExC and parameters estimation received from u_i to the next user in the path towards user u_j.

When user u_j receives from one of his/her direct friends the ExC coming from user u_i, user u_j replies by selecting a random subset of S neighbors of his/her own RNC and save them in his/her ExC. Next, user u_j updates the entries path in ExC by adding his/her ID to the first part of the entries path and sends ExC to the next user on the path towards user u_i. After that, user u_j sends the local parameters estimation to the next user in the path towards user u_i. Then, user u_j updates the entries path in the received ExC. Next, u_j discards entries pointing at u_j and entries already contained in u_j's RNC and updates his/her RNC to include all remaining entries, by firstly using empty cache slots, and secondly replacing entries among the ones sent to u_i. User u_j set *longevity* to zero for all entries of received ExC in the RNC and does not increase, though, any entry's *longevity* in the RNC until he/she initiates the exchanging process. After that, user u_j updates his/her own parameters estimation by calculating the weighted average with the parameters estimation coming from u_i. Finally, user u_j updates the parameters estimation of u_i and u_i's ID and the path to reach u_i inside his/her RNC.

Under this algorithm, after some cycles, the local parameters estimation of users converge exponentially fast to the global parameters estimation in each Maximization-step of the EM algorithm. Therefore, each user is able to compute new membership probabilities and check the stopping tolerance. Each user maintains the newly updated parameters estimation from the previous EM-iteration in a small cache, called *Estimation Cache* (EsC), of fixed size. When the cache is full, the parameters estimation stored for the longest time is replaced by the newly added parameters estimation. Therefore, after some iterations of clustering, all users will have a final clustering model and compute a final membership probability to belong to their most fitting cluster.

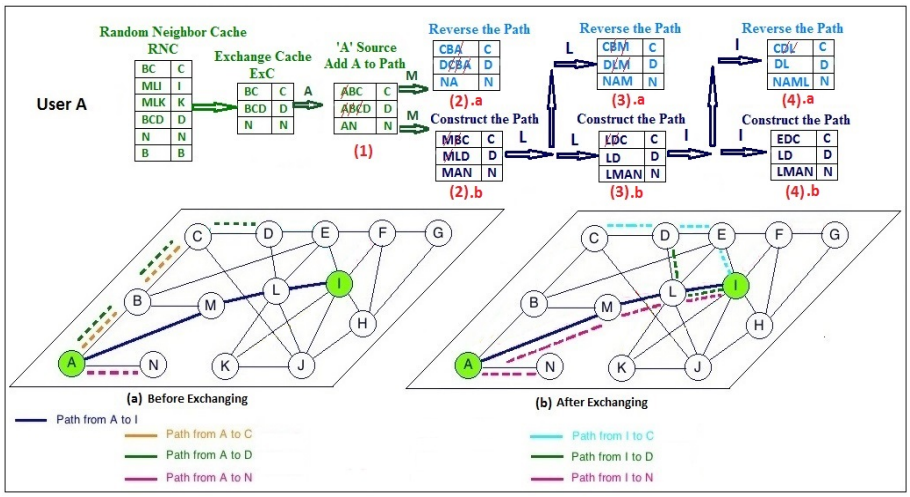


Fig. 2: (a) before and (b) after the neighbours exchanged between A and I

3.3 User Behaviour-based Group Identification in DOSNs

Our goal is to have similar users in each group based on their social and individual features. Feature vectors of each user are given as input to the algorithm in Figure 1. Therefore, each user can assign a cluster number (behavior group) to him/herself based on the feature vector by considering the maximum membership probability among them.

User's features vector includes two types of features: individual and social features. Individual features are those, like age, gender, but also those that impact the possible users' behaviors, like, education and nationality. In addition to individual information, in order to measure users' attitude in online socialization, we consider the following social features:
Number of Friends: social users with a lot of friends have different patterns than isolated users with few friends;
Activity Level: unlike active users that write a lot of posts, passive users do not send any information to others. We calculate this feature as the sum of: a) number of posts that a user sends to others from the first day of joining the community, b) number of likes that a user performs on posted items, and c) number of comments a user writes for posted items;
Percentage of public profile items: the assumption is that users with all profile information (100%) public are more social.

4 Experiments

To perform the experiments on a real graph, we used the Facebook dataset crawled and used in [1]. The author crawled the profiles information and friendship links of 75 users that launched the application as seed. Then, the application crawled the information related to these 75 users' friendships. We removed those profiles that have many missing features and obtained a graph by considering the largest connected component which includes 13,000 user profiles, plus the 75 seed users, with a total of about 461,700 friend links, 6,150,892 likes and 1,742,709 comments. Totally, around 7,000 users have more than 75% profile information as public.

4.1 Results for convergence of the clustering model

The experiments are ran with $S = 50$ [10]. During the exchanging process the mean of the local parameters estimation, $\mu_{i,c}$ of each user u_i, $i = 1, \ldots, N$ for each cluster in cycle c, $c = 1, \ldots, \Re$ is always the global correct mean μ_c, but, the variance measure σ_{c^2} that expresses the deviation of the local estimations from the correct mean in the given cycle c decreases over the set of all local estimations on the average by factor γ, with $\gamma < \frac{1}{2\sqrt{e}}$ [10]. In general, when the variance tends to zero, then all users hold the global correct mean μ_c.

$$\mu_c = \frac{1}{N}\sum_{i=1}^{N}\mu_{i,c} \qquad \text{and} \qquad \sigma_c^2 = \frac{1}{N-1}\sum_{i=1}^{N}(\mu_{i,c} - \mu_c)^2$$

The convergence factor between cycle c and $c+1$ is given by $\sigma^2_{c+1}/\sigma^2_c$. We plot the convergence factor (values are averages for 20 independent runs) as a function of the number of cycles, as shown in Figure 3a. It is clear to see that the speed of the convergence of the variance is fast and it decreases exponentially after few cycles. Thus, means that, after a small number of cycles, all users, including the isolated users with low friendship links, will have accurate estimations of the global correct mean μ in each M-step, when no failures occurred. From this experimental result, choosing the number of cycles $\Re$ equal to 120[1] is sufficient to reach a convergence.

4.2 Coping with User Failure

In a dynamic network, users continuously join and leave the network and they fail in some situations. In this section, we consider the performance of the clustering model when some percentage of users fail in each cycle of the exchanging parameters estimation.

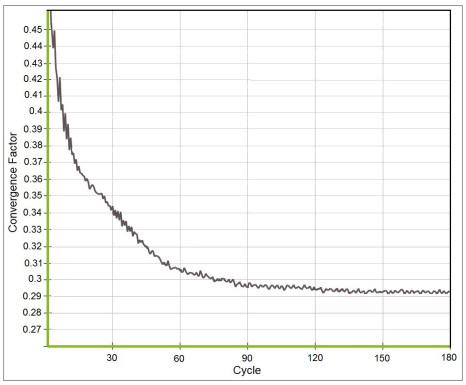
As we mention before, each user maintains the parameters estimation he/she receives from other users in the network in his/her ENC. If a user u_i sends his/her parameters estimation to a user u_j to exchange and performs averaging and, waiting for $\triangle T$ time, he/she does not receive any answer, u_i checks the ENC to verify if he/she has the parameters estimation of user u_j from previous cycles or not. If yes, u_i performs the average with those previous values and updates his/her parameters estimation. Otherwise, he/she skips the exchanging step. We need to mention that the user selection method in [11] takes care of the failure of those users within the path between user u_i and u_j. We consider the effect of these missing exchanges on the final value of the global μ of the clustering model. Towards this goal, we remove 50% of users in each cycle of the exchanging protocol and run independently the Newscast EM for 20 times. We show the result in Figure 3b. The y-axis shows the variance of the 120th cycle to the first cycle. The figure shows that if the failure happens in the first cycles, the result of the global μ of the clustering model is so far from the real correct global μ. But, if this failure happens in the next cycles (especially after the 100th cycle), the variance tends to zero.

4.3 Clustering Results

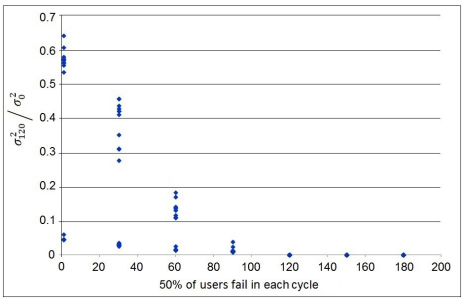
In Table 1a, we can see the performance of randomly initialized Newscast EM and centralized EM, by setting different number of clusters. The result in this table shows the average number of EM iterations for each user to achieve convergence in Newscast EM (around 42 ± 3) and centralized EM (around 38 ± 2) that are almost near. After convergence, the value of μ_i for each cluster at each user will converge to the true global μ. For example, Table 2a shows this value for feature Activity level in centralized EM and Newscast EM that are almost equal. Also, Table 2b shows the values of μ for each cluster for the feature Number of friends.

[1] We assume that all users agree on the number $\Re$ of peer sampling cycles, and $\Re$ is large enough to guarantee convergence to the final parameters estimation.

Fig. 3: The convergence factor and the variance after users failure



(a) Convergence factor



(b) The variance of μ at cycle 120 (for 20 independent run) after the failure of 50% of users in each cycle

Table 1: EM iterations and the ratio of users for different number of clusters

(a) EM iterations

	EM Iteration	
NoClusters	CentralizedEM	NewscastEM
3	38	41.25
5	38	42.53
7	39	41.38
10	38	43.19

(b) The ratio of users (RU) in each cluster

ClusterNo	10	7	5	3
Cluster1	20.27%	32.43%	47.29%	41.89%
Cluster2	20.27%	21.62%	22.97%	32.43%
Cluster3	14.86%	18.91%	10.81%	25.67%
Cluster4	10.81%	9.45%	12.16%	
Cluster5	9.45%	6.75%	6.75%	
Cluster6	6.75%	5.4%		
Cluster7	1.35%	5.4%		
Cluster8	5.4%			
Cluster9	5.4%			
Cluster10	5.4%			

Table 2: The value of μ for two features

(a) Activity level

ClusterNo	*CentralEM*	*NewscastEM*
C1	27.5	27.35
C2	52.53	52.25
C3	48.31	48.05
C4	11301.68	11299.987
C5	6869.46	6868.299
C6	335.83	334.062
C7	18338.18	18336.647

(b) Number of friends

ClusterNo	*CentralEM*	*NewscastEM*
C1	536.39	535.56
C2	272.91	271.47
C3	194.42	193.39
C4	438.35	436.04
C5	276.87	275.41
C6	963	961.93
C7	953.6	951.58

4.4 Dominant User Behaviors

Identifying the number of clusters or user behaviors is related to the nature of the dataset. Therefore, there is no correct or incorrect numbers of groups to find. Each increment in the number l of clusters yields to a new group and similarly a new behavior. On the other hand, if we consider a small number of groups by aggregating some behaviors, we will have the most dominant behaviors, but, we may miss some relevant behaviors. In this paper, we assume that we know the best number of clusters. However, we analyze the quality of the clustering for various values of l. In Table 1b, we report the ratio of users that belong to each

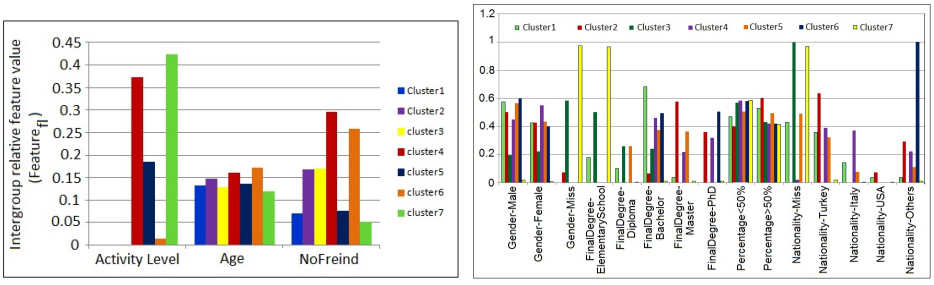
cluster, by considering different numbers of clusters. When we have 10 clusters, the percentage of users that belong to the '7th cluster' is 1%, that is, very small in size. By setting the number of cluster to 5, we will have around 50% of users concentrated in one group (1sh cluster) that will be a huge cluster, and we are not able to precisely identify their behavioral pattern. For the number of clusters equal to 3, we will have the most dominant behaviors, but, we will miss a lot of behavioral patterns. Therefore, we consider the number of clusters equal to 7 in the rest of the experiments.

In the next experiment, we analyze the influence of each feature value on the quality of the discovered groups. Then, among all features, we remove those features that have the same distribution in all clusters. More precisely, we discard those features whose existence in the clustering will not propose a new behavioral group. In order to have a measure for the distribution of each feature value for all clusters, we use the inter-group relative feature value [20], denoted by $Related_{fl}$, which measures how a feature f of cluster l is related to the same feature of the other clusters. It is computed as follows:

$$Related_{fl} = \frac{feature_{fl}}{\sum_{l=1}^{K} feature_{fk}} \tag{5}$$

where $feature_{fl}$ is the value of feature f in cluster l. This allows us to see if the value of a feature is evenly distributed in all clusters or concentrated in a single cluster. For example, we can see in Figure 4a the distribution of all categorical features such as age, number of friends and activity level. This figure shows that the distribution of the feature 'age' in all clusters is nearly the same. Then, we remove this feature since it can not help us to define any behavioral pattern. For discrete features, such as: gender, nationality and percentage of public profile items, we consider the fraction of users that have the same value for that feature in each cluster as shown in Figure 4b. We removed 'gender' and 'percentage of public profile items' and we consider all the remaining features. Based on the experimental results, we are able to find the most dominant be-

Fig. 4: The distribution of features in each cluster



(a) Categorical features (b) Discrete features

haviors, that are shown in Figure 5. These dominant behaviors are categorized

as follows: 1)*most active users with high number of friends:* users in cluster 7 have the highest activity level and the highest number of friends. These users have a bachelor or master degree and all of them are from Italy. 2)*very active users with medium number of friends:* these users are concentrated in cluster 4, have a medium activity level and a medium number of friends. Their education level is either elementary school, bachelor or PhD and they are from all countries except Italy and USA. 3)*active users with low number of friends:* these users are concentrated in cluster 5, and their education level is either diploma, bachelor or master and they are from all countries except USA. 4)*passive users with high number of friends:* these users are concentrated in cluster 6. These users have either a bachelor or PhD degree. They are from all countries except Italy. 5)*very passive users with medium number of friends:* these users are in cluster 1. These users are from all education levels. They are from all countries except Italy. 6)*most passive users with lowest number of friends:* these users are concentrated in cluster 3. Their education level is either bachelor or master and they are from all countries except USA. 7)*most passive users with low number of friends:* these users are in cluster 2 and most of them have bachelor. They are from all countries.

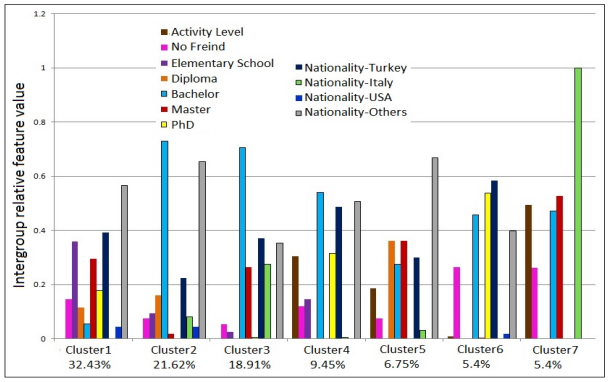


Fig. 5: The most dominant behaviors with percentage of users in each group

5 Related Work

In decentralized setting, there are several distributed clustering algorithms like building and using a single clustering model for each user individually based on his/her local dataset[9]. Other approaches are sharing the data between users [24]. But, the communication cost is high to share all the raw data between users. Another approach is to organize the clustering model in a hierarchical fashion [23], [9] by which local clustering models are computed first for each user individually, and sent to a logically higher-level user that aggregates local models. But, users need to share all their private information that is a big issue in our application as well as in some other applications.

Other related work are in the area of distributed computation in large distributed systems. In deterministic averaging techniques, each user repeatedly selects all his/her immediate neighbours to update the local parameters estimation. For example, authors in [8] proposed an EM algorithm based on this approach. However, this technique is not suitable for social networks, because of the existence of some high degree users. Another kind of technique is probabilistic gossip-based approaches [3], [12] where, at each iteration, each user repeatedly selects a uniform random user and both users compute the average of their parameters estimation. For instance, the newscast model in [12], the gossip-based protocols in [3] and Newscast EM are all based on gossip learning. Since for applying Newscast EM in social networks, the network needs to be fully connected, we use gossip-based peer sampling service on social networks.

6 Conclusion

We identify behavioral groups of users by applying the Newscast EM algorithm on top of DOSNs. We combine the ability to access random users on social networks with distributed clustering model to identify group of behavioral patterns. For future directions we are planning to find a solution for defining the best number of clusters in a distributed manner and also for exchanging information in a secure way for avoiding malicious users manipulating the local parameters estimation of the clustering model. Furthermore, we will use the identified behavioral patterns for risk assessment in DOSNs.

Acknowledgment

This work is supported by the iSocial EU Marie Curie ITN project (FP7-PEOPLE-2012-ITN).

References

1. Cuneyt Gurcan Akcora, Barbara Carminati, and Elena Ferrari. Privacy in social networks how risky is your social graph? In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 9–19. IEEE, 2012.
2. Leman Akoglu, Mary McGlohon, and Christos Faloutsos. Oddball- spotting anomalies in weighted graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 410–421. Springer, 2010.
3. Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 3, pages 1653–1664. IEEE, 2005.
4. Paul S Bradley, Usama Fayyad, and Cory Reina. Scaling em (expectation-maximization) clustering to large databases. Technical report, Technical Report MSR-TR-98-35, Microsoft Research Redmond, 1998.
5. Anwitaman Datta, Sonja Buchegger, Le-Hung Vu, Thorsten Strufe, and Krzysztof Rzadca. Decentralized online social networks. In *Handbook of Social Network Technologies and Applications*, pages 349–378. Springer, 2010.

6. Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. Generalized louvain method for community detection in large networks. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 88–93. IEEE, 2011.
7. Tina Eliassi-Rad, Keith Henderson, Sprios Papadimitriou, and Christos Faloutsos. A hybrid community discovery framework for complex networks. In *SIAM Conference on Data Mining*, 2010.
8. Dongbing Gu. Distributed em algorithm for gaussian mixtures in sensor networks. *Neural Networks, IEEE Transactions on*, 19(7):1154–1166, 2008.
9. Shohei Hido, Seiya Tokui, and Satoshi Oda. Jubatus: An open source platform for distributed online machine learning. In *NIPS 2013 Workshop on Big Learning, Lake Tahoe*.
10. Márk Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten Van Steen. Gossip-based peer sampling. *ACM Transactions on Computer Systems (TOCS)*, 25(3):8, 2007.
11. Mansour Khelghatdoust and Sarunas Girdzijauskas. Short: Gossip-based sampling in social overlays. In *Networked Systems*, pages 335–340. Springer, 2014.
12. Wojtek Kowalczyk, Márk Jelasity, and A Eiben. Towards data mining in large and fully distributed peer-to-peer overlay networks. In *Proceedings of BNAIC'03*, pages 203–210, 2003.
13. Wojtek Kowalczyk and Nikos A Vlassis. Newscast em. In *Advances in neural information processing systems*, pages 713–720, 2004.
14. Maciej Kurant, Minas Gjoka, Carter T Butts, and Athina Markopoulou. Walking on a graph with a magnifying glass: stratified sampling via weighted random walks. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 281–292. ACM, 2011.
15. Maciej Kurant, Athina Markopoulou, and Patrick Thiran. Towards unbiased bfs sampling. *Selected Areas in Communications, IEEE Journal on*, 29(9):1799–1809, 2011.
16. Zhong Li, Cheng Wang, Siqian Yang, Changjun Jiang, and Xiangyang Li. Lass: Local-activity and social-similarity based data forwarding in mobile social networks. *Parallel and Distributed Systems, IEEE Transactions on*, 26(1):174–184, 2015.
17. Yao Liu, Qiao Liu, and Zhiguang Qin. Community detecting and feature analysis in real directed weighted social networks. *Journal of Networks*, 8(6):1432–1439, 2013.
18. Dongyuan Lu, Qiudan Li, and Stephen Shaoyi Liao. A graph-based action network framework to identify prestigious members through member's prestige evolution. *Decision Support Systems*, 53(1):44–54, 2012.
19. Jianguo Lu and Dingding Li. Sampling online social networks by random walk. In *Proceedings of the First ACM International Workshop on Hot Topics on Interdisciplinary Social Networks Research*, pages 33–40. ACM, 2012.
20. Marcelo Maia, Jussara Almeida, and Virgílio Almeida. Identifying user behavior in online social networks. In *Proceedings of the 1st workshop on Social network systems*, pages 1–6. ACM, 2008.
21. Seyed Ahmad Moosavi and Mehrdad Jalali. Community detection in online social networks using actions of users. In *Intelligent Systems (ICIS), 2014 Iranian Conference on*, pages 1–7. IEEE, 2014.
22. Fatemeh Rahimian, Amir H Payberah, Sarunas Girdzijauskas, Mark Jelasity, and Seif Haridi. Ja-be-ja: A distributed algorithm for balanced graph partitioning. In *Self-Adaptive and Self-Organizing Systems (SASO), 2013 IEEE 7th International Conference on*, pages 51–60. IEEE, 2013.
23. Sutharshan Rajasegarar, Christopher Leckie, and Marimuthu Palaniswami. Hyperspherical cluster based distributed anomaly detection in wireless sensor networks. *Journal of Parallel and Distributed Computing*, 74(1):1833–1847, 2014.
24. Yu Xia, Zhifeng Zhao, and Honggang Zhang. Distributed anomaly event detection in wireless networks using compressed sensing. In *Communications and Information Technologies (ISCIT), 2011 11th International Symposium on*, pages 250–255. IEEE, 2011.