Discovering Social Circles in Ego Networks

JULIAN McAULEY and JURE LESKOVEC, Computer Science Department, Stanford University

People's personal social networks are big and cluttered, and currently there is no good way to automatically organize them. Social networking sites allow users to manually categorize their friends into *social circles* (e.g., "circles" on Google+, and "lists" on Facebook and Twitter). However, circles are laborious to construct and must be manually updated whenever a user's network grows. In this article, we study the novel task of automatically identifying users' social circles. We pose this task as a multimembership node clustering problem on a user's ego network, a network of connections between her friends. We develop a model for detecting circles that combines network structure as well as user profile information. For each circle, we learn its members and the circle-specific user profile similarity metric. Modeling node membership to multiple circles allows us to detect overlapping as well as hierarchically nested circles. Experiments show that our model accurately identifies circles on a diverse set of data from Facebook, Google+, and Twitter, for all of which we obtain hand-labeled ground truth.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Clustering

General Terms: Algorithms, Experimentation

Additional Key Words and Phrases: Community detection, social circles, ego networks, machine learning

ACM Reference Format:
Julian McAuley and Jure Leskovec. 2014. Discovering social circles in ego networks. ACM Trans. Knowl. Discov. Data 8, 1, Article 4 (February 2014), 28 pages.
DOI: http://dx.doi.org/10.1145/2556612

4

1. INTRODUCTION

Online social networks allow us to follow streams of posts generated by hundreds of our friends and acquaintances. The people we follow generate overwhelming volumes of information, and to cope with the 'information overload,' we need to organize our personal social networks [Agarwal et al. 2008; Chen and Karger 2006; El-Arini et al. 2009]. One of the main mechanisms for users of social networking sites to organize their networks and the content generated by them is to categorize their friends into *social circles*. Social circles are lists of people that can be used for content filtering, for privacy, and for sharing groups of users that others may wish to follow. Practically all major social networks provide such functionality—for example, "circles" on Google+, and "lists" on Facebook and Twitter.

Examples of circles from a user's personal social network are shown in Figure 1. The "owner" of such a network (the "ego") may form circles based on common bonds and attributes between themselves and the users whom they follow. In this example, the ego may wish to share their latest TKDD article only with their friends from the computer science department, whereas their baby photos should be shared only with their immediate family; similarly, they may wish to limit the amount of content

Author's address: J. McAuley and J. Leskovec, Department of Computer Science, Stanford University, Stanford, CA; email: {jmcauley, jure}@cs.stanford.edu.

DOI: http://dx.doi.org/10.1145/2556612

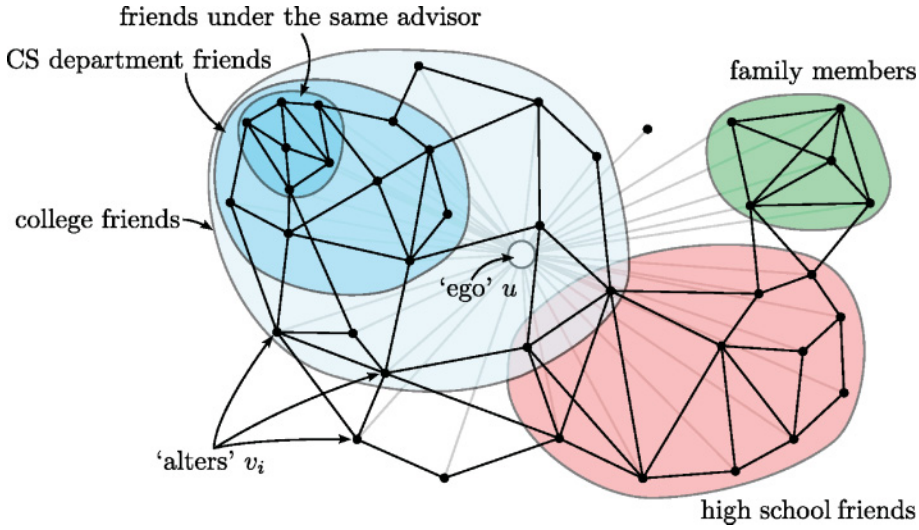


Fig. 1. An ego network with labeled circles. The central user, the ego, is friends with all other users (the alters) in the network. Alters may belong to any number of circles, including none. We aim to discover circle memberships and to find common properties around which circles form. This network shows typical behavior that we observe in our data: approximately 25% of our ground-truth circles (from Facebook) are contained *completely* within another circle, 50% overlap with another circle, and 25% of the circles have no members in common with any other circle.

generated by their high school friends. These are precisely the types of functionality that *circles* are intended to facilitate.

Currently, users in Facebook, Google+, and Twitter identify their circles either manually or in a naïve fashion, for example, by identifying friends sharing certain features or properties in common. Neither approach is particularly satisfactory: the former is time consuming and does not update automatically as a user adds more friends, whereas the latter fails to capture individual aspects of users' communities and may function poorly when profile information is missing or withheld.

Present work. In this article, we study the problem of automatically discovering users' social circles. In particular, given a single user with her personal social network, our goal is to identify her circles, each of which is a subset of her friends.

Circles are user specific, as each user organizes her personal network of friends independently of all other users to whom she is not connected. This means that we can formulate the problem of circle detection as a clustering problem on her ego network, the network of friendships between her friends. In practice, circles may overlap (a circle of friends from the same hometown may overlap with a circle from the same college), or be hierarchically nested (among friends from the same college, there may be a denser circle from the same degree program). We design our model with both types of behavior in mind.

Figure 1 illustrates the task. Here, we are given a single user u, and we form a network between her friends v_i. We refer to the user u as the *ego* and to the nodes v_i as *alters*. The task is then to identify the circles to which each alter v_i belongs, as in Figure 1. In other words, the goal is to find communities/clusters in u's ego network.

Generally, there are two useful sources of data that help with this task. The first is the set of edges of the ego network. We expect that circles are formed by densely connected sets of alters [Newman 2006]. However, different circles overlap heavily—that is, alters belong to multiple circles simultaneously [Ahn et al. 2010; Palla et al.

2005], and many circles are hierarchically nested in larger ones (as in Figure 1). Thus, it is important to model an alter's memberships to multiple circles. Second, we expect that each circle is not only densely connected but that its members also share common properties or traits [Mislove et al. 2010]. Therefore, we need to explicitly model the different dimensions of user profiles along which each circle emerges.

We model circle affiliations as latent variables and similarity between alters as a function of common profile information. We propose an *unsupervised* method to learn which dimensions of profile similarity lead to densely linked circles. After developing a model for this problem, we then study the related problems of *updating* a user's circles once new friends are added to the network and using weak supervision from the user in the form of "seed nodes" to improve classification.

Our model has two innovations. First, in contrast to mixed-membership models [Airoldi et al. 2008], we predict *hard* assignment of a node to *multiple* circles, which proves critical for good performance [Gregory 2010b]. By hard assignment, we mean that each node has a binary membership to each circle rather than a partial or probabilistic membership. Second, by proposing a parameterized definition of profile similarity, we learn the dimensions of similarity along which links emerge [Feld 1981; Simmel 1964]. This extends the notion of homophily [Lazarsfeld and Merton 1954; McPherson et al. 2001] by allowing different circles to form along different social dimensions, an idea related to the concept of Blau spaces [McPherson 1983]. We achieve this by allowing each circle to have a different definition of profile similarity so that one circle might form around friends from the same school and another around friends from the same location. We learn the model by simultaneously choosing node circle memberships and profile similarity functions to best explain the observed data.

We evaluate our method on a new dataset of 1,143 ego networks from Facebook, Google+, and Twitter, for which we obtain hand-labeled ground truth from 5,636 circles.

Experimental results show that by simultaneously considering social network structure as well as user profile information, our method performs significantly better than natural alternatives and the current state of the art. Besides being more accurate, our method also allows us to generate automatic explanations of why certain nodes belong to common communities. Our method is completely unsupervised and is able to automatically determine both the number of circles as well as the circles themselves. We show that the same model can be adapted to deal with weak supervision and to update already-complete circles as new users arrive.

A preliminary version of this article appeared in NIPS 2012 [McAuley and Leskovec 2012].

1.1. Related Work

Although a circle is not precisely the same as a community, our work broadly falls under the umbrella of community detection [Lancichinetti and Fortunato 2009b; Schaeffer 2007; Leskovec et al. 2010; Porter et al. 2009; Newman 2004]. Whereas classical clustering algorithms assume disjoint communities [Schaeffer 2007], many authors have made the observation that communities in real-world networks may overlap [Lancichinetti and Fortunato 2009a; Gregory 2010a; Lancichinetti et al. 2009; Yang and Leskovec 2012] or have hierarchical structure [Ravasz and Barabási 2003].

Topic-modeling techniques have been used to uncover "mixed memberships" of nodes to multiple groups, and extensions allow entities to be attributed with text information. Airoldi et al. [2008] modeled node attributes as latent variables drawn from a Dirichlet distribution so that each attribute can be thought of as a partial membership to a community. Other authors extended this idea to allow for side information associated with the nodes and edges [Balasubramanyan and Cohen 2011; Chang and Blei 2009; Liu et al. 2009]. A related line of work by Hoff et al. [2002] also used latent node

attributes to model edge formation between 'similar' users, which were adapted to clustering problems in Handcock et al. [2007b] and Krivitsky et al. [2009].

Classical clustering algorithms tend to identify communities based on node features [Johnson 1967] or graph structure [Ahn et al. 2010; Palla et al. 2005] but rarely use both in concert. Our work is related to Yoshida [2010], in the sense that it performs clustering on social-network data, and Frank et al. [2012], which models memberships to multiple communities. Another work closely related to ours is Yang and Leskovec [2012], which explicitly models hard memberships of nodes to multiple overlapping communities, although it does so purely based on network information rather than node features. Our inference procedure is also similar to that of Hastings [2006], which treats nodes' assignments to communities as a maximum a posteriori inference problem between a set of interdependent variables.

Finally, Chang et al. [2009], Menon and Elkan [2011, 2010], and Vu et al. [2011] model network data similar to ours; like our own work, they model the probability that two nodes will form an edge, although the underlying models do not form communities, so they are not immediately applicable to the problem of circle detection.

The rest of this article is organized as follows. We propose a generative model for the formation of edges within communities in Section 2. In Section 3, we derive an efficient model parameter learning strategy. In Section 4, we describe extensions to our model that allow it to be used in semisupervised settings in order to help users update and maintain their circles. In Section 5, we show how to scale the model to large ego networks. We describe the datasets that we construct in Section 6. We give two schemes for automatically constructing parameterized user similarity functions from profile data in Section 7. Finally, in Section 8, we describe our evaluation and experimental results.

2. A GENERATIVE MODEL FOR FRIENDSHIPS IN SOCIAL CIRCLES

In the following discussion, we describe our model of social circles. We desire a model of circle formation with the following properties:

(1) Nodes within circles should have common attributes, or aspects.
(2) Different circles should be formed by different aspects—for example, one circle might be formed by family members and another by students who attended the same university.
(3) Circles should be allowed to overlap, and smaller circles should be allowed to form within larger ones. For example, a circle of friends from the same degree program may form within a circle from the same university, as in Figure 1.
(4) We would like to leverage both profile information and network structure in order to identify circles.
(5) Ideally, we would like to be able to pinpoint *which* aspects of a profile caused a circle to form so that the model is interpretable by the user.

The input to our model is a (directed or undirected) ego network $G = (V, E)$, along with profiles for each user $v \in V$. The center node u of the ego network (the ego) is not included in G, but rather G consists only of u's friends (the alters). We define the ego network in this way precisely because creators of circles do not themselves appear in their own circles. For each ego network, our goal is to predict a set of circles $\mathcal{C} = \{C_1 \dots C_K\}$, $C_k \subseteq V$ and associated parameter vectors θ_k that encode how each circle emerged.

We encode user profiles into pairwise feature vectors $\phi(x, y)$ that in some way capture what properties the users x and y have in common. θ_k then tells us which dimensions of $\phi(x, y)$ are important for a particular circle C_k. For example, one dimension of $\Phi(x, y)$ might be a binary indicator that takes the value 1 if both x and y attended a certain

school; the corresponding dimension of θ_k would then be high if the circle C_k consists of users who attended that school.

For the sake of generality, we first describe a model that can be applied using arbitrary feature vectors $\phi(x, y)$; later, we propose several ways to construct feature vectors $\phi(x, y)$ from user profiles to suit our particular application.

We describe a model of social circles that treats circle memberships as latent variables. Nodes within a common circle are given an opportunity to form an edge, which naturally leads to hierarchical and overlapping circles. We will then devise an *unsupervised* algorithm to jointly optimize the latent variables and the profile similarity parameters to best explain the observed network data.

Our model of social circles is defined as follows. Given an ego network G and a set of K circles $\mathcal{C} = \{C_1 \dots C_K\}$, we model the probability that a pair of nodes $(x, y) \in V \times V$ form an edge as

$$p((x, y) \in E) \propto \exp\Bigg\{ \underbrace{\sum_{C_k \supseteq \{x,y\}} \langle \phi(x, y), \theta_k \rangle}_{\text{circles containing both nodes}} - \underbrace{\sum_{C_k \not\supseteq \{x,y\}} \alpha_k \langle \phi(x, y), \theta_k \rangle}_{\text{all other circles}} \Bigg\}. \tag{1}$$

For each circle C_k, θ_k is the profile similarity parameter that we will learn. The idea is that $\langle \phi(x, y), \theta_k \rangle$ is high if both nodes belong to C_k, and low if either of them do not. The parameter α_k trades off these two effects—that is, it trades off the influence of edges within C_k compared to edges outside of (or crossing) C_k. This probability therefore captures the intuition that edges are likely to form within circles and unlikely to form outside of them. This is achieved by *rewarding* edges that appear within circles according to $\langle \phi(x, y), \theta_k \rangle$ and *penalizing* edges that appear outside of circles according to $\alpha_k \langle \phi(x, y), \theta_k \rangle$. The presence of α_k trades off this reward against this penalty.

Since the feature vector $\phi(x, y)$ encodes the similarity between the profiles of two users x and y, the parameter vector θ_k encodes which dimensions of profile similarity caused the circle to form so that nodes within a circle C_k should "look similar" according to θ_k. Note that the pair (x, y) should be treated as an *unordered* pair in the case of an undirected network (e.g., Facebook) but should be treated as an *ordered* pair for directed networks (e.g., Google+ and Twitter).

Considering that edges $e = (x, y)$ are generated independently, we can write the probability of G as

$$P_\Theta(G; \mathcal{C}) = \prod_{e \in E} p(e \in E) \times \prod_{e \notin E} p(e \notin E), \tag{2}$$

where $\Theta = \{(\theta_k, \alpha_k)\}^{k=1 \dots K}$ is our set of model parameters.

Defining the shorthand notation

$$\begin{aligned} d_k(e) &= \delta(e \in C_k) - \alpha_k \delta(e \notin C_k), \\ \Phi(e) &= \sum_{C_k \in \mathcal{C}} d_k(e) \langle \phi(e), \theta_k \rangle \end{aligned}$$

allows us to write the probability from Equation (1) as $p(e \in E) \propto \exp\{\Phi(e)\}$ ($\delta(c)$ is an indicator function that takes the value 1 when the condition c is satisfied). Note that $\Phi(e)$ collapses both the positive and negative terms from Equation (1) into a single expression that now sums over all circles.

The probability $p(e \in E) \propto \exp\{\Phi(e)\}$ is still unnormalized—that is, it takes a value between 0 and ∞. To normalize it, we pass it through a logistic function that maps it

to a value between 0 and 1. This also implicitly defines $p(e \notin E)$. Specifically, we have

$$p(e \in E) = \frac{e^{\Phi(e)}}{1+e^{\Phi(e)}},$$
$$p(e \notin E) = 1 - p(e \in E) = \frac{1}{1+e^{\Phi(e)}}.$$

We can now write down the log-likelihood of G:

$$l_\Theta(G;\mathcal{C}) = \sum_{e \in E} \Phi(e) - \sum_{e \in V \times V} \log(1+e^{\Phi(e)}). \tag{3}$$

Next, we describe how to optimize node circle memberships $\mathcal{C}$ as well as the parameters of the user profile similarity functions $\Theta = \{(\theta_k, \alpha_k)\}$ $(k = 1 \ldots K)$ given a graph G and user profiles.

3. UNSUPERVISED LEARNING OF MODEL PARAMETERS

Treating circles $\mathcal{C}$ as latent variables, we aim to find $\hat{\Theta} = \{\hat{\theta}, \hat{\alpha}\}$ to maximize the regularized log-likelihood of Equation (3):

$$\hat{\Theta}, \hat{\mathcal{C}} = \underset{\Theta, \mathcal{C}}{\operatorname{argmax}} l_\Theta(G;\mathcal{C}) - \lambda\Omega(\theta). \tag{4}$$

We solve this problem using coordinate ascent on Θ and $\mathcal{C}$ [MacKay 2003]. That is, we alternate between fitting our latent variables (the circle memberships) and fitting our model parameters (Θ). In practice, the following two steps are iterated until convergence:

$$\mathcal{C}^t = \underset{\mathcal{C}}{\operatorname{argmax}} l_{\Theta^t}(G;\mathcal{C}) \tag{5}$$

$$\Theta^{t+1} = \underset{\Theta}{\operatorname{argmax}} l_\Theta(G;\mathcal{C}^t) - \lambda\Omega(\theta). \tag{6}$$

We optimize Equation (6) using L-BFGS, a standard quasi-Newton procedure to optimize smooth functions of many variables [Nocedal 1980]. Computing partial derivatives, we obtain

$$\frac{\partial l}{\partial \theta_k} = \sum_{e \in V \times V} -d_e(k)\phi(e)_k \frac{e^{\Phi(e)}}{1+e^{\Phi(e)}} + \sum_{e \in E} d_k(e)\phi(e)_k - \frac{\partial \Omega}{\partial \theta_k} \tag{7}$$

$$\frac{\partial l}{\partial \alpha_k} = \sum_{e \in V \times V} \delta(e \notin C_k)\langle \phi(e), \theta_k \rangle \frac{e^{\Phi(e)}}{1+e^{\Phi(e)}} - \sum_{e \in E} \delta(e \notin C_k)\langle \phi(e), \theta_k \rangle. \tag{8}$$

To optimize Equation (6), we observe that for fixed $\mathcal{C} \backslash C_i$, we can solve $\operatorname{argmax}_{C_i} l_\Theta(G; \mathcal{C} \setminus C_i)$ by expressing it as pseudo-boolean optimization in a pairwise graphical model [Boros and Hammer 2002]. Pseudo-boolean optimization refers to problems defined over boolean variables. In our setting, we use boolean variables to define whether or not a node is a member of a particular circle.

In particular, we will show next that in this context, optimizing the memberships of a particular circle can be written in the form

$$\underset{X}{\operatorname{argmax}} \sum_{x_i, x_j \in X} E_{i,j}(x_i, x_j),$$

where x_i and x_j are boolean variables, and $E_{i,j} : \{0, 1\}^2 \rightarrow \mathbb{R}$ is a pairwise energy. This is the general form of (pairwise) pseudo-boolean optimization, where we optimize a real-valued objective defined over binary variables. Next, we show that optimizing the memberships of a single circle can be expressed in this form.

Formally, we find an expression for the conditional log-likelihood of Equation (3), conditioned on the memberships of nodes to every circle except a particular circle C_k, and show that it can be written as a sum of pairwise binary energies.

In our setting, binary variables encode the memberships of nodes to a particular circle C_k. There are four possible cases that we must consider: (1) an edge appears outside the circle, (2) a nonedge appears outside the circle, (3) an edge appears inside the circle, and (4) a nonedge appears inside the circle. "Outside the circle" also includes the case where one of the nodes appears inside the circle but the other does not—that is, where the pair of nodes "cross" the circle. Intuitively, we want to maximize the energy and thus cases (2) and (3) should have high energy (we prefer edges inside circles), whereas (1) and (4) should have low energy (we do not want to many edges outside the circles). The precise energy should also depend on $\langle \phi(e), \theta_k \rangle$, which encodes how compatible the features of i and j are with the circle's parameters θ_k.

In order to achieve this, we first define (for a pair of nodes e)

$$o_k(e) = \sum_{C_i \in \mathcal{C} \setminus C_k} d_k(e) \langle \phi(e), \theta_k \rangle,$$

where we are summing over all circles C_i *other than* the circle C_k whose members we are currently optimizing. This expression will take a positive value when nodes at endpoints of edge $e = (i, j)$ belong to many common circles (other than C_k) and a negative value otherwise.

We then define the pairwise energy E_e^k (parameterized in terms of the circle k, and a pair of nodes $e = (x, y)$) as

$$E_e^k(0,0) = E_e^k(0,1) = E_e^k(1,0) = \begin{cases} o_k(e) - \alpha_k \langle \phi(e), \theta_k \rangle - \log\left(1 + e^{o_k(e) - \alpha_k \langle \phi(e), \theta_k \rangle}\right), & e \in E \\ -\log\left(1 + e^{o_k(e) - \alpha_k \langle \phi(e), \theta_k \rangle}\right), & e \notin E \end{cases}$$

$$E_e^k(1,1) = \begin{cases} o_k(e) + \langle \phi(e), \theta_k \rangle - \log\left(1 + e^{o_k(e) + \langle \phi(e), \theta_k \rangle}\right), & e \in E \\ -\log\left(1 + e^{o_k(e) + \langle \phi(e), \theta_k \rangle}\right), & e \notin E \end{cases}.$$

$E_e^k(1, 1)$ corresponds to the case where both nodes belong to the circle, whereas in the other three cases, at least one node does not belong to the circle. Note that this captures the four possible cases described earlier (cases 1 through 4 are the four expressions in the preceding equation, respectively).

Finally, the optimization problem to infer memberships of a single circle becomes

$$C_k = \underset{C}{\operatorname{argmax}} \sum_{(x,y) \in V \times V} E_{(x,y)}^k(\delta(x \in C), \delta(y \in C)). \tag{9}$$

By expressing the problem in this form, we can draw upon existing work on pseudo-boolean optimization. Optimization problems of this form are known to be NP-hard in general, although efficient approximation algorithms are readily available [Rother et al. 2007]. We use the publicly available QPBO software described in Rother et al. [2007], which implements algorithms described in Hammer et al. [1984] and Kohli and Torr [2005] and is able to accurately approximate problems of the form shown in Equation (9). Essentially, problems of the type shown in Equation (9) are reduced to *maximum flow*, where boolean labels for each node are recovered from their assignments to "source" and "sink" sets, and the energies $E(x_i, x_j)$ are transformed to

capacities (albeit through a nontrivial transformation). Such algorithms have worst-case complexity $O(|N|^3)$, although the average case running time is far better [Kolmogorov and Rother 2007]. We solve Equation (9) for each circle C_k in a random order.

The two optimization steps of Equations (5) and (6) are repeated until convergence—that is, until $\mathcal{C}^{t+1} = \mathcal{C}^t$. The entire procedure is presented in Algorithm 1.

Finally, we regularize Equation (4) using the ℓ_1 norm,

$$\Omega(\theta) = \sum_{k=1}^{K} \sum_{i=1}^{|\theta_k|} |\theta_{ki}|,$$

which leads to sparse (and readily interpretable) parameters.

Our algorithm can readily handle all but the largest problem sizes typically observed in ego networks: in the case of Facebook, the average ego network has around 190 nodes [Ugander et al. 2011], whereas the largest network we encountered has 4,964 nodes. Later, in Section 5, we will exploit the fact that our features are binary, and that many nodes share similar features, to develop more efficient algorithms based on Markov Chain Monte Carlo (MCMC) inference. Note that since the method is *unsupervised*, inference is performed independently for each ego network. This means that our method could be run on the full Facebook graph (for example), as circles are independently detected for each user, and the ego networks typically contain only hundreds of nodes. In Section 4, we describe extensions that allow our model to be used in semisupervised settings.

ALGORITHM 1: Predict complete circles with hyperparameters λ, K.

Data: ego-network $G = (V, E)$, edge features $\phi(e) : E \to \mathbb{R}^F$, hyperparameters λ, K
Result: parameters $\hat{\Theta} := \{(\hat{\theta}_k, \hat{\alpha}_k)\}^{k=1\ldots K}$, communities $\hat{\mathcal{C}}$
initialize $\theta_k^0 \in \{0, 1\}^F$, $\alpha_k^0 := 1$, $C_k := \varnothing$, $t := 0$;
repeat
  for $k \in \{1 \ldots K\}$ **do**
    $\mathcal{C}_k^t := \text{argmax}_C \sum_{(x,y) \in V \times V} E^k_{(x,y)}(\delta(x \in C), \delta(y \in C))$;
    `// using QPBO, see (eq. 9)`
  end
  $\Theta^{t+1} := \text{argmax}_\Theta l_\Theta(G; C^t) - \lambda\Omega(\theta)$;
  `// using L-BFGS, see (eqs. 7 and 8)`
  $t := t + 1$;
until $C^{t+1} = \mathcal{C}^t$;

3.1. Hyperparameter Estimation

Estimating the number of circles. To automatically choose the optimal number of circles, we choose K to minimize an approximation to the Bayesian Information Criterion (BIC), an idea seen in several works on probabilistic clustering [Airoldi et al. 2008; Handcock et al. 2007a; Volinsky and Raftery 2000]. In this context, the BIC is defined as

$$BIC(K; \Theta^K) \simeq -2l_{\Theta^K}(G; \mathcal{C}) + |\Theta^K| \log |E|, \tag{10}$$

where Θ^K is the set of parameters predicted when there are K circles, and $|\Theta^K|$ is the number of parameters (which increases linearly as K increases). We then choose K so as to minimize this objective:

$$\hat{K} = \underset{K}{\text{argmin}}\, BIC(K; \Theta^K). \tag{11}$$

In other words, an additional circle will only be added to the model if doing so has a significant impact on the log-likelihood.

Regularizer. The regularization parameter $\lambda \in \{0, 1, 10, 100\}$ was determined using leave-one-out cross validation, although in our experience, λ did not significantly impact performance.

4. EXTENSIONS

So far, we have considered the cold-start problem of predicting complete sets of circles using nothing but node attributes and edge information. In other words, we have treated circle prediction as an *unsupervised* task. This setting is realistic if users construct their circles only after their ego networks have already been defined. On the other hand, in settings where users build their circles incrementally, it is less likely that we would wish to predict complete circles from scratch. We note that both settings occur in the three social networks that we consider.

In this section, we describe techniques to exploit partially observed circle information to help users update and maintain their circles. In other words, we would like to apply our model to users' personal networks as they change and evolve. Since our model is probabilistic, it is straightforward to adapt it to make use of partially observed data by conditioning on the assignments of some of the latent variables in our model. In this way, we adapt our model for semisupervised settings in which a user labels some or all of the members of their circles. Later, in Section 5, we describe modifications of our model that allow it to be applied to large networks by exploiting the fact that many users assigned to common circles also have common features.

4.1. Circle Maintenance

First we deal with the problem of a user adding new friends to an established ego network, whose circles have already been defined. Thus, given a complete set of circles, our goal is to predict circle memberships for a new node, based on that node's features, and their patterns of connectivity to existing nodes in the ego network.

Our strategy to solve this problem is to adapt our previous algorithms to handle partially observed data. Rather than estimating circle memberships for all users, we only estimate the circle memberships of new users while *conditioning upon* the circle memberships of existing users.

Since the circles of existing users are fully observed in this setting, we simply fit the model parameters that best explain the ground-truth circles $\bar{\mathcal{C}}$ provided by the user:

$$\hat{\Theta} = \underset{\Theta}{\operatorname{argmax}} l_{\Theta}(G; \bar{\mathcal{C}}) - \lambda \Omega(\theta). \tag{12}$$

As with Equation (6), this is solved using L-BFGS, although optimization is significantly faster in this case, as there are no longer latent community memberships to infer and thus coordinate ascent is not required.

Next, we must predict to which of the K ground-truth circles a new user u belongs. That is, we must predict $c^u \in \{0, 1\}^K$, where each c_k^u is a binary variable indicating whether the user u should belong to the circle C_k. In practice, for the sake of evaluation, we shall suppress a single user from G and $\bar{\mathcal{C}}$, and try to recover their memberships.

This can be done by choosing the assignment c^u that maximizes the log-likelihood of $\mathcal{C}$ once u is added to the graph. We define the augmented community memberships as $\mathcal{C}^+(c^u) = \{C_k^+(c_k^u)\}^{k=1\ldots K}$, where

$$C_k^+\left(c_k^u\right) = \begin{cases} \bar{C}_k \cup \{u\}, & c_k^u = 1 \\ \bar{C}_k, & c_k^u = 0 \end{cases}. \tag{13}$$

The updated community memberships (for the new node u) are then chosen according to

$$\hat{c}_u = \underset{c^u}{\operatorname{argmax}}\, l_{\hat{\Theta}}(G \cup \{u\}; \mathcal{C}^+(c^u)). \tag{14}$$

The expression presented can be computed efficiently for different values of c^u by noting that the log-likelihood only changes for terms including u, meaning that we need to compute $p((x, y) \in E)$ only if $x = u$ or $y = u$. In other words, we only need to consider how the new user relates to existing users rather than considering how existing users relate to each other; thus, computing the log-likelihood requires linear (rather than quadratic) time. To find the optimal c^u, we can simply enumerate all 2^K possibilities, which is feasible as long as the user has no more than $K \simeq 20$ circles. For users with more circles, we must resort to an iterative update scheme as we did in Section 3.

4.2. Semisupervised Circle Prediction

Next we consider the problem of using weak supervision in the form of seed nodes to assist in circle prediction [Andersen and Lang 2006]. We envision that this version of the problem could be applied to alleviate the burden of manually labeling circles by automatically predicting complete circles using a minimal amount of supervision.

Formally, the user manually labels a few users from each of the circles that they want to create, say $\{s_1 \dots s_K\}$. Our goal is then to predict K circles $\mathcal{C} = \{C_1 \dots C_K\}$ subject to the constraint that $s_k \subseteq C_k$ for all $k \in \{1 \dots K\}$.

Again, since our model is probabilistic, this can be done by conditioning on the assignments of some of the latent variables. That is, we simply optimize $l_\Theta(G; \mathcal{C})$ subject to the constraint that $s_k \subseteq C_k$ for all $k \in \{1 \dots K\}$. In the parlance of graphical models, this means that rather than treating the seed nodes as latent variables to be predicted, we treat them as evidence on which we condition. We could also include negative evidence (i.e., the user could provide labels for users who do *not* belong to each circle), or we could have users provide additional labels interactively, although the setting described is the most similar to what is used in practice.

5. FAST INFERENCE IN LARGE EGO NETWORKS

Although our algorithm is able to handle the problem sizes typically encountered in ego networks (i.e., fewer than 1,000 friends), scalability to larger networks presents an issue, as we require quadratic memory to encode the compatibility between every pair of nodes (an issue that we note is also present in the existing approaches that we consider in Section 8). In this section, we propose a more scalable alternative that makes use of the fact that many nodes belonging to common communities also share common features.

Although the model presented in the previous sections allowed for arbitrary feature vectors, for the scalable version of our algorithm we assume that features ϕ are binary valued. We note that the features to be presented in Section 7 satisfy this assumption.

Given that features are binary valued, as are community memberships, if there are K communities and F-dimensional features, there can be at most 2^{K+F} types of node. In other words, every node's community membership is drawn from $\{0, 1\}^K$, and every node's feature vector is drawn from $\{0, 1\}^F$, so there are at most 2^{K+F} distinct community/feature combinations. Of course, the number of distinct node types is also bounded by $|V|$, the number of nodes in the graph.

In practice, however, the number of distinct node types is much smaller, as nodes belonging to common communities tend to have common features. Community memberships are also not independent: in Figure 2, we observed both disjoint and hierarchically nested communities, which means that of the 2^K possible community memberships, only a fraction of them occur in practice.

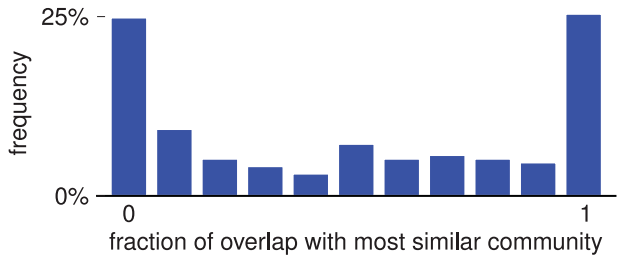


Fig. 2. Histogram of overlap between circles (on Facebook). A value of zero indicates that the circle does not intersect with any of the user's other circles, whereas a value of one indicates that a circle is entirely contained within another. Approximately 25% of circles exhibit the latter behavior.

In this section, we propose an MCMC sampler [Newman and Barkema 1999] that efficiently updates node-community memberships by "collapsing" nodes that have common features and community memberships. Note that the adaptations to be described can be applied to *any* types of feature (i.e., not just binary features). All we require is that many users share the same features; we assume binary features merely for the sake of presentation. Our implementation of these algorithms is available online.[1]

We start by representing each node using binary strings that encode both its community memberships and its features. Each node's community memberships are represented using $S : V \to \Sigma^K$, such that

$$S(x)[k] = \begin{cases} 1, & \text{if } x \in C_k \\ 0, & \text{otherwise} \end{cases}. \tag{15}$$

Similarly, each node's features are represented using the binary string Q, which, since our features are already binary, is simply the concatenation of the feature dimensions.

We now say that the type of a node x is the concatenation of its community string and its feature string, $(S(x); Q(x))$, and we build a (sparse) table $types : \Sigma^K \times \Sigma^F \to \mathbb{N}$ that counts how many nodes exist of each type.

In our setting, MCMC consists of repeatedly updating the (binary) label of each node in a particular community. Specifically, if the marginal (log) probability that a node x belongs to a community k is given by ℓ_x^k, then the node's new label is chosen by sampling $z \leftarrow \mathcal{U}(0, 1)$, and updating

$$S(x)[k] = \begin{cases} 1, & \text{if } z < \exp\left\{\frac{1}{T}\left(\ell_x^k(1) - \ell_x^k(0)\right)\right\} \\ 0, & \text{otherwise} \end{cases}, \tag{16}$$

where T is a temperature parameter that decreases at each iteration so that we are more likely to choose the label with higher probability as the model "cools."

Computing $\ell_x^k(0)$ and $\ell_x^k(1)$ (the probability that node x takes the label 0 or 1 in community k) requires computing $p((x, y) \in E)$ for all $y \in V$. However, we note that if two nodes y and y' have the same type (i.e., they belong to the same communities and have the same features), then $p((x, y) \in E) = p((x, y') \in E)$. In order to maximize the log-likelihood of the observed data, we must also consider whether (x, y) and (x, y') are actually edges in the graph. To do so, we first compute $\ell_x^k(0)$ and $\ell_x^k(1)$ under the assumption that no edges are incident on x, after which we correct for those edges incident on x. Thus, the running time of a single update is linear in the number of distinct node types, plus the average node degree, both of which are bounded by the number of nodes.

[1]Stanford Network Analysis Platform: http://snap.stanford.edu/snap.

ALGORITHM 2: Update memberships node x and circle k.

Data: node x whose membership to circle C_k is to be updated
Result: updated membership for node x
initialize $\ell_x^k(0) := 0$, $\ell_x^k(1) := 0$;
construct a dummy node x_0 with the communities and features of x but with $x \notin C_k$;
construct a dummy node x_1 with the communities and features of x but with $x \in C_k$;
for $(c, f) \in \mathrm{dom}(types)$ **do**
  // $c =$ community string, $f =$ feature string
  $n := types(c, f)$;
  // $n =$ number of nodes of this type
  if $S(x) = c \wedge Q(x) = f$ **then**
    // avoid including a self-loop on x
    $n := n - 1$;
  end
  construct a dummy node y with community memberships c and features f;
  // first compute probabilities assuming all pairs (x, y) are non-edges
  $\ell_x^k(0) := \ell_x^k(0) + n \log p((x_0, y) \notin E)$;
  $\ell_x^k(1) := \ell_x^k(1) + n \log p((x_1, y) \notin E)$;
end
for $(x, y) \in E$ **do**
  // correct for edges incident on x
  $\ell_x^k(0) := \ell_x^k(0) - \log p((x_0, y) \notin E) + \log p((x_0, y) \in E)$;
  $\ell_x^k(1) := \ell_x^k(1) - \log p((x_1, y) \notin E) + \log p((x_1, y) \in E)$;
end
// update membership to circle k
$types(S(x), Q(x)) := types(S(x), Q(x)) - 1$;
$z \leftarrow \mathcal{U}(0, 1)$;
if $z < \exp\{T(\ell_x^k(1) - \ell_x^k(0))\}$ **then**
  $S(x)[k] := 1$
else
  $S(x)[k] := 0$
end
$types(S(x), Q(x)) := types(S(x), Q(x)) + 1$;

In terms of Big-O notation, our MCMC algorithm computes updates in worst case $O(|N|^2)$ per iteration, whereas our pseudo-boolean optimization algorithm of Section 3 (which is based on maximum flow) requires $O(|N|^3)$. These are both loose, worst-case upper bounds: our MCMC algorithm is much faster if many nodes share common community affiliations, and maximum flow is typically much faster than cubic time. In practice, our pseudo-boolean optimization procedure is limited by quadratic memory requirements (energies must be stored for all pairs of nodes); this limitation is circumvented by our MCMC algorithm.

The entire procedure is demonstrated in Algorithm 2.

We also exploit the same observation when computing partial derivatives of the log-likelihood—that is, we first efficiently compute derivatives under the assumption that the graph contains no edges and then correct the result by summing over all edges in E.

6. DATASET DESCRIPTION

Although our method is unsupervised, we require labeled ground-truth data in order to evaluate its performance. We expended significant time, effort, and resources to

obtain high-quality hand-labeled data, which we have made available.[2–4] We were able to obtain ego networks and ground truth from three major social networking sites: Facebook, Google+, and Twitter.

From Facebook, we obtained profile and network data from 10 ego networks, consisting of 193 circles and 4,039 users. Data was obtained using a Facebook application.[5] To obtain circle information, we developed our own Facebook application and conducted a survey of 10 users (mostly Stanford graduate students), who were asked to manually identify all of the circles to which their friends belonged. It took each user around 2 to 3 hours to label their entire network. On average, users identified 19 circles in their ego networks, with an average circle size of 22 friends. Examples of circles that we obtained include students of common universities and classes, sports teams, relatives, and so forth.

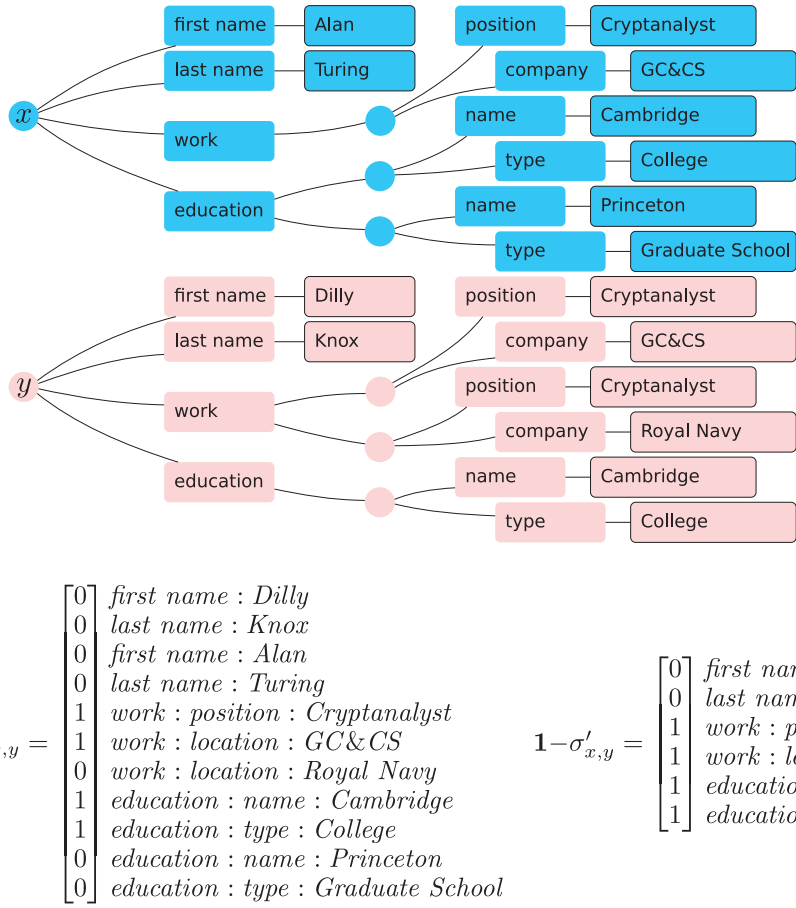
Figure 2 shows the extent to which our 193 user-labeled circles in 10 ego networks from Facebook overlap (intersect) with each other. Around one quarter of the identified circles are independent of any other circle, although a similar fraction is completely contained within another circle (e.g., friends who studied under the same advisor may be a subset of friends from the same university). The remaining 50% of communities overlap to some extent with another circle.

For the other two datasets, we obtained publicly accessible data. From Google+, we obtained data from 133 ego networks, consisting of 479 circles and 106,674 users. Data was collected from a Web page devoted to maintaining a list of publicly shared circles.[6] The Google+ circles are quite different from those from Facebook, in the sense that their creators have chosen to release them publicly, and because Google+ is a *directed* network (note that our model can very naturally be applied to both to directed and undirected networks). For example, many of our Google+ circles are collections of authors, politicians, or celebrities, who are presumably not close personal friends of the users who follow them. Finally, from Twitter, we obtained data from 1,000 ego networks, consisting of 4,869 circles (or lists [Kim et al. 2010; Nasirifard and Hayes 2011; Wu et al. 2011; Zhao 2011]) and 81,362 users. We started from a single Twitter user and proceeded in a breadth-first-search fashion, collecting data from users who had created at least two lists, stopping once we had 1,000 users. The ego networks that we obtained range in size from 10 to 4,964 nodes.

We acknowledge that there may be certain biases in our sample of users from whom we obtain ground truth. Our Facebook users are mostly graduate students, our Google+ users are those who have *chosen* to release their circles, and so forth. To address this, we are currently involved in a community effort to obtain larger, more representative datasets of the same form.[7]

Taken together, our data contains 1,143 different ego networks; 5,541 circles; and 192,075 users. The size differences between these datasets simply reflects the availability of data from each of the three sources. Our Facebook data is *fully labeled*, in the sense that we obtain *every* circle that a user considers to be a cohesive community, whereas our Google+ and Twitter data is only *partially labeled*, in the sense that we only have access to public circles. We design our evaluation procedure in Section 8 so that partial labels cause no issues.

[2]Facebook: http://snap.stanford.edu/data/egonets-Facebook.html.
[3]Google+: http://snap.stanford.edu/data/egonets-Gplus.html.
[4]Twitter: http://snap.stanford.edu/data/egonets-Twitter.html.
[5]http://snap.stanford.edu/socialcircles/.
[6]http://publiccircles.appspot.com/.
[7]https://www.kaggle.com/Facebook-Circles.



$$\mathbf{1} - \sigma_{x,y} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} first\ name : Dilly \\ last\ name : Knox \\ first\ name : Alan \\ last\ name : Turing \\ work : position : Cryptanalyst \\ work : location : GC\&CS \\ work : location : Royal\ Navy \\ education : name : Cambridge \\ education : type : College \\ education : name : Princeton \\ education : type : Graduate\ School \end{matrix}$$

$$\mathbf{1} - \sigma'_{x,y} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{matrix} first\ name \\ last\ name \\ work : position \\ work : location \\ education : name \\ education : type \end{matrix}$$

Fig. 3. Feature construction. Profiles are tree structured, and we construct features by comparing paths in those trees. Examples of trees for two users x (blue) and y (pink) are shown at top. Two schemes for constructing feature vectors from these profiles are shown at bottom. (1) (bottom left) We construct binary indicators measuring the difference between leaves in the two trees—for example, 'work→position→Cryptanalyst' appears in both trees. (2) (bottom right) We sum over the leaf nodes in the first scheme, maintaining the fact that the two users worked at the same institution but discarding the *identity* of that institution.

7. CONSTRUCTING FEATURES FROM USER PROFILES

Profile information in all of our datasets can be represented as a *tree* where each level encodes increasingly specific information (Figure 3, left). In other words, user profiles are organized into increasingly specific categories. For example, a user's profile might have an *education* category, which would be further separated into categories such as *name*, *location*, and *type*. The leaves of the tree are then specific values in these categories,such as *Princeton*, *Cambridge*, and *Graduate School*. Several works deal with automatically building features from tree-structured data [Haussler 1999; Vishwanathan and Smola 2002], but in order to understand the relationship between circles and user profile information, we shall design our own feature representation scheme.

We propose two hypotheses for how users organize their social circles: either they may form circles around users who share some common property *with each other*, or they may form circles around users who share some common property *with themselves*. For example, if a user has many friends who attended Stanford, then they may form a

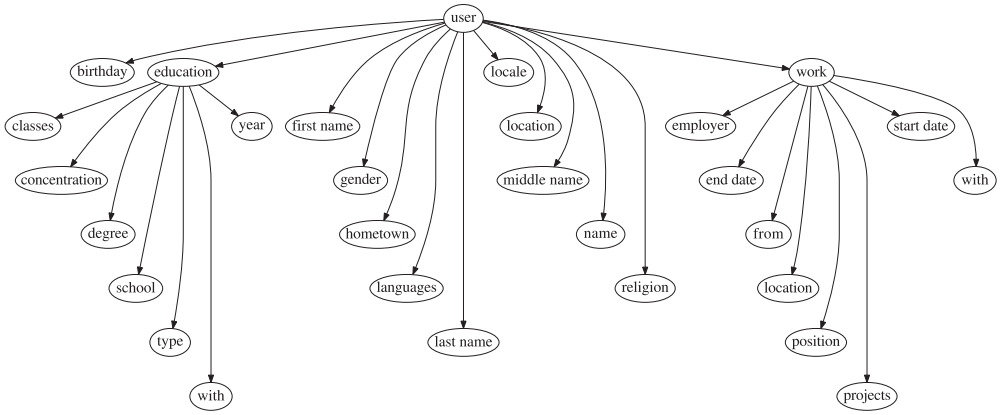


Fig. 4. Features obtained from Facebook. Each of the 26 leaf nodes is one category of features that we consider. Certain features (e.g., education) may appear multiple times with different values (e.g., for multiple schools that a user attended).

Stanford circle. On the other hand, if they themselves did *not* attend Stanford, they may not consider attendance to Stanford to be a salient feature. The feature construction schemes that we propose allow us to assess which of these hypotheses better represents the data we obtain.

From Google+, we collect data from six categories (gender, last name, job titles, institutions, universities, and places lived). From Facebook, we collect data from 26 categories, including users' hometowns, birthdays, colleagues, political and religious affiliations, and so forth. As a proxy for profile data, from Twitter we collect data from two categories, namely the set of hashtags and mentions used by each user during 2 weeks' worth of tweets. Categories correspond to parents of leaf nodes in a profile tree, as shown in Figure 3. The full set of features that we obtain from Facebook is shown in Figure 4.

We first propose a difference vector to encode the relationship between two profiles. A nontechnical description is given in Figure 3. Essentially, we want to encode those dimensions where two users are the same (e.g., Alan and Dilly went to the same graduate school) and those where they are different (e.g., they do not have the same surname). Suppose that users $v \in V$ each have an associated profile tree T_v and that $l \in T_v$ is a leaf in that tree. We define the difference vector $\sigma_{x,y}$ between two users x and y as a binary indicator encoding the profile aspects where users x and y differ (Figure 3, bottom left):

$$\sigma_{x,y}[l] = \delta((l \in \mathcal{T}_x) \neq (l \in \mathcal{T}_y)). \tag{17}$$

Note that feature descriptors are defined *per ego network*: although many thousands of high schools (for example) exist among all Facebook users, only a small number appear among any particular user's friends.

Although the difference vector presented has the advantage that it encodes profile information at a fine granularity, it has the disadvantage that it is high dimensional (up to 4,122 dimensions in the data we considered). One way to address this is to form difference vectors based on the *parents* of leaf nodes: this way, we encode what profile *categories* two users have in common but disregard specific values (Figure 3, bottom right). For example, we encode *how many* hashtags two users tweeted in common, but we discard *which* hashtags they tweeted:

$$\sigma'_{x,y}[p] = \sum_{l \in children(p)} \sigma_{x,y}[l]. \tag{18}$$

This scheme has the advantage that it requires a *constant* number of dimensions regardless of the size of the ego network (26 for Facebook, 6 for Google+, 2 for Twitter, as described earlier).

Based on the difference vectors $\sigma_{x,y}$ (and $\sigma'_{x,y}$), we now describe how to construct edge features $\phi(x, y)$. The first property we wish to model is that *members of circles should have common relationships* with each other:

$$\phi^1(x, y) = (1; -\sigma_{x,y}). \tag{19}$$

The second property we wish to model is that *members of circles should have common relationships to the ego of the ego network*. In this case, we consider the profile tree T_u *from the ego user u*. We then define our features in terms of that user:

$$\phi^2(x, y) = (1; -|\sigma_{x,u} - \sigma_{y,u}|) \tag{20}$$

($|\sigma_{x,u} - \sigma_{y,u}|$ is taken elementwise). These two parameterizations allow us to assess which mechanism better captures users' subjective definition of a circle. In both cases, we include a constant feature ("1"), which controls the probability that edges form within circles, or equivalently it measures the extent to which circles are made up of friends. Importantly, this allows us to predict memberships even for users who have no profile information, simply due to their patterns of connectivity.

Similarly, for the "compressed" difference vector $\sigma'_{x,y}$, we define

$$\psi^1(x, y) = (1; -\sigma'_{x,y}), \quad \psi^2(x, y) = (1; -|\sigma'_{x,u} - \sigma'_{y,u}|). \tag{21}$$

To summarize, we have identified four ways of representing the compatibility between different aspects of profiles for two users. We considered two ways of constructing a difference vector ($\sigma_{x,y}$ vs. $\sigma'_{x,y}$) and two ways of capturing the compatibility between a pair of profiles ($\phi(x, y)$ vs. $\psi(x, y)$). These features are designed to model the following behavior:

(1) Ego users build circles around common relationships between their friends (ϕ^1, ψ^1).
(2) Ego users build circles around common relationships between their friends and themselves (ϕ^2, ψ^2).

In our experiments, we assess which of these assumptions is more realistic in practice.

8. EXPERIMENTS

We first describe the evaluation metrics to be used in Sections 8.1 and 8.2, before describing the baselines to be evaluated in Section 8.4. We describe the performance of our (unsupervised) algorithm in Section 8.5 and extensions in Sections 8.7, 8.8, and 8.9.

8.1. Evaluation Metrics

Although our method is unsupervised, we can evaluate it on ground-truth data by examining the maximum-likelihood assignments of the latent circles $\mathcal{C} = \{C_1 \dots C_K\}$ after convergence. Our goal is that for a properly regularized model, the latent circles will align closely with the human-labeled ground-truth circles $\bar{\mathcal{C}} = \{\bar{C}_1 \dots \bar{C}_{\bar{K}}\}$.

To measure the alignment between a predicted circle C and a ground-truth circle $\bar{C}$, we compute the Balanced Error Rate (BER) between the two circles [Chen and Lin 2006],

$$BER(C, \bar{C}) = \frac{1}{2}\left(\frac{|C \backslash \bar{C}|}{|C|} + \frac{|C^c \backslash \bar{C}^c|}{|C^c|}\right). \tag{22}$$

This measure assigns equal importance to false positives and false negatives so that trivial or random predictions incur an error of 0.5 on average. Such a measure is preferable to the 0/1 loss (for example), which assigns extremely low error to trivial predictions. We also report the F_1 score:

$$F_1(C, \bar{C}) = 2 \cdot \frac{\text{precision}(C, \bar{C}) \cdot \text{recall}(C, \bar{C})}{\text{precision}(C, \bar{C}) + \text{recall}(C, \bar{C})}. \tag{23}$$

Treating $\bar{C}$ as a set of "relevant" documents, and C as a set of "retrieved" documents, precision and recall are defined as

$$\text{precision}(C, \bar{C}) = \frac{|C \cap \bar{C}|}{|C|}, \quad \text{recall}(C, \bar{C}) = \frac{|C \cap \bar{C}|}{|\bar{C}|}. \tag{24}$$

In practice, we find that the BER and the F_1 score product qualitatively similar results. The preceding scores compare a *single* predicted circle C to a *single* ground-truth circle $\bar{C}$. Next we describe how we compute the error for *sets* of circles $\mathcal{C}$ and $\bar{\mathcal{C}}$.

8.2. Aligning Predicted and Ground-Truth Circles

Since we do not know the correspondence between circles in $\mathcal{C}$ and $\bar{\mathcal{C}}$, we compute the optimal match via linear assignment by maximizing:

$$\max_{f:\mathcal{C}\to\bar{\mathcal{C}}} \frac{1}{|f|} \sum_{C \in \text{dom}(f)} (1 - BER(C, f(C))), \tag{25}$$

where f is a (partial) correspondence between $\mathcal{C}$ and $\bar{\mathcal{C}}$. That is, if the number of predicted circles $|\mathcal{C}|$ is less than the number of ground-truth circles $|\bar{\mathcal{C}}|$, then every circle $C \in \mathcal{C}$ must have a match $\bar{C} \in \bar{\mathcal{C}}$, but if $|\mathcal{C}| > |\bar{\mathcal{C}}|$, we do not incur a penalty for additional predictions that *could* have been circles but were not included in the ground truth. We use established techniques to estimate the number of circles so that none of the baselines suffers a disadvantage by mispredicting $\hat{K} = |\mathcal{C}|$. Similarly, when using the F_1 score, we compute the optimal match by maximizing:

$$\max_{f:\mathcal{C}\to\bar{\mathcal{C}}} \frac{1}{|f|} \sum_{C \in \text{dom}(f)} F_1(C, f(C)). \tag{26}$$

The preceding loss is designed to be insensitive to the fact that we have incomplete ground truth. That is, we allow for the possibility that we obtain only *some* of each user's possible circles. However, in the case of Facebook (where we have "complete" ground truth in the sense that survey participants ostensibly label *every* circle), our method should penalize predicted circles that do not appear in the ground truth. A simple penalty would be to assign an error of 0.5 (i.e., that of a random prediction) to additional circles in the case of Facebook. However, in our experience, our method did not overpredict the number of circles in the case of Facebook: on average, users identified 19 circles, whereas using the BIC described in Section 3.1, our method never predicted $K > 10$. In practice, this means that in the case of Facebook, we *always* penalize *all* predictions. Again we note that the process of choosing the number of circles using the BIC is a standard procedure from the literature [Airoldi et al. 2008; Handcock et al. 2007a; Volinsky and Raftery 2000], whose merit we do not assess in this article.

8.3. Estimating the Number of Circles for Nonprobabilistic Baselines: Network Modularity

Although for our algorithm, and other probabilistic baselines, we shall choose the number of communities using the BIC as described in Section 3.1, another standard criterion used to determine the number of communities in a network is the *modularity* [Newman 2006].

The BIC has the advantage that it allows for overlapping communities, whereas the modularity does not (i.e., it assumes all communities are disjoint); it is for this reason that we chose the BIC to choose $\hat{K}$ for our algorithm. On the other hand, the BIC can only be computed for *probabilistic* models (i.e., models that associate a likelihood with each prediction), whereas the modularity has no such restriction. For this reason, we shall use the modularity to choose $\hat{K}$ for nonprobabilistic baselines.

The modularity essentially measures the extent to which clusters in a network have dense internal, but sparse external, connections [Newman 2003]. If e_{ij} is the fraction of edges in the network that connect vertices in C_i to vertices in C_j, then the modularity is defined as

$$Q(K) = \sum_{i=1}^{K} \left\{ e_{ii} - \sum_{j=1}^{K} e_{ij} \right\}. \tag{27}$$

We then choose $\hat{K}$ so that the modularity is maximized.

8.4. Baselines

We considered a wide number of baseline methods, including those that consider only network structure, those that consider only profile information, and those that consider both.

Mixed Membership Stochastic Block Models [Airoldi et al. 2008]. This method detects communities based only on graph structure; the output is a stochastic vector for each node encoding partial memberships to each community. The optimal number of communities $\hat{K}$ is determined using the BIC as described in Equation (11). This model is similarto those of Liu et al. [2009] and Chang and Blei [2009], the latter of which includes the implementation of MMSB that we used. Since we require "hard" memberships for evaluation, we assign a node to a community if its partial membership to that community is positive.

Block-LDA [Balasubramanyan and Cohen 2011]. This method is similar to MMSB, except that it allows nodes to be augmented with side information in the form of "documents." For our purposes, we generate documents by treating aspects of user profiles as words in a bag-of-words model.

K-means Clustering [MacKay 2003]. Just as MMSB uses only the graph structure, K-means clustering ignores the graph structure and uses only node features (for node features, we again use a bag-of-words model). Here we choose $\hat{K}$ to maximize the modularity of $\mathcal{C}$, as defined in Equation (27).

Hierarchical Clustering [Johnson 1967]. This method builds a hierarchy of clusters. Like K-means, this method forms clusters based only on node profiles but ignores the network.

Link Clustering [Ahn et al. 2010]. Conversely, this method uses network structure but ignores node features to construct hierarchical communities in networks.

Clique Percolation [Palla et al. 2005]. This method also uses only network structure and builds communities from the union of small, densely connected subcommunities.

Low-Rank Embedding [Yoshida 2010]. This method uses both graph structure and node similarity information but does not perform any learning. We adapt an algorithm

Table I. Methods for Circle and Community Detection

Algorithm	Network Structure?	Node/Edge Features?	Overlapping Communities?	Hard Memberships?
MMSB	Yes	No	Yes	No
Block-LDA	Yes	Yes	Yes	No
K-means Clustering	No	Yes	No	Yes
Hierarchical Clustering	No	Yes	Yes	Yes
Link Clustering	Yes	No	No	Yes
Clique Percolation	Yes	No	Yes	Yes
Low-Rank Embedding	Yes	Yes	No	Yes
Multi-Assignment Clustering	No	Yes	Yes	Yes
Our algorithm	**Yes**	**Yes**	**Yes**	**Yes**

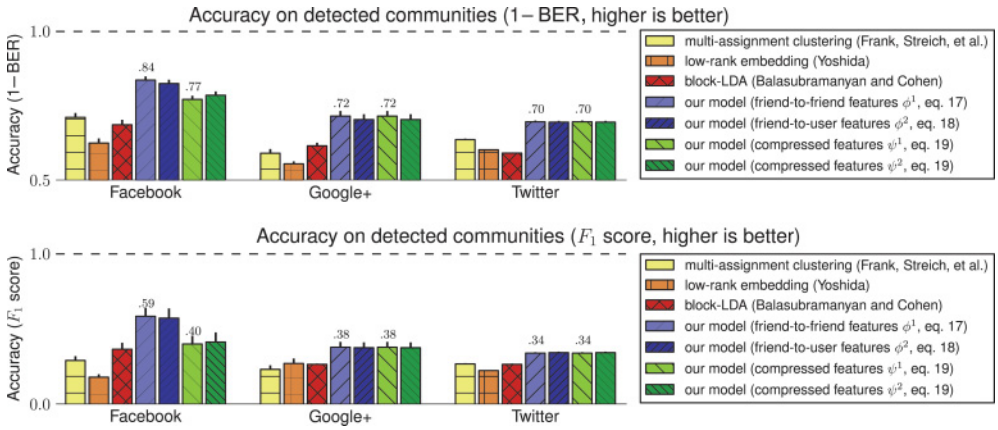


Fig. 5. Performance on Facebook, Google+, and Twitter, in terms of the BER (top) and the F_1 score (bottom). Higher is better. Error bars show standard error. The improvement of our best features ϕ^1 compared to the nearest competitor are significant at the 1% level or better.

described by Yoshida [2010], where node similarities are based on the cosine distance between profile bags of words. After our features are embedded into a low-dimensional space, we again use K-means clustering to detect communities, again choosing $\hat{K}$ to maximize the modularity.

Multi-Assignment Clustering [Frank et al. 2012]. Like ours, this method predicts hard assignments to multiple clusters, although it does so using only node features.

The methods presented (and our own) are summarized in Table I. Of the eight baselines highlighted, we report the three whose overall performance was the best, namely Block-LDA [Balasubramanyan and Cohen 2011] (which slightly outperformed mixed membership stochastic block models [Airoldi et al. 2008]), Low-Rank Embedding [Yoshida 2010], and Multi-Assignment Clustering [Frank et al. 2012].

8.5. Performance on Facebook, Google+, and Twitter Data

Figure 5 shows results on our Facebook, Google+, and Twitter data. The largest circles from Google+ were excluded as they exhausted the memory requirements of many of the baseline algorithms. Circles were aligned as described in Equation (25), with the number of circles $\hat{K}$ determined as described in Section 3. For nonprobabilistic baselines, we chose $\hat{K}$ to maximize the modularity, as described in Equation (27). In terms of absolute performance, our best model ϕ^1 achieves BER scores of 0.84 on

Facebook, 0.72 on Google+, and 0.70 on Twitter (F_1 scores are 0.59, 0.38, and 0.34, respectively).

The lower F_1 scores on Google+ and Twitter are explained by the fact that many circles have not been maintained since they were initially created (e.g., on Google+, we observe each circle as it was *when the user shared it* but do not see members who were added afterward). Thus, we achieve high recall (we recover the friends in each circle) but at low precision (we recover additional friends whom the user added after the circle was created or shared). This has a positive interpretation: our method is likely to be more accurate than our noisy ground truth represents.

Comparing our method to baselines, we notice that we outperform all baselines on all datasets by a statistically significant margin. Compared to the nearest competitors, our best-performing features ϕ^1 improve on the BER by 43% on Facebook, 26% on Google+, and 16% on Twitter (improvements in terms of the F_1 score are similar). Regarding the performance of the baseline methods, we note that good performance seems to depend critically on predicting *hard* memberships to *multiple* circles, using a combination of *node and edge* information; none of the baselines from Table I exhibit precisely this combination, which is a shortcoming that our model addresses.

Both of the features that we propose (friend-to-friend features ϕ^1 and friend-to-user features ϕ^2) perform similarly, revealing that both schemes ultimately encode similar information, which is not surprising, since users and their friends have similar profiles. Using the "compressed" features ψ^1 and ψ^2 does not significantly impact performance, which is promising because they have far lower dimension than the full features; what this reveals is that it is sufficient to model *categories* of attributes that users have in common (e.g., same school, same town) rather than the attribute values themselves.

We found that all algorithms perform significantly better on Facebook than on Google+ or Twitter. There are a few explanations. First, our Facebook data is *complete* in the sense that survey participants manually labeled *every* circle in their ego networks, whereas in other datasets, we only observe publicly visible circles, which may not be up-to-date. Second, the 26 profile categories available from Facebook are more informative than the 6 categories from Google+ or the tweet-based profiles that we built from Twitter. A more basic difference lies in the nature of the networks themselves: edges in Facebook encode *mutual* ties, whereas edges in Google+ and Twitter encode follower relationships, which changes the role that circles serve [Wu et al. 2011]. The latter two points explain why algorithms that use either edge or profile information in isolation are unlikely to perform well on this data.

8.6. Qualitative Analysis

Next we examine the output of our model in greater detail. Figure 6 shows results of our unsupervised method on example ego networks from Facebook and Google+. Different colors indicate true/false positives and negatives. Our method is correctly able to identify overlapping circles as well as subcircles (circles within circles).

Figure 7 shows parameter vectors learned for four circles for a particular Facebook user. Positive weights indicate properties that users in a particular circle have in common. Notice how the model naturally learns the social dimensions that lead to a social circle. The parameters shown in Figure 7 correspond to the social dimensions shown in Figure 4. Not shown is the parameter corresponding to the constant feature "1." In most cases, this feature has the highest weight: this reveals that membership to the same community provides the strongest signal that edges will form, whereas profile data provides a weaker (but still relevant) signal.

In light of the apparent importance of the constant feature, it is worth considering how well the model would perform *without* this feature or, indeed, with *only* this feature.

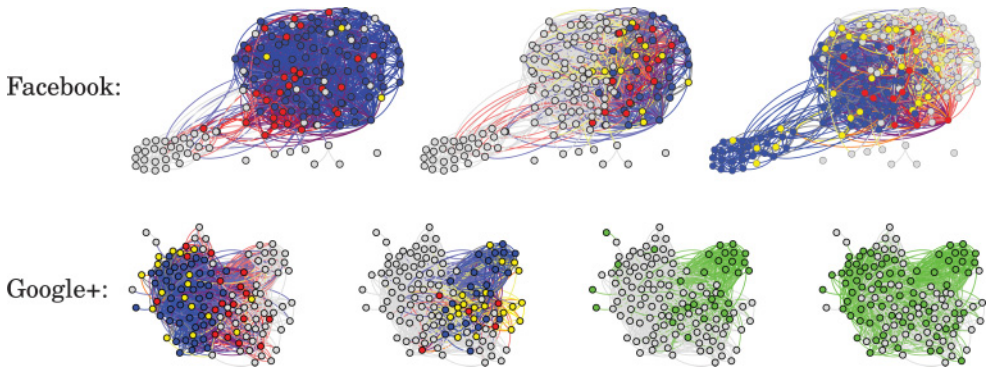


Fig. 6. Top: Three detected circles on a small ego network from Facebook, compared to three ground-truth circles (BER $\simeq$ 0.81). Blue nodes: true positives. Grey: true negatives. Red: false positives. Yellow: false negatives. Our method correctly identifies the largest circle (left), a subcircle contained within it (center), and a third circle that significantly overlaps with it (right). Bottom: Four detected circles on ego networks from Google+ (BER $\simeq$ 0.73). Green nodes in the two right networks show additional detected circles, whose accuracy cannot be evaluated because we only observed two circles in the ground truth.

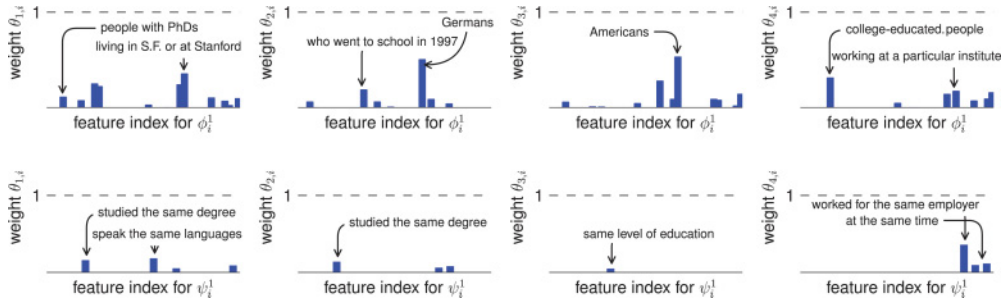


Fig. 7. Parameter vectors of four communities for a particular Facebook user. The top four plots show "complete" features ϕ^1, whereas the bottom four plots show "compressed" features ψ^1 (in both cases, BER $\simeq$ 0.78). For example, the former features encode the fact that members of a particular community tend to speak German, whereas the latter features encode the fact that they speak the same language. (Personally identifiable annotations have been suppressed.)

We performed this experiment on Facebook data, using the value of $\hat{K}$ as chosen by our best algorithm. Removing this feature harms performance, although not significantly, reducing the F_1 score to 0.53; in this case, other features are largely able to compensate for its absence.

More surprisingly, we obtain good performance using *only* this feature—that is, when predicting circle memberships using only edge information. On Facebook, using only edge information yields an F_1 score of 0.55. In other words, friendships alone are enough to accurately determine circle memberships, although profile information does yield a tangible benefit (results are similar in terms of the BER, and on other datasets, where profile features are not as informative as those on Facebook). However, using the BIC as in Equation (11) tends to overestimate $\hat{K}$ when using only a single feature (since the number of parameters is simply K itself); alternative methods to choose $\hat{K}$ would need to be considered under this scheme.

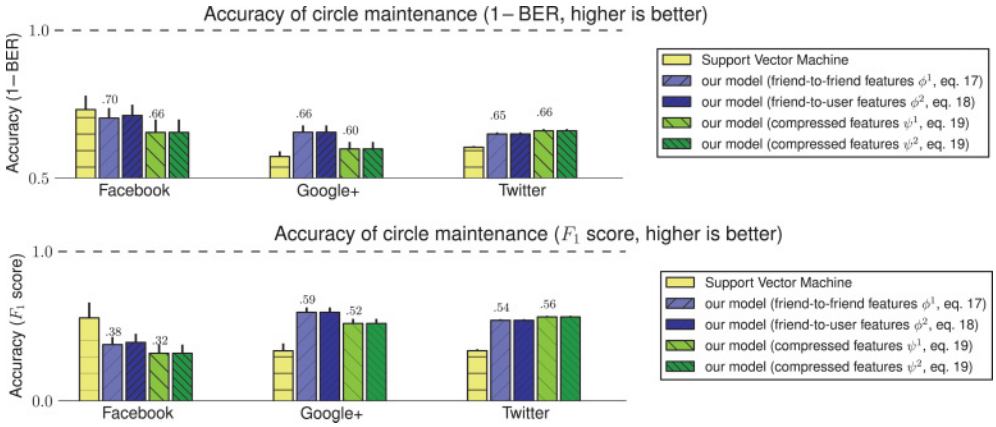


Fig. 8. Accuracy of assigning a new node to already-existing circles. Although a fully supervised SVM gives accurate results on Facebook (where node features are highly informative), our model yields far better results on Google+ and Twitter data.

8.7. Circle Maintenance

Next we examine the problem of adding new users to already-defined ego networks, in which complete circles have already been provided. For evaluation, we suppress a single user u from a user's ego network and learn the model parameters $\hat{\Theta}$ that best fit $G \setminus \{u\}$ and $\mathcal{C} \setminus \{u\}$. Our goal is then to recover the set of communities to which the node u belongs, as described in Section 4.1. Again we report the BER and F_1 score between the ground truth and the predicted set of community memberships for u. We use all of each users' circles for training, up to a maximum of 15 circles. This experiment is repeated for 10 random choices of the user u for each ego network in our dataset.

As a baseline, we compare the performance of our algorithm to that of a fully supervised Support Vector Machine (SVM) model. For each community C_k, we train a binary classifier that discriminates members from nonmembers based on their node features. Binary classifications are then made for each community independently.

Performance on this task is shown in Figure 8. On Facebook, Google+, and Twitter, our best-performing features ϕ^1 achieve BERs of 0.30, 0.34, and 0.34 (respectively), and F_1 scores of 0.38, 0.59, and 0.54. The SVM model achieves better accuracy when rich node features are available (which is the case for Facebook), although it fails to make use of edge information and does not account for interdependencies between circles. This proves critical in the case of Google+ and Twitter, where node information alone proves uninformative.

In practice, it is likely that many new friends may be added in close succession, so it may not be practical to refit the model for every new friend that is added, but rather to update the circle memberships of multiple new friends simultaneously. This setting can easily be handled be our method: observed circle memberships simply become evidence upon which we condition when maximizing the likelihood.

8.8. Semisupervised Circle Prediction

Our next task is to identify circles using a form of weak supervision provided by the user in the form of *seed nodes* as described in Section 4.2. In this setting, the user provides S seed nodes for each of K circles that they wish to identify. For evaluation, we select the K circles to be identified and the S seed nodes uniformly at random.

Without seed nodes (as in our initial experiments), the circles that are automatically identified by our algorithm may be quite different from those identified once seed

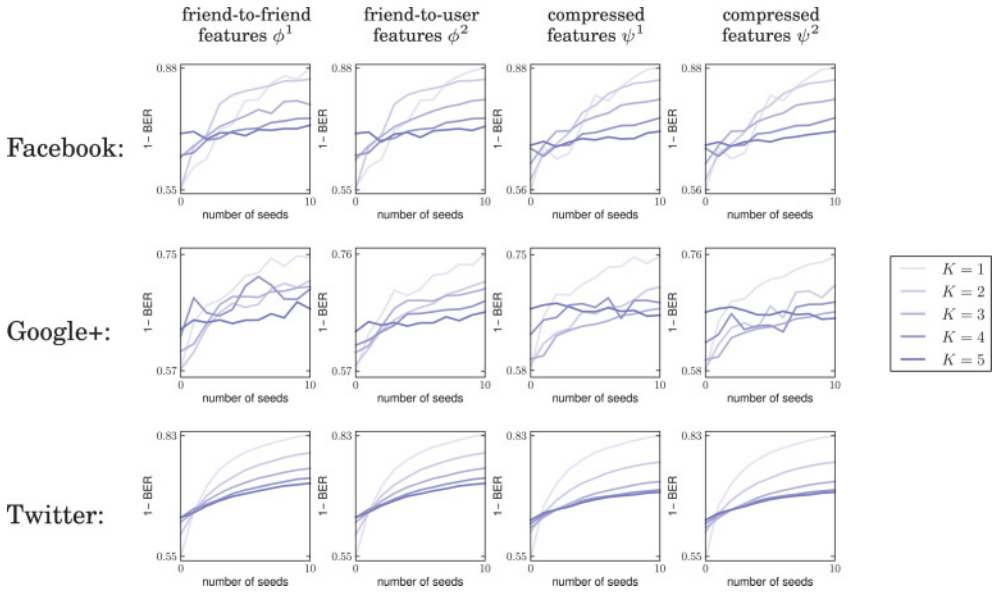


Fig. 9. Number of seeds versus accuracy (1 – BER) for different numbers of circles K. For each of the K circles being identified, the user provides the same number of seeds. Although providing additional seeds is generally beneficial to performance for all K, the benefit is most pronounced when the number of circles to be identified is small. Results in terms of the F_1 score are qualitatively similar and are omitted for brevity.

nodes are added. Similarly, there may be many circles containing the same seed nodes, meaning that different solutions may be chosen for different values of S. Thus, it is difficult to compare the loss of Equation (25) with and without seed nodes. To address this, we modify the matching objective of Equation (25) so that the K circles randomly selected for seeding must be the same as those matched when evaluating the loss. Thus, the loss is always evaluated on the same K circles for every number of seed nodes $S \in \{0 \dots 10\}$. Note also that for each value of K, performance is only evaluated on those ego networks with at least K ground-truth circles.

Figure 9 shows the performance of our algorithm for different numbers of seed nodes $S \in \{0 \dots 10\}$ and different numbers of circles $K \in \{1 \dots 5\}$. The same results in terms of the F_1 score are qualitatively similar and are omitted for brevity. We find that for all values of K, adding seed nodes increases the accuracy significantly, although the effect is most pronounced when the number of circles that the user wishes to identify is small.

Curiously, we find that whereas larger values of K lead to better prediction when there are no seeds, the opposite is true when there are many seeds. The former behavior may be explained by the simple fact that larger values of K are better able to fit the data, although the latter behavior is more enigmatic. Pleasingly, assuming that a user wishes to identify only a small number of circles at a time, the user can do so with very few seeds: for small K, most of the benefit is gained once only two or three seeds are provided.

8.9. Scalability Analysis

Figure 10 examines how our algorithm scales with the size of an ego network. Here we use the MCMC version of our algorithm described in Section 5. Figure 10 shows the total time taken to predict different numbers of circles in differently sized ego networks. Since the performance of our MCMC algorithm is a function of the number of circles

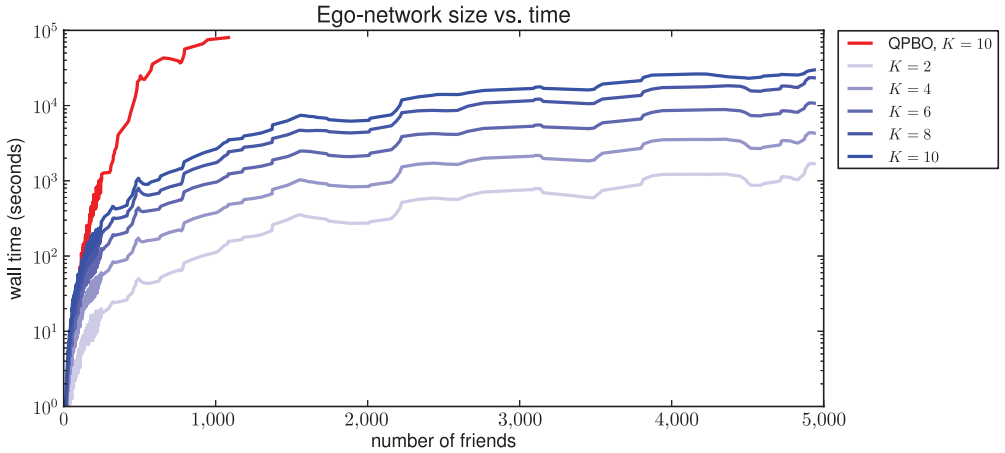


Fig. 10. Running time of our MCMC algorithm for different ego-network sizes and different values of K (the number of circles to be detected). For comparison, our inference algorithm from Section 3 (based on QPBO [Rother et al. 2007]) is shown for $K = 10$.

K and the feature dimensionality F, we fix the feature dimensionality at $F = 10$ for all ego networks, using the 10 most common features that appear in each ego network using the "friend-to-friend" features ϕ^1.

For comparison, Figure 10 shows the running time of inference using QPBO as described in Section 3. Although the two algorithms are competitive for up to a few hundred nodes, the QPBO algorithm becomes intractable for networks of around 1,000 nodes, since it requires us to optimize a probability distribution defined on *complete* graphs (in practice, in order to apply the QPBO algorithm in the previous experiments, we did not construct complete graphs but rather included only those edges whose influence on the likelihood was largest).

Although this version of the algorithm is not particularly efficient for small networks (identifying $K = 10$ circles on an ego network with 1,000 nodes requires around 1 hour), it has the advantage that it is easily able to scale to the largest ego networks that are ever encountered. For very large networks, the algorithm is able to take advantage of the fact that many nodes with the same features and community memberships can be "collapsed" so that the running time increases only modestly between 2,500 and 5,000 node ego networks.

Figure 11 shows the accuracy of our MCMC algorithm in terms of the BER and F_1 score. We note that the best performance of our algorithm is obtained on reasonably small ego networks, although in practice small networks account for the vast majority of our data (the data become very sparse for ego networks with more than 300 nodes). Note that the results for any particular value of K are slightly worse than those reported in Figure 5, since we are not selecting K using the BIC described in Section 3.1. Performance degrades only slightly for large ego networks, although it remains an open question whether this is due to the difficulty of optimization on large networks, or simply due to the fact that our model assumptions become increasingly violated as large networks become less "community like."

9. DISCUSSION AND FUTURE WORK

We have modeled circle detection as a problem that can be solved independently for each user. In practice, this assumption is advantageous, as it allows us to deal with several small problems independently, using sophisticated models that could not easily

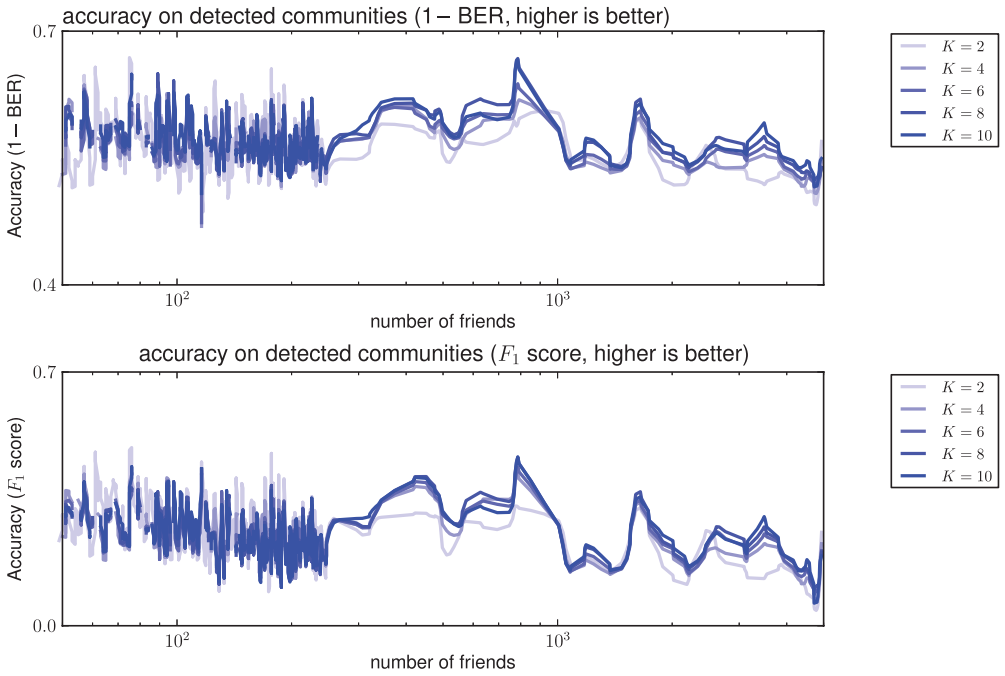


Fig. 11. Accuracy of our MCMC algorithm in terms of the BER (top) and the F_1 score (bottom). X-axis is log-scale.

scale to networks with millions of nodes. However, it is possible that circles could be more accurately predicted by exploiting relationships between the circles of multiple users. For example, if a user has a Stanford circle in his ego network, it is highly likely that users belonging to that circle will *also* have Stanford circles within their own ego networks. Alternately, if a Stanford *community* could be detected across the entire Facebook, Google+, or Twitter network, then a user's Stanford circle might simply be the intersection of his ego network with that community. Studying such models is an appealing avenue for future work.

We would also like to study semisupervised versions of our algorithm that make use of manually labeled data. Although most users presumably do not go to the effort of manually labeling their circles, it would be highly desirable to make use of the information from those few who *do*. If one user goes through the effort of labeling her 400 friends, this gives us *some* information that could potentially help to label the ego networks of those 400 users. Since each user's circles are likely to be very similar to those of her closest friends, we would simply need to update them for the friends the two users do not have in common. Of course, circles are meant to be *private* structures, meaning that such an algorithm would need to work in such a way that users could not discover how they were labeled in each other's circles.

Although the circles detected by our algorithm are similar to those that users manually labeled, we have not yet discussed those circles that our algorithm *failed* to predict. Critically, we only discover those circles that in some sense explain the graph structure. However, other circles may describe unique, yet important, relationships. For example, based on the common use cases of circles, a user may naturally consider a spouse to be a circle that contains only a single user. Our algorithm naturally fails to discover such circles, since they have no explanatory power in terms of the graph topology. Nevertheless,

such circles should be easy enough to detect based on node features, so it remains to build an algorithm capable of finding them. Rather than trying to encode such domain knowledge into the algorithm, it would be desirable to automatically *learn* features around which users commonly build circles. In our experience, the small amount of labeled data available to us was not enough to discover such patterns (which is why we opted for an *unsupervised* algorithm), although this issue may be ameliorated with more data.

We developed algorithms that scale to the largest ego networks that we encountered; however, we find that the best performance occurs on ego networks with up to a few hundred nodes but degrades significantly for networks with more than 1,000. It remains to be seen whether this is a shortcoming of our algorithm (due to the fact that optimization is more difficult for large networks), or whether the assumptions of our model simply break down at large scales. Our fundamental assumption that circles will be made up of close-knit groups of friends with common properties seems like a better fit to networks with at most a few hundred nodes.

We also found that performance on even the largest Facebook networks (i.e., more than 1,000 friends) was better than that obtained on small networks from Google+ and Twitter. This suggests that it is not merely the size of the networks that causes our model assumptions to become violated but rather the very nature of the networks themselves (in addition to the differences in the ground truth already mentioned). Naturally, a circle containing members of the same squash team (as we find on Facebook) is fundamentally different from a circle containing presidential candidates (as we find on Google+). It remains to design a circle detection algorithm that is tailored for networks with asymmetric following relationships.

10. CONCLUSION

Circles allow us to organize the overwhelming volumes of data generated by our personal social networks, although they are laborious to construct manually. We have designed an algorithm to automatically detect circles in ego networks, which we evaluated on a dataset of 1,143 ego networks and 5,541 ground-truth circles obtained from Facebook, Google+, and Twitter. In such data, we find circles that are disjoint, overlapping, and hierarchically nested, and we design our model with such behavior in mind. Our model is unsupervised but can also make use of weakly labeled data that may be available in practice. Experiments reveal that social circles can be accurately detected using a combination of both network and profile information.

ACKNOWLEDGMENTS

This research has been supported in part by NSF IIS-1016909, CNS-1010921, CAREER IIS-1149837, IIS-1159679, ARO MURI, DARPA XDATA, DARPA GRAPHS, ARL AHPCRC, Okawa Foundation, Docomo, Boeing, Allyes, Volkswagen, Intel, Alfred P. Sloan Fellowship, and the Microsoft Faculty Fellowship.

REFERENCES

D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, and J. Zachariah. 2008. Online models for content optimization. In *Neural Information Processing Systems*.

Y.-Y. Ahn, J. Bagrow, and S. Lehmann. 2010. Link communities reveal multiscale complexity in networks. *Nature*.

E. Airoldi, D. Blei, S. Fienberg, and E. Xing. 2008. Mixed membership stochastic blockmodels. *Journal of Machine Learning Research*.

R. Andersen and K. Lang. 2006. Communities from seed sets. In *WWW*.

R. Balasubramanyan and W. Cohen. 2011. Block-LDA: Jointly modeling entity-annotated text and entity-entity links. In *SIAM International Conference on Data Mining*.

E. Boros and P. Hammer. 2002. Pseudo-boolean optimization. *Discrete Applied Mathematics*.

J. Chang and D. Blei. 2009. Relational topic models for document networks. In *International Conference on Artificial Intelligence and Statistics*.

J. Chang, J. Boyd-Graber, and D. Blei. 2009. Connections between the lines: Augmenting social networks with text. In *Knowledge Discovery and Data Mining*.

H. Chen and R. Karger. 2006. Less is more: Probabilistic models for retrieving fewer relevant documents. In *Special Interest Group on Information Retrieval*.

Y. Chen and C. Lin. 2006. *Combining SVMs with Various Feature Selection Strategies*. Springer.

K. El-Arini, G. Veda, D. Shahaf, and C. Guestrin. 2009. Turning down the noise in the blogosphere. In *Knowledge Discovery and Data Mining*.

Scott L. Feld. 1981. The focused organization of social ties. *American Journal of Sociology.*

Mario Frank, Andreas P. Streich, David Basin, and Joachim M. Buhmann. 2012. Multi-assignment clustering for Boolean data. *Journal of Machine Learning Research.*

Steve Gregory. 2010a. Finding overlapping communities in networks by label propagation. *New Journal of Physics.*

Steve Gregory. 2010b. Fuzzy overlapping communities in networks. *CoRR* abs/1010.1523.

P. Hammer, P. Hansen, and B. Simeone. 1984. Roof duality, complementation and persistency in quadratic 0-1 optimization. *Mathematical Programming.*

M. Handcock, A. Raftery, and J. Tantrum. 2007a. Model-based clustering for social networks. *Journal of the Royal Statistical Society Series A.*

Mark S. Handcock, Adrian E. Raftery, and Jeremy M. Tantrum. 2007b. Model-based clustering for social networks. *Journal of the Royal Statistical Society.*

M. B. Hastings. 2006. Community detection as an inference problem. *Physical Review E.*

David Haussler. 1999. *Convolution Kernels on Discrete Structures*. Technical Report. University of California at Santa Cruz.

Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. 2002. Latent space approaches to social network analysis. *Journal of the American Statistical Association*.

S. Johnson. 1967. Hierarchical clustering schemes. *Psychometrika.*

D. Kim, Y. Jo, L.-C. Moon, and A. Oh. 2010. Analysis of Twitter lists as a potential source for discovering latent characteristics of users. In *CHI*.

P. Kohli and P. Torr. 2005. Efficiently solving dynamic Markov random fields using graph cuts. In *International Conference on Computer Vision*.

Vladimir Kolmogorov and Carsten Rother. 2007. Minimizing nonsubmodular functions with graph cuts: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

P. Krivitsky, M. Handcock, A. Raftery, and P. Hoff. 2009. Representing degree distributions, clustering, and homophily in social networks with latent cluster random effects models. *Social Networks.*

Andrea Lancichinetti and Santo Fortunato. 2009a. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E.*

A. Lancichinetti and S. Fortunato. 2009b. Community detection algorithms: A comparative analysis. arXiv:0908.1062.

Andrea Lancichinetti, Santo Fortunato, and Janos Kertesz. 2009. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics.*

P. Lazarsfeld and R. Merton. 1954. Friendship as a social process: A substantive and methodological analysis. In *Freedom and Control in Modern Society*.

J. Leskovec, K. Lang, and M. Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *WWW*.

Y. Liu, A. Niculescu-Mizil, and W. Gryc. 2009. Topic-link LDA: Joint models of topic and author community. In *International Conference on Machine Learning*.

D. MacKay. 2003. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

J. McAuley and J. Leskovec. 2012. Learning to discover social circles in ego networks. In *Neural Information Processing Systems*.

M. McPherson. 1983. An ecology of affiliation. *American Sociological Review.*

Miller McPherson, Lynn Smith-Lovin, and James M. Cook. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology.*

Aditya Menon and Charles Elkan. 2010. A log-linear model with latent features for dyadic prediction. In *International Conference on Data Mining*.

A. Menon and C. Elkan. 2011. Link prediction via matrix factorization. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*.

A. Mislove, B. Viswanath, K. Gummadi, and P. Druschel. 2010. You are who you know: Inferring user profiles in online social networks. In *International Conference on Web Search and Data Mining*.

P. Nasirifard and C. Hayes. 2011. Tadvise: A Twitter assistant based on Twitter lists. In *SocInfo*.

M. Newman. 2006. Modularity and community structure in networks. *In Proceedings of the National Academy of Sciences.*

M. E. J. Newman. 2003. Fast algorithm for detecting community structure in networks. *Physical Review E.*

M. E. J. Newman. 2004. Detecting community structure in networks. *European Physical Journal B.*

M. E. J. Newman and G. T. Barkema. 1999. *Monte Carlo Methods in Statistical Physics*. Oxford University Press.

Jorge Nocedal. 1980. Updating quasi-Newton matrices with limited storage. *Mathematics of Computation.*

G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. 2005. Uncovering the overlapping community structure of complex networks in nature and society. *Nature.*

M. A. Porter, J.-P. Onnela, and P. J. Mucha. 2009. Communities in networks.

E. Ravasz and A.-L. Barabási. 2003. Hierarchical organization in complex networks. *Physical Review E.*

C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. 2007. Optimizing binary MRFs via extended roof duality. In *Computer Vision and Pattern Recognition*.

S. E. Schaeffer. 2007. Graph clustering. *Computer Science Review.*

Georg Simmel. 1964. *Conflict and the Web of Group Affiliations*. Simon and Schuster.

J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. 2011. *The Anatomy of the Facebook Social Graph*. Preprint.

S. V. N. Vishwanathan and Alexander J. Smola. 2002. Fast kernels for string and tree matching. In *Neural Information Processing Systems*.

C. Volinsky and A. Raftery. 2000. Bayesian information criterion for censored survival models. *Biometrics.*

D. Vu, A. Asuncion, D. Hunter, and P. Smyth. 2011. Dynamic egocentric models for citation networks. In *International Conference on Machine Learning*.

S. Wu, J. Hofman, W. Mason, and D. Watts. 2011. Who says what to whom on Twitter. In *WWW*.

J. Yang and J. Leskovec. 2012. Community-affiliation graph model for overlapping community detection. In *International Conference on Data Mining*.

T. Yoshida. 2010. Toward finding hidden communities based on user profiles. In *ICDM Workshops*.

J. Zhao. 2011. Examining the evolution of networks based on lists in Twitter. In *International Conference on Internet Multimedia System Architectures and Applications*.

Received October 2012; accepted August 2013