Inferring Social Circles in Online Networks via a Hybrid Approach

Sepehr Abbasi Zadeh[1], Zeinab Abbassi[2], Pooyan Ehsani[3], Vahab Mirrokni[4], Ali Shameli[1], and Hadi Yami[1]

[1] Department of Computer Engineering, Sharif University of Technology
[2] Department of Computer Science, Columbia University
[3] Department of Mathematics, University of Tehran
[4] Google Research, New York
{sabbasizadeh,yami,shameli}@ce.sharif.edu
pooyan.ehsani@ut.ac.ir
mirrokni@google.com,zeinab@cs.columbia.edu

Abstract. With the number of friends growing in online networks, building tools that help users discover their social circles automatically or suggest friends to be added to specific existing circles are of practical importance. To this end, we develop clustering and recommendation algorithms which outperforms the current state-of-the-art algorithms while considering node features as well as the structure of the network. In particular, we first develop a classifier that can be used to suggest friends to be added to specific social circles. We then use this classifier for recommending new social circles to a user. At last, we introduce a new method for deciding on the number of clusters by selecting the most representative subset of discovered circles, which can be used in other similar problems in the literature. We perform experiments on various social networks to illustrate the accuracy and performance of our method.

1 Introduction

As more people join online social networks, users' friends lists grow and become more diversified. Therefore, many online social networks provide a feature that allows their users to classify their friends into social communities, also known as social circles or lists (we will refer to them as circles hereafter). This feature makes it possible for users to share their posts with the right people, filter what they want to see in their streams and in general organize their networks better. However, currently online social networks expect users to create these circles manually and do not provide this feature automatically. This can be time-consuming if it is done on an existing set of friends.

Another feature that is commonly provided in social network platforms is friend suggestion. It will be helpful to suggest users to specific circles. As a result, building tools that help users discover their social circles automatically or suggest friends to add to specific existing circles are of practical importance. To achieve this goal, clustering and recommendation algorithms need to be developed. These algorithms should take the structure of the social network and other available features into account. Most of the current clustering methods are either solely based on the structure of the graph or are very slow and are not scalable for large social networks. In this paper, we introduce a new scalable technique to achieve these goals. In particular, we develop a method that outperforms the current state-of-the-art algorithms while considering node features as well as the structure of the network. We first develop a classifier that can be used to suggest friends to be added to specific social circles. We then use this classifier for recommending new social circles to a user.

Our social circle discovery system is a multi-stage clustering algorithm that runs as follows: It first identifies a set of cores for the circles, then it expands each core using our circle classifier to generate a set of candidate circles, and finally it selects a subset of these circles as the final output. Experiments on various social networks illustrate the accuracy and performance of our method. Here we elaborate on our contributions in this paper.

1.1 Our Contributions

The algorithm presented in this paper has two main components. In the first part, we design a classification algorithm based on Naive Bayes which enables us to assign a user to multiple circles. In particular, given a user (which we refer to by the term *"actor."*) and his friends, our goal is to assign a new user to his circles. In order to do this effectively, we use both node features (such as career, location and education) and the network structure. We model each individual by a feature vector, i.e., a vector which represents that individual's set of friends as well as his features. Circles are user specific and each user organizes his personal network of friends independently from all other users to whom he is not connected with. This means that we can formulate the problem of circle detection as a clustering problem on his social network. In other words, the goal is to find a set of potentially overlapping clusters within the actor's social network. Our method supports overlapping circles by using a classifier to add extra nodes to each cluster. For clustering, our goal is to find clusters containing similar users while reducing the resemblance between different clusters. We achieve this goal by using a network specific algorithm to find a set of cores within the given network. Subsequently, we use our classifier in order to add extra nodes to each core and thus forming our final set of candidate clusters. We have also designed a fast approximation algorithm based on the current existing methods to choose the best subset of clusters within a set of candidate clusters which best represents a clustering for the actor's social network. We evaluate our method on the Facebook, Google+ and Twitter data sets (from SNAP) and the Kaggle training set from the "Learning Social Circles in Networks" competition. Our experimental results have shown that our method outperforms many state-of-the-art algorithms used to find communities or clusters within social networks.

1.2 Related Work

In this paper, we study the problem of discovering social circles. This problem was first suggested by J. Leskovec and J. McAuley in 2012 [15]. Even though social circles and communities are different concepts, e.g., a community is centered on a specific interest/topic, while a social circle can be created upon different social relationships (different circles containing family, friends, colleagues, etc.), our work is also closely related to community detection algorithms [12, 23, 13, 20, 18]. Various methods have been used to detect communities – the survey gathered by [20] provides a thorough review of these methods. Our algorithm supports overlapping as well as hierarchically nested social circles. While most of the previous researches in this area merely focus on finding disjoint circles or communities, many publications suggest that nowadays, individuals tend to form overlapping or even hierarchically nested social circles while organizing their friends in social networks [16, 2, 11, 7, 21].

Some old methods use only node features [8] or graph structure [1] in order to discover social circles in networks. Some other approaches work as follows: they assign an importance score to each edge of the graph, and then obtain the social communities by adding the edges to the graph one by one in the decreasing order of the edge values [26] or by removing edges from the initial graph in the increasing order of the edge values [5]. These algorithms result in a hierarchical partitioning of the nodes, referred to as a *dendrogram*.

However, recent methods consider both node features and graph structure to obtain more reliable and accurate results. For instance, the method proposed in [16] models circle memberships as latent variables and similarity between different individuals as a function of their social features. This paper suggests an unsupervised procedure to learn which dimensions of profile similarity lead to denser circles.

Our method is similar to [28] and [27] in the sense that we find a set of champion sets in the primary step of our algorithm, and then add extra nodes to each set using a classifier to form our final set of candidate clusters. To find our champion sets, we use some of the classical clustering methods such as identifying clique-like components or *K-Means*.

Various methods have been studied for the problem of estimating the number of clusters in a given data set [24, 9, 25, 22, 4]. However rather than explicitly trying to find the number of clusters in the network, we use the method proposed in [3] to approximate the score of a clustering which implicitly determines the best number of clusters among the set of candidate clusters.

The rest of this paper is organized as follows. We first design a classifying algorithm in Section 2. We use this method later on in Section 3 to derive our clustering algorithm. Finally, we evaluate our methods in Section 4.

2 Classifier

Actors create social circles to organize their communications and to share posts with the right set of people. Therefore, suggesting a user to be added to a specific circle is of practical importance. In this section we develop a classifier which recommends friends to specific circles. Moreover, we will also use this classifier in order to develop a special clustering algorithm (for discovering circles) in section 3.

Feature Vectors We model each user as a binary vector, called "feature vector". An actor such as u has a dimension in his vector which indicates his connectivity with each of his friends. We refer to these dimensions as *friendship dimensions*. In addition, u has a dimension in his vector for each feature f which is presented in the feature set of the actor. We refer to these dimensions as *feature dimensions*. The reason we do not consider all features of u in his vector is that usually, when constructing his circles, the actor only considers the features he shares with his friends and ignores other features.

Based on our experiments, another useful idea in suggesting a friend to a circle is to only consider friendship dimensions of the members of that particular circle and to ignore the remaining friendship dimensions. We have run our algorithms on the data sets considering this idea – the results can be seen in Fig. 1. These vectors are advantageous for us since they contain some new features about the structure of the graph. Fig. 2 provides an illustration of feature vectors and the filtration process.

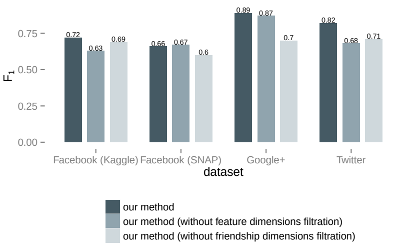

Fig. 1: Comparison between different versions of our classifier.

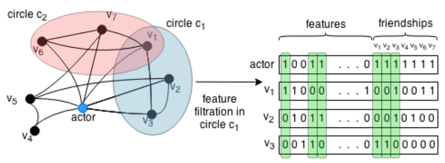
circle c2
circle c1
v7
v6
v1
v2
v3
v4
v5
actor
feature filtration in circle c1
features
friendships
v1 v2 v3 v4 v5 v6 v7
actor 1 0 0 1 1 . . . 0 1 1 1 1 1 1 1
v1 1 1 0 0 0 . . . 1 0 0 1 0 0 1 1
v2 0 1 0 1 1 . . . 0 0 0 1 0 1 0 0
v3 0 0 1 1 0 . . . 0 1 1 0 0 0 0 0

Fig. 2: The green columns represent the selected dimensions in circle c_1. Each dimension is selected if it either refers to a feature which the actor has or refers to an individual (in friendship dimensions) which is present in the circle

Classifying Method Our method for the classifier is based on the Naive Bayes classifier. In this classifier, for a given query $q =< f_1, f_2, ..., f_f >$ (a feature vector belonging to a specific individual) and circle C_k we have the following property:

$$P(C_k|q) = P(C_k)P(q|C_k) = P(C_k)P(< f_1, ..., f_f > |C_k) = P(C_k)\prod_{i=1}^{f} P(f_i|C_k).$$

Roughly speaking, a Naive Bayes classifier makes an independence assumption, i.e., it assumes that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. We indicate the probability of q being included in C_k by $p^+_{C_k}$. Furthermore, we also calculate the probability of q being excluded from C_k similarly and will call this probability $p^-_{C_k}$. Afterwards, we place q in C_k, *iff* $p^+_{C_k} \geq p^-_{C_k}$. This formulation clearly is not an innovation, but it achieves accurate results by the help of the previous mentioned feature vectors.

Time Complexity The pre-processing part of our algorithm takes $\mathcal{O}(nfk^2)$ in which n is the number of individuals, f is the number of features and k is the number of circles. In addition, to decide whether each individual belongs to a specific circle, we require $\mathcal{O}(f)$ operations. Therefore, to find the set of circles to which an individual belongs, we need $\mathcal{O}(fk)$ operations.

3 Clustering

In this section, we discuss the discovery of clusters or communities in a social network.

Problem Definition Given an actor's network, we would like to cluster the friends of the actor such that our clustering is as close to a set of social circles in the ground truth as possible. Moreover, it should be emphasized that the clusters can be overlapping, as well as hierarchically nested.

We have developed an accurate and fast method for clustering in actor's networks. From now on we refer to this method as *CCC*. In the first step of our algorithm, we choose a few sets of champions each of which represent the core of a specific cluster. Subsequently, we use our classifier in order to add additional users to these champions. Finally, we select a subset of champions which best represents a clustering for the actor's network.

3.1 Core finder

The set of initial cores that we provide for our clustering method plays a crucial role in the precision of our results. We expect that individuals in each circle form a densely connected network [19]. Also, based on our experiments, it is usually better if the selected cores are as disjoint as possible. We propose three methods which satisfy these criteria.

Finding Cliques. In this method, we find a set of cliques which represent our cluster set. In order to find optimal cliques, we assign a centrality index to each individual and then we start finding maximal cliques around vertices with high centrality indices. After finding each clique c, we remove the vertices that appear in c and repeat this procedure until no vertices are left. This guarantees that the discovered cliques are disjoint. For instance, a reasonable choice for centrality would be the PageRank.

Finding Dense Sub-graphs. This method is very similar to the previous one but rather than finding maximal cliques, in each step, we find the maximum density sub-graph of the actor's network. The density of a graph is the ratio of the number of edges to the number of vertices of the graph. Given an undirected graph, we can find its maximum density sub-graph in polynomial time [6] in contrast to the maximum clique problem which is NP-Hard.

Finding Cores Using *k*-means. Given a set of multi-dimensional vectors, k-means clustering aims to find a clustering of these vectors into k sets ($k \leq n$), so as to minimize the within-cluster sum of squares (WCSS)[14]. In order to solve this problem, we use an iterative algorithm which converges rapidly to a local solution[14]. In our experiments, we produced a set of seed cores by running the k-means algorithm for different values of k (5, 10, 15, 20, and 25) and stacking up the produced clusters as our final set of cores. This means that we have 75 cores to apply our classification method on.

3.2 Circlizer Method

In this section, we explain how to choose a subset of clusters S from a set of candidate clusters A as the desirable clustering solution for an actor's network. Intuitively, a desirable group of clusters consists of clusters which have dense friendship, and feature patterns internally and sparse friendships and features externally [17]. We measure these criteria by assigning a double value $Score_S$ to each set $S \subseteq Powerset(A)$. The higher this value, the more in alignment is set S with our described criteria. Eventually, we will choose the set with highest *score* as the final result.

Let $FeatureCount_f$ be the total number of individuals which have feature f. Moreover, let $CircleFeatureCount_{c,f}$ be the number of individuals in cluster c which have feature f. We will assign a vector $CVector_c$ to each cluster $c \in A$ in which $CVector_c[f]$ shows the importance of feature f in cluster c. Let $CVector_c[f]$ be $CircleFeatureCount_c[f]/FeatureCount[f]$. We will divide each vector $CVector_c$ by its length and will refer to the new normalized vector by $NormalCVector_c$. We define the $InternalDensity$ of S as follows:

$$InternalDensity_S = \frac{1}{|S|} \sum_{c \in S} \frac{\sum_{i \in c} \sum_{f \in F} PVector_i[f] \times NormalCVector_c[f]}{|c|}.$$

In which, F is the set of all the features and $PVector_i[f]$ is an integer set to 1 *iff* individual i has feature f and to 0 otherwise. The value of $InternalDensity_S$ becomes large as the clusters of S become more internally dense on average. Furthermore, to model external sparsity of clusters in S, we define $ExternalDistance_S$ as follows:

$$ExternalDistance_S = \frac{\sum_{c_1, c_2 \in S, c_1 \neq c_2} d(NormalCVector_{c_1}, NormalCVector_{c_2})}{|S| \times (|S| - 1)},$$

in which $d(U, V)$ is a metric distance between vectors U and V. In addition, we would also prefer our clustering to cover a reasonable portion of individuals. We define $Coverage_S$ as the fraction of individuals which participate in at least one of the clusters in S. Finally, we define $Score(S)$ as follows:

$$Score(S) = \alpha Coverage_S + \beta InternalDensity_S + \gamma ExternalDistance_S,$$

in which α, β and γ are trained constant variables regarding to the social network we are processing. For cases like Facebook, we observed that $Coverage$ can be very important, because most users tend to assign most of their friends to at least one circle in the process of forming their circles.

To find the set with maximum score, we would have to calculate $Score(S)$ for all $S \in Powerset(A)$ which can take a lot of time considering the burgeoning size of social networks. To find the set $S \in Powerset(A)$ with maximum score, we use the greedy algorithm proposed in [3]. They considered the Max-Sum Diversification problem and proposed a greedy algorithm of factor two to solve it. More precisely, they solve the following problem in polynomial time using a greedy approximation algorithm of factor two.

Max-Sum Diversification Let U be the underlying ground set (here it is the set of candidate clusters), and let $d(.,.)$ be a metric distance function on U. For any subset of U, let $g(.)$ be a non-negative, submodular set function measuring the value of a subset. Given a fixed integer p, the goal of the problem is to find a subset $S \subseteq U$ that:
maximizes $h(S) = g(S) + \gamma \sum_{u,v \in S} d(u,v)$
subject to $|S| = p$.
where $g(S) = \frac{p(p-1)}{2} \alpha Coverage_S + \frac{p-1}{2} \beta InternalDensity_S$.

After solving the above problem for all p, we choose the set S with the maximum score as the final result. It can be seen that $\frac{2h(S)}{p(p-1)} = Score(S)$ and therefore, by finding the set S that maximizes the function $h(S)$, we would be able to maximize $Score(S)$.

It just suffices to show that $g(S)$ is normalized, monotone and submodular. The normality of $g(S)$ is clear, since for $S = \emptyset$, $g(S) = 0$. In addition, one can easily observe that $g(S)$ is monotone because for any $u \in A \setminus S$, $g(S \cup u) \geq g(S)$. To show that $g(S)$ is submodular, we would have to prove that for any $S, T \subseteq A$, $S \subseteq T$ with $u \in A$, $g(T + u) - g(T) \leq g(S + u) - g(S)$. After factoring and discarding the constant factors, we have to show:

$$Coverage_{T+u} - Coverage_T \leq Coverage_{S+u} - Coverage_S$$

and we know that coverage function is a submodular function [10]. Thus we conclude that $g(S)$ is a submodular function; therefore, the greedy algorithm proposed in [3], which maximizes the marginal gain on fucntion g in each iteration, provides a polynomial-time 2-approximation solution for finding the set with the maximum score. This new method is fast enough to generate approximated results on huge data sets and it can easily be customized for different demands of scoring systems. From this point of view, this is a noteworthy addition to the literature on finding the number of the clusters to be chosen. Although this method approximately selects the most representative set of clusters, it can be verified through our experiments (Fig. 7) that it returns mostly identical results as the non-approximated (accurate) method.

Time Complexity The time complexity of our method strongly depends on the core finding algorithm which we deploy. Using the densest sub-graph algorithm, we would be able to achieve polynomial time complexity while not losing much accuracy in our results. We explain the time complexity of our algorithm in case of using densest sub-graph algorithm as a means to discover our cores. Finding the densest sub-graph in a network has a polynomial time algorithm. For instance, it is possible to find the densest sub-graph in $\mathcal{O}(M(n, n+m)logn)$ using Goldberg's algorithm in which $M(v, e)$ is the time required to find a minimum capacity cut in a network with v vertices and e edges. In order to find a number of dense sub-graphs, we iteratively find the densest sub-graph $G(V, E)$ of our actor's network and remove the edges of the network incident with V. It is clear that after each iteration, the number of non-isolated vertices of the network reduces. So in the worst case, our algorithm would run for $|V|$ iterations which leads to time complexity $|V| \times \mathcal{O}(M(n, n+m)logn)$.

4 Evaluation

In this section we describe the results of our experiments with different data sets and the evaluation metrics that we have used in them[5].

Data set In order to evaluate the performance of our methods we used two publicly available data set collections. One from the *SNAP collection*[6], consisting of three data sets from Facebook, Google+ and Twitter, and the other from the *Kaggle competition*[7] data set. Kaggle's data set is identical to Facebook data set in structure. We have summarized these data sets in Table 1.

Table 1: Structure of Data sets

Name	Actors	Nodes	Edges	Circles	Avg. Appearance of Each Person in Circles
Facebook (SNAP)	10	4039	88234	193	1.05
Google+	130	107614	13673453	465	0.59
Twitter	973	81306	1768149	4065	0.63
Facebook (Kaggle)	60	14579	188585	592	1.17

Metrics To measure the performance of our methods, we report the $F_1 = 2.\frac{precision.recall}{precision+recall}$ score of each experiment. A desirable property of the F1 measure is that for completely wrong, random, and completely correct recommendations, its values are 0, 0.5, and 1 respectively [8]. As a result, the possible range of values for F_1 measure spans the whole range between zero and 1, and therefore, it is suitable for distinguishing quality of different algorithms more clearly.

[5] For the purpose of reproducibility, we provide the code at (hided for obeying double-blind review policies)

[6] http://snap.stanford.edu/data/#socnets

[7] http://www.kaggle.com/c/learning-social-circles/data

[8] This is in contrast to the $1 - BER$ measure (used in [16]) whose values for these cases could be around 0.5, 0.75, and 1 respectively.

4.1 Classifying Experiments

In order to evaluate our classifying algorithm described in Section 2, similar to the evaluation method proposed in [16], we remove a single user u from the ground truth circles. Afterwards, the goal is to discover the set of circles to which node u belongs. Next, we calculate the F_1 score between the set of ground truth circles to which u belongs, and the set of predicted circles.

We have compared our classifying method with the one proposed in [16] as well as the Support Vector Machine (SVM) method in Fig. 3 for 10 random choices of the user u. Since our method was fast enough, in the same figure, we have also obtained the average F_1 score for all the users in the network. Based on the results of the experiments, our algorithm outperforms all the modern rival algorithms based on the quality of the output as well as the time complexity, because the most precise algorithms are usually very slow.

4.2 Clustering Experiments

We evaluate our method on ground-truth data by examining the correspondence between our final clusters and the ground truth using F_1 score. However, since there is no one (unique) correspondence between our clusters C and the ground truth clusters $\bar{C}$, we find the best matching between these two solutions by maximizing: $\max_{f:C\to\bar{C}} \frac{1}{|f|} \sum_{c\in dom(f)} F_1(c, f(c))$, where f is a partial correspondence between C and $\bar{C}$, i.e., in the case that $|C| \leq |\bar{C}|$, every cluster $c \in C$ must be matched to a $\bar{c} \in \bar{C}$. Since the number of predicted clusters in all our empirical results was always less than or equal the number of ground truth clusters, the other case ($|C| > \bar{C}$) was not a concern in our evaluations.

As stated earlier, after finding a set of candidate clusters S using the core finder algorithm described in Section 3.1 and applying the classifying method on each of them, we apply the Circlizer method in order to choose a subset of clusters from S which best represents a clustering of the actor's network. It should be noted that while our Facebook data sets (Kaggle and SNAP) are standard and hand-labeled, our Google+ and Twitter data sets are not well-established. First of all, in contrast to the Facebook data sets, these data sets are gathered by crawling their networks. In addition, the structure of the Facebook data set is fundamentally different from Google+ in a sense that in Facebook, friendships are mutual while in Google+, friendships are one sided. Moreover, the features assigned to twitter users are their used hashtags which in essence is totally different from the features proposed by Facebook and Google+ data sets.

Based on our experiments, we have observed that we should choose our clustering method based on the intrinsic characteristics of each data set. Some of the methods for core finding can perform very well on some data sets while producing weak results on other data sets. Fig. 4 illustrates the performance of the CCC algorithm based on different core finding methods for each data set.

In [16] they have considered eight other baselines for this experiment and their result is always the best. As a result, we just compare our algorithm with

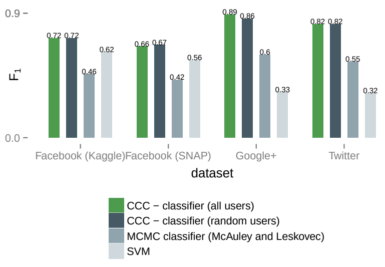

Fig. 3: Different methods of classifying. The green column illustrates the average F_1 score between the ground truth and the predicted set of community memberships for all the individuals (i.e., running our method on all friends of the actor).

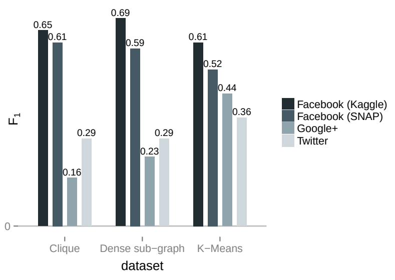

Fig. 4: Perfomance of CCC on Facebook (Kaggle and SNAP), Twitter and Google+ based on different core-finding methods.

their method. The F_1 score of our algorithm can be compared with this method in Fig. 5. In Fig. 6 we compare these two methods for all data sets. It is worth pointing out that our algorithm can find the optimum clustering of an actor **in less than a minute**, while the other baseline can only predict one circle in this time (on the same machine). And also it takes more than two hours to find nine circles for an actor on average (on Kaggle data set). Their proposed algorithm chooses the optimal number of circles using Bayesian Information Criterion (BIC). Accordingly they have to run their algorithm for all reasonable possible values of circles, which results in a running time much **more than two hours** for a single actor. We have found that by using cliques and dense sub-graphs as

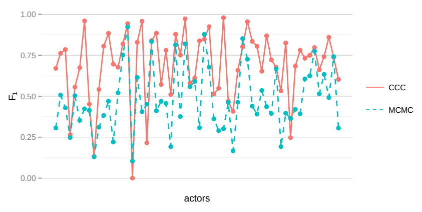

Fig. 5: Comparing the F_1 score of CCC with MCMC for different actors in Facebook (Kaggle) data set

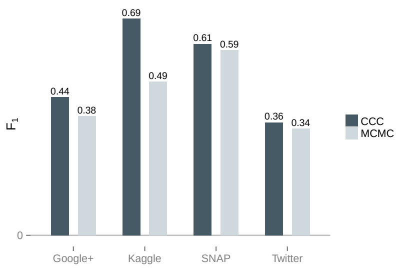

Fig. 6: Comparing the performance of CCC with the clustering method proposed by [16] on different data sets

our cores, we would be able to achieve promising results on the Facebook data sets (from Kaggle and SNAP). To find optimal cliques, we first find the PageRank of each node. Then in each step of our core finding algorithm, we try to find a maximum clique containing the node with the maximum PageRank. Then we remove the discovered clique from the network and continue this process until there are no more vertices left in the network. Afterwards, we apply our classification method to add extra nodes to each core and subsequently, we apply our Circlizer on the set of candidate clusters to find the optimal subset as the final result. Our Circlizer had three constants α, β and γ. For the Facebook data set, we found that setting $\alpha = 1, \beta = 1$ and $\gamma = 1$ produces good results. We have also compared the results of the 2-approximation Circlizer with the initial Circlizer method on the Facebook (Kaggle) data set using the same settings as the Facebook data set in (Fig. 7) while using cliques as our initial cores. Fig. 8 illustrates the step by step performance of the CCC method on all the actors of the Facebook (Kaggle) data set.

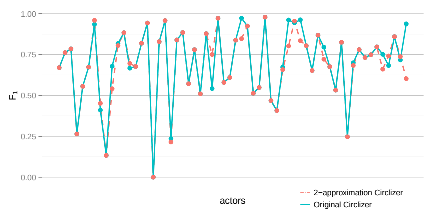

Fig. 7: Comparing the results of the 2-approximation Circlizer with the initial Circlizer for different actors in Facebook (Kaggle) data set

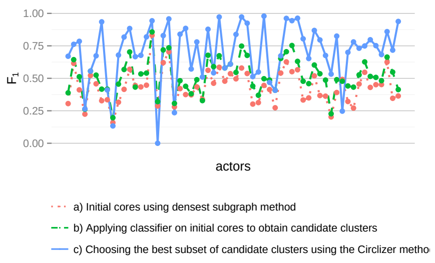

Fig. 8: Step by step performance of the CCC algorithm for different actors in Facebook (Kaggle) data set

By analyzing the ground truth clusters of Google+ and Twitter, we found that their data sets are different from Facebook in a sense that they are not necessarily internally dense or externally distant from each other. Based on our empirical results, using the k-means method as the core finder provides more accurate results. It should be noted that since there are many clusters of various sizes in the actor's network, we applied k-means using different values of k and stacked up all the discovered clusters as the set of our final cores. Afterwards, we applied the classifier method and finally the Circlizer method with $(\alpha, \beta, \gamma) = (1, 0.00, 0.00)$ in Google+ and $(1, 0.14, 1)$ in Twitter data set to obtain our final clustering.

5 Conclusion

In this paper, we propose a multi-stage algorithm to discover social circles over social networks. Our method takes advantage of both users' features and the structure of the network. Leveraging these two components, combined with machine learning and graph mining techniques, we manage to design a specialized social circle discovery system that outperforms previous methods. Moreover, our method has better computational complexity, and therefore, is more scalable and more efficient. While our data sets may be small, all of our existing implementation is easily applicable to huge data sets. We applied our algorithm on publicly available data sets, and did not have access to an online social network platform to run live experiments. It would be great to apply these algorithms in a real-world social network setting, and report results from live experiments.

References

1. Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.
2. J. Baumes, M. Goldberg, and M. Magdon-Ismail. Efficient identification of overlapping communities. In *Intelligence and Security Informatics*, pages 27–36. Springer, 2005.
3. A. Borodin, H. C. Lee, and Y. Ye. Max-sum diversification, monotone submodular functions and dynamic updates. In *Proceedings of the 31st symposium on Principles of Database Systems*, pages 155–166. ACM, 2012.
4. S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome biology*, 3(7):research0036, 2002.
5. M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.
6. A. V. Goldberg. *Finding a maximum density subgraph.* University of California Berkeley, CA, 1984.
7. S. Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, 2010.
8. S. C. Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967.
9. R. Kothari and D. Pitts. On finding the number of clusters. *Pattern Recognition Letters*, 20(4):405–416, 1999.
10. A. Krause and D. Golovin. Submodular function maximization. *Tractability: Practical Approaches to Hard Problems*, 3:19, 2012.
11. A. Lancichinetti and S. Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.
12. A. Lancichinetti and S. Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.
13. J. Leskovec, K. J. Lang, and M. Mahoney. Empirical comparison of algorithms for network community detection. In *Proceedings of the 19th international conference on World wide web*, pages 631–640. ACM, 2010.
14. D. J. MacKay. *Information theory, inference, and learning algorithms*, volume 7. Cambridge: Cambridge university press, 2003.
15. J. Mcauley and J. Leskovec. Learning to discover social circles in ego networks. *In Neural Information Processing Systems*, 2012.
16. J. Mcauley and J. Leskovec. Discovering social circles in ego networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(1):4, 2014.
17. N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan. Clustering social networks. In *Algorithms and Models for the Web-Graph*, pages 56–67. Springer, 2007.
18. M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133, 2004.

19. M. E. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.
20. M. A. Porter, J.-P. Onnela, and P. J. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.
21. E. Ravasz and A.-L. Barabási. Hierarchical organization in complex networks. *Physical Review E*, 67(2):026112, 2003.
22. S. Salvador and P. Chan. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 576–584. IEEE, 2004.
23. S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.
24. C. A. Sugar and G. M. James. Finding the number of clusters in a dataset. *Journal of the American Statistical Association*, 98(463), 2003.
25. R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423, 2001.
26. S. Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.
27. J. J. Whang, D. F. Gleich, and I. S. Dhillon. Overlapping community detection using seed set expansion. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2099–2108. ACM, 2013.
28. B. Yan and S. Gregory. Detecting communities in networks by merging cliques. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on*, volume 1, pages 832–836. IEEE, 2009.