Finding Friends in Hidden Communities

An Analysis on Structure in Social Networks

Pawan Paleja
University of California, San Diego
A15586236

Tae Kwang (Jason) Chung
University of California, San Diego
A14477899

ABSTRACT

As the world goes through this time of extreme isolation and disconnection, cultivating our social networks online is more important now than ever. One of the great strengths of these virtual networks is their ability to connect us with friends we might have lost or forgotten through rigorous data-mining and recommender systems. In this paper, we provide a case study of using community detection to predict friendships in a social network graph of connections and user-specified social circles. We compare two different models of community detection, disjoint partitioning and hierarchical clustering as well as multiple classification models for this predictive task. Finally, we provide results on dimensionality reduction and hyperparameterization on the models discussed.

KEYWORDS

Social Networks, Recommender Systems, Latent Variables, Networks

ACM Reference format:
Pawan Paleja and Tae Kwang (Jason) Chung. 2020. Finding Friends in Hidden Communities. In *Proceedings of Data Mining and Recommender Systems, Assignment 2, Fall 2020 (CSE 158),* 6 pages.
https://doi.org/

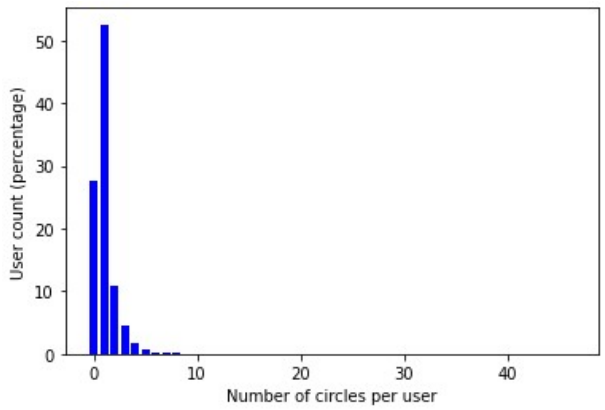

Figure 1: A visualization of the social network parsed from the dataset

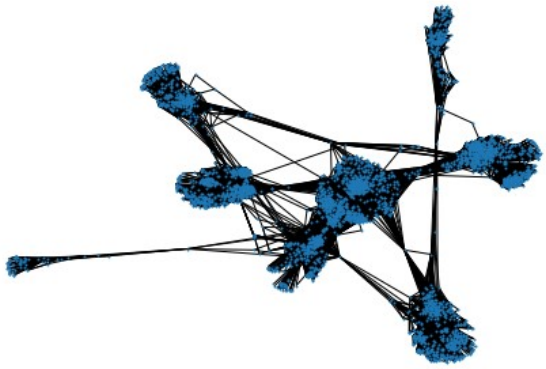

Figure 2: A visualization of the social network parsed from the dataset

1 DATASET

The dataset[1] we used, Social Circles dataset from Facebook, consists of node features, circles and ego networks. It also contains a list of all edges in the undirected social network graph where each edge represents a connection between two users. As we can see from Figure 2, much of the graph is clustered into regions of extremely high density. However, those regions are themselves connected to one another. The dataset also contained 193 'ground-truth' ego networks, where users self-identified groups of their social circles. However, as one can see from Figure 1, nearly 30% of users are not accounted for inside a given social circle. Furthermore, the total number of users found from the .circles and .edges files was 40 nodes less than the total number of nodes in the graph, which motivated us to leverage other graph clustering algorithms for a better performance from the model we choose.

2 PREDICTIVE TASK

2.1 Task

Our task is as follows: Given two users $user_1$ and $user_2$, predict whether or not they are friends. We model this as a link prediction problem, i.e. given a graph G on n vertices, remove uniformly at random a set of edges E'; then, given a query $e \in E(\bar{G})$ (where

[1] https://snap.stanford.edu/data/egonets-Facebook.html

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/

$E(H)$ denotes the edge set of H and $\bar{G}$ is the complement graph of G, correctly predict if $e \in E'$.

2.2 Evaluation

We test our models on an unseen list of edges that are labeled positive if they exist in the graph and labeled negative if they exist in the complementary graph (i.e. 1 if connection exists and 0 otherwise).

2.3 Assessing Validity

We predict these edges' labels and score based on the accuracy of the predictions, and examine which model provides the best accuracy. We will also look at the average *Precision*/*Recall* scores of our models to see if there is any noticeable skew to predicting positively or negatively for any given sample.

2.4 Baseline Models

2.4.1 Trivial Model. Our first model was very trivial model and it was inspired by the baseline metric we used in assignment 1, which is that given a query of two users $user_1, user_2$, predict that they ARE friends if either user is 'popular', i.e. if it has a high degree past some threshold t. Shown in Figure 3 is scoring following of our optimization of this threshold t, where the best threshold was chosen that gave the best accuracy on the validation set (in our analysis, the best value for t was given to be 100).

	Training	Validation	Test
TPR	0.545831	0.549886	0.545875
TNR	0.773825	0.777602	0.779027
FNR	0.454169	0.450114	0.454125
FPR	0.226175	0.222398	0.220973
Accuracy	0.659655	0.664098	0.662443
BER	0.340172	0.336256	0.337549
Average Precision	0.613691	0.61519	0.61567
Recall	0.545831	0.549886	0.545875

Figure 3: Scoring of model described in 2.4.1

2.4.2 A Slightly Better Baseline Model. Our next model was to use the dataset's 'ground-truth' data, that is the user reported social circles $C = [c_1, c_2, ..., c_k]$ that partition the graph into hierarchical communities of nodes in the network. For this, we built a classifier by extracting for each $(user_1, user_2)$ sample a feature v. We define v as:

$$v_i = 1 \text{ if } (user_1, user_2) \in E(S(c_i)) \text{ else } 0$$

where $E(S(H))$ denotes the edge set of the induced subgraph of G on the vertex set H. We then trained a Logistic Regression model on the $sample \times feature$ matrix extracted. Results obtained on the validation and test set can be seen in the figures below

2.5 Models To Use

We decided to use three models and compare them with themselves and with the baseline models, these models being: Logistic Regression model, Support Vector Classification model, and the Gaussian Naive Bayes model. We chose the Logistic Regression model because we are predicting a binary label from a set of independent feature, Support Vector Classification model because not only it is one of the most popularly used models on classifying a labeled data but also we expected the data to be separable due to the low nubmer of circles each user was in and Naive Bayes model with the Gaussian Naive Bayes algorithm for classification to address the potential overfitting issue and also because we expected out features to be categorical.[2]

	Training	Validation	Test
TPR	0.631563	0.634288	0.635857
TNR	0.982384	0.981335	0.98123
FNR	0.368437	0.365712	0.364143
FPR	0.017616	0.018665	0.01877
Accuracy	0.806707	0.808351	0.808532
BER	0.193026	0.192188	0.191456
Average Precision	0.798974	0.798335	0.799712
Recall	0.631563	0.634288	0.635857

Figure 4: Scoring of model described in 2.4.2

3 MODELS

3.1 Motivation

We realized from our second baseline test that there was high potential for the predictive capacity of social circles as latent variables for predicting edges between nodes in the graph. However it is likely that our model is weighed down by the fact that for many users, we simply do not have any circle to place them in at all, as we observed above in Section 1. Thus, our task then becomes to accurately model these latent variables ourselves and then use them to predict links in the graph. This model for feature categorization makes sense intuitively as well. For many of us, social interaction comes from the groups we associate with - for example: clubs, schools, jobs, hobbies, etc. - thus if two individuals share an overlap in one or more of these social circles, intuition would point to predicting that they would themselves share an edge.

3.2 Latent Variable Extraction

We attempted and compared two different types of community detection philosophies: Disjoint and Hierarchical. For each, after clustering the graph into communities $C = [c_1, c_2, ..., c_k]$, we applied the same feature extraction method as above, where feature v is defined again as:

$$v_i = 1 \text{ if } (user_1, user_2) \in E(S(c_i)) \text{ else } 0$$

3.2.1 Disjoint Partitioning. Partitioning the graph into disjoint communities means that any given node (representative of a user) can only be represented by one community - i.e. social circle. Obviously, this is not accurate to most individuals, however, the particular community detection method we chose to represent this philosophy - community detection by label propagation[6] - reported good accuracy to the ground-truth social circles of graph representations of social networks and boasted near-linear complexity in the number of edges in the graph, which is useful for scalability. The algorithm is described in Algorithm 1.

As one can clearly see from Algorithm 1, this allows for communities to be detected beyond the strict metric of k-cliques, where each nodes is connected to every other node in the community,

[2] For further rationalization, see the corresponding sections in 3.4

Algorithm 1: labProp

Result: Set of communities C of nodes
$G = (V, E)$ Initialize C, where $c_i = \{v_i \in V\}$, so that each vertex is in its own community **while** $v \in c$ *is in a different community than the majority of its neighbors* **do**
  For each $v \in V$, in random order, move v to the community c' the majority of its neighbors are in, breaking ties at random.
end

but still enforces that each community is more connected to its members than to the outside. This relaxation is particularly useful because our model specifically removes edges where one might have had a small clique, thus making it potentially more robust than this method.

	Training	Validation	Test
TPR	0.977209	0.977717	0.976881
TNR	0.866629	0.86573	0.867298
FNR	0.022791	0.022283	0.023119
FPR	0.133371	0.13427	0.132702
Accuracy	0.922003	0.92155	0.922094
BER	0.078081	0.078277	0.07791
Average Precision	0.871581	0.87012	0.871624
Recall	0.977209	0.977717	0.976881

Figure 5: Scoring of Disjoint Partitioning with Logistic Regression

3.2.2 Hierarchical Clustering. On the other hand, a hierarchical clustering allows for nodes to be represented by multiple communities, which more closely relates to the latent structure we are trying to represent. Initially, we attempted to use the algorithm given in [1]. This algorithm recursively partitions the graph into connected components by repeated removal of edges that are predicted to be 'between' clustered communities. However, this proved to be computationally intractable due to the the complexity itself being $O(|E|^2|V|)$ and due to memory usage of the networkx subgraph object which is required for this recursion. Instead, we used an algorithm based on clustering vertices by their closeness according to several short random walks [5]. We chose this because it boasted an average case complexity of $O(|V||E|)$, which is considerably better as well as high accuracy in uncovering community detection in reference to the ground-truth it was evaluated.

3.3 Data Parsing

The total data is the union of the list of positive edges in the original graph (see Figure 1) and a random sample of the list of negative edges from the complementary graph so that the ratio of positive and negative edges is 1:1. We label each positive edge with 1 and the negative edge with 0. We then randomly shuffle the edges and split the total data into training, validation and test data with each respective set representing 50%, 25%, and 25% of the total data.

3.4 Classification

3.4.1 Naive Bayes. The Naive Bayes classifier applies the Bayes theorem with the naive assumption of conditional independence between every pair of features given the value of the class variable. We chose the Gaussian Naive Bayes model for classifying the dataset because it reduces overfitting and also our features are categorical. By looking at the result table from above, we can see that the average is approximately .667, which is not an improvement from our baseline models.

	Validation
TPR	0.337244
TNR	0.994486
FNR	0.662756
FPR	0.005514
Accuracy	0.666886
BER	0.334135
Average Precision	0.662135
Recall	0.337244

Figure 6: Scoring of Naive Bayes Classifier

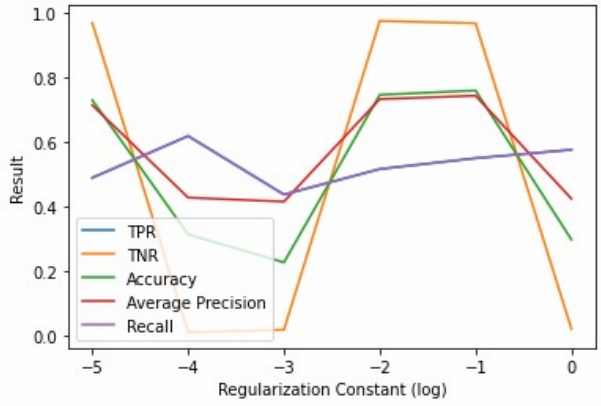

Figure 7: Scoring of Support Vector Classifier

3.4.2 Support Vector Classification. The Support Vector Machine was chosen because it was one of the most commonly used models on classifying a labeled data of moderately large size ($< 100k$). Also, because our data was expected to be separable (due to the fact that we noticed a large number of users in 0-3 social circles in the ground-truth data), we decided to utilize the Support Vector Classification algorithm with a relatively low iteration upper bound. By looking at the result graph from above where the x-axis represents the different regularization constants ranging from $1.5e-10$ to 1.0, it is obvious that the model is rather unstable and the accuracy shows an unpredictable pattern. Because of this, it is clear that this model is not the most ideal for our dataset.

3.4.3 Logistic Regression. A logistic regression model was chosen mainly because we are essentially predicting a binary label from a list of independent features. By looking at the result graph from above, we can see that the accuracy is above 0.93 with regularization constant of 1.0. Clearly, the logistic regression model outperformed the other two models (Gaussian Naive Bayes, Support Vector Classifier), but we realized that we could address the issue of scalability / overfitting.

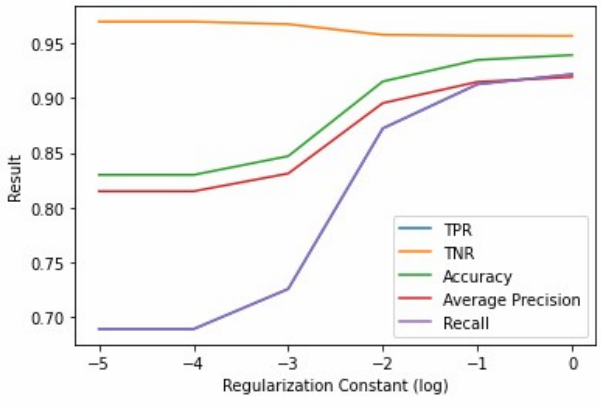

Figure 8: Scoring of Logistic Regression

3.5 Optimization

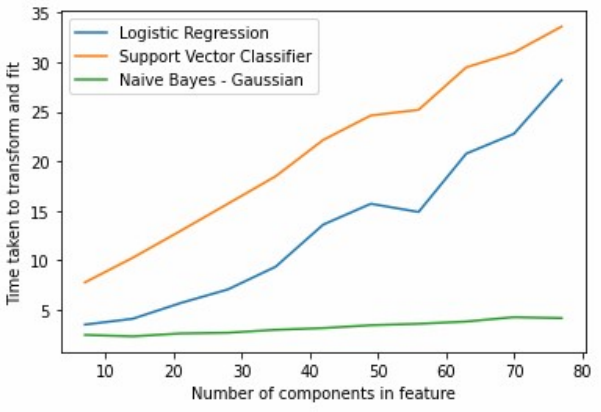

Figure 9: Effect of PCA on Model Accuracy

We have addressed the optimization of the models by leveraging the regularization pipeline (see Figures 6, 8, 7), which was done by testing different regularization constants for the Support Vector Classification model and the Logistic Regression model. By generating models with different regularization constants, we were able to find the most optimal constant for each model which actually differed between the two models (0.1 for the Support Vector Classification model and 1.0 for the Logistic Regression model). Additionally, as shown in Figure 9, reducing the dimensionality of the features through Principle Component Analysis (PCA) allowed for the Logistic Regression and Naive Bayes models to remain robust with less features, but faster runtime in regards to model fitting.

3.6 Scalability / Overfitting

By using the regularization pipeline mentioned above, the two models were able to minimize the possibility of learning a more complex / flexible model by constraining and shrinking the coefficient estimates, which led to avoiding the risk of overfitting.

In terms of scalability, we utilized the Principal Component Analysis (PCA) to check which features were least informative and reduce

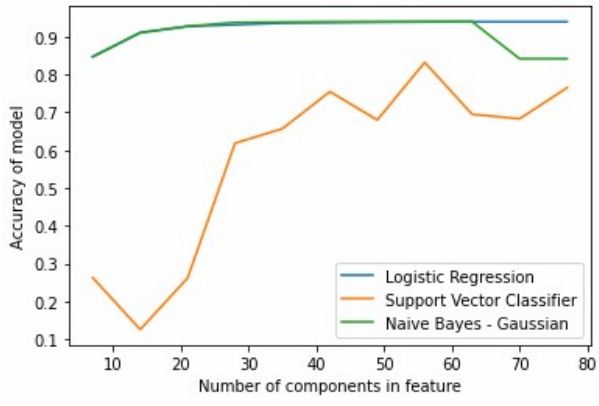

Figure 10: Effect of PCA on Model Accuracy

Number of components in feature	Logistic Regression	Naive Bayes - Gaussian
77	0.939298	0.841716
70	0.939298	0.841716
63	0.939298	0.939366
56	0.939026	0.939252
49	0.938096	0.939343
42	0.936895	0.939298
35	0.935988	0.938482
28	0.931319	0.937552
21	0.927194	0.927194
14	0.91042	0.91042
7	0.846975	0.846975

Figure 11: Comparison of PCA Effect on Naive Bayes and Logistic Regression

the dimensionality of the dataset while minimizing information loss. By looking at the Figure 7 graph, it is clear that the PCA significantly decreased the accuracy of Support Vector Classification model while significantly increased the accuracy of the Gaussian Naive Bayes model. In fact, the Gaussian Naive Bayes model yielded the highest accuracy when the PCA reduced the number of features from 77 to 63. This proves that the Gaussian Naive Bayes model was heavily affected by scalability and the PCA was able to ameliorate the issue. Finally, we notice as per Figure 7 we extrapolate that our SVC classifier was slightly overfit, as applying more regularization strength gave a better accuracy. Additionally, as per Figure 10, applying a dimensionality reduction on the set of features considerably improved the accuracy of Naive Bayes, indicating that the model was also overfit to the large number of unhelpful features.

3.7 Unsuccessful attempts

3.7.1 Feature Extraction. Additionally, we attempted to utilize the Girvan-Newman algorithm for achieving a hierarchacal clustering of the graph. We thought that removing the edges in the graph with the highest betweenness-centrality would yield an accurate set of social circles as has been documented by [1], but this attempt was unsuccessful because there were too many edges in the graph for the library algorithm to compute, and thus the model generation never completed (as shown in section 4.2.2).

3.7.2 Model Generation. Our initial thought for a good classification algorithm was the Random Forest Classification model, because we believed that leveraging multiple decision trees would be able to classify the binary encoding of the feature space. However the actual accuracy ended up significantly lower than the baseline models, even less than random chance.

4 RELATED WORKS

4.1 Associated Paper

As stated in the introduction, this paper uses the dataset extracted by Leskovec and McAuley for [3]. In that paper, the dataset was used in a similar sense, in that in [3], the authors were attempting to model social circles. However, their task was to model these circles as closely to the 'ground-truth' user-reported social circles as possible. It might have been worth exploring utilizing their modeling of the latent variables by means of user profile mining, however given that their goal was to accurately predict the social circles of a given user and we already had access to the 'ground-truth' data ourselves and had made a classifier from it, we felt that we it would be more productive to predict communities *beyond* what users themselves classify.

4.2 Related Datasets

4.2.1 2012 NIPS Study. The aforementioned study [3] also conducted similar analyses on data scraped from publicly availible Google+ and Twitter profiles. However, their latent variable modelling was much less accurate on these datasets, which they attributed to the fact that the graphs and profiles were were 1) Incomplete, 2) Directed not undirected connection, meaning community structure is less easily explainable 3) and possibly out-of-date due to the nature of their public availability. However, we took this as further evidence to the unreliability of users in documenting their own social circles, strengthening our beliefs in the hypothesis that an unsupervised learner would lend itself more to link prediction then relying on the ground-truth data.

4.2.2 Facebook Kaggle Competition. We also looked at the much larger scale Facebook Kaggle Competition[3], which posed a similar question of link prediction. However, the graph itself was much larger and sparser, and was actually directed instead of undirected. Specifically, we looked at a case study of one individuals's attempt [2] at a link prediction model for this dataset. Similar to our model, he leveraged only the network features of the graph, without accounting for profile data. Unlike our model however, he utilized multiple metrics for computing similarity between two nodes in the graph, not just community detection (e.g. Jaccard Similarity, betweenness-centrality, PageRank, etc.) However, one class of features he used was random-walk clustering, same as ours. He was able to get extremely good results just leveraging network structure with an accuracy of 0.93, which is quite similar to ours (though admittedly, his analysis was done on a much larger and more difficult dataset). Ultimately, this gives even more credence to our choice to disregard profile features and focus specifically on unsupervised community detection to extract our features.

[3]https://www.kaggle.com/c/FacebookRecruiting/discussion

4.3 Survey of the State-of-the-Art

4.3.1 In Regard to Hierarchical Clustering. Link prediction is a very popular topic of research, so there are too many methods in the literature to discuss here. However, among the state of the art, hierarchical clustering appears to be a very favorable metric for this task [4]. However in that same paper, it noted that for graphs greater than the order of thousands of nodes, this method can approach intractability. We noticed something similar, which was that a number of the hierarchical clustering methodologies we tried were unable to return a result in a reasonable amount of time. Only when we relaxed our constraints to simple random walks, were we able to get a clustering quick enough to be useful. However this type of clustering ignores other potentially useful vertex and neighborhood metrics, so of course this is the trade-off. Hierarchical clustering is noted again by [7] to be exceptionally useful for link prediction. However, he also noted that empirical analysis lends itself to the hypothesis that the model is more susceptible to confounding by edges between nodes that are dissimilar in terms of clustering, because such edges are very likely to exist. This could lead to a lower accuracy in correctly classifying edges that are actually in the graph, which we see in our model as well, as the True Positive Rate (TPR) lags noticeably behind the True Negative Rate in all of our models.

4.3.2 Other Models. Interestingly, in [4], the disjoint partitioning philosophy of clustering was reported to outperform hierarchical clustering. This is not what we document in our own experiments, with the former noticeably outperforming the latter. This could be for a number of reasons, one could be that our data was already comprised of ego-networks, giving more room to form hierarchies inside those ego-networks that better fit the network structure. Thus, it could be potentially informative if we were to extend our model to more generic social networks.

5 RESULTS AND CONCLUSION

5.1 Feature Extraction Comparision

Comparing the methods of community detection as described in 3.2, we see that our hypothesis that hierarchical clustering would better model the latent variables we were trying to depict is supported by our results. Comparing the same models and same metrics, we see that Logistic Regression with no regularization or dimensionality reduction yields a higher accuracy on unseen data using the feature representation depicted in 3.2.2. As we stated before, this can be explained by the fact that individuals themselves likely belong to a number of overlapping social circles, not just one, thus using this community detection methodology, we can better model actual human behavior and get more accurate predictions.

5.2 Model Comparison

5.2.1 PCA Transform. As seen in Section 3, before PCA transform, Logistic Regression was the clear winner, easily beating Naive Bayes and Support Vector Classification. However, after running a Principle Component Analysis and transforming the training data, Naive Bayes was much more competitive with Logistic Regression. This is likely because, Naive Bayes assumes conditional independence between feature elements, however given the hierarchical

structure of our communities, this is not a valid assumption to make. Thus, a PCA transform, which helps account for the covariance, strengthens Naive Bayes classifying accuracy. Additionally, from Figure 10, we can infer that the Bayes Classifier was overfit to the hierarchical communities, because reducing the dimensionality vastly increased the accuracy of it. This is supported by the same analysis on Logistic Regression, which showed robustness in accuracy up to a dimensionality reduction of 25%. This has some ramifications for our hypothesis, because it is possible that the large number of communities (given the overlapping), confused Naive Bayes, which could easily be due to extraction method of hierarchical clustering.

5.2.2 Regularization. Given the fact that our Logistic Regression and Support Vector Classifier both gave maximal it these models to the training data with relatively weak regularization, we can be confident in saying that neither of those models were considerably overfit to the training data.

5.3 Comparison to Baselines

As seen in Section 2.4.1 and 2.4.2, the baseline models had an accuracy of 0.6624 and 0.8085. The Gaussian Naive Bayes model yielded an accuracy of 0.9387, which is much higher than both of the baseline models. The first baseline model from 2.4.1 did not yield a high accuracy most likely because the approach of computing the popularity and comparing the value with a threshold was not ideal for predicting the friendship between the two users since those two values are independent from one another and no metric of network connection is used to quantify similarity. The second model from 2.4.2 also did not yield high accuracy most likely because the user-specified circles from the original data would have been biased by the user themselves. On the other hand, the Gaussian Naive Bayes model was able to yield a significantly higher accuracy because the model is based on an algorithm that assumes conditional independence between features and it is also not based on a biased dataset.

5.4 Parameter Interpretation

Circle Index	Size of Circle	Density of Circle
1	232	0.6037841468875952
22	73	0.5654490106544902
26	38	0.7396870554765291
9	158	0.31661694751269853
2	286	0.2535394430131272
0	226	0.22025565388397247
20	25	0.8666666666666667
10	125	0.1993548387096774
13	40	0.4576923076923077
17	54	0.32634521313766596
4	424	0.18222266827244749
3	125	0.196

Figure 12: Size and Density of Most Influential Communities

In the pre-PCA feature representation, our model's parameters to the weight given to a particular edge being in the subgraph of a given community, in regards to the model. Figure 12 depicts the top 10 highest weights and some statistics for them. On a first glance, no pattern emerges, however it does seem to be that the extremes of large/small and dense/sparse are represented in the ranks. One might infer that a community that is particularly dense would correspond to predicting true, whereas sparse might correspond to predicting false. However, more rigorous analysis is needed to explore this hypothesis.

5.5 Next Steps

Some further areas of research could be extrapolating this model to other datasets, such as the ones used by [3]. Additionally, a more rigorous analysis on the latent variables found by both methods for community detection, especially when compared to the ground-truth data could potentially be interesting as well. Finally, we could also try to see if incorporating additional, profile-related features into our model would be beneficial to its capacity for link prediction.

ACKNOWLEDGMENTS

The authors would like to thank Professor McAuley for the dataset as well as all the lectures for the course. The authors would also like to thank the TAs in the course for providing insights throughout the course helping our homeworks and answering our questions during office hours and on Piazza.