Social Network Circle Discovery Using Latent Dirichlet Allocation

Frank Fan
Stanford University
Department of Mathematical and Computational Science
ffan9@stanford.edu

Jaimie Xie
Stanford University
Department of Computer Science
jaimiex@stanford.edu

Matthew Kim
Stanford University
Department of Computer Science
mdkim@stanford.edu

Abstract

Online Social Networks, such as Facebook, provide a great interface for connecting with others, whether they are acquaintances or close friends. However, there is no distinction made between different *social circles*, or clusters of friends who share some common feature(s). In this paper, we explore ways to apply *Latent Dirichlet Allocation (LDA)*, an unsupervised learning algorithm traditionally used for topic detection in textual corpora, to automatically detect social circles among a subject's friends. For each friend, which we will consider as *documents*, we take in account both the profile features and users' friends, comparable to word "tokens." Finally, we will analyze our results by finding the cost-minimizing assignment from our circles to the ground-truth circles, based on the *Balanced Error Rate (BER)*.

1 Introduction

As of August 2015, Facebook, an immensely popular online social networking service, had over 1.59 billion monthly active users. The site allows users to create a user profile and add other users as "friends." Users can then categorize their friends into lists, such as "Close Friends," or "People From Work." However, with the average Facebook user having about 338 friends, manually picking out these friend circles becomes a laborious process. The purpose of our experiment is to explore algorithms that can automatically detect these circles, so that users can more easily personalize to whom they share their information. For example, a user would most likely not want to share the same information with acquaintances than with family members.

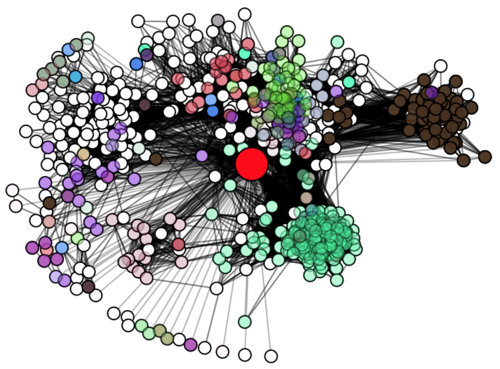
Figure 1: Graphical interpretation of an ego network. The large, red node signifies the ego node. The alters are colored according to the ground-truth circles they were placed in.

In order to detect these circles, we will consider three sources of information: the user's profile features, the user's friends' features, and the network structure. In general, we want the friends in each circle to share certain common features (such as same school, same work, etc.), and also have many common friends within the circle. (Connectivity within a circle can also provide information on which groups are more casual and which are more tight knit.)

Finally, we should also consider the possibility of friends belonging in multiple circles. For example, someone who went to the user's university could also be a coworker. Therefore, our problem at hand is not a traditional clustering problem, where each example falls in one cluster. We will address the multi-cluster problem by using an algorithm that determines a multinomial distribution of the circles for each of the users' friends. We will explain this algorithm in more detail in the latter sections.

2 Background

This section gives background information on the methods that have been explored to solve the task of Social Circle Discovery. We will then discuss Latent Dirichlet Allocation, which is a method from Natural Language Processing that we will apply to our task.

2.1 Related Work

McAuley and Leskovec (2012) developed a novel method that builds a probabilistic model of an ego graph based on connectivity between nodes, and the circles that exist among the nodes. The circles are treated as a latent variable in the optimization of the likelihood of the graph. Petkos et al. (2015) used Latent Dirichlet Allocation in social circle discovery, but only used individual user-features and id's of neighbors in model training.

2.2 Latent Dirichlet Allocation (LDA)

For social circle discovery, we turn to Latent Dirichlet Allocation (LDA), originally devised by Blei et al. (2003) for topic modeling in Natural Language Processing. This involves treating users in a network as "documents," and user features as "words." The primary hope is to produce a mixture model of social circles for each user, based on their features: birthday, workplace, etc. Not only is this an intuitive extension of the field of linguistic topic-modeling, LDA proves to be a more time-efficient algorithm than traditional methods, while achieving comparable results (Petkos et al. 2015).

LDA is a generative algorithm that views documents as mixtures of topics, with each topic being a multinomial distribution of words. LDA models the production of each document in a corpus in the following fashion:

1. Produce an N $\sim$ Poisson(η), the length of the document.
2. Produce a $\theta \sim$ Dirichlet(α). θ represents the distribution of topics within a document.
3. For each word w_i in the document,
   (a) produce a topic $z_i \sim$ Multinomial(θ).
   (b) produce word $w_i \sim$ Multinomial(z_n, β)

From this model, we can formulate the probability of a document, $\mathbf{w}$:

$$p(\mathbf{w}|\alpha, \beta) = \frac{\Gamma \sum_i \alpha_i}{\prod_i \Gamma(\alpha_i)} \times$$

$$\int \left(\prod_{i=1}^{k} \theta_i^{\alpha_i - 1} \right) \left(\prod_{n=1}^{N} \sum_{i=1}^{k} \prod_{j=1}^{V} (\theta_i \beta_{ij})^{w_n^j} \right) d\theta$$

Since this formulation produces a log-likelihood maximization problem that is intractable, Blei et al. (2003) propose an EM procedure for learning the model parameters.

Furthermore, in line with work done by Hoffman et al. (2010) and Rahurek and Sojka (2010), we utilize an online-learning variant of LDA for this project.

3 Experiment

In the previous section we discussed the LDA method, which we will now apply to the problem of social circle discovery. This involves modeling the user's friends as "documents," features as "words," and social circles as "topics" in the documents. The features we used not only included individual friends' features (birthday, workplace, etc.), but also the id's of each user's friends to capture some sense of the connectivity of the graph. We also added features which each friend shares with the user/ego node. Concretely, for each friend "document," we have the feature labels for each exhibited feature on his/her profile, the feature labels for each feature he/she shares with the user-node, and the user IDs of all the friends that he/she is connected to by an edge.

To evaluate the performance of our LDA algorithm, we also ran two variants of K-means clustering, using just the feature vectors of the friends' profiles. However, since in many cases, the feature dimensions exceeded the number of friends, we compressed the feature vectors using tSVD (See Section: 3.2).
To summarize, we ran the following algorithms:

- LDA using the network structure, user's profile, and friends' profiles. We set the number of circles, k = the number of ground-truth circles. We will refer to this algorithm as "LDA"
- LDA as above, except we set k using the AIC_C selection algorithm described in section 3.1. This will be "LDA+C"
- K-means clustering using only the compressed feature vectors of the user's friends. We set k = the number of ground-truth

circles. We will refer to this algorithm as "KMEANS"

- K-means as above, except we set k using the AIC_C selection algorithm described below. This will be "KMEANS+C"

3.1 Parameter Tuning

With the LDA algorithm, it is vital to pick the number of topics, which we will denote by K, to model. We accomplish this with a stepwise procedure through a grid search of varying values of K, to choose the model that minimizes the AICc (Petkos et al. 2015):

$$AIC_c = -2LL + 2p + \frac{2p(p+1)}{N-p-1}$$

Where LL is the log-likelihood of the model with respect to the dataset, N is the total number of words across all documents (total number of features summed across all users) and $p = K(M-1) + D(K-1)$ is the effective number of parameters, with K being the number of topics (circles), M the number of distinct words (features) and D the number of documents (users).

To obtain the log-likelihood of a model, we utilized the perplexity measure produced by the online-LDA and used the following formula relating perplexity and log-likelihood (Blei et al. 2003):

$$Perplexity(C) = exp(\frac{-LL(C)}{N})$$

with C representing all the documents in a corpus (i.e., all users in a network).

The traditional AIC criterion helps to choose the model with the greatest likelihood, while penalizing models with large numbers of parameters (which mitigates over-fitting). The AIC_c criterion, however, also corrects for finite sample sizes that are small with respect to the dimension of the parameter space (Hurvich and Tsai 1989). We see, also, that as the number of words N in the corpus increases without bound, the third term of the AIC_c drops out and we are left with the formula for AIC.

3.2 Baseline Comparison

To set a baseline comparison, we decided to use K-Means clustering, which is capable of assigning each user to only one circle, so is expected to be less robust than our LDA algorithm.

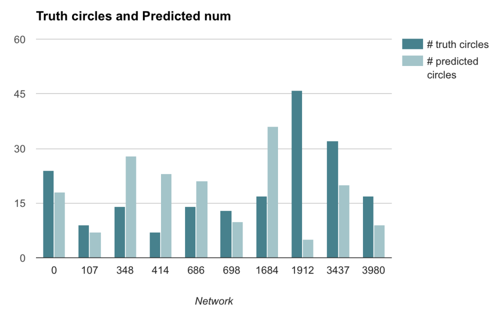

Figure 2: The number of circles predicted may not always correlate with the ground-truth number of circles.

To pre-process the data before K-Means clustering, we used the truncated SVD method (Berry et al. 1995). Whereas most implementations of the tSVD require one to specify the number of eigenvalues to keep, We use our own tSVD implementation, adapted with a rule proposed by Leskovec et al. 2014, which allows us to avoid human-inspection of the SVD eigenvalues (for a drop-off), and to avoid hard-coding the number of eigenvalues to keep:

- With our data matrix, with rows representing users and columns representing the feature values for each user, we formed the SVD $(U\Sigma V^*)$

- By a rule of thumb (Leskovec et al. 2014), we truncated the maximum number of eigenvalues such that: the sum of squared remaining eigenvalues is at least .9 of the sum of squared original eigenvalues. This allows us to capture a good amount of the variation in the data, while reducing uninformative dimensions.

- Let's say we truncated the SVD to the m largest eigenvalues. Our tSVD is then denoted by: $U_m \Sigma_m V_m^*$. To form the dimension-reduced dataset, we simply take $U_m \Sigma_m$.

3.3 Data

The data that we will use for training/testing is provided by the Stanford Network Analysis Project, and all of our data comes from Facebook. The data is divided into *ego networks*, which consists of the *ego* node, all of the nodes it is connected to (called *alters*), and all of the edges there may be among these nodes. Within each ego network, we have the following:

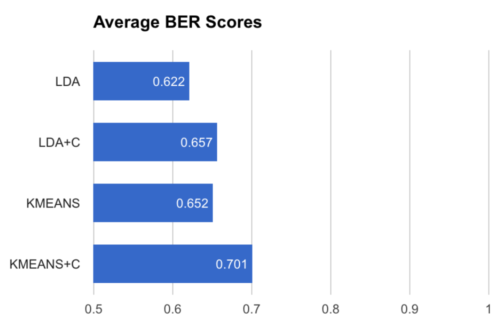

Figure 3: The average BER Scores across the four algorithms that were tested. BER scores are obtained by subtracting the average BER error from 1, so a higher score means better performance.

- Circles: these are the circles that the user manually chose, the *ground-truth circle*. The circles are not necessarily disjoint, so one user can be in multiple circles. We will compare our results with these sets of users.
- Edges: this contains every edge in the ego network, other than the implicit edges that connect each *alter* to the *ego node*. An edge, (n_1, n_2) signifies that alters n_1 and n_2 are ”friends” on Facebook.
- Features: for each alter, we are given a binary array, where a 1 in index i signifies that feature i is satisfied (and 0 otherwise). The features are constructed in a tree-structure, where example features include:
  - education:university:Stanford
  - education:university:Harvard
  - education:year:2018, etc.
- Feature names: this contains the names of the features that correspond with the feature arrays. In general, we will just use the numerical labeling of the features.

3.4 Results/Analysis

After running the LDA Algorithm, we get the multinomial distributions of the circles for each of the friends in the ego-network. At this point, we can choose a cut-off probability to choose which circles each user actually should be assigned to. For example, if a user is assigned a probability $< .05$ of being in circle A, then this user is likely not actually in circle A. In choosing this cut-off probability, we have to consider how many circles we are actually predicting. Let N be the number of circles we predicted, then we will place user u in circle C if $Pr(u \in C) > 1/N$

In our K-Means Algorithm, each user was automatically labeled into one circle.

Once we have established these circles, we want to be able to directly compare the automatically produced circles with the *Ground-truth circles*, which are the circles that the ego-user manually chose. To do this, we must determine an optimal mapping from our circles to the circles which the ego-user hand-picked. First, we need to determine some error/cost function which we would like to minimize. For the purpose of our experiment, we used the Balanced Error Rate (BER), as did Petkos et al (2015). If we let $C = \{C_1, C_2, ..., C_K\}$ be the set of automatically produced circles, and $\bar{C} = \{\bar{C}_1, \bar{C}_2, ..., \bar{C}_K\}$ be the set of ground-truth circles. Then, we can define the BER as:

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

The BER cost function equally weights the fraction false-positives and false-negatives. If we compute the BER for every pair $(C_i, \bar{C}_j)$, we can construct the cost matrix where the $ij - th$ entry is $BER(C_i, \bar{C}_j)$. (Note that since the number of circles which we predicted does not always match the number of truth-circles, our cost matrix is not always a square-matrix. The number of matchings that we will get in this case will be $\min(|C|, |\bar{C}|)$)

We want to find a circle matching $f : C \to \bar{C}$, which gives us the least total error. If we were to try every possible f and then compute the cost, this would take $O(n!)$. However, with the Kuhn-Munkres algorithm, we can solve the assignment problem in $O(n^3)$ time.

For our final *BER score*, we take the average of the BER rates from each circle assignment, then subtract that from one:

$$BER_f = \frac{1}{|f|} \sum_{C \in dom(f)} (1 - BER(C, f(C)))$$

For each algorithm we average the BER_f values from each of the 10 ego-networks. We get the following results: KMEANS reports a BER score of .652, KMEANS+C obtains a score of .701, LDA a score of .622, and LDA+C a score of .657.

For both our K-means and LDA algorithms, we achieved better results when we predict the number of circles using AIC_C, rather than just setting

k = the number of ground-truth circles. This is because in the latter case, we are overfitting the data - in one of the networks (for LDA), using our predicted number of circles ($k = 5$ rather than $k = 46$) improved the BER score from .632 to .851. Many of the ground truth circles only contained 1-2 people, and by abstracting away these circles, we actually got better results.

Another surprising result was that the K-Means algorithms, which used only the feature vectors of the user's friends, but did not consider the network structure or user's own profile features, did better than the LDA algorithms, which considered all three components. However, our implementation of the LDA algorithm places larger weight on the network structure because most of the user's friends have many more connections within the ego-network than 1s in their feature vectors. Since we are treating each connection as a *word* in the *documents* (the user's friends), the documents will largely be composed of network structure. This implies that profile features (even using the compressed vectors) may tell us more about circle formations.

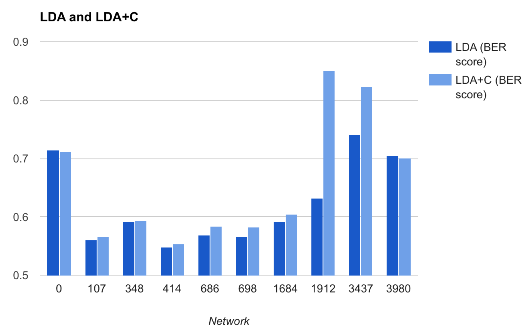

Figure 4: LDA performs better for most trials when we predict the number of circles using AIC_C selection, rather than using the number of ground-truth circles.

4 Future Work

We hope to explore other combinations of features to include in our LDA model, such as interaction terms and indicators of edge strength. We also hope to devise ways to factor network-connectivity into our model-building without having it overwhelm the other features present. Another area of work would be to explore parameter tuning with BIC and AIC and compare results with AIC_c selection to verify our theoretical decision to use AIC_c.

References

Berry, M. W., Dumais, S. T., and O'Brien, G. W. 1995. Using linear algebra for intelligent information retrieval *SIAM review*, 37(4), 573-595.

C. Hurvich and C. Tsai, 1989. Regression and time series model selection in small samples. *Biometrika*, 76, 297307, 1989.

David M. Blei, Andrew Y. Ng, Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993-1022.

Matthew D. Hoffman, David M. Blei, and Francis Bach. 2010. Online Learning for Latent Dirichlet Allocation. *Advances in Neural Information Processing Systems 23 (NIPS 2010).*

Julian McAuley and Jure Leskovec. Learning to discover social circles in ego networks. *Proc.c of NIPS.*

Leskovec, J., Rajaraman, A., and Ullman, J. D. Mining of Massive Datasets. *Cambridge University Press.* 2014.

Pedregosa et al. Scikit-learn: Machine Learning in Python *JMLR* 12, pp. 2825-2830, 2011.

Georgios Petkos, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2015. Social Circle Discovery in Ego-Networks by Mining the Latent Structure of User Connections and Profile Attributes. *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining.*

Radim Rahurek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. *ELRA.*