Facebook User Networks Based on Information Retrieval from Media Fan Pages: Network Analysis and Recommendation System

Sudharsan Vasudevan, Yahoo! Inc., Sunnyvale, CA, USA
Liangliang Zhang, Department of Electrical Engineering, Stanford University, CA, USA

***Abstract* — Facebook fan pages are major source for identifying users interests. They drive a lot of traffic to many sites outside of Facebook. If we can analyze such fan pages, we could come up better recommendation algorithms based on how users are interacting in a social network around fan pages. Such recommendations can be used to power many new recommendation engines such as Yahoo recommends, Taboola and Outbrain that have come up in the past few years. The important reason we took Facebook fan pages is because most sites have their presence in Facebook and most users who come to the websites navigate from Facebook itself.**

As part of the paper we analyze how users in Facebook fan pages behave across multiple fan pages and we assess if the data will be good enough to create recommendation engines for the websites. For the project we crawled Facebook fan pages to get users activities and tried to understand how users interact with the articles in the fan pages. We also extracted important terms that summarize the articles and created user interests based on these extracted terms. We find that the users interests across fan pages roughly follow power law. So to identify the distribution, we explore other functions such as lognormal and DPLN (Double Pareto Log Normal) to identify what kind of distribution the Facebook fan pages follow. Based on the findings we decided to use terms extracted from articles that users have liked to implement the recommendation engine. To be specific we recommend using what articles and terms users like for the recommendation engine rather than what fan pages they like. It is found that user-user collaborative filtering algorithm works best in case of such fan sites mainly because users have varied interests across multiple fan sites.

***Key words* — Media Websites, Social Network, Skewed Power Law Distribution, Recommendation Engine**

I. Introduction

Recently, the number of people involved in social networks dramatically increased throughout the world. Analyzing user activities on social networks is one of the most effective ways to understanding user behaviors. The public media sites, such as the sites of CNN, NYTimes, BBC, etc., on Facebook contain lists of articles and Facebook users can like, share and comment on them. They provide very useful data collecting sources to build user interest model based those activities and gain information though mathematical analysis on these data.

In this paper, we will generate a user graph G (V, E) with tuple (V, E) as vertices and edges. Each vertex (node) V is represented by a user and E denotes the set of edges, which represents how close a pair of users. The edges are established based on the users common interests in the network. Three per-user characteristics are used to generate the edges:

(1) The similarity between users likes, comments and shares of articles.
(2) The similarity between user activities on article sources.
(3) The similarity between user activities on extracted terms.

The distances between two sets of articles, sources or terms could determine each of these similarities. The calculation procedure will be described in later parts of this report.

Based on the network we built, the first step is to analyze the behavior of these networks, including finding the most influential users, node degree distribution, clustering coefficient, network diameter, etc., which helps to understand the networks. Comparison studies among the networks established upon different per-user characteristics are performed for better user interest modeling. Heavy tail distribution is then calculated for the node distributions for all the three networks, we also derived the best fit by using Double Pareto Log Normal (DPLN) distribution. A recommendation system is then built based on extracted terms of the articles and a collaborative filtering method is implemented on what articles a specific user would like based on the derived users profiles.

II. Related Work

Network analysis is one of the hottest topics today and there are many existing literatures in this field. The node degree distribution of a network is one of the most useful characteristics that classify in interpreting a network, so that most of the researches begin with plotting the node distribution of a network. The node degree distribution of a network obeys the power law if the probability of a node with degree k is proportional to the exponential function of k, which is a linear curve in log-log plot with coefficient alpha as the negative of the gradient, with equation $f(x) = Cx^{-\alpha}$. This so-called Power Law Distribution (PLD) is a wide emerging behavior of

This report is for 2014 fall CS224W project. Submitted on Dec. 9, 2014.
Sudharsan Vasudevan is with Yahoo! Inc., Sunnyvale, CA, USA.
Liangliang Zhang is with the Department of Electrical Engineering, Stanford University, CA, USA. (Email: lianglia@stanford.edu)

real-world networks, in which a few nodes have high degrees and many nodes have small degrees. It clearly differentiates these real-world networks from "human fabricated" networks, such as random generated networks, or traffic networks, etc., where most of the nodes have similar degrees. While PLD is a general phenomenon, deviations from PLD happened in certain kind of networks, and can be named by Skewed Power Law Distribution (SPLD).

Exponential cut-offs, which decays exponentially for higher degree nodes, and lognormal distributions $f(x = k) = \frac{A(\mu,\sigma)}{k} exp[-\frac{(lnk - \mu)^2}{2\sigma^2}]$, are two common SPLDs, which are mentioned in [1]. Another deviation was found in [2] where they analyzed dataset based on a large collection of Call Data Record from a cellular network, in which they found skewed PLD in the distribution of the number of phone calls, the total talking minutes and distinct number of calling partners as a function of customer based on are unable to be fitted by either power law or log-normal distribution since the curve in log-log plot has much sharper turning when the x-axis (partners) exceed a certain point (called "heavy tails"). It is found that the distribution is best fit by DPLN.

There are also many works on recommendation systems. Common methods include collaborative filtering [3], content-based filtering [4], or hybrid recommendation system. In Julian's paper [5], they proposed a statistical model that considers latent dimensions in both the review rating and review text based on Amazon's review data. The review text is found to give more accurate result when is incorporated in LDA model. In our project, while it is difficult to directly work on user's comments, we are able to calculate similarity of users based on article texts. It will provide more accurate prediction and is beneficial for new users. Even only a few likes we got from the users, we are able to get a lot of information from the liked articles.

III. ALGORITHMS

The algorithms we used in this report contains several parts and we will present them in the order of their appearing sequence.

Most Influential Users: To calculate the most influential users, Hyperlink-Induced Topic Search (HITS) method is used. The algorithm is shown below where articles are represented by 'a' and users are represented by 'u'.

Algorithm 1: HITS

1. Initialize: $a_j^{(0)} = 1/\sqrt{n}, u_j^{(0)} = 1/\sqrt{n}$ for all j
2. Loop until converge:
3. -- $\forall i, Authority$: $a_i^{(t+1)} = \sum_{j \to i} h_j^{(t)}$
4. -- $\forall i$, Hub: $h_i^{(t+1)} = \sum_{i \to j} a_j^{(t)}$
5. -- $\forall i, Normalize$:
6. $\sum_i (a_i^{(t+1)})^2 = 1$
7. $\sum_j (h_j^{(t+1)})^2 = 1$
8. Converge criteria:
9. $\sum_i (a_i^{(t+1)} - a_i^{(t)})^2 < \varepsilon$
10. $\sum_i (h_i^{(t+1)} - h_i^{(t)})^2 < \varepsilon$

Then we sort the users according to their hub score and pick the top K users (K = 0.1 ~ 1 million).

Algorithm for similarity calculation: The first part is to calculate the similarity between users in order to build the network. We used Jaccard distance for similarity calculation and it is shown that this method is good enough to reveal the network characteristics while keeping a low calculation overhead. More accurate similarity calculation might be necessary for recommendation systems, which we will discuss in more detail in later report. The Jaccard distance JS of two sets A and B is defined as:

$$JS(A,B) = \frac{|S(A) \bigcap S(B)|}{|S(A) \bigcup S(B)|}$$

Where the set surrounded by '| |' are the number of elements of the set. In our case, A and B are the sets containing article IDs or extracted terms or site source IDs of two users, and JS(A, B) represents how closely A and B are related.

Distributed Algorithm for similarity calculation:
For the test data we randomly picked 20% of users ratings on 20% articles. We took the rest of the training data and calculated the terms for a given Facebook user based on articles he or she liked. Please note that the term calculation algorithm will be covered soon.

We then calculated two different kinds of similarity:
1) The Jaccard similarity between two users based on the common terms between them. This will later be used for implementing user-user collaboration system.
2) The Jaccard similarity between two articles based on the common terms between them. This will later be used for implementing item-item collaboration system.

The algorithm to calculate Jaccard similarity between two users based on terms is given below that can be easily extended to calculate article similarity also.

Algorithm 2: Distributed Jaccard Similarity Calculation

01. Input: $fbId, \{terms\}$
02. Map 1:
03. -- $input: < fdId, \{terms\} >$
04. -- $for\ each\ term\ in\ \{terms\}$:
05. $emit < term, (fbId, count(\{terms\}) >$
06. Reduce 1:
07. -- $input: < term, \{(fbId, count)\} >$
09. -- $emit: < term, \{(fbId, count)\}, \{(fbId, count)\} >$
10. Map 2:
11. -- $input: < term, \{(fbId1, count1)\},$
$\{fbId2, count2 >$
12. -- $for\ each\ (fbId1, count1)\ in\ \{(fbId1, count1)\}$:
$for\ each\ (fbId2, count2) in\ \{(fbId2, count2)\}$
$if\ fbId1 < fbId2$:
combined_Count = count1+count2
$emit: < (fbId1, fbId2), combined\_Count >$
13. Reduce 2:

14. -- $input$: $< (fbId1, fbId2), combined_{Count} >$
$JS = count(\frac{\{combined\_Count\}}{MAX(\{combined\_Count\})}) - count(\{combined\_Count\}))$
15. -- $emit$: $< fbId1, fbId2, JS >$

Fitting Skewed PLD: The first idea is to fit the skewed PLD with lognormal distributions with equation:

$$f(x = k) = \frac{A(\mu, \sigma)}{k} exp[-\frac{(lnk - \mu)^2}{2\sigma^2}]$$

where μ is the mean and σ is the variance, and it is actually the logarithm of normal distribution (so called lognormal distribution).

But lognormal doesn't fit well with our data, so that Double Pareto Log Normal Distribution (DPLN) was used, which was reported to have very good fit for heavy tail cases. The data are plotted using the distribution function:

$$f(x) = \alpha\beta/(\alpha+\beta)[exp\left(\alpha v + \frac{\alpha^2\tau^2}{2}\right)x^{-\alpha-1}\Phi(\frac{logx - v - \alpha\tau^2)}{\tau}) + x^{\beta-1}exp(-\beta\tau + \beta^2\tau^2)\Phi^c(\frac{logx - v + \beta\tau^2)}{\tau})]$$

where Φ (Φ^c) are the cumulative distribution function (CDF) (or complementary of CDF) of normalized distribution N(0, 1), and α, β are the roots of the second order real coefficient equation based on Geometric Brownian Motion.

$$\frac{\sigma^2}{2}z^2 + \left(\mu - \frac{\sigma^2}{2}\right)z - \lambda = 0$$

Where λ is the exponential distribution parameter, μ and σ are constants. There is also a neat method called PowerTrack to fit the data with only four parameters, and they found the network of cellular users evolves in a generative process that consistent with DPLN which confirmed the correctness of the underling foundation of the model.

Though the DPLN distribution fits well for the heavy tail part, the distribution at small node degrees are distorted under the curve. It is also computational expensive compared to simple power law or lognormal. We noticed that DPLN fits well for heavy tail part but the discrepancy at low node degree part is huge. As a result, we combined the power law distribution and DPLN distribution to fit the data:

$$f(x) = \lambda_1 g(x) + \lambda_2 h(x)$$

Where f(x) is the final distribution and g(x) and h(x) are power law distribution and DPLN distribution. λ_1 and λ_2 are fitting parameters and both are functions of x. The final distribution is normalized with normalization factors hidden in λ_1 and λ_2 .

Term Extraction: We then extracted terms for each article by analyzing the description of the article and the article title. For each article the extraction of terms involves the following process:
(1) Removing the stop words from the data.
(2) Lower casing of ASCII characters,
(3) Identifying frequently occurring unigrams, bi-grams, tri-grams terms across articles with a min frequency of 5.
(4) Identifying and ignoring low tf_idf terms, where tf_idf is the multiplication between term frequency and inverse document frequency [6]
(5) Identifying the top significant terms for article based on 2 factors
---- Presence of the term 'i' in the title of document D
---- Importance of the term calculated based on

$$D_i = tf\_idf(i) * \left[1 + T_i + \sum_{j \in D, j \neq i} \frac{1}{\exp[NGD(i, j)]}\right]$$

where D is the given document, D_i is the importance of term i in document D, $NGD(i, j)$ is the Normalized Google Distance given by [9], and T_i is 0 or 1 based on presence of term 'i' in title. We sort D_i and pick the top N terms to get the most frequent N terms of document D.

Distributed Algorithm for Collaborative Filtering: We used the trained similarity data for implementing 2 kinds of collaborative filtering.
1) User similarity based Collaborative Filtering
2) Article similarity based Collaborative Filtering

We calculated User-Article rating based user similarity and article similarity as suggested in [11] using a distributed algorithm because we had a lot of users across a lot of articles. Given below is the distributed algorithm used for calculating rating based on user similarity that can be easily extended to calculate rating based on article similarity also.

Algorithm 3: Distributed Algorithm for Collaborative Filtering

01. Input: $fbId1, fbId2, similarity(S)$

02. 1st Map-Reduce Phase:
03. Map 1:
04. -- $input$: $< fdId1, fbId2, S >$
05. -- $emit < fdId1, (fbId1, fbId2, S) >$
06. $emit < fbId2, (fbId2, fbId1, S) >$
07. Reduce 1:
08. -- $input$: $< fbId1, fbId2, S >$
09. -- sum = 0
10. -- for each S in {fbId1, fbid2, S}:
11. sum += S
12. -- for each fbId1, fbId2, S in {(fbId1, fbId2, S)}
13. $emit$: $< fbId1, fbId2, S, sum >$

14. 2nd Map-Reduce Phase:
15. -- $input1$: $< fbId1, fbId2, S, sum >$
16. -- $input2$: $< fbId, article >$//the list for which rating has to be calculated
17. Map 1:
18. -- get: $< fbId1, fbId2, S, sum >$
19. -- $emit$: $< fbId1, (fbId1, fbId2, S, sum >$
20. Map 2:
21. -- get: $< fbId, article >$
22. -- $emit$: $< fbId, article >$
23. Reduce:
24. -- get: $< fbId, \{(fbId1, fbId2, S, sum), (article)\} >$

```
-- articleList = {}, fbSimilarityList = {}
-- for each item in {(fbId1, fbId2, S, sum), (article)}:
        if item is of type(article)//has single tuple
            articleList.append(item)
        else:
            fbSimilarityList.append(item)
-- for each article in articleList:
        for each (fbId1, fbId2, S, sum) in fbSimilarityList:
            emit: < fbId1, fbId2, article, S, sum >

3rd Map-Reduce Phase:
-- input1: < fbId1, fbId2, article, S, sum >
-- input2: < fbId, article >//the list for which rating has to be calculated
Map 1:
-- get: < fbId1, fbId2, article, S, sum >
-- emit: < (fbId2, article), (fbId1, fbId2, article, S, sum >
Map 2:
-- get: < fbId, article >
-- emit: < (fbId, article), article >
Reduce:
-- get: < (fbId, article), {(fbId1, fbId2, article, S, sum), (article)} >
-- articleList = {}, fbSimilarityList = {}
-- for each item in {(fbId1, fbId2, article, S, sum), (article)}:
        if item is of type(article)//has single tuple
            articleList.append(item)
        else:
            fbSimilarityList.append(item)
-- for each article in articleList:
        for each (fbId1, fbId2, article, S, sum) in fbSimilarityList:
            emit: < fbId1, fbId2, article, S, sum >

4th Map-Reduce Phase:
-- input: < fbId1, fbId2, article, S, sum >
Map:
-- get: < fbId1, fbId2, article, S, sum >
-- emit: < (fbId1, article), (S, sum) >
Reduce:
-- get: < (fbId, article), {(S, sum)} >
-- emit: < fbId, article, SUM({(S)})/Max({sum}) as rating >
```

The output of the 4th phase will be fbId, article and the corresponding rating given by the collaborative filtering. In case of article-article similarity can be very similarly derived from the same algorithm with minor modifications.

IV. Facebook Data Collection

Facebook currently provides Open Graph API [7] that we used to collect data from the top 500 FB fan pages ordered based on user likes and user activity [8]. We implemented a web crawler that extracts from fan pages the list of articles (this contains article id, date, link and description about the article) and for each article the list of users who likes, commented or shared that particular article. We limited the number of articles crawled in each fan page to a max of 550 because we realized in most scenarios that covers at least 2 months of content. We also considered the articles that were created on or before 15th of October and it took about 2 weeks to collect the data.
We accumulated data for about:

- 235623 articles,
- ~105 million users,
- ~1.13 billion activities with ~9.97 likes, ~0.51 comments and ~0.01 shares on average for each user.

One of the main problems that we faced in processing this data is that the open graph API from Facebook restricts users with 1 query per second. So we had to run multiple crawlers in a distributed framework and merge the data back to cover all the 500 major Fan pages. First we created a pipeline of crawlers that extract first just top 500 articles from each of the 500 sites and federated it into specialized crawlers that only extract likes or shares or comments for a given article. This approach did not scale very well as the ratio of Like to share to comments was around 10 : 0.5 : 0.02. We then implemented a pipeline that extracts the articles as before but then uses one dedicated crawler to crawl all likes, shares and comments for a given article. This performed much better than the previous architecture. We ran the term extraction collected article separately to save time. We analyzed the terms from description of the comments and shares also but they were mostly irrelevant terms such as names and emoticons.

Based on the term extraction method explained in the last part, we extracted around 330k unique terms from 235k pages. We have extracted an average of 10 terms per page as in some cases the number of valid terms after extraction is very less and many get eliminated due to stop words and lower casing.

V. Basic Analysis on Facebook User network

The main problem we had with processing the data is that as the number of user nodes was pretty huge the processing time was pretty high too. As a result, distributed systems like Hadoop [10] was used in our project for efficient calculations. So we wanted to come up with the right amount of subset of the data such that it will resemble the population on the whole with little information loss. We realized that of the 100 million users many users have very little number of likes, comments and shares. We wanted to evaluate if the coverage will decrease if we decrease the number of user nodes.

The total disk space required to store all users is around 700GB. It takes too much resources to store this huge amount of data and too long to run algorithms. As a result, it is required to tradeoff between the number of users we calculate and the accuracy of the results. There are two approaches to select a sub-group of users. One is to select most important user, and this sub-group can be calculated using HITS method, and find the sub-group based on how much score a user got. The other approach is to select users with most sufficient data, where we calculate how many user activities are related to each user and sort the user according to the number of their activities. To better implement our network, both influential users and active user are important, so that we balance them by selecting users from both groups with a ratio 1 : 1. We found that if we choose the top 100,000 users in this way, the generated graph covers all the 500 sites, about 218,500 articles and about 300,000 terms. This implies that the most of the to users do end up being active on 92% of the data. The strong connected component (SCC) is larger than 99% of the network.

One important reason to use influential users is that they could represent the overall performance of the social network better.

Also the reason for using the influencer node is that a user sees their influencing contacts performing an action such as joining an online community, or liking an article and might decide to perform the action themselves too. Please note that after figuring out the properties and features of influential users, we could apply our algorithm to the general case too. Thus the set of influential users is a good subset of the entire network. We also filtered the list of likes, comments and shares only to those influencing users.

As part of processing the data the first problem we had was to calculate the most important terms for a given article. Considering there are about 230,000 articles and with thousands of words per article calculating TF-IDF and Google normalized distance was very expensive. We used Hadoop to distribute and calculate the parts of data such as TF-IDF for terms as suggested in CS246 [11]. Here are the three kinds of data we collected and reformulated based on the above data:

- Article-User frequency: article id and the number of users liked/commented/shared it
- Terms-User frequency: terms and the number of users related to this terms
- User-Article frequency: user id and the number of articles he liked/commented/shared

Power-law distributions and the processes that generate them are widely believed to characterize many real-world phenomena. In the previous section, we discussed the results of using well-known heavy-tailed distributions (power-law and lognormal) to fit the distributions of our chosen data. We attempted to model the observed data using power-law.

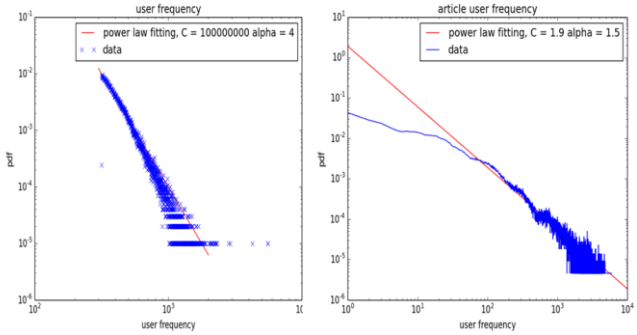

Figure. 1. The PDF as a function of (left) article frequency and (right) article user frequency.

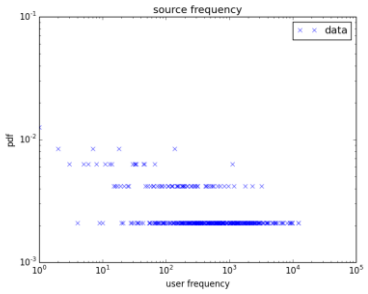

Figure. 2. The PDF as a function of source frequency.

As discussed above, we use our datasets to estimate the empirical density functions and fit them to the standard statistical distributions we expected (based on the shape of the distribution curves and on graph data analysis). We plot on a log-log scale the empirical estimate of P(X = x) from our data by group separately. The term article (terms, or source) frequency is the times an article (or terms, source) has been liked/shared/commented divided by the total number of likes/shares/comments of all articles (terms, sources). As shown in Figure 1 and 2, the power-law fits nicely to our data and alpha calculated by using Maximum Likelihood Estimation (MLE) is also specified in the graph itself. The result with generated figures and parameters, the exponential co-efficient C and alpha are shown in Figure 1. It is observed that the article and source frequency follows roughly power law with minimum cut-off around 10^2. But the source frequency data is scattered due to the limited number of source sites we have (500 sites).

VI. User Network Analysis Results and Findings

Most Influential Users: In order to select the most important user subgroup to generate user-user network, the most influential users are calculated based on HITS method. The users are sorted in descending order according to the hub/authority score. For comparison study, we also sorted the users according to the article numbers the user interacts, and find the overlap between them.

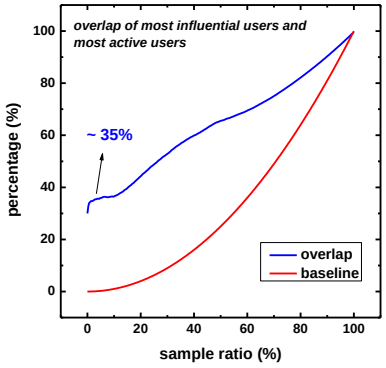

Figure. 3. The overlap between most influential users and most active users as a function of sample ratio.

Figure 3 shows the overlap between most influential users and the most active users. Around 1.45 million users are randomly chosen from all the users in this study. We sorted the users according to the article numbers the user interacts, and pick a subset of users in descending order, from 0% to 100%. The red curve is the baseline of the overlap percentage if we randomly choose two x% from some data. The blue curve is the overlap between users chosen according to its HITS results and article numbers. We find that with sample ratio < 10%, about 35% of users appear in both groups. This indicates that the users' activities did contain information on how important the users are, and it is a reasonable choice to choose users according to HITS results to balance the computational difficulty.

Comparison of Node Distribution in Different Networks: Three groups of graphs were generated based on article based, term based and site based similarities. We evaluated our

findings by plotting node distributions of multiple networks and marking their differences. For each pair of users, we determined whether there is an edge between them based on their similarities. There are two methods to generate the graph. *Method (1)* is to link two users with probability P in proportional to their similarity. Since the similarities in our previous calculations are already normalized, we just link the pair of users with P equal to the similarity of them. *Method (2)* is to set a threshold similarity and connect a pair of users if they have similarity above this threshold. Multiple thresholds are picked from similarity = 5% to 90% for comparison studies. It is first assumed that all the node distributions obey power laws and try to find the co-efficient alpha. Then we will discuss the deviation and the potential solution for a better fitting strategy.

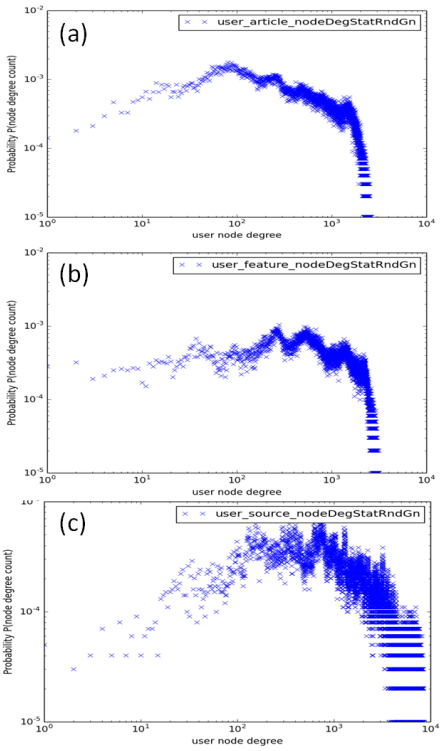

Figure 4 (a) Node distribution of article based user graph. (b) Node distribution of term based user graph. (c) Node distribution of site based user graph.

Figure. 4 (a)(b)(c) show the node distributions of article based, term based and site based user graphs. It is observed that none of them obey power law distribution and all three figures have a bump in medium node degree (~ 10^2 level). They have some characteristics of power law distributions but have sharper falling trend with high node degrees and is adjusted by randomly generated networks for low node degrees. It is consistent with our generating process since we link each pair with a probability equal to their similarity, and the major portion of the nodes will thus have a medium degree.

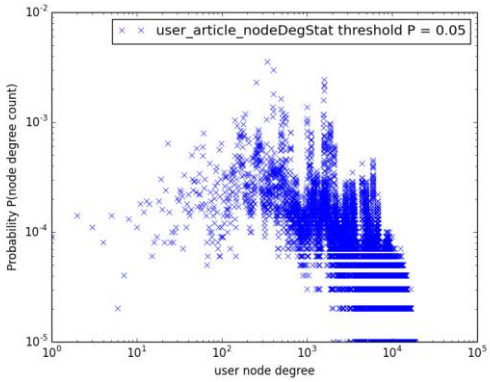

Figure 5. PDF as a function of user article node degrees with threshold similarity = 5%

It is expected that Method (2) would generate different graph node degree distributions from method (1) since we are assigning edges just based on whether its similarity is above the threshold or not. Figure 5 shows the node degree distribution based on article based similarity. The threshold we picked is similarity = 5%. The points on this figure has larger maximum of node degrees and generally obey the trend of that in Figure 5(a), but with much larger fluctuations and many scattering/noisy points. The points get cleaner and a power law distribution reveals just after we slightly increase the threshold similarity to 30%, as shown in Figure 6(a). There is also a heavy tail that couldn't be well fitted by power law. As we increase the threshold, the curve is cleaner but fewer nodes are connected, as shown in Figure 6(b)(c)(d), which are the node distribution for threshold similarity = 50%, 70% and 90%.

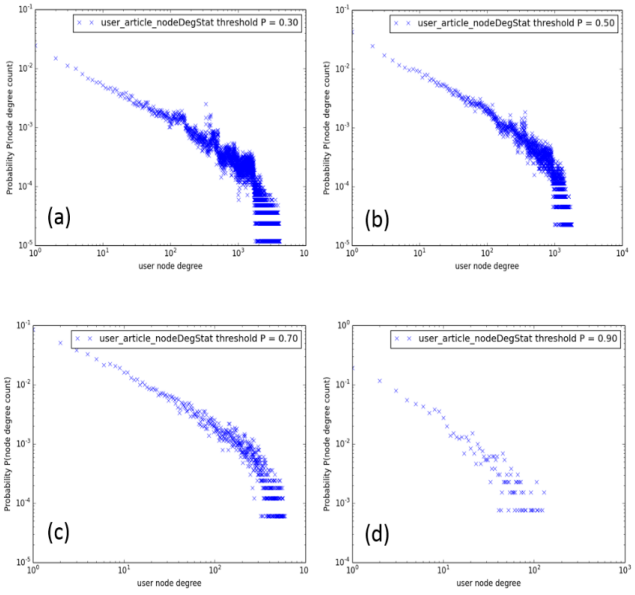

Figure 6. PDF as a function of user article node degrees with (a) threshold similarity = 30%, (b) threshold similarity = 50%, (c) threshold similarity = 70%, (d) threshold similarity = 90%.

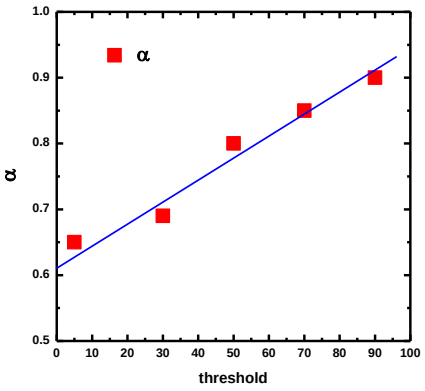

Figure 7. The alpha parameter in fitted power law distribution in article user network with multiple thresholds.

The α co-efficient increases when the threshold similarity increases, which means the curve on a log-log axis figure is getting steeper, because we are removing those links with low similarities and the nodes with higher degrees have more low similarity links and is more affected by this process. A plot of alpha as a function of similarity thresholds is shown in Figure 7. This process links the figure generated by method (1) and method (2), and it indicates that by setting the threshold similarity, useful characteristics are extracted from noisy scattered data.

We have also calculated the node distributions with different similarity bases and compared their fitting parameters. It is noticed that the graph generated based on sites has larger fluctuation than article or term based graphs, due to the small number of sites we have (we only have 500 sites) and the discreteness of the data we collected based on these sites. However, nice and clean curve can be extracted with threshold = 70%, and it is demonstrated that even we only have limited information, the user-user graph can still be established with good accuracy.

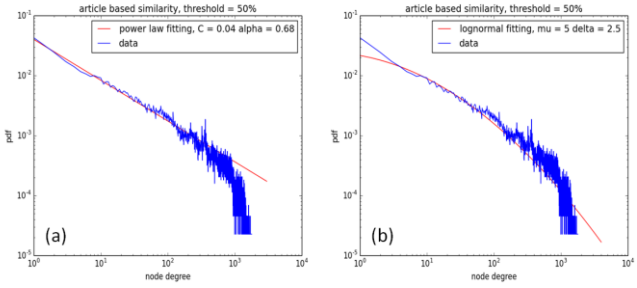

Figure 8. PDF as a function of node degree and the fitting by (a) power law distribution and (b) lognormal distribution.

Distribution Analysis and Fitting: The distribution function is investigated in this part. The results of multiple fitting will be shown below and we will evaluate the effectiveness of the fitting algorithms by the discrepancies of the fitted curve with the original plots. First, we should notice is that the distributions cannot be well fitted by a power law distribution because of the tail at higher node degrees, as illustrated in Figure 8(a). Lognormal distribution, which is the logarithm of normal distribution, is usually used to fit skewed power distributions since its probability density function (pdf) on log-log plot has a tail like behavior. The pdf of lognormal distribution is:

$$PDF(lognormal) = \frac{1}{x\sigma\sqrt{2\pi}} exp(-\frac{(lnx-\mu)^2}{2\sigma^2})$$

Where are have μ as the mean and σ as the standard deviation.

The fitting of article based node distribution with threshold 50% is shown in Figure 8(b). However, we still observed discrepancies between the theoretical calculation and extracted data within both low and large node degree regions. As a result, some other algorithms should be used to fit the data, such as DPLN, which works very good on distributions with heavy tails.

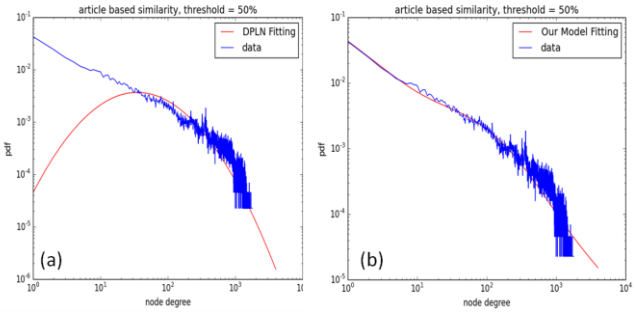

Figure 9. PDF as a function of node degree and the fitting by (a) power law distribution and (b) lognormal distribution.

The heavy tail part is fitted very well by DPLN, but it doesn't work well with low node degrees. As instructed in the algorithms, we proposed a combination of power law and DPLN and the fitting results is shown in Figure.9(b). The discrepancies between fitted curve and original data are much smaller, indicating better fitting we got with our proposed model.

Clustering Coefficient: The results of clustering coefficients for user-user network based on term features are shown in Figure 10.

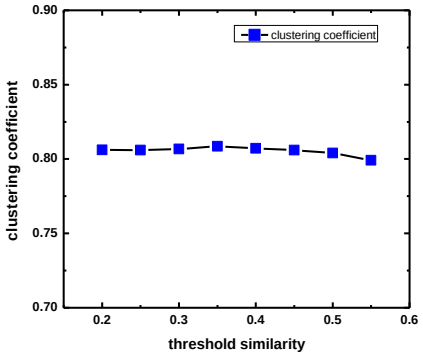

Figure 10. The clustering coefficient of user-user network based on term features.

A distributed algorithm was implemented to compute the clustering coefficient due to the large number of nodes there are in the network, as was discussed in the algorithms part. It is observed that changing the threshold similarity doesn't change the clustering coefficient, as it always stay at around 0.8. Since we didn't include the nodes with zero neighbors, the constant coefficient is an indication that the network structure is resistant to removing less similarity structures.

Suggesting Reputable Curators: As part of the project we also calculated the top sources based on HITS and found out that the most ranked fan pages are different from the fan pages that have the most number of users following them or the ones with the most of number of articles published. This could be used as one of the features as part of the recommendation system, and is helpful to evaluate the effectiveness of data crawling on those website for future study.

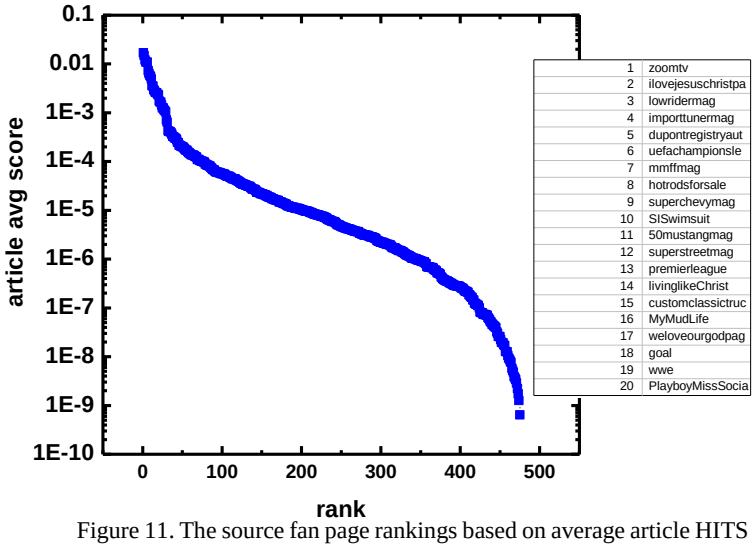

Figure 11. The source fan page rankings based on average article HITS score.

The sources can be ranked by the average of its article rankings, or by the average scores of its articles. The latter one was chosen in this work since we are emphasizing the articles with higher scores, or in other words, with better ranks. The source fan page ranking based on average article HITS score is shown in Figure 12. The top 20 source sites are also listed on the right side of the figure.

Recommendation System: The recommendation system is built based on our extracted terms and collaborative filtering. Detailed process was discussed in the algorithms part. To evaluate the system, part of the data is reserved for evaluation, as shown in Figure 12.

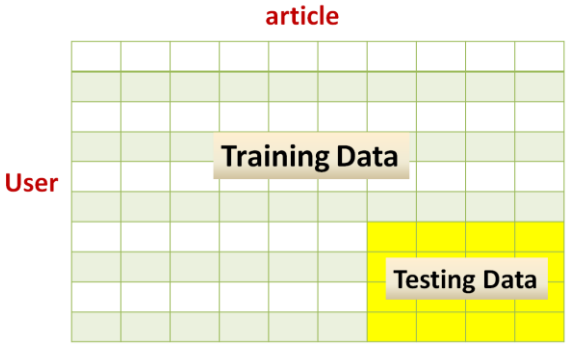

Figure 12. Illustration of the training and testing data set.

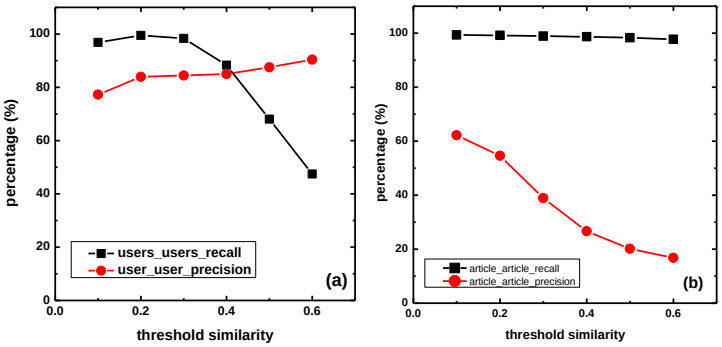

Figure 13. The recall and precision of the recommendation system based on (a) user-user similarity and (b) article-article similarity.

For the recommendation system based on user-user's similarity, as shown in Figure 13(a), the precision are around 80%, and increases as a function of threshold similarity, and the recall has a peak near similarity ~ 20%, and drops more suddenly when similarity increases. As a result, the recommendation systems always make "correct" prediction but may not be able to predict all the possible situations since the threshold is increasing. For the system based on article-article's similarity, as shown in Figure 13(b), the precision is much lower, starting from 60% and keeps decreasing to 20% when similarity increases, while the recall is almost 100% for all similarities. This indicates article-article based system can make accurate predictions but the lack of article data is a series limitation. It is also partly due to lots of users are having similar interests, and for a certain user, he/she will like lots of things in un-related fields. It is better to use user-user similarity for recommendation systems unless there is sufficient amount of data. We would like to optimize the recommender system better by using a better similarity metric such as LDA or Pearson's Correlation rather than Jaccard similarity.

VII. Conclusions

To summarize, we built user networks based on extracted user information of their likes/shares/comments on articles from Facebook fan pages. We first extracted the most influential users and then analyzed the network in detail. It was found that the node distribution roughly follows power law distribution. The skewed distribution is fitted by lognormal distribution and

DPLN method but both have discrepancies with the original data set. We proposed a new model which combined power law and DPLN distribution and demonstrated that it has the best fit to the node distribution to the Facebook user network. Comparison study among three different similarity based on article, terms and sites are done in terms of node distribution and clustering coefficient. A recommendation system based on our user-user similarity and article-article similarity is built and it is found the former one has better performance.

VIII. Acknowledgement and Contribution

The authors would like to thank Prof. Jure Leskovec and Clifford Huang and other TAs for their help in this project.

Sudharsan proposed this project and we would like to credit everybody in the team equally for their contribution in every part of the project from data collection to network analysis to term extraction to building a recommendation system based on influential users.

IX. References

[1] D. M. Pennock, G. W. Flake, S. Lawrence, E. J. Glover, and C. L. Giles. "Winners don't take all: Characterizing the competition for links on the Web." Proceedings of the National Academy of Sciences, 99(8):5207–5211, 2002.

[2] M. Seshadri, S. Machiraju, A. Sridharan, J. Bolot, C. Faloutsos, J. Leskove, "Mobile call graphs: beyond power-law and lognormal distributions", KDD, 2008

[3] John S. Breese, David Heckerman, and Carl Kadie (1998). "In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence (UAI'98)". |chapter= ignored (help)

[4] Peter, Brusilovsky (2007). *The Adaptive Web*. p. 325. ISBN 978-3-540-72078-2.

[5] J. McAuley, J. Leskovec, "Hidden Factors and Hidden Topics: Understanding Rating Dimensions with Review Text." ACM Conference on Recommender Systems (RecSys), 2013.

[6] Tf-Idf (Link: http://nlp.stanford.edu/IR-book/html/htmledition/tf-idf-weighting-1.html)

[7] Facebook open graph API (Link: https://developers.facebook.com/docs/graph-api)

[8] 500 top media/entertainment FB Fan pages (Link: http://bit.ly/14g9IVq)

[9] Normalized google distance (Link: http://en.wikipedia.org/wiki/Normalized_Google_distance)

[10] Hadoop (Link: http://hadoop.apache.org)

[11] Recommender systems. (Link: http://web.stanford.edu/class/cs246/slides/07-recsys1.pdf)

[12] Association rules. (Link: http://web.stanford.edu/class/cs246/slides/02-assocrules.pdf)