Gossip-Based K-Means Membership Protocol

Rishina Tah
Columbia University
116th & Broadway
New York, New York- 10025
+1-917-971-9248
rt2545@columbia.edu

Surashree Kulkarni
Columbia University
116th & Broadway
New York, New York- 10025
+1-765-838-9818
ssk2197@columbia.edu

ABSTRACT
The two foremost examples of efficiently and carefully engineered, reliable networks are the Internet and the telephone network. Apart from these, there are several unpredictable networks which also need algorithms for communication and information dissemination- the biggest, most important in today's times being *social networks*. Gossip algorithms are fundamentally based on the inherent unreliability and non-synchronicity of information exchange, or in layman terms, *rumours*. The immense simplicity of these helps us analyze social networks using various interdisciplinary tools like Optimization, Percolation, Random Graphs, etc. The focus of our paper is to use the *Aggregating Gossip Algorithm* on dataset obtained from SNAP (Stanford Network Analysis Project).

General Terms
Algorithms, Measurement, Performance, Design, Reliability, Experimentation.

Keywords
Gossip Algorithms, Social Networks, Rumours, Aggregates, K-Means Clustering, Groups, Clusters.

1. INTRODUCTION
With the invention of firstly, the telephone network, and next, the Internet, no person is too far away to converse with. These massive, methodical networks help us communicate up to very long distances, at very high speeds. In contrast to these networks, there has also been a sudden emergence of different types of large networks that do not primarily serve as a communicating tool. Some of these include sensor networks, peer-to-peer (P2P) networks, mobile ad-hoc networks, and social networks. This paper is dedicated to exploring social networks [2].

Facebook, LinkedIn, Twitter, etc. are prime examples of enormous networks that have over a billion users, and generate tremendous amount of data every moment. The ubiquitous presence of these great social networks renders the need to develop algorithms to analyze the social communication between two users.

1.1 What is Gossip?
According to Wikipedia, gossip is defined as *idle talk or rumour, especially about personal or private affairs of others*.

However, in this paper, we treat gossip as any piece of information exchanged between two nodes. We have used datasets containing a technical paper citation graph. Here, gossip between two nodes implies Paper X saying, "Hey, I have been cited m times, what about you?" to Paper Y, and Paper Y replying, "Oh, great. I've been cited n times."

To analyze this kind of information exchange, we need simple and robust gossip algorithms.

1.2 Why Gossip Algorithms?
In most of these recent next-gen networks, the degree of unpredictability in infrastructure is high- one cannot say when a new user joins a social network, leaves it, goes offline for an extended period of time, or becomes overly active for some time. Thus, an advertiser will need an efficient social network algorithm for efficient advertising in a manner that if user A deletes his Facebook (say) account, his friends should not suddenly become unavailable to receive this advertising [2].

Hence, this brings us to our agenda: write an effective Gossip algorithm that-

a) At each node i, utilizes only "local" information, that is, information obtained from neighbours of i, N(i).

b) Does not require synchronization between i and its neighbours N(i).

1.3 What are Gossip Algorithms?
Consider a high school cafeteria where students get together every day to eat lunch. Each student sits with one other student, chosen at random, to share gossip. Alice tells Bob that Eve got accepted into Juilliard. The next day, Bob tells this to Dave, while Alice to Charles. In this way, this piece of information spreads in a robust manner.

A gossip protocol expressed in more technical terms is one that follows certain conditions [8]:

- The core of the protocol involves periodic, pair-wise, inter-process interactions.
- The information exchanged during these interactions is of bounded size.
- When agents interact, the state of at least one agent changes to reflect the state of the other.
- Reliable communication is not assumed.
- The frequency of the interactions is low compared to typical message latencies so that the protocol costs are negligible.
- There is some form of randomness in the peer selection. Peers might be selected from the full set of nodes or from a smaller set of neighbours.

- Due to the replication there is an implicit redundancy of the delivered information.

1.4 Gossip Algorithm Types

- Dissemination Protocols- These use gossip to spread information, basically by flooding a network [8].
- Anti-entropy Protocols- These are used for repairing replicated data, by comparing replicas and reconciling differences [8].
- Protocols that compute aggregates- These calculate a network-wide common aggregate by sampling information at the nodes are combining their values at every step [8].

This paper focuses on the last protocol.

1.5 K-Means Clustering

The k-means clustering algorithm aims to partition n observations into k clusters in which each observation is assigned to a cluster when it passes a certain minimizing criterion for that cluster, known as the "inertia" or within-cluster sum-of-squares (WCSS)- intuitively, finding the nearest mean[1].

There are two steps to k-means:

i) Assignment step: Assign each observation to the cluster whose mean yields the least WCSS.

ii) Calculate the new means to be the "centroids" of the observations in the new clusters.

The algorithm has converged when the assignments no longer change.

Our algorithm, as a last step, uses k-means clustering.

1.6 Communities & Groups

The principle of homophily dictates that a person is always attracted to another person *like him*. This gives rise to groups or communities that are connected together due to a certain common factor- location, workplace, school, etc. For example, interaction between colleagues in a company X on LinkedIn or friends in college Y on Facebook will be most, as compared to their individual interactions with nodes outside this group. Thus, the formation of communities is inevitable [9].

Our gossip algorithm uses k-means to essentially find out all these clusters and based on the citation score of each cluster, find out the most popular papers of say year x. This result can be paralleled to other datasets to find out "popular clusters"- that is, clusters that have higher scores.

1.7 Datasets

We have run our algorithm on the citation network dataset obtained from the Stanford Large Network Dataset Collection [6].

This citation graph covers all the citations within a dataset of 34,546 papers with 421,578 edges. If a paper *i* cites paper *j*, the graph contains a directed edge from *i* to *j*. If a paper cites, or is cited by, a paper outside the dataset, the graph does not contain any information about this [4].

The data covers papers in the period from January 1993 to April 2003 (124 months) [5].

2. OUR ALGORITHM

Input: Citation Graph

Output: Disjoint k clusters of Papers.

Algorithm:

i) Read the graph and for each paper, store the *citation number*-

$C(i)$ = Number of times Paper *i* has been cited in a citation dictionary.

ii) Gossip by pair-wise information exchange:

for every pair of Papers (u,v)

if ($| C(u) - C(v) | <= \alpha$)

Assign to *u* and *v*, new citation numbers = $(| C(u) - C(v) |)*0.5$

iii) Use k-means clustering to find clusters of most and least cited papers.

The algorithm takes as input a citation graph. Each line in the input file consists of a pair of nodes *i* and *j*, where Paper *i* has been cited by Paper *j*. Compute the citation number for each paper, that is, calculate for each paper, the number of times it has been cited in other papers. These citation numbers are stored in a citation dictionary.

The algorithm then iterates over all the nodes, forming pairs and checking if the citation number of node 1 is "close enough" to citation number of node 2. If the difference between the citation number of node 1 and node 2 is less than the *closeness constant* α, then the citation number of node 1 and node 2 is reset to the average of the citation number of the two nodes.

Once all the citation numbers are reset, k-means clustering algorithm clusters papers based on their new citation numbers. We can thus find most-cited and least-cited papers.

3. RESULTS & ANALYSIS

The citation graph has a dataset of 34,546 papers with 421,578 edges. Using our algorithm, we cluster our graph nodes into k=10 clusters. The k-means algorithm uses the difference between the citation numbers as a metric to assign membership to clusters. The program takes only 5 seconds to run on an average. The output consists of the cluster centroids and list of nodes that are in the clusters.

Our idea behind using a gossip based membership protocol was that the pair wise information exchange can help in decreasing the computation required in the metric calculation for k-means clustering process.

As the output, we can get most-cited and least-cited papers in different clusters. This algorithm can be applied to other datasets, especially over data from social networks, to find popular users and not-so-popular users. Advertisers can thus find which users to target.

One issue with the algorithm is that there are redundant cases where pairs (x,y) that already know the gossip they were going to exchange are explored.

4. EXPERIMENTAL TOOLS

The source code that implements our algorithm is written in Python. We have used the NetworkX, Math and Random libraries. The NetworkX graph library is used to read the citation graph [10].

5. FUTURE WORK

Future work involves:

The study of:

i) Other gossip algorithms combined with other clustering algorithms and comparing their results against the algorithm proposed in this paper.

ii) Epidemic algorithms- Gossip protocols can be used to propagate information in a manner similar to the way that a viral infection spreads in a biological population [3].

Also, analysis of the results by changing in k-means clustering-

a) Number of iterations

b) Selection of seeds (centroids)

c) Number of clusters

-to get the optimal solution.

Additionally, analysis of the effect of formation of groups due to the exchange of gossip- both positive and negative, including how fast does information spread considering the nature of information.

6. REFERENCES

[1] Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011. DOI= http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[2] Shah, D. (2009). Gossip Algorithms. Foundations and Trends® in Networking, 3(1). DOI= http://web.mit.edu/devavrat/www/GossipBook.pdf

[3] Massoulié, L., & Draief, M. (2009). Epidemics and Rumors in Complex Networks (pp. 1–125). Cambridge University Press.

[4] J. Leskovec, J. Kleinberg and C. Faloutsos, ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2005. DOI= http://www.cs.cmu.edu/~jure/pubs/powergrowth-kdd05.pdf

[5] J. Gehrke, P. Ginsparg, J. M. Kleinberg. SIGKDD Explorations 5(2): 149-151, 2003. DOI= http://www.cs.cornell.edu/home/kleinber/kddcup2003.pdf

[6] Stanford Network Analysis Project. DOI= http://snap.stanford.edu/

[7] K-Means Clustering, Wikipedia. DOI= http://en.wikipedia.org/wiki/K-means_clustering

[8] Gossip Protocol, Wikipedia. DOI= http://en.wikipedia.org/wiki/Gossip_protocol

[9] Chaintreau, Augustin, What are natural communities, divisions? Who is likely to love/hate my blog?, COMS-6998-2: Social Networks. DOI= http://socialnetworksfall14.wikischolars.columbia.edu/file/view/SN14-6Groups.pdf/527349238/SN14-6Groups.pdf

[10] NetworkX Documentation. DOI= https://networkx.github.io/