People and Entity Retrieval in Implicit Social Networks

Suman K Pathapati, Subhashini Venugopalan, Ashok Pon Kumar, Anuradha Bhamidipaty
IBM Research - India
Bangalore, Karnataka – 560045
{spathapa, subhvenu, ashokponkumar, abhamidi}@in.ibm.com

***Abstract*—Online social networks can be viewed as implicit real world networks, that manage to capture a wealth of information about heterogeneous nodes and edges, which are highly interconnected. Such abundant data can be beneficial in finding and retrieving relevant people and entities within these networks. Effective methods of achieving this can be useful in systems ranging from recommender systems to people and entity discovery systems. Our main contribution in this paper is the proposal of a novel localized algorithm that operates on the sub graph of the social graph and retrieves relevant people or entities. We also demonstrate how such an algorithm can be used in large real world social networks and graphs to efficiently retrieve relevant people/entities.**

***Index Terms*—Social Networks, Social Graph, Implicit Social Networks, Retrieval.**

I. Introduction

Social networks on the internet have become ubiquitous in the past decade with nearly three quarters of all internet users belonging to at least one online social network [1]. The advent of explicit social networks and the interconnectedness of entities on the internet are leading to the emergence of a large number of implicit and explicit social networks. An implicit social network is an implied social network [2] where people are connected to each other through entities; explicit social networks are formed by people explicitly connecting to each other. A consequence of this explosion of implicit people and entity networks is that people and their connections are instantly accessible on a single platform. At the same time people connect to entities on the internet in a much more intimate way by liking, favoriting, rating, bookmarking, and contributing. One of the implications of this kind of behavior is the emergence of implicit social networks. Implicit social networks capture rich amounts of information about people and their interaction with the world wide web and online entities [3]. In this paper we are primarily interested in finding an effective means for discovering relevant people and entities by mining the information captured in implicit social networks.

Finding relevant people online is a challenging task. Traditional Information Retrieval (IR) science is aimed at retrieving unstructured digital documents. The problem of identifying and retrieving relevant people and implicit entities poses a distinct challenge. Unlike digital or web documents whose data is self-contained, the representation of people and implicit entities online is a poor, noisy and highly distributed sampling of their true attributes. Hence there is a dearth of both structured and unstructured data that can be used to effectively discover/retrieve relevant people/entities using traditional information retrieval methods. Mining implicit social networks and the information they capture can be highly beneficial in overcoming the limitations of traditional IR in discovering relevant people/entities. If this problem can be effectively addressed there are a lot of applications where it would be useful. These range from searching for prospective employees [3] to targeting consumers with personalized ads and recommendations [4]. However, any technique or algorithm that is aimed at traversing implicit social networks to retrieve a highly targeted set of relevant people or entities should be efficient, decentralized, localized, and robust to network change in order to be effective.

II. Related Work

Implicit social networks have been well studied both offline and online. Even before the emergence of social networks online, implicit social network analysis based on the email traffic was done by Schwartz and Wood [5]. There have also been extensive studies on user similarity and familiarity by Guy et al [6]. In their paper, they categorize the sources of similarity among people into three different categories, namely place, thing and people. They also look at the overlap between the people in these three categories of sources. With the advent of the large number of online social networks, there has been a lot of interest in this area to mine relationships. Recommendation systems [7], [8] vary in functionality, which ranges from recommending people to objects of interest. Bonhard et al [8] go on to describe how the profile similarity of users can be used to enhance the recommendation systems for finding objects of interest.

ReferralWeb [9] uses the co-occurrence of names of people in close proximity in the documents available on the web to create a network, which can be used to improve the search results. It is based on the theory that, people who are close together in a referral chain trust each others' work more than a work by a third party. Ramanathan et al [10] analyse implicit social networks that are created based on the ability to share files with other people. Xiao et al [11] explore implicit social networks created based on access log parameters. Cetintas et al [12] propose a discriminative probabilistic model for professional networks that identifies latent content and graph classes for people. They use profile content features, social graph features, and social network usage statistics and learn a specialized similarity model for each latent class. Zhang et al [13] analyze many graph search algorithms that are used for finding experts. They create a controlled, simulated dataset based on the well known publicly available Enron email dataset [14]. They then examine the Breath First Search, Random Walk Search, Best Connected Search [15], Weak Tie Search [16], Strong Tie search, Hamming Distance Search [17], Cosine Similarity Search [18], Information Scent Search [19] algorithms on this social graph.

The Breath First Search does not use any similarity filtering and keeps traversing all the child nodes from each node in the graph. Though it is good for finding the nodes that are closest to the source [13], it does not necessarily return relevant results. Random Walk Search just spreads through the social graph randomly choosing one node for each traversal, and Zhang et al [13] use it as a benchmark to compare other search algorithms. Best Connected search [15] always takes the node that has the most connections, thus going through the most well connected people in the network. Though Best Connected Search performed better it turned out that a few nodes assumed more significance compared to the other ones, there by becoming the masters who control the performance of the entire network.

In Weak Tie Search, during node traversal, the node with the least number of incoming connections from the user is chosen as the next node to traverse. Whereas in Strong Tie Search, during node traversal, the node with the most number of incoming connections from the current node is chosen as the next node to traverse. Though the Weak Tie Search works better than the Strong Tie Strategy in cases where the search is performed in order to find persons to ask for help, they performed badly when compared to Best Connected Search.

Hamming Distance Search chooses the node with the most uncommon set of connected nodes compared to the current nodes. Since the very definition of Hamming Distance Search favors nodes with most outbound connections, the Cosine Similarity Search improves upon it by dividing the Hamming distance by the total number of outbound connections that a node has. According to Zhang et al [13], Hamming Distance Search does as good as the Best Connected Search.

The information Scent Search ignores the connections to find the next node, and rather uses the similarity of the profile of the node compared to the query to select the next node. As it turns out its performance is only as close to that of Weak Tie Search.

Alsaleh et al [2] proposes a hybrid approach for the recommender systems, in which a combination of the explicit and implicit social network information is used in the recommendation generation process. It first tries to cluster users and then use compatibility scores between pairs to recommend.

A. Contribution

In this paper we suggest a novel way to address the problem of people and entity retrieval in implicit social networks. We propose a new, efficient, decentralized algorithm for autonomic implicit network traversal and people/entity discovery in unstable networks. Our algorithm uses the concept of cashflow and combines the advantages of a similarity measure, like in information search scent algorithm, to direct the crawl in the social graph and retrieve relevant entities.

B. Organization of this paper

In section III we describe in detail the topology of explicit, implicit and hybrid social networks with the help of a formal notation. In section IV we propose a new algorithm for people/entity retrieval in implicit social networks and explain the various factors that affect the behavior and outcome of the algorithm. In section V we carry out extensive experimentation on a real world implicit social network to validate and observe the algorithm and its constraints. Finally in section VI we summarize our work and conclude with a look at the future direction of this work.

III. Social Network Topology

The emergence of the internet in the last 15 years, the increasing affordability of consumer electronic/computing devices, and the decreasing cost of bandwidth have created conditions conducive for the rise of online social networks. As a consequence of this rise, human relationships have been digitized online at an unprecedented scale. Along with the emergence of these explicit social networks, there have also emerged implicit and hybrid social networks [2]. In the broadest sense ‘an online social network is a network of people connected to each other through an indicated or implied relationship’.

A. The Social Graph

The social graph is a discrete graph containing vertices and edges. The vertices in a social graph indicate people or other entities within a social network, the edges indicate relationships between the entities.

B. Directionality of Edges

In a social graph the edges can be directed or undirected. In an undirected social graph, the relationship is always a two way relationship. For example, on LinkedIn two people can be connected to each other only if both the people confirm the indicated relationship [20]. So, a social graph $G = (V, E)$, where V is the set of vertices in the social graph and E is the set of unordered pair of vertices from V that represents all edges in the social graph. i.e.

$$E \subseteq \{\{u, v\} : u, v \in V\}$$

Some relationships are unidirectional and need not be reciprocal. For example, on Twitter one can follow a set of people even if the people in the set do not reciprocate by following the original user [21]. So in this case $G = (V, A)$ where V is the set of vertices in the social graph and A is a set of ordered pairs of vertices from V, where order indicates the directionality of a relationship/edge. i.e. $A \subseteq \{(u, v) : u, v \in V\}$ and,

$$\text{for } \{u, v\} \in V,\ (u, v) \in A \ \nRightarrow\ (v, u) \in A$$

However, on most social networks, the social graph is made up of both directed and undirected relationships. For example, on Facebook [22] one can have an undirected relationship with other people as friends, at the same time they can also have a directed relationship with entities by liking them. So the social graph of a generic social network with both directed and undirected edges can be written as $G = (V, E, A)$ where V is the set of vertices in the social graph, E is the set of unordered pairs of vertices from V that represent all undirected edges in the social graph and A is the set of ordered pairs of vertices from V that represent all the directed edges within the social graph. i.e $G = (V, E, A)$

$$E \subseteq \{\{u, v\} : u, v \in V\}$$

$$A \subseteq \{(u, v) : u, v \in V\}$$

C. Explicit Social Networks

Explicit social networks are social networks where the vertices V in the graph G indicate people and the set of directed edges A, and undirected edges E represent the explicit relationships between people i.e the relationships/edges are explicitly indicated by the people on the network. For example, an explicit social network in Twitter allows one to either be a follower or a person being followed. i.e. $G = (V, E, A)$, and $V \subseteq \{P\}$ where P is the set of people on the social network.

D. Implicit Social Networks

Implicit social networks are social networks where the vertices V in the graph G comprise a set of people P and a set of non-people entities O and the each ordered and unordered pair in the set A and E respectively comprise of a one vertex from set P and one vertex from set O. This means in an implicit social network people are not connected to each other directly through a relationship; the only relationships in these networks are between people and non people entities. Hence, people are indirectly connected to each other on the network through other non people entities. For example, IMDb [23] can be thought of as an implicit social network where people like/rate movies, but two people rating or commenting on the same movie are indirectly related to each other.

$$G = (V, E, A)$$

$V \subseteq \{P, O\}$ where P is the set of people on the social network.

$$E \subseteq \{\{u_p, v_o\} : u_p \in P, v_o \in O\}$$

$$A \subseteq \{(u_p, v_o) : u_p \in P, v_o \in O\}$$

E. Hybrid Social Networks

Hybrid social networks are social networks where the vertices V in the graph G comprise a set of people P and a set of non-people entities O and the each ordered and unordered pair in the set A and E respectively comprise of a vertices drawn from V. This means in a Hybrid social network people can be explicitly connected to each other through relationships or they can be indirectly or implicitly connected to each other through relationships with non people entities on the network.

In this paper we are primarily interested in exploring the implicit social network aspects of implicit and hybrid social networks.

IV. Cashflow Algorithm

Finding relevant people and entities online is a challenging task. It is simple enough to search for somebody by their name or profession number of their other digitized attributes. But a vast majority of human attributes cannot be expressed or queried through a text box. Luckily, with the emergence of implicit and hybrid social networks and the way in which people connect with other entities online, some of these hidden attributes are captured within the network. It is our aim to use this explicitly/implicitly captured information within the network to find relevant people and entities within a social network. The fundamental underlying assumptions we make about relevance, similarity, people and entities in an implicit or hybrid social network is that

- "Similar people are connected to the same entities"
- "Similar entities are connected to the same people."

This implies that if two people are connected to the same thing then this connectivity implies that they are similar in some respect. We build on top of this underlying assumption to find a way to retrieve relevant people. Such an assumption also makes it easy for querying for relevant people based on entities rather than attributes. Eg: people on IMDb who like the movie Casablanca.

A. Relevance

From the above assumption it becomes clear that the social graph needs to be crawled or traversed to find relevant people. When you query using an entity as the input, 1) the people that are connected to it directly are the most relevant people that are retrieved. 2) The next most relevant people are people who are connected to this initial set of people by means of other entities. One can view this as a bi-partite graph between set of nodes of people and a set of nodes of entities. The primary aspects of relevance that are to be considered are

- Existence of a path (connectivity) from the input query node.
- Length of the path or distance from the input query node.

So, the more distant a person is from the initial query node on the social graph the less relevant/similar he is. This implies relevance/similarity is a localized phenomenon in our interpretation of the implicit social graph. This kind of localization of similarity/relevance means that the algorithm will only be operating on sub graphs within the global social graph rather than traversing the whole network. This makes the algorithm efficient, tractable and its complexity independent from the size of the social graph. These qualities also allow the algorithm to be run online in real time.

B. Similarity

However, leaving similarity/relevance up to connectivity and locality on the social graph would be a naive approach when there is rich attribute data about the nodes on the graph. Using this node attribute data we can further refine the similarity/relevance metrics of the algorithm which would at the same time improve the precision of the algorithm by fetching more relevant results and improve its time and space complexity by not pursuing paths in the social graph whose nodes are not relevant (even though they are connected to relevant nodes). Thus an additional factor that will determine relevance/similarity is

- a similarity metric based on the attributes of the node along with the interaction of the node with other entities in the network.

Without this node level similarity irrelevant results would accumulate at each level of graph traversal and seriously affect the precision and performance of the algorithm. However this mix of graph based or node attribute based similarity should be judged on a case by case basis. In some situations attribute similarity might not make sense, but they do add weight in a number of real world applications. The picture of the algorithm that emerges from the above discussion is a localized graph traversing algorithm that takes into account the node attributes and interactions when calculating similarity/relevance.

C. Cashflow

The central idea behind the cashflow algorithm is to seed the input query node or the starting node with some capital/cash. This capital/cash is then spread to its children and grand children. Children nodes that are more strongly connected get a proportionally bigger amount of capital. Nodes whose attributes are similar to the attributes of the query node get to keep and redistribute more capital. At each node the capital decays such that the farther away the capital travels from the root node the smaller is the total amount of capital propagating through the social graph. Finally propagation of cash or capital through the social graph stops when the capital/cash at any node falls below a given threshold. Since the relevance of nodes decreases the farther they are from the root node, beyond some point the nodes are no longer considered relevant. These boundary points are decided by the decay of capital and the threshold. In the end, the nodes of the subgraph that have the most cashflow (i.e have received and distributed maximum amount of cash) are the most relevant nodes. If G is the social graph then the cashflow algorithm for a node would look like the following pseudocode:

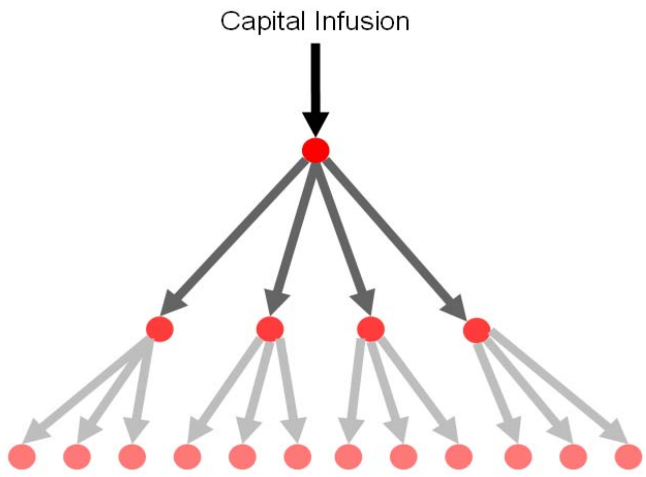

Fig. 1. A pictorial depiction of cashflow.

Algorithm 1 Cashflow Algorithm

```
function CASHFLOW($node, capital, decay, threshold, root$)
    $similarity \leftarrow getSimilarity(node.attr, root.attr)$
    $curCash \leftarrow capital * decay * similarity$
    $node.totalCash += curCash$
    if $curCash < threshold$ then
        return
    else
        for all $child$ in $node.getChildren()$ do
            $child.deposit = curCash/(child.edgeweight)$
            CASHFLOW($child, deposit, decay, threshold, root$)
            $Enqueue(child)$
        end for
    end if
end function

$root.deposit \leftarrow capital$
Enqueue($root$)
while Queue not empty do
    $node \leftarrow Dequeue()$
    $capital \leftarrow node.deposit$
    CASHFLOW($node, capital, decay, threshold, root$)
end while
```

In the pseudocode (1) $node.attr$ and $root.attr$ indicate the attributes of the $node$ and $root$ respectively. $curCash$ denotes the current cash or the cash received by the node. The edge-weight of a node is normalized by taking its weight as a ratio of the total edge weights of all its siblings connected to the same parent node in the given context. The value of decay is between 0 and 1, i.e. $0 \leq decay \leq 1$. *Note:* the concept of decay here is slightly counter-intuitive, i.e. a high $decay$ value means more cash and a low decay value indicates lesser current cash. At the end of the algorithm's execution, the nodes are sorted by their $totalCash$ attribute. The nodes with the highest $totalCash$ (or score) are the most relevant nodes.

The pseudocode also calls the $getSimilarity()$ function to compute the similarity of two nodes. In an implicit social network we are dealing with different kinds of nodes i.e people and entities. The similarity measure used in the algorithm should be customized to take these differences into account depending on the application at hand. In the next section, where we give the details of the algorithm's implementation on a real world implicit social network, we also describe our choice for the similarity measure.

V. GITHUB

Github [24] is a web based hosting service for 'git' repositories. 'git' is a distributed version control system that allows developers to collaboratively work with source of software projects. Github is the most popular source repository for opensource projects. Users can create projects, fork projects, follow users or projects and contribute to projects by sending push requests to the repository owner. Github since its inception has changed the way people develop opensource software. Besides having a vast userbase and a large number of projects, github also provides a REST API to get access to its public repositories and user specific data. Github is also a very good example of a real world implicit social network. Github can be viewed as a bipartite graph of users and repositories, meaning people are connected to each other through their connection to common or similar repositories. This kind of social graph topology and context can lend itself to many useful applications.

One example of such an application could be a developer search engine for recruiters at startups. Recruiters usually search for developers by searching for skills and platforms against a resume database. But the precision of these methods are poor, i.e out of the resumes that are fetched from a database only a very small percentage of them may actually be suitable for the job. One of the main reasons for this kind of low precision is the inability to accurately articulate the desired qualities, skills and background using standard textual queries alone. This is where the data encoded into an implicit social network like github can be extremely useful. Instead of searching for people based on what programming languages they know or what platforms they have worked on, one can simply select relevant people and projects on github and use the cashflow algorithm to find relevant nodes on the implicit social network. For example, if a small startup has a team of developers who are active on github and they want to hire a new team member, what they can do is, pick some of their existing team members with accounts (public) in github and generate a query to search for users with similar interests based on the cashflow algorithm. Such a query would return the people on github who have demonstrated a high degree of similarity to the existing team members as far as working on similar projects, programming languages, and platforms are concerned. Alternatively instead of using existing team members on github as a starting point, one could start with some initial set of relevant projects/repositories on github and get as a result, the list of people who have the greatest experience with the chosen projects. This is just one possible application of the data captured by github's implicit social network. In the rest of this section we demonstrate how one can use the cashflow algorithm in an implicit social network and demonstrate how the algorithm behaves. We experiment this on github's implicit social network.

A. The github social graph

Github is a complex social network with many different kind of nodes and complex connectivity between these nodes. In this subsection we simplify the social graph of github to consider only aspects that are relevant. Our github social graph has two kinds of nodes $projects$ and $contributors$. These nodes, along with the set of edges that connect them form the basic social graph. The nature of the elements in this graph are described in detail below.

a) Project: A project is a single repository or a set of repositories that have been forked from the same (master) repository. This assumption eliminates the complexities of forked repositories and its contributions and presents a much more clearer picture of the relationships between projects and contributors. It's also known that each project has a dominant programming language.

b) Contributor: A contributor is any github user that has committed/pushed code to a repository on github. By extension, any user that has contributed source code to a project is a contributor. It's safe to assume that each contributor is familiar with the programming languages of the projects to which he/she has contributed to.

c) Edge: A contributor is associated with a project if he/she has pushed code to the project repository. This association can be viewed as an edge. The edges can also be associated with a weight to indicate the normalized contribution of a contributor to the project. If a user has made 100 commits to a project with 1000 commits, his edgeweight for the project would be 0.1 .

d) Graph: The github social graph can be represented as $G(P, C, E)$, where G is an undirected, weighted, bipartite graph with one partition as the set of project nodes P and the other partition as a set of contributor nodes C. These two sets of nodes are connected to each other by a set of edges E. Each edge in E has an associated edge weight.

e) Cashflow: First, one or more root nodes on the social graph are picked to form the input query. The algorithm is then run on these input nodes by infusing them with some amount of capital each. A constant decay rate, and a threshold for the capital is set at the beginning of each run of the algorithm. The interplay between decay, threshold, capital, and network topology governs how the graph is crawled and the number of relevant results retrieved.

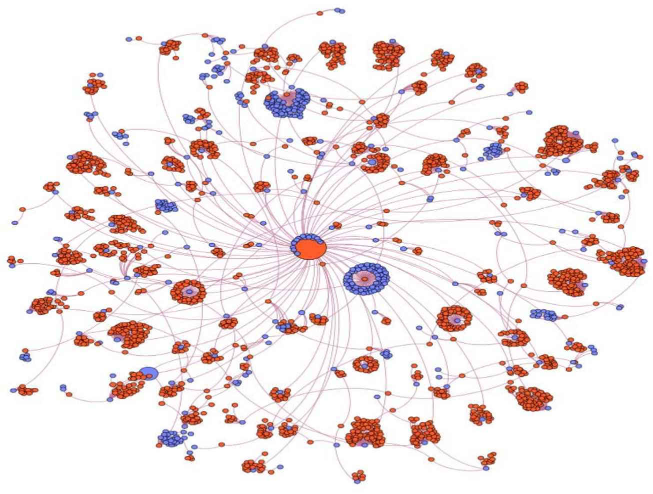

Fig. 2. A subgraph of the github implicit network used in our experiment. The graph is rooted at the 'jquery' project node (center).

B. Experimental Setup

We use data from Github to experiment and analyze the cashflow algorithm. The REST API of Github is used to fetch and gather raw data from the public repositories. This data is then processed to construct a simplified implicit social network subgraph with contributors and projects as the nodes. Edge weights are assigned for each contributor project edge. Any data that is fetched from github is cached locally. Moreover, the implicit social network subgraph that is constructed is also stored as a persistent graph database. During the execution of the Cashflow algorithm, data that is not present in the cache is fetched on demand.

According to the cashflow algorithm, the relevance or the score of a user depends on a) the $similarity$ function, b) $decay$ and, c) $threshold$. In our experiments, the similarity of users and projects is determined by considering the programming language as a similarity measure. Each project has one dominant programming language, which is obtained from the description of the project. Two projects are considered similar ($similarity = 1$) if they have the same dominant language otherwise they are dissimilar ($similarity = 0$). A user can have a list of familiar languages with varying proficiency, this is determined by his/her contributions to the different projects. We only consider the top 3 proficient languages of the user to determine similarity ($0 <= similarity <= 1$). Using the mentioned similarity function, several experiments were performed by varying the threshold and decay values, to observe changes in the resulting relevant users, and extent of the subgraph traversed.

The results shown in the following subsections were obtained by performing the experiments on the subgraph having its root node at the "**jquery**" project 2. The experiments have also been run on many other github projects, but for this demonstration we focus on "jquery" since it is a large project with close to a hundred contributors, and can thus help us study the algorithm better. For most of the experiments the $decay$ value was set at 0.9 unless mentioned otherwise. Recall that in the cashflow algorithm, a higher decay means lesser degradation with respect to depth (distance from the root node). Hence, inorder to observe how the cash flows to larger depths (in case it does) we have chosen a high decay value. The threshold was fixed at 0.0008 in most experiments. This low value for threshold will let us observe all non-negligible (incomparison with the capital) amounts of cash flowing in the network. We have also performed experiments with varying thresholds, $threshold \in \{0.0008, 0.008, 0.08, 0.8, 2.0\}$ to observe if there are any noticeable trends.

C. Distribution of Relevance

In the cashflow algorithm we are using total cashflow through a node to indicate its relevance. This implies that if we can observe the distribution of cashflow through the nodes we can observe the distribution of relevance in the social graph. In Fig. 3 we present the distribution of total cash (scores) for the most relevant contributor nodes (top 111) picked by our algorithm. The vertical axis represents scores of users and is plotted on a log scale. The users are placed on the horizontal axis. The graph in Fig. 3 shows three types of curves, to indicate the depth at which the user node is encountered. The nature of the cashflow algorithm allows cycles in the graph, and hence a node can accumulate points through a feedback mechanism when it is encountered multiple times during the traversal. Although a user node is present at only one depth, this specific graph (3) shows the scores of the user nodes at each depth they were encountered during the traversal. From the curves one can immediately observe that the distribution of cashflow/relevance is heavily skewed towards a few top contributors, with a large majority of contributors having much lower[1] scores in a smaller range. One can also notice that users that appear at more than one depth cause spikes in the graph, this is because the total cashflow is a positive additive quantity and grows when the same node is revisited. The corresponding cashflow distribution graph for projects (Fig. 4) also shows a

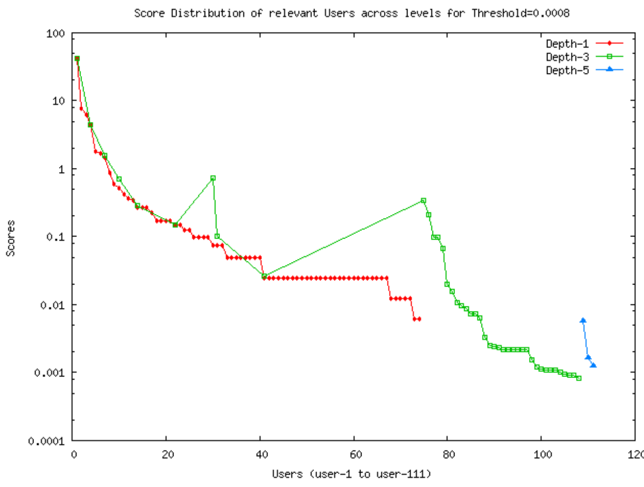

Fig. 3. Relevant Users' Scores at different depths.

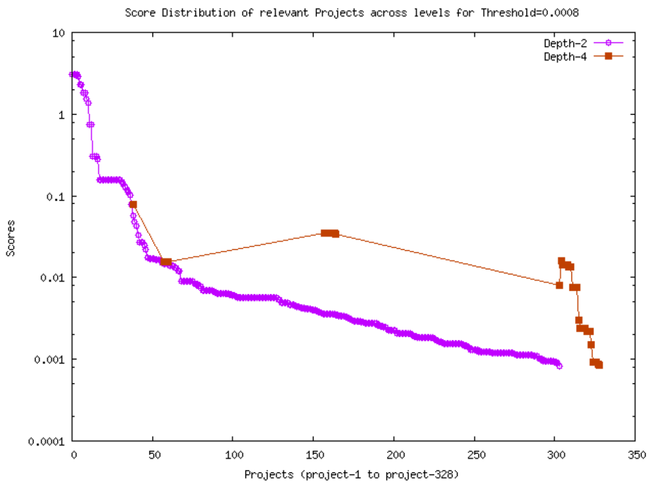

Fig. 4. Relevant Projects' Scores at different depths.

[1]Remember that scores are plotted on the log scale.

similar relevance distribution. There are about 10 projects with scores in the range 1 to 10. And a vast majority, of over 200 projects, with scores from 0.001 to 0.01.

The distribution of relevance is also interesting to observe at various depths, this is shown in Fig. 5. Here, we plot nodes at different depths against their total accumulated scores at the completion of the algorithm. Each curve in the graph represents the score distribution (in log scale) of a set of nodes at the same depth. From the Fig. 5 we can observe a steep fall-off in the relevance distribution of nodes at the first level of depth and the last level of depth. At levels in between (depth 2 to 5) we notice that the gradient is relatively more linear on the logscale. This kind of depth based distribution shows that on the first level the steep gradient is caused by high inequality of contribution at that depth and at the last couple of levels it's simply an indication of the low amount of cash flow propagated. The levels in between have moderate fall-off due to the wider redistribution of relevance starting from the second level.

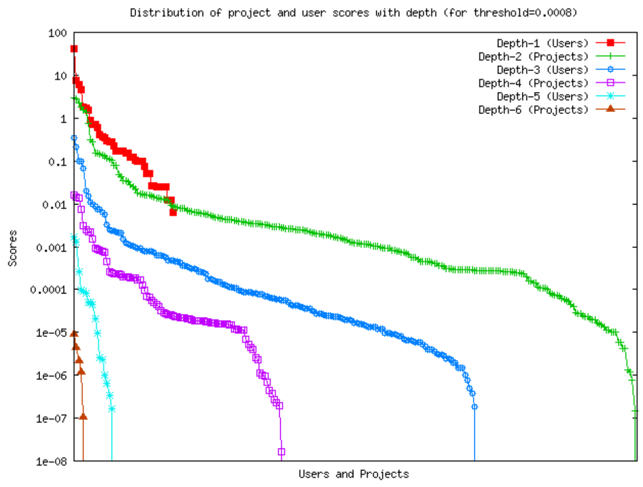

Fig. 5. Variation of Project and User Scores at different depths.

D. Relevance and Depth

The cashflow algorithm is built upon the assumption that the most relevant nodes are closer to each other in implicit social networks and are similar. One of the implications of this assumption is that, as depth to which cashflow is propagated increases, lower is the average capital infusion for the deeper nodes. The plot in Fig. 6 shows how cashflow or relevance score is distributed across nodes at different depths. The horizontal axis represents the depths traversed and the vertical axis represents the relevance scores of nodes. The curve on top is a plot of the sum of scores of all the nodes at the given depth. Since the scores are plotted on a log scale, what looks like a linear/gradual decrease in the score at each depth is infact an exponential drop off. This graph validates our theory that the cashflow algorithm does distribute the cash across depths in exponentially decreasing chunks and it is as intended. The gap between the relevance of different depths can be increased or decreased by tweaking the capital decay at each node.

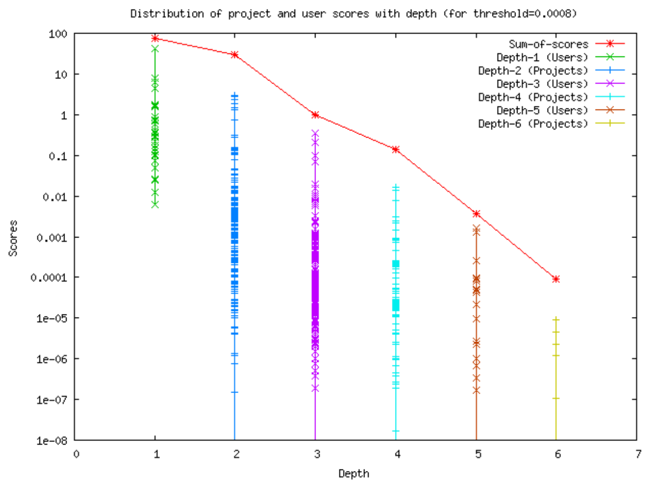

Fig. 6. Comparison of scores at different depts.

E. Threshold vs Relevant Nodes

The cashflow/relevance distribution in the social graph is governed by 3 things. *i)* the *decay* of cashflow at each node, *ii)* the *threshold* at which propagated capital infusion stops, and *iii)* the topology of the social graph. Of these, the topology of the social graph is out of scope of our control. However, *decay* and *threshold* are two values that can be used to tune the behaviour of the cashflow algorithm. With the interplay between these two values we can decrease or increase the number of relevant nodes that are retrieved and fine tune the quality of the retrieved results. The plot in Fig. 7 shows how varying threshold with a constant decay affects the number of relevant entities (users and projects) retrieved. As the threshold is increased there is a steep drop in the number of relevant entities. The drop is more gradual at larger thresholds.

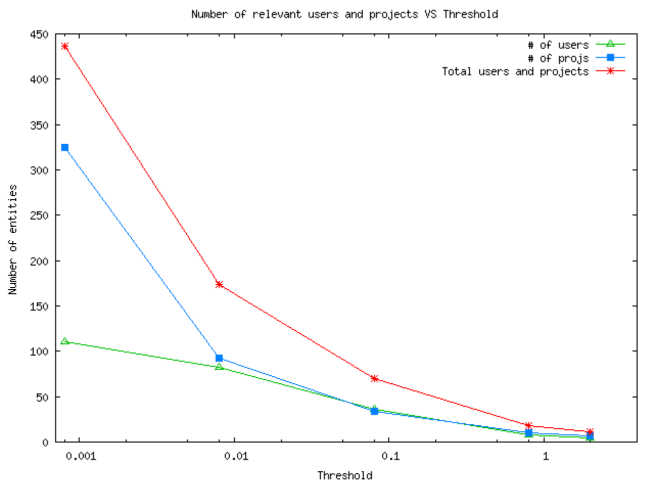

Fig. 7. Number of relevant items at different thresholds.

We can also vary the $decay$ parameter to affect the way and extent to which the graphs is traversed. As the $decay$ parameter in the algorithm is decreased, lesser cash flows from one level to the next (refer Note:IV-C).

F. Inference

The experimental section clearly demonstrates the cashflow algorithm behaves in the intended manner. The relevance of nodes is increased as nodes are revisited (and are more similar to our root). The algorithm allows us to quickly eliminate a large number of dissimilar (low score) nodes. More importantly, the most relevant retrieved exhibit a high degree of similarity with the root. We are also able control and tune the flow of scores and the crawl of the social graph by varying the $decay$ and $threshold$ parameters. We have successfully observed the behaviour of the algorithm in a real world implicit social network and have gained insights into how relevance/cash is spread through the system. These observations arm us with a deeper knowledge of the cashflow algorithm and can help us make effective use of it in other real world situations.

VI. Conclusion

In this paper we have discussed about explicit, implicit and hybrid social networks and their rapid emergence in todays connected world. We have also explored related work in the field of retrieving relevant users and entities in different kinds of social networks. Further we discussed the topology of explicit, implicit and hybrid social networks in section III. Then we presented a novel algorithm to retrieve relevant people and entities in an implicit social network in section IV called the cashflow algorithm. In section V we thoroughly demonstrated the behaviour of the cashflow algorithm in a real world implicit social network, Github.

The cashflow algorithm can be similarly used in a myriad other implicit social networks and situations. For example, finding people who are most likely to like a movie from their ratings data from IMDb, or, recommending friends to add in twitter based on the similar links and assets they share. From our experiments, we know that the Cashflow algorithm is simple enough to achieve much more complex behaviour and results. E.g. Instead of querying with a single node, we can choose a set of nodes to be the seeds or root nodes. The algorithm can be run unchanged and give out the most relevant results. The only change required will be with the similarity function used, inorder to take into account parameters from multiple seed nodes. We can use this sort of querying to generate multimodal queries where relevant nodes have a mixture of relevant qualities. We plan to explore these and other avenues in our future work.

References

[1] A. Moerdyck and InSitesConsulting, "940 million social media users in the world," http://blog.insites.eu/2010/03/22/940-million-social-media-users-in-the-world/, 2010, [Online].

[2] S. Alsaleh, R. Nayak, Y. Xu, and L. Chen, "Improving matching process in social network using implicit and explicit user information," in *Proceedings of the 13th Asia-Pacific web conference on Web technologies and applications*, ser. APWeb'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 313–320. [Online]. Available: http://dl.acm.org/citation.cfm?id=1996794.1996834

[3] D. Frey, A. Jégou, and A.-M. Kermarrec, "Social market: Combining explicit and implicit social networks," in *SSS*, 2011, pp. 193–207.

[4] M. Bertier, D. Frey, R. Guerraoui, A.-M. Kermarrec, and V. Leroy, "The Gossple Anonymous Social Network," in *ACM/IFIP/USENIX 11th International Middleware Conference*, Bangalore, India, Nov. 2010. [Online]. Available: http://hal.inria.fr/inria-00515693/en/

[5] M. F. Schwartz and D. C. M. Wood, "Discovering shared interests using graph analysis," *Commun. ACM*, vol. 36, pp. 78–89, August 1993. [Online]. Available: http://doi.acm.org/10.1145/163381.163402

[6] I. Guy, M. Jacovi, A. Perer, I. Ronen, and E. Uziel, "Same places, same things, same people?: mining user similarity on social media," in *Proceedings of the 2010 ACM conference on Computer supported cooperative work*, ser. CSCW '10. New York, NY, USA: ACM, 2010, pp. 41–50. [Online]. Available: http://doi.acm.org/10.1145/1718918.1718928

[7] P. Kazienko, K. Musia, and T. Kajdanowicz, "Multidimensional social network in the social recommender system," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 41, no. 4, pp. 746 – 759, 2011. [Online]. Available: http://dx.doi.org/10.1109/TSMCA.2011.2132707

[8] P. Bonhard, C. Harries, J. McCarthy, and M. A. Sasse, "Accounting for taste: Using profile similarity to improve recommender systems," vol. 2, Montreal, QC, Canada, 2006, pp. 1057 – 1066.

[9] H. Kautz, B. Selman, and M. Shah, "Referral web: combining social networks and collaborative filtering," *Communications of the ACM*, vol. 40, no. 3, pp. 63 – 65, 1997. [Online]. Available: http://dx.doi.org/10.1145/245108.245123

[10] M. K. Ramanathan, V. Kalogeraki, and J. Pruyne, "Finding good peers in peer-to-peer networks," in *In International Parallel and Distributed Computing Symposium (IPDPS*, 2002.

[11] J. Xiao and Y. Zhang, "Measuring similarity of interests for clustering web-users," in *Proceedings of the 12th Australian Database Conference ADC 2001.* IEEE Computer Society, 2001, pp. 107–114.

[12] S. Cetintas, M. Rogati, L. Si, and Y. Fang, "Identifying similar people in professional social networks with discriminative probabilistic models," Beijing, China, 2011, pp. 1209 – 1210. [Online]. Available: http://dx.doi.org/10.1145/2009916.2010123

[13] J. Zhang, "Searching for expertise in social networks: A simulation of potential strategies," in *Proc. of the 2005 Int. ACM SIGGROUP conf. on Supporting Group Work.* Press, 2005, pp. 71–80.

[14] W. Cohen, "Enron email dataset," http://www-2.cs.cmu.edu/ enron/, 2004.

[15] L. A. Adamic and E. Adar, "How to search a social network," *Social Networks*, vol. 27, p. 2005, 2005.

[16] M. Granovetter, "The Strength of Weak Ties," *The American Journal of Sociology*, vol. 78, no. 6, pp. 1360–1380, 1973.

[17] R. W. Hamming, "Error detecting and error correcting codes," *Bell Syst. Tech. J*, vol. 29, no. 2, pp. 147–160, 1950.

[18] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*, M. Granovetter, Ed. Cambridge University Press, 1994.

[19] B. Yu and M. P. Singh, "Searching social networks," vol. 2, Melbourne, Vic., Australia, 2003, pp. 65 – 72.

[20] Wikipedia, "Linkedin — Wikipedia, the free encyclopedia," http://en.wikipedia.org/wiki/LinkedIn, 2011, [Online].

[21] Twitter-Help-Center, "Twitter basics," http://support.twitter.com/groups/31-twitter-basics, 2011, [Online].

[22] "Facebook," http://www.facebook.com, 2011, [Online].

[23] IMDb, "Internet movie database," http://www.imdb.com, 2011, [Online].

[24] GitHub-Inc., "Github – social coding," https://github.com/, 2011, [Online].