

Ankush Garg
September 2009
Computer Science

Keyword based Social Networks: Models, Algorithms and Analysis

Abstract

In online social networks, users create a profile by setting some attributes that help in characterizing the user. We call these profile attributes the keywords of the user. In this work, we focus on social networks based on keywords and study how keywords can be effectively used to model and search in such networks. We make two main contributions in this thesis.

First, we study the relationship between semantic similarity of user keywords and the social network topology. We present a ‘forest’ model to categorize keywords and define similarity functions between a pair of users. Based on that, we model the social network topology and validate the model against a real life social network graph.

Second, we study unstructured keyword based social networks and propose an information flow model to disseminate keywords. Then, based on the given information flow, we design and develop a search algorithm to find users who have keywords, present in the search query, as part of their profile attributes. The search algorithm is based on a linear combination of topological distance and trust metrics. We observe an improvement in orders of magnitude when the search algorithm is compared to breadth first search.

Keyword based Social Networks: Models, Algorithms and Analysis

by

Ankush Garg

B.Tech., Computer Science and Engineering (IIT Delhi, India) 2007

THESIS

Submitted in partial satisfaction of the requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved by the Committee in Charge:

Dr. S. Felix Wu

Committee Chair and Professor of Computer Science

Dr. Charles U. Martel

Professor of Computer Science

Dr. Norman S. Matloff

Professor of Computer Science

2009

To my parents: Smt. Sheela Garg and Late. Dr. K. C. Garg

मातृदेवो भव। पितृदेवो भव॥ - तैत्तिरीयोपनिषद् १.११.२
Let your mother and father be like God unto you. – Taittiriya Upanishad 1.11.2

यं माता पितरौ क्लेशं सहेते संभवे नृणाम्।
न तस्य निष्कृतिः शक्या कर्तुं वर्षशतैर् अपि॥ - मनु स्मृति २.२२७
That trouble (and pain) which the parents undergo on the birth of (their)
children, cannot be compensated even in a hundred years. – Manu Smriti 2.227

Abstract

In online social networks, users create a profile by setting some attributes that help in characterizing the user. We call these profile attributes the keywords of the user. In this work, we focus on social networks based on keywords and study how keywords can be effectively used to model and search in such networks. We make two main contributions in this thesis.

First, we study the relationship between semantic similarity of user keywords and the social network topology. We present a ‘forest’ model to categorize keywords and define similarity functions between a pair of users. Based on that, we model the social network topology and validate the model against a real life social network graph.

Second, we study unstructured keyword based social networks and propose an information flow model to disseminate keywords. Then, based on the given information flow, we design and develop a search algorithm to find users who have keywords, present in the search query, as part of their profile attributes. The search algorithm is based on a linear combination of topological distance and trust metrics. We observe an improvement in orders of magnitude when the search algorithm is compared to breadth first search.

Acknowledgments

I would like to express my gratitude to my advisors, Dr. S. Felix Wu and Dr. Charles U. Martel, for guiding this work with utmost interest and care. I am grateful to them for giving me the freedom to explore my ideas. I also thank them for teaching excellent courses on Social Networks and Advanced Algorithms respectively, that laid the foundation for my decision to work with them.

I would like to thank Dr. Norman S. Matloff for finding the time to serve on my thesis committee and providing his valuable feedback.

I am thankful to my colleague, Prantik Bhattacharyya, for the assistance and support provided by him. It was a nice experience working with him on various concepts and writing papers together. I extend my thanks to members of the DSL research team, namely Lerone Banks, Juan Lang, Shaozhi Shawn Ye, Thomas Tran, Kelcey Chan, Xiaoming Lu, Matthew Spear, George and Ken Gribble for providing an enjoyable research environment. A special thanks to Mika, Xiaoming and Matt's daughter, for her cherubic responses during the weekly meetings.

I owe a special gratitude to my brother, Vandit Garg, for supporting and motivating me during tough academic times. The encouragement provided by him went a long way in helping me to move ahead in life in spite of the stuttering problem. I would also like to thank my mother, Smt. Sheela Garg, for being a constant source of strength in my life.

Finally, I would like to express my gratitude to my friend Sampath Goud who helped me in zeroing in on the Sanskrit verses used on the dedication page.

Contents

Abstract	iii
Acknowledgments	iv
Contents	v
List of Figures	viii
List of Tables	ix
1 Introduction	1
1.1 The Role of Keywords	1
1.2 Research Contributions	2
1.2.1 Social Network Model based on Keyword Categorization	3
1.2.2 Information Flow and Search in Unstructured Keyword based Social Networks	3
1.3 Organization	4
2 Social Network Model based on Keyword Categorization	6
2.1 Introduction	6
2.2 Why Categorize Keywords?	8
2.3 Social Network Modeling	11
2.3.1 Forest Structure	11

2.3.2	Similarity Functions	13
2.3.3	Social Graph	15
2.4	Evaluation Methodology	16
2.4.1	Analyzing the Facebook Network	16
2.4.2	Analyzing the Simulated Network	17
2.5	Results and Discussion	18
2.6	Conclusion and Future Work	21
3	Information Flow and Search in Unstructured Keyword based So-	
	cial Networks	22
3.1	Introduction	22
3.2	Background	25
3.3	Information Flow Model	28
3.3.1	Keyword Propagation Process	28
3.3.2	Identity Privacy Issues	31
3.3.3	Information Maintenance	31
3.4	Search Algorithm	32
3.4.1	Selecting Topologically Closer Nodes	32
3.4.2	Selecting Trustworthy Nodes	34
3.4.3	Querying with Keywords	35
3.5	Simulation Methodology	38
3.5.1	Graph Generation	39
3.5.2	Trust Distribution	39
3.5.3	Information Propagation: Keywords and Policies	40
3.5.4	Query Distribution	41

3.6	Results	41
3.6.1	Performance at various ρ values	42
3.6.2	Performance at various hop values	43
3.7	Related Work	46
3.8	Concluding Remarks	47
4	Conclusion	48
	References	54

List of Figures

2.1	Number of Distinct Keywords vs Keyword Frequency on log-log scale	9
2.2	Keyword Frequency Distribution	10
2.3	Forest with two component trees	12
2.4	$s(u,v)$ vs $k(u,v)$: All User Pairs	19
2.5	$n(u,v)$ vs $k(u,v)$: Direct Friends	20
3.1	Example of the community based social network	26
3.2	Network with restrictive propagation policy	42
(a)	No. of query messages vs ρ	42
(b)	Targets found/query message vs ρ	42
3.3	Network with liberal propagation policy	43
(a)	No. of query messages vs ρ	43
(b)	Targets found/query message vs ρ	43
3.4	Analysis of network (liberal policy) for varying hops	44
(a)	%age of targets found vs hops	44
(b)	Targets found/query message vs hops	44

List of Tables

2.1	Sample Users with Keywords	18
2.2	Weak Similarity with respect to Z	18
3.1	Keyword Forwarding Table	29
3.2	Keyword Received Table	33
3.3	<u>Q</u> uery <u>M</u> essages and <u>T</u> argets <u>F</u> ound per query at different hops for the network with liberal policy and $\rho = 0.5$	45

Chapter 1

Introduction

Social interactions form an integral part of the human behavior. With the advent of the internet, a new platform has emerged for human beings to interact. A whole range of online social networks (OSNs) like Facebook [1], Orkut [2], LinkedIn [3], etc. have emerged that allow users to discover, establish and maintain their relationships with others. Based on profile information and mutual interest, users become friends and join communities on the social network that allows them to interact and share information with each other. In this work, we focus on social networks based on keywords and study how keywords can be effectively used to model and search in social networks.

1.1 The Role of Keywords

In online social networks, users have to create a profile by setting some attributes that help in characterizing the user on the social network. We call these profile attributes the keywords of the user. These keywords not only define the interests,

passions and other traits of a user but also allow users to become friends with people sharing common attributes. An important property of social networks is that people tend to have attributes similar to those of their friends. Moreover, people adopt activities based on the activities of the people that they are currently friends with, and they form new friendships as a result of their existing activities [4].

In Davis Social Links (DSL) [5, 6], keywords are being used to develop a new internet model. The current communication model on the internet depends on the concept of ‘routable identity’, i.e. an identity is all we need to communicate with the owner of the identity, e.g. IP address, email address, etc. The key idea of DSL is to route packets from one network entity to another only via the social network paths between them. Further, the social paths are based on the keywords that are set by users as their profile attributes. Depending on the ranking of social paths the receiver can efficiently prioritize incoming packets (or messages) thereby having a better control over who can contact him. Thus, we can say that DSL tries to mimic the communication model in our human society and leverages online social networks for designing the next generation internet communication model. Deriving motivation from the DSL model, we study the role that keywords play in social networks from different perspectives.

1.2 Research Contributions

In this thesis, we address two important questions. First, how to categorize keywords effectively so that they can be used to quantify the similarity between different users and thereby how to model the social network? Second, how to disseminate

keywords and search for users in an unstructured keyword based social network (e.g. DSL)? We summarize the main research contributions made by this thesis in the following subsections.

1.2.1 Social Network Model based on Keyword Categorization

A user profile on an online social network is characterized by its profile entries or keywords. We study the relationship between semantic similarity of user keywords and the social network topology. First, we present a ‘forest’ model to categorize keywords and define the notion of distance between keywords across multiple categorization trees (i.e., a forest). Second, we use the keyword distance to define similarity functions between a pair of users and show how social network topology can be modeled accordingly. Third, we validate our social network topology model, using a simulated social graph, against a real life social graph dataset.

1.2.2 Information Flow and Search in Unstructured Keyword based Social Networks

In online social networks (OSNs), user connections can be represented as a graph. The network formed has distinct properties that distinguish it from other graph topologies. In this work, we consider an unstructured keyword based social network topology where each edge has a trust value associated with it to represent the mutual relationship between the corresponding nodes. Users have keywords as their profile attributes that have policies associated with them to define abstractly the flow of keyword information and the accessibility to other users in the network.

We also address privacy concerns as outlined in works on future OSN architectures.

Two key contributions are made in this part of the work. First, we develop an information flow model to disseminate keyword information when users add keywords as their profile attributes. Second, for keyword based queries, we design and develop a search algorithm to find users with the specified keywords in their profile attributes. It is based on a linear combination of topological distance and trust metrics. It is also dynamic in nature such that it adapts itself for each individual node during the search process. We observe an improvement in orders of magnitude when the search algorithm is compared to breadth first search.

1.3 Organization

In chapter 2, we discuss the proposed social network model based on keywords. We first analyze Facebook profiles to understand the keyword usage patterns and then, develop the metrics for measuring the similarity between user profiles. This work has been presented at the International Conference on Advances in Social Networks Analysis and Mining (ASONAM '09) [7] held at Athens, Greece in July, 2009 and appears in the conference proceedings.

Chapter 3 presents our proposed algorithms for information flow and search in unstructured keyword based social networks. We compare the search algorithm with breadth first search (BFS) for different keyword policies. We also present a brief description of the search algorithms from the literature related to peer-to-peer networks. This work has been accepted for publication in Workshop on Social Mobile Web (SMW '09) [8] to be held in conjunction with the IEEE International Conference on Social Computing (SocialCom '09) at Vancouver, Canada in August,

2009 and will appear in the proceedings of SocialCom '09.

We conclude the thesis in chapter 4 with a summary of the results. Here, we also discuss the possibility of combining both the techniques and the development of new search algorithms.

Chapter 2

Social Network Model based on Keyword Categorization

2.1 Introduction

In real life, individuals become friends when they share common interests or passions. Sociologists have termed this tendency of human beings as ‘homophily’. Similarly, on online social networks (OSNs), like Facebook [1] or Orkut [2], users establish friendships when they discover similar profile characteristics. The growth of LinkedIn [3], a social networking website, demonstrates the impact of profile information very well. Its purpose is to help people build professional networks and find career development opportunities. Using LinkedIn, employers can look into the profile information of users to search for potential employees. Similarly, it helps employees look for potential employers. We feel that categorizing profile information and correlating it with network topology constitutes an important step towards the study of OSNs.

Social networks is a widely researched area. Milgram [9] tried to ascertain if people in the society are linked by small chains. He asked people to forward letters to their friends who they thought were likely to know the target person. Thus, people implicitly made decisions based on their view of the geographical location or professional links of their friends and the associated likelihood of successful delivery of the letter. Lattice Model [10] uses geographical distance, a user trait, to model social networks. Models based on interest [11] and hierarchy [12] have also been proposed to model the friendship behavior of people. In Davis Social Links (DSL) [5], the social map is defined on the basis of keywords that are set by social peers as their profile attributes. Information transfer takes place only when a social path exists between the end users. Thus, it seems that keywords will play an important role in the development of future OSNs.

A typical user profile on an OSN is characterized by its profile entries (keywords) like location, hometown, activities, interests, music, etc. It is important to understand the use of keywords and how they can be used effectively to classify content in OSNs. Consider the scenario, where a newcomer in the city, say Bob, would like to find people interested in soccer. As he doesn't know anyone yet, he tries his OSN profile to search for soccer enthusiasts in the city but uses the word 'football' for the query. Though, both the words 'soccer' and 'football' can refer to the same sport, Bob's query returns no successful results because traditional residents use the word 'soccer' for the game. The system fails to understand the underlying semantic relationship between the keyword entered by Bob and profile entries of other users. This shows the importance of extracting relationship(s) from the diverse information provided by users.

Linguists have long been studying such relationships between words. Methods

like Latent Semantic Indexing [13] explored semantics based relationship among digital data. Similarity between users as a function of their topological distance was studied in [14]. Here, we study the relationship between semantic similarity of user keywords and the social network topology. First, we present a ‘forest’ model to categorize keywords and define the notion of distance between keywords across multiple categorization trees (i.e., a forest). Second, we use the keyword distance to define similarity functions between a pair of users and show how social network topology can be modeled accordingly. Third, we validate our social network topology model, using a simulated social graph, against Facebook [1] data.

Section 2.2 presents our findings on keyword usage patterns and discusses the need to categorize keywords. Section 2.3 describes the ‘forest’ model to categorize keywords. Here, we also propose functions to quantify similarity between users and a social network model based on those functions. Section 2.4 deals with the methods that we used to evaluate and validate our model. Section 2.5 presents results of experiments analyzing the proposed model and realistic data. We conclude in section 2.6 with possible extensions.

2.2 Why Categorize Keywords?

A typical profile on any OSN consists of numerous sections (e.g. Orkut [2] has Social, Professional and Personal sections; Facebook has Basic, Education & Work and Personal Information sections) that characterize the user. These sections are further sub-divided into various fields, e.g. the Personal section on Facebook [1] has Interests, Activities, Favorite Movies, Books, etc. as fields. We call the entries in these fields Keyword(s) as they represent user attributes. To understand the

keyword usage patterns we analyzed 1265 unique Facebook¹ user profiles [15]. Most of the fields contained proper nouns (e.g. movie names, albums, etc.) as entries, hence, for all evaluation purposes, we restricted ourselves to keywords found in the *Interests* (which contained words mostly from an English dictionary) field.

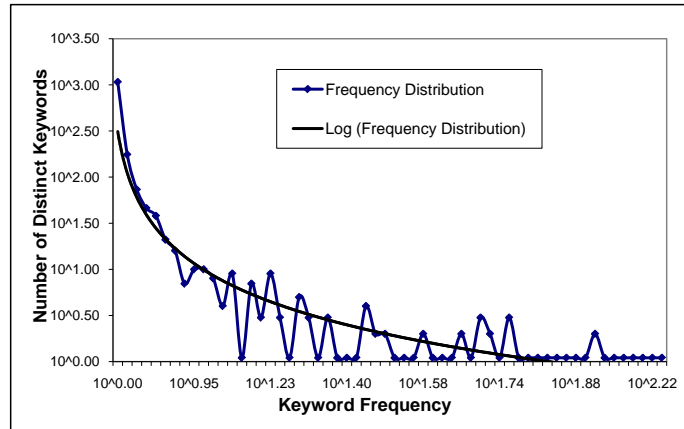


Figure 2.1: Number of Distinct Keywords vs Keyword Frequency on log-log scale

We looked at the magnitude of information given by users and how keywords can be processed to extract meaningful knowledge. On average, each user provided 5.8 keywords for the *Interests* field and the keyword set contained 1573 unique keywords. To analyze the distribution of keywords, we plotted the number of distinct keywords for a given keyword frequency on a log-log scale (see figure 2.1). We divided the keyword frequency into four categories to represent keywords with different frequencies (see figure 2.2).

The trend line (solid continuous line) for the graph in figure 2.1 shows an exponential drop in the number of distinct keywords as the keyword frequency increases. The distribution shows consistency with similar results on tag distribution over web applications [16]. It follows the Zipf's Law because the occurrence

¹We are thankful to Matthew Spear for providing us the Facebook data. Readers are directed to [15] for details.

frequency of a keyword increases as its popularity increases in the frequency list. Thus, we can infer that most of the keywords entered by users are distinct. Figure 2.2 also substantiates this observation as only 14% of the keywords belong to the high frequency category. This means that only a fraction of keywords are repeatedly used by different users and a large percentage of keywords (44% of the total) occur with very low frequency.

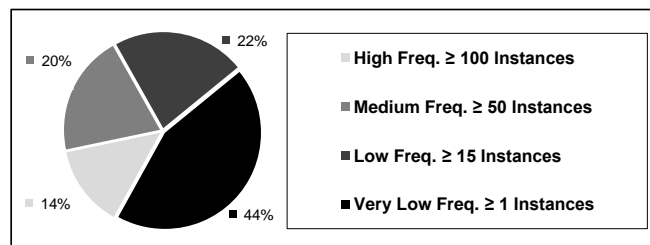


Figure 2.2: Keyword Frequency Distribution

Two important conclusions can be drawn using the above discussion. First, as the number of unique keywords is large there may be some relationship between these different keywords. This observation uses the fact that different topics in which users are interested can be generalized to a small number as there are limited ‘community’ categories in OSNs. The large number of unique keywords must also be related to these limited categories. Second, it is possible that the very low frequency keywords (which constitute almost half of the total) aren’t very dissimilar either with each other or to the other 56% of keywords due to the extensive usage scope of English words. Thus, to develop a social network model based on keywords, there is a need to explore the hidden relations among keywords and to categorize them. For instance, in Bob’s case, if the OSN could understand the relationship between ‘soccer’ and ‘football’ it might give better results for Bob’s query.

2.3 Social Network Modeling

In this section, we first describe a method of categorizing keywords in a data structure to utilize the underlying relationship amongst them. Then, we define functions to quantify the distance between keywords and similarity among social peers based on the distance between their keyword pairs. Finally, we talk about correlating social network topology with keywords using the similarity functions.

2.3.1 Forest Structure

We want a data structure that can help define distance between keywords by capturing hidden relations between them. It must employ methods to clearly distinguish between related and unrelated keywords. A single hierarchical structure (e.g. by a modification of [12]) will be insufficient as it will fail to capture important characteristics of keywords. First, it is not always possible to relate all the words, e.g. ‘earthquake’ and ‘soccer’, in a single structure. The distance between such unrelated words must come out to be relatively larger than that between related words. Second, the data structure must capture all meanings of a word as it can be used in different contexts (or in different syntactic categories). E.g., according to WordWeb [17], the word ‘stern’ could mean ‘severe’ as an adjective and ‘rear part of a ship’ as a noun. We propose a forest structure to store keywords where each tree in the forest contains related keywords. As a keyword could have more than one meaning it could occur in different trees. This way, we use multiple hierarchical trees (i.e. a forest) to measure distance between keywords.

Different methods could be used for arranging the keywords in the forest. A method based on etymological relations can be used to construct the forest. For

instance, in a language like English, which has been derived from Latin, Greek, etc., most of the words have a root associated with them. Wordinfo [18] lists 61,362 English words which have either Latin or Greek roots. A root word with its derived words can be put in a single tree. E.g. the words ‘equine’ (horse), ‘equestrian’ (horse rider), ‘equestrienne’ (female horse rider) and ‘equestrianism’ (horsemanship) that come from the Latin root ‘*equus*’ (a horse) can form one tree. All such trees taken together can form the forest.

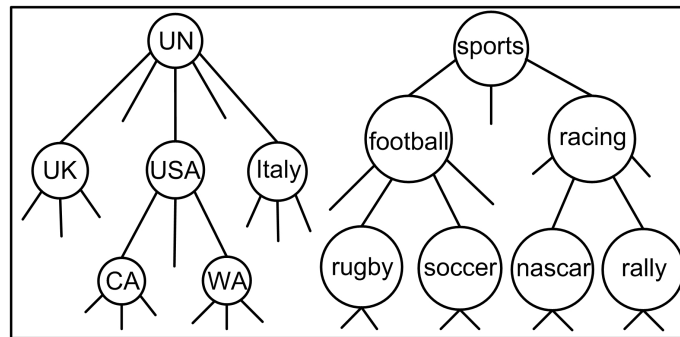


Figure 2.3: Forest with two component trees

Another way is to place semantically related keywords in the same tree. E.g. a tree can be made with the keywords related to ‘sports’. The next levels could contain various sports like ‘football’, ‘racing’, etc., each having its own sub-tree. Similarly, we can have another tree for all the countries under the keyword ‘United Nations’ (or ‘UN’) as shown in figure 2.3. Since, ‘soccer’ is a hyponym of ‘football’, Bob’s query will get better results as the semantic similarity between the two keywords has been captured by this structure.

2.3.2 Similarity Functions

Now we define the notion of distance between keywords based on the forest structure. Let there be t trees (T_1, T_2, \dots, T_t) in the forest F . Consider two keywords K_a and K_b such that both of them belong to the same tree. Let LCA be the least common ancestor of K_a and K_b . Also, assume $d(LCA, K_a)$ to be the depth of K_a from the LCA . E.g., in figure 2.3, if $K_a = soccer$ and $K_b = racing$ then $LCA = sports$ and $d(LCA, K_a) = 2$. Let us also define the maximum of the distances of K_a and K_b from the LCA as d_{LCA} i.e. $d_{LCA}(K_a, K_b) = \max(d(LCA, K_a), d(LCA, K_b))$.

Definition 1. *If K_1 and K_2 are two keywords, then the distance, $D(K_1, K_2)$, between them is given as:*

$$D(K_1, K_2) = \begin{cases} d_{LCA}(K_1, K_2) & \text{if } \exists T_i \text{ s.t. } K_1, K_2 \in T_i \\ \infty & \text{if no such } T_i \text{ exists} \end{cases} \quad (2.1)$$

If more than one such T_i exists, then the distance is set to the minimum of all the corresponding d_{LCA} 's.

If K_1 and K_2 don't have any relation then $D(K_1, K_2)$ is ∞ . Also, the minimum of all d_{LCA} 's is used to account for multiple occurrences of keywords in F . These observations justify the benefits of a forest structure (as explained in section 2.3.1) over a simple hierarchical model to store keywords. A possible metric to define the distance between two keywords could have been the sum of the depths of the keywords from their LCA (i.e. $D(K_1, K_2) = d(LCA, K_1) + d(LCA, K_2)$). But, we believe that to capture the distance between keywords from a generic common point (i.e. the LCA), $\max(d(LCA, K_1), d(LCA, K_2))$ is more appropriate as 'max' function gives the farthest distance from the generic point.

Now, we will define the similarity functions between social peers. Let the set of nodes (or social peers) in the social graph be V . Assume that a social peer w ($\in V$) has N_w keywords and let K_i^w ($1 \leq i \leq N_w$) be his/her keywords. Consider two peers u and v on the network. Let $k(u, v)$ ($N_u \times N_v$) be the total number of keyword pairs that they have. Also, let $n(u, v)$ be the number of keyword pairs (K_i^u, K_j^v) such that K_i^u and K_j^v belong to the same tree in F .

Definition 2. For two social peers u and v on the network, the ‘weak similarity’, $s(u, v)$, between them is:

$$s(u, v) = \frac{n(u, v)}{k(u, v)} \quad (2.2)$$

Definition 3. For two social peers u and v on the network, the ‘strong similarity’, $S(u, v)$, between them is:

$$S(u, v) = \frac{\sum_{1 \leq i \leq N_u, 1 \leq j \leq N_v} e^{-D(K_i^u, K_j^v)}}{k(u, v)} \quad (2.3)$$

We call the function s ‘weak similarity’, as it doesn’t take into account the position of keywords in a tree, i.e. keywords with distinct distance values will contribute equally towards the weak similarity. The function S is called ‘strong similarity’, as it also considers the relative positions of keywords in the forest as keywords with greater distance contribute less towards the similarity value. Exponential function was a natural choice for the definition because it has a finite value at the boundary conditions for $D(K_i^u, K_j^v)$ (as $e^{-0} = 1$ and $e^{-\infty} = 0$). The value of $S(u, v)$ decreases as the distance between the keywords increases implying that u and v share lesser interests or attributes. It may happen that

strong similarity is numerically smaller than the weak similarity but still it is a relatively more accurate definition as it captures more information. The similarity functions could provide good parameters for performing decentralized search (e.g. [10, 12, 19]) by finding similar friends in OSNs. They may also help in finding potential friends and tackling the link prediction problem (e.g. [20]) in social networks.

2.3.3 Social Graph

In this section, we formalize steps required to generate the social graph using the similarity functions. Assume that two people ‘A’ and ‘B’ share many common interests. It is likely that they know each other as they may have met somewhere because of their common interests. In our model, keywords exactly represent the interests or attributes of a social peer. Hence, if two people share many common keywords (a high value of similarity) then it is likely that they may have a link between them on the social graph. Formally, let the probability that a social peer u is willing to establish a friendship with another social peer v be proportional to $S(u, v)$. We divide $S(u, v)$ by $\sum_w S(u, w)$ (the normalizing constant) to obtain a probability distribution.

Definition 4. *The probability $p(u, v)$ that u is willing to create a friendship with v is given by:*

$$p(u, v) = \frac{S(u, v)}{\sum_w S(u, w)} \quad (2.4)$$

We used the ‘strong similarity’ to define $p(u, v)$ as it is a stronger indicator of similarity. Note that $p(u, v) \neq p(v, u)$ even though $S(u, v) = S(v, u)$ because

the denominators are different. This captures the fact that both peers might not be equally interested in having a friendship with each other. Let the vertex set V contain a vertex for each peer on the social network. Consider two random independent trials with probability $p(u, v)$ and $p(v, u)$ for the (u, v) peer pair. Join u and v with an edge if both trials yield a positive result i.e. both peers want to establish a friendship. Repeat the above process for all pairs of vertices to get the set of undirected edges E . Then, $G = (V, E)$ is the social network based on keyword similarity among peers. Next, we talk about the techniques that we used to evaluate the effectiveness of this social network model.

2.4 Evaluation Methodology

Now, we discuss the methods used to evaluate the above social network model. We considered two networks and compared the similarity values to observe the effectiveness of the ‘forest’ structure in correlating profile keywords with network topology (corresponding results are given in section 2.5). One network represented a realistic scenario (Facebook [1] data as mentioned in section 2.2) while the other was generated through simulation of our social network model.

2.4.1 Analyzing the Facebook Network

We used WordNet [21] as the database of English words to build the forest structure. It relates different words by using their sets of cognitive synsets. We used a Java API [22] to look at the meronyms, synonyms, holonyms, hypernyms, hyponyms, derived terms and set of similar words for a keyword. For a user u , we allowed each of its keywords to build its own tree with semantically similar

words. The lookup was continued recursively to a depth of three to get trees for all keywords. Thus, we built trees of words which were interlinked by concepts and meanings to form the forest structure (F_u) for a user. Defining a set of general keywords and expanding them to trees would have been a relatively more effective way of capturing the characteristics of keywords. As that would have required help from linguists, we adopted a rather simpler idea to construct the forest and thus, term our results as preliminary.

Once F_u was obtained, we checked if the keywords of other users belonged to the trees of F_u . If any such keyword was found (say for node v) then the value of $n(u, v)$ was incremented. This way we computed $n(u, v)$ values for all possible (u, v) pairs. The number of keyword pairs $k(u, v)$ is given by $N_u \times N_v$. Then, from $n(u, v)$ and $k(u, v)$, we computed the ‘weak similarity’ ($s(u, v)$) values between all pair of users. The Facebook dataset had more than 1.5 million user pairs of which 7827 pairs were direct friends.

2.4.2 Analyzing the Simulated Network

In this section, we discuss how we generated a social network graph and determined the ‘weak similarity’ among its users. The simulated graph had the same number of nodes (1265) as the Facebook data. We assumed a keyword set of 1500 distinct keywords. To mimic a realistic scenario, we used Zipf’s distribution to inflate the keyword set to a pool of 6400 keywords such that it contained multiple copies of some randomly chosen keywords. Each user was assigned 6 distinct keywords on an average from the inflated pool.

To generate the forest F , we sub-divided the pool of unique keywords into 185 trees. The trees had different numbers of keywords and varying depths to simulate

User	Interests
A	wakeboarding, softball, fishing, jesus, god, learning, backpacking
B	running, hiking, hurricanes, tornadoes
C	basketball, dancing, shopping, pictures
Z	running, soccer, tennis, foosball, hiking, knitting, art, tea, lime , pie

Table 2.1: Sample Users with Keywords

User	$k(Z, \text{User})$	$n(Z, \text{User})$	$s(Z, \text{User})$
A	70	4	0.057
B	40	5	0.125
C	40	8	0.200

Table 2.2: Weak Similarity with respect to Z

real semantic relations of words. Forest F allowed us to compute ‘strong similarity’ values, $S(u, v)$ for users u and v , across different user pairs. Once $S(u, v)$ values were known for all user pairs, the probability $p(u, v)$ (as defined in Equation 2.4) was determined. When $p(u, v)$ and $p(v, u)$ both were above a minimum threshold value (θ), an undirected edge (i.e. friendship) was established between nodes u and v . Since $p(u, v) \neq p(v, u)$, an edge was established only when both peers satisfied the minimum interest level, i.e. when $p(u, v) \geq \theta$ and $p(v, u) \geq \theta$. In this way, we got the social network graph, $G = (V, E)$, according to the model presented in section 2.3.3. Once the social graph G and the forest F were known, we computed the ‘weak similarity’ values across all pairs of nodes.

2.5 Results and Discussion

In this section, we present the results obtained from the analysis of the real and simulated networks. First, let us look at four Facebook users with their keywords,

for the *Interests* field (table 2.1). All the users are interested in some type of sporting events. We compare the results of the ‘weak similarity’ of the first three users with the user Z in table 2.2. It can be seen that $s(Z, C)$ is maximum even though Z and C have fewer keyword pairs ($k(Z, C) = 40$). This is because their profiles match for keywords which can be derived from ‘athletic sports’ (e.g. pairs formed from basketball, dancing, running, soccer, tennis, etc.). Also both are interested in arts (C has ‘pictures’ and Z has ‘art’) implying that Z has more common interests with C than with A or B . A and Z are least similar as A is mostly interested in water sports (and not athletic sports as Z) and doesn’t share any other common interest with Z even though they both have a large number of keyword pairs. This shows the effectiveness of characterizing keywords using semantic relationships and that the content of keywords becomes more important than their number for finding similarity values.

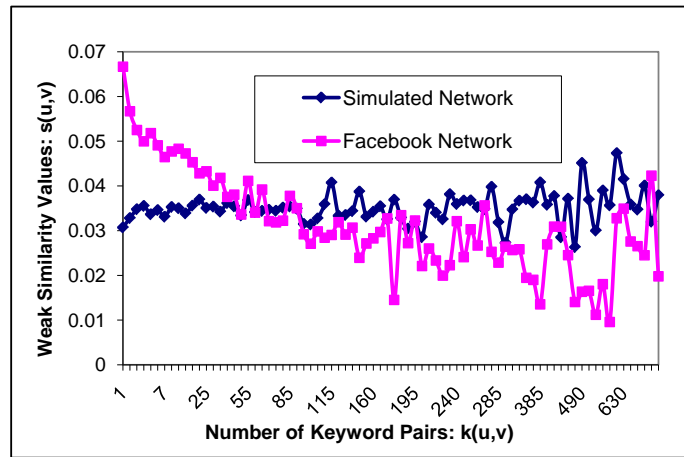


Figure 2.4: $s(u,v)$ vs $k(u,v)$: All User Pairs

Figure 2.4 shows the variation of weak similarity, $s(u, v)$, with the number of keyword pairs, $k(u, v)$, for the real and simulated networks across all user pairs. The pattern followed by curves of both the graphs seems to be similar in nature,

suggesting that our social network model is successful in mimicking behavior of the realistic data. Thus, we can say that our model seems to be an effective way to analyze social network topology. A more rigorous analysis using strong similarity would be desirable but we don't do that as the forest structure has not been prepared yet which will require help from linguists.

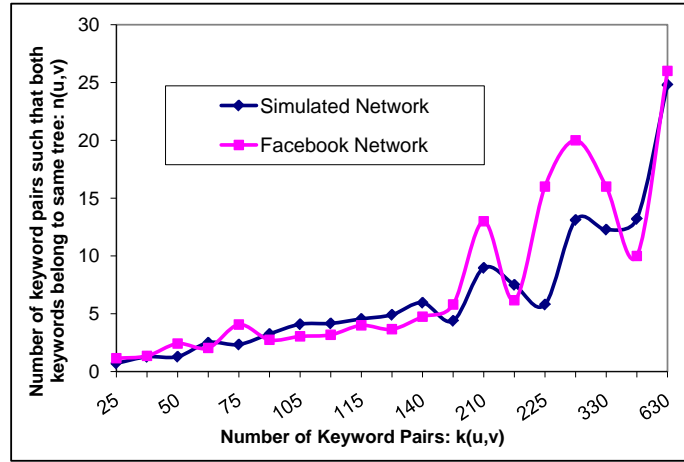


Figure 2.5: $n(u,v)$ vs $k(u,v)$: Direct Friends

The variation of the number of keyword pairs belonging to the same tree, $n(u,v)$, with different number of keyword pairs, $k(u,v)$, for user pairs that are direct friends is given in figure 2.5. The curve shows that $n(u,v)$ increases proportionally with $k(u,v)$. This indicates that the weak similarity between user pairs is independent of the number of keyword pairs. From here, we can conclude that no user can successfully alter the similarity value with other users by inflating his profile with unnecessary information. Thus, similarity functions can provide good metrics, which are immune to irrational user activity, to model friendship behavior.

2.6 Conclusion and Future Work

In this chapter, we studied the importance of categorizing keywords and defined a ‘forest’ structure to quantify the similarity between seemingly unrelated user profile information available on OSNs. Based on the similarity functions, we formalized a model of social network topology. We evaluated the effectiveness of our model by simulating and comparing it with a realistic dataset and preliminary results show that our model faithfully emulates the behavior shown by the real OSN. This led us to conclude that the use of keywords is an effective way of modeling and analyzing OSNs. Our model also provides good metrics, immune to irrational user activity, to model friendship behavior.

In future, we would like to explore better methods, based on machine learning techniques, to construct the forest structure. The forest structure may have to be iteratively updated if one could empirically find potential relations between keywords in different trees using statistical methods. We also intend to gather more real OSN data and analyze that data to form a deeper understanding of the correlation between profile keywords and the social network topology. Also, we would like to study the variation of similarity between user profiles at different topological distances and over different times to address the link prediction problem in OSNs.

Chapter 3

Information Flow and Search in Unstructured Keyword based Social Networks

3.1 Introduction

In online social networks (OSNs), user connections can be represented as a graph. The network formed has distinct properties that distinguish it from other graph topologies. E.g., it has high average node degree, high value of clustering and displays small world properties [23, 24]. For accurate results, these properties need to be considered during analysis of OSNs. For instance, when a user searches for other users (or user characteristics), the network's search scheme may have to search and sort through a large set of results even at small topological distances. Additionally, users consider search results relevant based on their position in the network rather than globally relevant results.

In this work, we consider an unstructured keyword based social network topology where a link between two nodes represents friendship between them. Each edge has a trust value associated with it to represent the mutual relationship between the corresponding nodes. Users have keywords (profile attributes) as their characteristics which have policies associated with them to define abstractly the flow of keyword information and the accessibility to other users in the network. Such keyword based social networks, which try to mimic human behavior, could potentially be used for internet routing [5, 6]. For instance, assume that Bob has an interest in soccer. He would like to establish a contact with someone who is also interested in soccer. His natural tendency would be to check if any of his direct friends plays soccer. Even amongst them, he would prefer those friends with whom he shares a closer (more trustworthy) relationship. Then, if he wants to find more soccer playing people, he would go to his friends of friends, etc. Keyword based social networks try to incorporate this real life human social behavior in OSNs through the use of keywords by allowing communication only through the available trustworthy social paths.

This chapter makes two key contributions. First, we develop an information flow model to disseminate keyword information when users add keywords as their profile attributes. To address privacy concerns as outlined in works on future OSN architectures [25, 5, 6, 26], we restrict the identity of users during information flow to only their direct friends. Beyond that, even though information about keywords is allowed to flow, the identity of users is not propagated. Second, we design and develop a search algorithm based on keyword information. The search problem is broadly defined as the scenario when a user queries with a set of keywords to contact other users (termed as *targets*) who have all those keywords as their profile

attributes. In real life, as with Bob, people first seek information from their friends, then friends of friends and so on. Also, people tend to inherently grade their friends based on the mutual relationship that they share with each other. Keeping this in mind, we aim at finding a subset of these targets which are topologically closer to the querying node and have high trust on the edges connecting them to the querying node. We believe that our contributions can be applied to other network applications such as peer to peer networks or mobile social networks that may show graph properties similar to an unstructured social network.

When knowledge about users and their keyword information is available to each individual node, aided by a centralized system (like Facebook, Orkut, etc.), then the search problem reduces to sorting through a list of targets who match the set of keywords in the query and construct the good result set. Here, we broaden the scope to consider future OSN architectures [25, 5] with a decentralized architecture where no node has access to complete information about all users. For these decentralized settings, searching for targets becomes a challenging problem. Further, the unstructured nature of the network increases the difficulty. Simplest would be to broadcast the network with queries till all possible targets are found, but this is inefficient and unscalable. Other decentralized search techniques based on either breadth first search or random walks are also not good candidates as they don't utilize the level of information available, in the form of user keywords and their policies, in a keyword based social network. The search algorithm we present utilizes the information available in the keyword based social network as it looks for targets. It uses a linear combination of two primary metrics: a distance metric to find topologically closer targets and, a trust metric to find paths with high trust values between the querying node and the target. It is also dynamic in nature such

that it adapts itself for each individual node during the search process.

We describe the unstructured keyword based social network model in section 3.2. Section 3.3 presents the proposed model for information flow in the social network. Here, we also discuss how to maintain recent information and address the privacy concerns of users. Section 3.4 describes the search algorithm and how it adapts itself for different nodes. Section 3.5 deals with the setup used for evaluation and results are presented in section 3.6. Section 3.7 covers the related work in this field. Finally, we conclude in section 3.8 with possible extensions.

3.2 Background

Here, we focus on unstructured keyword based social networks where the user topology is represented by a set of nodes and edges. Let the social network be an undirected graph $G = (V, E)$, where V represents the set of nodes in the social network and E is the set of edges between the vertices in V . A link exists between nodes i and j if both the nodes want to be friends with each other. Each edge has a trust value associated with it to represent the mutual relationship between the corresponding nodes. The concept of trust can be generalized to include different models, that characterize different friendship levels [27], such as the frequency or quality of interactions, privacy settings during information sharing, etc. The value of trust lies between 0.0 and 1.0 with a higher value representing more trust.

In a keyword based social network, users are characterized by keywords [6, 7]. A node v declares its *Profile Attributes* (K_v^{PAtt}) which is a set of keywords and each keyword has a policy associated with it [5]. Each node also has another set of attributes called the *Friendly Attributes*, K_v^{FAtt} , that is the set of keywords that

v receives from its friends. The Profile Attributes represent the information that the user wants other users to know about itself whereas Friendly Attributes is the information that the user receives from other users. The policy definition in [5] uses a broader definition based on subjective parameters that is beyond the scope of this work. Here, we consider the policy for a (profile attribute) k of v as a tuple:

$$\forall k \in K_v^{PAtt}, \exists Policy(k) = [D, T]$$

where, D is the maximum distance (in hops) of nodes from whom v is willing to get contacted with k as the keyword and, T is the minimum trust that each link must have from the query-issuer to v on the social path.

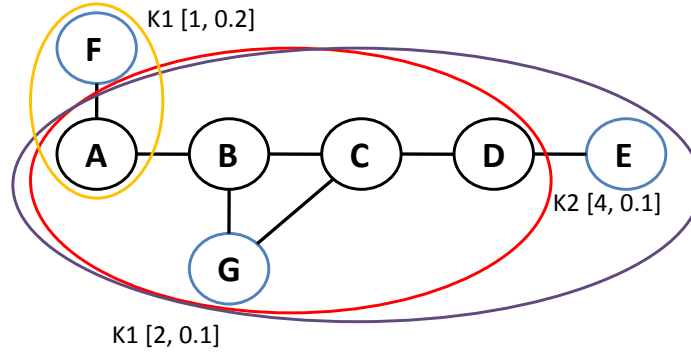


Figure 3.1: Example of the community based social network

Figure 3.1 shows an example graph for the given framework. Nodes F and G , each sets keyword $K1$ and E sets $K2$ as their profile attribute with the policies shown. For simplicity, assume that all links have a trust value higher than 0.5. The ovals show the nodes to which the keywords will be propagated depending on their policy during the initial information flow phase. A user u can issue a query with keywords Q_k from K_u^{FAtt} which will enable him to establish contact with other users who have all the keywords in Q_k as their profile attributes. The set of

potential targets for a querying node u is defined as: $Targets_{Potential} = \{v : v \in V$ where $\forall k \in Q_k, k \in K_v^{PAtt}$ && u satisfies $Policy(k)$ set by $v\}$. E.g., when node A issues a query $Q_k = \{K1\}$, $Targets_{Potential} = \{F, G\}$ and if node B issues the same query then $Targets_{Potential} = \{G\}$. Returning to Bob's example from the introduction, he can issue a query with $Q_k = \{soccer\}$ assuming $soccer \in K_{Bob}^{FAtt}$. Targets would be all users who have 'soccer' as their profile attribute such that Bob satisfies the policies set by them. Bob can reach the targets through those of his direct friends who have 'soccer' as their friendly attribute implying that there exists a social path through them. Bob can also issue more specific queries. E.g., $Q_k = \{soccer, Davis CA\}$ if he wants to find soccer playing users in the city of Davis in California.

To find all the potential targets a search algorithm may have to inspect a huge number of nodes, due to the high node degree and clustering coefficient, making the search process impracticable. Thus, it would suffice to find good targets from $Targets_{Potential}$ where a target is defined as good when the path connecting it to the querying node is small in length and has high trust on the edges. The definition of a good target has been kept at an abstract level since its tough to quantify goodness owing to its subjective nature from a querying node's point of view. Results and targets describe the same concept and have been used interchangeably. In the next sections, we develop an information flow model and a search algorithm to find this subset of targets.

3.3 Information Flow Model

Once a user joins the network and adds keywords as his Profile Attribute(s), information needs to be propagated in the network so that other users can search and contact him. The information flow model primarily needs to satisfy three conditions while spreading the profile attributes. First, it must propagate the information properly using minimal resources. Second, it must address the privacy concerns of users [26]. Finally, it must ensure that nodes maintain the most recent information i.e. information about changes in friendship(s) or keyword policies must be propagated to nodes quickly.

3.3.1 Keyword Propagation Process

When a user v adds a keyword K in its Profile Attributes with policy $P(K) = [D, T]$, the keyword has to be propagated to other nodes. Other nodes receive this keyword and store it as a Friendly Attribute. Information is spread out in the form of propagation messages where the structure of a propagation message is as follows: $\langle PID, Keyword, Hops_{remaining}, Hops_{covered}, Trust \rangle$, where PID represents the propagation identity, $Keyword$ is the keyword that is being propagated, $Hops_{remaining}$ is the number of hops that still need to be covered, $Hops_{covered}$ is the number of hops that have been covered so far and $Trust$ is the trust that was set in the policy by the node who added the keyword. PID is a randomly generated large binary string and we assume that the number of bits in PID are sufficiently large so that the probability of two different nodes generating the same PID for the same keyword is negligible. The motivation to use PID is to avoid searching or propagating around cycles in the graph which could be numerous owing to the

high clustering coefficient. *PID* makes sure that a propagation message does not go around a cycle from both directions and thereby avoids unnecessary messages.

Keyword	Propagation Data	Friends
K_1	$\{PID_1, Hr_1, Hc_1, T_1\}$	F_1, F_2, F_3
	$\{PID_2, Hr_2, Hc_2, T_2\}$	F_2, F_4
K_2	$\{PID_6, Hr_6, Hc_6, T_6\}$	F_1, F_3, F_4

Table 3.1: Keyword Forwarding Table

Each node $w \in V$ maintains a Keyword Forwarding Table, FT_w , to store which keywords (along with their propagation data: Hr represents $Hops_{remaining}$, Hc represents $Hops_{covered}$ and T is the *Trust*) it has propagated to which of its direct friends so far. Table 3.1 shows an example of the keyword forwarding table. It shows that keyword K_1 was propagated to four direct friends by w and it was propagated to F_2 with two different policies (see below for explanation).

When a node gets the same keyword to propagate with different propagation data, it needs to decide if it should forward (and to which of its friends) the keyword with the newer policy because if the new data has already been covered (or included) in earlier propagation data then sending the new propagation message is unnecessary. We say that propagation data 1 is non-inclusive of propagation data 2 if

$$((\Delta Hr > 0) \parallel (\Delta T < 0) \parallel (\Delta Hc < 0))$$

where, $\Delta x = x_2 - x_1$. If the new propagation data says that more hops are remaining or the new trust is lower, then the keyword needs to be propagated again. If the new message has covered fewer hops then the keyword also needs to be propagated to store correct search information (section 3.4.1).

Consider the scenario when u receives a propagation message from v . It will

Algorithm 1: Processing Propagation Message

Input: v sends prop. message $\langle pid, K, hr, hc, T \rangle$ to u

```

1 if  $((K \in FT_u) \ \&\& \ (pid \in FT_u))$  then
2   if new prop. data non-inclusive w.r.t stored prop. data corresponding to
   pid then
3     update  $\{pid, hr, hc, T\}$  in  $FT_u$ ;
4     update search information;
5     foreach (friend  $z (\neq v)$  of  $u$ ) do
6       if  $((hr > 1) \ \&\& \ (Trust_{uz} > T))$  then
7         send  $\langle pid, K, hr - 1, hc + 1, T \rangle$  to  $z$ ;
8         add/update  $z$  (for  $pid$ ) in  $FT_u$ ;
9     else drop message;
10 else if  $((k \in FT_u) \ \&\& \ (pid \notin FT_u))$  then
11   foreach (friend  $z (\neq v)$  of  $u$ ) do
12     if new prop. data non-inclusive w.r.t all stored prop. data
     corresponding to  $z$  then
13       add  $\{pid, hr, hc, T\}$  and  $z$  to  $FT_u$ ;
14       update search information;
15       if  $((hr > 1) \ \&\& \ (Trust_{uz} > T))$  then
16         send  $\langle pid, K, hr - 1, hc + 1, T \rangle$  to  $z$ ;
17     else drop message;
18 else
19   add  $K$  and  $\{pid, hr, hc, T\}$  to  $FT_u$ ;
20   update search information;
21   foreach (friend  $z (\neq v)$  of  $u$ ) do
22     if  $((hr > 1) \ \&\& \ (Trust_{uz} > T))$  then
23       send  $\langle pid, K, hr - 1, hc + 1, T \rangle$  to  $z$ ;
24       add  $z$  (for  $pid$ ) in  $FT_u$ ;

```

process the message using Algorithm 1. The main step is to see if the new propagation data has already been included for each friend or not and then forward the message accordingly. The search information that needs to be updated (steps 4, 14 and 20) will be discussed later (section 3.4.1). To initiate the propagation of keyword K with policy $P(K)$ ($= [D, T]$), a node generates the propagation message

$\langle pid, K, D, 0, T \rangle$ and processes it according to Algorithm 1.

3.3.2 Identity Privacy Issues

To respect privacy requirements, any social network model and corresponding applications must be designed so that user privacy is protected as information flows within the graph. Our design of message propagation removes the identity of users as their information flows deeper in the graph, i.e. at distances further away from their direct friends. The keywords, propagation data and *PID*'s are stored but they are insufficient to reveal any relevant information about the people who are not direct friends of the user. As the identity of only the direct friends is stored, the network has the capability of supporting both anonymous and identified messages (depending on the application) without compromising the privacy of users making the message propagation model more general and suitable for wider use.

3.3.3 Information Maintenance

As friendships change, the topological structure of the graph changes. Policies of individual keywords may also be changed by users. Such updates should also be reflected in the information that is stored by users. Thus, the information flow process must notify nodes about such updates so that each user has access to the latest information. To account for such scenarios, we introduce the concept of timely updates for propagated keywords that have flowed into the network. Each node will send beacons (which could be the $(keyword, PID)$ pair) after a certain δt time to tell other nodes that the corresponding keyword still exists in the network. Thus, nodes need to store only that data which was received within the last δt time.

If a node wants to change the policy it can generate a new *PID* and propagate the keyword again. The data stored for the earlier *PID* will be removed by other nodes as they will not get the beacon anymore. Nodes can use the forwarding table to forward beacons to other nodes. This process not only takes care of updated link topology but also makes sure that nodes have the most recent information about friendly attributes and their policies. The numerical value of δt will depend on the underlying physical network and the application and hence, is beyond the scope of this work.

3.4 Search Algorithm

In this section, we design and develop an algorithm to search in the keyword based social network. The search algorithm is composed of three key components. First, it uses a linear combination of a distance metric and a trust metric to define the ‘value’ of a direct friend. Second, a threshold function that helps in dynamically pruning the network. The last component is the query message processing algorithm.

3.4.1 Selecting Topologically Closer Nodes

In real life, people seek information first from their friends, then friends of friends and so on. We model this scenario by giving precedence to targets that are topologically closer to the querying node. In order to decide which of the direct friends of a node will be suitable to route queries in terms of distance, the node stores information about the distance traveled by received keywords to make an intelligent decision.

Keyword	Max. Hops	(Friend, Min. Hops)
K_1	$Hmax^{K_1}$	$(F_9, Hmin_{F_9}^{K_1}), (F_5, Hmin_{F_5}^{K_1})$
K_2	$Hmax^{K_2}$	$(F_2, Hmin_{F_2}^{K_2}), (F_6, Hmin_{F_6}^{K_2})$

Table 3.2: Keyword Received Table

Each node w stores a Keyword Received Table, RT_w , in which entries are indexed by keywords. For each keyword k , we define $Hmax^k$ as the farthest distance covered by k as it traveled from nodes (who added k as a profile attribute) to w . This value is updated whenever a new propagation message for keyword k comes by comparing the stored $Hmax^k$ with the $Hops_{covered}$ field of the message and updating $Hmax^k$ with the greater value. Also, the node stores which of its direct friends forwarded the keyword to it. For each such friend, say F , the hops of the closest target ($Hmin_F^k$) reachable through F with keyword k is also stored. $Hmin_F^k$ can be updated during the propagation process (steps 4, 14 and 20 in algorithm 1); if the $Hops_{covered}$ for the propagation message from F with keyword k is less than $Hmin_F^k$ then the value is updated. E.g., table 3.2 shows that w received two keywords (K_1 and K_2) from its direct friends during the propagation process and it stores the corresponding $Hmax^{K_1}$ and $Hmax^{K_2}$ for each keyword. It also stores $Hmin_F^K$ values for each keyword for all direct friends who propagated that keyword. As with the Forwarding Table only the information from the previous δt time is stored in the Keyword Received Table.

To differentiate between direct friends of a node w , we define the distance value as follows:

Definition 5. *The distance value for a direct friend u of w who forwarded all the*

keywords in a keyword set S_k to w is:

$$DV(u, S_k) = \frac{\min_{k \in S_k} Hmax^k - \max_{k \in S_k} Hmin_u^k}{\min_{k \in S_k} Hmax^k} \quad (3.1)$$

For a set of keywords the maximum of $Hmin$'s needs to be taken because no node before that distance exists which propagated all the keywords in S_k . Similarly, minimum of $Hmax$ is taken for a set of keywords as no node after that distance can exist which propagated all the keywords. A higher distance value for a friend tells that there is a better chance of finding a target close by through this particular friend. If paths of different lengths exist to the same target then the distance value will be higher for those direct friends through whom shorter paths exist. The distance value has been normalized w.r.t to the farthest target and it lies between 0 and 1.

3.4.2 Selecting Trustworthy Nodes

The next step towards finding a set of good results is to probe edges with high trust value between the connecting nodes. To achieve this we define the scaled trust function.

Definition 6. *The trust value for a keyword set S_k for a direct friend u (which forwarded all the keywords in S_k) of w is:*

$$TV(u, S_k) = \frac{T_{wu}}{T_{max}^{S_k}} \quad (3.2)$$

where T_{wu} is the trust of the (w, u) link and $T_{max}^{S_k}$ is the maximum trust amongst all friends of w from whom it received all the keywords in the set S_k .

The scaled trust defined above orders all those direct friends which forwarded all the keywords in S_k to w . This naturally helps w to find targets through those friends with whom he/she shares higher values of trust. We have used scaled trust to normalize the value. Next, using the two parameters defined so far, we discuss how a node can query the network for targets.

3.4.3 Querying with Keywords

Now, we describe the querying process that a user uses to search for targets. Nodes consider two important factors as they route query messages through the network: a) Value of Friends b) Threshold Function. The first parameter helps a node to determine the value of direct friends while the second parameter dynamically sets threshold values so as to reduce the number of edges the algorithm needs to inspect.

Definition 7. *The value for a keyword set S_k for a direct friend u (which forwarded all the keywords in S_k) of w is:*

$$V(u, S_k) = \rho \times DV(u, S_k) + (1 - \rho) \times TV(u, S_k) \quad (3.3)$$

where ρ lies between 0 and 1.

The parameter ρ decides the relative weight that is given to scaled distance value and to scaled trust value. Thus, if some application requires those paths which are very close to the querying node then it can set ρ close to 1 and if more precedence is given to trustworthy paths then ρ can be set close to 0. The structure of a query message is as follows: $\langle QID, Q_k, T_{min}, Hops_{done}, Hops_{left} \rangle$ where QID is a randomly generated binary string, Q_k is the set of keywords used for querying, T_{min} is the minimum trust on the path that has been traversed by the

query message so far, $Hops_{done}$ is the number of hops that the query message has traversed and $Hops_{left}$ is the number of hops left for the message to travel. QID is the identity of the query and helps in distinguishing between different queries. Each node u also stores the QID 's (and the corresponding direct friend who forwarded that QID) that it has processed in a table (QID_u) to avoid processing the same query multiple times.

Now, when a node processes a query with a particular keyword set, it needs to decide to which of its direct friends it should send the query to. We propose that the query should be sent to only those friends whose 'value' is above a threshold value (Θ). The threshold value is crucial to the behavior of the search strategy as it helps decide how aggressively the searched network is pruned. As the degree of nodes could vary drastically, a system wide constant value would prove insufficient to meet the requirements of the query algorithm. Hence, we model the threshold value to be a function of degree rather than a constant value. Consider a peer u on the social network and let us assume that $N_u^{Q_k}$ friends of u forwarded the keywords in Q_k to u . A static value of threshold may be significantly higher or lower than the 'value' of majority of friends in $N_u^{Q_k}$. This may lead either to drastic pruning or generation of excessive query messages, ultimately degenerating the search process. The threshold value should dynamically adjust itself according to the 'value' of the friends of u in $N_u^{Q_k}$. This will make the algorithm adaptive where each node calculates the threshold depending on its degree and the characteristics of its friends. Thus, we propose the threshold to be dependent on $N_u^{Q_k}$ for the

node u .

$$\Theta(u, Q_k) = \max_{w \in N_u^{Q_k}} V(w, Q_k) - f(N_u^{Q_k}) \times \left(\max_{w \in N_u^{Q_k}} V(w, Q_k) - \min_{w \in N_u^{Q_k}} V(w, Q_k) \right) \quad (3.4)$$

where the function f should be such that $f(1) = 1$, $\lim_{N_u^{Q_k} \rightarrow \infty} f(N_u^{Q_k}) = 0$ and it should be a monotonically decreasing function. This ensures that as $N_u^{Q_k}$ increases the threshold approaches the maximum ‘value’ reducing the number of friends to which the query needs to be sent. The function f decides the extent to which the search tree is pruned and hence, we call it the *Pruning Function*. The whole class of functions $g(x) = x^{-p}$ for $p \geq 0$ satisfy the conditions of the pruning function. When $p = 0$, $f(N_u^{Q_k})$ will always set the threshold value to the least possible ‘value’ giving the breadth first search (BFS) mechanism. As p increases the threshold value reaches the maximum ‘value’ at a faster pace, pruning the search tree more drastically. Thus, by varying p we get many pruning functions with varying slopes.

Algorithm 2: Processing Query Message

Input: v sends query message $\langle qid, Q_k, T_m, H_d, H_l \rangle$ to u

- 1 **if** ($qid \in QID_u$) **then** drop message;
- 2 **else**
- 3 add qid to QID_u ;
- 4 **if** ($(Q_k \subseteq K_u^{PAtt}) \ \&\& \ (\forall k \in Q_k, [H_d + 1, T_m] \text{ satisfies Policy}(k)_u)$) **then**
 send success message to v ; /* u is a target */
- 5 **if** ($H_l > 0$) **then**
- 6 **foreach** (friend $z (\neq v)$ of u such that z sent the keywords in Q_k) **do**
- 7 **if** ($Trust_{uz} < T_m$) **then** $T_m \leftarrow Trust_{uz}$;
- 8 **if** ($V(z, Q_k) \geq \Theta(u, Q_k)$) **then**
- 9 send $\langle qid, Q_k, T_m, H_d + 1, H_l - 1 \rangle$ to z ;

When a node u receives a query message from node v , it determines the ‘value’ of each direct friend, the threshold value and then processes the query message using Algorithm 2. Once a target is found, a success message is generated which goes back to the query-issuer (user QID table, QID_u , helps in deciphering the reverse path) and this completes the search process. To initiate a query, the querying node sets QID to a new qid , Q_k to the set of query keywords, T_{min} to 1 (maximum allowed value), $Hops_{done}$ to 0 and $Hops_{left}$ to H_l (maximum hops to which it wants to search) generating the query message $\langle qid, Q_k, 1, 0, H_l \rangle$. By checking the ‘value’ of a node with a threshold and sending the message to only those friends which are above the threshold, we achieve the following benefits:

- This reduces the number of messages sent in the network. Aggressively pruning the search tree and reducing the load on the network helps in making the unstructured social network more scalable.
- The ‘value’ function chooses those targets that are topologically closer and uses links which have high trust returning good results amongst all obtainable results.

Analytical methods to analyze such designs are difficult to develop due to the lack of structure in the social network and the presence of multiple parameters in the algorithm. Thus, we use simulations to evaluate our algorithm.

3.5 Simulation Methodology

The evaluation methodology consists of four steps: generation of graphs with properties of social networks (high average values of node degree and clustering) as well

as small world properties (low diameter), distribution of trust among the edges, assignment of keywords to user nodes with corresponding policies and their propagation and finally, issuing queries from a set of nodes to see how the algorithm performs. We assumed that the time taken by the graph topology to change or for a user to update keywords and/or its policies is much greater than the time taken to search for a query. Thus, for simulation we used a static graph environment.

3.5.1 Graph Generation

In this work, we used a graph of one thousand nodes using the model given in [28] with properties observed during the measurement of existing online social networks [23, 24]. The properties of the generated graph are: average node degree = 18.63, diameter = 5 and clustering coefficient = 0.399. The small value of the diameter, high average node degree and high clustering coefficient emulate the properties possessed by realistic OSN graphs. Due to time considerations, we evaluated a graph of one thousand nodes only as a preliminary evaluation step at this stage.

3.5.2 Trust Distribution

The next step was to model the trust distribution among the edges. The classification of trust values and its distribution have been taken from [27]. The trust values are broadly classified into five categories: ‘Blind Trust’, ‘High Trust’, ‘Medium Trust’, ‘Low Trust’ and ‘Don’t Know’. We assigned a value of 0.9, 0.7, 0.5, 0.3 and 0.1 to each category respectively. The highest value of trust (‘Blind Trust’ = 0.9) signifies high trust among the nodes while the lowest value (‘Don’t Know’ = 0.1) implies the presence of negligible trust. The edges were picked randomly to assign

the trust values. The number of edges assigned to each category was based on the survey results presented in [27] where the final values of distribution are given as 19.28%, 12.27%, 34.35%, 25.95% and 8.15% respectively.

3.5.3 Information Propagation: Keywords and Policies

We set our information propagation process into two environments by varying the depth in the policy to see how the search algorithm performs in each of the environments.

- **Restrictive Policy:** In this case, all users uniformly set the policy associated with each keyword. When a keyword is added by a user, the maximum depth value (D) is set to 2, i.e. a keyword can travel from its originator to a maximum of two hops. The choice of two hops for restrictive policy is significant as it represents the ‘friend of a friend’ radius. This policy represents the case when users are very restrictive about their information being propagated in the network.
- **Liberal Policy:** When users are allowed to set any policy for keywords. Here, when a keyword is added by a user, a random integer between zero and diameter of the graph is set as the depth value in the policy. This helps us to observe the performance of the algorithm in a more realistic situation where some people are private in nature and concerned about who gets to access any information about them (and thus, set the depth value to 1 where only direct friends can access the information) whereas other people let their information be accessible from many hops away in the network.

We next assigned to each keyword the weighted average of the trust values (that came out to be 0.4828) of the edges, as its policy, so that it uniformly propagates throughout the network. We randomly selected 100 nodes from the graph and initiated the process of keyword addition to these nodes and then propagated those keywords using Algorithm 1.

3.5.4 Query Distribution

Since, the search algorithm dynamically sets the value of the threshold by looking at the node degree and by applying the pruning function, query nodes must be selected so they are representative of the various node degrees in the network. Thus, we sorted the nodes according to their node degree and picked 100 nodes with their degrees ranging from the minimum node degree (14) to the maximum node degree (26) of the generated graph. For each of the query nodes, we started the search algorithm and analyzed its behavior.

3.6 Results

We analyzed the performance of the algorithm by comparing the number of query messages generated and the number of targets found for three different pruning functions at different values of ρ and hops with the corresponding values for BFS. The pruning functions were obtained by setting p to 0.5, 1 and 2 such that $f(N_u^{Q_k}) = 1/\sqrt{N_u^{Q_k}}$, $1/N_u^{Q_k}$ and $1/(N_u^{Q_k})^2$ respectively. Different values of p were used to achieve different levels of pruning with $p = 0.5$ giving minimum pruning and $p = 2$ giving maximum pruning. We modeled BFS by setting $p = 0$ in the pruning function. Results have been normalized and averaged over 100 queries.

3.6.1 Performance at various ρ values

We varied ρ between 0.00 and 1.00 to obtain different linear combinations of distance and trust metrics. Figures 3.2 and 3.3 show the results for the networks with restrictive policy and liberal policy respectively. Figure 3.2a shows that the number of query messages sent during BFS is ~ 121 while in the worst case evaluation of all pruning functions, the number of query messages is ~ 12 , showing a significant reduction in the number of query messages generated. Similar observations can also be made from figure 3.3a as the number of messages sent per query drops by orders of magnitude as compared to BFS. Here, the number of query messages is relatively larger as compared to figure 3.2a, because due to liberal policy, keyword(s) travel longer distances in the network.

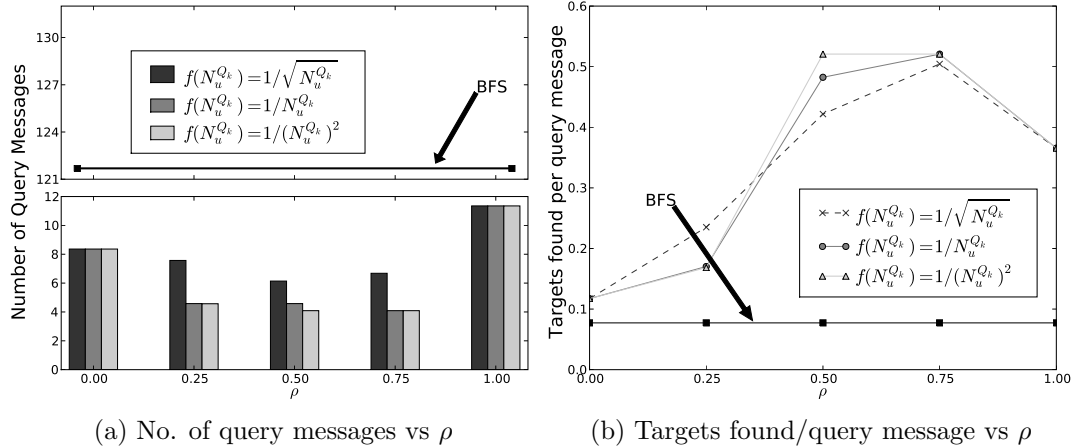


Figure 3.2: Network with restrictive propagation policy

Figures 3.2b and 3.3b show the targets found per query message for both the networks. The peak values observed for pruning functions are 0.5 and 0.135, whereas the corresponding values for BFS are 0.1 and 0.005 respectively. This means that our algorithm is much better at converting a generated query message

to a successful probing message.

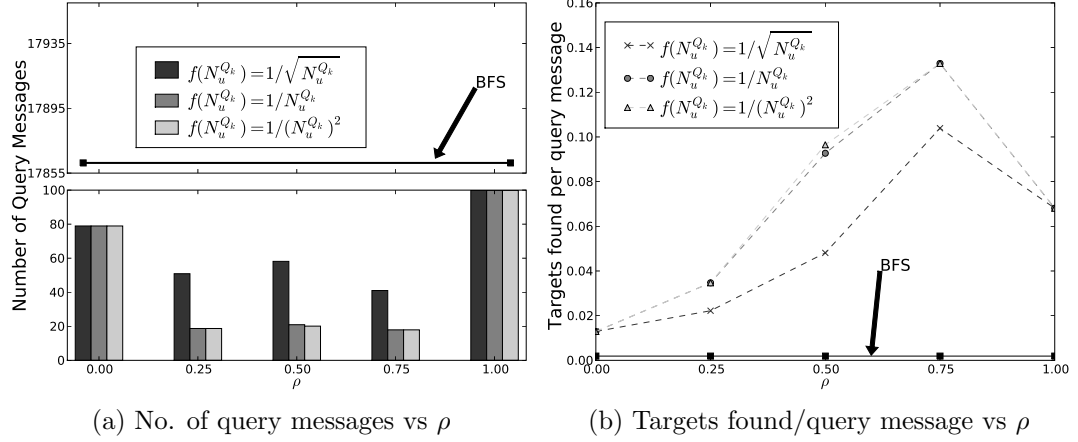


Figure 3.3: Network with liberal propagation policy

We also observe that for extreme values of ρ i.e. when distance or trust are considered separately, there is a sharp dip in the performance. For values of ρ between 0.50 and 0.75, the performance attains the peak level showing that a balanced combination of distance and trust metrics is best. We skip mentioning the absolute number of targets found as that can be determined from sub-figures of figures 3.2 and 3.3.

3.6.2 Performance at various hop values

All results in this section are given for targets found within different hops. We start by explaining the expected behavior of the algorithm. Let's assume that there are t potential targets when we probe the network till first hop. BFS should report each of these as it does not differentiate between their 'values' (definition 7). Since, the search algorithm differentiates between the values and adjusts the threshold to prune, it should return only a subset of these targets. This differentiation is

based on trust of the edges as distance metric is the same for all of them and contributes equally towards the ‘value’. Thus, the final subset should contain a fraction of t . When the search process probes up to two hops, such differentiation should continue and targets who have high trust on both the edges on the path connecting them to the querying node should be returned whereas BFS will again return a much larger set. The number of targets found at the second hop should increase when compared to the number of targets found till the first hop. But, the number of targets found as fraction of the number of targets found by BFS should decrease at second hop. This trend should continue while searching up to higher hops.

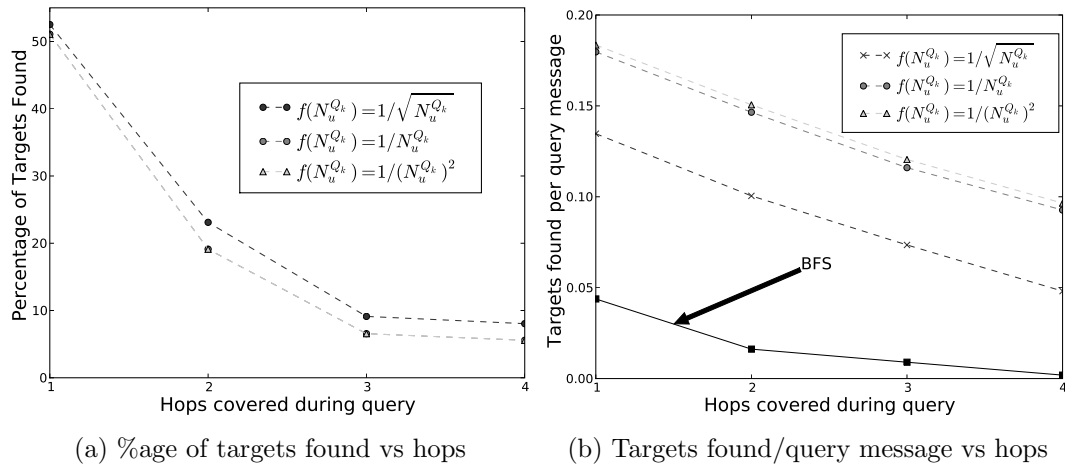


Figure 3.4: Analysis of network (liberal policy) for varying hops

The expected behavior can be observed from table 3.3 and figure 3.4a. For pruning function with $p = 0.5$, the number of targets found using our algorithm are 0.73 and 1.68 at first and second hops while BFS finds 1.39 and 7.27 targets respectively (see table 3.3 for other hop values). The corresponding % values when considered as fractions of BFS are $\sim 52.5\%$ and $\sim 23.1\%$ (figure 3.4a). The number

of targets found for pruning functions with $p = 1$ and $p = 2$ are the same (table 3.3). An important feature of our algorithm is that it finds these targets with significantly fewer query messages than BFS (see figure 3.4b). For instance, at first hop, the number of targets found per query message vary in the range of 0.13 to 0.17 for different pruning functions which is much higher than the value of 0.004 observed for BFS. The same trend continues at higher hops. The trend line falls for all cases at higher hops because of the exponential growth in the number of probed links (table 3.3).

Hop Values →	1		2		3		4	
Pruning Function ↓	QM	TF	QM	TF	QM	TF	QM	TF
$f(N_u^{Q_k}) = 1/\sqrt{N_u^{Q_k}}$	5.42	0.73	16.71	1.68	34.83	2.56	58.15	2.80
$f(N_u^{Q_k}) = 1/N_u^{Q_k}$	3.95	0.71	9.49	1.39	15.86	1.84	20.93	1.94
$f(N_u^{Q_k}) = 1/(N_u^{Q_k})^2$	3.87	0.71	9.23	1.39	15.27	1.84	20.10	1.94
BFS ($f(N_u^{Q_k}) = 1$)	31.77	1.39	448.88	7.27	3116.83	28.06	17861.28	34.80

Table 3.3: Query Messages and Targets Found per query at different hops for the network with liberal policy and $\rho = 0.5$

In all graphs, it can be seen that as the denominator of the pruning function increases, the absolute number of results obtained decreases while the targets found per query message increases. This is because the search tree is pruned more effectively leading to fewer query messages. Thus, the pruning function decides how much of the network has to be searched. Depending on the steepness of the pruning function, the curve of these graphs may change but relative differences between values observed for BFS and pruning functions will remain, demonstrating the efficiency of the algorithm. We refrain from comparing the performance of our algorithm with other related works (section 3.7) as they either use structured networks or lack the level of information, in the form of keywords and their policies, that we used to design the algorithm.

3.7 Related Work

Search in social networks has been studied under different contexts. Structural knowledge about the network has been used to model search algorithms. In [19], the structure of the social network is based on an organizational hierarchy. Structural knowledge based on geographical distance, interest, etc. was used in [10, 11, 12]. These works base their search on greedy strategy as the linking probability between nodes was modeled to be proportional to the structural knowledge. But, in unstructured social networks these strategies find limited success as searching in an unstructured social network is a tougher problem than searching in a structured network [19]. We also studied works in peer to peer networks to gain insights on unstructured and decentralized networks.

Breadth First Search (BFS) has been used to search in decentralized and unstructured networks [29]. In random BFS [30], a node chooses some of its direct friends randomly to send the query message. In intelligent BFS [30], a node ranks peers for each query (using the cosine similarity) based on their past replies. In directed BFS [31], the query initiator chooses those friends who it thinks can give good results and after the first hop the search follows the flooding process. In another technique [32], the probability of sending the query keeps on decreasing as the distance covered by the query increases. In iterative deepening [29, 31], search is continued with successively increasing TTL (hops to live for the message) till a successful result is found. Methods based on indexing or storing information of all nodes within some radius from a node [31, 33] have also been proposed.

Other search algorithms use random walks [34]. The k random walker algorithm [29] starts k random walks simultaneously instead of a single random walker. In the

adaptive version [35, 36], the parameters k and TTL are adaptively updated. In two level random walk algorithm [37], the query initiator sends k_1 random walkers with some TTL and when the TTL expires, the last nodes send another k_2 random walkers with a new TTL. In [38] the high degree nodes are utilized while performing the search through random walks. Hybrid search schemes [39] involving a mixture of random walks and flooding have also been proposed.

3.8 Concluding Remarks

This chapter modeled the flow of information in keyword based social networks. We developed a search algorithm for the given information flow settings that showed improvement in orders of magnitude when compared to BFS. The algorithm concentrated on finding a subset of results that have good characteristics. We do so with special focus on decentralization and privacy as proposed in future social network architectures. We believe that the algorithms presented can be adapted for network applications that may show graph properties similar to an unstructured social network.

As future work, we would like to explore this work in multiple directions. First, we want to evaluate the search strategy for real social network graphs as we presented a limited evaluation on a synthetic graph here. Second, we are interested in further modeling the threshold function by incorporating the distribution of ‘value’ of friends between the minimum and maximum values. Third, we would like to extend the definition of trust to bidirectional trust for an edge and model accordingly. Finally, a broader expansion of the search algorithm using semantics of query keywords will form the next stages.

Chapter 4

Conclusion

This thesis makes two important research contributions towards the study of social networks based on keywords. We believe that the algorithms proposed can be used in applications and networks that are unstructured and are similar in nature to what we assumed in this work. In this chapter, we discuss and summarize the important contributions made.

- **Social Network Model based on Keyword Categorization**

Chapter 2 concentrated on the problem of quantifying similarity amongst various users on a social network based on their profile attributes. We first analyzed a Facebook data set to gain insights into keyword usage patterns. It was found that a large percentage of keywords are used rarely by users. But, as these keywords might be related to the more commonly used keywords there is a need to categorize keywords semantically. Thus, we proposed a ‘forest’ model to categorize keywords and defined the notion of distance between different keywords. Based on that, we quantified the similarity between users and came up with a social network model.

- **Information Flow and Search in Unstructured Keyword based Social Networks**

In chapter 3, we presented a model for information flow in an unstructured keyword based social network. To address privacy issues, the identity of users is not propagated beyond their direct friends during the information flow. Then, based on the given information flow we designed and developed a search algorithm to find targets in the social network. The algorithm is based on a linear combination of topological distance and trust metrics. A threshold function was also defined to dynamically prune the network that is searched for a given query. We compared our approach with breadth first search and found that it performed better for a simulated network.

An ideal approach would be to combine both the aforementioned techniques so that the search process not only looks for topologically closer and more trustworthy targets but also for targets having profile attributes that are semantically similar to the keywords in the query. For instance, let's consider Bob once again. When he uses the keyword 'soccer' as the query keyword the targets that have 'football' as their profile attribute could also be of interest to him. A direct friend of Bob who has 'football' might be a better target than say a user having 'soccer' but who is three hops away. Thus, after combining these techniques, a metric could be defined to quantify the quality of the targets found. The metric should be based on the topological distance, trust and the semantic similarity of keywords.

Once the search algorithm that uses the combination of both the techniques is developed, another possible enhancement could be using the physical network to speed up queries. The search algorithm that we have developed mainly concen-

trates on the virtual network (or the application layer). The more popular queries could be cached in the physical layer itself which could help in speeding up the search process for similar queries. The new query could be answered by using the data stored at the physical layer without sending the query to the application layer. We leave the development of modified search algorithm and the use of physical network for speeding up queries as future work.

References

- [1] Facebook Inc. Facebook: An online social networking website. <http://www.facebook.com/>.
- [2] Google Inc. Orkut: An online social networking website. <http://www.orkut.com/>.
- [3] LinkedIn Corporation. LinkedIn: An employment related online social networking website. <http://www.linkedin.com/>.
- [4] David Crandall, Dan Cosley, Daniel Huttenlocher, Jon Kleinberg, and Siddharth Suri. Feedback effects between similarity and social influence in online communities. In *KDD '08: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 160–168, New York, NY, USA, 2008. ACM.
- [5] Lerone Banks, Prantik Bhattacharyya, and S. Felix Wu. Davis social links: Leveraging social networks for future internet communication. In *FIST '09: Proceedings of the Workshop on Trust and Security in the Future Internet*, 2009.
- [6] Lerone Banks, Shaozhi Ye, Yue Huang, and S. Felix Wu. Davis social links: Integrating social networks with internet routing. In *LSAD '07: Proceedings of the 2007 workshop on Large Scale Attack Defense*, pages 121–128, 2007.
- [7] Prantik Bhattacharyya, Ankush Garg, and S. Felix Wu. Social network model based on keyword categorization. To appear in *ASONAM '09: Proceedings of the International Conference on Advances in Social Networks Analysis and Mining*, 2009.
- [8] Ankush Garg, Prantik Bhattacharyya, Charles U. Martel, and S. Felix Wu. Information flow and search in unstructured keyword based social networks. To appear in *SMW '09: Proceedings of the Workshop on Social Mobile Web*, 2009.

- [9] Stanley Milgram. The small world problem. *Psychology Today*, 61:60 – 67, May 1967.
- [10] Jon Kleinberg. The small-world phenomenon: An algorithm perspective. In *STOC '00: Proceedings of the 32nd annual ACM symposium on Theory of computing*, pages 163–170. ACM, 2000.
- [11] Oscar Sandberg. *The Structure and Dynamics of Navigable Networks*. PhD thesis, Chalmers University, 2007.
- [12] Jon Kleinberg. Small-world phenomena and the dynamics of information. In *Advances in Neural Information Processing Systems*, pages 431–438. MIT Press, 2001.
- [13] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407, 1990.
- [14] Lada A. Adamic, Orkut Buyukkokten, and Eytan Adar. A social network caught in the web. *First Monday*, 8(6), June 2003.
- [15] Matt Spear, Xiaoming Lu, Norman S. Matloff, and S. Felix Wu. Inter-profile similarity (ips): A method for semantic analysis of online social networks. In *Complex '09: Proceedings of the 1st International Conference on Complex Sciences: Theory and Applications*, 2009.
- [16] Zhichen Xu, Yun Fu, Jianchang Mao, and Difu Su. Towards the semantic web: Collaborative tag suggestions. In *WWW '06: Proceedings of the Collaborative Web Tagging Workshop*, 2006.
- [17] Antony Lewis. Wordweb: English thesaurus and dictionary. Published by WordWeb Software, available at <http://www.wordweonline.com/>.
- [18] Wordinfo: Word information about english vocabulary. Published by Senior Scribe Publications, available at <http://www.wordinfo.info/>.
- [19] Lada Adamic and Eytan Adar. How to search a social network. *Social Networks*, 27(3):187 – 203, 2005.
- [20] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology*, 58(7):1019–1031, 2007.
- [21] Christiane Fellbaum. *Wordnet: An Electronic Lexical Database*. Bradford Books, 1998.

- [22] Daniel C. Howe. Rita wordnet. Java based API to access Wordnet, available at <http://www.rednoise.org/rita>.
- [23] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and analysis of online social networks. In *IMC '07: Proceedings of the 7th ACM SIGCOMM Conference on Internet measurement*, pages 29–42, New York, NY, USA, 2007. ACM.
- [24] Yong-Yeol Ahn, Seungyeop Han, Haewoon Kwak, Sue Moon, and Hawoong Jeong. Analysis of topological characteristics of huge online social networking services. In *WWW '07: Proceedings of the 16th International Conference on World Wide Web*, pages 835–844, New York, NY, USA, 2007. ACM.
- [25] Michael Chisari. The future of social networking. In *W3C Workshop on the Future of Social Networking*, 2009.
- [26] Leucio Antonio Cutillo, Refik Molva, and Thorsten Strufe. Privacy preserving social networking through decentralization. In *WONS '09: 6th International Conf. on Wireless On-demand Network Systems and Services*, Feb 2009.
- [27] Karan Sarda, Priya Gupta, Debdoot Mukherjee, Smruti Padhy, and Huzur Saran. A distributed trust-based recommendation system on social networks. In *HotWeb '08: Proceedings of the 2nd IEEE Workshop on Hot Topics in Web Systems and Technologies*, 2008.
- [28] M. E. J. Newman and D. J. Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263:341–346, 1999.
- [29] Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In *ICS '02: Proceedings of the 16th International Conference on Supercomputing*, pages 84–95, New York, NY, USA, 2002. ACM.
- [30] Vana Kalogeraki, Dimitrios Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *CIKM '02: Proceedings of the 11th International Conference on Information and knowledge management*, pages 300–307, New York, NY, USA, 2002. ACM.
- [31] Beverly Yang and Hector Garcia-Molina. Improving search in peer-to-peer networks. In *ICDCS '02: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 5, Washington, DC, USA, 2002. IEEE Computer Society.

- [32] R. Gaeta, G. Balbo, S. Bruell, M. Gribaudo, and M. Sereno. A simple analytical framework to analyze search strategies in large-scale peer-to-peer networks. *Performance Evaluation*, 62(1-4):1–16, 2005.
- [33] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 23, Washington, DC, USA, 2002. IEEE Computer Society.
- [34] Christos Gkantsidis, Milena Mihail, and Amin Saberi. Random walks in peer-to-peer networks: algorithms and evaluation. *Performance Evaluation*, 63(3):241–263, 2006.
- [35] Nabhendra Bisnik and Alhussein Abouzeid. Modeling and analysis of random walk search algorithms in p2p networks. In *HOT-P2P '05: Proceedings of the 2nd International Workshop on Hot Topics in Peer-to-Peer Systems*, pages 95–103, Washington, DC, USA, 2005. IEEE Computer Society.
- [36] Nabhendra Bisnik and Alhussein A. Abouzeid. Optimizing random walk search algorithms in p2p networks. *Computer Networks*, 51(6):1499–1514, 2007.
- [37] Imad Jawhar and Jie Wu. A two-level random walk search protocol for peer-to-peer networks. In *SCI '04: Proceedings of The 8th World Multi-Conference on Systemics, Cybernetics and Informatics*, USA, 2004.
- [38] Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. *CoRR*, cs.NI/0103016, 2001.
- [39] C. Gkantsidis, M. Mihail, and A. Saberi. Hybrid search schemes in unstructured peer-to-peer networks. In *Proceedings of IEEE INFOCOM*, pages 1526–1537, 2005.