Content Management and Hashtag Recommendation in a P2P Social Networking Application

Keerthi Nelaturu

Faculty of Graduate and Postdoctoral Studies

University of Ottawa

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements for the degree of

Master of Computer Science

Directed By
Prof. Iluju Kiringa & Prof. Tet Hin Yeap

©Keerthi Nelaturu, Ottawa, Canada, 2015

Acknowledgements

First of all, I would like to sincerely thank my supervisors, Dr.Iluju Kiringa and Dr.Tet Hin Yeap for their academic support and understanding which have consistently helped to steer this research in a productive direction. I would say, without encouragement from Dr. Iluju Kiringa for taking up thesis work I would not have had this success. This research work helped me understand a great deal about peer-to-peer systems in detail. Ideas from Dr. Tet Hin Yeap when working on our application have helped me a lot in thinking out-of-the box.

I would like to acknowledge highly of Dr.Ying Qiao (Postdoctoral fellow, University of Ottawa) for her tremendous support and guidance during the course of this research. Her valuable suggestions have consistently helped me in this thesis work, and have greatly contributed to the quality of the thesis. She has helped me at each step through my thesis without which I would never have made it through.

I also appreciate my team mates for their warm encouragement and co-operation.

Finally, I would like to give my special thanks to my parents, friends and god for their endless love, support, encouragement and blessings all through the years of study.

Abstract

In this thesis focus is on developing an online social network application with a Peer-to-Peer infrastructure motivated by BestPeer++ architecture and BATON overlay structure. BestPeer++ is a data processing platform which enables data sharing between enterprise systems. BATON is an open-sourced project which implements a peer-to-peer with a topology of a balanced tree.

We designed and developed the components for users to manage their accounts, maintain friend relationships, and publish their contents with privacy control and newsfeed, notification requests in this social networking application.

We also developed a Hashtag Recommendation system for this social networking application. A user may invoke a recommendation procedure while writing a content. After being invoked, the recommendation procedure returns a list of candidate hashtags, and the user may select one hashtag from the list and embed it into the content. The proposed approach uses Latent Dirichlet Allocation (LDA) topic model to derive the latent or hidden topics of different content. LDA topic model is a well developed data mining algorithm and generally effective in analyzing text documents with different lengths. The topic model is further used to identify the candidate hashtags that are associated with the texts in the published content through their association with the derived hidden topics.

We considered different methods of recommendation approach for the procedure to select candidate hashtags from different content. Some methods consider the hashtags contained in the contents of the whole social network or of the user self. These are content-based recommendation techniques which matching user's own profile with the profiles of items.. Some methods consider the hashtags contained in contents of the friends or of the similar users. These are collaborative filtering based recommendation

techniques which considers the profiles of other users in the system. At the end of the recommendation procedure, the candidate hashtags are ordered by their probabilities of appearance in the content and returned to the user.

We also conducted experiments to evaluate the effectiveness of the hashtag recommendation approach. These experiments were fed with the tweets published in Twitter. The hit-rate of recommendation is measured in these experiments. Hit-rate is the percentage of the selected or relevant hashtags contained in candidate hashtags. Our experiment results show that the hit-rate above 50% is observed when we use a method of recommendation approach independently. Also, for the case that both similar user and user preferences are considered at the same time, the hit-rate improved to 87% and 92% for top-5 and top-10 candidate recommendations respectively.

Contents

1 Introduction **1**

1.1 Motivation and Objective . . . 1

1.2 Contributions . . . 3

1.3 Thesis Organization . . . 3

2 Background **5**

2.1 Peer-to-Peer Networks . . . 5

2.1.1 Introduction . . . 5

2.1.2 Architecture . . . 7

2.1.3 Challenges in Peer-to-Peer networks . . . 8

2.2 Social Networking . . . 9

2.2.1 Overview . . . 9

2.2.2 Popular Social Networking Applications . . . 10

2.2.2.1 Facebook . . . 11

2.2.2.2 Twitter . . . 12

2.2.2.3 LinkedIn . . . 12

2.2.2.4 Pinterest . . . 14

2.2.2.5 WhatsApp . . . 14

2.2.3 Privacy Issues in Social Networking . . . 15

2.3 Peer-to-Peer Social Networking Applications . . . 16

2.3.1 Need for Decentralization . . . 16

2.3.2 PeerSoN . . . 16

2.3.3 Safebook . . . 18

2.3.4 PESCA . . . 19

2.3.5 LifeSocial.KOM . . . 20

2.3.6 My3 . . . 20

2.3.7 Vegas . . . 21

2.4 Hashtag recommendation systems . . . 21

2.4.1 Tagging in online content . . . 21
2.4.2 Recommendation Systems . . . 22
2.4.3 Typical techniques for Hashtag Recommendation . . . 24
2.4.3.1 TF-IDF . . . 25
2.4.3.2 TF-IDF based recommendation systems . . . 25
2.4.3.3 Probabilistic topic model - LDA (Latent Dirichlet Allocation) . . . 28
2.4.3.4 Topic model based recommendation systems . . . 30
2.4.3.5 Other techniques . . . 32
2.4.4 Peer-to-Peer Recommendation Systems . . . 33

3 Content Management, Architecture and Components 35
3.1 Overview . . . 35
3.2 System Architecture . . . 37
3.2.1 Client Peer . . . 38
3.2.2 Bootstrap Peer . . . 39
3.2.3 Server Peer . . . 40
3.3 Content Management Components . . . 40
3.3.1 User Account Information . . . 41
3.3.1.1 Database Schema . . . 41
3.3.1.2 Operations . . . 42
3.3.2 User generated content . . . 45
3.3.2.1 Posts & Comments . . . 45
3.3.2.2 Blogs & Articles . . . 45
3.3.2.3 Attachments . . . 47
3.3.3 Content from Social Network . . . 47
3.3.3.1 Notification Requests & NewsFeed . . . 47
3.3.4 Sequence Diagram for Operations . . . 48
3.4 Comparison Analysis . . . 48

4 Hashtag Integration 52
4.1 Overview . . . 52
4.2 Hashtag Recommendation Method . . . 52
4.2.1 Proposed Approach . . . 52
4.2.2 Implementation . . . 53
4.3 Data Structure and Algorithms . . . 55
4.3.0.1 Database Schema . . . 55

4.3.0.2 LDA Operations 56
4.3.0.3 Hashtag Recommendation Algorithms 59
4.4 Summary 63

5 Experimental Results **65**
5.1 Overview 65
5.2 Experiment Setup 65
5.3 Experiments and Results 66
5.4 Comparison with Related Work 68

6 Conclusion and Future Work **73**
6.1 Summary 73
6.2 Future Work 74

References **76**

List of Figures

2.1 P2P Data Sharing Example 7
2.2 P2P Architecture 8
2.3 Facebook Login/Sign up Page 11
2.4 Twitter Search Page 13
2.5 LinkedIn Profile View 13
2.6 Pinterest Search after User login 14
2.7 Safebook Architecture 19

3.1 BestPeer++ Architecture components 37
3.2 Our P2P Architecture 38
3.3 Client Peer Node 39
3.4 User Profile in Bootstrap Peer 41
3.5 User Profile in Server Peer 42
3.6 Sequence Diagram for User Registration 43
3.7 User Registration Graphical User Interface 44
3.8 Privacy Settings Database Model 45
3.9 Posts Database Model 46
3.10 Blogs and Articles Database Model 46
3.11 Attachments Database Model 47
3.12 NewsFeed Database Model 48
3.13 Sequence Diagram for Content Posting 49

4.1 Categories for Recommendation Modes 54
4.2 Steps in LDA Processing 55
4.3 Database Model for Hashtag recommendation 56

5.1 Hit-rate vs Number of Topics 68
5.2 Average Hashtags Per Post vs Number of Similar Users 69

List of Tables

2.1 List of distinguished Social Networking Sites 10

3.1 Comparison of P2P Social Networking Applications 51

5.1 Datasets Details 66

5.2 Hit-rate for Top-k recommendations with User Preferences and Similar Users 69

5.3 Comparison with TF-IDF Hashtag Recommendation Methods 71

5.4 Comparison with topic model based hashtag recommendation systems 72

Nomenclature

CBR Content-based Recommendation System

CF Collaborative Filtering

DHT Distributed Hash Table

DOS Denial-of-Service

IDF Inverse Document Frequency

LDA Latent Dirichlet Allocation

OSN Online Social Networking

P2P Peer-to-Peer systems

SN Social Network

SNS Social Networking Site

TF Term Frequency

TIS Trusted Identification Service

TPS Trusted Proxy Set

TREC Textual Retrieval Conference

UOT User Online Time

Chapter 1

Introduction

This thesis studies the content management of an online social network application. The application has a peer-to-peer overlay network infrastructure. The users of the online social network may publish and share content between friends. We further proposed and developed a hashtag recommendation approach for users to include hashtags in their content easily.

1.1 Motivation and Objective

Over the past few years a large increase in social media or on-line social networks in various fields was observed. Currently there are many online social networks that are predominant in the market. They include - Facebook, Twitter, LinkedIn, Pinterest, Google Plus etc. All of these online social networks have the common major goal as to associate people online. Furthermore, they are able to group the users with similar interests, help with the development of communities, serve as a platform for businesses or for communication with the masses.

Content is imperative in social networks. In general, for any online social network, content refers to any kind of data that is being exchanged, or made available for either personal use or for use by others in the same network. This might include conversations in online chat, media, personal information etc. Facebook supports posts which may contain text messages, images or location information. Twitter supports unique feature of short text messages called tweets. Pinterest is primarily an online photo sharing application. One similarity between all of the social networking applications is that, the content in these social networks is mostly user-generated content (UGC).

There is one other common similarity in all of the current social networking ap-

plications, that is, the underlying architecture. Most of the current social networks adopt centralized server architecture. This kind of architecture has both its pros and cons. In centralized architecture, we have all the applications running with their data at one location, at which one or more large computers are connected. Pros include ease of maintenance - any administration or upgrade on the system can be easily done across the components of all the applicationss. Backup and restore mechanisms are easy to implement since its just one central location and security mechanisoms can be incorporated in a simple manner. On the other hand cons include, bottleneck in performance and privacy concerns from user data perspective. In order to avoid these defects a different line of architecture pattern called the distributed or peer-to-peer architectures are being employed. Peer-to-peer systems support for user data privacy, scalability, and availability avoiding single point of failure.

Keeping this in view, we are working towards the development of a unique social networking application which has peer-to-peer architecture. The architecture is inspired from BestPeer++[28] and BATON overlay network[49]. The nodes in this architecture are categorized into three types: bootstrap peer, server peer and client peer. Bootstrap peer acts like the administrator for the network. It governs the node joining and leaving in the network. Also, it keeps a check on the health of the network. Server peer, stores the data of the users. Each server may be the data center for more than one client. A client is a user who needs to install the java based application in order to join the network.

In the social networking application that we developed, content is organized according to the architecture of the P2P overaly network. Content is categorized under posts, comments, blogs and articles. Content of as user is stored in the server peer of the user. As the user genereated content increases it becomes hard to organize ones own data. Tagging has been a way of organizing data in many of the social networking sites like Flickr, Facebook, Twitter etc. We make use of Hashtags which is one way to tag content. Hashtags are short words with continuous characters without any space in between. They are identified by the presence of '#' before the words. They can be used anywhere within the messages, phrases etc. They have been mainly used for categorizing or highlighting an event, topic, news, individuals etc. This concept has been employed in many social networking sites till date and has become popular with the start of Twitter Social Networking website. Until now these have been used for media broadcasting and business, promotions etc. We developed a hashtag recommendation approach for our online social networking platform to suggest suitable hashtags to a user.

To summarize, below we list the main objectives for this thesis:

1. To discuss a detailed overview on the architecture employed for our peer-to-peer social networking application.

2. Give an overall understanding of the content management components included in our application with their data structure and operations.

3. To provide user with hashtag recommendations when a text is ready to be published on to the network.

4. In the end, we also provide with the experimental results conducted on the application for evaluating the recommendation approach.

1.2 Contributions

This thesis mainly targets on re-modeling E-commerce platform to a social networking application. First, the architectural features of the peer-to-peer social networking application are mentioned in detail. Then we discuss the database design and operation implementations of the content entities chosen for this social networking application. The next major portion of this thesis focuses on the hashtag recommendation system. We choose topic analysis model, Latent Dirichlet Allocation(LDA)[23] methodology for implementing the recommendation system. LDA is topic mode, used to find general topic distribution in large collections of documents. This topic distribution is then made use for various applications like information retrieval and data mining etc. We further present the methods developed for the recommendation system. The recommendation system includes the components for users to initiate a recommendation operation and select a hashtag from candidate list. The recommendation system has an approach. We developed different methods for this approach. We discuss each method and the algorithms implemented for the methods in recommendation system. Finally, we examine the experiment results to understand the performance of the recommendation model.

1.3 Thesis Organization

Organization of this thesis is as below.

Chapter 2, details on the peer-to-peer architecture advantages and challenges. It also includes an in depth literature review on social networking application evolution,

various popular social networking applications that are present in the current market and disadvantage of their architecture. We then discuss the idea of have a peer-to-peer architecture for social networking applications and mention about the research on a few of the peer-to-peer based social networking applications. In the last section of the background, we explain various recommendation systems models and techniques. Importantly, we discuss two of the major techniques, TF-IDF[104] and LDA Topic Models[23], used for most of the hashtag recommendation systems.

Chapter 3, illustrates the architecture of our peer-to-peer based social networking application and the content management components. In this chapter, we discuss the modified implementations of Bestpeer++[28] architecture and BATON[49] overlay network. We also describe the database design and distributed operations performed on the content management modules. In the end, we provide a comparison analysis with the other available peer-to-peer social networking applications.

Chapter 4, we discuss in detail the hashtag recommendation methodology, the database design implemented for the same, operations and the algorithms included in the model.

Chapter 5, presents the details on the datasets used for experiments, the test setup environment and all of the experiments performed to ensure the correctness of algorithms and to calculate the performance of the algorithms.

Chapter 6, gives the conclusion and future work for this thesis.

Chapter 2

Background

We begin this chapter by giving brief overview on peer-to-peer networks, various architecture patterns, their characteristics and mention about challenges that are observed commonly in peer-to-peer networks. Further in the chapter we discuss about evolution of social networking applications and detail on some prominent applications each having a distinctive purpose. Also, we specify about privacy issues often observed with current social networking applications and we review various peer-to-peer social networking applications on their architectures. Finally, we detail on recommendation system approaches, related work on hashtag recommendation systems and factors to be considered for peer-to-peer based recommender systems.

2.1 Peer-to-Peer Networks

In this section we mention in detail about peer-to-peer networks, their architecture and general challenges seen in peer-to-peer networks.

2.1.1 Introduction

In the current scenario with the increasing requirement for great deal of computing power, using traditional standalone computer systems would restrict an application's efficiency. Getting this capacity in terms of memory size, processing speed is not possible with single user systems. Distributed Computing, is the area of study which supports handling large scale computing requirements. It basically involves network of computers connected not just for utilizing its potential but also, for providing the network with themselves as resources. These computers need not be located in the same geographical area. Two major classifications in distributed systems are :

Client-Server and Peer-to-Peer Systems.

Client-Server Systems: Server is a machine which provides clients with all the needed resources. It acts like a service provider. Number of clients might be connecting to the same server. Server and client communicate over a computer network which has dedicated lines of connection between them. A simple example is Internet Browser. Browser acts as a client when user tries to access a web page. Whereas the server is the provider which has all the data required to serve the web page request.

Peer-to-Peer Systems (P2P): Pure case of P2P networks have multiple computers of different capacities connected via overlay network. Overlay network is a logical network on top of the physical network. There is no specific identification of client and server in this architecture. All the nodes(computers) connected in the network acts as resource as well as providers.

Our research mainly focuses on P2P Networks. Existence of P2P networks dates back to 1979, though it started with USENET and fidoNET in 1984[94]. As defined earlier, multiple nodes are connected across a logical network and each node is self-governed. At any point of time not all nodes are physically connected. Hence, emphasis is on the overlay network. For example, consider a case of P2P data sharing application as shown in Figure 2.1. Each peer or node has a metadata information stored. This information contains details on content at the node, details on what data might be available at other nodes based on previous communications, and much more information might be encapsulated in the metadata at a node. Suppose Peer 3 needs a file. First a query search by Peer 3 reveals location of the file, assume it to be at Peer 1. A routing algorithm based on the network topology and restrictions determines a path via which both nodes can communicate.

There are a wide range of applications for P2P networks. Examples:

1. As discussed above they can be used for File Sharing applications
2. For large scale computing tasks like SETI@home[19] project
3. Helps in anonymous sharing of resources due to control that a peer has on its own content
4. Collaborative communication like Chat rooms, Instant messaging services and many more..

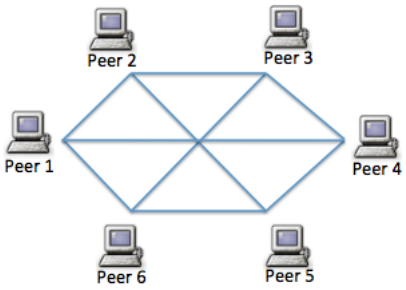

Figure 2.1: P2P Data Sharing Example

2.1.2 Architecture

In a broader sense P2P Networks are classified as Centralized and Decentralized P2P systems Figure 2.2.

In centralized systems, certain fixed number of servers act as mediators between client peers. They direct the peers to the node locations after which a direct physical connection is established between the peers as in a normal P2P system. Examples: Napster[5], BOINC[18] etc. Though these kind of systems are faster in action, the disadvantage lies in the fact that, the central servers might be cause of failure of the entire system or they might also face bottle neck issues.

Decentralized systems are the roots for traditional P2P systems. There are no special servers assigned. All nodes are equally responsible. These systems may be further classified into Structured and Unstructured based on the network topology maintenance.

For Unstructured networks the topology is not fixed. They mainly make use of flooding and random walk techniques for finding the location of the requested resource [69]. These kind of networks work only when there is good bandwidth for easy spread of messages. The drawback with this is the unnecessary message exchange between nodes which do not have any information or data related to the query and also since it dependents on the bandwidth, scalability might be an issue. Examples: FreeNet[32], Gnutella[79] etc.

On the other hand, structured networks make use of DHT (Distributed Hash Table) for storing the information about the nodes and the data. They have the information in the form of key/value pairs. This is maintained mainly by the peers themselves. DHT basically acts as both lookup and forward service. It also helps nodes in identifying the destination location in flexible manner. Although, the overhead lies in maintaining the DHT, a DHT has to be kept updated of nodes that join

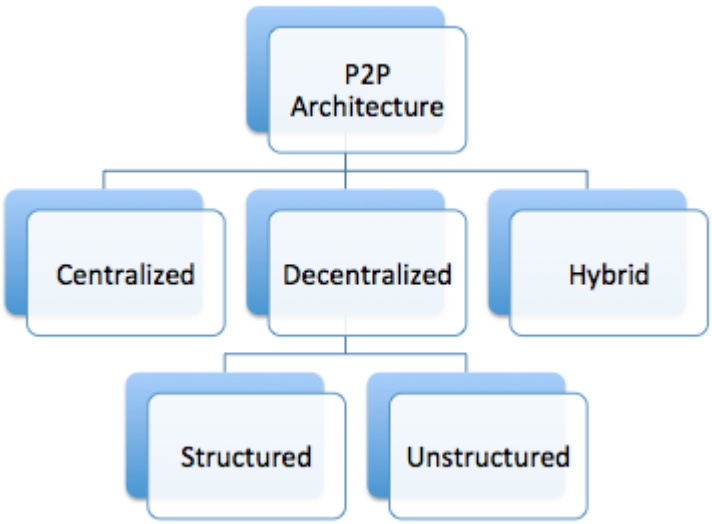

Figure 2.2: P2P Architecture

and leave the system. This might sometimes cause an update in the overall system. Examples: OceanStore[58], PAST[37] etc.

Apart from these two kinds of classification, a new sector of systems have evolved over the time. They are the Hybrid P2P systems. These are formed by combining features of both centralized and decentralized systems. Certain peers are marked as Super Peers in these systems which act like servers and work as a lookup and forward service. Examples: BestPeer[14].

2.1.3 Challenges in Peer-to-Peer networks

P2P networks come with some excellent utility factors. Since it is a large network of systems, any organization or application can make use of the network to easily propagate information. Also, if we consider many of the content delivery or data sharing applications built with P2P as the architecture, the main attraction for a user is the control they have on their content. It depends on the willingness of the user whether to take complete responsibility of the content or to share it with a pool of resources that will also have control on the data. Another major benefit is resource utilization. In good old days, we built super computers for all the huge computation tasks. But now, we could utilize a network with potentially underused resources for the same. Issue of single point of failure is not seen in P2P networks. If one system goes down, we have many back up systems ready for replacing the

same. Current generation is flooded with many collaborative applications with users located in different places across the world. A simple P2P file sharing application like BitTorrent[1] enables sharing of content across continents without the need for establishing complex structures.

Though all the above factors discussed show off P2P networks, they come with a number of challenges and issues. Consider a file sharing application, a P2P network without good replication scheme faces the threat of resource availability. Nodes join and leave the network at their choice. There is no restriction or compulsion on the peers to be available. This causes an unreliable behavior in the network. In order to avoid the issues with availability, many P2P systems employ replication strategies which raises the next big task to handle. Validity and consistency of the data across the network. A kind of version check has to made each time data is updated at the origin node. This might push in extra traffic in the network. Impartiality has to be maintained through out the system. No one node must be flooded with requests while some nodes are not being made use up to their ability. Motivation must be provided in the interest of peers. The other service which became a big problem for the system is anonymity. In P2P system, the peer has a choice to be unidentified. This might be fruitful for the peer, but does affect the trust element in the network. This forces the system to implement some trust and reputation mechanisms which would evaluate the integrity of the node and also the data provided by the node. Lastly, apart from all of the issues, security threats are pretty common in these networks. Some of these threats include DOS (Denial-of-Service) attack, Sybil attacks etc.. might be easily forced on the system [89, 94].

2.2 Social Networking

2.2.1 Overview

Dictionary meaning for Networking is "interact with other people to exchange information and develop contacts". Social Networking specifically deals with internet applications or websites that facilitate in developing inter-personal relationships or those that promote networking. Their foundation is based on "six degrees of separation" principle [60]. It says any two random persons in this world can get connected in no more than six steps. Most of the Social Networking applications are motivated by this idea.

Online Social Networking(OSN) site helps an individual to create a "digital iden-

Year Established	Name of the OSN Website
1997	Sixdegrees.com
1999	Cyworld, LiveJournal
2002	Friendster
2003	MySpace, LinkedIn, Hi5, Delicious
2004	Orkut, Flickr, Facebook
2006	Twitter
2008	Yammer
2010	Instagram
2011	Pinterest, Google+

Table 2.1: List of distinguished Social Networking Sites

tity" and also enable for communication irrespective of time, space and platform shifts[39]. The first of OSN applications was sixdegrees.com. This was launched by Andrew Weinreich in 1997. It lasted until 2001. The site provided with many features like creation of personal profiles, maintaining of friend lists - a friend in social networking is anyone who you authenticate to be able to view profile and allow them to communicate using the application[39], instant messaging and a search for finding new people through the existing connections. Though this website could not last long due to revenue and spam issues at that time, it gave a push to many other new application. In 2002, Friendster[25] another OSN came into lime light. The main idea was to utilize it as an online dating application. It was founded by Canadian computer programmer Jonathan Abrams. It was one the sites which attained one million users in that period. The site allowed people to create a profile which detailed on personal interests and demographical information and also post public comments about one another. Unlike the six degrees concept, Friendster allows for connection between people who are at a distance of four degrees[25]. Over the time many new OSN's have come up like Facebook[2], Orkut[6], Twitter[48] etc. According to Alexa the internet company which provides web traffic data, Friendster ranking has gone down from 40 to 800 by November 2010. Later, the website got revamped and in 2011 it was announced as an online gaming community. In table 2.1, we list some of the dominant social networking websites of time along with their year of establishment.

2.2.2 Popular Social Networking Applications

In this section we review few reputed social network applications.

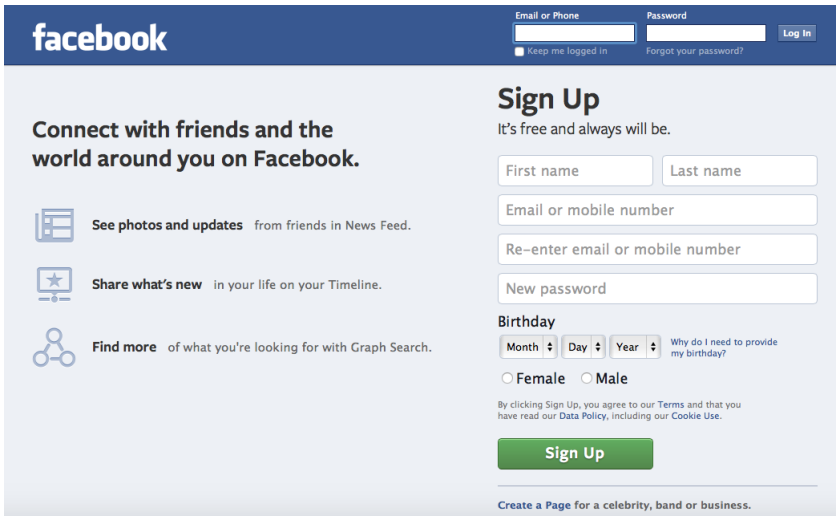

Figure 2.3: Facebook Login/Sign up Page

2.2.2.1 Facebook

Facebook[2] is termed as the brand of Social Networks. Its fame has reached a state that almost 80% of studies on Social Networks have been targeted at Facebook[81]. It was launched in February 2004, by Mark Zuckerberg and his friends. According to a Facebook[2] as mentioned in [72], there 1.39 billion monthly active users currently on Facebook and its a 13% increase over last year. Some of the features of Facebook are creating user profile, adding content like posts, comments, photos, videos etc, connecting with friends online, gaming and also an API that allows for developing apps on Facebook.

Encompassing all the features mentioned above, Facebook has emerged into a perfect social networking platform - a base for the upcoming applications. Recently, Facebook also established itself as a platform for online business via targeted advertising. Various means have been enabled for accessing and sharing content to Facebook. It is also linked up with Android, IOS and Windows mobile operating systems, which helps users to use Facebook when on the move. Many of the websites and mobile applications provide for options to share content directly to Facebook and in some they use Facebook as part of their authentication procedures. On the downside, varios studies [16, 17, 62, 70, 52] indicate that there have been many concerns as to trust, privacy and even behavioral issues like social networking addiction among students, adults etc. that have increased with the introduction of Facebook.

2.2.2.2 Twitter

Twitter[48] is also an OSN similar to Facebook. It was launched in July 2006 by Jack Dorsey, Evan Williams, Biz Stone and Noah Glass. Most of the content on Twitter is public unless a user restricts his account to be private. Users don't have to register in order to view Twitter blogs. They could just use the twitter search for viewing content posted in the twitter timelines. Twitter content consists of short-messages with 140 as character limit. These messages are called "tweets". Twitter timeline is a user space where all the tweets posted are listed in chronological order. In twitter, people can follow others profiles, by which tweets of the user and the follower are displayed on their respective timelines. It also permits them to exchange direct messages[11]. There are some common functionalities specific to twitter. Use of '@username' in a message, mentions the tweet on the referred users timeline even if he/she is not followed. Selecting the @mention timeline would list all the references made to the user, this might include tweets by themselves or by other twitter users. Re-tweeting is another concept of Twitter, where in the user shares a tweet already posted in the twitter timeline either by copying the content text, sharing the link that have been already shared. The other main tool that came into existence because of twitter is the use of Hashtags. A hashtag is any text which is preceded by '#' symbol. A hashtag typed in along with a tweet by a user forms a link to the tweet and any future tweets that use the same hashtag. A click on the hashtag redirects the user to a timeline which lists all the tweets referring to that specific hashtag.

Twitter currently boasts of 288 million monthly active users with 500 million tweets per day[48]. Like Facebook, twitter has moved on to different sources for access like mobile applications and integration with third-party applications for sharing. Twitter has been in use for many purposes. Mainly for information diffusion which might include about latest events happening, conferences, disasters. As mentioned in [80], it has been in use for crime investigations, communication at work and during media alerts. Studies[106, 101, 65] on twitter relate to Tag Recommendation systems, privacy issues, hashtag aggregations (on which we concentrate in this thesis going further) etc.

2.2.2.3 LinkedIn

LinkedIn[4] is also a Social Networking Site (SNS) introduced in May 2003. It is completely business oriented application which facilitates collaborative environment between job seekers and employers of various organizations around the world. Job

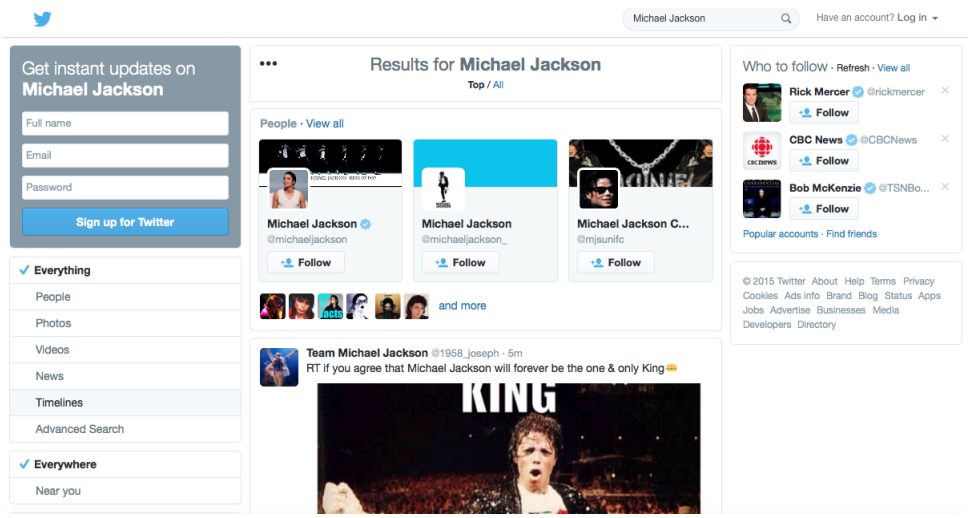

Figure 2.4: Twitter Search Page

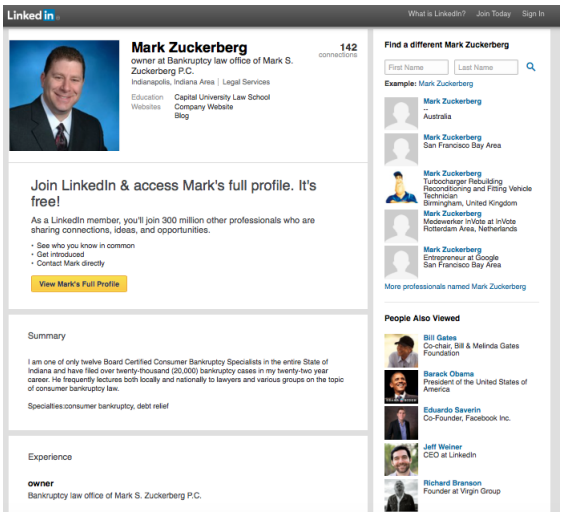

Figure 2.5: LinkedIn Profile View

seekers get to create their profile which can be viewed by employers depending on their choice. They can even apply for job postings. Users can connect with people by sending them invitations and improve their network, follow other users professional updates, join groups of various registered organizations. Recruiters can engage with the network by posting details on new jobs available, provide with useful information like recruiting tips and insights etc. Organizations on the other hand can show case their companies work culture which might attract good resource pool. There is even provision for premium membership which favors for more upgraded profile views with insights into organizational details etc.

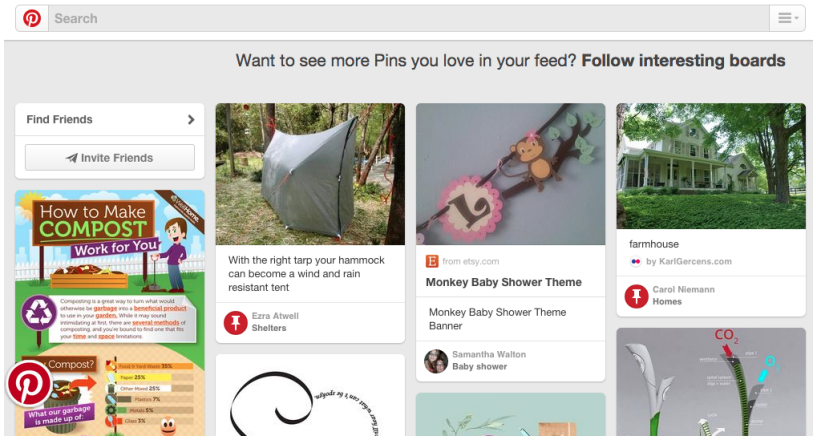

Figure 2.6: Pinterest Search after User login

2.2.2.4 Pinterest

Until now, the main mode of communication in the SNS were using messages, although other digital content could be used even. Pinterest[8] is held under a different category of SNS, its main focus is on "visual bookmarks". Visual bookmarks are images from various third-party applications and direct from users themselves. A user is given a Board under their profile which can be used to "pin" various of these bookmarks and also categorize them according to one's own wish. They are even provided with search option to view content pinned by other users, follow others content and even send messages to others. They even support for integration with browser which allows for easy pinning of content that you view on numerous web pages.

2.2.2.5 WhatsApp

Formerly, we have discussed several applications which are mainly targeted towards web browsers. In the current generation substantial applications being developed are directed towards mobile technologies. Multitudinous mobile social networking applications have come into spotlight. Examples include WhatsApp[74], WeChat[13], Viber[12] etc. We discuss WhatsApp in order to complete all classes of social networking applications. WhatsApp is an instant messaging application which connects with all the contacts in a mobile through their mobile numbers. The main restriction here though is that the contact should also have the same application installed. WhatsApp was founded by Brian Acton and Jan Koum in 2009. As of October 2014, WhatsApp has around 600 million active users[74]. It has been passed on across diverse platforms - Android, iOS and Windows. It not only serves for text messaging, but users can share images, videos (restricted length), voice notes recorded instantly.

Special features like showing the last seen status of the users, read receipts as to whenever user reads the messages, attract customers. Recently, in January 2015 they have made the application available as Web Client to be used on desktops.

2.2.3 Privacy Issues in Social Networking

In the previous section we discussed a diverse set of social networking applications. All of the major SNS have one major issue to be addressed - Privacy control. As we know, in OSN's the bulk of data or content getting added daily is user-generated. As per our understanding for this type of data, there are two kinds of issues specific to privacy.

1. When user is adding content to the SNS, they might not know that content they are adding is open for public. Here we quote as said by many: "Once something is online it doesn't go away".

2. The other case is, once we add some data on to an SNS we never know what purpose this might be used further without our acknowledgement by the service provider.

In the first case, privacy issues come up due to the lack of user's knowledge on privacy settings provided in the SNS. According to Barnes S.B [21], among the high volume of end-users, social media appeals to teenagers the most. Many cases have come into sight with pedophiles stalking for preys by way of these social networks. Solutions proposed by Barnes S.B [21] call for the involvement of parents, schools and even government officials in order to regulate these mishaps. They propose some technological and business strategical changes to be made by SNS.

Our main concentration for the social media application was on privacy issues caused because of the second case mentioned above. Many of the current social networking companies like Facebook, Twitter, Google etc are facing serious charges due to their disregard to data protection legislation. For most of the OSN's their business revenue models are based on mining the private and personal data collected using profiles registered. Behavioral patterns are formed based on the profile gathered and as a result of data-mining studies conducted on them. This information obtained is then sold further to numerous third-party services without consumers approval. Facebook's marketing tool Beacon[50] launched in 2007 was all about social advertising also termed as targeted advertising(recommendations to users and their friends based on their online shopping behavior) was one such story. Facebook hoped for a beaming 15-billion revenue over this concept, although it was forced to withdraw due

to privacy concerns. In a similar fashion, Google also faced the wrath of government privacy regulations. Google's StreetView project[84] started between 2006 and 2010, installed many cars patrolling streets for images as part of their map collection. Investigations revealed that they have collected personal data from public wi-fi routers which was breach of privacy. Twitter faced many attacks from hackers which exposed personal data in many accounts and also spams from celebrity profiles. Peer-to-Peer archictecture could help suffice these issues and attacks. In the next section, we discuss various P2P based social networking applications in research.

2.3 Peer-to-Peer Social Networking Applications

2.3.1 Need for Decentralization

A visible deduction for the security and privacy concerns discussed above is because of the central authority that most of the OSN applications have on consumer data. Nearly all of the of the SNS in the current market have adopted for a centralized architecture. Though the user gets a feel of managing his data using the privacy settings provided by the applications, the master control on the data is in the hands of the OSN provider. In such an application even if a user deletes his account from the site, the data of the user still remains with the provide. P2P infrastructure was proposed as a solution in many studies [33, 66]. P2P networks as discussed before provide distributed control over the data. Some of the main concerns like Privacy, Access Control, Ownership of data and information propagation can be controlled at a very minute levels by the users. This has been proven true at least from the use of the many P2P data sharing applications. Many researchers have applied the same strategy for building social networks and we discuss some of them going further in this section.

2.3.2 PeerSoN

PeerSoN[24, 26, 85] is a German based P2P OSN prototype model. Their main objective was to provide user with complete control over the data and encryption mechanisms to authorize distribution of content. They employed OpenDHT as their distributed lookup service. The reason for this was they needed an "always online" service, which would make it possible for peers to bootstrap into the network. OpenDHT consisted of super-peers which would also help in finding the files and also in the encryption mechanisms. For the peers in PeerSon the unique GUID for identifying each

peer was composed by applying MD5 hashing over the users email address. Once a peer is online their GUID's are listed with the DHT. DHT also maintains the status as to which device the user is online from mobile or local etc as part of GUID, which they call Extended GUID. Time-out function is also included with OpenDHT. This enables the DHT to find out about peers who might have force logged out of their clients, file versioning etc. Synchronous messages when peers are online was easy to implement in P2P network even, since both the peers communicating are online. For Asynchronous messages though they had to use few solutions. Delegates directed by OpenDHT was solution implemented in this prototype. In this, when a peer wants to communicate with another peer who is not online, they cache the message for that interim period in another neighbor peer who is online. This neighbor peer is chosen from the delegates list already chosen by the receiver and stored at OpenDHT super-peer. Once the receiver logs in again into the application, it checks with the OpenDHT the delegates which are online and gets the missed message details from the neighbor peers. Access Control[24] is another major area to be concentrated for P2P OSN. When a peer uploads a file, the file is added to a group and this group is assigned a list of authorized users. A Key List object is maintained for each file group which is signed by the owner with his private key. Owner has a public key with which he encrypts with the objects of a file group. Owner shares Readers key with authorized users, who can decrypt the object using the key. This is the process for Read Access Control. Similar mechanism along with delegates is used for Write Access Control.

A complete pure web platform is not a possibility because of availability issue. For the P2P infrastructure users are needed to download client in order to join the network. Unless PeerSoN is a very popular application it cant be said its an "anywhere available" OSN. File availability was increased by means of caching and replication at neighbor peers. OpenDHT was a prototype for testing in PlanetLab in that period. So, it was an impossible task to extend the OpenDHT for any new changes needed for PeerSon. OpenDHT also had other disadvantage of restricted key length(20 bytes) and value length(1024 bytes). Although, it allowed for each key to have two values. The team also dint implement keyword search and an exact-match-string has to be passed, if we need some friend details or file information.

2.3.3 Safebook

Safebook[34] is online privacy-preserving social network. Its main objectives were to provide user data protection, privacy and also to make use of real-life trust relationships in order to increase the trust factor in the network. A decentralized three-tier architecture was suggested for the prototype. The layers are Social Network (SN) layer, P2P substrate layer with Trusted Identification Service (TIS) and Internet layer. The SN layer consists of user forming Matryoshkas, concentric circle like structures around the user at the core. A node in this network should have a friend who is already part of this network. When a node tries to join it needs to provide identity proofs, these can be by face-to-face meetings or by exchange of credentials like passport etc. This avoids the impersonation attacks that might happen in general OSN. When a user creates an account he is provided with Node Identifier and Pseudonym by TIS. Pseudonyms are used by other peers which are outside the Matryoshkas circle to communicate with a node. Hence, integrity and privacy are maintained in this system.

There are three kinds of nodes in Matryoshkas, one is core peer node, second innermost peers which are the highly truster relationships by the core node - they are called mirrors and third, entrypoints which are on the outermost circle of Matryoshkas. All online communications are handled by core node, while offline communications are handled by mirrors. Entrypoints handle any data request that comes to the core node. Path creation is the next step after node identifier and pseudonym are assigned. For this task, the core node contacts the friend node which send a register message within a span of hops and also a time-to-live assigned to the message. Whichever node is available registers itself in the p2p lookup service as an entrypoint. Each friend or peer in a matryoshka is assigned a trust level by which further permissions on data requests are based. Closely associated contacts are assigned as mirrors and data is replicated on them.

Data is assigned three privacy type - private, protected and public. Private is not shared with any of the peers, protected data is replicated with the mirrors after encryption and public data is published without encryption for viewing by everyone. The only sellout with this architecture is the lag in performance due to hop-by-hop exchange of messages and also the encryption mechanism to be done at each level of the Matryoshka. Though TIS is a central server, since it doesn't have the ability to contact users or private data.

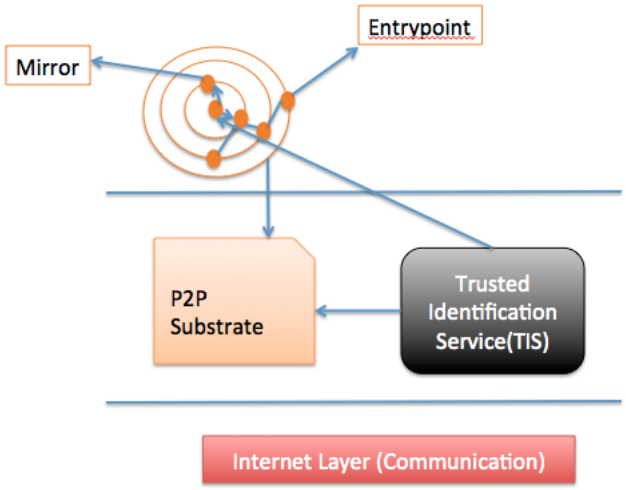

Figure 2.7: Safebook Architecture

2.3.4 PESCA

PESCA[82] is another P2P OSN developed with the goals of data privacy and availability. A User Online Table (UOT) is framed based on users availability history. This table even categorizes based on the device and location from which the user might be online. Example: A user at work might be online from his desktop at work. Same user might login on a mobile when at home. This pattern might change for each user during weekends. Also timezone of the location from which the user is online also will be considered. When a user wants to share content, there are two kinds of friends he might select - 1. Online Direct/Indirect Friends 2. Audiences TimeZone. For the first kind, the user shares the data with friends in his circle, who are online at the time of replication. For the second, user replicates copies on to his friends who have their UOT similar to that of the audiences who would request for the data. Apart from the friends characteristics there are few more factors, based on which the authors devised a greedy algorithm. This algorithm would then give out the list of friends who serve the best for storing the replicas. These factors concentrate on the amount of storage each friend might have to give away for the sake of the user. They are : 1. Amount of storage selected friends would contribute 2. Maximum number of replicas the user want to create 3. Frequency at which the data should be made available 4. Type of devices from which the replica candidates might be available. In PESCA, they use the Broadcast Encryption (BE) [41] algorithms and strategies for exchange of information. A user is assigned a GUID which is generated by hashing their email address. A virtual identifier which facilitates when communicating with

other peers is also generated. A specific secret key is also generated by the user for each friend. This secret key is used as an authentication parameter when sharing the data.

2.3.5 LifeSocial.KOM

LifeSocial.KOM[43] is a plugin-based P2P architecture for OSN. The main motif is to provide with all the functionalities similar to current OSN and reduce the cost of service providers. It uses structured P2P overlay for storing data objects which can be retrieved in a secure manner. FreePastry and PAST are used for ID-based routing and data retrieval of objects. Data objects are of two types - final or distributed linked objects. Final objects contain the data like photos, profile names etc. Distributed linked objects have links which forward further to final objects. Public key and symmetric cryptographic key are used for encryption. Users are identified by their public keys. Each data object is encrypted with symmetric key, which is then encrypted with public keys of all users who can read the data. This goes for only publicly available data. Replication strategy used in PAST is implemented in this system.

2.3.6 My3

My3[71] is P2P social network developed with profile availability as the main goal of the system. User profile is made available even when the peer is not online, by replicating them at peer nodes which are friends. Friends are chosen for replication based on two factors: geographical location and online availability of a node. The list of users, where the profile for a particular node is replicated, is called Trusted Proxy Set(TPS) . TPS computation and online time graph is computed based on factors mentioned before. Online time graph is connected graph with edges between friend and atmost two trusted friends. By this the replication points are decided. For each friend, the replication for a user is at a node which has overlapping timezone with the friend. This node is called mount point. For a users profile, a friend would access the mount point. If the availability is to be increased, then the authors suggested for making all of the friends with replications as mount points and to make the closest available node the primary mount point. Also, the replicas are to be made only at highly trusted nodes to minimize replica overhead. To minimize the access cost, a user assigns the nearest friend as the mount point. The address and TPS details for a user are stored in DHT at the resource that user provides.

2.3.7 Vegas

Vegas[38] is a P2P OSN with a design that maximizes security and privacy. The authors intentions are not to replicate behavior of Web OSN like Facebook, but to provide for a social networking infrastructure which can be extended. They try to reduce social network pollution, which means unwanted friend relationships added by mistake. Complete control of data is made available to the user. A users profile is made available all time. Mobility support is also added for communication between users. Users in vegas know about only their immediate friends. Peers do not have any idea on the relationships between their friend nodes. A data store is used for profile availability and to share content. Data stores examples are Amazon C3, Dropbox etc. Friend relationship is also added either by face-to-face communication between the users or via email exchanging or by coupling mechanism. Coupling is a process in which, a user sends a coupling request to its friends, if he thinks that they might become friends. Social search and directory services are suggested as the enhancements with this architecture.

2.4 Hashtag recommendation systems

2.4.1 Tagging in online content

Tags have been used to refer to the meta-data of online digital content. These meta-data describes the implicit information of such kind of content. The digital content in social networks can include textual or digital images or videos or anything data that is exchanged over the social web. Whenever a digital content is passed on via social media there is some implicit information that gets attached and is made available for the end-users. For Example: a photo shared will add in the EXIF information, a blog created will have the date when it has been published. This kind of information though not mandatorily added by user, gets attached to the content. Since its "data about data", we call it metadata. Adam Mathes defines metadata as "data that allows to collate information, and helps users find relevant information"[67]. Metadata can be added explicitly by user, these take the mode of free-form keywords added along with the digital content. These keywords are called as "Tags". Tags[76] constitute an area of study called "folksonomy". Folksonomy (folk + taxonomy) a word authored by Thomas Vander Wal[95] deals with user generated tags, which don't have any structured arrangement or parent-child relationships. They help many large information retrieval systems by organizing content with complementary wisdom

provided by users along with the content. Many social networks have provided tagging in various forms. There are:

1. Picture Tagging - Adding keywords to pictures
2. Username Tagging - Tagging users on pictures, textual content etc.
3. Geo Tagging - Tagging the location from where the user is adding content on the social network.

One type of tagging mechanism called "Hashtags", was invented by Twitter social networking application way back in August 2007. Chris Messina first used it for posting messages directly related to groups [68]. Hashtag is a string of words or characters preceded by '#' symbol. Example: #TGIF. Initially it was used to identify related messages under one shade. First, when a user adds hashtag to a message it is created as a hyperlink – clickable text. Once a hashtag is placed along with a message, in future whoever searched twitter for that hashtag is displayed with all the results related to that hashtag – the ones which have the same hashtag in the message or the ones which might have similar meaning. Purpose of hashtags can be interpreted in many ways. It can be used for forming groups, an individual can organize his content for personal use, highlight events or happenings like disasters [80], companies also started using this as strategy for marketing termed as – hashtag marketing, gaming, bashtagging, rhetorical hashtags etc. [68]. This thesis focuses on making use of hashtags for organizing and linking content.

2.4.2 Recommendation Systems

The other research point in this thesis is hashtag recommendation system. Here, we introduce detail on recommendation systems. Recommendation system is a facility that has been used in web applications for "*predicting the user responses to options*"[61]. It involves the technique which is used to make suggestions to the users based on certain selected criterion. Recommender systems or recommender systems have been there since 1990's[15]. Initially the recommender problems focused on estimating ratings of items that were not seen or purchased yet by the user. Basically, the problem statement of recommender systems can be stated as: If there is a set of users U in a system, then there is a set S of items that can be recommended to these users, which might make it simple for the users in searching for needed items. Recommendation methods have been classified into two major types: Content-based and

Collaborative Filtering. They focus on different perspectives while making recommendations. Content-based methods are specifically on the "properties of items", and collaborative filtering methods are on the "relationship between users and items"[61].

Content-based recommendation systems (CBR), base their suggestions on the similarity between the items. One major reason behind this content-based approach is that a user always selects similar items. Therefore, during the recommending operation, these systems match the profiles of the items with the profiles of the users. The user profile describes the preferences and interests. These preferences and interests are collected by the system through analyzing the information of the features extracted from the items that the user has purchased or shown interests previously. The major task in these systems is to identify specific feature sets for the items or content. In the case that items are content, the features for content can be identified by using keywords or tags associated with them. Identifying similarities between content based on these feature sets and also including the user preferences as the benchmark is what goes on in content-based recommendation systems. User preferences can be evaluated based on their likes and dislikes, profile attributes, ratings provided by the users on previous items of interest or comments/feedback given by the same user[64]. There have been many of the application which used this approach for recommendations like Music Genome Project[7] which recommends music to users based on the keywords associated with music files and user likes, Fab system[20] recommends web pages to users etc.

Content-based recommendation becomes better approach for suggestions in a system where there are not many users. This is because, in a recommending operation for a certain user, such approach does not require data from other users. However, these systems also have some corresponding limitations[64]. First, the annotations that are added to the content either automatically or manually always will have limited details. It has been seen that the keywords identified for web pages might not contain any information about the media embedded in these web pages. Another limitation is termed as over-specialization. When recommendation is based on the content already rated by a user, the concentration is restricted to the area already visited by the user. The data outside the domain of the likes of the user might not be considered. Although the dependency with other users in the system is reduced, for a new user, proper recommendations can not be made until sufficient data about the user's interests have been collected.

Collaborative-filtering recommendation systems, base their suggestions on the similarity of the user's choices on two items. The users of these choices always have

similar tastes[15]. For example in [31], Collaborative filtering (CF) method employs the nearest-neighbor algorithms to recommend products to a target customer based on the preferences of the neighbors, who have similar interests as of this customer. GroupLens[56], Video Recommender[46] and Ringo[87] were the earliest applications using collaborative filtering systems. In CF systems, the user data considered for identifying similar users can be classified as implicit and explicit[53]. The explicit data includes ratings values of items, user's interests / preferences, and feedback or comments obtained from a user. Inference from explicit data can be easily done by observation. The implicit data can be derived from the data collected by monitoring user's activities or, user's behaviors amongst groups, looking up content published by the users, following user's navigation data by means of web mining techniques, and observing items on for expended time. Though CF methods avoid some of the limitations of the CBR methods mentioned above, there are some drawbacks even with the CF methods. One of them is the same as CBR methods. In order to compare, the interests of a new user with those of others, the CF methods need the information about the ratings or items the user is interested is needed. Other problems are related to the new content added to the application and also sparsity issues. For a new item, it takes some substantial amount of time for the system to collect rating details from other users. Without this data, this new item might not be taken into consideration by the recommendation system at all. Sparsity issues are caused by the fact that, the number of ratings already obtained is usually very small compared to the number of ratings need to be predicted[15]. Some content might be rated high by a small number of users with peculiar interests. Considering user's profile information apart from the rating data will avoid such sparsity issues.

To avoid limitations of different techniques, many applications implement hybrid recommendation strategies wherein they use both content-based and collaborative filtering techniques are adopted. For our hashtag recommendation, we use hybrid recommendation technique.

2.4.3 Typical techniques for Hashtag Recommendation

In this section we discuss two commonly used techniques by hashtag recommendation systems. One is the TF-IDF weighting method is most popular scheme used for information retrieval[23] and other one is Latent Dirichlet Allocation (LDA) which we use in our hashtag implementation.

2.4.3.1 TF-IDF

Term Frequency-Inverse Document Frequency (TF-IDF) is the method generally used in text mining and information retrieval systems. It calculates the importance of a word in a document against a corpus. TF-IDF also is offset by the frequency of the word in the corpus. The weight in this kind of measure is indicated by a vector space model. Term Frequency (TF) of a term in a particular document measures the number of times a term appears. For a term, if it occurs atleast once in the document then the value in the vector space will be non-zero. Equation2.1 shows the tf weight calculation, where $m_{ij}$is the number of times a term (t_i) appears in document (d_j). The denominator calculates the total number of terms in the document (d_j)[77].

$$tf_{ij} = \frac{m_{ij}}{\sum_k m_{kj}} \tag{2.1}$$

Inverse document frequency (IDF) of a term is calculated upon overall corpus not just one document. It gives the importance of a term in the complete document corpus. For a word, a higher value of idf indicates, that word appears only in few documents. This implies the word is of low significance even though it has a higher tf value. Equation 2.2, shows the idf weight for a term (t_i), $m_i$is the number of documents the term has appeared in. M is the total number of documents in the whole corpus.

$$idf_i = log\frac{M}{m_i} \tag{2.2}$$

TF-IDF weight for a term (t_i) can be calculated using equation 2.3

$$TF - IDF_{ij} = tf_{ij}.idf_i \tag{2.3}$$

In this method, generally document length also plays as a factor. Longer documents tend to have higher values due to the increased number of words and word repetitions. Hence, while calculating the weights of the terms, this approach always normalizes these weights with the length of the documents. Using the vectors created by the TF-IDF method, a recommendation system is able to measure the similarity between two documents by calculating the Cosine distance or the Euclidian Distance of their vectors.

2.4.3.2 TF-IDF based recommendation systems

Zangerle et al, proposed a hashtag recommendation for Twitter in 2011[104]. Their main aim was to recommend appropriate hashtags to a user in entering a tweet and,

meanwhile, avoid the use of synonymous hashtags for any tweet that user enters. Their approach describes a three step process:

1. Obtain existing tweets similar to the typed tweet from a crawled dataset.
2. They retrieve the hashtags used in those similar tweets.
3. Rank the retrieved hashtags and list

Tf-idf weights are used for calculating the similarity between tweets. The documents corpus used in this tf-idf approach is the set of existing tweets in the system. For each word in the tweet entered by the user, the system calculates tf-idf weight. The sum of weights of the words in the tweet is computed afterwards. Therefore, the final score would be higher if more words in the query are matched in the tweet dataset. For any tweet in the system if the score is above a certain threshold then it is considered to be included in the results. Once the similar tweets are determined, the recommended hashtags are extracted from them, the final step is to rank these hashtags and list them in an order. The authors provide three types of ranking methods:

1. Overall Popularity Ranking – Hashtag candidates are ranked by their popularity in the system. This ranking considers the number of times a hashtag has already been used in the crawled set. The more often a hashtag has been used, the higher rank the hashtag has in the list.

2. Recommendation Popularity Ranking - This basically counts the appearances of the hashtags in the recommended candidate tweets. The more number of times a hashtag is used in these tweets, more popular that hashtag is.

3. Similarity Ranking - Its based on the comparison between the similarity score of the tweet entered by the user and the candidate tweets for the recommendation. The hashtags in the tweets with similar scores make the top of the ranked list.

Once a ranking method is chosen and hashtags are ordered, depending on a selected k value, $top - k$ recommendations are listed for the user. After testing on a self-crawled dataset of 12 million tweets the authors evaluated that their approach has good promising results with 45-50% recall values.

Kywe et al, devised a personalized hashtag recommendation approach[59]. Basically they not only considered the similar tweets, but also the preferences of similar users. Hence, this method includes both content-based and collaborative filtering based recommendation approaches. This approach, considers the hashtags used by the users with similar preferences, and those contained in the similar tweets, then it ranks the hashtags on the union of numbers of the usages of the hashtags gained from both the methods. They used the tf-idf scheme in calculating both kinds of similarity.

For finding the hashtag candidates from similar users, this approach represents a user u_j with weight vector as in equation 2.4, where $w_{ij}$is the preference weight of the user u_j towards a hashtag h_i

$$u_j = \{w_{1j,}w_{2j,}w_{3j,}.....,w_{i|H|}\} \tag{2.4}$$

$$w_{ij} = TF_{ij}.IDF_i \tag{2.5}$$

$$TF_{ij} = \frac{Freq_{ij}}{Max_j}, IDF_j = log(\frac{N_u}{n_i}) \tag{2.6}$$

where $Freq_{ij}$ is the usage frequency of hashtag $h_i$by user u_j, Max_j= maximum hashtag usage frequency by u_j, N_u= total number of users, and n_i =number of users who use $h_i$before[59].

$$Sim(u, u_i) = \frac{u.u_i}{||u||.||u_i||} \tag{2.7}$$

Similarity score between users is calculated using the cosine similarity defined in equation 2.7. Users are ranked based on these scores and top k users are returned upon the request of similar users.

The same tf-idf method is used again in finding hashtag candidates from similar tweets where the user-weight vector is replaced with the tweet-weight vector Once both steps are done, this approach generates a set of hashtag candidates by combining the set of those hashtags used by the similar users and those contained in the similar tweets. These candidates are ranked by the numbers of their usage. The number of usage of a hashtag is the total of the numbers of the hashtag used by the similar users and the numbers of the occurrence in the similar tweets.

A variation of TF-IDF scheme was proposed by **Eriko Otsuka et al** for recommending hashtags[75]. In their method, they considered the association between the hashtags and the terms in tweets. They have two maps to store the information of such associations: Term-hashtag Frequency Map (THFM) and Hashtag-Frequency Map(HFM). Using the data stored in these two data structures, two new weighing factors can be calculated. Hashtag Frequency(HF) is the frequency at which a hashtag appears together with a given term. Inverse hashtag ubiquity(IHU) highlights the hashtags that are tightly associated with a small number of terms. The hashtags associated with a large number of terms are discouraged in the recommendation.

$$hf_{t,h} = \frac{THFM[t][h]}{\sum_{h'} THFM[t][h']} \tag{2.8}$$

where t is a term that occurs with a hashtag h.

$$ihu_h = log \frac{|Corpus_{NH}|}{HFM[h]} \tag{2.9}$$

where $|Corpus_{NH}|$ denotes the number of all terms in the corpus with hashtags removed.

HF score for a hashtag is calculated using equation 2.8 and IHU score is calculated using the equation 2.9. These scores are finally used for ranking hashtags. Test results indicate this method performs better than cosine similarity with kNN(k-nearest neighboring) algorithm. In kNN method, k-nearest neighbors are found that are similar to a tweet and hashtags in these neighbors are used for recommendation. But the tweets that are considered for this method are only user's tweets. **Mina Jeon et al** in [51] used TF-IDF method to identify keywords by extracting them from tweets using TF-IDF. The authors have considered three factors when ranking the candidate hashtags: tweet similarity, hashtag popularity and user interested hashtags. Their main purpose was to help a user organize once own data using hashtags. Their application performed well until top-3 recommendations. In [86] the authors concentrate on recommending hashtags, only to the tweets containing hyperlinks. For the task they consider set of features: similar tweets, similar web pages, domain of the link, names entities in the link page etc. TF-IDF along with cosine similarity score is used in order to find hashtags based on content similarity.

According to Evan Zangerle et al, TF-IDF and cosine similarity metrics are reported as best[103]. However, Wang et al and Godin et al, in their research for twitter based hashtag recommendation systems indicated that these are good for microblogs with restricted lengths and mostly suitable for the content-based recommendation systems[100, 42].

2.4.3.3 Probabilistic topic model - LDA (Latent Dirichlet Allocation)

Topic models are based on the idea that documents are the mixture of topics, where a topic is a probability distribution over words[90]. A topics model is a generative model.In a generative model, a joint probability distribution is defined over a set of observed and hidden random variables. The joint distribution can be used to generate observable random variables in a generative process. Furthermore, a conditional

distribution on hidden random variables can be obtained with the use of the joint distribution and the observed variables. The conditional distribution is also termed as posterior distribution[22]. As a generative model, a topic model can be used for creating documents in a generative process. Meanwhile, the probability distributions of topics can be estimated with the inverse operation of the generative process. Such an operation extracts topics from existing set of documents. A topic model always revolves around word and document distributions progressively.

Over the past decade, there have been many topic models that came into existence. The base for all them is that a document is mix of topics. But the difference falls in when making the statistical assumptions. Latent Dirichlet Allocation (LDA) is the one of the simplest topic models. The intuition for LDA is the same as all the other topic models. But the main characteristic is that, all the documents in LDA share the same set of topics. Each document has a probability over each of these topics. The computational problem for LDA is to observe a set of documents and identify the topic-document and topic-word distributions. These probability distributions can further be used for inferring the topic structure of any other documents. LDA also follows the generative model definition. In LDA, the observed variables would be the words of the documents, and the hidden random variables would be the topics.

Here, we describe LDA more formally as defining the topic mixture for each document i.e $P(t/d)$, with a topic mentioned by a distribution over words i.e $P(w_i/t)$. It is written as,

$$P(w_i/d) = \sum_{j=1}^{t} P(w_i/t_i = j)P(t_i = j/d) \tag{2.10}$$

where $P(w_i/d)$ is the probability of ith word in a given document d and $t_i$is the topic and $P(t_i = j/d)$ is the probability of identifying a word(w_i) from topic j appearing in document d. $P(w_i/t_i = j)$ is the probability of picking a word from a topic j. The topic-document $P(t/d)$ and topic-word $P(w_i/t)$ distributions can be estimated by using a corpus of documents[57]. In general convention, θ denotes the topic distributions and ϕ denotes topic-word distributions. Gibbs sampling algorithm is one of the approaches used for extracting topics from a corpus. It uses an iterative process which stops until the target distribution is achieved. In an iterative round, each word in the corpus is considered and the estimations for the probability of assigning that word to a topic is done with equation 2.11, conditioned on other word tokens in the same topic. From this conditioned distribution, a topic is sampled and stored as a new topic assignment[90].

$$P(t_i = j|t_{-i}, w_i, d_i, .)\alpha \frac{C^{WT}_{w,j} + \beta}{\sum_{w=1}^{W} C^{WT}_{wj} + W\beta} \frac{C^{DT}_{d,j} + \alpha}{\sum_{t=1}^{T} C^{DT}_{d,t} + T\alpha} \tag{2.11}$$

where C^{WT}maintains count of all topic-word assignments, C^{DT}has the document-topic assignments, t_{-i} represents all topic-term and document-topic assignments except for the current assignment t_i, for word w_i, $\alpha$and β are the hyper parameters for the Dirichlet priors, works as smoothing factor for the counts. The estimated distributions can be further used in the operations of a recommendation system.

2.4.3.4 Topic model based recommendation systems

Godin et al, proposed a content-based recommendation system specifically for microblogging services like twitter[42]. They considered the language classified tweets as their dataset. On such a dataset, they used LDA topic model to find its topic-assignment. To prepare the dataset, they performed sampling and pre-processing. In the pre-processing stage, they first removed the URLs, HTML entities, digits, punctuations and hash characters. They also removed tweets not having more than one word and also re-tweets. In next step, they eliminated the non-english tweets from the whole corpus by using the unsupervised language classifier (i.e. a binary classifier implemented using Naive Bayes technique). Then, they performed the parameter estimation procedure by using the LDA topic analysis on the tweets classified as English. The output parameters of the analysis are the distributions of topic-word and of the document-topic separately. Once a new tweet is typed in by a user, the inference procedure derives the topics that cover the tweet as well as the distributions of these topics by running the LDA procedure on the tweet with the topic-word distribution that has been previously estimated. The keywords candidates can be retrieved from those topics of the new tweet. These candidates can be ranked according to their probability in the topics and the topic distribution of the new tweet. So, they perform inference of topics on tweet, count the topic-word values and depending on the required number of hashtag recommendations they ordered values can provide the top-k keywords for hashtags to the user.

Though, this method has the advantage of portability over different lengths of datasets, it only provides with the keywords that a user can use for a hashtag rather than hashtags themselves.

Jieying She and Lei chen, provided with a hashtag recommendation using topic model, briefly referred to as TOMOHA. In this research, authors developed aa algorithm using supervised topic model, modified from Twitter-LDA[88]. In this

work, each topic is associated with a hashtag distribution, and the associated hashtags are assumed to be the topic labels of the topic.. It is assumed that a tweet covers one local topic and the back-ground topic as well. The generative process of a tweet with the TOMOHA model is as follows. Once a user enters a tweet, first, a local topic is selected. Then each word of the tweet is sampled either from the local topic or the background topic Differently, the generative process of the TOMOHA-Follow considers followees of the user. While a user is generating a tweet, he/she samples a topic either from his/her local topics or from those of its followees.

The parameter estimation process of the models are trained locally by passing in the dataset through multiple iterations via a certain number of processors in parallel. The algorithm performs really well and better than TF-IDF scheme nonetheless, the hit ratio is worse for both of them when considered for top-5 recommendations than top-10.

Yuan Wang et al also proposed a hashtag recommendation system using topic model analysis [100]. They developed a hybrid function (as described in equation 2.12) which ranks the hashtag candidates input from topic semantic analysis and also from user preferences.The user preferences are calculated based on a collaborative filtering technique. In equation 2.12, $P(h|d)$ is the probability score of the relevant hashtag for a microblog-message d, $P(h|\mathrm{u})$ is the probability scores of user preferences.

$$Score(h/u, d) = \lambda \,.\, P(h|d) \;+\; (1 - \lambda) \,.\, P(h|u) \tag{2.12}$$

For calculating relevant hashtags, instead of considering single message, they included the aggregated values of a set of similar messages. So, for building the profile of a hashtag h, all the messages containing that hashtag are studied. A vector D_h(the profile of hashtag d) that describes the distribution of a hashtag over terms is prepared for each hashtag. For computing the relevancy score of the hashtag against the current message, the profile vector is passed in further to the cosine similarity function as in equation 2.13.

$$P(h|d) = \frac{D_h.\, d}{||D_h||.\, ||d||} \tag{2.13}$$

User preferences are stored in three representation matrices of content, hashtag and topic, respectively. Using cosine similarity they get the users with the preferences close to the user that is looking for hashtags. Using this similarity score, they find hashtags that the user might be interested in. The hashtag candidates gained from above steps are further combined according to their scores calculated by Equation

2.12. In the experimental results of their study, we could observe that a scheme considering topic semantics is superior to TF-IDF scheme. Furthermore, by considering the preferences or interests of similar users, their approach outperforms those based on content-based methods only.

Jianjun Yu and Yi Shen, also worked on a novel hashtag recommendation by considering three features hashtag popularity, hashtag content similarity and hashtag time sensitivity[102].They have used LDA in measuring the similarity between the user's posts and the posts with hashtags. Different from the other methods, their approach consider the con the hashtag life time as one of the factors . However, they haven't provided any experimental results for us to understand how far their approach was successful. Another work by **Qi Zhang et al**[105], also uses topic model for getting the topic structure from the microblogs. They also include user's personal interests and the time factor as to the importance of hashtags (global trending hashtags considered) for extracting topics. Since they considered the microblogs to be restricted in length, they have made assumption that each blog will have covered one topic. Huang et al[47] also uses LDA for similar topic model.

LDA or any of the topic models can be used for texts with varied lengths[83]. It is also good for the case considering users who are new to an application or having fewer connections[78].

2.4.3.5 Other techniques

There have been many solutions for hashtag recommendations. Till now we have discussed the major techniques used by most of them i.e TF-IDF and Topic models. In this subsection, we discuss briefly on the other strategies.

In [91], authors compute the discriminative term weights for each term to evaluate the relation between that term and hashtag. A term more popular with a hashtag is associated with that hashtag. Once all the list of terms for each of the popular hashtags are associated, these associations can be used further when recommending the hashtags. **Abhineshwar Tomar et al**, in their research on hashtag recommendation used distributed word representations[92]. They form the word distribution vectors and pass these as training sets for Deep Forward Neural Network. These word representation help in suggesting the words that surround the content. This model takes into consideration the temporal sensitivity of hashtags. Bursting hashtags another new hashtag prediction algorithm, that suggests the hashtags that become part of trending topics was proposed by **Shoubin Kong et al**[55]. They use the strict timeline windows like 24hours period for identifying the hashtags which are related

to bursting topics or any news or events.

2.4.4 Peer-to-Peer Recommendation Systems

In this section we give a brief overview on the Peer-to-Peer recommendation systems implemented until now.

Most of peer-to-peer systems developed are for file sharing or digital media sharing. Since the arrival of e-commerce in P2P systems, many recommendation methods have been proposed. Majority of these approaches concentrate on either content-based[54, 27] or collaborative filtering based methods[73, 96]. A very few of them take both methods into consideration[36, 63]. Bulk of recommender systems have been implemented for client-server systems. But they face two major problems: one is data sparsity and scalability. Data sparsity refers to case when there is more of user data and less of information with the data for use during the computation of similarity scores. Also, with the increase of user and data in the servers, the amount of computations to be performed also increase dramatically[97]. P2P networks on the other hand need to be aware of different factors like dynamic network topology, user privacy, trust and encryption mechanisms[93].

In the e-commerce or file sharing P2P systems, user ratings and feedback are considered as the input data for measuring the interests and similarity scores[54]. And only the recent ratings are considered for measuring the similarities, as they will portray the current interests of the users. Most of them implement event driven push strategies i.e whenever a file is added by a user or product is purchased by the user all of the information is updated. In any P2P recommendation system, storage of the recommendation data is not located with all of the peers. Also, peers will be joining and leaving the network abruptly. Choosing neighbor sets of peers for having the information is another big task. Cosine similarity metric is used by larger part of the systems for determining the similarity scores. Feng Guo and Shaozi Li[45], in their recommendation systems used an agent at each peer for triggering recommendation system, each time the user starts searching for files. They even clustered peers into overlapped groups, with each group having an authority peer. A peer is awarded authority role, when majority of the users recommendations are being considered by others. This authority peer acts as the trust central for a cluster.In [97], the authors consider the records downloaded by a user for calculations of interests. Ming-Hung Chen et al[30], in their music recommendation system considered three types of factors for their collaborative filtering mechanism. These factors include profile

information, content history and also content is associated with attributes which help when recommending. One of the studies even used ontologies for developing a semantic based recommender system[40]. Overall, as per our observations most of the P2P systems prefer collaborative filtering methods for recommendations. Nearest neighbors and trust factors are major motivations for making the recommendation systems more fruitful for the users.

Chapter 3

Content Management, Architecture and Components

In this chapter we present details on the architecture and components of our peer-to-peer social networking application and the content management modules. We give a detail overview on content categories and their respective data structures and distributed operations.

3.1 Overview

In the previous chapter (Section 2.2 and Section 2.3), we examined in detail on privacy issues with social networks and how peer-to-peer networks could be used to improve the security and privacy of social networks. Wang Sen one our teammates worked on developing a decentralized P2P architecture for his thesis in 2014[98]. The main purpose for this architecture is to achieve a stable e-commerce platform with P2P infrastructure. This application also consisted of some minimal social networking features like adding friends, search friends and chat with friends. We absorbed the same pattern and converted the application from e-commerce platform to a complete social networking application. The architecture is inspired from BestPeer++[28, 29], a large scale data processing platform. BestPeer++ enables sharing of enterprise data across multiple nodes in a server farm. All of the nodes in server farm are structured using BATON P2P overlay network[49]. In this section, we briefly mention about BestPeer++ and BATON.

BestPeer++[28] is an extension of the BestPeer architecture[14], it provides a common sharing environment for corporate networks, with the use of cloud computing. BestPeer++ gives an outstanding performance results due to the incorporation

of features like HadoopDB implementation and support for MapReduce in distributed query processing. This gives BestPeer++, the capacity for handling large-scale data and query processing. Main purpose of BestPeer++ was to provide a common platform for corporate networks to share data. Since corporate data is highly scalable both in terms of users and amount of data, a large-scale network, with strict privacy considerations, high availability and data consistency is needed.

BestPeer++ is a cloud service model. Any business that wants to use the service, just has to register themselves and create a BestPeer++ instance, into which they can export data for further processing. This also gives an option for pay-as-you-go query processing model with the help of cloud computing. There are two main components in BestPeer++ - core and adapter. Adapter has two parts, one is an interface to the service and the other part contains adapters which implement this interface with the help of service provider APIs. The core component consists of query processing and the P2P overlay for serving responses to the queries. There are two kinds of elements in core, bootstrap peer and normal peer. When a business creates an instance, a database server is assigned to that particular instance. This server is then included into the structured P2P overlay arrangement, along with all the other servers. So a normal peer here is a server of a particular business instance. Figure 3.1 shows the components of the BestPeer++ architecture.

Responsibilities are divided between bootstrap peer and normal peer. The whole network has a single bootstrap peer. This is the server through which normal peers try to join the network. It works like an administrator for the network. Some of the tasks performed by this bootstrap peer are - auto scaling (when an instance exceeds its storage or to perform load balancing), auto fail-over (when a node in the P2P overlay has failed and had to be removed from the network) and the main task of node joining/leaving. For a normal peer, primary effort goes in data loading and indexing. It also does the schema mapping, query processing and execution, along with data loading. When a new business is added to the network, data is loaded from the corporate production to the instance. When this process is being done, normal peer tries to do schema mapping i.e mapping the local business schema to the global peer schema.

All the normal peers are organized in P2P overlay called, BATON(Balanced tree overlay network) [49]. This is the crux for BestPeer++ functionality. It provides the interface for node joining, leaving, adding or removing data etc. It arranges all nodes in tree structure. BATON allows for processing both exact and range queries. Each node maintains two range details - 1. About the range of data index

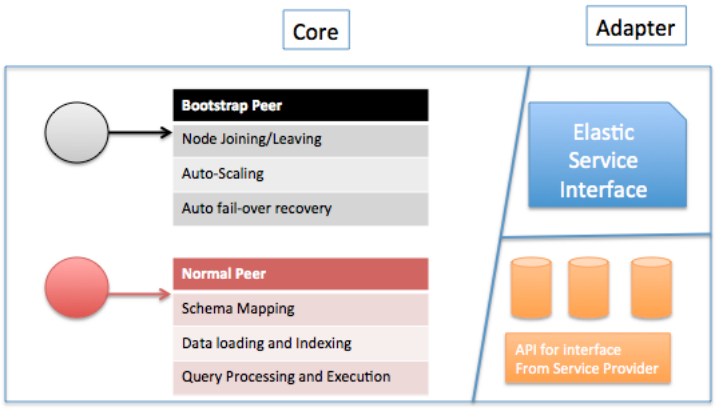

Figure 3.1: BestPeer++ Architecture components

it maintains 2. Information on the range its subtree maintains. It has two kinds of load balancing techniques when arranging data in the nodes. When a node is loaded, it balances with the help of the adjacent nodes which are not highly loaded yet. If there are no nodes that are loaded low, then a global shift of non-adjacent to the overloaded nodes region is made. BATON also provides for three types of indexes - table, column and range. BestPeer++ also provides for role-based distributed access control. A local administrator from the business is assigned to add in users into the systems, and assign roles. Whenever a query reaches a normal peer, depending on the user role from whom the query is passed, the results are retrieved. Queries which need large-scale processing are passed on to the MapReduce support structure. For maintaining consistency with respect to the data, BestPeer++ maintains snapshot, which it compares with the new snapshot always just before the data loading is done. When a node fails, all the queries are held up until the backup is restored on to the system. With all these features, cloud computing, database and P2P overlay support BestPeer++ is highlighted as a better data sharing application than any other P2P data sharing systems available. Hence, we choose the same for our P2P social networking application.

3.2 System Architecture

BestPeer++ is a two-layered architecture. In the current P2P application we have three-tier architecture - bootstrap peer, server peer and client peer as in Figure 3.2. Wang Sen has upgraded the functionalities with operations like product and store management, adding/deleting/updating product information and advertisement module, which are useful features for e-commerce platform. The same archictecture

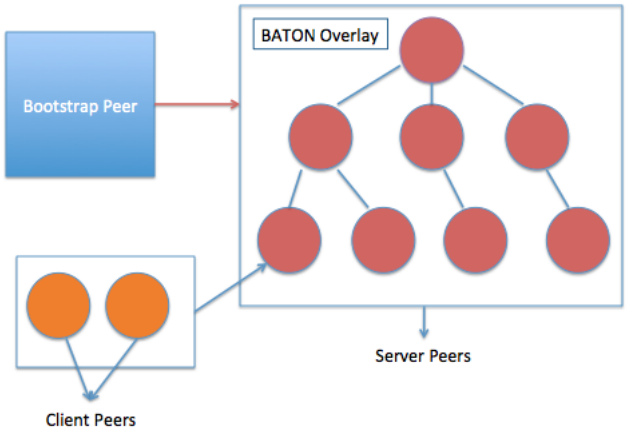

Figure 3.2: Our P2P Architecture

is used for building the social networking application. It is complete Java based application. We use PostgreSQL open source relational database for storing all the data. Further in this section we describe each of the components and their respective operations.

3.2.1 Client Peer

In our application, each user whoever wants to join the network need to use a client side user interface on their PC or mobile device. This user is called the Client Peer. We do not store any data on the client side. All of the data pertaining to a user is stored in the database on the server side. The User Interface helps the user in interacting with the application.

In Figure 3.3 below we brief the purpose for each of the elements in client peer is mentioned[98]:

- **Online Shopping Management:** Shopping cart, advertisement module and product recommendations implementations is managed via these modules.
- **Personal Store Management:** Clients can use the user interface to communicate with the server and manage products addition/deletion in their personal stores.
- **Peer Maintainer & Peer Event Management:** This has the complete information as to the client peer and the other peers a client might be connected to. It consists of user store information, physical information as to the friends.

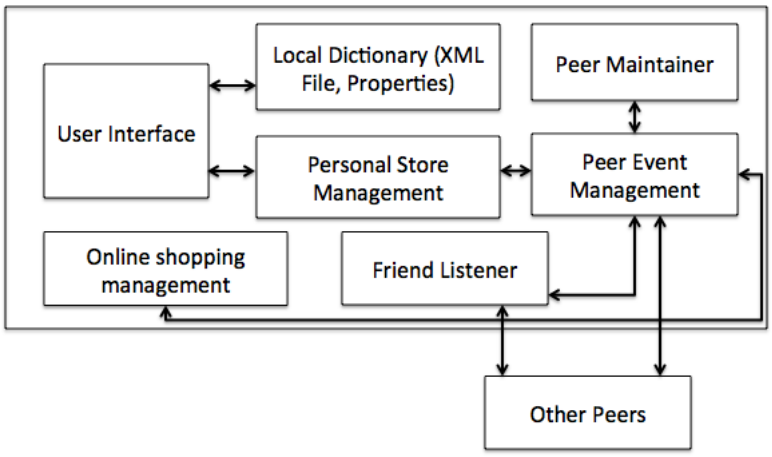

Figure 3.3: Client Peer Node

- **Friend Listener:** This implementation waits for any request from clients friends or any other peer trying to communication with the client.

3.2.2 Bootstrap Peer

Bootstrap peer in the current architecture, has the same administrator role as in BestPeer++. Single bootstrap peer node accounts for the health of the whole network. Monitors the node joining and leaving. The auto-failover and auto-scaling supposed to be in the bootstrap peer have not been implemented in our system. Users need to register and login via the bootstrap peer each time they connect to the network. Apart from the components in BestPeer++, there were some additional responsibilities added to the bootstrap peer for both e-commerce and social networking purposes. Below we itemize purpose of each of these operations:

- **Advertisement Manager:** Information as to the advertisements belonging to the most popular products is stored in bootstrap peer. An election algorithm or advertisements of new products in store is communicated to the bootstrap peer by the server peer.

- **User Profile Storage:** User login and registration details are store in bootstrap peer. Information as to the profile of the user like hobbies and privacy settings preferred by the user which are major for the social networking functionality are preserved under this module.

- **Store Profile Storage:** Copy of store information is preserved with the bootstrap peer which is used for checking the correctness.

- **User Friend Storage:** Friend requests for clients are saved in this component.

3.2.3 Server Peer

Client data is stored in the Server Peer nodes. Each server peer is responsible for more than one client node at a time. All of the server peer nodes form the BATON overlay structure. User login information is stored even in the server peers. Major concerns for the server peers lies in the management of P2P overlay structure and user data. Each server is responsible for clients within a particular URI (User Resource Index) range.

Server Peer is the alias for Normal Peer in BestPeer++. Most of the functionalities in normal peer have been imported to server peers with some updates. The schema mapping module was discontinued as all the data exchanged in the application has the same mapping. Data loader is used during the data retrieval process. Data Indexer is major for the BATON overlay network, as each of the data stored in server depends on the range. It also helps during the forward and lookup requests. For the query execution, we used JPA instead of pure SQL language[98]. BATON tree node information is also stored with the server along with the physical details of bootstrap peer.

3.3 Content Management Components

Until now we have discussed the architecture for our P2P social networking application. Most of this included work done by Wang Sen for this thesis[98]. From here on, we concentrate on the contributions for this thesis. First and foremost of which includes the content management implementation for the social networking application. In our effort to modify the e-commerce platform to social networking application, we needed to include various categories of user content. This was done on observation of the most frequently used features in the current social networking applications.

The content management components consists of firstly user account information which included personal information like name, email, hobbies etc. The other part of content management comprises of the user generated content. This includes posts, comments, blogs, articles and attachments. Users can also add other users as friends who are already registered in the network. Batch registration is also a new feature added to this component. Each of the friends of a user can be given different levels of access depending on the privacy rule. We provided privacy settings module both

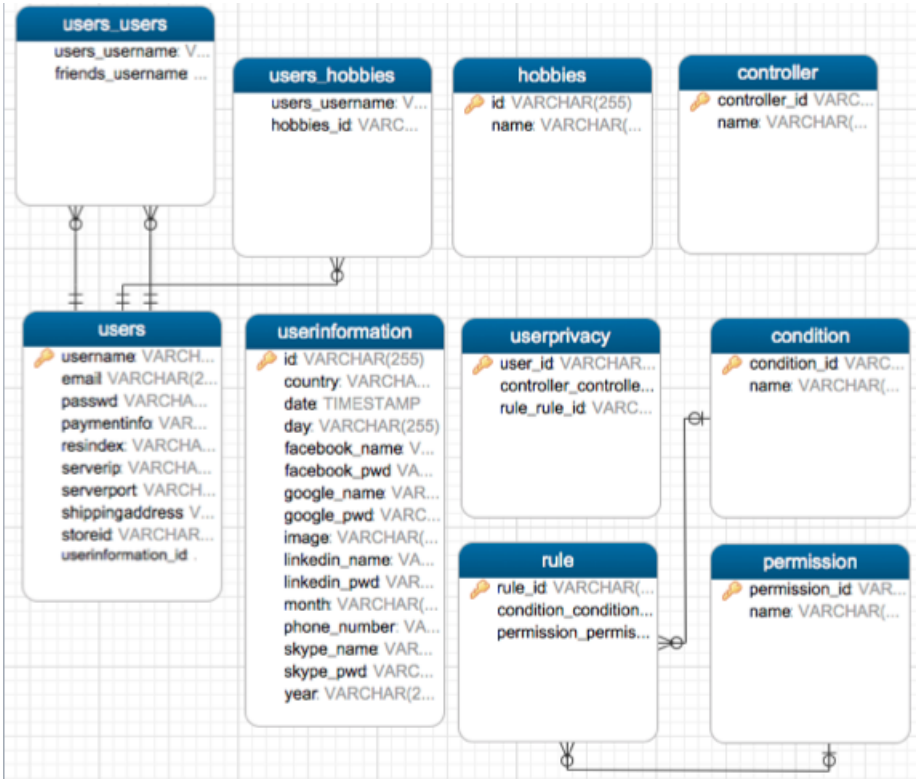

Figure 3.4: User Profile in Bootstrap Peer

at user profile level and user generated content level. The final component is the notification and newsfeed module. Notification consists of the middleware that is used for sending clients notifications regarding any friend requests or content updates. Newsfeed shows content from the network displayed in chronological order.

All of the content except for the user profile module is managed in the server peer. In this section, we specify in detail the database design and operations performed for each of the modules.

3.3.1 User Account Information

3.3.1.1 Database Schema

User profile information is stored both in bootstrap and server peer. Figure 3.4 shows the database model for User Profile in bootstrap peer and Figure 3.5 shows the database model for User Profile in server peer. Below we specify objective of each table.

- **Users:** Contains high-level user information captured during the registration process like username, email, password and address etc. Username acts like the

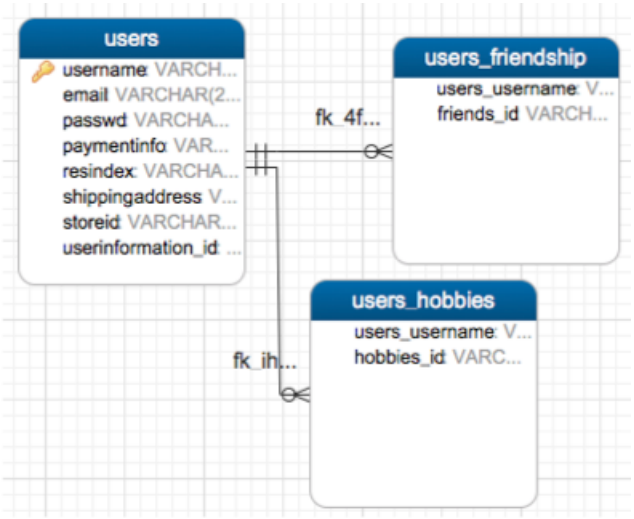

Figure 3.5: User Profile in Server Peer

unique id to identify each client in the network.

- **UserInformation:** This table contains detailed user information which client might add either during registration or after logging-in into the application.
- **UsersFriendship & Users_Users:** As and when a user adds a friend to his profile, the username value of the friend is added to this table.
- **User_Hobbies, Hobbies:** Each user can have multiple hobbies. These two tables store the relevant information.

3.3.1.2 Operations

User Registration/Login: When a client is using the application for the first time, he/she is supposed to register in order to join the network. User Registration flow is described by means of sequence diagram in Figure 3.6. The process starts with client sending registration request to the bootstrap peer. Bootstrap peer then needs to find a server node and assign it to the client. This is done by using the BATON tree structure. Once a server is chosen for a client, a client join request is forwarded to the host server. This operation ends with server sending the physical information to the bootstrap peer. Once a user has registered, an email is send for verification of the user credentials with a password text automatically generated by the system. For user to login to the system, this code is needed. After entering this code and resetting the password, the users will be taken to their respective home screens. We also have batch registration option made available, for scenarios like when a user

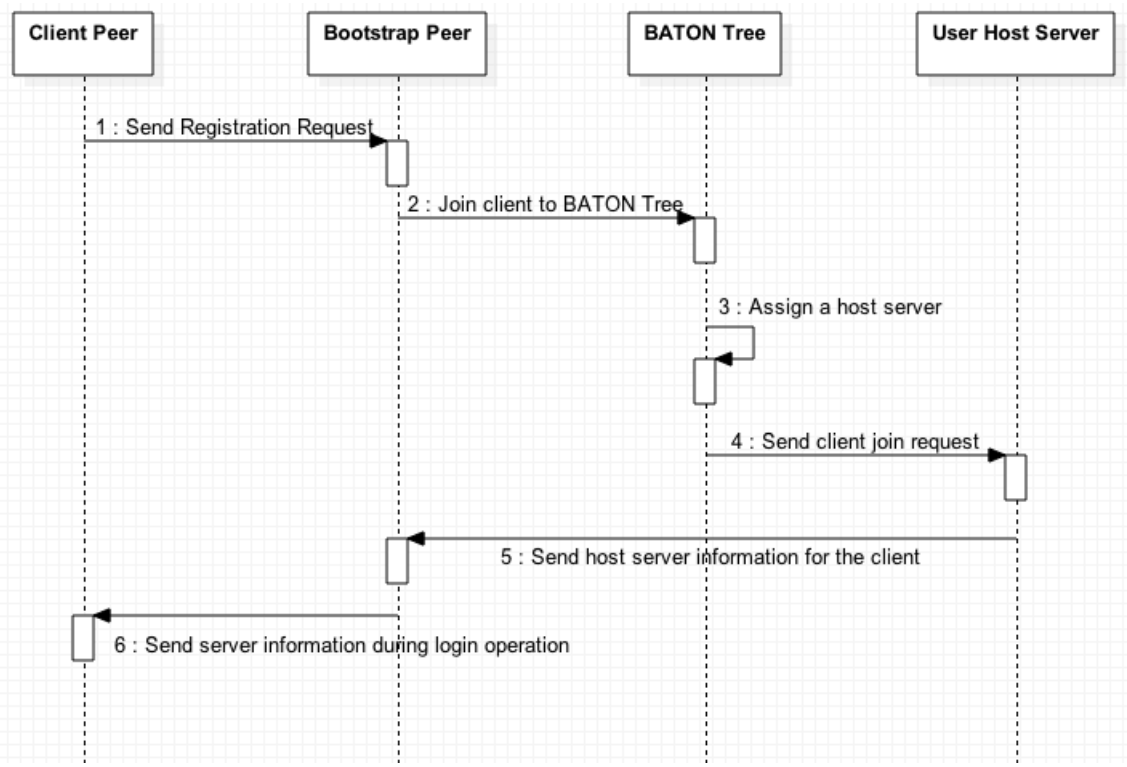

Figure 3.6: Sequence Diagram for User Registration

wants to invite multiple friends to the network space. All the client needs to do is add all email addresses of the respective friends in excel book. Next the client can upload the file using the Friends Space. Automatically generated password is sent to the user who has uploaded the file. Clients can use this password, login to the application and reset password on log on. User login also goes through similar steps except for one step at which instead of assigning a new host server, the clients registered host server is found from bootstrap peer.

User Privacy Settings: We have a privacy settings module, both at the profile level and content level. Each of the content type whether it is post, blog, article or profile attribute is termed as Information Block. Each block, for each user can be assigned a Rule. Block rule table stores the information block and rule information for a particular user. Each rule has two elements. One is Role, which identifies the scope for the information block in a rule. For now we have three types of scope defined:

1. Self - Items under this category are available for viewing and editing only to the users who created the items.

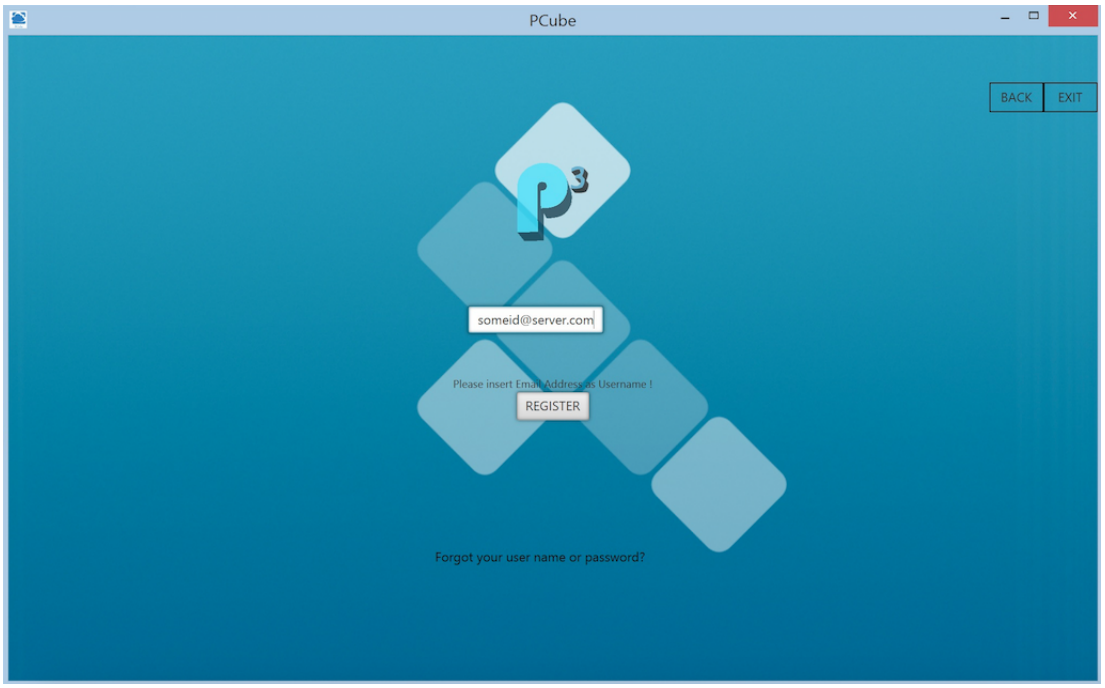

Figure 3.7: User Registration Graphical User Interface

2. Public - Items under this category are available for everybody in the network for whom the profile of the user is available.

3. Friends - Items under this scope can be seen and responded to, only by friends of the user who has created them.

Access Privilege is another element of rule. It defines the authorization level for a particular rule. For now we have only one kind of access privilege i.e "Publish" which allows for both read and write access to the role.

Suppose a user publishes a post in his profile, if the user assigned the post rule as "Self", then this post is not available to any one except for the user himself. The user can update or delete the post without any restriction. In another situation, if the post was assigned a rule type as "Public", then immediately a notification is sent to all of the friends of the user and also this post is visible for anyone who is available on the network. Additionally, for this post other users can add comments, delete their own comments and even update them. Similar scenario is seen when "Friends" rule is applied for the post. Except for the condition that, the post is available for only the friends of the user.

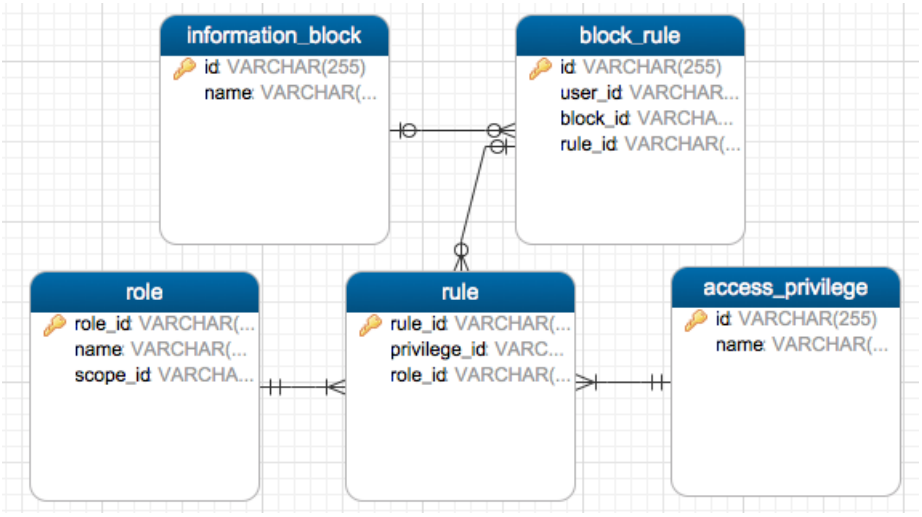

Figure 3.8: Privacy Settings Database Model

3.3.2 User generated content

3.3.2.1 Posts & Comments

A post is text with multiple strings. There is no restriction to the length of the post. When users add a post they can set the privacy rules: public, private and friends only. A post can also contain attachments. Any file type can be added to a post as attachment. Once a user adds a post, depending on the privacy settings added to the post, other users can view or comment on the post. Comments will be added in chronological order under a post. There is no restriction as to the number of comments a post can have. A comment also can have attachment added along with the text. Figure 3.9 shows the database model diagram for posts and comments.The operations that can be performed on posts and comments include: add, delete, update and get (by userID for posts and postID for comments).

3.3.2.2 Blogs & Articles

A blog is a collection of article. Each user can create any number of blogs. Each blog can have multiple articles. Each article has to be assigned a category. When creating blog or article user can set the privacy rules. Privacy rules of articles are prioritized over the ones of blogs. Articles can also have attachments. As of now we have not restricted the number of attachments for each article. Figure 3.10 shows the database model for blogs and articles. Operations on blogs and articles are similar to the ones for posts and comments. They are add, delete, update, get by id for blogs and get by id, blogid, and userid for articles.

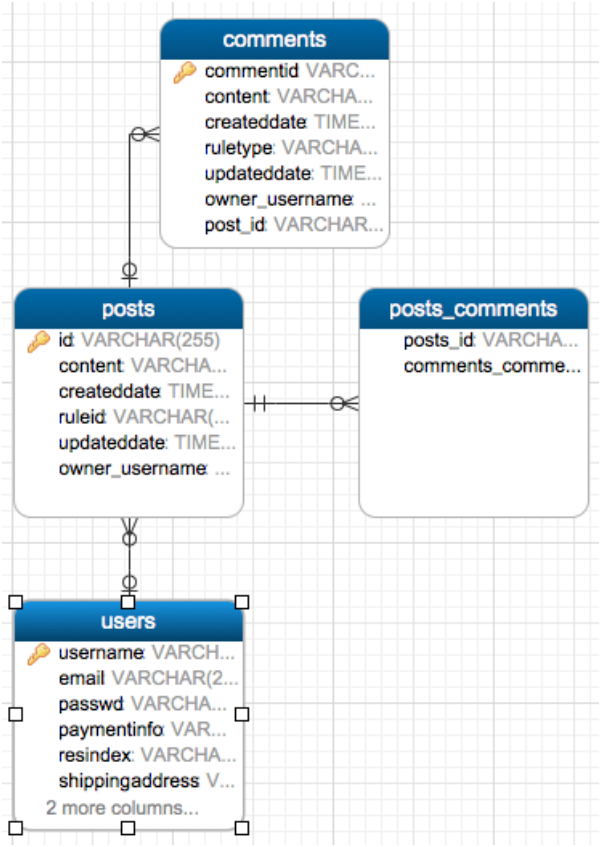

Figure 3.9: Posts Database Model

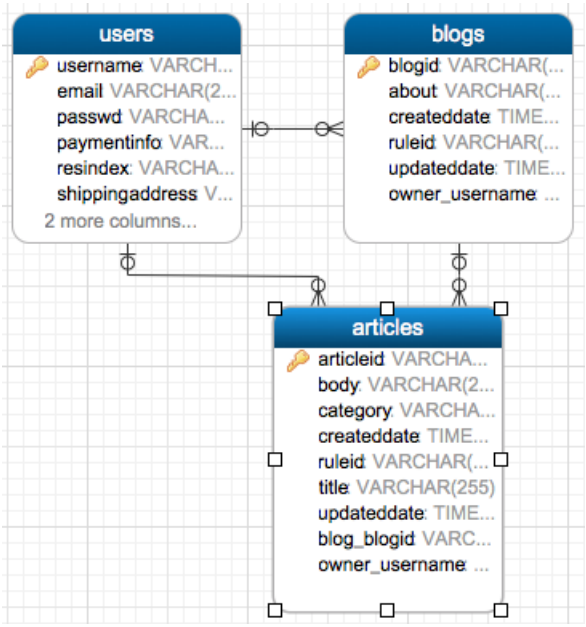

Figure 3.10: Blogs and Articles Database Model

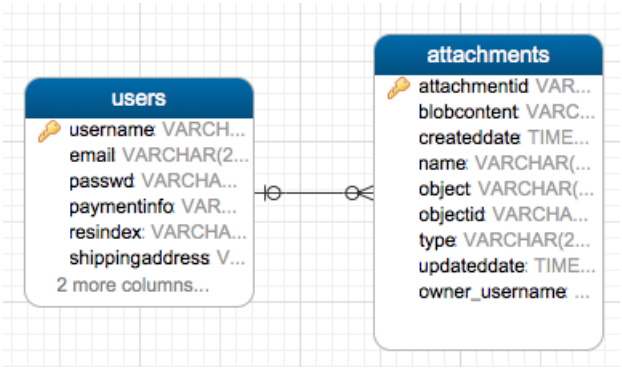

Figure 3.11: Attachments Database Model

3.3.2.3 Attachments

An attachment can be of any file type. We could even attach an audio file with size restricted to 10MB. Each of the content types described before can have multiple attachments. There is no restriction on the number of attachments a content type can have. Figure 3.11 is the database model for attachments. In the attachments table, the content is stored in blob format. Object column points to the type of content type. It can be post, comment or article. ObjectId stores the unique identifier of the content type. Operations for attachments are add, delete, get by object type and id.

3.3.3 Content from Social Network

3.3.3.1 Notification Requests & NewsFeed

For a social networking application, newsfeed component lists all the updates from the user's friends. Updates comprise of any changes in content types discussed until now. Adding new post by a friend or adding a comment to any of the already existing posts might cause the newsfeed component to be triggered. Newsfeed has two parts in it. One is NewItem and other is the Notification middleware. Each of the content type being generated by the clients are added as NewsItem and a category type (either content or request) is assigned to that item. Request category is assigned to the NewsItem if a clients sends a friend request. Once an item is added the notification action is triggered for all users for whom the item is available. Notification action waits until a response is obtained from the user if it is a request. This notification module is used as means to inform about the clients updates to his/her friends. In Figure 3.12, we give the database model for newsfeed and notification.

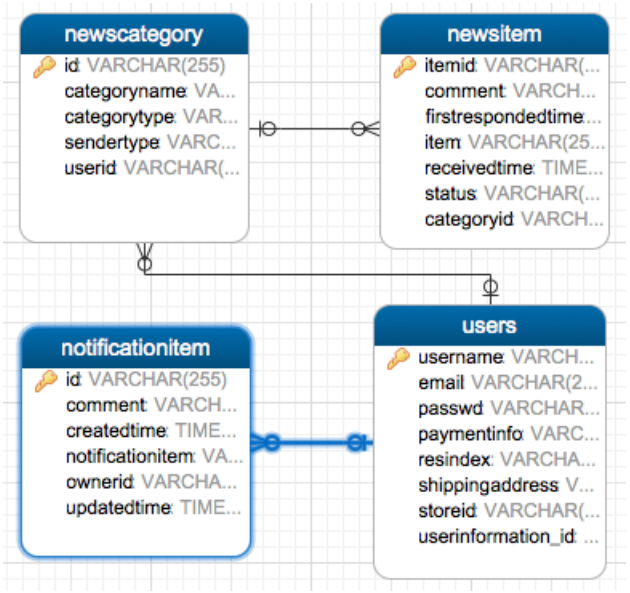

Figure 3.12: NewsFeed Database Model

3.3.4 Sequence Diagram for Operations

In this part, we present the sequence diagram common for all of the content type operations. In Figure 3.13 we show the sequence diagram for add request, common for any of the content types (post, comment, blog, article or attachment). First the user triggers this request by using user interface. The request is sent to the server peer. Server peer adds this request along with the data to be added in a message format and forwards it further to the BATON tree. In BATON tree, a search for the current clients server node is done. Once the server node is found, the data is added to the server. A notification is fired towards both the client's host server and the friend's host server depending on the rules set along with the request. Finally, a request success message is directed to the client.

All of the operations for the content types follow similar pattern except for the get requests instead of step 4 (storing the data), data retrieval is done and the same is forwarded further.

3.4 Comparison Analysis

In this section, we compare our current P2P social networking model with some of the previously mentioned P2P social networking applications in Section 2.3. Table 3.1 gives a high-level overview on feature comparison between Safebook[34], Pesca[82], PeerSON[85] and our application.

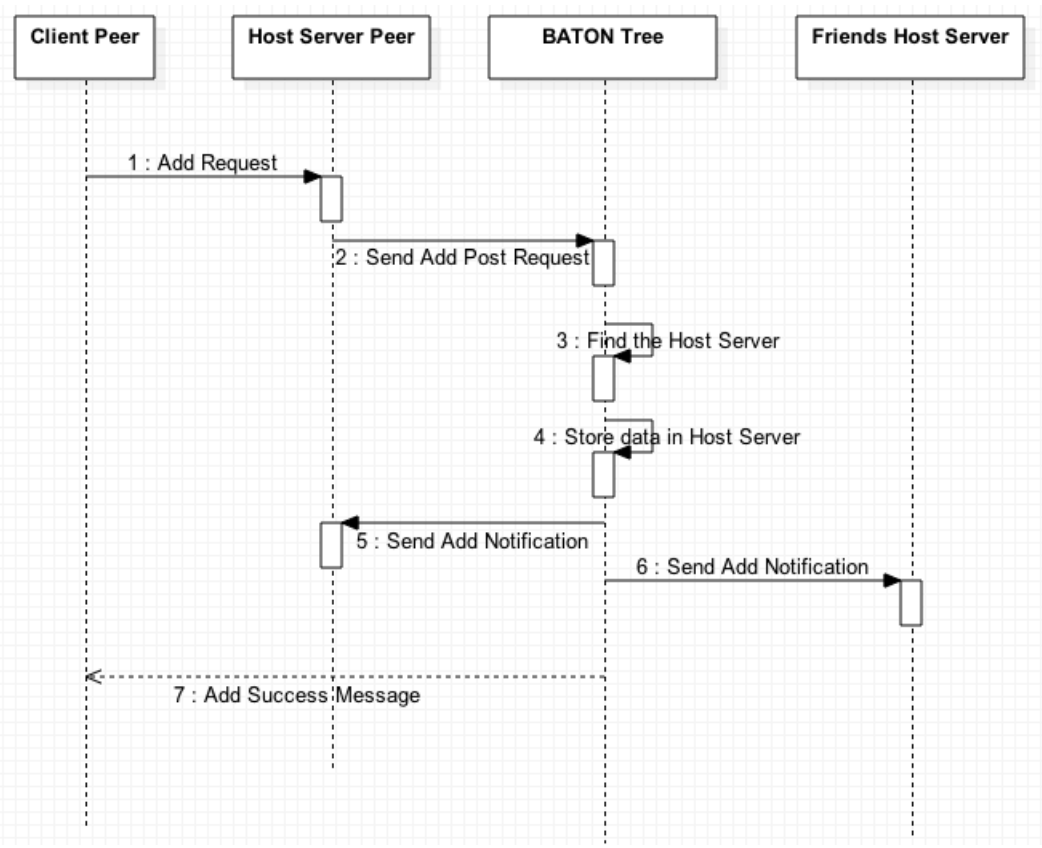

Figure 3.13: Sequence Diagram for Content Posting

Most of the applications developed before have DHT usage for storing physical information with regards to peers. Having DHT instigates additional overhead of maintenance and traffic. LifeSocial.KOM[43] is a plugin based P2P social networking application. This also has the structured P2P overlay implementation. The disadvantage with this implementation is the whole application load is on the nodes in the network. My3[71] another similar application uses the trust relationships between peers in providing the functionality of a decentralized storage infrastructure. But this again involves sharing control of data with a trusted set of users, which is the kind of situation we are trying to avoid. Vegas[38] is a P2P social networking application which concentrates mainly on security of the users. In their attempt to restrict the access to the social graph for unauthorized users, the encryption mechanism implemented requires the management of three times more keys than general public key encryption algorithm. Also, there is an expectation to reduce the number of relationships a user might have. This would restrict the users need to maintain relationships across the network, which would daunt the main motto of a social network.

Our application on the other hand makes use of the bootstrap peer in-order to manage the peer information. Any user who wants to use or join the network has to install the java-based executable file on their device. Not being a web application might initially be a trouble in picking up users. But otherwise, when it comes to privacy, security and content-wise options our application could do better because of the content categorization and the overlay network structure. Performance evaluation in a real-time environment is yet to be done. We plan to release a version for students of certain classes in the university for the purpose of evaluation.

Features	Safebook	Pesca	PeerSON	Our Application
Structure	Matryoshka for nodes (mirrors, entry points), DHT substrate for lookup service	DHT in user's storage space	OpenDHT based lookup service	BestPeer++ and BATON overlay network
Objective	Decentralization and trust between friends	Data privacy and availability	1.Fully distributed and Access control through encryption 2. Prevention of impersonation attacks	1. Fully Distributed 2. Privacy at very granular level 3. Availability
Node Identifier	Trusted Identification Server (TIS) creates the node ids	Global Unique Identifier (GUID) generated from hashing email address	Global Unique Identifier (GUID) obtained from hashing email or a public key	Email id used to identify users
Account Creation	Based on invitation only	Need to register with PESCA application	Peer needs to connect to the DHT lookup service for getting other peer information	Install application and register with the help of bootstrap peer.
Encryption Mechanism	Public-Key Infrastructure (PKI) for encryption	Broadcast encryption for encrypting data	Threshold based encryption using Byzantine fault-tolerant protocols	No encryption mechanism implemented
Disadvantages	1. Limited guarantees on availability 2. TIS is a centralized structure 3. Access to unsolicited friendship requests 4. Management overhead for DHT-weaker trust model and additional signal traffic 5. Performance yet to be evaluated	Possible leakage of data when using traffic analysis techniques.	1. Works only with other PeerSON enabled devices 2. Management overhead for DHT-weaker trust model and additional signal traffic. 3. No keyword search 4. Users can leave without logging out	1. No encryption implemented for data 2. Need to install java based application on PC or mobile device for usage

Table 3.1: Comparison of P2P Social Networking Applications

Chapter 4

Hashtag Integration

4.1 Overview

We have reviewed tagging using hashtags in on-line social network platform, and the techniques used by the recommendation systems on those hashtags in Section 2.4.1. Most of the social networks like Facebook, Twitter, LinkedIn etc have hashtags features included in their applications. Hashtag is a tag added to any string of text posted in a social network preceded by "#" symbol[80]. It allows for categorizing or organizing the content. Once the content is published on to the network, the tag connects to all of the other content which have used the same tag. On click of this tag, a user is shown all of the content with this tag in a chronological order. Various problems have been researched around hashtags. For example, as discussed in our background, numerous hashtag recommendation systems have been proposed[59, 42, 88, 100, 102, 105], hashtag classification based on sentiments[35, 99], hashtag usage analytics[59, 104] etc.

In this chapter, we propose a hashtag recommendation approach specific to our P2P social networking application.We discuss the proposed approach, the database schema, the operations performed on hashtags and the algorithms for the suggested approach.

4.2 Hashtag Recommendation Method

4.2.1 Proposed Approach

The proposed hashtag recommendation approach lists out hashtag candidates for a content entered by the user. If no related hashtags are found, this approach may

suggest the user with the hashtags that have been used previously or with those related to the user or to the content. The approach also advises the user with some keywords for creating a new hashtag. We adopt a hybrid recommendation system for our social network platform considering both types of recommendation: content and collaborative filtering approaches. The hashtag recommendation systems that we have reviewed in the Section 2.3 have lagged in two issues. First, they use only one of the recommendation approaches. In the case that an approach is chosen, a major part of hashtags the user might be interested in is being omitted. For example, the content-based techniques might not include some of the tags being created by similar users or the friends in the suggested tags. Similarly, the collaborative filtering based techniques might neglect the tags related to the posted content or those popular in the overall system.. Second, as per our knowledge, none of the ideas reviewed till now have given a user an opportunity to choose the recommendation method he might be interested in.

Hence, considering these drawbacks in the previous research, our approach contains several recommendation modes. The users may control the recommendation system by selecting one or multiple modes. They receive the candidate hashtags recommended by the selected modes. These modes are classified into the following categories. The categories considered are:

1. Global content common for all of the users
2. User preferences evaluated based on their content previously added
3. Hashtags created by users with similar preferences as current user
4. Hashtags created by the friends of the user and are related to the users content being created
5. Overall popular hashtags in the whole social network platform

Also, in the case that any of the methods returns zero tags, the proposed approach even recommends with keywords based on the chosen mode.

4.2.2 Implementation

Unlike twitter which has restricted the lengths of the text for its tweets, content with different types in our application can have varied lengths without any restrictions. In such cases, as reviewed in Section 2.4.3.2 topic model for recommendation systems is a better technique. For our approach we considered to adopt topic analysis technique to evaluate the content similarities and user preferences on content. We further chose Latent Dirichlet Allocation(LDA) process using Gibbs Sampling method for topic

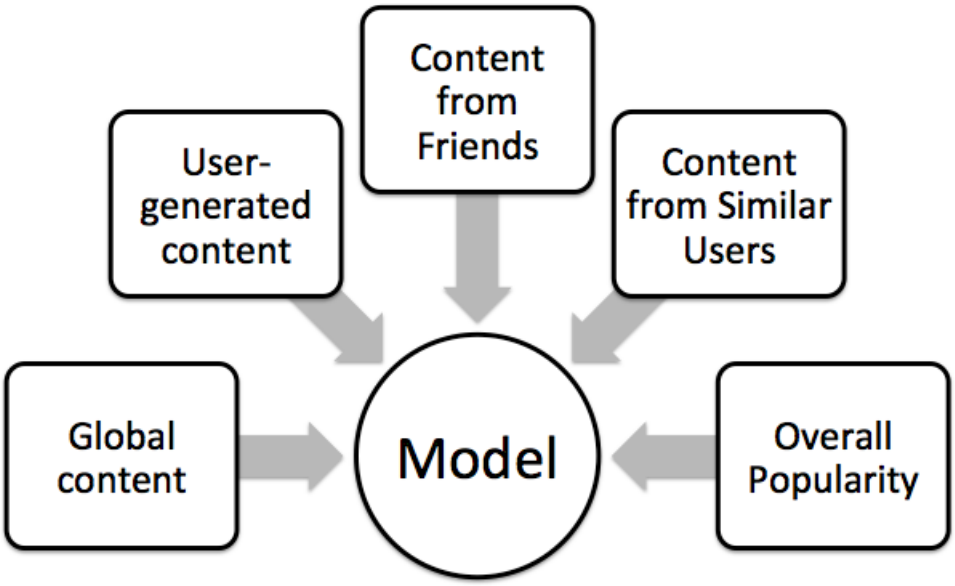

Figure 4.1: Categories for Recommendation Modes

classification. Most of the research discussed using topic models have considered only topic-word distributions. The proposed approach goes further to the next step and extracts "topic-hashtag" probability distributions.The LDA process is done in three phases as shown in Figure 4.2.

Training or Base LDA Model: In the training phase, random content collected across various networks is passed to the procedure implementing the Gibbs algorithm. The procedure estimates the initial topic-word and document distributions of the Base LDA model.The documents in this situation refers to any single post, comment or article.

Estimation or Recommendation Mode: This is phase at which content passed for LDA procedure differs from each of the categories mentioned before. For each mode selected mode, we pass in the content and generate the updated topic-word, document distributions along with topic-hashtag distribution. For each of the topics and documents, we calculate hashtag distribution over documents can be calculated using equation 4.1 where d is the document, h is the hashtag and t is the topic.

$$P(h/d) = \sum_{i=1}^{n} P(h/t_i)\, P(t_i/d) \tag{4.1}$$

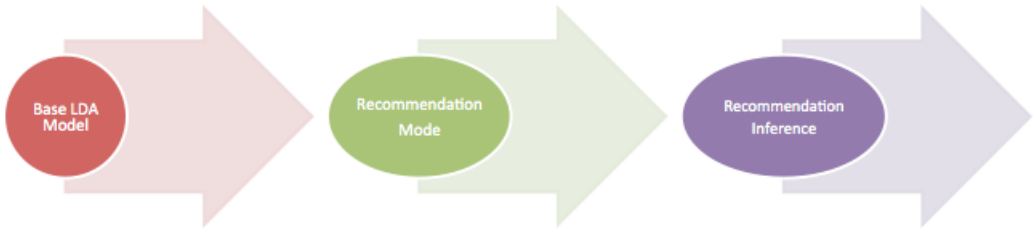

Figure 4.2: Steps in LDA Processing

Recommendation Inference: This is the phase in which a user is suggested with candidate hashtags. Content a user enters is passed to the model to evaluate the topic-hashtag distributions and order them according to their probability scores. This procedure also uses the equation 4.1 only d refers to the union of the words in the content.

At all times during the process, if a user's content exceed a certain threshold then we re-evolve the topic distribution for the user on the whole.

4.3 Data Structure and Algorithms

In this section, we discuss the design of the database schema that is used to store the data of the recommendation approach and the algorithms that realize different modes of the approach in detail.

4.3.0.1 Database Schema

Figure 4.3 shows the Entity-Relationship (ER) model for hashtag recommendation function in the social network platform. The HashtagInfo table stores each of the hashtags entered by a user. Along with the hashtags we also save the keywords and topics of the content items of the user. The text field stores the hashtag text and the keywords field stores the words highlighted for a content when the LDA procedure is applied. These data are obtained by performing LDA model estimation on the content passed in. LDAEstimations table contains the information of he LDA model parameters for each of the recommendation mode mentioned before. These parameters are contained in the files generated by the LDA model estimation phases. These files are stored in the hard disks of server peers. The ContentType field in this table identifies which mode type the model refers to. The UserTopics table contains the preferred topics their topic-word and topic-hashtag distributions of each user.

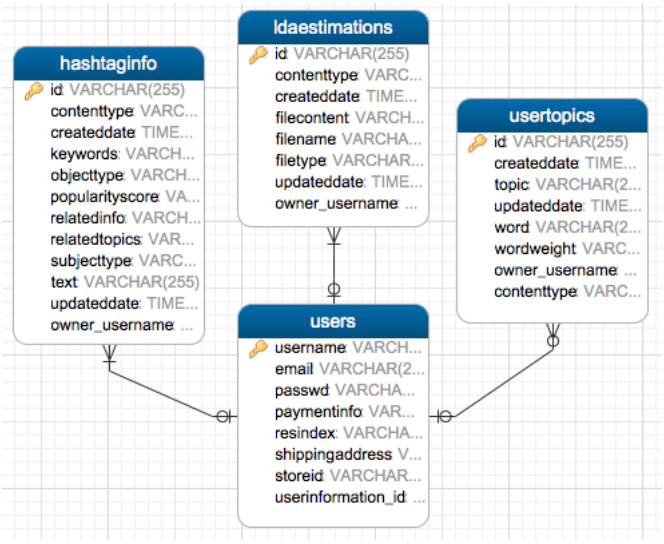

Figure 4.3: Database Model for Hashtag recommendation

This is the mainly used for identifying similar users.

4.3.0.2 LDA Operations

For LDA operations in our application we use JGibbLDA open source library. It is a java implementation with APIs for LDA procedure using Gibbs Sampling[3]. JGibbLDA has been used in various applications, for example, Information Retrieval, Content-Based Image Clustering and Text/Web Data Mining etc. It runs very fast and effectively analyses the hidden/latent topic structures of large-scale datasets, including large collections of text/web documents. This library is not only used for text and natural language processing but also for image processing too. It provides functions for LDA parameter estimation and inference. We present the data format of the input and output of the library here.

Input Data Format: For training/estimating a model, the library requires its input data to be organized as follows:

[M]
[document1]
[document2]
....
[documentM]

The first line indicates the total number of documents. Each line after that is a document. Each document has list of words or terms separated by blank character.

Output Data Format: The library has its output organized in different files:

1. others: Contains the general information of the model such as the number of topics, the alpha and beta hyper-parameters, number of documents, number of words and the Gibbs sampling iteration at which the data got saved.

2. phi: Contains the word-topic distributions

3. theta: Contains the topic-document distributions

4. tassign: Contains the topic assignments of words in training data

5. twords: Contains mostly likely words of each topic

There are two main functions provided by the JGibbLDA tool: estimation and inference. We use the two methods, which corresponds to the two functions, for our LDA operations. All of the operations of hashtag recommendation include either one of these operations. We discuss two of the major operations here: Updating recommendation LDA models with new data for estimation, a updating user topics of interests with new data.

Updating recommendation LDA model with new data for estimation: For all users, the system updates the model of each mode periodically at regular intervals. We identify different modes by the following strings: "global", "personal", "similar-users", "user-friends", "overall-popularity".

The GetLDAData operation takes userid and mode type as input and gives out LDA model data or NULL if there is none. JGibbLDA doesn't pre-process data for stop-words, special characters etc. So we do the pre-processing before we pass any of the content. Algorithm 4.1 describes the algorithm involved in adding data for LDA training or estimation and topic-hashtag distribution generation.

$V_{doc}$is the vector listing all of the documents and the count of number of documents in the required input format.

Z is the topic-word distribution from the estimation of the data.

w denotes a word.

pscore has the probability distribution score for a word.

Updating User Topics of Interests with new data: Algorithm 4.2 gives the steps for getting the topics of users interest based on the content. This is performed in LDA model estimation phases. All of the user content is passed and based on their estimation topics are identified. Among these topics the proposed approach identifies the prominent topics for the user. This is done by combining the prominent topics of each document. Those topics are selected according to the topic-document distribution.

In the case that a topic probability score crosses a threshold (F_x) then that topic is highlighted in that document. In this way we find topics that are standing out from

overall content. In the case that a topic is frequently used in a number of documents (larger than a threshold value (F_y)), then that topic is called as topic of user interests. These topics of interests of a user are updated regularly whenever the content counted for the user crosses a threshold. In algorithm 4.2,

Z_U is a list containing the topics of interests collected for a user U.

d_i points to each content of the user.

Algorithm 4.1 Adding data for recommendation LDA model estimation and identifying topic-hashtag distributions

Input: $V_{doc}(doc \in D)$, modeType (global/personal/user-friends/similar-users), u (user)

```
if GetLDAData(u, modeType) = null then then
  Z = performLDAEstimation(Vdoc)
  AddModelInLDAEstimations(methodType, u, Z)
  for w ∈ Z(z, w) do
    if w = hashtag then
      AddWordProbabilityScore(w, pscore, modeType, Z)
    end if
  end for
end if
```

Algorithm 4.2 Get User Topics of Interest

Input: $V_{doc}(doc \in D)$, modeType (global/personal/user-friends/similar-users), F_x(topic-document probability score threshold value), F_y(topic-frequency threshold score), u (user)

Output: Z_U(topics of interest for user U)

```
if GetLDAData(u, modeType) = null then then
  Z = GetLDAData(u, methodType)
  for d_i ∈ Z do
    if z ∈ Z is in d_i and z_d(i) > F_x then
      Z_F(z) = Z_F(z) + 1
    end if
  end for
  for z ∈ Z do
    if Z_F(z) > F_y then
      add z to Z_U
    end if
  end for
end if
```

4.3.0.3 Hashtag Recommendation Algorithms

In this subsection, we present the algorithms for Updating LDA model with added hashtags and recommending hash- tags with each of the method types mentioned in the proposed approach.

Updating LDA model with newly published content containing hashtags: This algorithm (Algorithm 4.3) is called up once a user publishes the content. Hashtags in the content submitted are identified this is done using parseContentForHashtags method. Once a list of hashtags are ready to be submitted, the initial step is to check if they are already present in the current server.

If hashtag is already there in the current node, then we update the topics($h(t)$), keywords and popularity score$h(pscore)$ of the hashtag. Popularity score of a hashtag is the obtained by computing the average number of the appearance of the hashtag normalized with total number of hashtags.

If a hashtag is new, then the proposed approach creates an entry for the hashtag in the current peer server, update all of the information like topics, keywords, ruleType and popularity score. RuleType is the privacy setting assigned to the content (presented in chapter 3). For all the method types except for "personal", we only consider hashtags that have ruleType as "public".

Algorithm 4.3 Updating LDA model with newly published content containing hashtags

Input: ruleType, U (user), d (content user has entered)

```
H = parseContentForHashtags(d)
for h ∈ H do
  if FindInHashtagInfo(h) ≠ null then
    h(pscore) = ((h(pscore)*|H_U|+1)/|H_U|)
    if h(modeType) = modeType then
      T_m = performLDAInference(d)
      UpdateHashtagRelatedTopics(T_m)
      UpdateHashtagRelatedKeywords(T_m)
    end if
  else
    T_m = performLDAInference(d)
    UpdateHashtagRelatedTopics(T_m)
    UpdateHashtagRelatedKeywords(T_m)
    h(pscore) = 1/|H_U|
    UpdateRuleTypeHashtag(h)
  end if
end for
```

Updating the list of popular hashtags in Bootstrap: Hashtags popular in the overall network are added and updated at regular intervals in bootstrap peer. Each time we update the data, all of the previous hashtags are cleared from the bootstrap peer. Based on the popularity score, in each server we identify top-k hashtags and add the same to bootstrap node along with the server information. Algorithm 4.4 gives the steps involved for the this method. The top-k algorithm for hashtags in BATON tree is as : 1) Right after a hashtag is created, the hashtag is published into BATON tree, and an entry for the hashtag is created in the BATON tree 2) Right after a hashtag is used, the usage of the hashtag is published, and the popularity of the hashtag is incremented by 1 3) Top-k hashtags are selected at each server peers, 4) From leaves to the root, a parent aggregates the top-k hashtags in its subtree.

Algorithm 4.4 Updating the list of popular hashtags in Bootstrap

```
Input: n (top hashtags), S (all server nodes)
  ClearAllHashtagsInBootstrap()
  for s ∈ S do
    H_s = GetTopNHashtags(s,"public")
    for h ∈ H_s do
      AddHashtagInfoToBootstrap(h, s)
    end for
  end for
```

Now we examine the algorithms of all of the five recommendation modes mentioned in the first subsection of this chapter.

Recommending Hashtags with Global content mode: Global content is randomly chosen content from the whole network. This is might even be similar to the content used for training the LDA. For this mode we consider both base LDA model data and also the daily basis global content with trending topics. We show the running procedure for this mode in Algorithm 4.5. In the operations for any of the modes, the initial operation would be to identify the topics in the content that is ready to be published by using the JGibbs library. Once the topics are identified, the proposed approach retrieves the hashtags that have the similar topic distribution from the current node.

Recommending Hashtags with Users preferences mode: In this method, we consider the topics of interests of a user strongly correlate with that of preferences of this user. In this mode, the approach lists two types of hashtags. One contains those previously used by the user, and the other contains those selected from topic-hashtag distribution according to the user's topics of interests. Once the approach gets the

Algorithm 4.5 Recommending Hashtags with Global content mode

Input: modeType ("global"), U (user), d (content user has entered)
Output: H (candidate hashtags list)

```
Server s = GetCurrentUserServerNode(U)
T_m = performLDAInference(d, modeType)
for z ∈ T_m do
  Z_s(z, h) = GetTopicHashtagsFromServer(s,z)
  for h ∈ Z_s(z, h) do
    add h to H
  end for
end for
```

topics of the published content (T_m), for each topic, it picks up the hashtags that have ruleType as "personal" and has the topic in its related topics i.e hashtags covering similar topics. This approach also retrieves the hashtags related to the current topic, using the topic-hashtag distribution in current node i.e the host server peer of the user. Algorithm 4.6 shows the process described until now for getting hashtags by users preferences.

Algorithm 4.6 Recommending Hashtags with Users preferences mode

Input: modeType ("personal"), U (user), d (content user has entered)
Output: H (candidate hashtags list)

```
Server s = GetCurrentUserServerNode(U)
T_m = performLDAInference(d, modeType)
for z ∈ T_m do
  Z_s(z, h) = GetTopicHashtagsFromServer(s,z)
  for h ∈ Z_s(z, h) do
    add h to H_z
  end for
  for z ∈ T_m do
    for h ∈ GetAllHashtagsFromUser(U) do
      if h(z)containsz then
        add h to tempH
      end if
    end for
    H = OrderHashtagsByPopularityScore(tempH) + H_z
  end for
end for
```

Recommending Hashtags with Similar Users mode: For determining users with similar interests, this approach uses the topics of interest property of multiple users. We find the users based on their topics of interest. We identify the users with

the topics that are both common to the content and current users interests using the cosine similarity score. For now, the approach covers only the users in the same server peer. In Algorithm 4.7,

$T_F$contains the topics that are common to the users interests and to the content being published.

$F_z$is a popularity score threshold which we use in-order to restrict the number of hashtags being retrieved.

We use cosine similarity metric as in equation 4.2 to calculate the similarity score between two users based on their topic distribution where u identifies the current user and u_k identifies rest of the users compared with the current user.

$$Sim(u, u_k) = \frac{u \ . \ u_k}{||u|| \ . \ ||u_k||} \tag{4.2}$$

Algorithm 4.7 Recommending Hashtags with Similar Users mode

Input: modeType ("similar-users"), U (user), d (content user has entered), F_z(threshold value for hashtag popularity score)
Output: H (candidate hashtags list)

```
Server s = GetCurrentUserServerNode(U)
T_m = performLDAInference(d, modeType)
T_U = GetUserTopicsOfInterest(U)
for z ∈ T_m do
  if T_U contains z then
    add z to T_F
  end if
end for
U_I = GetUsersWithTOI(T_F)
for u ∈ U_I do
  H_U = GetHashtagsFromUser(U)
  for h ∈ H_U do
    if h(t)containsT_Fandh(pscore) > F_z then
      add h to H
    end if
  end for
end for
```

Recommending Hashtags with Friends mode: Algorithm 4.8 shows the steps for getting candidate hashtags by considering users friends interests. This is almost same as the previous algorithm mentioned before for similar users. The only difference though is the type of users considered. In this case, we consider all of the friends of current user.

Algorithm 4.8 Recommending Hashtags with Friends mode

Input: modeType ("friends"), U (user), d (content user has entered), F_z(threshold value for hashtag popularity score)
Output: H (candidate hashtags list)

```
Server s = GetCurrentUserServerNode(U)
T_m = performLDAInference(d, modeType)
U_I = GetUsersFriendList(U)
for u ∈ U_I do
    H_U = GetHashtagsFromUser(U)
    for h ∈ H_U do
        if h(t)containsT_F and h(pscore) > F_z then
            add h to H
        end if
    end for
end for
```

Recommending Hashtags by overall popularity mode: This mode type lists hashtags that are popular in the neighboring nodes. The hashtags are collected from the bootstrap peer, based on the chosen server node. The nodes that we consider are left and right siblings, adjacent node and the current node. BATON overlay network makes it possible to identify these nodes with ease. We combine the hashtag candidates recommended from these nodes with topics similar to the current content. In the final step, we choose four scale parameters representing each node ($F_{l,}F_{r,}F_{a,}F_u$) such that sum of these parameters is equal to 1. Algorithm for this method is listed in 4.9.

4.4 Summary

The proposed approach considers various modes in recommendation. It allows the users of the social network platform to select the mode they want. This provides a personalized environment for the users. Another idea not yet implemented is to introduce the choices as advanced options for the users. In this, the user will be given two options i.e hashtags by global or personal content. If the user wants to use other methods for recommendation, then he/she needs to go for advanced settings and make a certain payment for usage. In the next section, we evaluate these methods in our prototype implementation and present the experimentation results for the same.

Algorithm 4.9 Recommending Hashtags by overall popularity mode

Input: modeType ("overall-popularity"), U (user), d (content user has entered)
Output: H (candidate hashtags list)

```
Server s = GetCurrentUserServerNode(U)
Server s_l = GetLeftSibling(U)
Server s_r = GetRightSibling(U)
Server s_a = GetAdjacentSibling(U)
Server H_l = GetHashtagsBootstrap(s_l)
Server H_r = GetHashtagsBootstrap(s_r)
Server H_a = GetHashtagsBootstrap(s_a)
Server H_U = GetHashtagsBootstrap(s)
T_m = performLDAInference(d, modeType)
for z ∈ T_m do
  for h ∈ H_l do
    if z not in h then
      H_l = H_l − h
    end if
  end for
  for h ∈ H_r do
    if z not in h then
      H_r = H_r − h
    end if
  end for
  for h ∈ H_a do
    if z not in h then
      H_a = H_a − h
    end if
  end for
  for h ∈ H_U do
    if z not in h then
      H_U = H_U − h
    end if
  end for
end for
F_l + F_r + F_a + F_u = 1
H = F_l * H_l + F_r * H_r + F_a * H_a + F_u * H_U
```

Chapter 5

Experimental Results

5.1 Overview

In this chapter, we discuss the experiments performed for verifying the correctness of the hashtag algorithms and for evaluating the effectiveness of these algorithm. First, we describe the setup environment, including the different kinds of datasets we used and also the pre-processing procedure we did for using these datasets as the input for the experiments. In our approach for the hashtag recommendation in the P2P social network platform, we designed five modes that we integrated into the system. We conducted experiments for each of the modes to evaluate the correctness and effectiveness of the developed algorithms.

5.2 Experiment Setup

All of the experiments were performed on a standalone computer with 16GB RAM and MAC OS X or Windows 7 operating systems. We developed a prototype application with all of the content management components and JUnit test cases for hashtag implementation. In our prototype, we have two server peers, one bootstrap peer and fifteen client peers. As mentioned in Chapter 4, we use JGibbLDA library[3] for performing LDA operations. We set the default values for hyper-parameters $\alpha = 50.0/k$ and $\beta = 0.1$ as suggested in [44] by Griffiths and Steyvers where k is the number of topics considered. For all the experiments we run the LDA operations through 500 iterations of Gibbs Sampling. The contents used by these experiments are, datasets obtained from three different sources. The first was from Textual Retrieval Conference (TREC) 2011 microblog track[10]. We choose Tweets2011 corpus. This corpus

Source	#tweets	Pre-processed
TREC Microblog (Tweet2011) corpus	16 million	No
Twitter Streaming API	10000	No
Sentiment140	1,60,0000 + 500	Yes

Table 5.1: Datasets Details

comprises of 16 million tweets collected over a period of two weeks between 24th January 2011 until 8th February. We used Twitter tools API provided by TREC Microblog track to extract tweets. The second source was Twitter web site. We use the Twitter Streaming API to extract tweets. We captured 10000 tweets with trending topics in specific intervals of time for two days. Third source is from Sentiment 140[9] project created by the students from Stanford University for the purpose of Sentiment analysis of topics in tweets. This collection consisted of two datasets: one was training data with 1,60,0000 tweets and one was test data with 500 tweets. As mentioned in 5.1, only the Sentiment140 data is pre-processed, where any special characters or emoticons etc are removed. Before passing the data to the LDA functions, we selected the tweets in these datasets, removed the special characters or any characters other than english letters. The special characters "#" are kept, since it indicating the beginning of a hashtag.

5.3 Experiments and Results

We perform experiments on each of the recommendation modes mentioned in our proposed approach. For evaluating the effectiveness of this recommendation approach, we consider hit-rate of the results from an execution of a recommendation activity as the criteria. A recommendation activity starts from invoking the recommendation function on a content upon a user's request to returning the results to the user. The equation to calculate the Hit-rate of the results is defined in equation 5.1. We identify a result as hit if atleast one of the recommended hashtags is a hashtag used for the content.

$$Hit - rate = \frac{Number\ of\ hits}{Number\ of\ content\ items\ considered} \tag{5.1}$$

There were three sets of experiments performed. One compares the hit rate percentage over the number of topics for each of the recommendation modes except for the mode in which we recommend hashtags based on their overall popularity. We considered the Twitter2011 dataset for this experiment. For each of the recommendation

modes we passed in 20,000 pre-processed tweets for the base LDA model training and parameter estimation. Then, we used the Sentiment140 dataset for evaluating the recommendation results. We did not perform any pre-processing on the Sentiment140 dataset. Of this sentiment140 dataset, we choose tweets its training set with minimum one hashtag, removed those hashtags and passed it as user entered text for testing. Each method was tested with 250 tweets. Figure 5.1 shows the graph plotted for four categories of recommendations with hit-rate against topics. Here we discuss the process followed for the experiments. Along with comparing the actual hashtags used in the content, we also performed subjective evaluation with the five evaluators. The evaluators where asked to mark the recommended hashtags as relevant and non-relevant. Majority views of the votes were considered for the final results. Initially for all the methods we started of with 50 topics for the LDA. Since topic-hashtag distribution is the main case that we consider for our proposal, with more topics we expected more hashtags.

Recommendation with global content was not satisfactory as it wouldn't consider any of the user content or user topics of interest. Maximum was 41.3% of hit-rate with global content that too at 500 topics. The results for User Preference based and similar users based recommendations were promising and we were able to see 55% and 57.6% of hit-rate respectively. For the recommendations from similar user and friends we used 5 clients as the users under comparison. Recommendation mode using friends content and interest could give approximately 50% of hit-rate. The other observation we made was on the cases with the number of topics between 300-500 topics; however there was not much of improvement with the results. So, for our application 300 topics would be the ideal number of topics to be considered. Overall Popularity based recommendation needs more server nodes to be evaluated. We were able to set-up only two server nodes. With two server nodes, the algorithm correctness was tested and we were able to retrieve the trending hashtags from the two server nodes.

The second experiment was performed to test the recommendation mode with similar users. This was done using the dataset obtained from Twitter with its Streaming API. As mentioned before we collected 10000 tweets of trending topics from specifically the ones with some of the politicians tagged in them. We wanted to check the maximum number of hashtags out of the total recommendations that would be relevant for the given content. We distributed around 600 tweets to each of the clients, increasing the number of clients at each step. When there was only one user we were not able to retrieve any related recommendations. At 10 and 15 client count we were able to retrieve 4 relevant hashtags of the recommendations made. Hence, as the

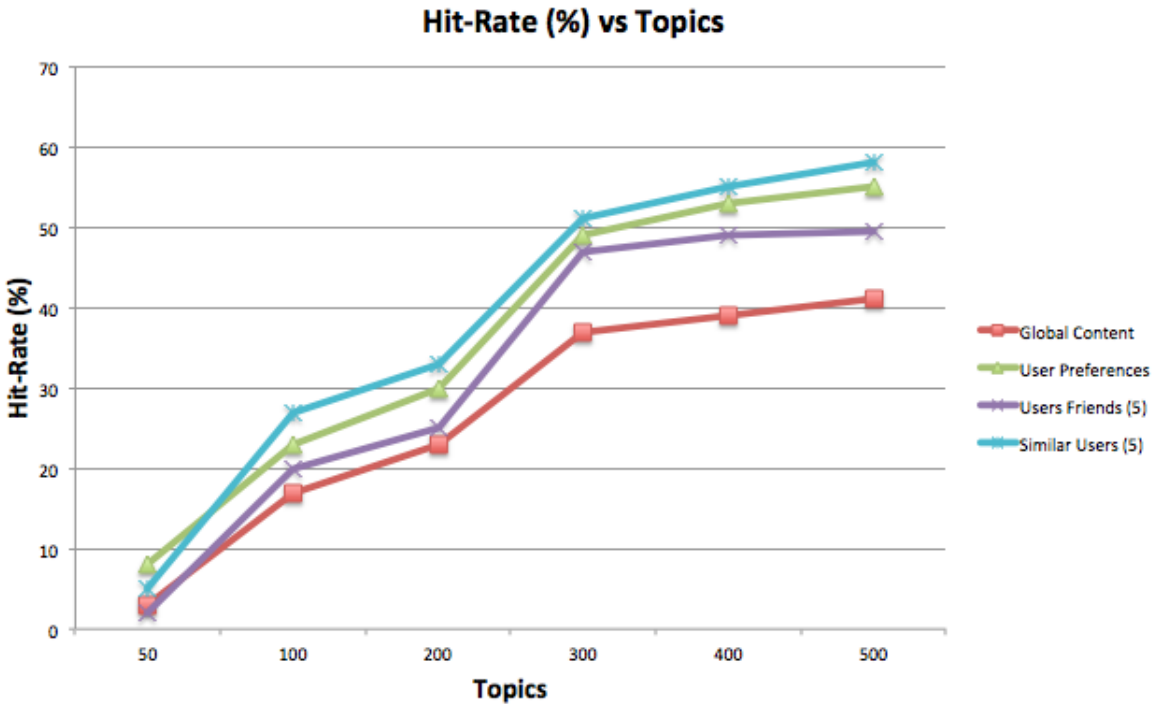

Figure 5.1: Hit-rate vs Number of Topics

users increase we would be able to provide with top-k recommendations with $k = 5$. Figure 5.2 shows the results for this experiment.

The last experiment was to check hit-rate when we used both user preferences and similar users recommendations at the same time. We used the same data from Twitter Streaming API for this experiment too. We tested for top-k recommendations when $k = 5$ and $k = 10$. For hit-1 this choice was good and we were able to acquire around 87% and 92% hit-rate for top-5 and top-10 recommendations respectively. The hit-all rate which checks for a match for all of the hashtags in a tweet, was around 63% when we used top-5 recommendations and it was still below 50% when top-10 recommendations were considered.From the results of this experiment we intuit that may be by combining more than one mode together the proposed approach could provide better results.

5.4 Comparison with Related Work

Until now, we evaluated our application as a standalone system. Based on the results of our experiments we are able to compare with the existing hashtag recommendation

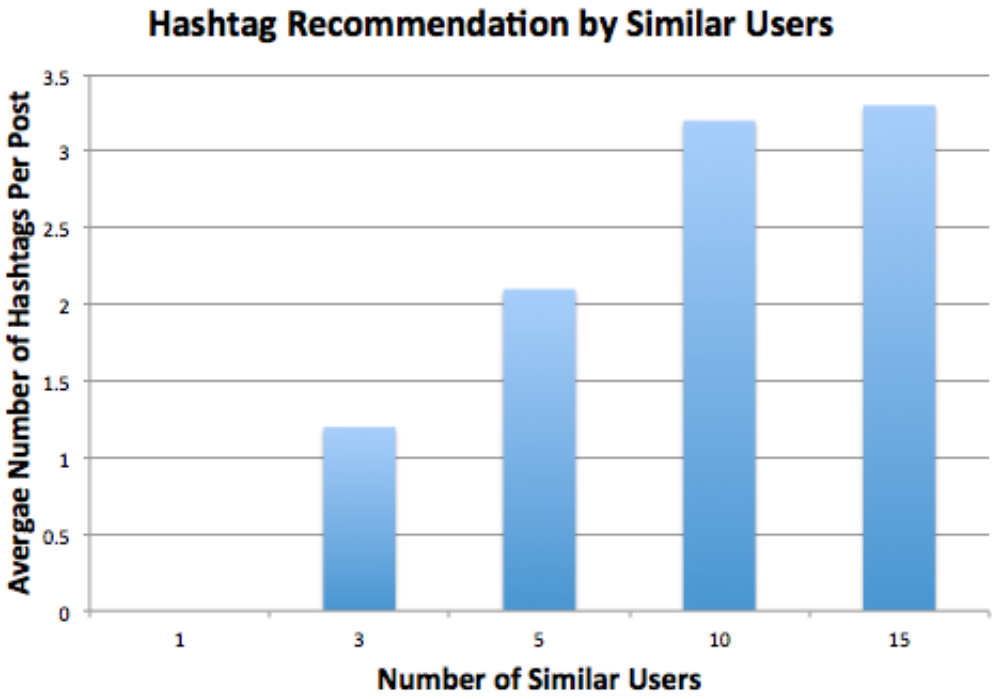

Figure 5.2: Average Hashtags Per Post vs Number of Similar Users

k	Measurement	Values
5	Hit-1	0.87
5	Hit-all	0.63
10	Hit-1	0.92
10	Hit-all	0.49

Table 5.2: Hit-rate for Top-k recommendations with User Preferences and Similar Users

systems. Table 5.3 gives the comparison details with tf-idf based hashtag recommendations. Most of them used Precision and Recall as their evaluation parameters. They are both defined as mentioned in equations 5.2 and 5.3 respectively. H_{rec} is the set of recommended hashtags and H_{orig} is the set of hashtags from the original tweet[104].

$$precision(H_{rec}) = \frac{|H_{rec} \bigcap H_{orig}|}{|H_{rec}|} \tag{5.2}$$

$$recall(H_{orig}) = \frac{|H_{rec} \bigcap H_{orig}|}{|H_{orig}|} \tag{5.3}$$

Table 5.4 shows the comparison of our recommendation approach with other topic model based recommendation system comparison. Most of the recommendation proposals (Zangerle et al[104], Mina Jeon et al[51], Godin et al[42], Jianjun Yu and Yi Shen[102]) use only content-based methodology. Using both content-based and collaborative filtering techniques like our proposal and in Jieying She and Lei Chen's approach[88] results in better hit-rate as shown in 5.4. Godin et al[42] even though has good hit-rate results, suggests only keywords for the recommendations. The authors do not consider the hashtags already in use and also no collaborative filtering techniques implemented, which reduces the scope of hashtags considered. As per our knowledge, none of the recommendations developed are for Peer-to-Peer network topology. By using the P2P features like scalability and maintenance our approach could achieve a better performance over the other studies. The overlay network structure with all the server peers organized in BATON give a better search performance.

Feature	**Zangerle et al**	**Kywe et al**	**Mina Jeon et al**	**Our Model**
Technique	TF-IDF	TF-IDF	TF-IDF	LDA-topic model
Content-based	Yes	Yes	Yes	Yes
Collaborative Filtering	No	Yes	No	Yes
Factors Considered	Similar tweets only	Tags from similar users and Similar Tweets	User interests, Similar tweets and popularity	Global content, User interests, Similar Users, Similar content, Popularity and hashtags from friends
Network Model	For twitter web application	For twitter web application	Android Application	Peer-to-Peer Java Application
Evaluation Criteria	Precision and Recall	Hit-rate	Precision and Recall	Hit-rate
Drawbacks	No user preferences considered at all. Only content based.	When user is new, only similar tweets are considered	Only content based.	Hit rate to be improved for global content based method.

Table 5.3: Comparison with TF-IDF Hashtag Recommendation Methods

Feature	**Godin et al**	**Jieying She and Lei chen**	**Jianjun Yu and Yi Shen**	**Our Model**
Content-based	Yes	Yes	Yes	Yes
Collaborative Filtering	No	Yes	No	Yes
Methodology	Suggested keywords for hashtags using topic model	Assume one local topic and one background topic per tweet. Also considered other user's followers topics.	Considered hashtag popularity, content similarity and time factor.	Global content, User interests, Similar Users, Similar content, Popularity and hashtags from friends
Evaluation Criteria	Hit-rate	Hit-rate	None mentioned	Hit-rate
Results	For top-5 hit-1 rate was 80%. For top-10 91% hit-1 rate. Hit-2 for top-5 was 51%.	For top-5 hit-1 rate was 82% and hit-all rate was 75%. For top-10 hit-1 rate was 89% and hit-all rate was 84%.	No Results	Hit-rate for collaborative filtering techniques together is around 87% hit-1 for top-5 and 92% hit-1 for top-10 recommendations.
Drawbacks	Only keywords not actual hashtags suggested	Only followers considered for topics.	No experimental results mentioned. Collaborative filtering techniques should have been added.	Hit-rate for each method tested independently. Above 50% observed for all of the methods except for global content method.

Table 5.4: Comparison with topic model based hashtag recommendation systems

Chapter 6

Conclusion and Future Work

6.1 Summary

This thesis starts with introduction of peer-to-peer architecture. It presents the detailed information of some existing social networking applications. These applications provide the target with different usages of social networks. Facebook[81] is mainly developed for the purpose of building and maintaining social relationships online, Twitter[48] also an online social network with short messaging service has similar purpose as of Facebook. LinkedIn a professional social network used as a communication link between employers and employees. WhatsApp[74], a mobile messaging service also used for communication between any of the users who have the application installed on their mobiles. We have discussed the goals of these social networking applications in detail.

We then focused on challenges in the social networking applications and the need for change in the architecture to peer-to-peer network topology. Challenges faced when using the current social networks includes mainly the issue of privacy and security for the user-generated content. There is a very minimal control of the user on the data generated. Many issues like Facebook Beacon[50] and data collection procedures by Google for Google Street View[84] project are some implementations which led to the breach of privacy. These issues are mostly due the fact that all of the above application have a central authority which governs the control of data. Peer-to-Peer networks provide more distributed control over the data. Therefore, we foresee that a social network application with a peer-to-peer overlay network would be superior to those applications providing services with centralized data centers.

We discussed briefly on some of major P2P social networking applications like PeerSON[85], Safebook[33], PESCA[82] etc. Most of these previous research have

implemented DHT services for the sake of storing the peer information, which adds an extra maintenance over head for the application. Safebook has limited guarantees on the availability of the system. PeerSON does not provide for a proper log out mechanism and also no keyword search is provided. None of them have had primary focus on the content categorization in the network.

This work has two purposes: One we introduce the P2P social networking architecture and content management components like user profile management, posts, blogs, privacy control, newsfeed and notification middleware of our P2P social networking application. Second is the hashtag recommendation approach proposed for this system using Latent Dirichlet Allocation[23] topic model. It is model which identifies hidden topics from a set of pre-processed documents. We specifically concentrate on identifying topic-hashtag distributions out of these hidden topics. These are further used for the recommendations. Our research uses both content-based and collaborative filtering methods for the recommendations which can be selected by the user on his own choice. Also, we provide the recommendations by considering content from the neighboring nodes in the network which would allow us for the fast processing of the recommendations.

The experiment results show more than 50% hit-rate for three of the collaborative filtering approaches. The hit-1 rate for top-5 and top-10 recommendations for hashtags considered from similar users and user content is the better than any of the topic model based hashtag recommendation systems. Also, using only similar users method guarantees that the approach is good for top-3 recommendations.

6.2 Future Work

There are some limitations as to the proposed recommendation methodology. We still have to test the performance of the algorithms in P2P simulated environment with more number of server nodes. Without which we were not able to test the overall popularity method. The next thing would be to consider a top-k recommendation system for all of the methods mentioned.

Till now, we worked on prototype implementation. There are some things in this research to be considered for future work. First, we would like to include e-commerce functionalities into the social networking application. As mentioned in Chapter 4, we would give the recommendation methods to the user as part of advanced settings and include more than one method for a recommendation. Our final goal is to get a completely functional P2P social networking application with e-commerce capabilities

out into the market. We use relational database for storing both the bootstrap and server peer data. With the users increasing, at some point we need to consider moving to Big data solutions. Also, we need add in some encryption mechanisms for securing the client data stored on the server.

References

[1] Bittorrent, http://www.bittorrent.com/.

[2] Facebook, https://www.facebook.com/.

[3] Jgibblda, http://jgibblda.sourceforge.net/.

[4] Linkedin, https://www.linkedin.com/.

[5] The napster home page, http://www.napster.com.

[6] Orkut, https://orkut.google.com/.

[7] Music genome project, http://www.pandora.com/mgp.shtml.

[8] About pinterest, https://about.pinterest.com/en/whats-pinterest.

[9] Sentiment 140, http://www.sentiment140.com/.

[10] Trec twitter2011 datasets, http://trec.nist.gov/data/tweets/.

[11] Twitter followers, https://support.twitter.com/articles/14019-faqs-about-following.

[12] Viber, http://www.viber.com/en/.

[13] Wechat, http://www.wechat.com/en/.

[14] . Bestpeer: A self-configurable peer-to-peer system. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, pages 272–, Washington, DC, USA, 2002. IEEE Computer Society. URL http://dl.acm.org/citation.cfm?id=876875.878980.

[15] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge*

and Data Engineering, IEEE Transactions on, 17(6):734–749, June 2005. ISSN 1041-4347. doi: 10.1109/TKDE.2005.99.

[16] Sulaiman Ainin, M. Muzamil Naqshbandi, Sedigheh Moghavvemi, and Noor Ismawati Jaafar. Facebook usage, socialization and academic performance. *Computers and Education*, 83(0):64 – 73, 2015. ISSN 0360-1315. doi: http://dx.doi.org/10.1016/j.compedu.2014.12.018. URL http://www.sciencedirect.com/science/article/pii/S0360131515000160.

[17] Fawzi Hasan Altaany and Firas Ajil Jassim. Impact of facebook usage on undergraduate students performance in irbid national university: Case study. *CoRR*, abs/1306.4779, 2013. URL http://arxiv.org/abs/1306.4779.

[18] David P. Anderson. Boinc: A system for public-resource computing and storage. In *Proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID 04)*, 2004.

[19] David P. Anderson, Jeff Cobb, Eric Korpela, Matt Lebofsky, and Dan Werthimer. Seti@home: An experiment in public-resource computing. *Commun. ACM*, 45(11):56–61, November 2002. ISSN 0001-0782. doi: 10.1145/581571.581573. URL http://doi.acm.org/10.1145/581571.581573.

[20] Marko Balabanović and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, March 1997. ISSN 0001-0782. doi: 10.1145/245108.245124. URL http://doi.acm.org/10.1145/245108.245124.

[21] S.B. Barnes. 'a privacy paradox: Social networking in the united states' first monday 11(9), http://firstmonday.org/article/view/1394/1312.

[22] David M Blei. Probabilistic topic models. *Communications of the ACM*, 55(4): 77–84, 2012.

[23] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944937.

[24] O. Bodriagov and S. Buchegger. Encryption for peer-to-peer social networks. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 1302–1309, Oct 2011. doi: 10.1109/PASSAT/SocialCom.2011.158.

[25] danah michele boyd. Friendster and publicly articulated social networking. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, pages 1279–1282, New York, NY, USA, 2004. ACM. ISBN 1-58113-703-6. doi: 10.1145/985921.986043. URL http://doi.acm.org/10.1145/985921.986043.

[26] Sonja Buchegger, Doris Schiöberg, Le-Hung Vu, and Anwitaman Datta. Peerson: P2p social networking: Early experiences and insights. In *Proceedings of the Second ACM EuroSys Workshop on Social Network Systems*, SNS '09, pages 46–52, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-463-8. doi: 10.1145/1578002.1578010. URL http://doi.acm.org/10.1145/1578002.1578010.

[27] Jui-Hung Chang, Chin-Feng Lai, Yueh-Min Huang, and Han-Chieh Chao. 3prs: a personalized popular program recommendation system for digital tv for p2p social networks. *Multimedia Tools and Applications*, 47(1):31–48, 2010.

[28] G. Chen, T. Hu, D. Jiang, P. Lu, K.-L. Tan, Hoang Tam Vo, and S. Wu. Bestpeer++: A peer-to-peer based large-scale data processing platform. In *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pages 582–593, April 2012. doi: 10.1109/ICDE.2012.18.

[29] Gang Chen, Tianlei Hu, Dawei Jiang, Peng Lu, Kian-Lee Tan, Hoang Tam Vo, and Sai Wu. Bestpeer++: A peer-to-peer based large-scale data processing platform. In *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ICDE '12, pages 582–593, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4747-3. doi: 10.1109/ICDE.2012.18. URL http://dx.doi.org/10.1109/ICDE.2012.18.

[30] Ming-Hung Chen, KC-J Lin, Chui-Chiu Kung, Cheng-Fu Chou, and Chang-Jen Tu. On the design of the semantic p2p system for music recommendation. In *Parallel and Distributed Processing with Applications (ISPA), 2010 International Symposium on*, pages 442–448. IEEE, 2010.

[31] Huan-Ming Chuang, Li-Chuan Wang, and Chu-Ching Pan. A study on the comparison between content-based and preference-based recommendation systems. In *Semantics, Knowledge and Grid, 2008. SKG '08. Fourth International Conference on*, pages 477–480, Dec 2008. doi: 10.1109/SKG.2008.89.

[32] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *International Workshop on Designing Privacy Enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 46–66, New York, NY, USA, 2001. Springer-Verlag New York, Inc. ISBN 3-540-41724-9. URL http://dl.acm.org/citation.cfm?id=371931.371977.

[33] L.A. Cutillo, R. Molva, and T. Strufe. Privacy preserving social networking through decentralization. In *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, pages 145–152, Feb 2009. doi: 10.1109/WONS.2009.4801860.

[34] L.A. Cutillo, R. Molva, and T. Strufe. Safebook: A privacy-preserving online social network leveraging on real-life trust. *Communications Magazine, IEEE*, 47(12):94–101, Dec 2009. ISSN 0163-6804. doi: 10.1109/MCOM.2009.5350374.

[35] Dmitry Davidov, Oren Tsur, and Ari Rappoport. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 241–249, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1944566.1944594.

[36] Fady Draidi, Esther Pacitti, and Bettina Kemme. P2prec: a p2p recommendation system for large-scale data sharing. In *Transactions on large-scale data-and knowledge-centered systems III*, pages 87–116. Springer, 2011.

[37] P. Druschel and A. Rowstron. Past: a large-scale, persistent peer-to-peer storage utility. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 75–80, May 2001. doi: 10.1109/HOTOS.2001.990064.

[38] M. Durr, M. Maier, and F. Dorfmeister. Vegas – a secure and privacy-preserving peer-to-peer online social network. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 868–874, Sept 2012. doi: 10.1109/SocialCom-PASSAT.2012.42.

[39] C. Dwyer, S.R. Hiltz, and George Widmeyer. Understanding development and usage of social networking sites: The social software performance model. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 292–292, Jan 2008. doi: 10.1109/HICSS.2008.476.

[40] Alberto García-Sola and Juan A Botía. Semantic overlay networks for social recommendation in p2p. In *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)*, pages 274–283. Springer, 2009.

[41] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *Proceedings of the 28th Annual International Conference on Advances in Cryptology: The Theory and Applications of Cryptographic Techniques*, EUROCRYPT '09, pages 171–188, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-01000-2. doi: 10.1007/978-3-642-01001-9_10. URL http://dx.doi.org/10.1007/978-3-642-01001-9_10.

[42] Fréderic Godin, Viktor Slavkovikj, Wesley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using topic models for twitter hashtag recommendation. In *Proceedings of the 22nd international conference on World Wide Web companion*, pages 593–596. International World Wide Web Conferences Steering Committee, 2013.

[43] K. Graffi, C. Gross, D. Stingl, D. Hartung, A. Kovacevic, and R. Steinmetz. Lifesocial.kom: A secure and p2p-based solution for online social networks. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 554–558, Jan 2011. doi: 10.1109/CCNC.2011.5766541.

[44] Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[45] Feng Guo and Shaozi Li. A document recommendation system based on clustering p2p networks. In *Cooperative Design, Visualization, and Engineering*, pages 226–233. Springer, 2007.

[46] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: 10.1145/223904.223929. URL http://dx.doi.org/10.1145/223904.223929.

[47] Zhuoye Ding Qi Zhang XuanJing Huang. Automatic hashtag recommendation for microblogs using topic-specific translation model. In *24th International Conference on Computational Linguistics*, page 265. Citeseer, 2012.

[48] Twitter Inc. About twitter, https://about.twitter.com/company.

[49] H. V. Jagadish, Beng Chin Ooi, and Quang Hieu Vu. Baton: A balanced tree structure for peer-to-peer networks. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 661–672. VLDB Endowment, 2005. ISBN 1-59593-154-6. URL http://dl.acm.org/citation.cfm?id=1083592.1083670.

[50] A. Jamal and M. Cole. A heuristic evaluation of the facebook's advertising tool beacon. In *Information Science and Engineering (ICISE), 2009 1st International Conference on*, pages 1527–1530, Dec 2009. doi: 10.1109/ICISE.2009.45.

[51] Mina Jeon, Sanghoon Jun, and Eenjun Hwang. Hashtag recommendation based on user tweet and hashtag classification on twitter. In *Web-Age Information Management*, pages 325–336. Springer, 2014.

[52] Edson C. Tandoc Jr., Patrick Ferrucci, and Margaret Duffy. Facebook use, envy, and depression among college students: Is facebooking depressing? *Computers in Human Behavior*, 43(0):139 – 146, 2015. ISSN 0747-5632. doi: http://dx.doi.org/10.1016/j.chb.2014.10.053. URL http://www.sciencedirect.com/science/article/pii/S0747563214005767.

[53] Ahmad A. Kardan and Mahnaz Ebrahimi. A novel approach to hybrid recommendation systems based on association rules mining for content recommendation in asynchronous discussion groups. *Inf. Sci.*, 219:93–110, January 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2012.07.011. URL http://dx.doi.org/10.1016/j.ins.2012.07.011.

[54] Jae Kyeong Kim, Hyea Kyeong Kim, and Yoon Ho Cho. A user-oriented contents recommendation system in peer-to-peer architecture. *Expert Systems with Applications*, 34(1):300–312, 2008.

[55] Shoubin Kong, Qiaozhu Mei, Ling Feng, and Zhe Zhao. Real-time predicting bursting hashtags on twitter. In *Web-Age Information Management*, pages 268–271. Springer, 2014.

[56] Joseph A. Konstan, Bradley N. Miller, David Maltz, Jonathan L. Herlocker, Lee R. Gordon, and John Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, March 1997. ISSN 0001-0782. doi: 10.1145/245108.245126. URL http://doi.acm.org/10.1145/245108.245126.

[57] Ralf Krestel, Peter Fankhauser, and Wolfgang Nejdl. Latent dirichlet allocation for tag recommendation. In *Proceedings of the Third ACM Conference on Recommender Systems*, RecSys '09, pages 61–68, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-435-5. doi: 10.1145/1639714.1639726. URL http://doi.acm.org/10.1145/1639714.1639726.

[58] John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishan Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao. Oceanstore: An architecture for global-scale persistent storage. *SIGPLAN Not.*, 35(11):190–201, November 2000. ISSN 0362-1340. doi: 10.1145/356989.357007. URL http://doi.acm.org/10.1145/356989.357007.

[59] Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. On recommending hashtags in twitter networks. In *Social Informatics*, pages 337–350. Springer, 2012.

[60] Wanqing Tu Lei Zhang. Six degrees of separation in online society. In *In: Proceedings of the WebSci'09: Society On-Line*, March 2009.

[61] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[62] Eden Litt, Erin Spottswood, Jeremy Birnholtz, Jeff T. Hancock, Madeline E. Smith, and Lindsay Reynolds. Awkward encounters of an "other" kind: Collective self-presentation and face threat on facebook. In *Proceedings of the 17th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '14, pages 449–460, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2540-0. doi: 10.1145/2531602.2531646. URL http://doi.acm.org/10.1145/2531602.2531646.

[63] Zhaobin Liu, Wenyu Qu, Haitao Li, and Changsheng Xie. A hybrid collaborative filtering recommendation mechanism for p2p networks. *Future generation computer systems*, 26(8):1409–1417, 2010.

[64] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

[65] S. Mahmood. New privacy threats for facebook and twitter users. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 Seventh International Conference on*, pages 164–169, Nov 2012. doi: 10.1109/3PGCIC.2012.46.

[66] Ching man Au Yeung, Ilaria Liccardi, Kanghao Lu, Oshani Seneviratne, and Tim Berners-Lee. Decentralization: The future of online social networking. In *W3C Workshop on the Future of Social Networking Position Papers*, 2009. URL http://www.w3.org/2008/09/msnws/papers/decentralization.pdf.

[67] Adam Mathes. Folksonomies-cooperative classification and communication through shared metadata. *Computer Mediated Communication*, 47(10):1–13, 2004.

[68] P. McFedries. Hashtag, you're it [technically speaking]. *Spectrum, IEEE*, 50(4): 24–24, April 2013. ISSN 0018-9235. doi: 10.1109/MSPEC.2013.6481692.

[69] Yuanyuan Yang Min Yang. An efficient hybrid peer-to-peer system for distributed data sharing. In *IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2008)*, pages 1–10, 2008.

[70] Axelle Moreau, Stéphanie Laconi, Manon Delfour, and Henri Chabrol. Psychopathological profiles of adolescent and young adult problematic facebook users. *Computers in Human Behavior*, 44(0):64 – 69, 2015. ISSN 0747-5632. doi: http://dx.doi.org/10.1016/j.chb.2014.11.045. URL http://www.sciencedirect.com/science/article/pii/S0747563214006529.

[71] R. Narendula, T.G. Papaioannou, and K. Aberer. My3: A highly-available p2p-based online social network. In *Peer-to-Peer Computing (P2P), 2011 IEEE International Conference on*, pages 166–167, Aug 2011. doi: 10.1109/P2P.2011.6038730.

[72] Dan Noyes. The top 20 valuable facebook statistics – updated february 2015 https://zephoria.com/social-media/top-15-valuable-facebook-statistics/.

[73] Hidehiko Okada and Makoto Inoue. Evaluation of p2p information recommendation based on collaborative filtering. In *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments*, pages 449–458. Springer, 2007.

[74] Parmy Olsen. Whatsapp hits 600 million active users, founder says - http://www.forbes.com/sites/parmyolson/2014/08/25/whatsapp-hits-600-million-active-users-founder-says/.

[75] Eriko Otsuka, Scott A Wallace, and David Chiu. Design and evaluation of a twitter hashtag recommendation system. In *Proceedings of the 18th International Database Engineering & Applications Symposium*, pages 330–333. ACM, 2014.

[76] Alexandre Passant, Philippe Laublet, John G. Breslin, and Stefan Decker. A uri is worth a thousand tags: From tagging to linked data with moat. *Int. J. Semantic Web Inf. Syst.*, 5(3):71–94, 2009. URL http://dblp.uni-trier.de/db/journals/ijswis/ijswis5.html#PassantLBD09.

[77] Xueping Peng, Yujuan Cao, and Zhendong Niu. Mining web access log for the personalization recommendation. In *MultiMedia and Information Technology, 2008. MMIT '08. International Conference on*, pages 172–175, Dec 2008. doi: 10.1109/MMIT.2008.166.

[78] Marco Pennacchiotti and Siva Gurumurthy. Investigating topic models for social media user recommendation. In *Proceedings of the 20th International Conference Companion on World Wide Web*, WWW '11, pages 101–102, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0637-9. doi: 10.1145/1963192.1963244. URL http://doi.acm.org/10.1145/1963192.1963244.

[79] M. Portmann, P. Sookavatana, S. Ardon, and A. Seneviratne. The cost of peer discovery and searching in the gnutella peer-to-peer file sharing protocol. In *Networks, 2001. Proceedings. Ninth IEEE International Conference on*, pages 263–268, Oct 2001. doi: 10.1109/ICON.2001.962351.

[80] Liza Potts, Joyce Seitzinger, Dave Jones, and Angela Harrison. Tweeting disaster: Hashtag constructions and collisions. In *Proceedings of the 29th ACM International Conference on Design of Communication*, SIGDOC '11, pages 235–240, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0936-3. doi: 10.1145/2038476.2038522. URL http://doi.acm.org/10.1145/2038476.2038522.

[81] Brunner SR Rains SA. What can we learn about social network sites by studying facebook? a call and recommendations for research on social network sites. In *New Media and Society*, August 2014.

[82] F. Raji, M. Davarpanah Jazi, and A. Miri. Pesca: a peer-to-peer social network architecture with privacy-enabled social communication and data availability. *Information Security, IET*, 9(1):73–80, 2015. ISSN 1751-8709. doi: 10.1049/iet-ifs.2013.0256.

[83] Daniel Ramage, Susan T Dumais, and Daniel J Liebling. Characterizing microblogs with topic models. *ICWSM*, 5(4):130–137, 2010.

[84] K. Sangani. Who owns... your personal data? [consumer tech ownership]. *Engineering Technology*, 5(11):28–29, July 2010. ISSN 1750-9637.

[85] Doris Schiöberg. A peer-to-peer infrastructure for social networks. *Diplom thesis, TU Berlin, Berlin, Germany*, 2008.

[86] Surendra Sedhai and Aixin Sun. Hashtag recommendation for hyperlinked tweets. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 831–834. ACM, 2014.

[87] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating “word of mouth”. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 210–217, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co. ISBN 0-201-84705-1. doi: 10.1145/223904.223931. URL http://dx.doi.org/10.1145/223904.223931.

[88] Jieying She and Lei Chen. Tomoha: Topic model-based hashtag recommendation on twitter. In *Proceedings of the Companion Publication of the 23rd International Conference on World Wide Web Companion*, WWW Companion '14, pages 371–372, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-2745-9. doi: 10.1145/2567948.2577292. URL http://dx.doi.org/10.1145/2567948.2577292.

[89] M. Srivatsa and L. Liu. Vulnerabilities and security threats in structured overlay networks: A quantitative analysis. In *In Proc. ACSAC*, pages 252–261, 2004.

[90] Mark Steyvers and Tom Griffiths. Probabilistic topic models. *Handbook of latent semantic analysis*, 427(7):424–440, 2007.

[91] Amara Tariq, Asim Karim, Fernando Gomez, and Hassan Foroosh. Exploiting topical perceptions over multi-lingual text for hashtag suggestion on twitter. In *FLAIRS Conference*, 2013.

[92] Abhineshwar Tomar, Frederic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. Towards twitter hashtag recommendation using distributed word representations and a deep feed forward neural network. In *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*, pages 362–368. IEEE, 2014.

[93] Amund Tveit. Peer-to-peer based recommendations for mobile commerce. In *Proceedings of the 1st international workshop on Mobile commerce*, pages 26–29. ACM, 2001.

[94] Beng Chin Ooi Vu Quang Hieu, Mihai Lupu. *Peer-to-Peer computing : principles and applications*. Springer, 2010.

[95] Thomas Vander Wal. Folksonomy, http://vanderwal.net/folksonomy.html.

[96] Chian Wang and Dai-Yang Lin. A movie recommendation mechanism based on user ratings in the mobile peer-to-peer environment. In *HCI International 2011–Posters' Extended Abstracts*, pages 610–614. Springer, 2011.

[97] Jun Wang, Jian Peng, and Xiaoyang Cao. A distributed collaborative filtering recommendation model for p2p networks. In *Collaborative Computing: Networking, Applications and Worksharing*, pages 1–10. Springer, 2009.

[98] Sen Wang. uopstore: an ecommerce platform with a peer-to-peer infrastructure. Master's thesis, School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Canada, 2014.

[99] Xiaolong Wang, Furu Wei, Xiaohua Liu, Ming Zhou, and Ming Zhang. Topic sentiment analysis in twitter: A graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1031–1040, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063726. URL http://doi.acm.org/10.1145/2063576.2063726.

[100] Yuan Wang, Jishi Qu, Jie Liu, Jimeng Chen, and Yalou Huang. What to tag your microblog: Hashtag recommendation based on topic analysis and collabo-

rative filtering. In *Web Technologies and Applications*, pages 610–618. Springer, 2014.

[101] Feng Xiao, Tomoya Noro, and Takehiro Tokuda. News-topic oriented hashtag recommendation in twitter based on characteristic co-occurrence word detection. In *Proceedings of the 12th International Conference on Web Engineering*, ICWE'12, pages 16–30, Berlin, Heidelberg, 2012. Springer-Verlag. ISBN 978-3-642-31752-1. doi: 10.1007/978-3-642-31753-8. URL http://dx.doi.org/10.1007/978-3-642-31753-8.

[102] Jianjun Yu and Yi Shen. Evolutionary personalized hashtag recommendation. In *Web-Age Information Management*, pages 34–37. Springer, 2014.

[103] Eva Zangerle, Wolfgang Gassler, and Günther Specht. On the impact of text similarity functions on hashtag recommendations in microblogging environments. *Social Network Analysis and Mining*, 3(4):889–898, 2013.

[104] Gassler W. Zangerle E. Recommending hashtags in twitter. In *Workshop on Semantic Adaptive Social Web 2011, in connection with the 19th International Conference on User Modeling, Adaptation and Personalization (2011)*.

[105] Qi Zhang, Yeyun Gong, Xuyang Sun, and Xuanjing Huang. Time-aware personalized hashtag recommendation on social media.

[106] Wei Zhao, Ziyu Guan, and Zheng Liu. Ranking on heterogeneous manifolds for tag recommendation in social tagging services. *Neurocomputing*, 148(0): 521 – 534, 2015. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2014.07.011. URL http://www.sciencedirect.com/science/article/pii/S0925231214008893.