Enhancing Recommendations for Conference Participants with Community and Topic Modeling

by

Bharath Reddy Pasham

LIU-IDA/LITH-EX-A--13/007—SE

2013-02-12
Master Thesis

Enhancing Recommendations for Conference Participants with Community and Topic Modeling

by

Bharath Reddy Pasham

LIU-IDAB---LITH-EX-A-13/007—SE

2013-02-12

Supervisor: Prof. Dr. Mattias Jarke, RWTH Aachen
Examiner: Prof. Dr. Patrick Lambrix
Abstract

For a researcher it is always important to increase his/her social capital and excel at their research area. For this, conferences act as perfect medium where researchers meet and present their work. However, due to the structure of the conferences finding similar authors or interesting talks is not obvious for the researchers. One of most important observation made from the conferences is, researchers tend to form communities with certain research topics as the series of conferences progresses. These communities and their research topics could be used in helping researchers find their potential collaborators and in attending interesting talks.

In this research we present the design and implementation of a recommender system which is built to provide recommendation of authors and talks at the conferences. Various concepts like Social Network Analysis (SNA), context awareness, community analysis, and topic modeling are used to build the system. This system can be considered as an extension to the previous system CAMRS (Context Aware Mobile Recommender System). CAMRS is a mobile application which serves the same purpose as the current system. However, CAMRS uses only SNA and context to provide recommendations. Current system, CAMRS-2, is also an Android application built using REST based architecture. The system is successfully deployed, and as part of thesis the system is evaluated. The evaluation results proved CAMRS-2 provides better recommendations over its predecessor.
Abstract
Contents

Abstract

1 Introduction
  1.1 Motivation .................................................. 1
  1.2 Thesis Goal .................................................. 1
    1.2.1 Realization ............................................. 2
  1.3 Thesis Structure ........................................... 2

2 State of the Art
  2.1 SNA ............................................................ 5
    2.1.1 Communities and Communities of Practice .............. 6
    2.1.2 Link Prediction in Social Networks .................... 7
  2.2 Recommender Systems ...................................... 8
    2.2.1 Collaborative Filtering ................................ 9
    2.2.2 Content Based Recommender Systems ................ 11
    2.2.3 Hybrid Recommender Systems .......................... 12
  2.3 Context Aware Recommender Systems ....................... 12
  2.4 Community Detection ...................................... 14
    2.4.1 Detecting Communities ................................ 14
    2.4.2 Greedy Approach ..................................... 15
  2.5 Topic Modeling ............................................ 17
    2.5.1 Various Approaches ................................... 18
    2.5.2 Topic Modeling from Titles of Documents ............ 18
    2.5.3 TF-IDF .................................................. 20
    2.5.4 Tagging .................................................. 21
  2.6 Existing Systems .......................................... 22
    2.6.1 Conference Navigator .................................. 22
    2.6.2 CollabSeer .............................................. 23
    2.6.3 Lanyrd ................................................... 23

3 Conceptual Approach ........................................... 25
  3.1 Community Modeling ....................................... 25
  3.2 Academic Event Modeling .................................. 27
    3.2.1 Academic Event ....................................... 27
    3.2.2 Session Formats ....................................... 28
  3.3 Scenarios .................................................. 29
## Contents

3.4 Use Case .......................................................................................................................... 29
  3.4.1 User Roles and Interactions ......................................................................................... 30
3.5 Recommendations Approach ............................................................................................. 30
  3.5.1 Bibliographic Data ........................................................................................................ 30
  3.5.2 Communities .................................................................................................................. 33
  3.5.3 Community Theme ........................................................................................................ 35
  3.5.4 Similarities Computation .............................................................................................. 36
  3.5.5 Similarity and Context ................................................................................................... 38
  3.5.6 Recommendation .......................................................................................................... 38

4 Architecture and Implementation .......................................................................................... 41
  4.1 Architecture ....................................................................................................................... 41
    4.1.1 System Architecture ...................................................................................................... 41
  4.2 Implementation .................................................................................................................... 43
    4.2.1 Overview of Tools and Technologies ............................................................................ 43
      4.2.1.1 Presentation Layer Tools .......................................................................................... 43
      4.2.1.2 Application Layer Tools .......................................................................................... 50
      4.2.1.3 Database Layer Tools .............................................................................................. 55
    4.2.2 Data Sources .................................................................................................................. 56
    4.2.3 Recommendation .......................................................................................................... 58
      4.2.3.1 Operational Principle of RESTful Web Service ......................................................... 60
      4.2.3.2 The CAMRS-2 Recommendation Engine ................................................................. 60
    4.2.4 CAMRS-2 Client ............................................................................................................ 65
      4.2.4.1 User Interface ............................................................................................................. 65
      4.2.4.2 Environment Sensing Module .................................................................................. 69
  4.3 Summary ............................................................................................................................. 70

5 Evaluation ............................................................................................................................. 73
  5.1 Data Set ................................................................................................................................ 73
  5.2 Method .................................................................................................................................. 74
  5.3 Results and Comparison ...................................................................................................... 75

6 Conclusion and Future Work .................................................................................................. 77
  6.1 Conclusion ........................................................................................................................... 77
  6.2 Future Work .......................................................................................................................... 78
    6.2.1 Communities .................................................................................................................. 78
    6.2.2 Topic Detection .............................................................................................................. 78
    6.2.3 Application Improvement .............................................................................................. 78

7 Bibliography .......................................................................................................................... 81
List of Tables

2.1 Different Hybrid Methods .............................................. 12
2.2 Time Complexities of different hierarchical clustering algorithms ... 16
2.3 Existing Systems .......................................................... 22
3.1 AERCS Data Summary ...................................................... 33
4.1 Comparing Porter Stemmer and Stanford CoreNLP .................. 55
5.1 CAMRS-2 and CAMRS-1 Talk Recommendations ..................... 75
## List of Figures

### 2.1 Community structure of a network with three communities . . . . . . . 14

### 3.1 ER Diagram of Community ........................................... 26

### 3.2 CAMRS - 2 Use Case .................................................. 31

### 3.3 Recommendation Process ............................................. 32

### 3.4 Using pre determined communities and CF techniques for Talk recom-

### mendations ................................................................. 40

### 4.1 CAMRS-2 System Architecture ....................................... 44

### 4.2 QR representation of text informatik .................................. 50

### 4.3 Program Schema ....................................................... 57

### 4.4 Talks Schema .......................................................... 58

### 4.5 Talk Schema ........................................................... 59

### 4.6 Author Recommendation .............................................. 62

### 4.7 Talk Recommendation ................................................ 63

### 4.8 Recommended Authors and Recommended Talks ....................... 68

### 4.9 An Author's Profile Page and an Author's Research Interests .......... 68

### 4.10 Program Schedule and a Talk Details ................................ 69
1 Introduction

1.1 Motivation

Conferences are occasions where researchers from different research areas come together to present their work and discuss with others and also one of the most used mediums by the researchers to learn and engage with other researchers. Academic events are considered most helpful platform for budding researchers, who try meet new researchers and attend interesting talks to propel at their research area. Kumar is one such enthusiast Ph.D. student, who has coauthored few research papers with his fellow researchers and looking forward to collaborate with other researchers. Fortunately he gets to attend a conference which holds few talks on research topics he is currently working. It occurs to Kumar that this academic event would be the place where he can attend interesting talks and meet new researchers with whom he could collaborate in the future. Due to the importance and prominence of academic events there are a large number of attendants from various research areas, and learning about his potential collaborators at conferences turns out to be highly difficult for Kumar. Even choosing a talk to attend is not easy for a novice researcher like Kumar. Since, conferences are structured in such a way, they consist multiple parallel tracks, and each track has a series of talks. So, choosing to attend a talk is missing the opportunity of attending other talks in the same time frame, and furthermore, the description available about the talks is very limited (e.g. title of the talk) making it much hard to choose. All such circumstances, more often than not, lead young researchers to miss important talks and lose out on meeting their potential collaborators.

Situations like these are not confined to newbies, often there are situations even a senior researcher could find difficulty in finding his potential collaborators or attending important sessions at conferences. There are few systems available, which could help researchers, to get information about upcoming events and attendees at the event, but none of them provide real-time recommendations about the both potential collaborators and sessions to attend.

1.2 Thesis Goal

The goal of the thesis is to investigate the role of community and topics for context aware recommendations in conferences.
1.2.1 Realization

CAMRS-2 (Context Aware Mobile Recommender System Version 2) is an android mobile application which tries to address the problem mentioned above. CAMRS-1 or CAMRS [37] was developed on Academic Event Recommender System for Computer Scientists [29] (AERCS). AERCS is a visualizing tool built on DBLP dataset which visualizes coauthor network, citation network and on AERCS one can browse list of conferences and select particular series to see the detailed analysis and visualization of a community. CAMRS-2, which will likewise be built on AERCS, will be the extension of CAMRS in several ways. CAMRS-2 uses similar concepts of CAMRS, such as,

- **Recommender System**: Collaborative filtering techniques will be made use of to make recommendations about user potential collaborators and sessions that would interest him.

- **Context**: To make better recommendations several context factors will be considered. For instance, time factor is considered so that the system does not recommend events that have already ended.

- **Social Network Analysis**: Link prediction techniques will be used to predict the user’s potential collaborators and the networks are further analyzed by various analysis metrics to make better predictions.

In addition to the above concepts, CAMRS-2 employs two more techniques which are considered as a vital part of the system.

- **Communities**: Network clustering algorithms are used to detect communities among co-author network, citation network and Event co-participation network. The detected communities play the main role while making different recommendations.

- **Topic Modeling**: Statistical approaches are performed on corpus that belongs to the community to model topics and the detected topics are annotated to the respective community. Topic detection avails on making recommendations depending on the research area of an author.

1.3 Thesis Structure

The rest part of thesis is structured as follows:

- **Chapter 2**: The chapter is dedicated to the state of art in the areas of Social Network Analysis, Recommender Systems, Context Awareness, Community Detection, Topic Modeling. Several techniques and algorithms which serve the purpose of the system are discussed in detail. The chapter is concluded with mentioning the existing systems which try address our thesis goal, but fall short in many areas.

\[http://bosch.informatik.rwth-aachen.de:5080/AERCS/\]
\[http://dblp.uni-trier.de/\]
• **Chapter 3**: The conceptual approach of CAMRS-2 is presented in this chapter. The chapter begins with modeling communities according to ANT model, and later the life cycle of a conference is described in detail. An use-case diagram with possible actors and their actions of the system is put into this chapter. Finally, the recommendation approach, which can be considered heart of the chapter is extensively explained.

• **Chapter 4**: System Architecture and Implementation are of primary focus in this chapter. All the tools, technologies, and libraries used during the development of the system are discussed. The system architecture is explained with the help of diagrams. The data sources which are used during the communication of system client and server are mentioned. At the end, the process of making recommendations and displaying to the user is described in detail.

• **Chapter 5**: In this chapter the method used to evaluate the system is presented. Later on, the results are shown and a comparison is made with the previous system output.

• **Chapter 6**: The report is ended with a conclusion describing the process of the research and implementation. Few proposals regarding the future work are mentioned along the way.
1 Introduction
2 State of the Art

In this chapter we shall have a theoretical understanding of SNA, recommender systems, context awareness, community detection, and topic modeling. Prevalent algorithms and techniques of each concept are explained in this chapter.

2.1 SNA

In this section, we will discuss about social network, social network analysis, communities and communities of practice. We will conclude this section with how link predictions can be made in social networks.

Social networks are made of individuals or organizations or any social entity which are related to one and another. The relation between the entities is socially meaningful, such as friendship, co-authorship, etc, [25]. Social networks are distinct from one discipline to the another, thereby having different definitions among different fields of study. The most general definition is from Wasserman since it describes social network in the natural world. He defined social network as, “Social network is a map of individuals and the way how they are related to each other” [59].

Social Network Analysis (SNA) is an interdisciplinary field of research, which focuses on relationships between social entities in the social network. SNA assumes nodes in the network as actors and links between them as relations and measures the flow between different nodes. SNA can be performed by collecting data from the social network which allows understanding the details of network. Social network data, which is collected from social networks, consists of various elements [25]. From the definition by Wasserman and Faust [59], social network data can be considered as a social relation system which can be characterized by a set of actors and their social ties. There are several ways to retrieve social network data, and the most common ways are observations, interviews and questionnaires.

The important factor of analyzing social network is to realize the essential facts about the network. By analyzing, it is possible to comprehend important, powerful and influential actors among the network. There are three common metrics, which could determine certain facts about social network. They are degree, closeness and betweenness.

• Degree centrality in a network is how many ties does an actor has. The more the number of ties, more powerful the Actor is considered. This is because, having more ties provides many alternatives to satisfy the needs of the Actor. This way actor is less dependent on other individuals, when he requires access to a resource.
• Closeness centrality denies considering only degree centrality. Degree centrality considers only the number of ties of an actor but does not consider the number of indirect ties to all the other actors. If other actors can’t have any ties with outer network, then the actor is considered central only in local neighborhood. Closeness centrality emphasizes the distance from an actor to all other actors by considering the distance from each actor to the others.

• Betweenness centrality measures the importance of an actor by counting the number of shortest paths the actor is part of.

These metrics are used by organizations when they plan to restructure the personnel. Restructuring the personnel network without analyzing may result in severe impact. For instance, an organization relocates few of its personnel without analyzing the network, and it happens that most of the personnel relocated have high betweenness. People with high betweenness usually act as a bridge in the network, removal of these employees evidently results in jeopardy of an ongoing process at the company. SNA is also helpful to determine potential connections, which would help in decision making regarding changes in structure of the network.

2.1.1 Communities and Communities of Practice

As long we are discussing social networks and social network analysis, it is worth to have a brief note about communities and communities of practice, since communities are the distinct parts which form a network and have also been studied widely. However, we shall discuss in much detail why we detect communities and how can we detect them in complex networks in the section 2.4.

Community, in general terms, is constituted by people who interact with each other and live in some proximity. Communities can be considered parts of network with a common feature. One of the examples of a community is neighborhood. People who live in a neighborhood form a community, since they all share similar location and interact with each other.

Community of Practice (CoP) is much more specific than a community, CoP are formed (intentionally/unintentionally) by people who engage often to collectively learn, share and improve their common interest. The term CoP was coined by Etienne Wenger an educational theorist and practitioner. Wenger defined CoP as, “Communities of practice are groups of people who share a concern or a passion for something they do and learn how to do it better as they interact regularly” [60]. From the definition, it is clear that not every community is a CoP. As said before neighborhood can be considered as a community, but from the definition neighborhood is not a CoP. According to Wenger there are three crucial characteristics for CoP.

• The domain: People who formed CoP share common domain of interest.

• The community: People in CoP often interact with each other, help each other, share information and perform activities together to pursue their passion towards the domain of interest.
2.1 SNA

- **The practice**: Members in CoP are called practitioners. The practitioners interact regularly and share resources, such as, tools, experiences, and domain related information.

The appropriate example of CoP is communities formed in co-author network. The communities in co-author network are formed between members of the network who co-author frequently. These communities possess the characteristics of CoP. Members in these communities have a common domain of interest, they often collaborate with each other, interact frequently by attending conferences and publish research papers by which they try to pursue their passion.

2.1.2 Link Prediction in Social Networks

Social networks are highly dynamic objects; they drastically grow over time by the addition of new edges, which signify the appearance of recent interactions in the underlying social structure. The process of predicting the future links in social networks can be modeled as *link prediction problem*. Link prediction in social networks has been extensively studied recently. For example, in co-authorship network, link prediction techniques try to predict the links that might appear in future, implying authors who haven’t co-authored before could publish paper together.

The link prediction problem can be formally described as follows. Given, two times $t$ and $t'$, where $t < t'$, and a snapshot of a social network’s graph $G = (V, E_t)$ where the set $V$ represents the individuals and the set $E_t$ represents the ties existing among them at time $t$, then we try to construct the set $G[t, t']$ which contains edges that are not present initially in $G$ and are expected to appear during the interval $[t, t']$.

There exist different methods for link prediction, and they can be classified into two classes: neighbor-based and path-based. Both use adapted techniques from graph theory and social network analysis. All these methods assign connection weight $score(x, y)$ to pair of nodes $<x, y>$, based on the input graph $G$. The pairs with high score have the more probability of forming a link in the future.

**Neighbor-based Methods**

For node $x$, let $\Gamma(x)$ denote the set of neighbors of $x$ in $G$. Many approaches have been proposed on the basic idea that two nodes $x$ and $y$ are more likely to form a link in the future if their sets of neighbors $\Gamma(x)$ and $\Gamma(y)$ largely overlap.

- **Common Neighbors**: The $score(x,y)$ is calculated by determining the number of neighbors both $x$ and $y$ have in common.

  $$score(x,y) = |\Gamma(x) \cap \Gamma(y)|$$  \hspace{1cm} (2.1)

- **Jaccard’s Coefficient**: This measure compares number of common neighbors $x$ and $y$ have to number of neighbors $x$ and $y$ have together.

  $$score(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$ \hspace{1cm} (2.2)
• **Adamic/Adar** \[3\]: This measure gives more weight to the nodes with low degree. Entailing, if a person known to both \(x\) and \(y\) is isolated will be given much weight than person who has several connections.

\[
score(x, y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{\log |\Gamma(z)|}
\]

(2.3)

### Path-based Methods

Unlike neighbor-based methods, path-based methods rely on shortest-path distance between the two nodes to determine the connection weight.

- **Katz\(\beta\)**: This measure sums weights of all paths between two nodes exponentially damped by length \[27\].

\[
score(x, y) = \sum_{l=1}^{\infty} \beta^l \cdot |\text{paths}_{x,y}^{<l>}|
\]

where \(\text{paths}_{x,y}^{<l>}\) is set of all length-\(l\) paths from \(x\) to \(y\).

- **Graph Distance**: The link is predicted between the nodes depending upon the length of the shortest-path between pair of nodes.

### 2.2 Recommender Systems

People rely on recommendations in everyday life to make decisions. The sources for recommendations come in different formats, such as, spoken words from a person, column in a newspaper and in many other ways \[56\]. To augment the recommendation process and to make user time more productive recommender systems are built. Recommender systems \[46\] are the software tools which are built to predict items which would interest the user and recommend those items according to the probability of user interest.

The importance of recommender systems has been increased considerably in recent times. The best way to depict the current picture would be to introduce about **Netflix Prize** \[10\]. Netflix, an American company, which provides streaming media on-demand, conducted an open competition popularly known as Netflix Prize. The competition was to build an algorithm which can predict the ratings of movies given by the Netflix users. Netflix provided competitors with large datasets containing the user id, movie id, grading and date of grading. The provided sets have only half of user ratings and competitors are supposed to built an algorithm which would predict the rest items ratings. The submitted predictions are compared to the true grades, and the group with least RMSE \[56\] was awarded 1 million dollars. Though such competition is conducted by only Netflix, the usage of recommender systems is prevalent among many other companies. The most prominent companies like Amazon \[34\], ebay \[49\], Pandora use recommender systems to recommend items to the users.

Generating recommendations can be done using three different approaches: i) **Collaborative Filtering Methods** ii) **Content-based Methods** iii) **Hybrid Methods**, and we will discuss about these in detail.
2.2 Recommender Systems

2.2.1 Collaborative Filtering

Goldberg et al. who built the first recommender system Tapestry [19], coined the term Collaborative Filtering (CF). The basic assumption of CF is that, if two users $x$ and $y$ rate an item similarly, or posses similar behavior, are expected to rate other items similarly [20].

To predict the ratings of items, CF techniques maintain the database of users and items, commonly referred as User-Item database. The user-item database consists, list of $m$ users $u_1 \ldots u_m$, list of $n$ items $i_1 \ldots i_n$, and for every user, $u_i$, list of items $I_{ui}$ which user $u_i$ has rated. There are two different CF techniques which use user-item database to predict the ratings.

Memory-Based CF Techniques

Memory-based are widely used in commercial systems like Amazon.com, they are easy to implement and highly effective [22]. Memory-based CF algorithms maintain user-item database and every user will be part of a certain group. The algorithm tries to determine neighbors of the active user and predict items which user would prefer.

To predict the items for user, these algorithms compute the similarity between either items or users. Item-based CF algorithm is used to compute similarity between items and User-based CF algorithm is used to compute similarity between users. One of these approaches will be used to predict the ratings of items for a user.

To compute the similarity between items $i$ and $j$, item-based algorithm determines users who have rated both the items and apply similarity computation $w_{i,j}$, between items which are co-rated by the users [48]. To compute similarity between users $u$ and $v$, user-based algorithm calculates similarity $w_{u,v}$ between users $u$ and $v$, who have both rated same items. The following are common measures which are used to compute similarity between either users or items.

- **Correlation based similarity:** Pearson correlation based similarity is used to compute similarity between either users or items.

  For user-based algorithm, the Pearson correlation between user $u$ and $v$ is

  $$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}}$$

  where $I$ is set of items which are co-rated by both the users $u$ and $v$, and $\bar{r}_u$ is the average rating of all co-rated items by user $u$.

  For item-based algorithm, the Pearson correlation between item $i$ and $j$ is

  $$w_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2} \sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}}$$

  where $U$ is set of users who have rated both the items $i$ and $j$, and $\bar{r}_i$ is the average rating of item $i$ by the users in set $U$. 

9
Vector cosine based similarity: Given, $R$, a $m \times n$ matrix, where $m$ being the users in database and $n$ being the items in the database. The similarity between items $i$ and $j$ is computed as cosine of the $m$ dimensional vectors of corresponding $i$th and $j$th column of matrix, $R$.

Vector cosine similarity between items $i$ and $j$ is

$$w_{i,j} = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\| \cdot \|\vec{j}\|}$$

(2.7)

where “$\cdot$” denotes dot product of the vectors.

The problem with vector cosine based system is that, it does not consider different scale. To overcome this drawback, adjusted cosine similarity is proposed which is similar to Pearson correlation.

As said earlier these algorithm initially find the neighbors of the user and try to predict the ratings of the items. Once neighbors are determined using similarity computation, nearest neighbors are chosen depending on the similarity. Then weighted aggregate of nearest neighbors ratings is used for predictions for the user.

For a user $a$ to make a prediction on item $i$, weighted average of all the ratings by users in set of nearest neighbors of $a$ is calculated by the following formula:

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u) \cdot w_{a,u}}{\sum_{u \in U} |w_{a,u}|}$$

(2.8)

where $U$ is set of nearest neighbors of user $a$, $w_{au}$ is the similarly between $a$ and $u$, and $\bar{r}_a$, $\bar{r}_u$ are the average ratings of users $a$ and $u$ on all other rated items.

Once the predictions are made the algorithm recommends user with the Top-N items that would interest the user.

Model-Based CF Techniques

Model-based CF algorithms are developed to overcome the shortcomings of Memory-based algorithms [7, 12]. These algorithms initially develop a model of user-ratings and provide item recommendations. These algorithms adopt probabilistic approach and predict the value of an item depending upon other items, which user has already rated. The model can be built by different machine learning algorithms, couple of them are, Bayesian Belief Net CF Algorithms and Clustering CF Algorithms.

Bayesian Belief Net (BN) CF Algorithm: Bayesian Belief Net is a directed, acyclic graph $<N, E, \Theta>$, which consists nodes $n \in N$ which represent random items and the nodes are connected with directed edges $e \in E$ (the edges are directed from parent to child). The edges between nodes represent probabilistic association between variables and a quantifying probability table $\Theta$ is used to show how much a node depends on its parents [13].

The Simple Bayesian algorithm: The simple bayesian algorithm uses naive Bayes classification for collaborative filtering tasks. The features are assumed to be
independent given a class \( [55] \), and given all the features the probability of a certain class \( p(C_j | f_1, f_2, \ldots, f_n) \) can be determined by computing \( p(C_j) \Pi^m p(f_i | C_j) \), where \( p(C_j) \) and \( p(f_i | C_j) \) are estimated from training data and \( C_j \) refers to class \( j \) and \( f_i \) refers to feature \( i \). The ratings of an item for a particular user can be predicted by the following equation.

\[
\text{class} = \arg_{j \in \text{classSet}} \max p(\text{class}_j) \Pi_o P(X_o = x_o | \text{class}_j) \tag{2.9}
\]

the above formula computes probabilities and returns the class with highest probability as the predicted class. It uses incomplete data and to calculate the probability it uses observed data (the values with subscript \( o \) are the observed data).

**Clustering CF Algorithms**: For a given collection of data objects, clusters are formed such that data objects that belong to a cluster are similar between them and dissimilar to any other cluster data objects \([21]\). The similarity between the data objects can be computed using Pearson correlation and Minkowski distance.

### 2.2.2 Content Based Recommender Systems

Often user interacts with the items presented by the system which interests him, though the process looks transparent Content Based Recommender System (CBRS) maintains the items description user interacted and builds user profile to provide recommendations. CBRS initially focuses on analyzing items description because recommender systems vary upon the items representations \([42]\). An item can be described by structured data, which is set of attribute values, or unstructured data, which contains unrestricted text and cannot be understood easily by the system. For unrestricted text, CBRS uses \( tf^*idf \) \([47]\) to create structured representation of unrestricted text. The items in real world are mostly semi-structured, for instance, hotels have set of attributes like name, location, cuisine which have structured attribute values, but reviews about the hotels contain unrestricted text.

Once CBRS analyzes items which user has interacted, by their description it tries to build the user’s profile. User profile has various types of information, model of user’s preference and history of user interaction with the recommender system are two of them. The history of user will be provided as training set to the machine learning algorithms which create the user model. Creating of user model basing upon history of his preferences is a form of classification learning \([42]\). After creating a user model, given a new item, the recommender system predicts if the item will interest the user. Many classification algorithms try to predict the probability of user liking the new item and provide the list of items sorted according to the probability.

Content Based Recommender Systems are implemented to overcome the drawbacks of CF techniques such as, cold start problem \([56]\), sparsity \([56]\). However, CBRS faces its own limitations; the recommendations are relatively poor to that of CF recommender systems if the content information isn’t sufficient enough.
## 2 State of the Art

### 2.2.3 Hybrid Recommender Systems

Hybrid Recommender System combines two or more recommendation techniques to overcome individual drawbacks and to gain better performance [6]. Depending on different combinations of recommendation techniques there are different hybrid techniques. They are briefly presented in table 2.1 [13].

For instance, when CF recommendation techniques and CB recommendation techniques are combined, Hybrid Recommender system builds user profile basing upon his previous history, and simultaneously tries to find the similar users who have rated items similar to that of active user. Later the recommendations are produced using user profile and user neighbors ratings on new items. Though Hybrid methods overcome the drawbacks of individual recommender systems, implementation of Hybrid recommender systems is highly complex.

### 2.3 Context Aware Recommender Systems

In this section the details about context and its importance in recommender systems is described.

#### Context

Context is widely studied among different disciplines and computer science is one of them. Since its wide use, context has several definitions in every discipline; eventually context now has over hundred different definitions [8]. In general terms, context is something used by humans, to make the conversation less ambiguous. Context has different definitions in ubiquitous and mobile computing too. Initially, context was defined as the location of the user, the people around the user, the objects around, and changes in these elements. Later the definition was refined several times, but we shall concern the closest definition in the computing environment [2].
2.3 Context Aware Recommender Systems

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

From the definition, it is clear that the context is not the whole information about a certain entity, but the information that is relevant to the situation. For example, age can be considered as useful information when an application tries to suggest movies, but the same information is not pertinent for an application which recommends a pizzeria.

Context Awareness

Similar to context, context-awareness has several definitions. The first definition for context-aware application was from Schilit and Theimer [50]. Few other researchers [23] [41] defined context aware computing as the ability to sense, locate and respond accordingly to the user environment and the device themselves. The following is the most general definition for context-aware computing, given by Dey et al. [2].

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’s task.

Context awareness has been used in several domains to make the systems produce better results. Recommender Systems is no exception in using context awareness for producing better recommendations [5]. Adonavicius et al. [4] demonstrated that contextual information helps to increase the quality of recommendations in certain settings.

A typical recommender system would start building initial set of set ratings already available to the system, and then predicts the ratings of other items with rating function $R$.

$$R : User \times Item \rightarrow Ratings \quad (2.10)$$

The above recommender system is a typical two dimensional; it considers just users and items for recommendation process. We can turn the rating function $R$ into three dimensional by including context. Revised $R$ is,

$$R : User \times Item \times Context \rightarrow Ratings \quad (2.11)$$

The contextual information is composed of certain factors. Each factor has set of attributes which are used by recommender systems to provide better recommendations.

The required contextual information can be obtained in several ways. Few of them are, by explicitly requesting user required information or implicitly determining, such as finding user current location from his the co-ordinates or time from the request time-stamp. The other possible way is inferring the contextual information by using data mining methods on user history available to the system.
24 Community Detection

We have briefly discussed in the section 2.1.1 about communities and how they are different from community of practice. In this section, we will discuss why we need to find a community structure in a network and how the community structure is detected in a network.

Community Structure is a common property of many networks, which can be described as the division of network nodes into groups, such that connections inside the group are dense than connections between the groups [39]. There are several advantages by detecting community structure in a complex network. The whole network can be easily understood, summarized and visualized by detecting communities [39]. It is quite easy to organize the network as parts rather than handling it altogether. Community structure allows us to find the underlying facts about the network, such as, by detecting community structure in a citation network one can find related papers on a topic. For visual understanding of network, community and community structure, look at figure 2.1 [39].

2.4.1 Detecting Communities

The two main approaches to study community structures in a network are graph partitioning and hierarchical clustering. However, hierarchical clustering is preferred over graph partitioning while detecting communities in complex networks like co-authorship network, citation network, etc. Since, with graph partitioning user can fix the number and size of the groups (communities), but we desire that the method to determine the number and sizes of the groups basing upon the structure of the network.

Hierarchical clustering is broadly divided into two classes [39], agglomerative and
2.4 Community Detection

**Agglomerative** is an approach in which similarity between the vertex (nodes) pairs is calculated, and edges (connections) are added to the pairs which have high similarity. The network is initially empty with n vertexes and no edges, and the method adds edges to pairs with high similarity, and this procedure is halted at any point, and the resulting components are communities.

**Divisive**, on the other hand, works the other way around. Divisive methods start with the whole network and remove edges between the pairs with the least similarity. Divisive methods overcome the drawback of Agglomerative methods, which have the tendency, to form core communities and leave the periphery.

Both the agglomerative and divisive methods can be represented by a form of a tree or dendrogram. Agglomerative methods start from the bottom of the tree and Divisive start from top of the tree.

After detecting the communities in a network, we need to examine the quality of the community structure. To quantify the quality of community structure, Newman et al. proposed a metric called modularity, $Q$ [39]. The modularity $Q$ for a partition into $q$ communities of arbitrary size is defined formally as

$$Q = \sum_{i=1}^{q} (e_{ii} - a_i^2)$$ (2.12)

with

$$a_i = \sum_{j=1}^{q} e_{ji}$$ (2.13)

$e_{ij}$ represents fraction of edges between nodes of group $i$ and $j$. Thereby, $e_{ii}$ represents fraction of edges connection nodes in group $i$ internally. $a_i$ denotes the overall fraction of edges connecting to nodes in $i$ and $a_i^2$ corresponds to the expected fraction of internal edges given a random assignment of nodes into communities. From the definition of community structure [39], the value of $Q$ represents the quality of community structure. The value of $Q = 0$ when every node is a community, however, strong community structures have $Q$ value closer to 1 [39]. A decent community structure has $Q$ value with in the range of $0.3 - 0.7$.

Table 2.2 [44] presents different hierarchical clustering algorithms and their respective time complexities. Two complexities are presented for few algorithms; first one represents worst case time and the second in case of sparse graphs. $M, N$ represent number of nodes and number of edges in the network respectively and $d$ is the depth of the dendrogram.

### 2.4.2 Greedy Approach

As seen in the table 2.2 there are different approaches in detecting community structure in a network. However, many of those approaches are not suitable for large networks [16]. Early approaches like spectral partition [18], hierarchical clustering [51] are only suitable for specific graphs and perform rather poorly in many other cases [10].
<table>
<thead>
<tr>
<th>Author</th>
<th>Year</th>
<th>Time Complexity</th>
<th>Short Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortunato</td>
<td>2004</td>
<td>$O(N^3)$</td>
<td>Information centrality</td>
</tr>
<tr>
<td>Zhou and Lipowsky</td>
<td>2009</td>
<td>$O(N^3)$</td>
<td>Brownian particles</td>
</tr>
<tr>
<td>Pons and Latapy</td>
<td>2004</td>
<td>$O(MN^2)(O(N^2 \log N))$</td>
<td>Random walks</td>
</tr>
<tr>
<td>Newman</td>
<td>2004</td>
<td>$O((M + N)(O(N^2)))$</td>
<td>Greedy optimization of modularity</td>
</tr>
<tr>
<td>Newman and Girvan</td>
<td>2004</td>
<td>$O(M^2N)(O(N^3))$</td>
<td>Greedy optimization of modularity</td>
</tr>
<tr>
<td>Girvan and Newman</td>
<td>2002</td>
<td>$O(M^2N)(O(N^3))$</td>
<td>Edge Betweenness</td>
</tr>
<tr>
<td>Duch and Arenas</td>
<td>2005</td>
<td>$O(N^2 \log N)$</td>
<td>Extremal optimization (of modularity)</td>
</tr>
<tr>
<td>Radicchi et al.</td>
<td>2004</td>
<td>$O(N^2)$</td>
<td>Edge-clustering coefficient</td>
</tr>
<tr>
<td>Donetti and Munoy</td>
<td>2004</td>
<td>$O(N^3)$</td>
<td>Spectral analysis</td>
</tr>
<tr>
<td>Clauset et al.</td>
<td>2004</td>
<td>$O(Md \log N)(O(N \log^2 N))$</td>
<td>improved version of Newman</td>
</tr>
<tr>
<td>Wakita and Tsurumi</td>
<td>2007</td>
<td>$O(Md \log N)(O(N \log^2 N))$</td>
<td>improved version of Clauset et al.</td>
</tr>
</tbody>
</table>

Table 2.2: Time Complexities of different hierarchical clustering algorithms

The most often used algorithm proposed by Newman and Girvan also falls behind when used for large networks. This is due to the high time consumption of the algorithm, which has an undesirable time complexity of $O(m^2n)$ and $O(n^3)$ on a sparse network.

To overcome the above mentioned hindrances, Clauset and Newman proposed a new algorithm based on the greedy optimization \cite{38} of modularity. The algorithm in \cite{38} uses greedy optimization, in which every vertex is initially a community and later repeated joins are made among those communities which result in highest increase in modularity $Q$. The main idea of greedy optimization is to store the adjacency matrix of the graph as an array of integers and merge pairs of rows and columns are as the corresponding communities. However, this approach \cite{38} is not time and memory efficient in case of sparse graphs since the merging and storage of matrix involves a lot of elements with value 0. The algorithm proposed by Clauset and Newman \cite{16} improves the time and memory efficiency of the previous algorithm by eliminating needless operations.

In the latest algorithm, the network is imagined as a multi-graph, in which every community is represented as vertex, many edges connect one vertex to another, and edges internal to the communities are represented as self-edges. In the adjacency matrix of the multi-graph, joining of the two communities $i$ and $j$ would replace the $i$th and $j$th rows with their sum. In the approach \cite{38}, the same process is done explicitly on the entire matrix. However, in case of a sparse matrix the operation performed in approach \cite{38} can be carried out efficiently by using data structures. Since calculating $\Delta Q_{ij}$ and finding the pair $i,j$ with the largest $\Delta Q_{ij}$ is time consuming, the current approach maintains and updates a matrix of value of $\Delta Q_{ij}$. The $\Delta Q_{ij}$ is only stored between communities which are joined by one or more edges, since joining communities with no edges would not alter $Q$ in anyway.

The algorithm in total maintains three data structures, \textit{i}) The sparse matrix as mentioned before, which contains $\Delta Q_{ij}$ for each pair $i,j$ of communities which have at least one edge between them. \textit{ii}) The approach maintains a max-heap $H$ which contains the largest element of each row of the matrix $\Delta Q_{ij}$ together with the community
labels $i, j$. iii) Finally an vector array with elements $a_i$ is used.

Before starting the algorithm $\Delta Q_{ij}$ and $a_i$ values are initialized as follows,

$$
\Delta Q_{ij} = \begin{cases} 
1/2m - k_i k_j/(2m)^2 & \text{if } i, j \text{ are connected,} \\
0 & \text{otherwise,}
\end{cases}
$$

and

$$
a_i = \frac{k_i}{2m}
$$

where $m$ is number of the edges in the graph, $k_i$ is the degree of the vertex $i$.

The algorithm works as follows:

1. Initially values of $\Delta Q_{ij}$ and $a_i$ are computed, and max-heap is populated with the largest element of each row of the matrix $\Delta Q$.
2. The largest $\Delta Q_{ij}$ from $H$ is selected, corresponding communities are joined, matrix $\Delta Q$ is updated along with max heap $H$ and $a_i$ and $Q$ is incremented by $\Delta Q_{ij}$.
3. Step 2 is repeated until only one community remains.

Depending on the joining communities and labeling the combined communities, the approach has few rules that can be followed. Over the course of this approach $Q$ has single peak and after the largest $\Delta Q_{ij}$ becomes negative all the $\Delta Q$ can only decrease.

Compared to many other algorithms, the approach from Clauset and Newman is time and memory efficient. This algorithm is reported to run in time $O(m d \log n)$, where $m$ is number of vertices, $n$ is the number of edges and $d$ is the depth of dendrogram. In the sparse network the run time of the algorithm is $O(n \log^2 n)$. For networks, such as, co-authorship network and citation network which have over million vertices and around 10 million edges, this algorithm is believed to be efficient in detecting community structure in the network.

### 2.5 Topic Modeling

In this section we will discuss about topic modeling of text documents and how tagging data can help in various ways.

Over past many years vast amount of information is produced and stored in different forms. Consequently, it has been incredibly difficult to organize, analyze and find required data. The process to handle large amount of data (text) and label them with the respective topics can be characterized as topic modeling. Topic modeling provides a view of themes at corpus-level, with the best part being it can be automated. However, automated topic modeling on other form of data (video, audio, image, etc) is far from reach.

Topic modeling is usually addressed by different terms topic detection, topic discover, topic clustering are few of them. Topic modeling has produced fruits.
over past few years because of the recent extensive research\cite{26, 54}. Algorithms which model topics of text documents can be perceived as two-phased. 

1) extract keywords from either corpus or from single document 

ii) model topics using detected keywords.

### 2.5.1 Various Approaches

The extraction of keywords can be done using one of four approaches\cite{28}.

1. **Simple statistic approach:** Statistical information of words is used by these methods to detect the keywords in the document. These methods are simple, easy and input requirements are limited, furthermore they don’t need training data. These methods focus on the term frequency, position of keywords rather than the linguistics features of the text. Few of the statistic methods are PAT-Tree, word co-occurrences\cite{61, 36}, word frequency\cite{52}, etc.

2. **Linguistics approach:** These methods focus on linguistic features, such as, parts of speech, syntactic structure (e.g. NP-chunks-appropriate nouns for content description). Lexical analysis, Syntactic analysis, disclosure analysis and more are included in linguistic approach.

3. **Machine learning approach:** Machine learning algorithm are supervised learning models. They consider extracting keywords from a document as a classification problem. These methods build a model by learning from the training set and built model is used to find if the new document contains keywords or ordinary words. The most common machine learning algorithms are, naïve Bayes, SVM, LDA\cite{11}.

4. **Hybrid approach:** The hybrid approach is to combine the above mentioned approaches or use additional available data (e.g. html tags)\cite{24}.

Modeling the documents with topics, after the extraction of keywords is relatively easy task. In fact, many approaches consider the extracted keywords as the topics and label the documents with the topics (keywords) the document contains. Few approaches try to filter the keywords from the keyword-set depending upon its frequency and label the documents with the remaining keywords in the keyword-set.

### 2.5.2 Topic Modeling from Titles of Documents

Statistical methods are predominately used for extracting of topics from large volume of documents. The main reason behind this is, the other popular approaches like LDA model\cite{11} need pre-specified number of latent topics and manunal topic modeling. LDA model becomes highly impractical when we desire to extract topics from millions of documents. Unlike LDA model, statistical approaches will need no prior knowledge of topics, and they are also time efficient.

Statistical approaches have their unique ways in determining the topics of documents or corpus. Algorithms like FTC and HFTC\cite{9} and Apriori-based cluster documents on frequent itemsets. These approaches assume documents as a bag of words and find
frequent itemsets. However, the semantic information is not well preserved. Zhou Chong et al. \cite{15} have presented work that preserves semantics while forming topics. The authors consider a window within which itemsets are found which are candidates for the topics. The relative position of the words within the window are considered insignificant.

The statistical approach proposed by Shubankar et al. \cite{52} tries to extract the key-
words from the titles of the corpus. This approach does not consider the title bag of
words, and thereby does not lose semantics. However, it does not consider the relative
position of words like Zhou Chong et al. \cite{15} and might lose semantics in few cases.
This approach believes that a title of a document gives a fairly good high-level descrip-
tion of its content \cite{52} and abstract of the document is never taken into consideration,
since abstract holds a lot of irrelevant phrases as noise \cite{52}. In the approach from
Shubankar et al. \cite{52}, a closed frequent keyword-set is formed top-down dissociation of
keywords from the phrases present in the titles of papers on a user-defined minimum
support. Before moving further, we shall discuss various terms defined in \cite{52} during
the formation of keyword sets and finding the closed frequent keyword-sets to form
the topics.

- **Phrase**: A phrase $P$ is defined as run of words between two stop-words in the
title of the document.
- **Keyword-set**: A keyword-set $K$ is an n-gram substring of a phrase.
- **Frequent keyword-set**: A keyword-set is said to be frequent if its count in the
corpus is greater than or equal to a user-defined minimum support.
- **Closed Frequent keyword-set**: A closed frequency keyword-set is defined as
a frequent keyword-set none of whose supersets has the same cluster of research
papers as it. Each closed frequency keyword-set represents a unique topic $T$.

Now the algorithm is boiled down to four steps,

1. **Phrase extraction**: In this phase all the phrases from the titles of corpus are
extracted. As defined above, phrase is run of words between two stop-words.
The comprehensive list of English stop-words consists of 671 words. Now every
paper is mapped to the corresponding phrases in the paper title. Later, the
mapping is reversed, so that, every phase is mapped to every research paper it
belongs.

2. **Keyword-set extraction**: A Keyword-set $K$ is a substring of a phrase $P$ as
defined. Since only the substrings of the phrases are considered, the relative
ordering is maintained, therefore, preserving the semantics of the phases. Be-
cause solely considering the substrings rather than power set of phrase, finding
keyword-sets requires $O(n)$ instead of $O(2^n)$.

3. **Frequent keyword-set formation**: After extraction of keyword-sets, frequent
keyword-sets can be formed basing upon the user-defined minimum support.
Frequent keyword-sets are keyword-sets which appear equal or more than times of
user-defined minimum support. The support of keyword-sets is calculated during
generation of keyword-sets from phrases. In the first phase of this algorithm
phases are extracted and in the next phase keyword-sets are extracted, during the second phase the support of every keyword-set is maintained and incremented accordingly. After the second phase is completed, the keyword-sets with less than user-defined minimum support are removed; and the remaining keyword-sets are frequent keyword-sets.

4. **Closed frequent keyword-sets as topics**: This is the final phase of the algorithm which detects the topics of the corpus. Since we have frequent keyword-sets now, we need to eliminate non-closed frequent keyword-sets, sets which have the similar amount of support to that of their super sets. For this task, frequent keyword-set are stored in a level-wise manner, with the number of keywords in keyword-set representing its level. Once frequent keyword-sets are at the disposal, for every keyword-set of length $i$, the algorithm is iterated over the list of keyword-sets of length $(i+1)$. If a $i$-length keyword is a substring of $(i+1)$-length keyword-set and the support of both are equal, we remove $i$-length keyword-set since it is non-closed. This phase is executed until no non-closed frequent keyword-sets remain. The remaining rest frequent keyword-sets are considered as closed frequent keyword-sets. These closed frequent keyword-sets are considered as topics of the papers and the topics are used as similarity measure to cluster the research paper.

After testing the algorithm on the research papers in DBLP dataset, with minimum support 100, the following are the top topics among the research papers. 1-length top topics are system, model, network, 2-length top topics are neural network, real time, sensor network and 3-length top topics are wireless sensor network, support vector machine, ad hoc network. However, one can receive different top topics depending on the assumption of n-gram of a word. For instance, ieee 802.11 can be either considered as a 2-gram keyword or can be represented as ieee 802 11 and considered as 3-gram keyword.

The approach from Shubankar et al. [52] has a wide variety of applications. Unlike traditional approaches like Apriori, current approach identifies topics by forming closed frequent keyword-set. The simplicity of the algorithm reflects in its performance during the execution. The deciding factor of the run time is the minimum support. The performance of algorithm improves as the support is increased.

### 2.5.3 TF-IDF

TF-IDF (term frequency-inverse document frequency), is a numerical statistic which indicates how important a word is to a document in a collection or corpus [61]. Tf-idf is often used in text mining and information retrieval. The tf-idf value increases with the number of times a word appears in a document but is offset by the frequency of the word in corpus, this ensures that more general words do not have higher value [1]. Search engines use a variation of tf-idf as a tool in scoring and ranking the document's relevance with a given user query.
2.5 Topic Modeling

TF-IDF is the product of two static terms, TF (Term frequency) and IDF (Inverse Document Frequency). Both TF and IDF can be calculated in various ways. The most trivial way of determining term frequency is finding the frequency of the term. The other ways include a boolean frequency, logarithmic frequency and normalized frequency. We shall discuss normalized frequency here as we use it.

Let \( t \) be the term and \( d \) be the document in which the term \( t \) is present, the term frequency \( tf(t,d) \) is given as

\[
\text{tf}(t,d) = \frac{f(t,d)}{|w \in d|}
\]

(2.16)

where \( f(t,d) \) is the frequency of term \( t \) in document \( d \) and \( |w \in d| \) is the total number of words in the document \( d \).

And inverse document frequency \( idf(t,D) \) is given as,

\[
idf(t,D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}
\]

(2.17)

where \( |D| \) (Cardinality of \( D \)) is the total number of documents in the corpus and \( |\{d \in D : t \in d\}| \) is the number of documents in \( D \), which have term \( t \) in them.

Since we have \( tf \) and \( idf \), \( tf-idf \) is calculated as follows

\[
\text{tfidf}(t,d,D) = \text{tf}(t,d) \times \text{idf}(t,D)
\]

(2.18)

The higher \( tfidf \) value is achieved by a term with high term frequency in the document and low document frequency in the whole collection of documents. Since \( idf \) is a logarithmic function its value is always above 0, and there by making whole TF-IDF value greater than or equal to 0 all the time. A term, which appears in more number of documents, tend to have the ratio inside \( idf \)'s logarithmic function close to 0, bringing \( idf \) and consequently \( tfidf \) values close to 0.

2.5.4 Tagging

As we discussed topic modeling allows us to determine topics of a document, but to further make use of the detected topics, we would annotate the document with its topics and the annotations are called as tags. In web 2.0 environment, Tag is defined as the metadata added to web resources. \[32\]. So it is clear that we could add tags not only to the text documents but to other kinds of data. Tagging allows data to be easily organized, well understood and reduces search effort without accessing the data content. There are many popular web services, such as, delicious, Flickr, YouTube, etc., allow the user to tag the data. Certain web services like IEEE and ACM which

2. www.flickr.com
3. www.youtube.com
4. www.ieee.org
5. www.dl.acm.org/
store text documents annotates the documents themselves. By tagging data, system neither have to just rely on title of the data or process the whole data file to return results to the user query and it also enables system and users to find all the related contents on a single topic. Recent developments show that tagging has turned into a trend on the web and it has been even deployed in recommender systems to produce better recommendations [57].

2.6 Existing Systems

There exist systems which recommend a researcher potential collaborators and few other systems that try to suggest the talks at conferences. Since the systems are closely related to our system we shall discuss them in detail. To conclude, the table 2.3 with comparison of the existing systems is provided.

2.6.1 Conference Navigator

Conference Navigator (CN) is a web system which recommends talks to the attendees at conferences. CN employs community and other social navigation system techniques. The system provides two options i) Schedule Browser and ii) Personal Schedule Planner (PSP) to users. The schedule browser option allows users/attendees to browse all the talks that take place, and to access this option an user need not register with the system. The PSP option can only be used by a registered user. A registered user can affiliate himself to the communities already created by other users, or can create a new community himself. However, an user is not restricted from browsing talks that belong to other communities. After either finding or creating a community, user can add talks to the community, which he believes are related to the community. A registered user can mark a talk recommended if it is related to the community otherwise mark it as not relevant. To justify his decision user can also annotate the talk with little description. If a user finds a talk interesting, he can add it to his schedule as well. When user tries to browse communities, he can either browse all the talks that are added to the community, or the top 10 recommended, annotated, visited and scheduled talks of the community.
2.6 Existing Systems

CN 2, which was built on CN system, has added few extra features. Users in CN 2 are allowed to add tags to the talks, and the system also displays the most active users among different communities.

2.6.2 CollabSeer

CollabSeer [6], an online system, which is based on the CiteSeerX [7] dataset, discovers collaborators based on structure of co-author network and a user’s research interests. CollabSeer performs vertex based similarity between the authors in the co-author network to find potential collaborators and lexical analysis to determine authors research topics of interest. The vertex based similarity is computed using three similarity modules; Jaccard similarity, cosine similarity and relation strength similarity [14]. KEA [62] algorithm is used to detect the key phrases from the documents. Later the key phrases that are detected from documents written by an author are associated to him. Finally, both the vertex similarity score and lexical similarity score are amalgamated to calculate the collaboration recommendations.

The user interface of CollabSeer allows the user to enter name of an author to learn about the author’s potential collaborators. By default, only vertex similarity score is used to produce the potential collaborators list. The recent strength similarity is used as default vertex similarity module, however, user can change it to either Jaccard similarity or cosine similarity. The system also presents the topic of interest of the author, when user clicks one of the topics, the system reproduces the recommendation list based on both vertex similarity score and lexical similarity score.

2.6.3 Lanyrd

Lanyrd [8] is a web service which allows users to manage conferences and speaking appearances. The system aggregates lots of information about the conference and individual sessions. To use the system an user has to login through his twitter [9] account, and he can schedule his calendar by selecting events he wants to attend or track. The tracking option feeds user with the updates of artifacts (e.g. video, slides) that are being uploaded to the system during the event. An user is allowed to create event and sessions, and he can choose his role either as speaker, attendee or tracker.

The Lanyrd system doesn’t employ any technique to suggest user which events to attend but suggests depending on the events scheduled by user’s twitter contacts. The only preference given while making suggestion is, the latest upcoming event is suggested first. If one of the user’s contact is connected to a session at an event in any way, the Lanyrd system suggests the whole event to the user. However, details are made clear why the suggestion has been made. Every user has the privilege to

---

6 http://collabseer.ist.psu.edu
7 http://citeseerx.ist.pdu.edu
8 www.lanyrd.com
9 www.twitter.com
tag an event or session with related topics, since it is believed it makes easier for all users who are trying to find events or sessions related to certain research area would be hugely benefited. The Lanyrd system maintains the record of the past events and sessions scheduled by the user, thereby allowing to access all the details and artifacts of any particular event.
3 Conceptual Approach

In this chapter, the conceptual approach of CAMRS-2 is presented. We begin the chapter with modeling communities and later academic events. This allows a clear understanding of the approach we adopt to build the system. To make it even clear, the scenarios are exemplified. We conclude this chapter with a use case diagram and an extensive explanation of recommendation approach.

3.1 Community Modeling

For better understanding the community properties the community model is derived according to Actor Network Theory (ANT) [31] model. ANT was developed by French scholars, Michel Callon and Bruno Latour. Digital social networks are meeting point for the social and technology. Social networks can be generalized by examining digital social networks in terms of ANT. ANT provides an approach in which people and objects are not distinguished. In ANT, a network is formed by actors and their relations. According to ANT, set of social, technical, textual and conceptual actors who are involved together in a certain activity form a network. The actors, in ANT, may stand not only for humans but also any other objects, connoting even a network is a type of actor if its reactions to the change in environment can be predicted. Actors in ANT have properties and relations. Each actor has its own properties and different types of relations with other actors. Special relations between actors can be described with $i^*$-dependencies. $I^*$-dependency is part of $i^*$ framework [63], which is to model relation between actors. There are other types of actors in ANT model, which compose a network. We shall discuss all these actors in relative to communities.

- **Researcher:** The researchers stand for members in ANT model. Members in ANT model are the persons existing in the network.
- **Community:** Community stands for network in ANT model. Communities are formed by researchers who are assumed as actors, and the relation between them could be co-authorship, citation and event co-participation.
- **Academic events:** Academic events are considered as medium in ANT model. Researchers use this medium to present their research work in many ways and publishing papers is the most common way.
- **Artifacts:** Artifacts are the objects created by the members using the medium. Publishing papers during the academic events is one example. The artifacts can be used to trace the links between the members.
3 Conceptual Approach

Figure 3.1: ER Diagram of Community
As said researchers are considered as members in ANT model. This allows us to further categorize members to dig the details of the community. The members can be categorized as [17],

- The **Key members** or **core members**, these people have a lot of relations in the community and can be considered powerful in terms of centrality.
- The **Peripheral members**, these are members in the network with least relations. These members are often newbies who have published few papers in the conference.

The community model represented in figure 3.1 makes certain details clear about the communities. Every Community (network in ANT model) has one or more researchers (members in ANT model), and a researcher can belong to more than one community. The communities are formed by the different relations between the researchers, who use Academic Event (medium), to create and present their artifacts (e.g. papers, posters). The community theme is multivalued, because communities are formed by hundreds of researchers, and every one of them works on one or more research areas. Combining all the members of community research areas would result in a certain set of topics the community is working together. We call these topics as community theme. Further, a community theme may have several topics, but it doesn’t imply that everyone in that community is working on all the topics, it is only assumed every member in the community is working at least one of the topics of the community theme.

### 3.2 Academic Event Modeling

Since, in addition to author recommendations, the system also provides event recommendations, it is important learn about conferences and their structure.

#### 3.2.1 Academic Event

Academic events [17] are of different types, we shall discuss each of them briefly. The most common academic event is **conference**, where researchers present their work. Conferences usually comprise of paper publishing, keynote presentations, panel discussions, etc. The other type of academic event is **symposium**, which is similar to conference but usually has more informal meetings between researchers. The **workshop** is a series of educational and work sessions, which usually have narrow scope, and the size of workshops is relatively smaller to that of conferences and symposiums. The **Doctoral consortium** is for doctoral students who try to share their work and learn by interacting with other students and senior researchers. In **summer school/winter school** presentations are given to PG students by established academics. Because of wide familiarity of conferences all the academic events are generalized as conferences, and hence we shall use the term conference instead of academic event.
3 Conceptual Approach

The life-cycle of a conference can be divided into a number of phases. Initially the Program Committee (PC) Chair sets up the conference location, time and decides the schedule of sessions and their respective topics. Conferences call for paper submission and researchers who are willing to submit a paper register themselves and submit the paper and details of the paper. Later the PC chair assigns PC members to review the submitted papers. After reviewing the papers PC chair either accepts or rejects the papers depending on the reviews from PC members. Simultaneously participants present their works in different formats, such as, keynote presentation, paper session, etc. At the end of the conference, the whole conference proceedings are made available to all the participants.

3.2.2 Session Formats

Every conference has a series of sessions, and the sessions are in different formats [37]. We used term “event” throughout the document to refer any type of session in a conference. As said, sessions are of different formats, and they differ from conference to conference. Every conference has it’s own kind of session formats, so to generalize, we speak of sessions formats which are more common.

- **Paper**: In paper sessions, papers are submitted by individuals or group of people together (if they have similar topic). These sessions will allow the researchers who submitted their papers for acceptance to present their work. Paper session usually lasts for 15 minutes.

- **Poster**: Posters are graphical presentations used to present research work. Posters usually illustrate the methods used and results obtained during the research. Typically a room is reserved for posters and fellow poster presenters wander around by posing questions to other presenters.

- **Panel**: A panel session has 3-5 people called panelists commenting on a certain topic. A session chair is responsible for moderating these panel sessions. Panels are considered more interesting than paper sessions, because every panelist takes a different perspective on a topic and their comments on a topic are later questioned by the chair and audience. Panel sessions continue for an hour or so.

- **Round Table**: Round Table sessions consist of several small groups of conference participants discussing a specific topic. Every round table has 5-8 participants involved and these sessions last about 20 minutes.

- **Keynote**: Keynotes are the sessions in which the keynote presenter explains the themes of current day’s upcoming events. All the conference participants attend keynote session to get the insights of upcoming events. Keynote speaker usually speaks from 45 to 60 minutes.

- **Workshop**: Workshop sessions are used to help the attendees to acquire skills or refine existing skills on different research practices. Often, certain exercises are conducted that allow the attendees to practice using the acquired skills.

As it appears every session varies from other, therefore the system is expected to make recommendation by taking session formats as input to make better recommendations.
3.3 Scenarios

There are several possible scenarios, but we will discuss the scenarios which are characterized by the actors of the system.

Recommendng Events

John who has recently started his research in community analysis is pumped up to attend a conference at Berlin. The conference at Berlin is holding events on several topics, and community analysis is one of them. John decides to attend an event which is about community analysis, but he doesn’t want to return by just attending one event related to his research. He wishes to know what other events are the researchers attending who have attended the same event as John did. Now he could use our system, which uses several techniques, to analyze what other events are the researchers attending whom John is interested in, and recommends list of events to John according to his interest.

Alan, who is similar to John, but has published a couple of papers in previous conferences and wishes to attend events which are being attended by researchers who are working on similar research area. For this, he could use our system and since he has published papers before, our system can recommend events attending by researchers who work on the same topic as Alan.

Recommendng Authors

Charlie and Barney are budding researchers who submitted a paper on Social Network Analysis in previous conference at Vienna. Charlie and Barney expected publishing paper would increase their chances of getting to know researchers and would help the publish even more papers in future conferences. But, Charlie and Barney could not collaborate with new researchers, because lack of a system which would help them in finding their potential collaborators. Our system is mainly useful to researchers like Charlie and Barney, the system tries to find researchers who are working on similar topics that of the duo and also further analyzes the potential chance a researcher would collaborate with these researchers. Taking all these aspects into consideration, the system recommends researchers with high probability of collaboration.

3.4 Use Case

Use cases are used to list the interactions between the Actor and the System. There is no specific way to design a use case, the primary consideration is to list the interactions and capture functional requirements. Actors in our system are the users and the administrators, and their respective interactions with the system are depicted in the use case diagram 3.2. We shall now discuss different interactions the actors can make with the system.
3 Conceptual Approach

3.4.1 User Roles and Interactions

- **Event Administrators and Organizers:** These are responsible for organizing and administrating the academic event. These actors are responsible for scheduling the whole academic event and make sure there are no hassles during the event. Administrators and Organizers have extra privileges regarding our system, they are allowed to *create and edit the event schedule* on the CAMRS system. This is due to the fact they built the whole schedule of the academic event and they have details about every session at their disposal. However, every administrator or organizer has to *login* into the system to prove their identity and make use of the system.

- **User:** The users are the participants at the academic event who have registered with CAMRS system. A user can prove his identity by *logging in* into the CAMRS system. After login a user can interact with the system and make the most out of the academic event. A user can *view the schedule* of the academic event, and the schedule is updated swiftly on the user device should there be any changes made by administrators or organizers during the event. A user can *get list of recommended authors* who are his potential collaborators, and in addition he can also view their profiles. Other useful interaction a user can make is, to *get interesting talks* during the academic event recommended by CAMRS system, and view the topic, duration and presenters at the talk. To make the CAMRS to provide better recommendations, a user need to reveal his current location by using *check-in* service of the system.

3.5 Recommendations Approach

Since, communities, conferences, topic discovery and social networks have been modeled for a clear understanding. We now use those familiar concepts to build the system which provides users with two different recommendations, i) Author Recommendation ii) Talk Recommendation.

The recommendations computed by the system are not trivial, the system under goes a lot of states and has a lot of transitions during the computation. To make the process clear and comprehensible, we discuss the process in regard with the figure 3.3 which displays the system states and respective transitions.

3.5.1 Bibliographic Data

The primary sources for the bibliographic data are DBLP[^1], CiteSeerX[^2] and EventSeer[^3]. DBLP provides XML record and DTD for its dataset. The dataset contains data of

[^1]: http://www.informatik.uni-trier.de/~ley/db/
[^2]: http://citeseerx.ist.psu.edu
[^3]: http://eventseer.net/
3.5 Recommendations Approach

Figure 3.2: CAMRS - 2 Use Case
Figure 3.3: Recommendation Process
3.5 Recommendations Approach

<table>
<thead>
<tr>
<th>Data</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Events</td>
<td>32813</td>
</tr>
<tr>
<td>Series</td>
<td>4410</td>
</tr>
<tr>
<td>Authors</td>
<td>1138661</td>
</tr>
<tr>
<td>Publications</td>
<td>1964081</td>
</tr>
<tr>
<td>Global Co-authorship network</td>
<td>5766571 links</td>
</tr>
<tr>
<td>Global Citation network</td>
<td>14260753 links</td>
</tr>
</tbody>
</table>

Table 3.1: AERCS Data Summary

various types of publications that include articles, inproceedings, books, thesis and several other publication types. EventSeer.net is a repository for Call For Papers (CFP), which has most of CFPs for conferences in Computer Science. EventSeer.net, in addition to past conferences, indexes to the upcoming conferences. CiteSeer is a scientific literature digital library and search engine which focuses primarily on the literature in computer and information science. CiteSeerX has several features which create citation index that can be used for literature search and evaluation.

However, CAMRS-2 system is not directly reliant on these primary sources. The system primarily depends on the data provided by AERCS. Which has extracted all the relevant data from the primary sources. From the DBLP XML datasets, AERCS has extracted the authors, co-authorship network, conferences details and list of papers for each conference. AERCS used JAXP parser to parse all the DBLP XML objects. Other information, which is unavailable from DBLP datasets, has been extracted from EventSeer.net. AERCS used open-source libraries to extract data from EventSeer.net since the data available from EventSeer.net does not have a structure like DBLP. The locations and time of events are retrieved from EventSeer. CiteSeerX has not only the publications but also meta-data of every publication (conference, journal, etc). AERCS has taken the meta-data of every publication and built the citation network. Once the data has been extracted from all these sources, AERCS combined them into one unique dataset.

3.5.2 Communities

As the required data is already available in AERCS database, we can continue to build communities from the available networks. As can be seen in Table 3.1, AERCS holds two different networks co-authorship network and citation network. Detecting community structure in such a large network needs a time and memory efficient algorithm. As discussed in 2.4.2 the algorithm proposed by Clauset and Newman has been proven time and memory efficient.

In the diagram 3.3 it is clear that we build two different types of communities, i) global communities ii) local communities.
Global Communities

The global co-authorship network $G$ can be assumed as a weighted undirected graph with a set of nodes and edges. The nodes in the graph are the authors, $A$, and the edges, $E$, are the co-authorship relations between the authors with weights $W$. The edge set $E$ consists of edges $\{e_{ij}, e_{jk}, ..., e_{mn}\}$, where edge $e_{ij}$ represents that authors $a_i$ and $a_j$ have collaborated in the past and $a_i, a_j \in A$. The weight set, $W$, has weights for every edge in the graph. The weight $w_{ij}$ of the edge $e_{ij}$ represents the number of times that authors $a_i$ and $a_j$ have collaborated together.

The global communities, $GC$, are formed by running the algorithm [16] on the global co-authorship network $G$. Before processing the network graph $G$, it must be refined by removing the self-loops and multiple edges between the nodes. To remove self-loops, all the edges similar to $e_{ii}$ from the edge set $E$ are removed. The core graph consists multiple edges between two nodes, such as, $\{e_{ij}, e_{ij}\}$ in the $E$, we shall replace them with single $e_{ij}$. However, not to lose any detail from the graph, we sum the edges weight and the result edge would have the weight of the total sum of all edges that existed between the two nodes.

After the production of refined global co-authorship network $G$, we can detect the communities in the network by employing the community detection algorithm [16]. The resulted communities are denoted as $\{gc_0, gc_1, ..., gc_m\}$. The properties of every community varies from other and every community provides distinguished information. Every community in $GC$ has distinct authors, an author cannot be part of more than one community.

Local Communities

As seen in the figure 3.3, local communities are formed by considering both the conferences and networks. The conferences present in the AERCS database belong to an event series. All the conferences are conducted as a part of their respective event series. For instance, ICWL 2012 in Romania, is a conference that belongs to the event series ICWL.

The primitive idea of local communities is to build communities with authors specific to every series. Hence, the local communities are primarily series communities. For every series in the AERCS database we build a series co-authorship network $S_i$, where $i$ is the id of series. Similar to global network $G$, series network $S_i$ can be imagined as an undirected weighted graph. The nodes are the authors and the edges are the co-authorship relation between the authors. The authors set is represented as $A_i$, where $i$ is id of series. The $A_i$ set consists of authors who have participated in at least one conference that belongs to the $i^{th}$ series. The edge set $E$ is the co-authorship relations between the authors in $A_i$. As you can notice the edge set $E$ is not confined to the relations that have occurred during the series, but during all the series. The main reason behind this approach is, choosing only series specific relations might result in a weak and undesirable community structure.
3.5 Recommendations Approach

In the same fashion as above the series network is refined and community detection algorithm is applied on the series network $S_i$ to detect the communities. The series community set $SC_i$, where $i$ is the id of series, might have one to many series specific communities. The series communities are represented as $\{sc_{ij}, sc_{ij}, ..., sc_{im}\}$, where $sc_{ij}$ is the $j^{th}$ community of series $i$. This procedure is performed on every event series, and every event series has different number of communities and different community sizes.

3.5.3 Community Theme

From the above process we now have two types of communities, global and local. Now we move further by determining the theme of these communities individually. By determining the themes, we can realize the major research areas of the authors in a community. Community themes provide invaluable information which can be used in various ways. In CAMRS-2, community themes are used to determine the similarity between them.

To determine the community themes, the system needs already existing communities, topics of publications, and a statistical approach to present key research areas of the communities. We employ the statistical topic model proposed in [52] which was presented in 2.5.2. We use this approach on the publications which are available in AERCS. This approach is best suited to our system since we only have the titles of publications and besides trying to discern topics from the entire document is impractical given the number of documents being about 2 million. By executing the algorithm, we get keywords of every paper from its own title. We shall modify the approach by not only storing the closed frequency-set keywords but all the frequency-set keywords.

Now that we have topics of the papers, we try to associate every paper to its respective community. Since AERCS holds the publication data, we can realize the authors of a publication without hassle. Initially for every global community publications of authors belonging to the community are determined. Once all the publications of the community are retrieved, keywords of every publication that belongs to the community are accessed and stored along with their frequencies and community id. This procedure is repeated on all the communities, and it is possible that one publication might be present in several communities.

For the series communities, however, we only consider the publications that are presented during one of the series conferences. This is due to the fact, considering all the publications of the series community authors might mislead the primary research area of the event series. Otherwise, the procedure is similar to that of global communities.

Though we have communities and its keywords, we do not want to rely on the frequency of keywords to determine the theme of the communities. Instead, we chose a statistic approach TF-IDF[^4] to determine the theme of community. To avail tfidf

[^4]: \url{http://www.tfidf.com/}
3 Conceptual Approach

approach, we shall make few assumptions for our scenario. Since, in our case, the communities have a set of keywords, we assume communities as documents and keywords as terms. The corpus would therefore be all the communities.

Let $k$ be keyword in the community $c$, then the term frequency can be computed as follows,

$$tf(k, c) = \frac{f(k, c)}{|w \in c|}$$ \tag{3.1}

where $f(k, c)$ is the frequency of keyword $k$ in community $c$ and $|\{w \in c\}|$ represents number of keywords (including duplicates) in the community.

The inverse document frequency is computed as follows,

$$idf(k, C) = \log \frac{|C|}{|\{c \in C : k \in c\}|}$$ \tag{3.2}

where $|C|$ is total number of communities and $|\{c \in C : k \in c\}|$ represents number of communities have the keyword $k$.

Since $tf$ and $idf$ values are present, we can compute tf-idf to determine the importance of the keyword.

$$tfidf(k, c, C) = tf(k, c) \times idf(k, C)$$ \tag{3.3}

The above procedure can be used to detect the important keywords in the global communities without modifying the approach. However, we need to be cautious when determining important keywords in series communities. While computing tfidf on keywords of series communities following modifications should be done. For a keyword $k$ in a community $c$ of a series, the $tf$ is ratio of frequency of keyword $k$ in the community to the number of words in the community. The $idf$ is the ratio in logarithmic function is number of communities that are formed from the series network to the number of times the keyword $k$ has appeared in other communities that belong to the same series.

3.5.4 Similarities Computation

To find the similarity between two authors, we compute three different similarity values between them. i) Co-authorship Similarity ii) Citation Similarity iii) Community Similarity.

To determine the first two similarities we use link prediction techniques.

- Co-authorship Similarity ($S_{co-authorship}(x, y)$): The co-authorship similarity is computed by Jaccard coefficient.

$$S_{co-authorship}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

where $\Gamma(x)$ represents authors who have collaborated with $x$ and $\Gamma(y)$ represents authors who have collaborated with $y$. If authors $x$ and $y$ have already collaborated in the past, the co-authorship similarity score $S_{co-authorship}(x, y)$ is assigned high score.
### 3.5 Recommendations Approach

- **Citation Similarity** ($S_{citation}(x, y)$): Similar to co-authorship similarity, citation similarity is computed using Jaccard coefficient.

$$S_{citation}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

where $\Gamma(x)$ represents authors who are cited by $x$ and $\Gamma(y)$ represents authors cited by $y$. If author $x$ and author $y$ have cited each other on the past, citation similarity score is assigned high value.

- **Community Similarity** ($S_{comm}(x, y)$): The community similarity between author $x$ and $y$ is computed by computing the cosine similarity between the communities themes of $x$ and $y$.

For this we represent every community theme as a vector of keywords. All the community themes vectors have keywords present of the community as the vector elements. For example, the community theme vector $\hat{A}$ of community $A$ has elements $\{a_1, a_2, \ldots, a_n\}$, and every other community has a similar vectors with same size. where $\{a_1, a_2, \ldots, a_n\}$ belong to keyword set. All the missing keywords in the community are replaced with zeros.

The cosine similarity of two communities is computed as,

$$Cosine_{sim}(A, B) = \frac{\hat{A} \cdot \hat{B}}{\|\hat{A}\| \|\hat{B}\|} = \frac{\sum_{i=1}^{n} \hat{A}_i \times \hat{B}_i}{\sqrt{\sum_{i=1}^{n} (\hat{A}_i)^2} \times \sqrt{\sum_{i=1}^{n} (\hat{B}_i)^2}}$$

where $x$ and $y$ belong to the communities $A$ and $B$ respectively.

We consider three different cases when computing community similarity between two authors,

1. In this case we assume both the authors $x$ and $y$ have previously attended one of the current series conferences and belong to a series community. Author $x$ belongs to series community $sc_{im}$ where $i$ is the id of series and $m$ is the community id and author $y$ belongs to $sc_{in}$.

$$S_{community}(x, y) = \begin{cases} 1 & \text{if } m = n, \\ Cosine_{sim}(sc_{im}, sc_{in}) & \text{otherwise} \end{cases}$$

2. There exist situations where one of the author has not participated in any of the series previous conferences. In this case we try to compute the similarity between the series community of the authors who has attend past events of the series and the global community of the other author. Assuming author $x$ belongs to series community $sc_{im}$ and author $y$ belongs to global community $gc_n$.

$$S_{community}(x, y) = Cosine_{sim}(sc_{im}, gc_n)$$
3 Conceptual Approach

3. This case deals with computing community similarity between two authors who have never participated in any of the series conferences. For this situation we compute similarity between two author’s global communities.

\[
S_{\text{community}} = \begin{cases} 
1 & \text{if } m = n, \\
\text{Cosine}_{\text{sim}}(gc_m, gc_n) & \text{otherwise}
\end{cases}
\]

assuming \( x \) belongs to \( gc_m \) and \( y \) belongs to \( gc_n \).

3.5.5 Similarity and Context

This is penultimate step in the system. Here the system collects all the similarity scores along with other required information to prepare the data in appropriate form.

The total similarity between the authors \( x \) and \( y \) is formally given as,

\[
S_{\text{total}}(x, y) = \alpha S_{\text{co-authorship}}(x, y) + \beta S_{\text{citation}}(x, y) + \gamma S_{\text{community}}(x, y)
\]

(3.4)

where \( \alpha + \beta + \gamma = 1 \).

The constants in the equation are assigned value according to the importance of their corresponding attribute. Constant \( \gamma \) is given higher value, since \( S_{\text{community}} \) is most important factor, followed by \( \alpha \) and \( \beta \) (\( \gamma > \alpha > \beta \)).

Later the system collects the required information to make the recommendations depending on the context. Since context-aware recommender systems make better recommendations this information is vital to the system. The information includes complete schedule of the conference. The system holds the complete information about each talk, place, time, speakers, etc. All this information is stored for further use. The context used by the system is time and location. The current time is used to make talk recommendations which are in progress or future talks. The location context is used to keep track of every author activity during the conference. Both these contextual information will be used by the recommender system to make better recommendations.

3.5.6 Recommendation

The system finally provides two different recommendations i) author recommendation and ii) talk recommendation. All the data computed until previous state are used by this state, esp. the similarities and context are of primary concern. CF techniques are used to provide the recommendation and now we shall discuss procedure of generating the recommendations.

Author Recommendation

Nearest Neighbors: Nearest neighbors, \( N \), of the target user are determined by computing \( S_{\text{total}} \) between the target user and rest of the participants (who have not collaborated before with the target user).
3.5 Recommendations Approach

**Recommendations:** These scores are sorted in descending order and the top $K$ scores and attributed authors are chosen. The top $k$ nearest neighbors, $N_T$, are recommended to the target users as author recommendations.

**Talk Recommendation**

**Nearest Neighbors:** The nearest neighbors, $N$, are determined in the same fashion as done in Author Recommendation. However, no author is eliminated despite any previous collaborations.

The top $K$ authors of $N$ are considered as nearest neighbors, $N_T$.

**Recommendations:** As shown in figure 3.4, CF techniques are applied to make talk recommendations to the target user. While applying CF technique, authors in $N_T$ are considered as neighbors and talks are considered as items. Since talks can’t be rated, if an user is attending a talk it is rated as 1 otherwise 0.

Now every talk is rated with the respect to the authors in $N_T$ attending the talk. To get the rating of each talk we calculate the weighted average in 2.8. However, we make subtle changes to the equation to be appropriate for our current situation.

All the rating averages are assigned to zero, and similarity weight is replaced by $S_{total}$. The result equation to compute the score of talk $t$ to the user $a$ is,

$$\text{score}(a, t) = \frac{\sum_{u \in N_T} (r_{u,t}) \cdot S_{total}(a, u)}{\sum_{u \in N_T} | S_{total}(a, u) |}$$  \hspace{1cm} (3.5)

where

$$r_{u,t} = \begin{cases} 
1 & \text{if user } u \text{ is attending talk } t \\
0 & \text{otherwise}
\end{cases}$$

However scores will not be computed for every talk at a given moment, the talks which are currently in progress will only be considered. After computing the scores of all current talks, the talks with highest scores are recommended to the user.
Figure 3.4: Using pre determined communities and CF techniques for Talk recommendations
4 Architecture and Implementation

4.1 Architecture

Following the architecture approach of CAMRS-1, CAMRS-2 is also built using client-server architecture. By adopting client-server architecture the workload is divided among the client and server. The client makes the requests and server replies with a response. With this approach, it is viable to only have one server and still reply to multiple clients requests. The client-server approach perfectly suits our system, since we have several users who constantly access the data by making requests to the server.

Since the workload can be partitioned between client and server, most part of computation and storage is performed on the server, and client is only used to display the relevant content. This way client does not have to either store data or process data, it is confined to interact with the user. The computations of recommendations and storage of all the data requires to have a high configured system. Since the clients are mobile devices which have very limited storage and rudimentary processing capabilities assigning computations and storage to the server is felicitous.

4.1.1 System Architecture

The system is developed using 3-tier architecture. The 3-tier architecture allows a system to be developed in three different modules independent of each other. The three tiers are presentation layer, logic/application layer and data layer. Presentation layer is where the logic is presented, and this layer also allows user interaction. Logic layer also known as application layer provides the business logic and manages data access. Data layer is the data server which holds the data. All these tiers are independent of each other, changes in one tier do not effect other tiers. CAMRS-2 architecture is presented in the figure 4.1. The presentation layer is the android mobile device screen, and as can be seen in the figure Application layer has been divided into three different modules. Each of these have their own purpose, and the service module which has all the services has mutual interaction with other modules. Finally, we have database layer which apparently stores all the records of information about users, conferences, artifacts etc.

Presentation Layer

The presentation layer of the architecture is the user’s smartphone. The smartphone is a mobile phone which has advanced features and connectivity than a ordinary mobile
phone. Smartphone has functionalities of camera, GPS, and sophisticated smartphones also include touch-screens, web browsers, and high-speed data access using Wi-Fi and mobile broadband. The reason to use smartphone is to provide the recommendations and other details in more elegant way. Not only that, by use of smartphone we can infer location of the user which in turn helps us to provide better recommendations.

Application Layer

The application layer is where the business logic of the system is implemented and located. As seen in figure 4.1, the layer comprises of following three components:

1. **XML Objects**: This component is responsible mapping Java classes to XML representations. The component makes use of JAXB to provide this functionality. The services component makes use of this component in building XML messages and unwrap XML messages into Java objects.

2. **Services**: Services component, as the names suggests, provides different services to the user and also interacts with the rest two components as part of providing responses to the user. This component can be further divided into more specific services, which have their own purpose to serve.

   - The **User Service** is responsible to validate user credentials and also allows new user to register with the system. This service also allows users to edit their personal details such as, e-mail address, website, telephone and research interests. User service also holds the responsibility of keeping track of current location of users.

   - The **Recommendation Service** is solely responsible for providing author recommendations and talk recommendations during the conference. This service uses help of its fellow services while computing both the recommendations. This service is most dynamic of all the services, since the recommendations provided by the service change swiftly as conference progresses.

   - The **Conference Service** provides complete details of the conference. By using this service a user can browse the complete schedule of the conference. The level of detail provided by this service is impeccable, a user can view ever detail of subevents such as start-time, end-time, location, topic and talks within the subevent. Even the talks are presented with every detail including the presenters of the talk. Should there be any changes made to conference schedule this service is responsible to provide updated schedule when requested by the user.

   - The **Social Network Service** is responsible to provide additional data about the recommendations made to the user. Since SNA has been used in making recommendations, we use this service to provide details about relations between the user and recommendations made. For example, when an author recommendation is made, the social network service provides additional
4.2 Implementation

We shall begin this section with discussing tools and technologies used during development of the prototype followed by different representations of data sources used while sending and receiving requests. Finally, we conclude by explaining how recommendations are computed on server and how they are presented on client device.

4.2.1 Overview of Tools and Technologies

In this section we go through different tools and technologies used in presentation layer, application layer and database layer.

4.2.1.1 Presentation Layer Tools

Android

Android is a Linux based open source mobile operation system, which was developed...
Figure 4.1: CAMRS-2 System Architecture
4.2 Implementation

by Google et al. Due to its touchscreen capability Android has been used in various tablets as well. Large number of smartphone manufactures use Android platform, and Android also enjoys large network of developers community. Many positive factors lead android to become the world’s most widely used smartphone platform.

Android has wide variety of features supported. It allows direct manipulation user interface which allows user to perform actions like swipe the screen, pinch it, etc. Android also supports wide range of connectivities Wi-Fi, GSM are most popular among them. Though mobile phones are not well known for their storage, android takes a leap and provides storage database SQLite. This storage can be used by developers for developing their application. Other prominent features of android are Java support, streaming media support and Bluetooth.

On the top of Linux kernel there exists Android runtime, which includes Dalvik virtual machine and the core Java libraries. The Dalvik VM is Google’s implementation of Java, which is specifically optimized for mobile device. All the code written will be run within the VM. The primary differences between Dalvik VM from traditional VM is, Dalvik VM runs .dex files, and the core Java libraries that come with Android are different from Java SE and Java ME libraries.

Application Framework Layer is the layer that provides the high-level building blocks that can be used to created applications. The most important parts of framework are, Activity manager which controls the life cycle of applications. Content providers are the objects which encapsulate data that needs to be shared between applications. Location manager helps to locate the position of an Android phone and Notification manager can be used to events like receiving message, reminders, proximity alerts and more.

Android Application

Google provides Android SDK which includes comprehensive set of development tools. The SDK includes debugger, libraries, emulator, documentation, sample code, and tutorials.

To test an Android application a developer can either deploy the application on a Android device or Android SDK includes a mobile device emulator which also can be used for testing. An emulator is a virtual mobile device that can run on a computer. The functionalities of emulator are similar to a mobile phone, however tasks like taking pictures, making calls will not be possible with emulator.

In addition to emulator and other libraries, Android SDK has few objects which are considered as building blocks of an application.

- **Activity**: An activity is a user interface screen. An application can define more than one activity during the transition of application from one phase to another.

- **Intent**: An intent is a mechanism for abstract description of an action. For instance, if an application needs to send a message, the respective intent can be invoked to send a message.
• **Services:** A service is a background process which runs without user interaction. For example, a music player.

• **Content Providers:** Set of data wrapped up in a custom API to read and write is a content provider. Content providers are used to share global data between applications.

As said earlier, every application can define one or more activities and every activity during its lifetime remains in one of several standard states. The activity can always transit from one state to another, but the developer will never have any control over the transition of these states. All the transition is managed by the system, however the a developer can get notified when there is a change to occur in state of activity. The following are the standard states an activity can be in during its life time, *Starting, Running, Paused, Stopped, and Destroyed*. The transition between these states is caused when certain actions are performed.

**Eclipse**

Eclipse is a free software development environment which was written in Java and built by a open source software community (started by IBM). Eclipse is majorly used to develop Java applications, however, with the help of plug-ins it can also be used a development environment for many other programming languages. Eclipse supports development of various platforms like RCP, Server platform, Web Tools Platform.

Eclipse also supports rich selection of extensions, and Google’s ADT is one of them. Google’s Android Development Tool (ADT) is a plug-in for Eclipse which is used designed to provide a powerful integrated environment to build Android applications. Using ADT the task of setting up the project, creating application UI, adding required packages, debugging and deploying the application can be done from IDE itself. In addition Eclipse with ADT allows to view the log messages in the output pane, provides pre-built XML layouts and according to Google, ADT also gives the leverage of boost in developing Android applications.

**XML**

During the beginning of the Internet, HTML (HyperText Markup language) was being used as a simple way to present the data in web browser. Later with the huge popularity, the size of the Internet has grown exponentially but the intention behind HTML was never to store and handle data. So, years later XML (eXtensible Markup Language) was created by the W3C and has emerged next to HTML as a foundation language on the Internet to serve the storage purpose. XML is not only specification of storing information, but also a specification for describing the structure of that information. Unlike HTML, XML has no *tags* of its own, it always allows the developer to create the required tags. A sample XML document is presented in [4.1]. The document stores information about children in a family, one can notice all the tags are user defined and the tags describe the data they contain.

XML can be used to share data between organizations and systems, and in recent
4.2 Implementation

Listing 4.1: An sample XML document

```xml
<?xml version="1.0"?>
<my_children>
<child>
  <name>Henry</name>
  <age>18</age>
</child>
<child>
  <name>Shawn</name>
  <age>12</age>
</child>
</my_children>
```

times, XML is being considered as a standard format to exchange data. The main reason behind this is, XML is structured, easily parsed and in human-readable format. XML did not stop with this, many other extensions of XML were unveiled later. XSL (eXtensible Stylesheet Language) is used to specify how the XML document should be displayed. XSL itself is made of three different languages, XSLT, XPath and XSL-FO. By use of all the languages XSL allows to transform XML document into any format we need. In addition to displaying XML document, to define the structure of XML documents different ways were evolved. First of this is DTD (Document Type Definition) and later XML Schema Language. These both define what tags can be used in the XML document and what type of data can be contained in tags and attributes.

Since we also use XML to communicate between client and server, we will now discuss how we will build and parse XML documents on client side. Java provides API for processing XML documents, this API allows to validate and parse the XML documents. The most widely used XML parsers are i) SAX (Simple API for XML) parser and ii) DOM (Document Object Model) parser.

The DOM parser parses entire XML document and constructs a complete in-memory representation of the document. The constructed representation is a tree structure containing nodes in the XML document. There exists different types of tree nodes which represent data in XML document. The most common nodes are element nodes and text nodes. Since the DOM parser stores the complete representation of document, there exist no limitations in traversing. The possibility to traverse around the tree without limitation is the most important and useful feature of DOM parser.

The SAX parser also called light-weighted parser, because of the reason it requires less memory to parse a XML document. SAX parser unlike DOM parser does not build any in-memory representation of the XML document and therefore is much faster. SAX parser is event based, it informs the client by invoking call back methods depending on the structure of the XML document. Client can use the callback methods to implement further required actions. Since SAX does not store any representation of XML document it is not possible to traverse around.

For clients like smartphones, it is always better option to choose SAX parser instead
of DOM parser. The reasons are twofold, smartphones have limited memory and SAX parser is also faster than DOM parser. However, the only complication with SAX parser is its complicated way of handling events. SAX parses have been developed ages ago and hence makes it difficult for developer to handle the events in straightforward way. To overcome this drawback we chose XPP (XML Pull Parser). XPP is similar to SAX parser, it is very quick and event based. Nevertheless, XPP is less complicated to implement and is simplified to great extent. A sample snippet of XPP can be seen in 4.2.

XML messages can be either built using hard coded text string or by using third party libraries to make tasks easier and error free. Though in CAMRS-1 XML messages are built explicitly using Java strings, we chose to make use third party library to built XML messages. One has to always remember that the client here is smartphone and the size of application is often couple of hundred bytes, so we cannot import a library which itself is couple of mega bytes. XPP is a light-weighted library and suitable for clients like smartphones. XPP not only provides eloquent parser, but also, an efficient way to build XML messages as seen in 4.3. Hence XPP library is used to parse and write XML documents on the client side of the implementation.

Indoor Sensing Technology

To implement this feature we have used the similar technology used in CAMRS-1. Using services provided by Android it is not difficult to locate the position of a user using the Android device. Albeit the accuracy of the location details we chose not to use in-built Android services. The main reason for this is, Android services usually provide location details in form of longitudes and latitudes. To make use of these values we have to map every point of Conference location to longitude and latitude values, which is impractical.

Since we need location of user at room-level, it is better to employ a technology which can give us the details of the user regarding his position at room-level. For this we use ZXing (Zebra crossing), an image processing library implemented in Java. This is an open-source project, which can process 1D/2D barcode images. Unlike many other similar technologies ZXing does not send the data to server, instead, decodes the barcode on the device itself. To use this library a mobile should be equipped with a built-in camera. For CAMRS-2, similar to CAMRS-1 we use QR (Quick Response) codes to represent room id’s. QR codes were initially designed for automotive industry. But in recent times they have gained huge popularity because of its fast readability and other factors. QR code is constituted of black modules arranged in a pattern, which can be seen in 4.2. ZXing can be pre-installed as an application on mobile device, before embedding its functionality into a different application. Since QR code is one of the codes supported by ZXing, the library can be used in CAMRS-2 application to determine user location. However for this an user has to scan the QR code whenever

\[1\] http://www.xmlpull.org/
\[2\] http://zxing.org/w/decode.jspx
4.2 Implementation

Listing 4.2: Sample XPP code parsing XML document

```java
public void parseXML(InputStream is) {
    XmlPullParserFactory factory =
        XmlPullParserFactory.newInstance();

    factory.setNamespaceAware(true);

    XmlPullParser xpp = factory.newPullParser();

    xpp.setInput(is, "UTF-8");
    int eventType = xpp.getEventType();

    while(eventType != XmlPullParser.END_DOCUMENT) {
        if(eventType == XmlPullParser.START_DOCUMENT) {
            //DO SOMETHING
        }
        else if(eventType == XmlPullParser.START_TAG) {
            //GET THE TAG NAME
        }
        else if(eventType == XmlPullParser.TEXT) {
            //READ THE TEXT IN TAG
        }
    }
}
```

Listing 4.3: Sample XPP code building XML document

```java
public void buildXML(ClassName obj) {
    XmlSerializer serializer = Xml.newSerializer();

    serializer.startDocument("UTF-8",true);

    serializer.startTag("", "userprofile");
    serializer.text(obj.getUserName());
    serializer.endTag("", "userName");
    serializer.endTag("", "userprofile");

    serializer.endDocument();
}
```
he relocates to a different room.

4.2.1.2 Application Layer Tools

In this section we can have a look at all the tools used in the application layer.

RESTful Web service

A web service which fulfills the constraints of REST is a RESTful web service. Representational State Transfer (REST) is a software architecture style for distributed systems. REST has been introduced and defined by Roy Fielding in 2000 in his doctoral dissertation. In REST-style architecture a client sends a request to the server and the server replies with a response. All this communication is built around the transfer of resources representations. The resource can be any object that can be addressed and the representation of the resource is the current state of the object. REST allows every resource on the server to be addressed by an URI, and even the modifications of the resources can be done requesting a specific URI. The standardized format of URI is as follows:

```
scheme://host:port/path?queryString#fragment
```

A client server architecture should also be stateless in its communication to abide by REST constraints. Being stateless implies that no context of client being stored on server. Every request from client should hold all the required information of its own, and should not rely on any previous interactions. However, a client can cache responses from server. In addition to these, REST imposes few other constraints.

RESTful web services are collection of resources which can be accessed by via URLs and by using HTTP methods. The HTTP methods are usually compared to some of standard SQL operations. The following are the typical HTTP methods used during the implementation of the web services.
4.2 Implementation

- **GET:** This method is to query the server for specific information. This operation is idempotent, meaning it does not matter how many times you repeat this operation the result received will be the same. It is also a read-only operation.

- **PUT:** This operation is similar to that of insert or update in SQL. Whenever a PUT operation is performed the server knows the identity of the resource which should either be created or updated. This operation is idempotent, repeating PUT will not have any effect.

- **DELETE:** DELETE method is used to remove resources. Repeating DELETE on same resource will not effect the underlying service.

- **POST:** Unlike all other operations POST is nonidempotent. Every time POST operation is invoked the underlying service is modified in a unique way. POST operations may not return any information.

To create RESTful web services, JAX-RS (Java API for RESTful web services) a Java API can be used. JAX-RS can be used to create web services according to the REST architectural style. JAX-RS provides annotation to map resource classes as web sources. To build CAMRS-2 web services, an open source implementation of JAX-RS Jersey has been used. Jersey supports the annotations defined in JSR-311 and is easy for developers to build a RESTful web services with Java.

**JAXB**

Java Architecture for XML Binding (JAXB) is a Java implementation which allows to map Java classes to XML representations. By using JAXB a developer will be able to marshal Java objects to XML and can unmarshal a XML to Java objects.

JAXB has been part of Java SE 1.6 and one of the API in Java EE. Java provides certain tools with JDK, two of them are xjc and schemagen. xjc can be used to directly convert an XML schema to Java classes and schemagen can be used to convert annotated Java classes to a XML schema document. Developers use annotations from XML binding namespace to markup classes. An example of XML annotated class is provided in 4.4.

As seen in 4.4 the Java class is annotated with various annotations. The annotations @XmlRootElement, @XmlElement and @XmlAttribute represent root element of the XML document, XML element and XML attribute of root element respectively. JAXB offers several more annotations which allow to specify a well defined XML schema in terms of Java classes.

Once the Java class with XML annotations has been defined, we can use the features of Marshalling and Unmarshalling through instance of JAXBContext. To marshal, the instance of JAXBContext requires a Java object and it binds the object to respective JAXB class and returns an XML document as the result. Exact inverse way is performed when unmarshalling, a XML document and JAXB class is needed by the instance of JAXBContext to return the objects from the XML document.

[4http://jersey.java.net/](http://jersey.java.net/)
Listing 4.4: A Java class with JAXB Annotation

```java
@XmlRootElement(name="customer")
public class Customer {

    String name;
    int age;
    int id;

    public String getName() {
        return name;
    }

    @XmlElement
    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    @XmlElement
    public void setAge(int age) {
        this.age = age;
    }

    public int getId() {
        return id;
    }

    @XmlAttribute
    public void setId(int id) {
        this.id = id;
    }
}
```
4.2 Implementation

Apache Tomcat

Apache Tomcat or Tomcat is an open-source software implementation of Java Servlet and JavaServer Pages technologies. Tomcat also provides a pure HTTP web server environment for Java code to run. Since we implement RESTful web services in Java, Tomcat provides the best features to implement our web services. Modification of Tomcat configuration is usually done by modifying the XML configuration files. The XML files can be used to configure the server name, port addresses, and other critical server information.

Catalina is Tomcat servlet container, which implements the specifications for Servlet and JavaServer Pages (JSP). A Realm element present in Tomcat represents a database of username, passwords, and roles. By configuring Realm, a developer can define different roles like admin, user, etc. of the system. Coyote is the HTTP connector component of Tomcat, this component supports the HTTP 1.1 protocol for the web server. Jasper and Cluster are other two components of Tomcat. Jasper is the JSP engine and Cluster component is used to manage large applications.

There are many other web servers available, such as, IBM HTTP Server, Jetty, IIS, etc. Some of the web servers in market are non-free. In any event, implementing Servlets in Java is best supported by Tomcat, not only that, but also, Tomcat is a lightweight web server. Other important features of Tomcat are, scalability, performance, garbage collection, good support for connection pooling.

JDBC

Java-based data access technology or Java Database Connectivity is an API for Java programming language that provides methods to access and modify data in database. JDBC was released as part of JDK 1.1 and since it has been part of Java SE. JDBC is inclined towards relational databases, every relational database has its own JDBC drivers developed. For instance, Oracle database has OJDBC drivers, MYSQL has Connector/J to connect Java client to MYSQL database.

To access the database, a Java client has to load corresponding JDBC driver class, and later connection between the client and database can be made. However, a client needs to provide adequate credentials to get connected to the database after loading the required JDBC drivers.

Using the methods provided by JDBC the client can perform several SQL operations. There exists different types of Statements represented by JDBC to perform various SQL operation in different ways. JDBC also supports the invocation of stored procedures from the client.

R and Igraph

R is an open source programming language and environment for statistical computing and graphics. The usage of R language is extensive among statisticians and data miners. It is used mainly for development of statistical software and data analysis.
Listing 4.5: Sample R code

```r
> x <- c(1,2,3,4,5,6) # create a vector
> y <- x*2 # multiply the vector by 2
> print(y) # print the vector y
[1] 2 4 6 8 10
```

is mostly influenced from S language which was developed at Bell Laboratories. R is considered as different implementation of S, every program written in S can run under R unaltered.

Though it is widely believed that R is a statistical system, the R development team sees R as an environment within which statistical techniques are implemented. R can be easily extended via packages, and CRAN family provides many packages in addition to the packages supplied with R distribution.

R is an interpreted language and used through a command line interpreter. R supports many data structures which include vectors, scalars, matrices, data frames and lists. R also supports object oriented programming with generics, procedural programming with functions. A sample R code can be seen in 4.5. We have used R environment to detect the communities in the networks. R has also been widely used in social networks analysis. For network analysis we rely on the packages provided by igraph which is part of CRAN family. igraph is collection of packages for graph theory and network analysis and it has interface to R.

We use igraph packages with along R environment for community detection. igraph provides various community detection algorithms such as, Edge-Betweenness, Leading Eigenvector, Fast-Greedy, Walktrap, etc. To use one of these algorithms one has to load the required igraph libraries onto R environment. Depending on the desired output values and necessary network specific attributes a developer can program in R for a meaningful output.

**Stanford CoreNLP**

Stemming is common procedure in statistical topic modeling techniques. Stemming is the process to reduce inflected words to its root or base form. A program which perform stemming are called stemmers. Stemmers are responsible words which are in other form of their original base form, for example, “clustering” is other form of the base word “cluster”. After identifying such words stemmer reduces those words to its base form.

The main reason for stemming is, in topic modeling during detection of topics it is not desirable to differentiate two words which have the same meaning but are presented in different tenses or in different forms (singular/plural). The titles such as “Analyzing networks...” and “Network analysis..” have different words, however, by removing suffixes would only result in two words, analyze and network. Statistical Topic Modeling
4.2 Implementation

<table>
<thead>
<tr>
<th>Word</th>
<th>Porter Stemmer</th>
<th>Stanford CoreNLP</th>
</tr>
</thead>
<tbody>
<tr>
<td>example</td>
<td>exampl</td>
<td>example</td>
</tr>
<tr>
<td>really</td>
<td>realli</td>
<td>really</td>
</tr>
<tr>
<td>basis</td>
<td>basi</td>
<td>basis</td>
</tr>
<tr>
<td>mobilities</td>
<td>mobil</td>
<td>mobility</td>
</tr>
<tr>
<td>parity</td>
<td>pariti</td>
<td>parity</td>
</tr>
</tbody>
</table>

Table 4.1: Comparing Porter Stemmer and Stanford CoreNLP

methods which depend of the frequency of the words always try to reduce every term to their base forms, and hence use stemmers before modeling.

There are several types of stemming algorithms which are different in terms of performance and accuracy. The most widely used stemmer algorithm is written by Martin Porter, the algorithm is often referred as Porter Stemmer. Porter Stemmer is very fast way to perform stemming, nevertheless, the stemming is inaccurate in few cases and the words lose their original meaning after being stemmed.

To overcome this we have chosen to employ a tool provided by The Stanford NLP Group. Stanford CoreNLP is a natural language analysis tool which can accept English language text as input and provide the base forms of words, parts of speech, determine if the word is a company, people, etc., normalize other quantities, and carry out various other relative operations. Albeit its wide range of offerings we only chose to utilize the part which returns the base form of the words. The performance of the tool is comparable to that of Porter Stemmer, and the results are better than Porter Stemmer.

Stanford CoreNLP is written in Java and it is free to use. A developer can chose required tool from CoreNLP, as it is integration of several NLP tools.

4.2.1.3 Database Layer Tools

In this section we present tools and technologies used in Database layer.

Oracle Database

Oracle database 11g has chosen to be the database management system to store and maintain the data for CAMRS-2 system. Both AERCS and CAMRS-1 have used Oracle as their back end due to the features provided by Oracle. And Neither AERCS nor CAMRS-1 had any hassle with Oracle database. Using same Oracle database makes it relatively easy in accessing existing data and configuring tables according to previous structure of data. By choosing same database huge workload of migrating data can be avoided.

Oracle SQL Developer

http://nlp.stanford.edu/
Oracle has provided SQL Developer as an Integrated Development environment (IDE) to work with SQL in Oracle databases. SQL Developer is a free product and has been written in Java.

SQL Developer provides features like, opening new tabs for every database schema, syntax coloring, code error insights and bracket matching. Using SQL Developer one can export data from database to files in different formats. Importing data into database is also possible using SQL Developer with few mouse clicks. In addition, SQL Developer has few interesting features that can be used by a developer to visually understand and improve query performances. Few of them are **Explain Plan** and **SQL Tuning Advisor**.

### 4.2.2 Data Sources

In the path of CAMRS-1, XML schema are devised to describe the structure of the academic event. However, instead of one XML schema, in CAMRS-2 we devised couple of XML schemas. The reasons are may, firstly we realized it not a good option to let one XML document to hold entire details of the academic event, since modification and maintenance would be difficult. Moreover, parsing the whole XML document to get details of one talk is not effective way. To overcome the drawbacks, in CAMRS-2, we devised multiple XML schemas that will allow the XML documents to present data at required abstract level.

These XML schemas are made public. The administrators and organizers of the academic event, who desire to use CAMRS services during the academic event can build the XML document complying the respective schema definitions. After the submission of the data in the form of XML document, we parse the documents and store the data in our relational database.

The XML documents are **program**, **talks**, **talk**, **attendees**. The name of documents are the root elements of the XML documents. **Program** document holds the data of all the subevents in the academic event and **talks** has the information about every talk topic that will be held during the event. To present complete details of a talk, document **talk** is used. Finally, the **attendees** document has the list of the all the authors attending the academic event. We shall now look at the structures of the documents and have a description.

#### Program Schema

The diagrammatic presentation of the program schema can be seen in the figure [4.3](#). The root element of the document is **program** which has the name of the event as attribute and features list of the **subevent** elements. The difference between the CAMRS-2 Program schema and CAMRS-1 program schema is, the latter one not only has list of subevents but also every talk in the subevent. We have removed talks from the document to keep it concise.
4.2 Implementation

Figure 4.3: Program Schema

The element subevent has about 10 elements as its children, however, not all of them are mandatory. Elements id, eventId, startTime, endTime, date and location are made mandatory, since every subevent has an id and has to take place at certain time and at specified location. These mandatory elements are also necessary while making talk recommendations to the users. Other elements topic, type, chair are optional elements. Every child element of subevent appears at most once. However, a program can have multiple subevent elements.

Talks Schema

The structure of Talks schema is depicted in figure 4.4. The root of the document is talks which features list of talk elements. In CAMRS-1 talks were part of program schema, unlike in CAMRS-2.
4 Architecture and Implementation

Figure 4.4: Talks Schema

A subevent can have multiple talks during its time period. To allow user to get a glimpse of all the talks under the subevent every talk element is attributed with only two child elements id and topic. This structure allows user to just view the all the talk topics of a subevent, instead of viewing every minute detail of the talks.

Talk Schema

The talk schema is diagrammatic presentation is depicted in figure 4.5. The schema has talk as the root element. The child elements of talk are little similar to that of subevent element in program schema. Since every subevent has multiple talks, every talk has its own time frame. To indicate the schedule of every talk, the element talk has certain startTime and endTime which are within the duration of its respective subevent, but date and location are assigned from the subevent the talk belongs to. Other elements are id, topic, subEventId and authors. The element authors has list of author elements who are the presenters of the talk, which has id and name as its child elements.

Attendees Schema

This schema has attendees as root element and the element has list of attendee elements as its children. The attendee element is similar to the element author in Talks schema, it has id and name as child elements. Conference organizers and administrators can use this schema to build the document which features list of all the authors attending the academic event.

4.2.3 Recommendation

Assuming we have all the required data, we can proceed to take a look at the implementation of the Author and Talk Recommendations. We have already discussed in detail how we will approach to make approach in the section 3.5. Here, in this section we will look at how the approach has been implemented.
4.2 Implementation

Figure 4.5: Talk Schema
4.2.3.1 Operational Principle of RESTful Web Service

Recent times have seen RESTful web services emerging as prevailing design model for web services. Yahoo! Web service, flickr, and del.icio.us are some of the services which use REST style to expose their business logic.

As we discussed earlier, RESTful web services are collection of resources and every resource can be accessed via URLs by HTTP methods. To build the services we have used Jersey, an open-source implementation which supports annotations define in JSR-311. JAX-RS provides annotations which are mapped to the resource class (POJO) as a web resource. We shall now discuss about few annotations that are used in developing the system.

- The annotation @Path is the relative path to either a class or a method which is considered as a resource.
- The annotation @GET is annotated to the methods which act as HTTP GET methods. JAX-RS annotations @POST, @PUT and DELETE are HTTP POST, PUT and DELETE methods respectively.
- The @Produces annotation is used when a method is returning a certain object when the method is accessed via the URI. The return type can be any MIME media type.
- @Consumes is the opposite of the @Produces. The class method annotated with @Consumes specifies with media type will be accepted.
- The @PathParam annotation is used to bind the parameter to the path segment. Annotation @QueryParam is used to bind the parameter to the HTTP query parameter value. This is often used when a GET request is made. The JAX-RS annotation FormParam binds the parameter to the form value. When there exists a form with POST request this annotation is best used.

There exists few more other annotation defined in JAX-RS, which deal with the cookies, context, request header, etc.

In listing 4.6 you can see a GET method which can be invoked by the URI http://www.example.com/author?id=1234 which returns a plain text as response. It can be noticed the URI has the query parameter “id” and the annotation @QueryParam is used to bind the value.

4.2.3.2 The CAMRS-2 Recommendation Engine

The CAMRS-2 has the services which provide the both author and talk recommendations. We refer the services as the recommendation engine. In the following we will have a detailed discussion how the recommendation services are implemented in CAMRS-2.

Author Recommendation

The process of author recommendation is vaguely depicted in the figure 4.6. When a
4.2 Implementation

Listing 4.6: Example of a Service Implementation

```java
@Path("author")
public class Author {

@GET
@Produces("text/plain")
public String getAuthorName(@QueryParam("id") int authorId) {
    String authorName = authorName(int authorId);
    return authorName;
}
```

user wants to perceive which attendees at the current academic event are his recommended authors, the user can perform appropriate operation on his android device. When the operation is invoked, the HTTP client makes the request to the server to the appropriate URI which will return the user’s recommended scientists.

The HTTP client makes a GET request to the RecommendAuthors resource which is a Java class. The Java class has the necessary code to compute the top $K$ author recommendations. The top $K$ authors in the recommendation set are the Nearest Neighbors of the target user. The resource RecommendAuthors can be accessed by the relative URI /recauthors and query parameters authorid and eventid. A user with author id 1234 and attending event with id 5 can get his nearest neighbors by accessing the URI /recauthors?authorid=1234&eventid=5, this URI will return list of authors who are the user’s top $K$ recommended authors, and are attending event with id 5.

After receiving the request from the client the RecommendAuthors Java class will find the appropriate function which accepts GET requests and binds the value of query parameters. The getRecommendedAuthors function is invoked since it serves the purpose. The methods binds the query parameter to its parameters and later makes the connection to the database. Upon a successful database connection, the method executes a SQL query which returns the top $K$ recommended authors to the user who are attending the event.

For every row returned the method creates an object to the class Author which has id and name as class variables. Once all the rows are processed with the help of JAXB the list of recommended authors will be marshaled into a XML string. The XML string is returned as a response to the client. An example of a reply can be seen in listing [4.7].

From the listing [4.7] it is clear that the XML message only holds names and ids of the recommended authors. However, we decided to provide complete profile of an author if only the user explicitly requested through his android device. Therefore, another service is implemented which provides complete details of an author when a
4 Architecture and Implementation

HTTP Client
/recauthors
Java Servlet
Resource Class
RecommendAuthors.java
getRecommendedAuthors (@QueryParam ("authorid"), @QueryParam ("eventid"))
3: Retrieve Similarity Scores
4: Return the data
5: Send response as XML String
Android Device
Tomcat Server
CAMRS-2

Figure 4.6: Author Recommendation

Listing 4.7: Example of Reply to Author Recommendation Request

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<authors>
  <author>
    <id>5877</id>
    <name>Ralf Klamma</name>
  </author>
  <author>
    <id>321466</id>
    <name>Yiwei Cao</name>
  </author>
</authors>
```
Talk Recommendation

Similar to Author Recommendation, a pictorial representation of implementing talk recommendations is presented in figure 4.7. In case of author recommendations, we did not need user context to great context. But, in the case of talk recommendations, we rely on every user context at a great extent. From the section 3.3 after detecting the Nearest Neighbors we employ CF techniques to make talk recommendations to the target user. So as to provide better talk recommendation we expect every user of CAMRS-2 system to provide their location by using the embedded QR Code Scanner. To track the location of a user a resource UserLocation has been implemented. The UserLocation Java class implements several methods which help in updating request is made to it.

The complete profile includes author details like, email and website. On further requests the user can also have a look individually at the author’s publications, Common Coauthors, Common Citations and finally author Research Interests.
the location of users.

The three methods that are invoked during the change in context of user location are, setLocation, updateLocation, vacateLocation. When a user enters into a room and uses his CAMRS-2 Android application indicate his location, the HTTP client makes a POST request to the URI /location/set. This in turn invokes method setLocation which has form parameters authorid, locid and eventid to bind the values of user’s author id, room location id, and the attending event. The method setLocation later creates an entry in database about user’s location. The user can use his Android device to notify when he is vacating the room, for this the HTTP client makes a DELETE request to /location/vacate, and method vacateLocation is invoked to remove the user location entry from the database. The user can also relocate to another room before vacating his previous, when such situation arises, the HTTP client makes a PUT request to /location/update. The method updateLocation responsible to handle the update uses the same form parameters as setLocation method and modifies the user location entry in the database. It can be noticed we perfectly maintained the REST principles. We use POST request to create a new resource, PUT to update it, and DELETE to remove the resource.

The CAMRS-2 system will utilize the context information provided the users to make talk recommendations. A user can view the talks that are supposedly more interesting using the CAMRS-2 Android Application. When a request for recommended talks is made, a request is sent to /rectalks resource which is implemented as RecommendTalks Java class. The Java class includes the method which returns list of recommend talks to the target user. The getRecommendedTalks is the method which implements the concept proposed in section to return the recommendations. This method is accessed via the URI /rectalks/, to which the HTTP client makes a GET request. The method uses query parameters authorid and eventid. Once the parameters values are extracted the following sequence of steps are executed:

- From the available academic event schedule current talks are determined.
- User Nearest Neighbors are computed
- The scores of talks are calculated with the equation 3.5
- The talks are listed in decreasing order of their scores
- Finally the list of talks are returned

The method getRecommendedTalks maintains constant connection with the database while executing the above steps. Once the final step is computed, the talks are returned to the client in XML format. An sample of the recommended talks XML message can be seen in listing 4.8.
4.2 Implementation

Listing 4.8: Example of Reply to Author Recommendation Request

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<talks>
    <talk>
        <id>3</id>
        <topic>Discovering Hierarchical Relationships in Educational Content</topic>
    </talk>
    <talk>
        <id>5</id>
        <topic>Learning Analytics for Learning Blogospheres</topic>
    </talk>
</talks>
```

4.2.4 CAMRS-2 Client

Similar to CAMRS, CAMRS-2 is built with two major components. One of them is the user interface component which allows user to interact the device and the other is environment sensing module. The latter component is used to determine the location of the users. The purpose and implementation details of both the components are now discussed.

4.2.4.1 User Interface

The user interface of CAMRS-2 is an Android application. The application allows users to communicate with system by either touching the screen or using keypad controls. Whenever an interaction is made with the user interface, the application communicates with the server and upon receiving the response parsing is performed and later data is presented in appropriate format. This particular process can be broken in several stages, i) Receive user command ii) communicate with the server iii) parse the response from server iv) render the data in appropriate design. Steps i and iv are the functionalities which can be achieved using Android built-in mechanisms. However, ii and iii can be attained using third party libraries.

The built-in mechanisms are Activities, Intents and Services. All of these were already discussed briefly before. Activity is the user interface screen, a user can interact with the screen in the context of the activity. In other words, an activity is a focused thing a user can do. Few instances are, a list of items where user can choose one, a list of photographs with titles, canceling of confirming a request. An Android applications can consist several activities, all these activities are independent but work together cohesively. Every activity is implemented as an implementation of Activity base class. By default an activity does not have any layout, it is the responsibility of the developer to assign a layout to the activity. The design of layouts is defined in a XML file, all the layout files are located in res/layout folder of the Android application project. For handling
events a developer can override appropriate methods in the activity Java class which loads the layout file.

Other important XML file in the Android application project is Android manifest file. The manifest file keeps track of everything the application needs. Permissions required, activities involved should be defined the manifest file. The file is stored under the root folder named as AndroidManifest.xml. Every Android application has a manifest file. The information provided in manifest file is usually, name of the Java package for the application, components of the application, permissions required by the application, and what level of minimum API is required by the application to run.

In addition to activities, Intents are the other important building block of Android applications. Intents are used for various purposes in Android applications. Android applications use intents to start an activity and also make communication between various parts of the system. Intents are not confined to start an Activity with in the same application, but also, to start other applications. The activities which act upon receiving an intent have a intent receiver to listen for the intents and notifies the system what to do. Intent in addition with the action to perform have data. The data is usually used and modified by the activity or application which received the intent.

The communication between the Android device and the Restful web services is done by using the HTTP client in the Android application. Java provides a built-in HTTP client library which can be used to send requests to the server. Unlike CAMRS-1 where Apache HTTP client was used, in CAMRS-2 the basic HTTP client from Java is used. The reasons are two fold, the functionality offered is almost the same and Java client is much easier to implement. In addition to java.net.HttpURLConnection package which is used to make a connection to the server, java.net.URL package is used to create a URL in order to send a request. The process of sending request to an URI and receiving response is performed in sequence of steps.

The procedure begins with building an URL from the basic URI using the Java URL class. Later on a HTTP connection is made to the URL. The properties of the connection are set using various methods provided by Java library. The same connection can be used to send request and receive the response. A minimal Java code of the procedure is presented in listing 4.9. Until previous versions of Android the HTTP connection was allowed on the same thread of UI. However, during the recent developments it is strictly prohibited to have a HTTP connection and UI on same thread. To create a connection a developer has to create a new thread for its designated purpose. The main reason for this modification was, having HTTP connection and UI on the same thread severely degraded the application performance.

As can be seen in the listing 4.9, the response received is sent as a parameter to a static method of a Java class. The class imports the XPP (XML Pull Parser) library. The method parseXML() which is similar to that of the one in the listing 4.2 parses the XML response and creates the list of objects of certain Java class. The built
4.2 Implementation

Listing 4.9: Java code making a HTTP connection

```java
public void run() {
  URL url = new URL(BASE_URI+"\resourcePath?param1=paramvalue");

  HttpURLConnection httpConn =
    (HttpURLConnection)url.openConnection();
  httpConn.setDoInput(true);
  httpConn.setRequestMethod("GET");
  httpConn.setRequestProperty("Accept", "text/xml");
  httpConn.setRequestProperty("Accept-Charset", charSet);

  InputStream is = httpConn.getInputStream();
  objList = HandlerClass.parseXML(is);

  is.close();
  httpConn.disconnect();
}
```

objects are used by the Android application to present them in appropriate format to the user.

Every request from the user Android device to the Restful web services includes information about the user. For instance, when a user requests for his recommended authors, the URL consists of the server details, resource path and author id the user. As requesting user for his details every time is not desirable, CAMRS-2 requests every user to authenticate himself/herself before using the system services. The authentication process requires user to provide his user-name and password, in case the author does not have an account yet, he can create one. Once the user makes a successful login, the server responds with certain information which is unique to the user account. Since the data returned by the server is vital in every further request user makes, the client stores the values. For this, a Java class by name ApplicationContext is created. This class consists of class variables which are used to store the author’s user-name, author name, author id, author contact information, and the Base URI which is used for every request made to the server. ApplicationContext is also composed of several setter and getter methods, which are used to access and modify the variables. All the information stored in the class is used in various ways. The author id is used when sending a request to the server for recommended authors, recommended talks, etc. The author name is mostly used to notify the user on which author name is the current account being operated.

After logging in the user can perform several actions, few of them are, view recommended authors, recommended talks, schedule, other authors profile, etc. Few of the UI snapshots are presented in the figures [4.8][4.9][4.10].
4 Architecture and Implementation

Figure 4.8: Recommended Authors and Recommended Talks

Figure 4.9: An Author’s Profile Page and an Author’s Research Interests

68
4.2 Implementation

4.2.4.2 Environment Sensing Module

This module implementation is inspired from CAMRS-1, which also has a module for the same purpose and holds same name. The similarity between the CAMRS-1 and CAMRS-2 Environment Sensing Module is very close. Since both the modules use similar package libraries and offer the same functionality the resemblance cannot be unnoticed.

The Environment Sensing Module is entirely used to determine the location of the user. This module with the help of user input determines user’s current location and updates the variables. Though the module is quite similar to the previous one, the implementation of this module has been altered. The module consists of two Java classes, i) Location and ii) ManualPositioning. The first class has class variables which indicate the current location of the user and the latter one is used to scan the QR code. The ManualPositioning class has an intent which uses the package com.google.zxing.client.android to scan the QR code and return a string back to the called activity. Location class constitutes of few functions which are used by other activities and classes to understand and update the user current location. CAMRS-2 provides two implementations, check in and check out which can be used by user directly to indicate his location. These implementation use the functions in Location to keep user location context up to date. The following are the functions in Location:

- setLocation(): This method is invoked when user checks in. The method
4 Architecture and Implementation

has one parameter, which is used to update the variable responsible for holding the current user location.

- **vacateLocation()**: When a user leaves a room this method is invoked. The method updates the user current location with null. The main purpose of implementing this method is to make sure that the application does not provide false positive recommendations.

- **updateLocation()**: This method is similar to the method `setLocation`, but is only invoked if the user has not vacated before relocating to different room.

- **getLocation()**: Any class in the application can use this method to learn about the user current location.

Whenever a user uses the check-in functionality the QR code scanner is activated. This mode allows user to point the camera towards the QR code (the room name/id in QR code form), which automatically scans the code and returns a plain string response to the concerned activity of the application. Later on the system sends a request to the server requesting to update user location.

Along with location of users, the time of request is essential in recommending talks to the target user. In CAMRS-1, when a user requests for talk recommendations the client builds a URL which also consists of the time-stamp of the request. Therefore allowing the server to understand the current time and making talk recommendations accordingly. Albeit it’s successful implementation of the time context concept, this approach might fall back in few cases. To overcome this, in CAMRS-2 the request does not hold any details about the time-stamp, however, the server itself determines the current time from available built-in methods and computes the talk recommendations.

### 4.3 Summary

This chapter has been completely dedicated to the CAMRS-2 architecture and implementation. Beginning with the client server architecture, we continued till the sensing technologies on Android device. Every tool used during the development has been described, and related tools are also mentioned. To make it more clearer, examples and listings are presented appropriately. Tools like R, igraph which are used to detect communities are discussed. Comprehensive details are provided about the libraries used when detecting topics of papers. Concepts like RESTful web services and XML parsing are explained in more detail. Diagrams are used to present the reader with an abstract idea of the implementation. Comparison with CAMRS-1 implementation has been made to point the reason behind selection of different way of implementation.

All the technologies, tools and libraries used to develop the system can be replaced with their alternatives. Taking several factors into consideration current architecture
and implementation has been adopted. The whole system has been implemented in such a way that it could accommodate any changes and updates, without any trouble.
4 Architecture and Implementation
5 Evaluation

“You can’t always get what you want”-Rolling Stones

A system can never serve the purpose completely, it is always required to improve it and make it better. However, we always can determine how good a system is functioning by certain procedures. To do this, we evaluate our system to learn about its accuracy and performance. Evaluating a system depends on the system and purpose the system is serving. For a recommender system, one way to evaluate the system is to compare the recommendations made by the system to the user choices.

Despite all the techniques and concepts used in building CAMRS-2, eventually CAMRS-2 is a recommender system. Hence, the evaluation can be performed by comparing the recommendations to the original choices of the user.

For this, we have chosen to evaluate the system by comparing the recommendations to the choices made by authors during the recently held International Conference on Web-Based Learning (ICWL 2012). ICWL 2012 was held in Romania during the month of September in the year 2012. As the conference has already been finished, we requested the organizers of the event to provide the information about the conference, which included conference details, participants and other related information.

ICWL 2012 was continued for three days, the academic event had about 25 subevents (Paper Sessions, Workshops) and 60 talks during the three day period. The event had two parallel tracks, and a participant can choose either of them and attend the sessions that belong to the track. The keynotes lasted for 1 hour, and full papers and short papers lasted for 25 and 20 minutes respectively. The presentation time for workshop was decided by organizers accordingly.

5.1 Data Set

The data set provided by the organizers had the information which is enough to understand the conference structure and the participants. The event had 204 participants from various parts of the world. However, only 121 of them exist in the AERCS database, and hence, rest of them were not considered during the evaluation process.

Few interesting facts were discovered during the evaluation, out of 121 participants, only 30 have attended previous ICWL conferences. Almost 90 participants have not had any relation with ICWL during the past years. As discussed in previous sections, a author can belong to a series community, if only he had contributed during previous conference of the series. Implying, all the 90 participants do not belong to any series.
community. However, thanks to our robust approach, we can still use the global communities to determine the similarities between the authors.

In total all the 121 participants belong to 16 different global communities, from existing thirty thousand communities. Which is understandable, since it is well known that authors who collaborated in the past tend to attend similar conferences together.

From the available data set we have built co-authorship similarity, citation similarity and community similarity between the authors. The values are later used during recommendation process.

5.2 Method

The evaluation method is done in several steps, and assumptions are made as the process progresses. Output from every step is used in subsequent steps.

Currently, we only have the data of participants who have attended the event, but not participants of talks. CAMRS-2 being a system which recommends talks to the users, needs the details of participants who have attended the talks. For this, we have created a Google form (ICWL 2012 Questionnaire). The form holds all the talks that were presented during ICWL 2012. Every talk can be rated on a scale from 1 to 5, one being least relevant and 5 being most relevant to author’s research area.

This form is sent to the participants of ICWL 2012 through an e-mail. The recipient can now rate the talks he/she has attended and they are requested to provide their e-mail id or full name for later evaluation. A rated talk is assumed as attended, otherwise.

The form has been sent to all the participants of the event, and there were half a dozen replies. The amount of replies were low, and hence we had no choice but to provide recommendations to the replied authors and compare the results. The attendees of the talks are assumed to be the presenters and the chair of the subevent to which the talk belongs to. Not surprisingly the available data is least we could attain for the evaluation. However, we decide to proceed to run the recommendation algorithm and evaluate them.

The talk recommendation algorithm consumes time and author id as input to provide the recommendations. The author id set is built from the authors who have responded using the Google form. For the time parameter, a set of random timestamps were chosen and provided to the algorithm as input. The timestamps were chosen in such a way there is one hundred percent possibility that there exists at least one talk during the time-stamp. Not only that, since the event has two tracks, the time-stamps were built to fall into time-frames of talks of both tracks.

For every possible combination of author id and time-stamp, recommendations have been generated. As the event has multiple tracks, if two talks were recommended to
the author, they are placed in ascending ranking order. The recommended talk with rank 1 is believed to be more relevant than that of rank 2 talk. Later, the replies from Google form were written into the database.

Now for the evaluation of results, three different cases are considered. As the data we have from the form is converted into (author id, talk id, rating) format, we can conveniently chose author and talk pairs depending on the rating given by the author on the talk.

- **Case 1:** In this case no restrictions are laid on ratings of talks. Hence, all the talks attended by the author are considered.
- **Case 2:** In this case we lay a restriction on talks with lower ratings. Only the talks attended by author and been rated higher than 3 are considered. In other words, only the interesting talks of the authors are considered.
- **Case 3:** This case is similar to the previous case. But the rating is increased from 3 to 4. These talks can be considered as most perfectly apt to the author.

Now that we have three types of data sets, we can compare the recommendation to these data sets and evaluate our system.

### 5.3 Results and Comparison

Since the small data input sets we have, it is not completely possible to estimate the accuracy of the system. Nevertheless, we can still compare the results produced by CAMRS-2 to the results produced by CAMRS-1. We ran the CAMRS-1 algorithm on the available input data sets to make the comparison possible.

The results are presented in the table 5.1. The header row is comprised of three different cases, and the second row has total number of relevant (author, talk) sets of the respective case. The subsequent rows show number of matching pairs recommended by the two systems. Additionally, talks which are ranked as 2 by the systems are also presented in the table.

As can be seen from the table 5.1, CAMRS-2 outperforms CAMRS-1 in all the three cases. The community analysis has given the CAMRS-2 high leverage over CAMRS-1. Not just CAMRS-2 is better than CAMRS-1, the results are encouraging too.
5 Evaluation

It can be seen in table 5.1 except for in case-1, in rest of the cases the talk recommendations matched with the original author choices are higher than 50 percent. In the case-3, the system almost predicted 60 percent of author choices. Though in case-1 the recommendations made matched less than 50 percent, it can considered as good scenario. Since in case-1 there are no restrictions on the ratings, all the talks attended by authors have been considered. And close to 45 percent of talks were irrelevant to authors research area (from case-2), thus proving the system did not make false positive recommendations to the authors.

With limited amount of data, the system managed to perform good. Having complete details could improve the recommendations dramatically. In conclusion, the CAMRS-2 outperforms its predecessor and can make healthy recommendations even with minimal amount of details.
6 Conclusion and Future Work

6.1 Conclusion

The goal of the master thesis was to analyze previous existing system, CAMRS, and to improve the quality of recommendations. We have realized the goal can be achieved by employing community analysis and topic modeling techniques. In the next part we dedicated to the survey of state-of-the-art techniques and technologies in the fields of recommender system, SNA, community detection, and topic modeling.

Community analysis and topic modeling, which are considered as building blocks of the system are analyzed and explained in great detail. The importance of communities, detecting community structure, and approaches to detect community are exhibited. Greedy approach, proposed by Clauset and Newman is explained, as this approach perfectly serves the purpose of our requirement. As for topic modeling, the importance of it is discussed and different models of topic modeling are introduced. Statistical model which is adapted to detect the topics is explained in lucid manner. Finally, we had brief discussion about systems which try to serve the same purpose, but fall behind in many ways.

The next part of thesis is dedicated to the conceptual approach of the system. In this part we have modeled the communities and the academic events. The community is modeled according to the ANT model. The structure of academic event and different session formats are explained comprehensively. Possible scenarios where the system can be used is also presented in this part. Finally, this part is concluded with a use case diagram and the recommendation approach. The recommendation approach is elucidated to very basic detail, so as to provide and execute the system accurately.

The fourth part of the thesis is a complete discussion about the tools and technologies used during the development of the system. The part begins with the presentation of the system architecture. Description is made about the architecture chosen, and in relation with the architecture the system is explained. Later on, all the tools, technologies, and libraries used during development of different parts of the system are presented. Comparisons are made when an alternative technology is chosen from that of previous implementation. Several new tools and technologies were employed during building of the system. All of these are explained and their contemporary technologies are also mentioned.

In the penultimate part, we have evaluated our system and presented the results. Clear explanation of data set gathering, method of evaluation, and results are presented in this part. To prove the system improvement in the quality of recommendations, both CAMRS-2 and CAMRS-1 were executed and the results are compared and
6 Conclusion and Future Work

exhibited. From the results, the current system outperforms its predecessor in every possible way.

Finally, it can concluded that the system has achieved its goal by improving the quality of recommendations. The community analysis and topic modeling were proven to provide high leverage and make the system a better recommender system.

6.2 Future Work

The future work of the system can be directed in numerous possible directions. However, we would like propose few areas where we would like to see the system evolve.

6.2.1 Communities

In our current approach, the communities detected are non-overlapping communities, i.e., an author can only belong to one community. However, it is a profound fact that there are quite a few authors who are interdisciplinary. Confining such authors to one community might result in loss of invaluable information. Therefore, we would like to propose to evaluate the system by detecting overlapping communities instead of non-overlapping communities.

6.2.2 Topic Detection

Our current approach is fastest way to detect topics among a million documents. The statistical approach adopted, does not consider any other information in the document but title. Though this has proven to be a successful way to topic detection, it would be intriguing to see machine learning topic modeling approaches come into play. However, one should remember, machine learning approaches need training data sets.

6.2.3 Application Improvement

CAMRS-2 provides all the required functionality to the users. However, the security of the application needs to be improved to a great extent. Current architecture does not have a concrete method to verify author identity, which could allow impersonation (accidental/deliberate). This can be eradicated by requesting user to provide their e-mail address and verifying their identity.

The user-name and password are currently stored as class variables on the client system. Though this implementation is not completely vulnerable to attacks, it is still considered as bad implementation. The current class implementation can be replaced by either cookies or Android shared variables. Every request sent from client to server only consists required parameters. This approach could lead to several attacks, since REST does not encourage usage of sessions, the developer need to figure out a different way to make requests. One best possibility is to encrypt the URL request and add
the encrypted response as the query parameter. Once the server receives the request it can match the URI request with the key query parameter, which makes a match gets the response.
6 Conclusion and Future Work
7 Bibliography


Bibliography


Upphovsrätt

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare åfrån publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida www.ep.liu.se

Copyright

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

©Bharath Reddy Pasham