A Project Report

On

**DEVELOPING VOTING RECOMMENDER SYSTEM FOR ONLINE SOCIAL USERS**

Submitted for partial fulfilment of the requirements for the award of the degree of

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE AND ENGINEERING**

BY

**S.M.Manasa (160114733130)**

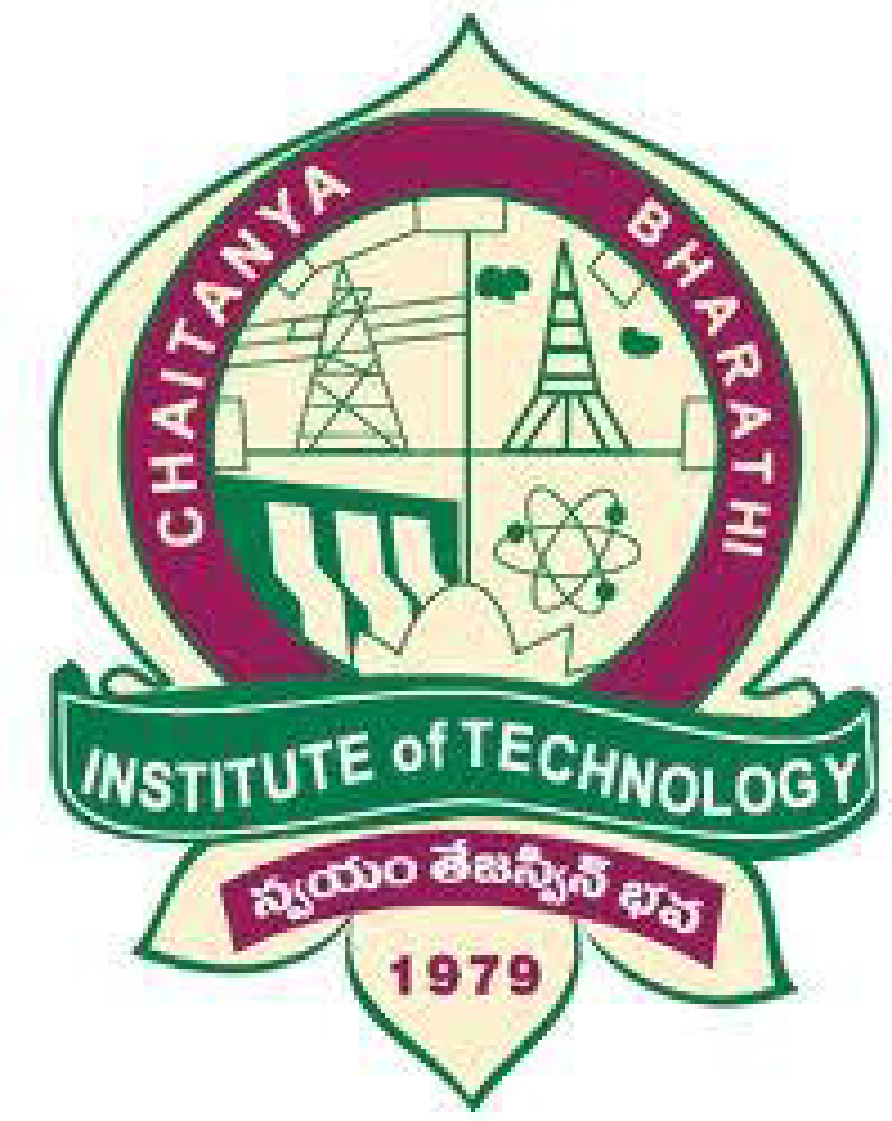
Under the guidance of

**M.Swamy Das**

Professor & HOD

Department of CSE

CBIT, Hyderabad.

. 

**Department of Computer Science and Engineering**

**Chaitanya Bharathi Institute of Technology(A)**

**Gandipet, Hyderabad -500075**

**April– 2018**

****

**CERTIFICATE**

This is to certify that the project work entitled “**DEVELOPING VOTING RECOMMENDER SYSTEM FOR ONLINE SOCIAL USERS”** is a bonafide work carried out by **S.M.MANASA (160114733130)** in partial fulfilment of the requirements for the award of Degree of **BACHELOR OF ENGINEERING** in **COMPUTER SCIENCE AND ENGINEERING** by **CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**, Hyderabad, under our guidance and supervision.

The results embodied in this report have not been submitted to any other university or institute for the award of any degree or diploma.

Head of the Department

**Dr. M. Swamy Das**

Professor

Department of CSE

CBIT, Hyderabad

**DECLARATION**

We hereby declare that the research work entitled "**DEVELOPING VOTING RECOMMENDER SYSTEM FOR ONLINE SOCIAL USERS**" is original and bonafide work carried out by us as a part of fulfilment for Bachelor of Engineering in Computer Science and Engineering, Chaitanya Bharathi Institute of Technology, Gandipet, Hyderabad, under the guidance of **M.Swamy Das**, Professor, Department of CSE, CBIT.

No part of the project work is copied from books/journals/internet and wherever the partition is taken, the same has been duly referred in the text. The reported are based on the project work done entirely by us and not copied from any other source.

**S.M.Manasa (160114733130)**

**ACKNOWLEDGEMENT**

We would like to express our deep felt appreciation and gratitude to **Sri**. **M.Swamy Das**, our project guide, for her skillful guidance, constant supervision, timely suggestion, keen interest and encouragement in completing the individual seminar within the stipulated time.

We wish to express our gratitude to **Dr. Y.Rama Devi,** Professorand **K.Mary Sudha Rani**, Assistant Professor, Project Coordinators, who have shown keen interest and even rendered their valuable time and guidance in terms of suggestions and encouragement.

We are honored to express our profound sense of gratitude to **Dr. M Swamy Das,** Head of Department, CSE, who has served as a host of valuable corrections and for providing us time and amenities to complete this project.

We gratefully express our thanks to **Dr.** **P.Ravinder Reddy,** Principal of our college and the management of CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY for providing excellent academic and learning environment in the college

We wish to express our heartfelt gratitude to the Members of Staff and all others who helped us in bringing up our project. We would also like to thank the Lab assistants and Programmers for helping us through our project.

**S.M.Manasa (160114733130)**

**ABSTRACT**

Social voting is an emerging new feature in online social networks. It poses unique challenges and opportunities for recommendation. The proposed work is implementing a set of matrix factorisation (MF) and nearest-neighbour (NN)-based recommender systems (RSs) that explore user social network and group affiliation information for social voting recommendation. From the experiments with online voting dataset, we propose that social network and group affiliation information can significantly improve the accuracy of popularity-based voting recommendation. We observe that social and group information is much more valuable to cold users than to heavy users.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No.** |
| Figure 3.1 | System architecture design | 14 |
| Figure 3.2 | UML diagrams | 17 |
| Figure 3.3 | Data Flow Diagram | 18 |
| Figure 3.4 | Use case diagram | 19 |
| Figure 3.5 | Sequence diagram | 20 |
| Figure 3.6 | Collaboration diagram | 22 |
| Figure 3.7 | Activity Diagram | 23 |
| Figure 3.8 | Class Diagram | 24 |
| Figure 3.9 | Matrix Factorization Model | 25 |
| Figure 3.10 | Nearest Neighbour Metapath Model | 30 |
| Figure 3.11 | User Dataset |  |
| Figure 3.12 | Rating Dataset |  |
| Figure 5.1 | Home screen for voting recommender system | 35 |
| Figure 5.2 | The screen where dataset being uploaded | 34 |
| Figure 5.5 | The screen after running Matrix factorization Model | 40 |
| Figure 5.6 | The screen after running UGUV model | 41 |
| Figure 5.7 | The screen after running UUV model | 42 |
| Figure 5.8 | The screen after running UVUV model | 43 |
| Figure 5.9 | Top K recommendation for the users shown in a chart | 40 |

**TABLE OF CONTENTS**

|  |  |
| --- | --- |
| Certificate | i |
| Declaration | ii |
| Acknowledgement | iii |
| Abstract | iv |
| List of Figures | v |
| 1. Introduction | 1 |
| 1.1Objective | 1 |
| 1.2 Problem Definition | 1 |
| 1.3 Existing System | 2 |
| 1.4 Proposed System | 2 |
| 1.5 Assumptions | 3 |
| 1.6 Organization of Report | 3 |
| 2. Literature Survey | 4 |
| 3. Methodology | 14 |
| 3.1 System Design | 14 |
| 3.1.1 Diagramatic Representations | 14 |
| 3.1.1.1 UML | 14 |
| 3.1.1.2 Basic concepts | 15 |
| 3.1.1.3 Building blocks of UML | 15 |
| 3.1.1.4 Data Flow Diagrams | 18 |
| 3.1.1.5 Use case diagram | 19 |
| 3.1.1.6 Sequence diagram | 20 |
| 3.1.1.7 Collaboration diagram | 21 |
| 3.1.1.8 Activity diagram | 22 |
| 3.1.1.9 Class Diagram | 24 |
| 3.1.2. Proposed Algorithms | 24 |
| 3.1.2.1 Multichannel Matrix  Factorization based RS | 24 |
| 3.1.2.2 Nearest-Neighbor Methods | 26 |
| 3.1.2.2.1 UGUV Model | 27 |
| 3.1.2.2.2 UUV Model | 27 |
| 3.1.2.2.3 UVUV Model | 28 |
| 3.2 Implementation | 31 |
| 3.2.1. Modules Description | 31 |
| 3.3 Dataset Information |  |
| 4. Results and Discussions | 34 |
| 4.1 Final output screenshots | 35 |
| 4.2 Discussion on results | 40 |
| 5. Conclusion and Future Work | 43 |
| References | 45 |
| Appendix | 47 |

**1.INTRODUCTION**

* 1. **Objective**

Online Social networks (OSN), such as Facebook and Twitter, facilitate easy information sharing among friends. A user not only can share his/her updates, in forms of text, picture, and video, with his/her direct friends, but also can quickly distribute those updates to all our friends providing greater connectivity in social networks . Many OSNs now offer the social voting function, through which a user can share with friends her opinions, e.g., like or dislike, on various subjects, ranging from user statuses, profile pictures, to games played, products purchased, websites visited, and so on. Taking like– dislike type of voting’s one step further, some OSNs, e.g., Sina Weibo, empower users to initiate their own voting campaigns, on any topic of their interests, with user customized voting options. The friends of a voting initiator can participate in the campaign or re-tweet the campaign to their friends. Other than stimulating social interactions, social voting also has many potential commercial values. Advertisers can initiate voting’s to advertise certain brands. Product managers can initiate voting’s to conduct market research. E-commerce owners can strategically launch voting’s to attract more online customers.

* 1. **Problem Definition**

The proposed work is developing Recommender System for online social votings, i.e., recommending interesting voting campaigns to users. Different from the traditional items for recommendation, such as books and movies, social votings propagating along social links. A user is more likely to be exposed to a voting if the voting was initialized, participated, or re-tweeted by her friends. A voting’s visibility to a user is highly correlated with the voting activities in her social neighborhood. Social propagation also makes social influence more prominent: a user is more likely to participate in a voting if his/her friends have participated in the voting. Due to social propagation and social influence, a user’s voting behavior is strongly correlated with his / her social friends.

* To implement a set of Recommender System models for online social votings using voting traces.
* To learn user-voting interests by simultaneously mining information on user-voting participation, user–user friendship, and user group affiliation.
* To generate Popularity voting chart, Top-k voting’s for recommendation and Hit-Rate.
  1. **Existing System**

A user is more likely to be exposed to a voting if the voting was initialized, participated, or re-tweeted by her friends. A voting’s visibility to a user is highly correlated with the voting activities in her social neighborhood. Social propagation also makes social influence more prominent: a user is more likely to participate in a voting if her friends have participated in the voting. Due to social propagation and social influence, a user’s voting behavior is strongly correlated with her social friends. Social voting poses unique challenges and opportunities for Recommender Systems utilizing social trust information. Furthermore, voting participation data are binary without negative samples. It is, therefore, intriguing to develop RSs for social voting. The increasing popularity of social voting immediately brings forth the “information overload” problem: a user can be easily overwhelmed by various voting’s that were initiated, participated, or re-tweeted by her direct and indirect friends. It is critical and challenging to present the “right voting’s” to the “right users” so as to improve user experience and maximize user engagement in social voting’s.

* 1. **Proposed System**

Toward addressing the challenges, we implement a set of novel RS models, including matrix-factorization (MF)-based models and nearest-neighbor (NN)-based models, to learn user-voting interests by simultaneously mining information on user-voting participation, user–user friendship, and user-group affiliation. We generate Popularity voting chart which gives information about the most popular votings, this data can be helpful for the cold users. We also mine information about the user-follower activities in the OSN and generate hit-rate.

* 1. **Assumptions**

In this project, our assumption is that there are many users and each user has a friend/ follower and there is a group of users. And each user gives a rating to a product varying between (1-5). Based on the rating given by the user our Recommender Systems recommend the product to the customers.

* 1. **Organization Of Report**

This thesis is organized as follows:

* Section I deals with Introduction to the project, objective, assumptions, problem statement, existing and proposed system.
* Section II explains about Literature Survey conducted before starting off the project.
* Section III gives the design details of the system like overall architecture of the system and design(UML) diagrams and Implementation Modules
* Section IV deals about result analysis.
* Section V discusses about conclusions and future work.
* References followed by Appendix consist of sample code segments.

**2. LITERATURE SURVEY**

Xiwang Yang [1] presented a set of MF-based and NN-based RSs for online social voting. Through experiments with real data, we found that both social network information and group affiliation information can significantly improve the accuracy of popularity-based voting recommendation, especially for cold users, and social network information dominates group affiliation information in NN-based approaches. We demonstrated that social and group information is much more valuable to improve recommendation accuracy for cold users than for heavy users. This is due to the fact that cold users tend to participate in popular votings. In our experiments, simple metapath-based NN models outperform computation intensive MF models in hot-voting recommendation, while users’ interests for non hot votings can be better mined by MF models. This is the first step toward thorough study of social voting recommendation.

Devooght, R., Kourtellis, N & Mantrach, A. proposed that [2] Advanced and effective collaborative filtering methods based on explicit feedback assume that unknown ratings do not follow the same model as the observed ones (not missing at random). In this work, we build on this assumption, and introduce a novel dynamic matrix factorization framework that allows to set an explicit prior on unknown values. When new ratings, users, or items enter the system, we can update the factorization in time independent of the size of data (number of users, items and ratings). Hence, we can quickly recommend items even to very recent users. We test our methods on three large datasets, including two very sparse ones, in static and dynamic conditions. In each case, we outrank state-of-the-art matrix factorization methods that do not use a prior on unknown ratings. In this work we proposed a new, simple, and efficient, way to incorporate a prior on unknown ratings in several loss functions commonly used for matrix factorization. We experimentally demonstrated the importance of adding such a prior to solve the problem of collaborative ranking. We also tackled the problem of updating the factorization when new users, items and ratings enter the system. We believe that this problem is central to real applications of recommendation systems, because new users constantly enter those systems and the factorization must be kept up to date to give them recommendations immediately after their first few interactions with the platform. We offer an update algorithm whose complexity is independent of the size of the data, making it a good approach for large datasets. In the future, we would like to explore how our methods perform under real workloads of updates with variable arrival rates of ratings per user and item. Furthermore, we would like to test the performance of our methods in platforms built to analyze streams of data such as Storm, Twitter’s Distributed Processing Engines platform.

Chen, W., Hsu, W., & Lee, M. L. stated that [3] Given the vast amount of information on the World Wide Web, recommender systems are increasingly being used to help filter irrelevant data and suggest information that would interest users. Traditional systems make recommendations based on a single domain e.g., movie or book domain. Recent work has examined the correlations in different domains and designed models that exploit user preferences on a source domain to predict user preferences on a target domain. However, these methods are based on matrix factorization and can only be applied to two-dimensional data. Transferring high dimensional data from one domain to another requires decomposing the high dimensional data to binary relations which results in information loss. Furthermore, this decomposition creates many matrices that need to be transferred and combining them in the target domain is non-trivial. Separately, researchers have considered using social network information to improve recommendation. However, this social network information has not been explored in cross domain collaborative filtering. In this work, we propose a generalized cross domain collaborative filtering framework that integrates social network information seamlessly with cross domain data. This is achieved by utilizing tensor factorization with topic based social regularization. This framework can transfer high dimensional data without the need for decomposition by finding shared implicit cluster-level tensor from multiple domains. Extensive experiments conducted on real world datasets indicate that the proposed framework outperforms state-of-art algorithms for item recommendation, user recommendation and tag recommendation. In this work, we have presented a novel collaborative filtering method for integrating social network and cross domain network in a unified framework via latent feature sharing and cluster-level tensor sharing. This framework utilizes data from multiple domains and allows the transfer of useful knowledge from auxiliary domain to the target domain. The results of extensive experiments performed on real world datasets show that our unified framework outperforms the state-of-the-art techniques in all the three recommendation tasks. We have also implemented the algorithm on a map-reduce infrastructure and have demonstrated its scalability.

Sachan, A., & Richariya, V proposed that [4] In recent days Product advertisement and viewer’s choice are two important parts of marketing. These two parts generate a system known as Recommender system. Recommender system plays a significant role in internet technology for data gathering and rating the data. There are four kinds of filtering techniques used in Recommender Systems. They are demographic, content, collaborative and hybrid. The most commonly used technique is collaborative filtering. Here, we describe a little about first three techniques but we mainly focus on collaborative filtering, their types and their major challenges such as cold start problem, data sparsely, scalability and accuracy etc. Collaborative Filtering is commonly and popularly used filtering technique even though it has some issues related to accuracy and scalability etc. There have been many researches and results given by many authors. They all focuses on Scalability, Cold Start, Sparsity and accuracy. But more work was not done on sparsity issue. Since today internet data is growing increasingly that’s why sparsity issue also increases as new records, items, things, music, data etc are increasing and loaded day by day. In future work, we research on sparsity issue as it is also the important challenge that recommender system faces today and in future also.

Zhang, Y., Cao, B., & Yeung, D. Y. mentioned that [5] Collaborative filtering is an effective recommendation approach in which the preference of a user on an item is predicted based on the preferences of other users with similar interests. A big challenge in using collaborative filtering methods is the data sparsity problem which often arises because each user typically only rates very few items and hence the rating matrix is extremely sparse. We address this problem by considering multiple collaborative filtering tasks in different domains simultaneously and exploiting the relationships between domains. We refer to it as a multi-domain collaborative filtering (MCF) problem. To solve the MCF problem, we propose a probabilistic framework which uses probabilistic matrix factorization to model the rating problem in each domain and allows the knowledge to be adaptively transferred across different domains by automatically learning the correlation between domains. We also introduce the link function for different domains to correct their biases. Experiments conducted on several real-world applications demonstrate the effectiveness of our methods when compared with some representative methods. We have addressed the multi-domain collaborative filtering problem, in which multiple rating prediction problems were jointly learned. We propose a probabilistic model which considers the correlation between different domains when leveraging all rating data together. Experiments conducted on several recommendation datasets demonstrate the effectiveness of our methods. Another way to alleviate the data sparsity problem in CF is to apply active learning. Unlike many conventional machine learning methods which wait passively for labeled data to be provided in order to start the learning process, active learning takes a more active approach by selecting unlabeled data points to query some oracle or domain expert to reduce the labeling cost

Cremonesi, P., Garza, P., Quintarelli, E., & Turrin, R. proposed that [6] Traditional recommender systems provide recommendations of items to users; recently, some of them also consider the context related to predictions. We propose a technique that relies on classical recommendation algorithms and post-filters recommendations on the basis of contextual information available for them. Association rules are exploited to identify the most significant correlations among context and item characteristics. The mined rules are used to filter the predictions performed by traditional recommender systems to provide contextualized recommendations. Our experimental results show that the proposed approach allows improving the output of classical algorithms proposed in the literature, especially in the case of unpopular items. We showed how contextual rules, representing frequent relationships between context information and the characteristics of the rated items, allow increasing the recall of state-of-the-art collaborative recommender systems. Our approach can be profitably applied to both personalized and non-personalized recommender systems.

Mnih, A., & Salakhutdinov, R. R. stated that [7] Many existing approaches to collaborative filtering can neither handle very large datasets nor easily deal with users who have very few ratings. We present the Probabilistic Matrix Factorization (PMF) model which scales linearly with the number of observations and, more importantly, performs well on the large, sparse, and very imbalanced Netflix dataset. We further extend the PMF model to include an adaptive prior on the model parameters and show how the model capacity can be controlled automatically. Finally, we introduce a constrained version of the PMF model that assumes that users who have rated similar sets of movies are likely to have similar preferences. The resulting model can generalize considerably better for users with very few ratings.

Koren, Y. stated that [8] Recommender systems provides users with some personalized advices for products and services. These systems also depend on Collaborative Filtering, where in past transactions are analyzed to establish connection between products and users. The two more successful approaches to Collaborative Filtering are latent factor models, which profile both users and products directly, and neighborhood models, which analyze the similarity between products or users. In this process, we introduce some innovations to both approaches. The factor model and neighborhood model can be combined now, leading to building a more accurate combined model. Further accuracy improvements are achieved by extending the models to exploit both explicit and implicit feedback by the users. These methods are tested on the Netflix data. Results are better than those previously published on that dataset. In addition to this, we suggest a new evaluation metric, which signifies the differences among methods, based on those methods performances at a Voting Recommender task. These proposed improvements to two of the most popular approaches to Collaborative Filtering. We suggested a new neighborhood based model, which unlike previous neighborhood methods, is based on formally optimizing a global cost function. This leads to improved prediction accuracy, while maintaining merits of the neighborhood approach such as explain ability of predictions and ability to handle new users without re-training the model. Second, we introduced extensions to SVD-based latent factor models that allow improved accuracy by integrating implicit feedback into the model. One of the models also provides advantages that are usually regarded as belonging to neighborhood models, namely, an ability to explain recommendations and to handle new users seamlessly. In addition, the new neighborhood model enables us to derive, for the first time, an integrated model that combines the neighborhood and the latent factor models. This is helpful for improving system performance, as the neighborhood and latent factor models address the data at different levels and complement each other. Quality of a recommender system is expressed through multiple dimensions including: accuracy, diversity, ability to surprise with unexpected recommendations, explain ability, appropriate Voting Recommenders, and computational efficiency. Some of those criteria are relatively easy to measure, such as accuracy and efficiency that were addressed in this work. Some other aspects are more elusive and harder to quantify. We suggested a novel approach for measuring the success of a top-K recommender, which is central to most systems where a few products should be suggested to each user.

Xiong, L., Chen, X., Huang, T. K., Schneider, J., & Carbonell, J. G. mentioned that [9] Real-world relational data are seldom stationary, yet traditional collaborative filtering algorithms generally rely on this assumption. Motivated by our sales prediction problem, we propose a factor-based algorithm that can take time into account. By introducing additional factors for time, we formalize this problem as a tensor factorization with a special constraint on the time dimension. Further, we provide a fully Bayesian treatment to avoid tuning parameters and achieve automatic model complexity control. To learn the model, we develop an efficient sampling procedure that is capable of analyzing large-scale data sets. This new algorithm, called Bayesian Probabilistic Tensor Factorization (BPTF), is evaluated on several real-world problems including sales prediction and movie recommendation. Empirical results demonstrate the superiority of our temporal model. By introducing a set of additional time features to traditional factor-based collaborative filtering algorithms, and imposing a smoothness constraint on those factors, BPTF can learn the global evolution of latent features. An efficient MCMC procedure is proposed to realize automatic model averaging and largely eliminates the need for tuning parameters on large-scale data. We show extensive empirical results on several real-world data sets to illustrate the advantage of temporal model over static models. In future works, we may adopt other types of observational models other than Gaussian, such as the exponential family distributions. For example, a Poisson model will be better suited for our sales problem. However, this may lead to more complicated posterior distributions for which Gibbs sampling is not applicable.

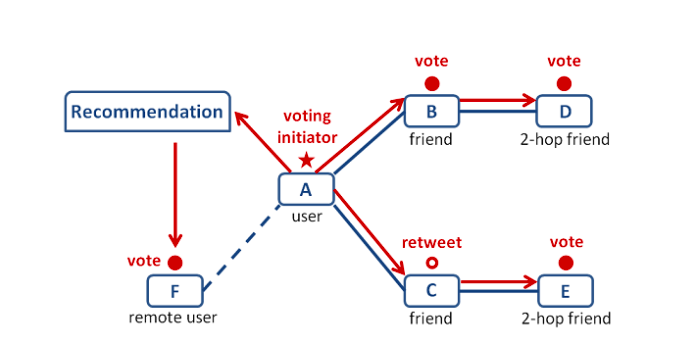
Ma, C. C. proposed that [10] As the market of electronic commerce grows explosively, it is important to provide customized suggestions for various consumers. Collaborative filtering is an important technique which models and analyzes the preferences of customers, and gives suitable recommendations. Singular Value Decomposition (SVD) is one of the popular algorithms used for collaborative filtering. However, directly applying conventional SVD algorithms to collaborative filtering may result in poor performance. In this report, we discuss problems that beginners may face and present effective SVD variants for collaborative filtering. A careful implementation of Singular Value Decomposition is effective for collaborative filtering. Our experiments show that, in general, batch learning or incomplete incremental learning requires a smaller learning rate, and has an unstable performance than complete incremental learning. We find that incremental learning, especially completely incremental learning which updates values after looking at a single training score, is the best choice for collaborative filtering with millions of training instances.

Adomavicius, G., & Tuzhilin, A. depicted [11] an overview of the recommender systems and describes the current scenario recommendation methods that are usually classified into the following categories: content-based approach, hybrid recommendation approach and collaborative approach. These improvements include, an enhancement in understanding the users and items, and incorporation of the contextual information into the recommendation process, and support for multi criteria ratings, and a provision of more flexible and less intrusive type of recommendations. Recommender systems made significant progress over the last decade when numerous content-based, collaborative, and hybrid methods were proposed. All the improvements in the present generation recommender systems surveyed still requires more enhancements to make recommendation approaches more effective. We reviewed various drawbacks of the current recommendation methods and discussed possible extensions that can provide better recommendation capabilities.

**3. Methodology**

**3.1 System Design**

Any user can initiate a voting campaign. After a voting is initiated, there are two major ways through which other users can see the voting and potentially participate. The first way is social propagation: after a user initiated or participated in a voting, all his/her followers can see the voting; a user can also choose only retweet a voting to his followers without participation. The other way is through voting recommendation list, which consists of popular votings and personalized recommendation. Recommendation List which consists of popular votings.

****

**Figure 3.1. The System Architecture for Voting Recommender System**

Figure 3.1 represents that if a target user initiates a post his followers can view the post and can give their ratings. If the followers participate in a voting it is also visible to his own friends/followers. The remote user can view the recommendation list which is obtained by mining user-user friendship information.

**3.1.1 Diagramatic representations**

**3.1.1.1 UML**

The Unified Modelling Language (UML) is a standard language for writing software blue prints. The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modelling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system.

**3.1.1.2 UML Concepts**

The Unified Modelling Language (UML) is a standard language for writing software blue prints. The UML is a language for

* Visualizing
* Specifying
* Constructing
* Documenting the artifacts of a software intensive system.

The UML is a language which provides vocabulary and the rules for combining words in that vocabulary for the purpose of communication. A modelling language is a language whose vocabulary and the rules focus on the conceptual and physical representation of a system. Modelling yields an understanding of a system.

**3.1.1.3 Building Blocks of the UML**

The vocabulary of the UML encompasses three kinds of building blocks:

**Things**

Things are the abstractions that are first-class citizens in a model; relationships tie these things together; diagrams group interesting collections of things.

There are four kinds of things in the UML:

* Structural things
* Behavioral things
* Grouping things
* Annotational things

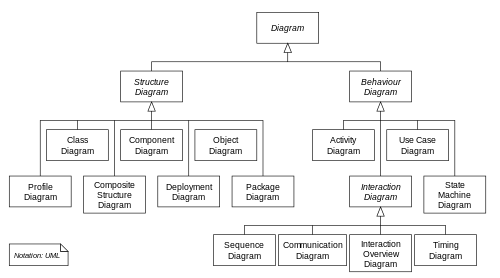
**Relationships**

There are four kinds of relationships in the UML:

* A **dependency** is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing (the dependent thing).
* An **association** is a structural relationship that describes a set links, a link being a connection among objects. Aggregation is a special kind of association, representing a structural relationship between a whole and its parts.
* A **generalization** is a specialization/ generalization relationship in which objects of the specialized element (the child) are substitutable for objects of the generalized element(the parent).
* A **realization** is a semantic relationship between classifiers, where in one classifier specifies a contract that another classifier guarantees to carry out.

**Diagrams**

UML has many types of diagrams, which are divided into two categories. Some types represent structural information, and the rest represent general types of behavior, including a few that represent different aspects of interactions. These diagrams can be categorized hierarchically as shown in the figure 3.2.

****

**Figure 3.2 all the UML Diagrams**

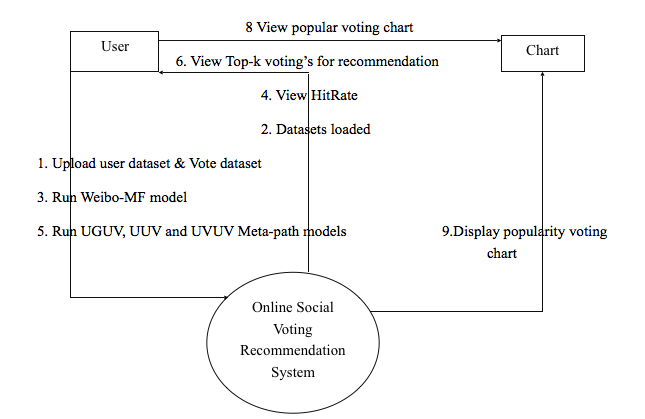
Structure diagrams emphasize the things that must be present in the system being modeled. Since structure diagrams represent the structure, they are used extensively in documenting the software architecture of software systems. For example, the component diagram describes how a software system is split up into components and shows the dependencies among these components. Behavior diagrams emphasize what must happen in the system being modeled. Since behavior diagrams illustrate the behavior of a system, they are used extensively to describe the functionality of software systems. As an example, the activity diagram describes the business and operational step-by-step activities of the components in a system. Interaction diagrams, a subset of behavior diagrams, emphasize the flow of control and data among the things in the system being modeled. For example, the sequence diagram shows how objects communicate with each other in terms of a sequence of messages.

**3.1.1.4 Data Flow Diagrams**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored.

**DFD Level 1**

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, we try to describe the system using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.



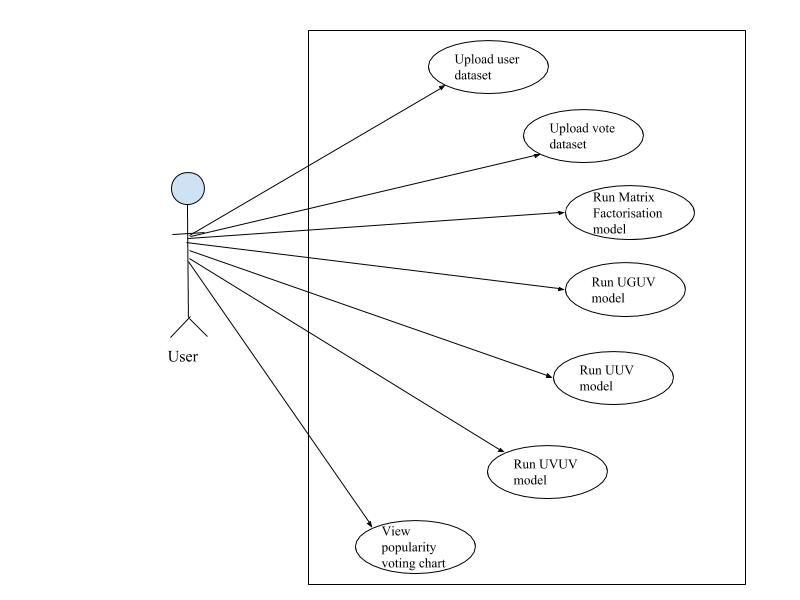
**Figure 3.3 DFD for Recommendation System**

The Figure 3.4 represents Level 1 DFD for the recommendation system. The user first uploads the vote dataset and user dataset. Later he may choose to run Matrix Factorization model, UUV model or UVUV model. They give us hit-rate of the voting, User scores according to the dataset and Top-k votings for recommendation respectively where we can alter the value of k. The top-k votings can be displayed in the form of popularity voting chart.

**3.1.1.5 Use Case Diagram**

A use case diagram in the [Unified Modelling Language](http://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of behavioural diagram defined by and created from a [Use-case analysis](http://en.wikipedia.org/wiki/Use-case_analysis). Its purpose is to present a graphical overview of the functionality provided by a system in terms of [actors](http://en.wikipedia.org/wiki/Actor_%28UML%29), their goals (represented as [use cases](http://en.wikipedia.org/wiki/Use_case)), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors.



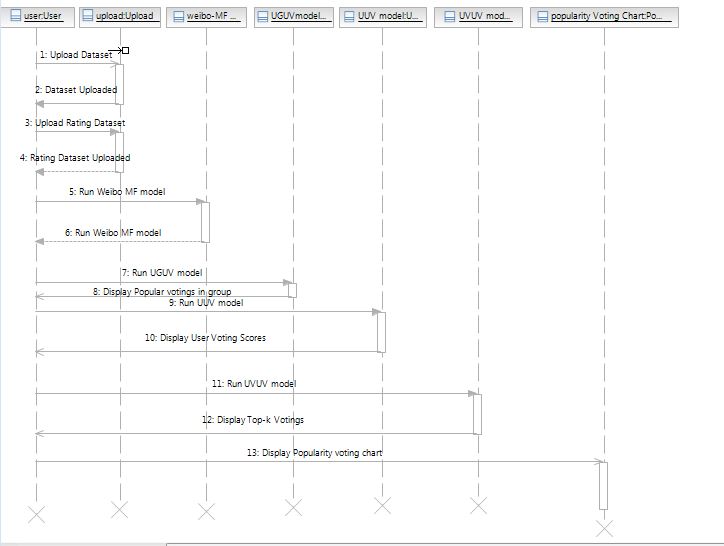
**Figure 3.4 the Use case diagram for Voting Recommender System**

In the Figure 3.5, One actor. 1.User. User uploads datasets and views the results as well as views popularity voting chart

**3.1.1.6 Sequence Diagram**

A sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

Sequence diagrams demonstrate the behaviour of objects in a use case by describing the objects and the messages they pass. The diagrams are read left to right and descending.



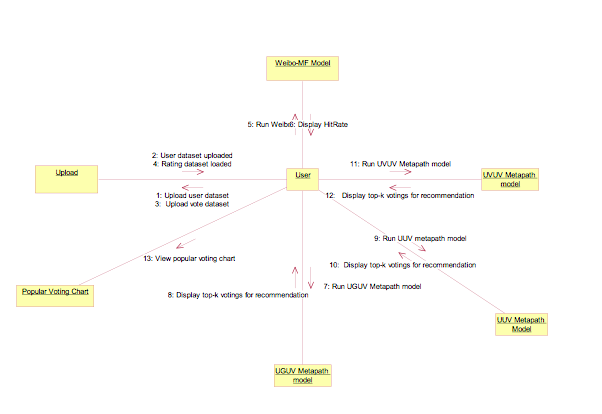
**Figure 3.5 the Sequence Diagram for Voting Recommender System**

In the Figure 3.6 user selects a dataset and uploads the dataset. By choosing to run Matrix Factorization model the System returns hit-rate of the product. If user runs, UUV model, the system returns the user scores of the individual users in the dataset. If the user runs UVUV model, the system returns the Top-k votings in the given dataset also a popularity voting chart of the same is displayed.

**3.1.1.7 Collaboration diagrams**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML).

A collaboration diagram resembles a flowchart that portrays the roles, functionality and behavior of individual objects as well as the overall operation of the system in real time. Objects are shown as rectangles with naming labels inside. These labels are preceded by colons and may be underlined. The relationships between the objects are shown as lines connecting the rectangles. The messages between objects are shown as arrows connecting the relevant rectangles along with labels that define the message sequencing.

**Figure 3.6 the Collaboration Diagram for Voting Recommender System**

In the Figure 3.7 the user object interacts with Matrix factorization object, UUV, UVUV, UGUV objects and Upload object to upload datasets. The user object runs model and the respective object provides with respective results. The user object runs the popularity voting chart object and it will in turn display the popularity voting chart.

**3.1.1.8 Activity Diagram**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

Upload User dataset

Upload Vote Dataset

Matrix Factorziation model

UGUV model

UUV model

UVUV model

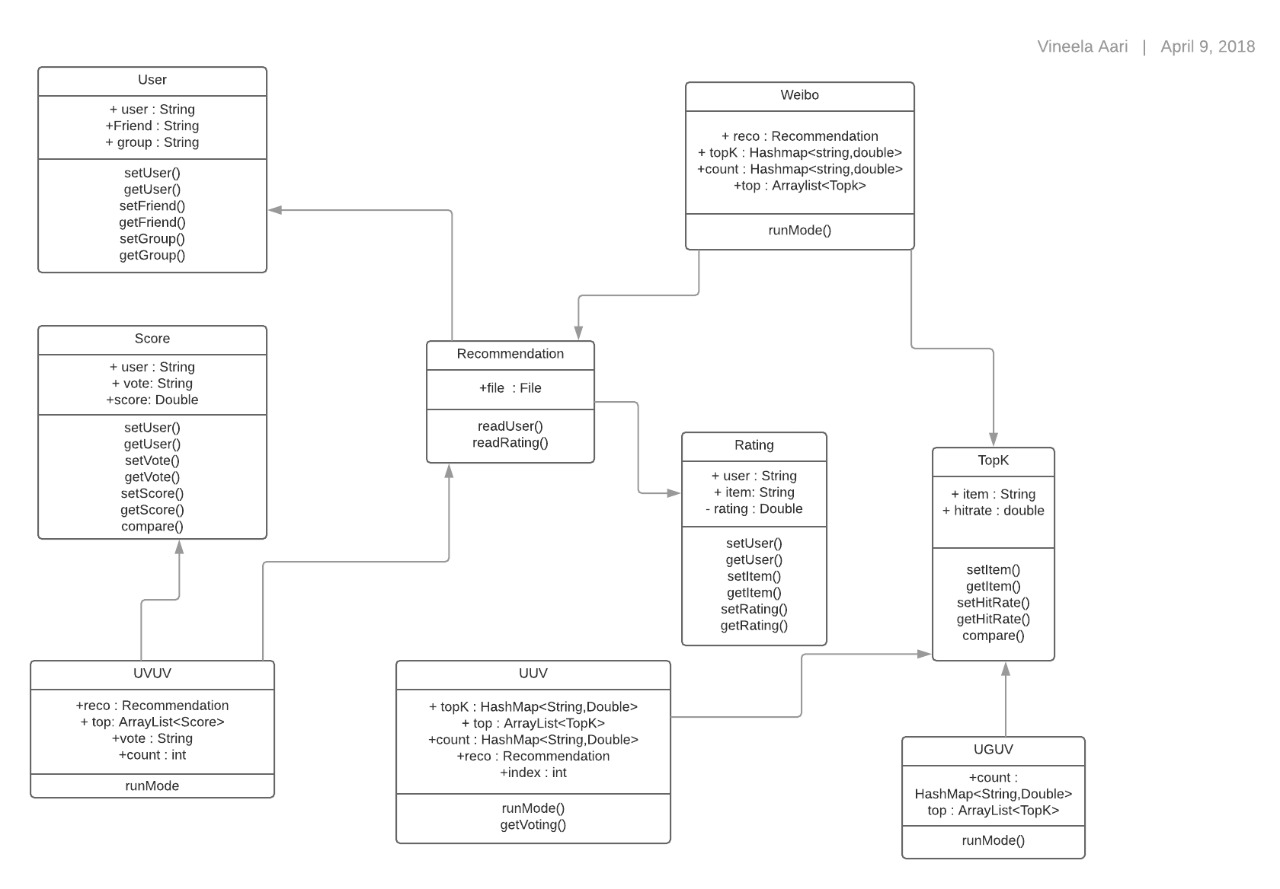
Popularity voting chart

**Figure 3.7 the Activity Diagram for Voting Recommender System**

In the Figure 3.8 User activities are uploading the dataset and selecting a model to run. The system displays the results by running the algorithm. Popularity chart can also be viewed by clicking on it.

**3.1.1.9 Class diagram:**

A class diagram is an illustration of the relationships and source code dependencies among classes in unified modelling language.

****

**Figure 3.8 the class diagram for Voting Recommender system**

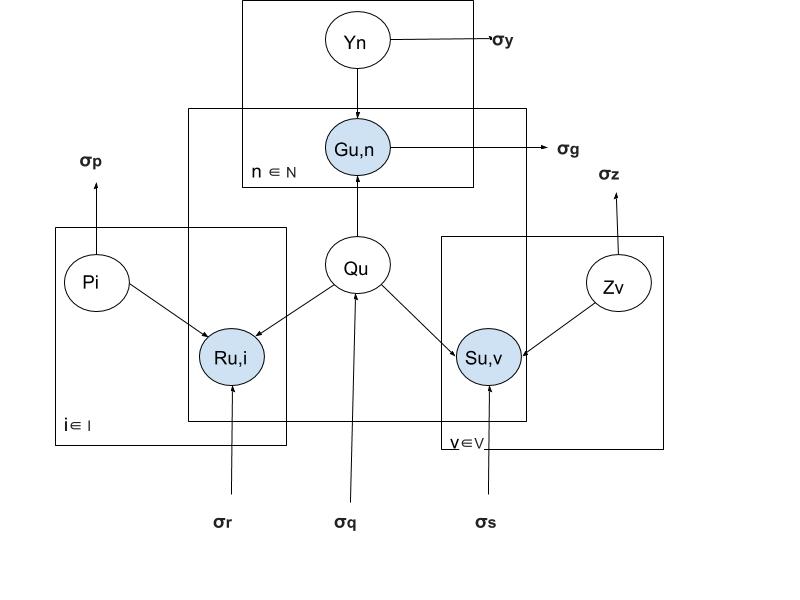
The class diagram in figure 3.8 gives information about all the classes in the voting recommender system. The classes namely are, User, Score, Recommendation, UVUV, UUV, UGUV, MF model, Rating, TopK. The dependencies between the classes are given in the figure.

**3.1.2 Proposed Algorithm**

**3.1.2.1 Multichannel Matrix Factorization based RS**

The social network information is represented by a matrix S ∈ Ru0×u0 , where u0 is the number of users. The directed and weighted social relationship of user u with user v (e.g., user u trusts/knows/follows v) is represented by a positive value S u,v ∈ (0, 1]. An unobserved social relationship is denoted by Su,v = sm, where sm = 0. The

user-group affiliation information is represented by matrix G ∈ R u0×n0 , where G u,n is binary and takes value 1 if user u joins group n, and 0 otherwise.



**Figure 3.10 Matrix Factorisation model**

The graphic model of Matrix Factorization is shown in Figure 3.10. The user-voting interaction Ru,i is determined by user latent feature Qu and voting latent feature Pi , user-group interaction Gu,n is determined by user latent feature Qu.

**Algorithm 1:**

**Data**: voting dataset

**Result**: Top-k Hit Rate for each post

// *Training part*

1 Load voting training data;

2 Initialize latent feature matrices Q and P;

// *Update latent features by ALS*

3 while Not Converge & Iteration Number is less than Iter\_Num do

4 Update Q by fixing P and minimizing Eq. (5);

5 Update P by fixing Q and minimizing Eq. (5);

6 end

// *Testing part*

7 for each user u in voting dataset for testing do

8 for each voting i in test dataset for user u do

9 Calculate the predicted rating of user u on voting i

as Rˆu,i = rm + Qu PTi ;

10 Put Rˆu,i into the queue recomm\_pool;

11 end

12 Sort recomm\_pool in an decreasing order according to the value of Rˆu,i ;

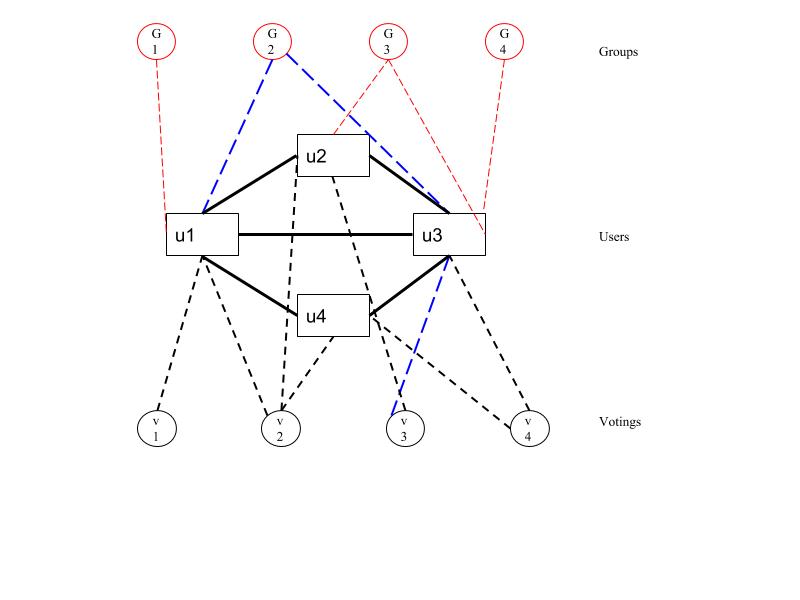
13 Select foremost K votings with largest Rˆu,i from recomm\_pool as the items for recommendation;

14 Calculate top-k hit rate for user u;

15 end

**3.1.2.2 Nearest-Neighbour Methods**

Other than MF approaches, NN-based recommendations have also been studied. NN methods are widely used in RSs. Thus, it is very intriguing to study the performance of NN models on social voting recommendation problem. In NN-based approaches, the neighbourhood of a user can be calculated using collaborative filtering, or it can be a set of directly or indirectly connected friends in a social network, or just a set of users with similar interests in a same group. This makes it convenient to incorporate social trust and user-group interaction into NN-based Voting Recommender .In this section, we try different approaches to construct nearest neighbourhood for a target user.

****

**Figure 3.11 Nearest Neighbour Metapath Models**

Figure 3.11 represents metapath models namely UUV, UGUV, UVUV where u denotes users, v denotes votings, G denotes groups. The dotted lines between different objects denotes the metapath. The line between users denotes user-user friendship. The dashed line between u and v denotes that the particular user took part in the particular voting. The dashed line between group and user denotes that the particular user belongs to the particular group.

1) **Metapath Neighbourhoods:** In heterogeneous information networks, objects are of multiple types and are linked via different types of relations or sequences of relations, forming a set of meta paths [15]. Metapath is a path that connects objects of different types via a sequence of relations. Different meta paths have different semantics. Sun et al. employ metapaths for clustering task in heterogeneous information networks.

a) **UGUV metapath**: the semantic of using U − G − U − V metapath for recommendation is finding users that in a same group with the target user,then recommending their votings to the target user. More specifically, UGUV works as follows.

1) For a target user u, UGUV searches for all the groups that u has joined. Denote the set of groups as Gu.

2) For each joined group g ∈ Gu, search for all the users that belong to group g.

3) Users in group g report their relevant votings.

4) Combine the reports of all groups. The score for a candidate voting i to the target user u is computed as Score u, i = g∈Gu,v∈gi

w(g)δi∈Iv (9)where δ is the Kronecker delta, Iv denotes the set of userv’s relevant votings, and w(g) is the weight of users in group g. In our later experiments, we try w(g) as a function of group size. We found that the best function of w(g) is to simply set w(g) = 1.

5) Rank recommended votings according to their scores,and return the top-k votings.

**Algorithm 2 Algorithm of UGUV Metapath**

**Data**: voting dataset

**Result**: Top-k votings for recommendation in group

1 Initialization;

2 for each target user u do

3 Find all groups g’s that user u has joined and put them in a set Gu;

4 for each joined group g ∈ Gu do

5 Find all user v’s in group g;

6 for each user v in group g do

7 User v reports its relevant votings and put them in a set Iv ;

8 for each candidate voting i ∈ Iv do

9 Scoreu,i+ = w(g);

10 end

11 end

12 end

13 Sort {Scoreu,i} in a decreasing order;

14 Return and recommend top k votings with highest scores to user u;

15 end

b) **UUV(m-hop) metapath**: the semantic of U −U − V (m − hop) metapath based recommendation is to recommend a target user the relevant votings of his followees within m-hops. UUV approach employs the BFS in social network to find users similar to the target user u. The scoring scheme is similar to the scheme employed in UGUV Score u,i = v∈N(s) ui ws(u, v) δi∈Iv (10)where N(s) u is the set of neighbors of u in social networks and w s(u,v) is the weight of user v. We set ws (u,v) = ws(dv ),where dv is the depth of user v in the BFS tree rooted at user u. By fixing 1-hop followees’ weight at ws(1) = 1, we tune the weight of 2-hop users. In our later experiments, we found the best value is ws(2) = 0.1. Votings are ranked according to their scores to form the recommendation list.

**Algorithm 3 Algorithm of UUV Metapath**

**Data**: voting dataset

**Result**: Voting scores of users

1 Initialization;

2 for each target user u do

3 Find all followees v’s within m-hops by BFS;

4 Put all those v’s in a set N(s) u ;

5 for each user v in N(s) u do

6 User v reports its relevant votings and put them in a set Iv ;

7 Set weight parameter ws(u,v) according to the

depth of user v in the BFS tree rooted at user u;

8 for each voting i in Iv do

9 Score u,i+ = w s(u,v);

10 end

11 end

12 Sort {Score u,i} in a decreasing order;

13 Return and recommend top k votings with highest scores to user u;

14 end

**c) UVUV metapath**: The semantic of U − V − U − V metapath-based recommendation is to find users that share votings with the target user, and the n recommend their relevant votings to the target user. For a target user u, UVUV works as follows.

1) Find all votings that u has participated in, and denote this voting set as Iu.

2) For each of the voting j ∈ I u, find the set of users who have participated in j. Denote the set of users as Nj .

3) Each user v ∈ Nj reports all the votings that he has participated in.

4) Aggregate the reports of all users to assign scores to votings as follows:

Score u, i = j∈ I u v∈ Njiw(v)δi∈Iv

**Algorithm 4 Algorithm of UVUV Metapath**

**Data**: Sina Weibo voting dataset

**Result**: Top-k votings for recommendation

1 Initialization;

2 for each target user u do

3 Find all votings j’s that user u has participated;

4 Put all those voting j’s into a set Iu;

5 for each voting j ∈ Iu do

6 Find all users v’s who ever participated in voting j and put them in a set Nj;

7 for each user v ∈ Nj do

8 Find all votings i’s that user v has participated and put them in a set Iv ;

9 for each voting i ∈ Iv do

10 Scoreu,i+ = w(v);

11 end

12 end

13 end

14 Sort {Scoreu,i} in a decreasing order;

15 Return and recommend top k votings with highest scores to user u;

16 end

**3.2. Implementation of Proposed Solution**

**3.2.1 Modules Description**

1.LOADING USER DATASET

2.LOADING VOTE DATASET

3.MATRIX FACTORIZATION ALGORITHM

4.UGUV METAPATH ALGORITHM

5.UUV METAPATH ALGORITHM

6.UVUV META PATH ALGORITHM

7.POPULAR VOTING CHART

**1. Loading User Data**

In this module, the user dataset which is from the Epinions website is loaded. Here our user dataset consists of user-id, follower/ friend id and the group to which he belongs to .This involves parsing the user dataset according to our need and load the user data and display it in the table format on the screen.

**2. Loading the Vote Dataset**

In this module, the vote dataset is from the Epinions website is loaded. Here our vote dataset consists of user-id, voting topic and the rating for that voting topic. This involves parsing the voting dataset according to our need and load the vote data and display it in the table format on the screen.

**3. Matrix Factorization Algorithm**

This module uses the matrix factorization method for recommendation of the voting topic to the remote users. This matrix factorization algorithm takes the rating of a product given by each user and gives the hit rate to that voting topic and recommends the remote users.

**4. UGUV Meta path Algorithm**

This module uses the nearest neighbor based recommendation of the voting topic to the remote users. This UGUV meta path algorithm particular product and gives the voting score to the voting topic and recommends the remote user.

**5. UUV Meta path Algorithm**

This module uses the nearest neighbor based recommendation. This UUV meta path algorithm the considers a particular user and displays the voting scores of the that particular user as an voting initiator and recommends to the remote user.

**6. UVUV Meta path Algorithm**

This module uses the nearest neighbor based recommendation. This UVUV meta path algorithm considers the user and the rating given by each user and gives the vote score based on the rating of the voting topic given by each user.

**7. Popular Voting Chart**

A popularity voting chart is generated based on which Recommendation results. So, based on seeing this popularity chart the customer can easily find which voting topic is recommended to him/her .The x-axis in the graph denotes the voting topic and the yaxis denotes the voting score.

**3.3 Dataset Information**

Epinions [12] is a consumer opinion site where users review various items, such as cars, movies, books, software, and so on, and assign ratings to the items. Users also assign trust values (i.e., a value of 1) to other users whose reviews and/or ratings they ﬁnd valuable. The dataset was collected by Paolo Massa in a 5-week crawl (November/December 2003) from the Epinions.com Web site.

The dataset contains 49,290 users who rated a total of 139,738 different items at least once, writing 664,824 reviews and 487,181 issued trust statements. Users and Items are represented by anonimized numeric identifiers. The dataset consists of 2 files.

ratings\_data.txt. : It contains the ratings given by users to items. Every line has the following format:

**user\_id item\_id rating\_value**

For example, 23 387 5 represents the fact "user 23 has rated item 387 as 5"

Ranges:

* user\_id is in [1,49290]
* item\_id is in [1,139738]
* rating\_value is in [1,5]

trust\_data.txt : It contains the trust statements issued by users. Every line has the following format:

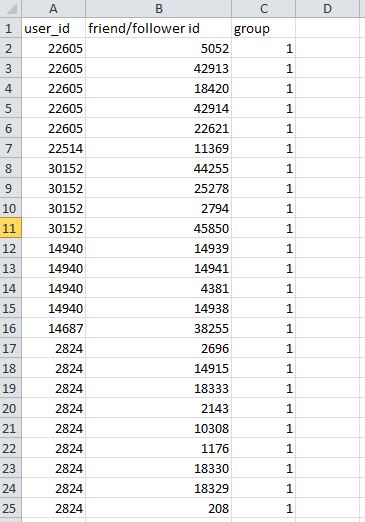
**source\_user\_id target\_user\_id**

For example, the line 22605 18420 represents the fact "user 22605 has expressed a positive trust statement on user 18420"

Ranges:

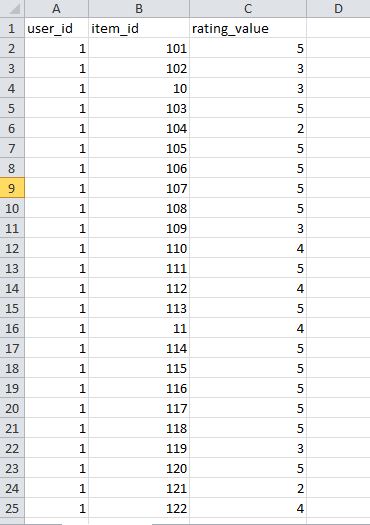
* source\_user\_id and target\_user\_id are in [1,49290]

If a target user trusted other users then the trusted users become friends/followers of the target user. Since trust is similar to social links/friendship in Online Social Network. We have considered item id as voting/post id to which the particular user has given the rating since we are considering Online Social Network where the items are nothing but posts initiated by various users. We are considering the user who has first rated the item as the voting initiator. As there is no explicit information about which group each user belongs, we considered all the users under one group.



**Figure 3.12 the user\_data.txt dataset**

In Figure 3.12 The dataset contains user\_id, follower/friend id, group as the features. The user id denotes unique identity of the target user. Follower id is the identity of the user who is a follower of the target user. Group denotes the group id to which the target user belongs. Here all the user come under same group so it is taken as 1.



**Figure 3.13 The rating\_data.txt dataset**

Figure 3.13 shows the rating dataset which contains user\_id, item\_id, rating\_value as features. User\_id denotes unique identity of the particular user. Item\_id denotes the post/voting that the user voted for. The rating\_value id is the value between 1 and 5 given to the post by the particular user.

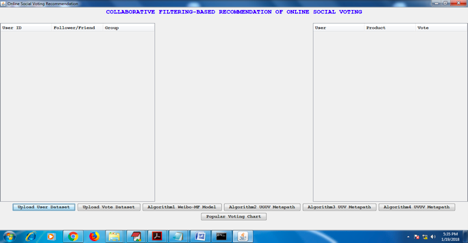
**4.RESULTS AND DISCUSSIONS**

**4.1 Results**

The proposed work presents a set of MF-based and NN-based Recommender Systems for online social voting. Through experiments with real data, it is observed that both social network information and group affiliation information can significantly improve the accuracy of popularity-based voting recommendation, especially for cold users, and social network information dominates group affiliation information in Nearest-Neighbor based approaches. The proposed work demonstrated that social and group information is much more valuable to improve recommendation accuracy for cold users than for heavy users. This is due to the fact that cold users tend to participate in popular voting’s. This is only the first step toward thorough study of social voting recommendation. As an immediate future work item, we would like to study how voting content information can be mined for recommendation, especially for cold voting’s. We are also interested in developing voting RSs customized for individual users, given the availability of multichannel information about their social neighbourhoods and activities.

**4.2 Screenshots**

**Home Screen**

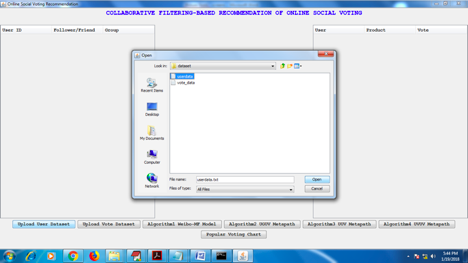


**Figure 5.1 shows the home screen**

**Description:**

This is the Home Screen, it will be loaded once you run your project batch file. It contains two tables , one table to load user data set and another table to load voting dataset. And it contains buttons to perform a respective task. One to load user dataset and another to load voting dataset. And remaining buttons to perform each algorithm.

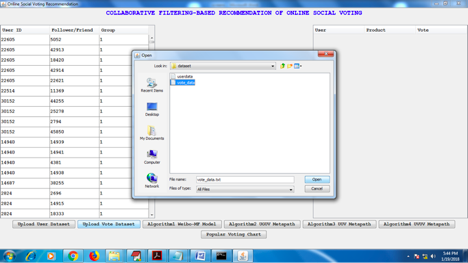
` **Upload User Dataset**



**Figure 5.2 shows the dataset being uploaded**

**Description**: The button Upload User dataset is used to load Epinions user data set. The user dataset contains user-id, his follower/friend and to the group which he belongs to.

**Upload Vote Dataset**

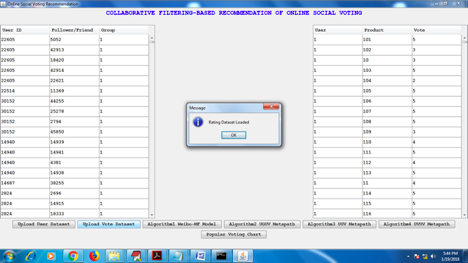


**Figure 5.3 shows the screen when user data is uploaded and vote data is selected**

|  |
| --- |
|  |

**Description**: The button Upload Vote dataset is used to load Epinions voting data set. The voting dataset contains user-id and the product to which he/she is giving voting and the rating contains the rating given by the respective user to the respective product.

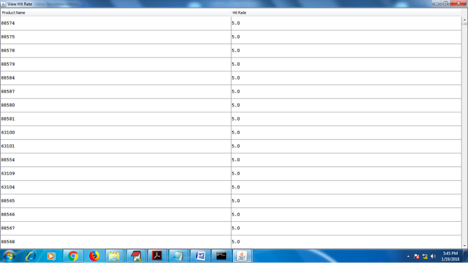
**User Dataset And Vote Dataset Loaded**



**Figure 5.4 shows the screen after loading both user and vote datasets**

**Description:** Epinions user and voting dataset is loaded into the tables. So, now this data can be used to perform our Algorithms.

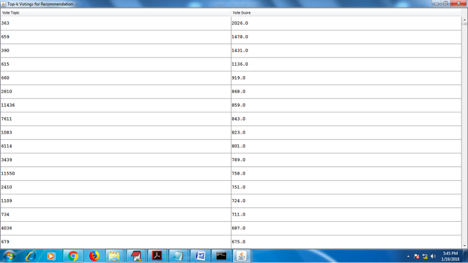
**Algorithm 1 Matrix Factorization Model**



**Figure 5.5 show the screen after running Matrix factorization Model**

**Description:** This module uses the matrix factorization method for recommendation of the voting topic to the remote users. This matrix factorization algorithm takes the rating of a product given by each user and gives the hit rate to that voting topic and recommends the remote users.

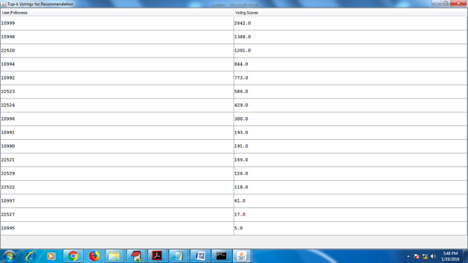
**Algortihm 2 UGUV Metapath Model**

****

**Figure 5.6 show the screen after running UGUV Model**

**Description**: This module uses the nearest neighbor based recommendation of the voting topic to the remote users. This UGUV meta path algorithm particular product and gives the voting score to the voting topic and recommends the remote user.

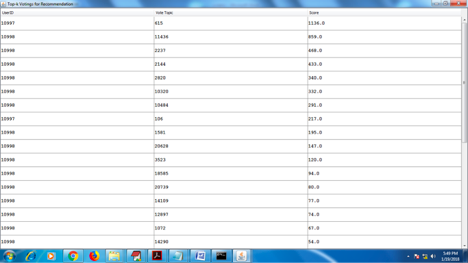
**Algorithm 3 UUV Meta Path model**

****

**Figure 5.7 show the screen after running UUV Model**

**Description**: This module uses the nearest neighbour based recommendation. This UUV meta path algorithm the considers a particular user and displays the voting scores of the that particular user as an voting initiator and recommends to the remote user.

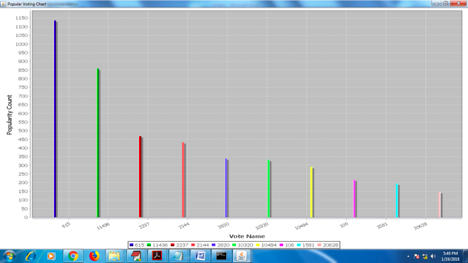
**Algorithm 4 UVUV metapath model**

****

**Figure 5.8 show the screen after running UVUV Model**

**Description :**This module uses the nearest neighbor based recommendation. This UVUV meta path algorithm considers the user and the rating given by each user and gives the vote score based on the rating of the voting topic given by each user.

**Popularity Voting Chart**

****

**Figure 5.9 show the Top-k voting recommendations in a chart.**

A **popularity voting chart** is generated based on which Recommendation results. So, based on seeing this popularity chart the customer can easily find which voting topic is recommended to him/her .The x-axis in the graph denotes the voting/post topic and the y-axis denotes the voting score (Popularity).

**6. CONCLUSION AND FUTURE WORK**

The proposed approach presents a set of MF-based and NN-based Recommender Systems for online social voting. Through experiments with real data, it is found that both social network information and group affiliation information can significantly improve the accuracy of popularity-based voting recommendation, especially for cold users, and social network information dominates group affiliation information in Nearest-Neighbor based approaches. The proposed approach demonstrated that social and group information is much more valuable to improve recommendation accuracy for cold users than for heavy users. This is due to the fact that cold users tend to participate in popular voting’s. In proposed approach, simple meta path-based NN models outperform computation intensive MF models in hot-voting recommendation, while users’ interests for non hot voting’s can be better mined by MF models. This is the first step toward thorough study of social voting recommendation.

As future work item, we would like to study how voting content information can be mined for recommendation, especially for cold voting’s. We are also interested in developing voting RSs customized for individual users, given the availability of multichannel information about their social neighborhoods and activities.

**REFERENCES**

[1] Xiwang Yang, "Collaborative Filtering-Based Recommendation of Online Social Voting", Published in: IEEE Transactions on Computational Social Systems ( Volume: 4, Issue: 1, March 2017 ). - **Base Paper**

[2] Devooght, R., Kourtellis, N & Mantrach, A.,”Dynamic matrix factorization with priors on unknown values”, In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining.

[3] Chen, W., Hsu, W., & Lee, M. L. (2013, August). Making recommendations from multiple domains. In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining.

[4] Sachan, A., & Richariya, V. (2013). A survey on recommender systems based on collaborative filtering technique. International journal of Innovations in Engineering and technology (IJIET), 2(2), 8-14.

[5] Zhang, Y., Cao, B., & Yeung, D. Y. (2012). “Multi-domain collaborative filtering”.

[6] Cremonesi, P., Garza, P., Quintarelli, E., & Turrin, R. (2011, October). Top-n recommendations on unpopular items with contextual knowledge. In 2011 Workshop on Context-aware Recommender Systems. Chicago.

[7] Mnih, A., & Salakhutdinov, R. R. (2008). Probabilistic matrix factorization. In Advances in neural information processing systems.

[8] Koren, Y. (2008, August). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining.

[9] Xiong, L., Chen, X., Huang, T. K., Schneider, J., & Carbonell, J. G. (2010, April). Temporal collaborative filtering with bayesian probabilistic tensor factorization. In Proceedings of the 2010 SIAM International Conference on Data Mining (pp. 211-222). Society for Industrial and Applied Mathematics.

[10] Ma, C. C. (2008). “A guide to singular value decomposition for collaborative filtering”.

[11] Adomavicius, G., & Tuzhilin, A. (2005).”Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions”. IEEE transactions on knowledge and data engineering.

[12] Trust-aware recommender systems. P Massa, P Avesani. Proceedings of the 2007 ACM conference on Recommender systems, 17-24