CHAPTER 1

INTRODUCTION

Recommender systems (RS) plays an important role to enable us to make effective use of information. Existing RS [3], [9] methods can be roughly categorized into three classes: Content-based methods, Collaborative Filtering [17], [21] (CF) based methods, Hybrid methods.

Content-based methods [5] adopt the profile of the users or products for recommendation. CF based [10], [11], [15], [20], [21] methods use past activities or preferences such as the ratings on items given by users for prediction, without using any user or product profiles. Hybrid methods [3], [4], [6] combine both content-based methods and CF based methods by ensemble techniques. Due to privacy issues, it is harder in general to collect user profiles than past activities.

In most traditional CF methods, only the feedback matrix, which contains either explicit feedback (also called ratings) or implicit feedback [12] on the items given by users, is used for training and prediction. Typically, the feedback matrix is sparse, since feedback is given to few items by the users. Due to this sparsity problem, traditional CF methods with only feedback information will suffer from unsatisfactory performance.

In many real applications, besides the feedback and item content information, there may exist relations (or networks) among the items [14], [16] which can also be helpful for recommendation. For example, if we want to recommend papers (references) to users the citation relations between papers are informative for recommending papers with similar topics. Other examples of item networks can be found in hyperlinks among webpages, movies directed by the same directors, and so on.

1.1 Types of Recommendation

Our task is to recommend articles that are not in their library but are potentially interesting. There are two types of recommendation:

- In-matrix prediction
- Out-matrix prediction

I. In-matrix prediction

This refers to the problem of making recommendations about those articles that have been rated by at least one user in the system. This is the task that traditional collaborative filtering [26] [10] can address.

II. Out-of-matrix prediction

This is sometimes called "cold start recommendation". This refers to the problem of making recommendations about those articles that have never been rated by any user in the system. A recommender system that cannot handle out-of-matrix prediction cannot recommend newly published papers to its users. This is the task that traditional collaborative filtering [6] [11] cannot resolve.

1.2 Recommendation setting

I. Item-Oriented setting

In an item-oriented setting [16], we need to recommend users to items. It is generally difficult to know which users could like an item if it has only been given feedback by one or two users. Companies face difficulties face when promoting new products (items). Moreover, user's ignorance of new items will result in less feedback on the new items, which will further harm the accuracy of their recommendations.

II. User-Oriented setting

For the user oriented setting, [9] we recommend items to users. It is difficult to predict what a user likes if the user has only given feedback to one or two items. However, in the real world, it is common to find that most users provide only a little feedback. Actually, providing good recommendations for new users with little feedback is more important than for frequent users since new users will only come back to the site (service) depending on the performance of the recommendation system. However, for frequent users, it is most likely that they are already satisfied with the site (service). If we manage to boost the recommendation accuracy for new or infrequent users, more of them will become frequent users, and then better recommendations can be expected with more training data. Therefore, improving the recommendation accuracy at an extremely sparse setting is key to getting the recommender systems working in a positive cycle.

To overcome the sparsity problem of CF-based models, many researchers have proposed to integrate auxiliary information into the model training and prediction procedures.

1.3 Topic Modeling

It is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents. Topic modeling [24] is a frequently used text-mining tool for discovery of hidden semantic structures in a text body. The "topics" produced by topic modeling techniques are clusters of similar words. A topic model captures this intuition in a mathematical framework, which allows examining a set of documents and discovering, based on the statistics of the words in each, what the topics might be and what each document's balance of topics is.

Topic models are also referred to as probabilistic [17] topic models, which refers to statistic algorithms for discovering the latent semantic structures of an extensive text body. By using topic-modelling [19] it is possible to learn good interpretable latent structures which are useful for recommendation. Various topic modelling algorithms exists, of which we use Latent Dirichlet Allocation [4][7] (LDA).

1.4 Latent Dirichlet Allocation (LDA)

Each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA [4]. This is identical to probabilistic latent semantic analysis (pLSA), except that in LDA the topic distribution is assumed to have a sparse Dirichlet prior. The sparse Dirichlet priors encode the intuition that documents cover only a small set of topics and that topics use only a small set of words frequently. In practice, this results in a better disambiguation of words and a more precise assignment of documents to topics. LDA is a generalization of the pLSA model, which is equivalent to LDA under a uniform Dirichlet prior distribution.

Here a topic is not strongly defined, neither semantically nor epistemologically. It is identified on the basis of supervised labeling and (manual) pruning on the basis of their likelihood of co-occurrence. A lexical word may occur in several topics with a different probability, however, with a different typical set of neighboring words in each topic.

Each document is assumed to be characterized by a particular set of topics [4]. This is akin to the standard bag of words model assumption, and makes the individual words exchangeable.

1.5 Fuzzy Logic

Fuzzy logic is an approach to computing based on “degrees of truth” rather than the usual “true or false” (1 or 0) Boolean logic on which modern computerized based. It is the form of logic in which the truth values of variables may be any real number between 0 and 1.

Fuzzy logic is usually used to measure the membership value belonging to a class. Using this a variable can be in more than one class with varying degrees of membership. The membership value lies between 0 and 1. The fuzzy logic is used to compute the membership of the cited document to the original document. Using this the network model of the documents is generated. This network structure is more informative than the binary model of the document network structure.

CHAPTER 2

PROBLEM STATEMENT

2.1 Scope

Our model ensures that this method improves the prediction, accuracy with lower empirical training time and stability by upgrading the regular recommendation algorithm using topic modelling and fuzzy logic for grouping of documents respectively determining the amount of relativeness between each pair of documents.

Of the several topic modelling varieties, Latent Dirichlet Allocation [7] model (LDA) is used. We use LDA [10] of the available methods because, the main field of interest is modeling relations between topics. The LDA is used to group documents under a topic. The LDA model is highly modular and can therefore be easily extended.

The fuzzy logic is used to enhance the amount of information regarding the link between two documents. The link is a fuzzy membership value evaluated using a Gaussian membership function. This value is used to evaluate the relativeness between documents. Since the relativeness is measured using fuzzy logic, the recommendation given by the model using this value will be more precise.

Using this model recommendation of documents (papers) to users is done by seamlessly integrating both feedback matrix and item (document) content information into the same model, which can address the problems faced by MF based [26], [36] CF. By combining MF and latent Dirichlet allocation [4] (LDA), the model achieves better prediction performance than MF based CF with better interpretable results. Moreover, with the item content information, RCTR-fuzzy can predict feedback for out-of-matrix items.

2.2 Existing System

Relational Collaborative Topic Regression (RCTR) [1], is a hierarchical Bayesian model. This model integrates the user-item feedback information, item-content information, and network structure among items into the same model. It jointly models the user-item feedback matrix and the item-content information (texts of articles). RCTR seamlessly incorporates topic modeling with CF to improve the performance and interpretability. For new items, RCTR is able to perform out-of-matrix prediction (cold-start prediction), using only the content information. To incorporate item relations for recommendation. The main contributions of RCTR are outlined as follows:

- RCTR seamlessly integrates the user-item feedback information, item content information and relational (network) structure among items into a principled hierarchical Bayesian model.
- Even if a new item has been given feedback only by one or two users, RCTR can make effective use of the information from the item network to alleviate the data sparsity problem.
- In cases where a new user has given feedback to only one or two items, RCTR can also make effective use of the information from the item network to improve the recommendation accuracy.
- In RCTR, a family of link (relation) probability functions is proposed to model the relations between items.
- Number of learning iterations are needed for RCTR to achieve satisfactory accuracy is less. As a consequence, the total empirical measured runtime of training RCTR is lower than that of training CTR.
- RCTR can learn good interpretable latent structures which are useful for recommendation.

Merits

- ➢ It is incorporates uses item-content information.
- ➢ It is able to do cold-start recommendation.
- ➢ It uses network structure of items.

Demerits

- ➢ The feedback is a binary model.
- ➢ More than one graph can be utilized for recommendation.

2.3 Proposed work

We propose a fuzzy based RCTR [1] model. It integrates the user-item feedback information, item-content information, and network structure among items into the same model. It uses topic modeling [4] and fuzzy logic to improve the prediction of the recommender system. The system is a three-step process. First step is Model initialization step, it generates the user vector, item vector, link information and the network structure of items. The next is the learning step, here the model learns the parameters and assigns each user to a topic based on the MAP evaluation that maximizes the log-posteriori and then the prediction is done for the items based on the learned model. The final step is to generate the set of recommended items based on the query given by the user.

2.4 Objectives

- To develop a novel hierarchical Fuzzy based system, called RCTR-FUZZY that uses fuzzy logic to measure the link between items for recommendations.
- To automatically predict the feedback rating of text documents.
- To provide a precise recommendation of text documents to the users.

CHAPTER 3

LITERATURE SURVEY

[1] **M. Balabanovic and Y. Shoham, "Fab: Content-based, collaborative recommendation," Commun. ACM, vol. 40, no. 3, pp. 66–72, 1997.**

In this paper, the content-based and collaborative based method are combined. In this text documents are recommended based on a comparison between their content and a user profile. The collaborative approach to recommendation is very different: Rather than to recommend items because they are similar to items, the user has liked in the past, it recommends items other similar users have liked. It creates a hybrid content-based, collaborative system. It also maintains user profiles based on content analysis, and directly compare these profiles to determine similar users for collaborative recommendation. Using this, users receive items both when they score highly against their own profile, and when they are rated highly by a user with a similar profile.

[2] **R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in Proc. Adv. Neural Inf. Process. Syst., 2007, pp. 1257–1264.**

In this paper, a Probabilistic Matrix Factorization (PMF) model is proposed which scales linearly with the number of observations and performs well on the large, sparse, and very imbalanced Netflix dataset. The PMF model is further extended to include an adaptive prior on the model parameters and finally a constrained version of the PMF model is developed based on the assumption that users who have rated similar sets of movies are likely to have similar preferences. In this model when the predictions of multiple PMF models are linearly combined with the predictions of Restricted Boltzmann Machines models, using this an error rate of 0.8861 that is nearly 7% better than the score of Netflix's own system is achieved.

[3] **D. Almazro, G. Shahatah, L. Albdulkarim, M. Kherees, R. Martinez, and W. Nzoukou, "A survey paper on recommender systems," CoRR, abs/1006.5278, 2010.**

In this paper a hybrid recommendation system is based on content boosted collaborative filtering approach is proposed in order to overcome sparsity and cold start problem item problems of collaborative filtering. The content-based part of the proposed approach exploits semantic similarities between items based on a priori defined ontology-based metadata in movie domain and derived feature-weights from content-based user models. Using this semantic similarity between items and collaborative based user models, recommendations are generated.

[4] **D. Agarwal and B.-C. Chen, "fLDA: Matrix factorization through latent dirichlet allocation," in Proc. 3rd Int. Conf. Web Search Data Mining, 2010, pp. 91–100.**

In this paper, a recommendation system using Matrix factorization through LDA is proposed. In this both user and item factors are regularized simultaneously through user features and the bag of words associated with each item. Specifically, each word in an item is associated with a discrete latent factor often referred to as the topic of the word. Then, user rating on an item is modeled as user's affinity to the item's topic where user affinity to topics (user factors) and topic assignments to words in items (item factors) are learned jointly in a supervised fashion. To avoid over fitting, user and item factors are regularized through Gaussian linear regression and Latent Dirichlet Allocation (LDA) priors respectively. fLDA is also able to identify interesting topics that explains user-item interactions. And performed recommendation on to users.

[5] **C. Wang and D. M. Blei, “Collaborative topic modeling for recommending scientific articles,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2011, pp. 448–456.**

In this paper, a topic modeling algorithm for recommendation is proposed for Researchers that have access to large online archives of scientific articles. As a consequence, finding relevant papers has become more difficult. Newly formed online communities of researchers sharing citations provides a new way to solve this problem. In this paper, we develop an algorithm to recommend scientific articles to users of an online community. Our approach combines the merits of traditional collaborative filtering and probabilistic topic modeling. It provides an interpretable latent structure for users and items, and can form recommendations about both existing and newly published articles.

[6] **W.-J. Li and D.-Y. Yeung, “Social relations model for collaborative filtering,” in Proc. AAAI Conf. Artif. Intell., 2011, p. 1.**

In this paper a probabilistic model for collaborative filtering (CF), called SRMCoFi is proposed. In this model both linear and bilinear random effects are integrated into a principled framework. The formulation of SRMCoFi is supported by both social psychological experiments and statistical theories. By this model is able to perform recommendation to users. The afore mentioned papers use various methods for recommendation. All these methods come under either collaborative based or content based methods. These suffer from sparse matrix problems of Feedback Matrix. It is sparse mice all items are not rated by each user. Hence sparsity develops. These models use user-profile, item profile & feedback of users. They fail to include the miscellaneous item information & the link between the items. The old models &the link between the items. The old models use a binary feedback matrix.

[7] **Y. Cai, H. fung Leung, Q. Li, H. Min, J. Tang, and J. Li, "Typicality-based collaborative filtering recommendation," IEEE Trans. Knowl. Data Eng., vol. 26, no. 3, pp. 766–779, Mar. 2014.**

In this paper, CF is classified into user-based CF and item-based CF. Here the user-based CF approach is used to find out the set of users who have similar interest patterns to a given user and recommend to the user those items that other users in the same set like, while the item-based CF approach aims to provide a user with the recommendation on an item based on the other items with high correlations.

Consolidated Analysis

The afore mentioned papers use various methods for recommendation. All these methods either collaborative based or content based methods, these suffer from sparse matrix problems of feedback matrix. It is sparse since all items are not rated by users, hence scarcity develops. These model use user profile, item profile and feedback of users. They fail to include the miscellaneous item information and link between the items. The old model use a binary feedback matrix because of which the interest of a user to a liked item cannot be measured, and hence the predictions are not precise.

CHAPTER 4

REQUIREMENT ANALYSIS

Requirements are the basic needs or constraints which are required to develop a system. Requirements are broadly classified as user requirements and the system requirements. These requirements can be collected while designing the system.

The two major classifications of requirements are software and the hardware requirements.

1. Functional Requirements.
2. Non-Functional Requirements.

4.1 FUNCTIONAL REQUIREMENTS

Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified.

4.1.1 HARDWARE REQUIREMENT

- Processor : Intel i5 Processor
- Memory : 2GB or above
- Hard disk : 100GB or above

4.1.2 SOFTWARE REQUIREMENTS

- Framework : .Net Framework 4.5
- Programming Tools: Microsoft Visual Studio 2012
- Operating System : Windows 8 or above

4.2 NON-FUNCTIONAL REQUIREMENTS

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e how fast the system reacts to user commands) or accuracy (i.e. how precise are the systems numerical answers).

- **Scalability**
  - By using distributed learning algorithm for RCTR-fuzzy model, which would make RCTR-fuzzy scalable for big data modeling.
- **Adaptability**
  - Our RCTR-fuzzy model can adapt for CTR-SMF setting with social networks among users.
- **Performance Characteristics**
  - Speed, throughput and execution time depends on the size of the input dataset.
  - Improved performance by efficiently integrating the item relations into the modeling.
- **Flexibility**
  - The Fuzzy Rule formulation of RCTR-Fuzzy is flexible to model more than one item network.

CHAPTER 5

DESIGN

5.1 SYSTEM ARCHITECTURE

The system architecture explains the overall working principle of the derived algorithm. The input for the algorithm is user profile information and the item information. The output is the recommended set of documents.

This is a three-step process. First one is Model initialization step, it generates the user vector, item vector, link information and the network structure of items [1]. The next is the learning step, here the model learns assigns each user to a topic [7], [24] based on the MAP evaluation that maximizes the log-posteriori and then the prediction is done for the items based on the learned model. The final step is to evaluate and generate the set of recommended items. The window returns a list of documents based on query given by the user and comparing it with the list of recommended articles.

5.1.1 Registration

The user (author) enters the profile information in the registration form. This information is stored in the database. Later it is used as information for constructing the user vector, to understand the user's domain and his interests. The registered user can either publish a new paper entering the metadata or can search for documents.

5.1.2 Publish

The Registered user publishes the new article. The article Meta information is updated to the database. The metadata consists of details about the published article.

5.1.3 User information

This is created to understand the user domain and his preference [7]. This is done by analyzing the papers published by the author. By this, User vector for all registered users is created. The vector is formed by constructing the vector space model derived from the user's published papers. And then a normal distribution [1] is applied to the user vector to get a precise understanding of the user.

5.1.4 Set topic proportions

The LDA algorithm requires a pre-initialization step. In this the dirichlet prior is set. The dirichlet prior is set to understand the variation in topic distribution. Using this the topics are populated evenly. In this step, the number of topics for distribution in the LDA algorithm is set [27], [1].

5.1.5 Construct item vector

A vector is generated for each item. This vector is formulated from the item-content information. The vector describes the item information (here the item is a document). The document vector is a mathematical representation of the document [7]. The vector is distributed using Multivariate normal distribution [27], [24]and an offset value is calculated to correct the item vector that incurs errors while generating the document vector as input to the LDA algorithm which categorizes the given documents into the set number of topics based on the co-occurrences of the words in the documents.

5.1.6 Binary link indicator

This model evaluates the link between a pair of documents given the item vector [27]. This link is evaluated using a similarity measures of documents, given the random variables. This link value is used to find the relation among the items.

The value lies between 0 and 1. When the value is close to 0, the link value is very low which means the document are different. when the value is close to 1, the link value is very high which indicates that the documents are highly similar [24].

5.1.7 Feedback matrix

This matrix is generated by referring the citations of each paper. The citation of a paper means that the paper is cited by the user in his paper [12]. The citation is used to generate a feedback matrix. In this matrix, the Number of citations for a paper is noted and Fuzzy logic is applied. This is used to measure the membership value of each document to the original document. For this calculation, a Gaussian function is used. This function returns a value between 0 and 1, which denotes the cited documents membership to the original document.

5.1.8 Generate learning parameter

This is used to learn the topic to which the user belongs to. The learning parameter is the MAP estimate of the random variables [24] , [27]that determine the user's affinity to each topic. The learning parameter is calculated for a pain of documents given a topic and user. The value represents the affinity of the user to each topic after which a MAP estimates is taken to find the topic of interest for the given user. Hence by using this all user's interest the documents published.

5.1.9 Rating prediction

The prediction is the possible rating that would be given by the user to the document. This prediction is done for all documents [12]. This prediction is done for all documents including those documents that previously have not been rated. For new items, out-matrix prediction algorithm is used, for already rated items in-matrix algorithm is used [27].

5.1.10 Query vector

This is the user entered query that has been modified to a vector format. This format is a mathematical representation of the query. This conversion is used to understand the query.

5.1.11 List of documents

It is generated by getting the top three topics the user is interested in. Then the documents belonging to that set of topics is retrieved and order in descending order of the predicted rating.

5.1.12 Similarity measure calculation

The similarity between the query and each document in the list of recommended documents is found. The documents with high similarity value is considered to be more related to query, hence the documents are ordered in descending order of similarity.

5.3 UML DIAGRAMS

Unified Modeling Language (UML) is a standard visual specification language for modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model.

5.3.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a use analysis. Its purpose is prevent a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases) and any dependencies between those use cases.

The main purpose of a use case diagram is to show that system functions are performed from actor. Roles of the actors in the system can be depicted in the fig 5.3.1.

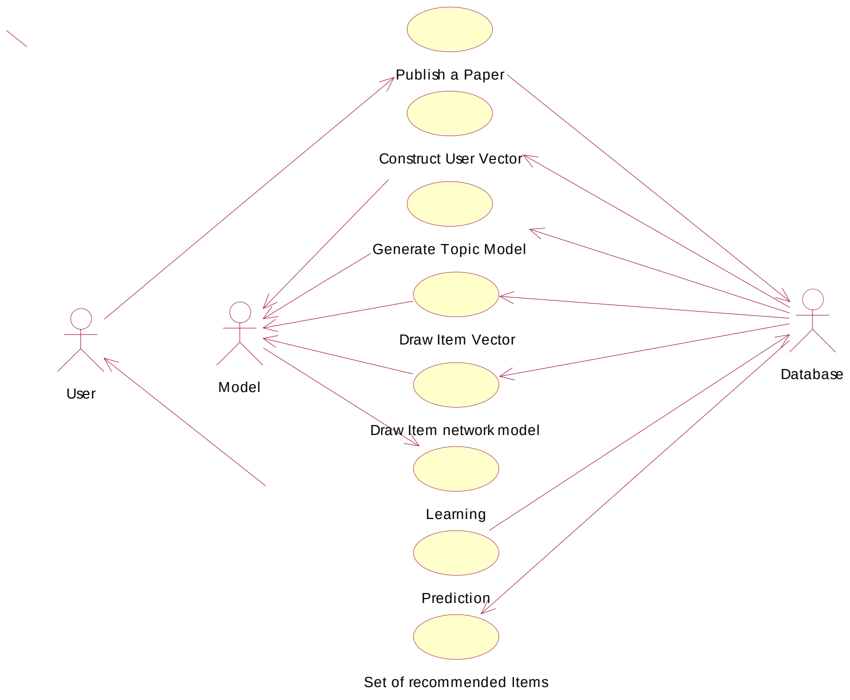


Fig 5.2 Use Case Diagram

5.3.2 CLASS DIAGRAM

A class diagram in the UML is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes and relationship between the classes. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view marked information. Protected visibility allows child classes to access information they inherited from a parent class.

In software engineering, a class diagram in the unified modeling language (UML) is a type of statistic structure diagram that describes the structure of a system's classes, their attributes, operation (or methods) and the relationship among the classes. A class diagram is an illustration of the relationships and source code dependencies among classes in the unified modeling language (UML) is depicted in the fig 5.3.2.

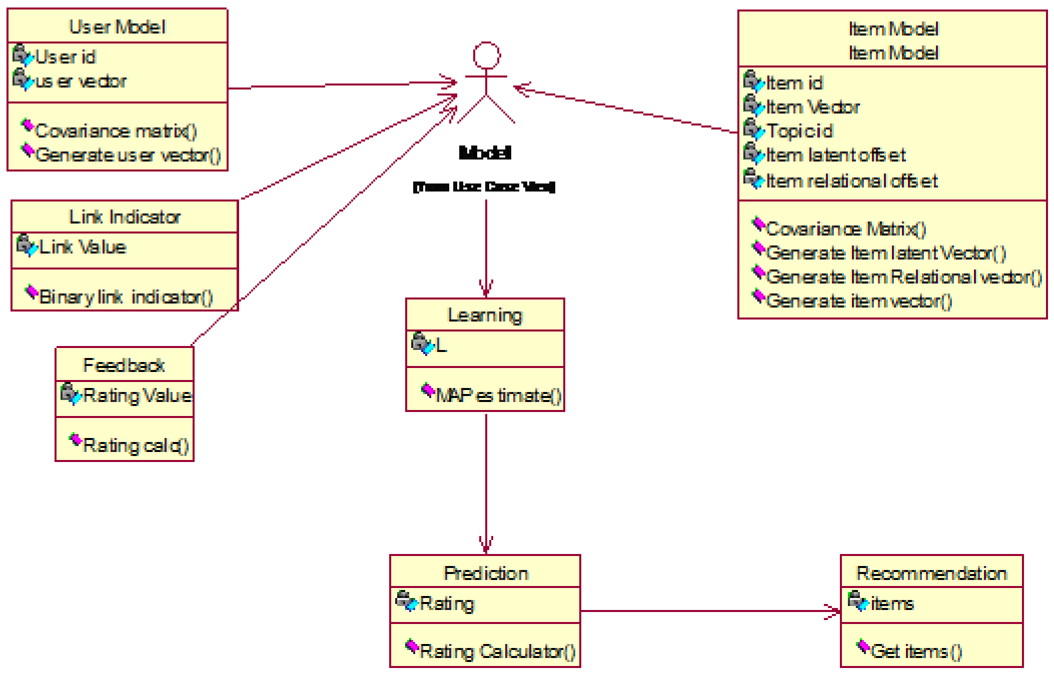


Fig 5.3 Class Diagram

5.3.3 SEQUENCE DIAGRAM

Sequence diagrams are type of interaction diagrams. This describes interactions among classes. These interactions are modelled as exchanged of messages. These diagrams focus on classes and the messages they exchange messages. These diagrams focus on classes and the messages they exchange to accomplish some desired behavior. The eminent sequence of procedure explained in the provenance light weight scheme is given below in the Fig. 5.3.3.

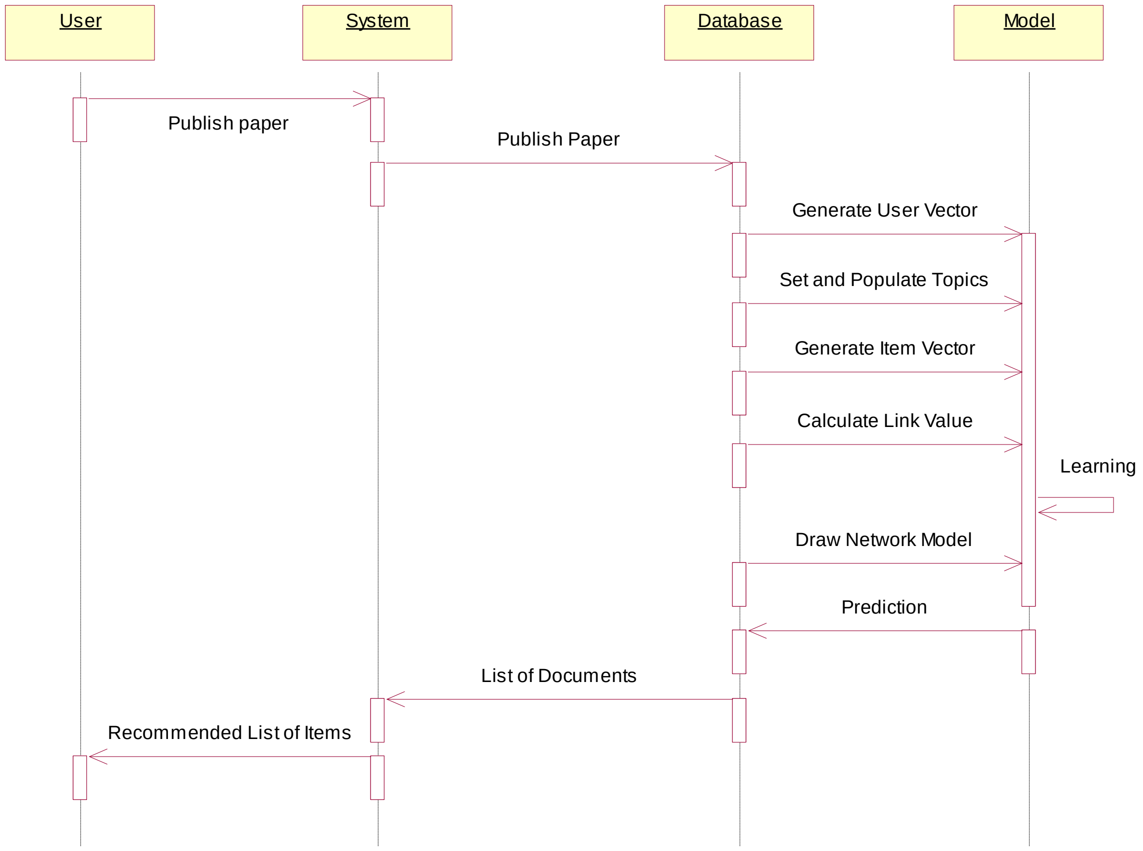


Figure 5.4 Sequence Diagram

5.3.4 COLLABORATION DIAGRAM

A collaboration diagram show the object and relationships involved in an interaction and the sequences of messages exchanges among the objects during the interaction. The collaboration diagram can be decomposition of a classes, class diagram or part of a class diagram show message being sent between classes objects is depicted in the fig 5.3.4.

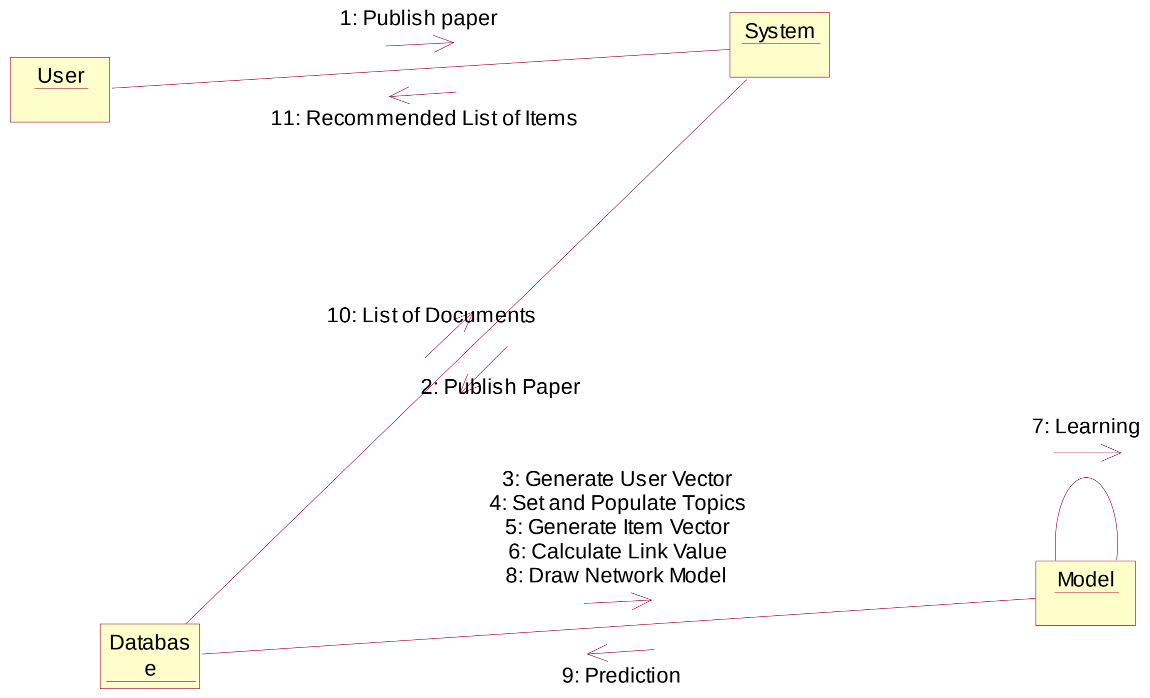


Fig 5.5 Collaboration Diagram

5.4 DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the ‘flow’ of a data through an information system. It differs from the flowchart as it shows the data flow instead of the control flow of the program. A data flow diagram can also be used for the visualization of data processing. The DFD is designed to show how a system is divided onto smaller portions and to highlight the flow of data between those parts.

Data Flow Diagrams (DFD) is an important technique for modeling a system’s high level detail by showing how input data is transformed to output results through a sequence of functional transformations. DFD consists of four major components: entities, processes, data stores and data flow.

5.4.1 DFD-Level 0

In this diagram, the user vector, item vector and the network model of the documents is generated. And the RCTR-fuzzy model is formed.

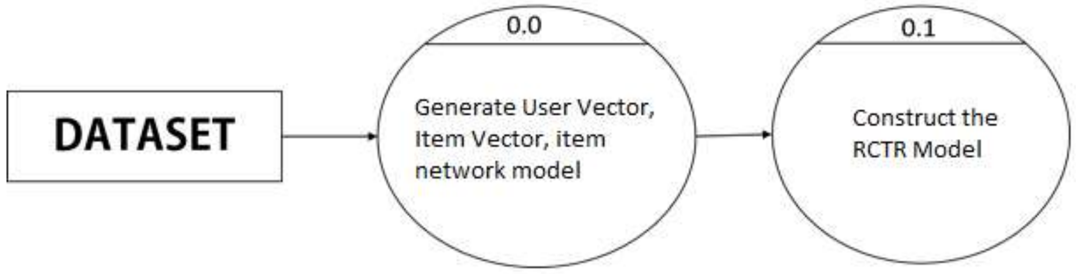


Fig 5.6 ***DFD-Level 0***

5.4.2 DFD-Level 1

In this diagram, the RCTR model does learning and assigns user to a specific topic. And the prediction for new documents is done.

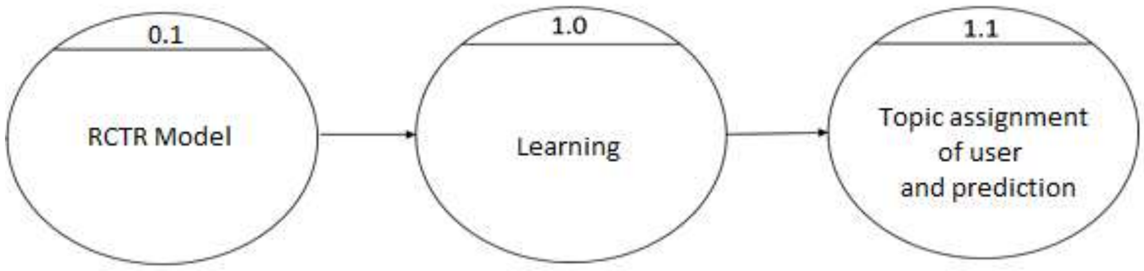


Fig 5.7 ***DFD-Level 1***

5.4.3 DFD-Level 2

In this diagram, the User query is matched with the set of recommended documents to generate the result.

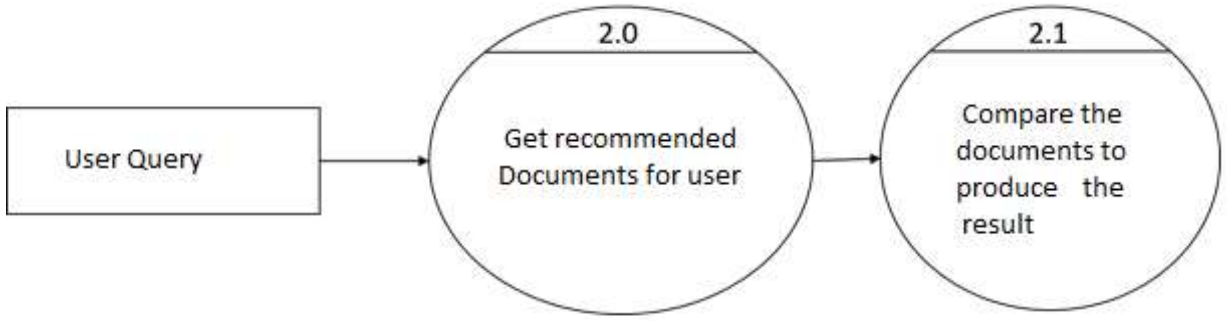

User Query
2.0
Get recommended Documents for user
2.1
Compare the documents to produce the result

Fig 5.8 ***DFD-Level 2***

CHAPTER 6
IMPLEMENTATION

6.1 Model Formulation

6.1.1 Generate User Latent Vector

The user vector is generated for all the users using a multivariate normal distribution available for normalizing a vector. This is done to understand the domain of the users and to understand the user's affinity to various class of documents.

The multivariate normal distribution is given by,

$$N(\mu, \Sigma) = \exp(\frac{-1}{2}(x-\mu)^T \sum^{-1}(x-\mu))/\sqrt{|2\pi\sum|} \quad (6.1)$$

Where μ, is the mean of the distribution?

Σ, is the covariance matrix.

The covariance matrix is,

$$\sum = \begin{bmatrix} E[(X_1-\mu_1)(X_1-\mu_1)] & E[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & E[(X_1-\mu_1)(X_n-\mu_n)] \\ E[(X_2-\mu_2)(X_1-\mu_1)] & E[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & E[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n-\mu_n)(X_1-\mu_1)] & E[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & E[(X_n-\mu_n)(X_1-\mu_1)] \end{bmatrix} \quad (6.2)$$

We adapt the distribution to calculate the latent user vector by using as,

$$u_i \sim N(0, \lambda_u^{-1} I_k) \quad (6.3)$$

Where λ_u, is user regularization parameter

I_k, is the k dimensional identity matrix.

6.1.2 For all documents,

a) Draw topic proportions $\theta_j \sim Dirichlet(\alpha)$

The dirichlet prior is set for each document. This prior calculates the probability of the document to be assigned in each topic.

$$Dir(\alpha) = \frac{\prod_{k=1}^{d} \Gamma(\alpha_i)}{\Gamma \sum_{i=1}^{k} a_i} \qquad (6.4)$$

b) Draw the item latent offset

The item latent offset vector is generated for all documents using a multivariate normal distribution available for normalizing a vector. This is done to understand the description of the document. Using this the document can be represented mathematically using a vector.

We adapt the distribution to calculate the latent user vector by using as,

$$\epsilon_j \sim N(0, \lambda_v^{-1} I_k) \qquad (6.5)$$

Where λ_v, is item regularization parameter

$I_{k,}$ is the k dimensional identity matrix

The normal distribution is given by

$$N(\mu, \Sigma) = \exp(\frac{-1}{2}(x-\mu)^T \sum^{-1}(x-\mu))/\sqrt{|2\pi\sum|} \qquad (6.6)$$

Where μ, is the mean of the distribution?

Σ, is the covariance matrix.

The covariance matrix is,

$$\sum = \begin{bmatrix} E[(X_1-\mu_1)(X_1-\mu_1)] & E[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & E[(X_1-\mu_1)(X_n-\mu_n)] \\ E[(X_2-\mu_2)(X_1-\mu_1)] & E[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & E[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n-\mu_n)(X_1-\mu_1)] & E[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & E[(X_n-\mu_n)(X_1-\mu_1)] \end{bmatrix} \qquad (6.7)$$

Now calculate item latent vector to be: $v_j = \epsilon_j + \theta_j$ (6.8)

c) Construct the fuzzified relational matrix

For network matrix construction, the citation of each paper is used. The paper is cited in the published paper is considered to be linked to that paper. In the matrix the number of citations is recorded as the value. If there is no citation, then the value is set to 0.

Fuzzy is applied have to calculate the membership of the document that is linked. This membership value is calculated using Gaussian function. This function gives the membership value between 0 to 1

Gaussian function is,

$$\boldsymbol{G(x) = ae^{-(x-b)^2/c^2}} \qquad (6.9)$$

Where

a is the height of the curve's peak,

b is the position of the center of the peak and

c controls the width of the "bell".

We modify this equation to our requirement. Using this function, we find the measure of membership of the cited document. We set the maximum membership value to 1. For the maximum limit to be 1, we keep “a” as 1. The maximum number of citations for a particular document would be not more than 20 and since the center of the function determines the maximum membership value of the cited document. We set b as 20. The width of the bell function defines the number of documents that are most significant. Since this number is less than 10, we set the c value to be 10.

d) Draw item relational offset

Relational offset is calculated using the link between the items. The link is then calculated as a distribution of documents. This distribution helps understand the link between the documents. This link refers the ideology and model of one document to another document. By doing this the document can be understood as a combination of documents.

This combination can be understood as a distribution of documents, using the normal distribution. We adapt the distribution to calculate the latent user vector by using Eqn-2 as the function

$$\boldsymbol{\tau_i} \sim \boldsymbol{N}(\mathbf{0}, \boldsymbol{\lambda_r^{-1} I_k}) \qquad (6.10)$$

Where $\lambda_{r,}$ is link regularization parameter

$I_{k,}$ is the k dimensional identity matrix

The multivariate normal distribution is given by,

$$N(\mu, \Sigma) = \exp(\frac{-1}{2}(x-\mu)^T \sum^{-1}(x-\mu))/\sqrt{|2\pi\sum|} \qquad (6.11)$$

Where μ, is the mean of the distribution?

Σ, is the covariance matrix.

The covariance matrix is,

$$\sum = \begin{bmatrix} E[(X_1-\mu_1)(X_1-\mu_1)] & E[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & E[(X_1-\mu_1)(X_n-\mu_n)] \\ E[(X_2-\mu_2)(X_1-\mu_1)] & E[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & E[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n-\mu_n)(X_1-\mu_1)] & E[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & E[(X_n-\mu_n)(X_1-\mu_1)] \end{bmatrix} \qquad (6.12)$$

Now set the item relational vector to be,

$$\boldsymbol{S_j} \sim \boldsymbol{\tau_j} + \boldsymbol{v_j} \qquad (6.13)$$

e) For each word in the document (item) w$_j$, which is denoted as W$_{jn}$

i. Draw topic assignment:$z_{jn} = mult(\theta_j)$ (6.14)

Topic assignment is done randomly for all the words in the document. After which a multinomial distribution is applied so as to calculate the probability value of each word in the document going to be assigned to a topic. This distribution calculates the probability value of each word to a specific topic. After which the word is mapped to the topic of maximum topic probability. This assignment is repeated several times to get correct matching of the words to the topic.

In probability theory, the multinomial distribution is a generalization of the binomial distribution. It models the probability of counts for rolling a k-sided die n times. For “n” independent trials each of which leads to a success for exactly one of k categories, with each category having a given fixed success probability, the multinomial distribution gives the probability of any particular combination of numbers of successes for the various categories.

When n is 1 and k is 2 the multinomial distribution is the Bernoulli distribution. When k is 2 and number of trials are more than 1 it is the binomial distribution. When n is 1 it is the categorical distribution.

The multinomial function is given by,

$$p\left(\frac{x}{\theta}\right) = \frac{n!}{\prod_{i=1}^{d} w_i!} \prod_{i=1}^{d} \theta_i^{w_i} \qquad (6.15)$$

Where, w_i : indicates word “i” of the vocabulary observed

$$w_i = \begin{cases} if\ word\ "i"\ is\ observed, & = 1 \\ otherwise, = 0 \end{cases}$$

$\boldsymbol{\theta_i}$ = probability that the word “i” is seen.

ii. Document assignment, Draw word: $w_{jn} = mult(\beta_{z_{jn}})$ (6.16)

The documents are now assigned to a specific topic. This assignment is calculated using the topic assignment of each word. This assignment is done by calculating the multinomial distribution. This calculates the probability of the word getting allocated to every topic. The topic with the maximum topic probability is assigned as the topic for the specific document.

The multinomial function is given by,

$$p\left(\frac{x}{\theta}\right) = \frac{n!}{\prod_{i=1}^{d} w_i!} \prod_{i=1}^{d} \theta_i^{w_i} \qquad (6.17)$$

Where, w$_i$: indicates word “i” of the vocabulary observed

$$w_i = \begin{cases} if\ word\ "i"\ is\ observed, & = 1 \\ otherwise, = 0 \end{cases}$$

$\theta_i =$ probability that the word “i” is seen.

6.1.3 Draw the parameter:

This is an adjusting parameter. This is used to reduce the noise value that is accumulated while finding the latent vectors. It is calculated using the formula.

$$\eta^{+} \sim N(0, \lambda_c^{-1} I_{k+1}) \qquad (6.18)$$

The multivariate normal distribution is given by,

$$N(\mu, \Sigma) = \exp\left(\frac{-1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)\right) / \sqrt{|2\pi\Sigma|} \qquad (6.19)$$

Where μ, is the mean of the distribution?

Σ, is the covariance matrix.

The covariance matrix is,

$$\Sigma = \begin{bmatrix} E[(X_1-\mu_1)(X_1-\mu_1)] & E[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & E[(X_1-\mu_1)(X_n-\mu_n)] \\ E[(X_2-\mu_2)(X_1-\mu_1)] & E[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & E[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n-\mu_n)(X_1-\mu_1)] & E[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & E[(X_n-\mu_n)(X_1-\mu_1)] \end{bmatrix} \qquad (6.20)$$

6.1.4 Draw a binary link indicator- For each pair of items (j, j`)

This indicator is used to calculate the link between a pair of documents. For this the document content is used. The comparison is done between a pair of item relational vector. This comparison gives the relativeness value of two documents.

It is calculated using the psi function:

$$I_{j,j'|s_j,s_{j'}} \sim \psi(.|s_j, s_{j'}, \eta^+) \qquad (6.21)$$

Which can be computed using,

$$\psi(.|s_j, s_{j'}, \eta^+) = [\sigma(\eta^+(s_j . s_{j'}) + v)]^\rho \qquad (6.22)$$

6.1.5 Draw the feedback- r_{ij} for each user item pair (i,j),

The feedback is re-computed using a normal distribution. It uses the previous rating value. A multivariate normal distribution is applied to understand the rating of the user to the set of items. It is calculated by,

$$r_{ij} = N(u_i^T v_j, c_{ij}^{-1}) \qquad (6.23)$$

The multivariate normal distribution is given by,

$$N(\mu, \Sigma) = \exp(\frac{-1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)) / \sqrt{|2\pi\Sigma|} \qquad (6.24)$$

Where μ, is the mean of the distribution?

Σ, is the covariance matrix.

The covariance matrix is,

$$\Sigma = \begin{bmatrix} E[(X_1-\mu_1)(X_1-\mu_1)] & E[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & E[(X_1-\mu_1)(X_n-\mu_n)] \\ E[(X_2-\mu_2)(X_1-\mu_1)] & E[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & E[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n-\mu_n)(X_1-\mu_1)] & E[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & E[(X_n-\mu_n)(X_1-\mu_1)] \end{bmatrix} \qquad (6.25)$$

6.2 Model Learning

In this model, all the parameters are treated as random variables. We focus on the importance of the relations among items. Hence, we adopt a maximum a-posteriori estimate for parameter learning in RCTR. The MAP estimate tries to maximize the log-posteriori of the random variables. We use a variation of the RCTR learning parameter. For the learning parameter, we modify the method of calculation of the item latent vector. The proposed learning model is learned using the following equation.

$$\mathcal{L}_j = \delta_j - \gamma_j \tag{6.26}$$

Where δ_j is, the parameters adopted from the RCTR model and γ_j is the parameter included using the fuzzified relational network structure.

δ_j is calculated using,

$$\delta_j = \rho \sum_{i_{j,j'=1}} \log \sigma\left(\eta^{+}\left(s_j . s_{j'}\right) + v\right) - \frac{\lambda_u}{2}\sum_i u_i^T u_i - \frac{\lambda_v}{2}\sum_j \left(v_j - \theta_j\right)^T \left(v_j - \theta_j\right) - \frac{\lambda_r}{2}\eta^{+^T}\eta^{+} - \sum_j \sum_n \log(\theta_{jk}\beta_{k,w_{jn}}) - \sum_{i,j}\frac{c_{ij}}{2}\left(r_{ij} - u_i^T v_j\right)^2 \tag{6.27}$$

Where, s_j is the item relational vector, ρ is a parameter for the link probability function, $\sigma(x)$ is the sigmoid function, v is set to unit vector, u_i is the user latent vector, $\theta_j$is the topic proportions, $\beta_{k,w_{jn}}$ is vocabulary of words in document “j”, r_{ij} is the rating of item “j” by user “i”, c_{ij} is a binary value denoting the relation, λ_r , λ_u and λ_v are parameters that are set to 0.01 and 100 respectively.

γ_j can be calculated using the modified network model of the documents. This model is modified using the Gaussian function which gives a probabilistic-numerical meaning to the binary network model of the items.

$$\gamma_j = \frac{\lambda_r}{2}\sum_j (N(0, \lambda_r^{-1} I_k), G(x))^T N(0, \lambda_r^{-1} I_k), G(x) \tag{6.28}$$

Where $(N(0,\lambda_r^{-1}I_k),G(x))$, represents the multivariate normal distribution of the fuzzified values from the Gaussian function,

$\lambda_{r,}$ is link regularization parameter

$I_{k,}$ is the k dimensional identity matrix

The multivariate normal distribution is given by,

$$N(\mu,\Sigma) = \exp(\frac{-1}{2}(x-\mu)^T\sum^{-1}(x-\mu))/\sqrt{|2\pi\sum|} \quad (6.29)$$

Where μ, is the mean of the distribution?

Σ, is the covariance matrix.

The covariance matrix is,

$$\sum = \begin{bmatrix} E[(X_1-\mu_1)(X_1-\mu_1)] & E[(X_1-\mu_1)(X_2-\mu_2)] & \cdots & E[(X_1-\mu_1)(X_n-\mu_n)] \\ E[(X_2-\mu_2)(X_1-\mu_1)] & E[(X_2-\mu_2)(X_2-\mu_2)] & \cdots & E[(X_2-\mu_2)(X_n-\mu_n)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(X_n-\mu_n)(X_1-\mu_1)] & E[(X_n-\mu_n)(X_2-\mu_2)] & \cdots & E[(X_n-\mu_n)(X_1-\mu_1)] \end{bmatrix} \quad (6.30)$$

And the Gaussian function is given by,

$$\boldsymbol{G(x) = ae^{-(x-b)^2/c^2}} \quad (6.31)$$

Where we set *a* as 1, *b* as 20 and *c* as 10.

6.3 Prediction

We use a point estimate of u_i, u_j and ϵ_j to calculate the predicted feedback. We use different formula for in-matrix and out-matrix prediction. Since the out- matrix prediction uses the already rated items information, we need to consider pre-rated items also.

In-matrix calculation is performed for matrix where atleast one rating is recorded for a user-item pair. For these items the previously recorded rating is also taken into account for prediction of ratings by the user.

The formula for in-matrix calculation is:

$$r_{ij} = \left(u_j^*\right)^T \left(\theta_j^* + \epsilon_j^*\right) = \left(u_j^*\right)^T v_j^* \tag{6.32}$$

Out-matrix calculation is performed for matrix where no rating is recorded for a user-item pair. These are new items or items that have not been rated by any users. For these items the previously recorded rating is also taken into account for prediction of ratings by the user.

The formula for out-matrix calculation is:

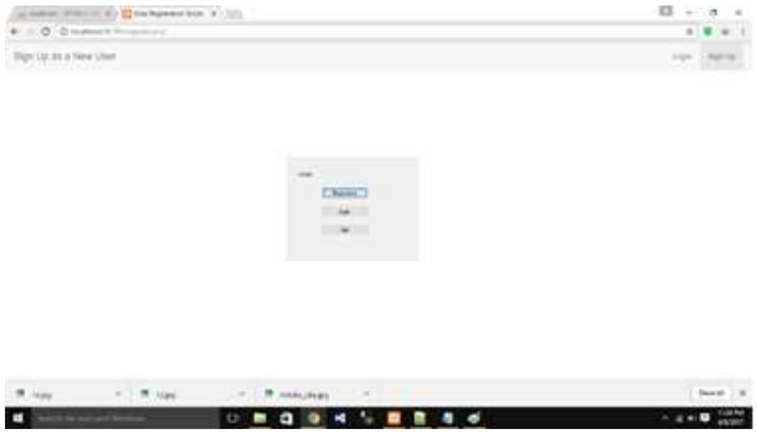
$$r_{ij} = \left(u_j^*\right)^T \theta_j^* \tag{6.33}$$

CHAPTER 7

TESTING

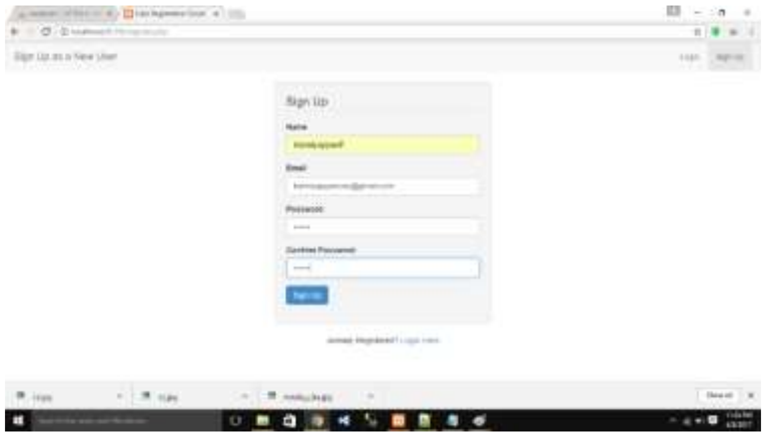
7.1 Home page

This is the home page. The user can either create a new account via the register action or can login to his account.



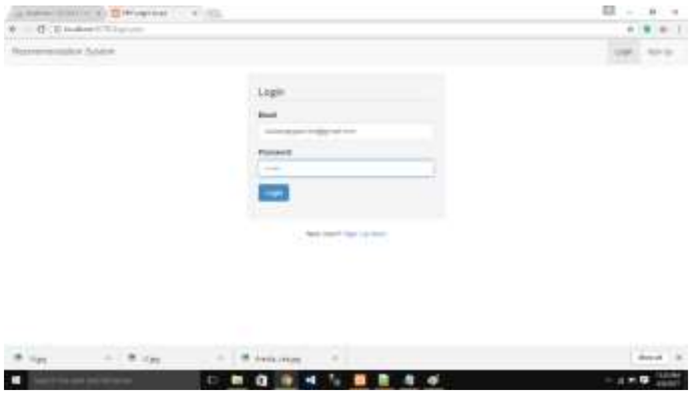
7.2 Registration

The users when click on the register option, this shows up. This opens up a form where the user is required to fill his basic details



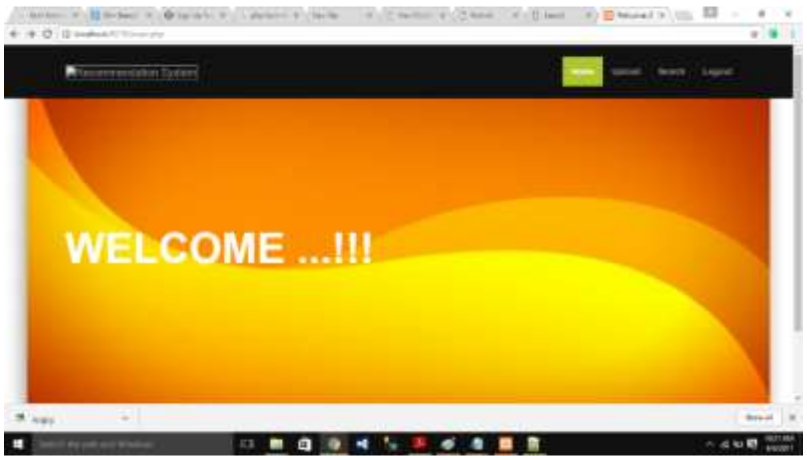
7.3 Login

In case of a returning user, the user can login into his account using his credentials.



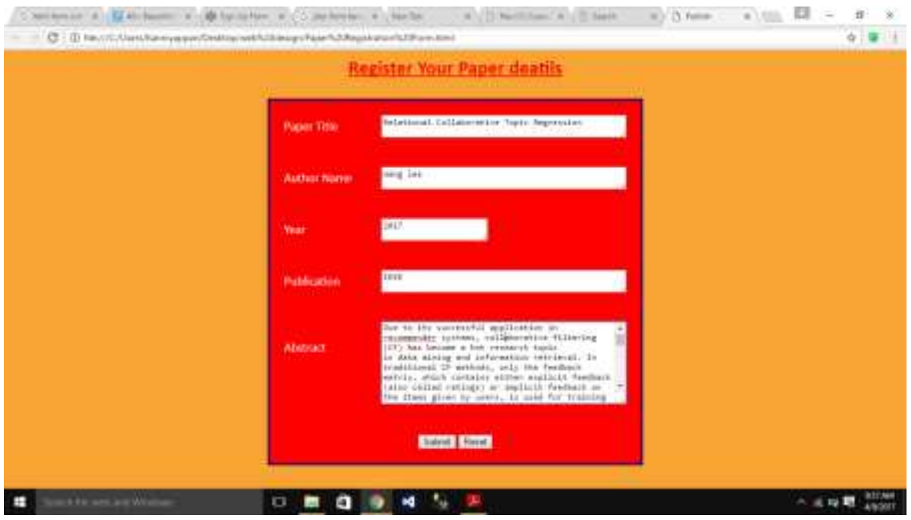
7.4 Account

The home of the user looks like this. The user can either upload a new paper or search for a document



7.5 Publish

In case of a publishing a new paper, the user has to enter the meta data of the paper to be published.

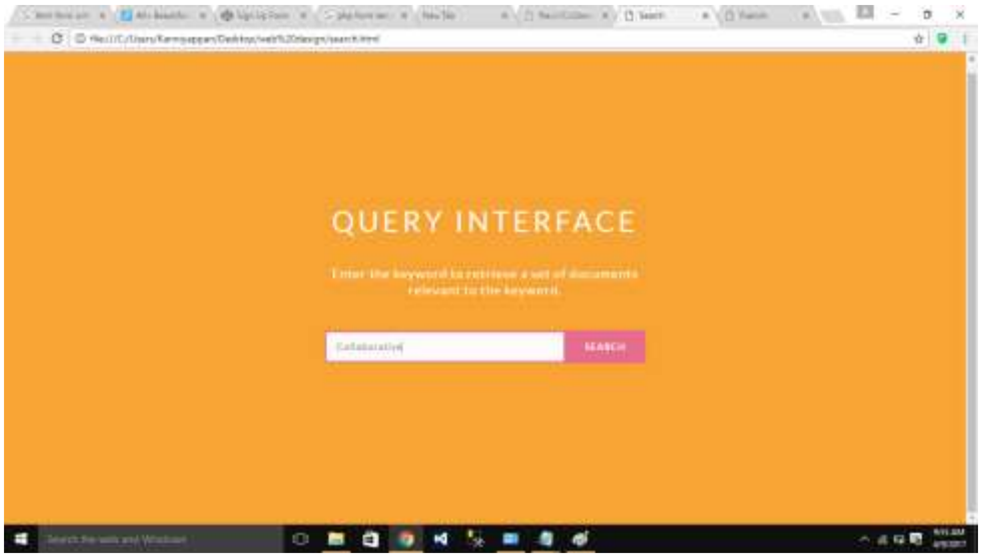


Register Your Paper deatils

Paper Title	Relational Collaborative Topic Regression
Author Name	wang lei
Year	2017
Publication	IEEE
Abstract	Due to its successful application in recommender systems, collaborative filtering (CF) has become a hot research topic in data mining and information retrieval. In traditional CF methods, only the feedback matrix, which contains either explicit feedback (also called ratings) or implicit feedback on the items given by users, is used for training

7.6 SEARCH

Here the user panel is provided, the user is able to give query and documents are retrieved based on the input query.



QUERY INTERFACE

Enter the keyword to retrieve a set of documents relevant to the keyword.

Collaborative

CHAPTER 8
RESULTS AND DISCUSSIONS

The proposed framework is tested for various high dimensional datasets. In this the most relevant feature for a specific data is selected by using the various feature selection algorithms in addition to Booster. Booster split the dataset into number of partitions and the feature selection is performed on each partition then combine the features. The number of features selected before and after applying the booster is varies. The performance improvement is analyzed by using Q-Statistic value.

PERFORMANCE ANALYSIS

The Precision, Recall and F-measure value are evaluated for recommendation system. Higher metric values provide higher the prediction accuracy of documents fort the given user. By using this the performance is significantly improved. The metrics that are used for performance analysis are evaluated and are discussed in the combining section.

8.1 Retrieval

Table.1 Retrieved documents vs. Total number of documents

Total no. of documents	No. of Documents		
		RCTR	RCTR-fuzzy
30	13	16	20
40	15	19	22
50	18	20	27
60	19	24	29

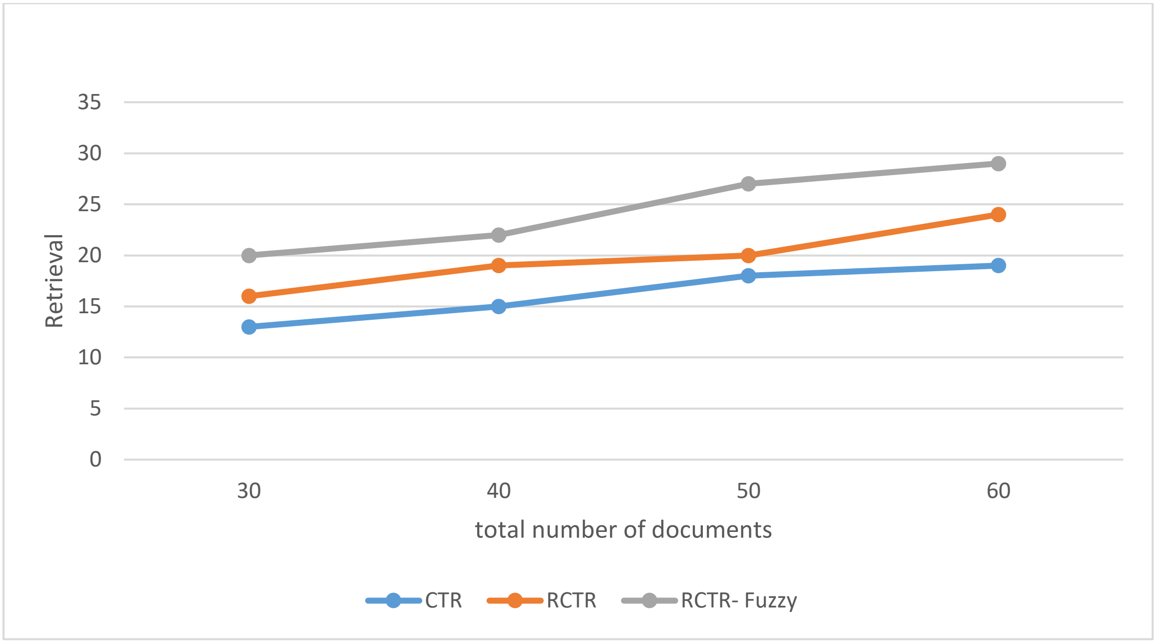


Fig 8.1 Retrieved documents

Retrieval is calculated using the formula,

$$\textbf{Retrieval} = \frac{\textbf{retrieved documents}}{\textbf{total number of documents}} \qquad \textbf{(8.1)}$$

This graph describes the retrieval for given a set of documents. It is a monotonically increasing graph. The retrieval of the RCTR-Fuzzy is more than other models, it is because of the inclusion of the fuzzy logic to evaluate the membership value of the cited document to the main document.

8.2 Precision

Table.2 Precision vs. Total number of documents

Total no. of documents	No. of Documents		
	CTR	RCTR	RCTR-fuzzy
30	0.41	0.45	0.52
40	0.52	0.61	0.69
50	0.55	0.64	0.78
60	0.58	0.69	0.83

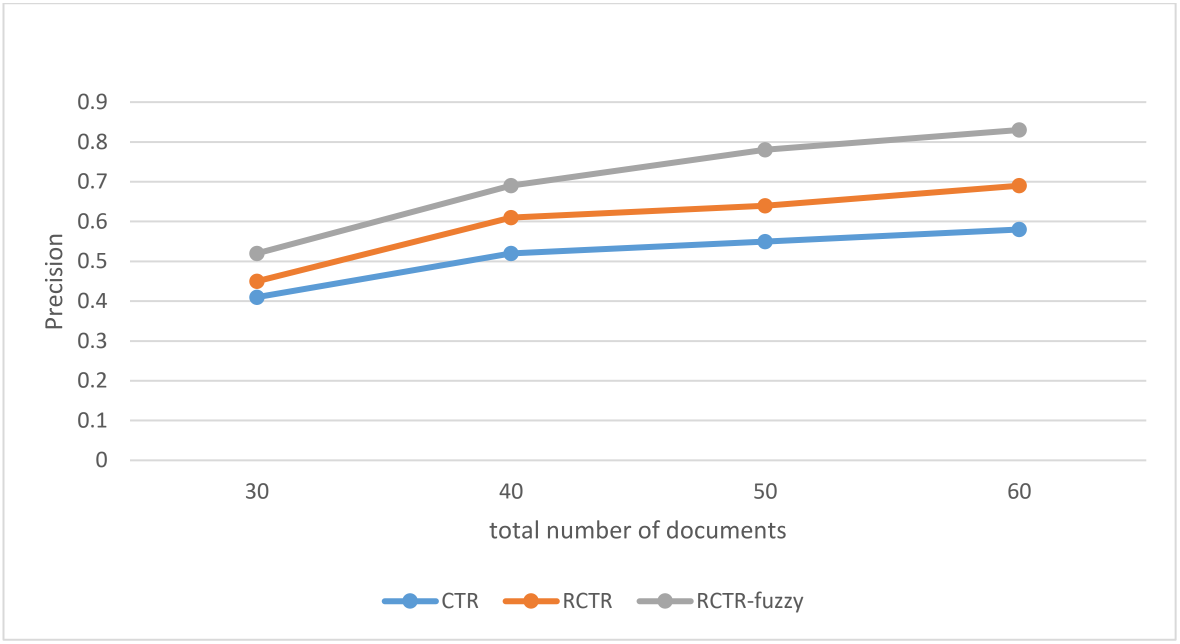


Fig 8.2 Precision

Retrieval is calculated using the formula,

$$\textbf{Precision} = \frac{\textbf{tp}}{\textbf{tp + fp}} \qquad \textbf{(8.2)}$$

This graph describes the retrieval for given a set of documents. It is a monotonically increasing graph. The retrieval of the RCTR-Fuzzy is more than other models, it is because of the inclusion of the fuzzy logic to evaluate the membership value of the cited document to the main document.

CASES

TN: True Negative

Case was negative and prediction was negative

TP: True Positive

Case was positive and prediction was positive

FN: False Negative

Case was positive and prediction was negative

FP: False Positive

Case was negative and prediction was positive

Each case is recorded for prediction and the user preference. The case negative implies that the user does not like that document. The case positive implies that the user like that document. The prediction positive implies that the system says that the user may like that document. The prediction negative implies that the system says that the user may not like user like that document.

These values are used to generate the graph for the precision and recall. Using these values the graph can be plotted. From which different recommender system can be compared. The recall and precision is one of the measures to compare the retrieval systems.

8.3 Recall

Table.3 Recall vs. Total number of documents

Total no. of documents	No. of Documents		
	CTR	RCTR	RCTR-fuzzy
30	0.42	0.62	0.85
40	0.48	0.68	0.82
50	0.5	0.65	0.87
60	0.5	0.72	0.9

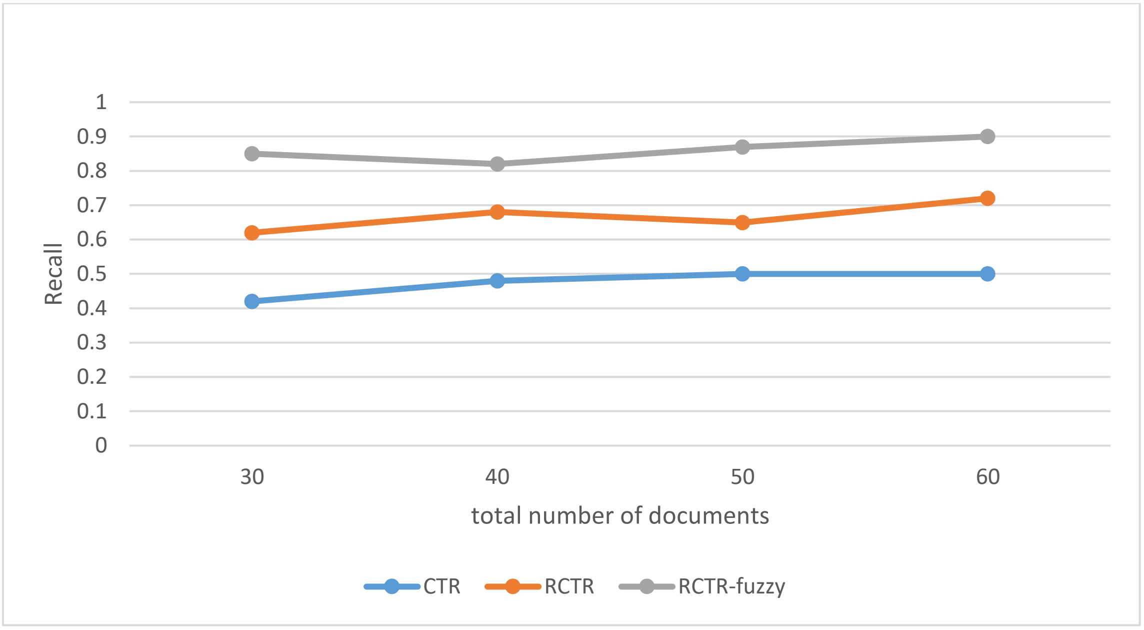


Fig 8.2 Recall

Retrieval is calculated using the formula,

$$\textbf{Recall} = \frac{\textbf{tp}}{\textbf{tp + fn}} \qquad \textbf{(8.3)}$$

This graph describes the retrieval for given a set of documents. It is a monotonically increasing graph. The retrieval of the RCTR-Fuzzy is more than other models, it is because of the inclusion of the fuzzy logic to evaluate the membership value of the cited document to the main document.

8.4 F-measure

Table.4 F-measure vs. Total number of documents

Total no. of documents	No. of Documents		
	CTR	RCTR	RCTR-fuzzy
30	0.473	0.625	0.8347
40	0.499	0.659	0.8099
50	0.524	0.66	0.8599
60	0.515	0.683	0.8526

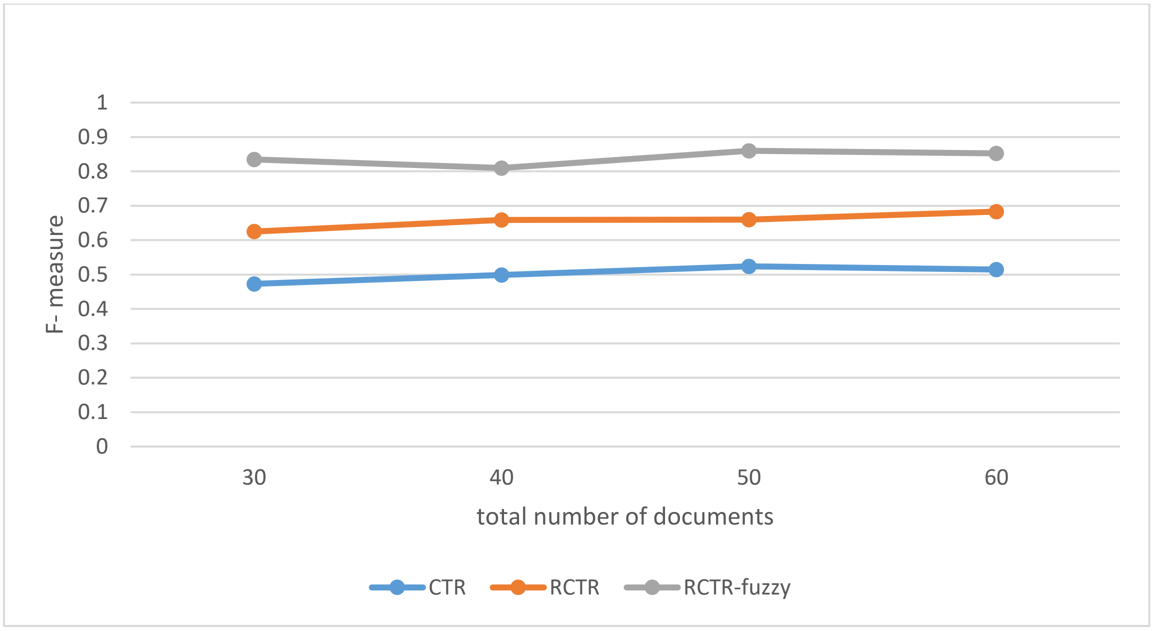


Figure 8.4 F-measure

Retrieval is calculated using the formula,

$$\textbf{F-measure} = 2^* \frac{\textbf{precison}*\textbf{recall}}{\textbf{precision}+\textbf{recall}}$$

This graph describes the retrieval for given a set of documents. It is a monotonically increasing graph. The retrieval of the RCTR-Fuzzy is more than other models, it is because of the inclusion of the fuzzy logic to evaluate the membership value of the cited document to the main document.

CHAPTER 9
CONCLUSION AND FUTURE WORK

9.1 Conclusion

In this paper implemented a fuzzy logic is introduced to determine the relativeness of document. After applying fuzzy to the form the network structure of documents the performance of the recommendation system is improved. The improved performance is calculated by using the Precision, Recall and F-measure.

9.2 Future Work

The Performance of the algorithm is still in good Standard. But it should be tested frequently for the upcoming various topic modelling and the document classification algorithms.

REFERENCES

[1] Hao Wang and Wu-Jun Li, “Relational Collaborative Topic Regression for Recommender Systems” IEEE transactions on knowledge and data engineering, VOL. 27, NO. 5, MAY 2015

[2] G. Adomavicius and A. Tuzhilin, “Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions,” IEEE Trans. Knowl. Data Eng., vol. 17, no. 6, pp. 734–749, Jun. 2005.

[3] D. Agarwal and B.-C. Chen, “Regression-based latent factor models,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2009, pp. 19–28.

[4] D. Agarwal and B.-C. Chen, “fLDA: Matrix factorization through latent dirichlet allocation,” in Proc. 3rd Int. Conf. Web Search Data Mining, 2010, pp. 91–100.

[5] M. Balabanovic and Y. Shoham, “Fab: Content-based, collaborative recommendation,” Commun. ACM, vol. 40, no. 3, pp. 66–72, 1997.

[6] B. Barragans-Martınez, E. Costa-Montenegro, J. C. Burguillo-Rial, M. Rey-Lopez, F. A. Mikic-Fonte, and A. Peleteiro-Ramallo,“A hybrid content-based and item-based collaborative filtering approach to recommend tv programs enhanced with singular value decomposition,” Inf. Sci., vol. 180, no. 22, pp. 4290–4311,2010.

[7] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” J. Mach. Learn. Res., vol. 3, pp. 993–1022, 2003.

[8] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez,“Recommender systems survey,” Knowl. Based Syst., vol. 46,pp. 109–132, 2013.

[9] R. D. Burke, “Hybrid recommender systems: Survey and experiments,” User Model. User Adapted Interaction, vol. 12, no. 4, pp. 331–370, 2002.

[10] Y. Cai, H. fung Leung, Q. Li, H. Min, J. Tang, and J. Li,“Typicality-based collaborative filtering recommendation,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 3, pp. 766–779, Mar. 2014.

[11] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, “An algorithmic framework for performing collaborative filtering,” in Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 1999,pp. 230–237.

[12] Y. Hu, Y. Koren, and C. Volinsky, “Collaborative filtering for implicit feedback datasets,” in Proc. IEEE Int. Conf. Data Mining, 2008, pp. 263–272.

[13] M. I. Jordan, Z. Ghahramani, T. Jaakkola, and L. K. Saul, “An introduction to variational methods for graphical models,” Mach. Learn., vol. 37, no. 2, pp. 183–233, 1999.

[14] Koren, R. M. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” IEEE Comput., vol. 42, no. 8,pp. 30–37, Aug. 2009.

[15] W.-J. Li and D.-Y. Yeung, “Social relations model for collaborative filtering,” in Proc. AAAI Conf. Artif. Intell., 2011, p. 1.

[16] G. Linden, B. Smith, and J. York, “Amazon.com recommendations: Item-to-item collaborative filtering,” IEEE Internet Comput., vol. 7, no. 1, pp. 76–80, Jan. 2003.

[17] Z. Ma, H. Yang, M. R. Lyu, and I. King, “SoRec: Social recommendation using probabilistic matrix factorization,” in Proc. ACM Int. Conf. Inf. Knowl. Manage., 2008, pp. 931–940.

[18] Y.-J. Park, “The adaptive clustering method for the long tail problem of recommender systems,” IEEE Trans. Knowl. Data Eng.,vol. 25, no. 8, pp. 1904–1915, Aug. 2013.

[19] S. Purushotham, Y. Liu, and C.-C. J. Kuo, “Collaborative topic regression with social matrix factorization for recommendation systems,” in Proc. Int. Conf. Mach. Learn., 2012, pp. 759–766.

[20] R. Salakhutdinov and A. Mnih, “Probabilistic matrix factorization,”in Proc. Adv. Neural Inf. Process. Syst., 2007, pp. 1257–1264

[21] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, “Item-based collaborative filtering recommendation algorithms,” in Proc. Int. World Wide Web Conf., 2001, pp. 285–295.

[22] I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock,“Methods and metrics for cold-start recommendations,” in Proc.Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval, 2002, pp. 253–260

[23] D. Q. Vu, A. U. Asuncion, D. R. Hunter, and P. Smyth, “Dynamic egocentric models for citation networks,” in Proc. Int. Conf. Mach.Learn., 2011, pp. 857–864.

[24] C. Wang and D. M. Blei, “Collaborative topic modeling for recommending scientific articles,” in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2011, pp. 448–456.

[25] H. Wang and W.-J. Li, “Online egocentric models for citation networks,” in Proc. Int. Joint Conf. Artif. Intell., 2013, pp. 2726–2732.

[26] Y. Zhen, W.-J. Li, and D.-Y. Yeung, “TagiCoFi: Tag informed collaborative filtering,” in Proc. ACM Conf. Recommender Syst., 2009,pp. 69–76.

[27] J. Chang and D. M. Blei, “Relational topic models for documentnetworks,” in Proc. Int. Conf. Artif. Intell. Stat., 2009, pp. 81–88.

APPENDIX

```
#For users

#load text mining library

library(tm)


#set working directory (modify path as needed)

setwd(“C:\\Project\\user”)


#load files into corpus

#get listing of .txt files in directory

filenames <- list.files(getwd(),pattern=”*.txt”)


#read files into a character vector

files <- lapply(filenames,readLines)


#create corpus from vector

docs <- Corpus(VectorSource(files))


#inspect a particular document in corpus
```

```
writeLines(as.character(docs[[30]]))


#start preprocessing

#Transform to lower case

docs <-tm_map(docs,content_transformer(tolower))


#remove potentially problematic symbols

toSpace <- content_transformer(function(x, pattern) { return (gsub(pattern, " ",
x))})

docs <- tm_map(docs, toSpace, "-")

docs <- tm_map(docs, toSpace, "'")

docs <- tm_map(docs, toSpace, "'")

docs <- tm_map(docs, toSpace, "•")

docs <- tm_map(docs, toSpace, """)

docs <- tm_map(docs, toSpace, """)


#remove punctuation

docs <- tm_map(docs, removePunctuation)

#Strip digits
```

```
docs <- tm_map(docs, removeNumbers)

#remove stopwords

docs <- tm_map(docs, removeWords, stopwords(“english”))

#remove whitespace

docs <- tm_map(docs, stripWhitespace)

#Good practice to check every now and then

writeLines(as.character(docs[[30]]))

#Stem document

docs <- tm_map(docs,stemDocument)


#fix up 1) differences between us and aussie english 2) general errors

docs <- tm_map(docs, content_transformer(gsub),

pattern = “organiz”, replacement = “organ”)

docs <- tm_map(docs, content_transformer(gsub),

pattern = “organis”, replacement = “organ”)

docs <- tm_map(docs, content_transformer(gsub),

pattern = “andgovern”, replacement = “govern”)

docs <- tm_map(docs, content_transformer(gsub),

pattern = “inenterpris”, replacement = “enterpris”)
```

```
docs <- tm_map(docs, content_transformer(gsub),
pattern = “team-“, replacement = “team”)
#define and eliminate all custom stopwords
myStopwords <- c(“can”, “say”,”one”,”way”,”use”,
“also”,”howev”,”tell”,”will”,
“much”,”need”,”take”,”tend”,”even”,
“like”,”particular”,”rather”,”said”,
“get”,”well”,”make”,”ask”,”come”,”end”,
“first”,”two”,”help”,”often”,”may”,
“might”,”see”,”someth”,”thing”,”point”,
“post”,”look”,”right”,”now”,”think”,”‘ve “,
“‘re “,”anoth”,”put”,”set”,”new”,”good”,
“want”,”sure”,”kind”,”larg”,”yes,”,”day”,”etc”,
“quit”,”sinc”,”attempt”,”lack”,”seen”,”awar”,
“littl”,”ever”,”moreov”,”though”,”found”,”abl”,
“enough”,”far”,”earli”,”away”,”achiev”,”draw”,
“last”,”never”,”brief”,”bit”,”entir”,”brief”,
“great”,”lot”)
docs <- tm_map(docs, removeWords, myStopwords)
```

```
#inspect a document as a check

writeLines(as.character(docs[[30]]))


#Create user-domain matrix

dtm <- DocumentTermMatrix(docs)

#convert rownames to filenames

rownames(dtm) <- filenames

#collapse matrix by summing over columns

freq <- colSums(as.matrix(dtm))

#length should be total number of terms

length(freq)

#create sort order (descending)

ord <- order(freq,decreasing=TRUE)

#List all terms in decreasing order of freq and write to disk

freq[ord]

write.csv(freq[ord],"word_freq.csv")


#for documents

#load text mining library
```

```
library(tm)


#set working directory (modify path as needed)

setwd(“C:\\Project\\docs”)


#load files into corpus

#get listing of .txt files in directory

filenames <- list.files(getwd(),pattern=”*.txt”)


#read files into a character vector

files <- lapply(filenames,readLines)


#create corpus from vector

docs <- Corpus(VectorSource(files))


#inspect a particular document in corpus

writeLines(as.character(docs[[30]]))


#start preprocessing
```

```
#Transform to lower case

docs <-tm_map(docs,content_transformer(tolower))


#remove potentially problematic symbols

toSpace <- content_transformer(function(x, pattern) { return (gsub(pattern, ” “,
x))})

docs <- tm_map(docs, toSpace, “-“)

docs <- tm_map(docs, toSpace, “’”)

docs <- tm_map(docs, toSpace, “‘”)

docs <- tm_map(docs, toSpace, “•”)

docs <- tm_map(docs, toSpace, “””)

docs <- tm_map(docs, toSpace, ““”)


#remove punctuation

docs <- tm_map(docs, removePunctuation)

#Strip digits

docs <- tm_map(docs, removeNumbers)

#remove stopwords

docs <- tm_map(docs, removeWords, stopwords(“english”))
```

```
#remove whitespace

docs <- tm_map(docs, stripWhitespace)

#Good practice to check every now and then

writeLines(as.character(docs[[30]]))

#Stem document

docs <- tm_map(docs,stemDocument)


#fix up 1) differences between us and aussie english 2) general errors

docs <- tm_map(docs, content_transformer(gsub),

pattern = “organiz”, replacement = “organ”)

docs <- tm_map(docs, content_transformer(gsub),

pattern = “organis”, replacement = “organ”)

docs <- tm_map(docs, content_transformer(gsub),

pattern = “andgovern”, replacement = “govern”)

docs <- tm_map(docs, content_transformer(gsub),

pattern = “inenterpris”, replacement = “enterpris”)

docs <- tm_map(docs, content_transformer(gsub),

pattern = “team-“, replacement = “team”)

#define and eliminate all custom stopwords
```

```
myStopwords <- c(“can”, “say”,”one”,”way”,”use”,

“also”,”howev”,”tell”,”will”,

“much”,”need”,”take”,”tend”,”even”,

“like”,”particular”,”rather”,”said”,

“get”,”well”,”make”,”ask”,”come”,”end”,

“first”,”two”,”help”,”often”,”may”,

“might”,”see”,”someth”,”thing”,”point”,

“post”,”look”,”right”,”now”,”think”,”‘ve “,

“‘re “,”anoth”,”put”,”set”,”new”,”good”,

“want”,”sure”,”kind”,”larg”,”yes,”,”day”,”etc”,

“quit”,”sinc”,”attempt”,”lack”,”seen”,”awar”,

“littl”,”ever”,”moreov”,”though”,”found”,”abl”,

“enough”,”far”,”earli”,”away”,”achiev”,”draw”,

“last”,”never”,”brief”,”bit”,”entir”,”brief”,

“great”,”lot”)

docs <- tm_map(docs, removeWords, myStopwords)

#inspect a document as a check

writeLines(as.character(docs[[30]]))
```

```
#Create document-term matrix

dtm <- DocumentTermMatrix(docs)

#link indicator

(z<- %*%x)

drop(z)

val.link(i,j)=ci.sigma(x, conf = 0.95, S.sq = NULL, n = NULL, summarized =
FALSE)

#convert rownames to filenames

rownames(dtm) <- filenames

#collapse matrix by summing over columns

freq <- colSums(as.matrix(dtm))

#length should be total number of terms

length(freq)

#create sort order (descending)

ord <- order(freq,decreasing=TRUE)

#List all terms in decreasing order of freq and write to disk

freq[ord]

write.csv(freq[ord],"word_freq.csv")
```

```
#load topic models library

library(topicmodels)


#Set parameters for Gibbs sampling

burnin <- 4000

iter <- 2000

thin <- 500

seed <-list(2003,5,63,100001,765)

nstart <- 5

best <- TRUE


#Number of topics

k <- 5
```

That done, we can now do the actual work – run the topic modelling algorithm on our corpus. Here is the code:

```
#Run LDA using Gibbs sampling

ldaOut <-LDA(dtm,k, method=”Gibbs”, control=list(nstart=nstart, seed = seed,
best=best, burnin = burnin, iter = iter, thin=thin))
```

```
#write out results

#docs to topics

ldaOut.topics <- as.matrix(topics(ldaOut))

write.csv(ldaOut.topics,file=paste(“LDAGibbs”,k,”DocsToTopics.csv”))


#top 6 terms in each topic

ldaOut.terms <- as.matrix(terms(ldaOut,6))

write.csv(ldaOut.terms,file=paste(“LDAGibbs”,k,”TopicsToTerms.csv”))


#probabilities associated with each topic assignment

topicProbabilities <- as.data.frame(ldaOut@gamma)

write.csv(topicProbabilities,file=paste(“LDAGibbs”,k,”TopicProbabilities.csv”))


#Find relative importance of top 2 topics

topic1ToTopic2 <- lapply(1:nrow(dtm),function(x)

sort(topicProbabilities[x,])[k]/sort(topicProbabilities[x,])[k-1])


#Find relative importance of second and third most important topics

topic2ToTopic3 <- lapply(1:nrow(dtm),function(x)
sort(topicProbabilities[x,])[k-1]/sort(topicProbabilities[x,])[k-2])


#write to file

write.csv(topic1ToTopic2,file=paste(“LDAGibbs”,k,”Topic1ToTopic2.csv”))

write.csv(topic2ToTopic3,file=paste(“LDAGibbs”,k,”Topic2ToTopic3.csv”))
```