Survey on Recommender Systems

Akshaya Aradhya V

Instructor: Dr.Margaret H. Dunham

PROJECT 1

I. Introduction and Overview:

Most of us are heavily dependent on internet and other web technologies for reasons like online shopping, searching information on the web, taking surveys for giving feedback etc. Recommender systems came into existence as users were interested in retrieving either personalized queries or for fetching more elaborate and diversified results related to their searches from a very large collection of items or information. The basic function of recommender systems was to provide recommendations to the users based on items or information related to their interests and in some cases, to provide guesses or ratings to each item which the user may prefer [3]. The ratings may be either real-valued like numbers ranging from 1-5 to indicate degree of preference [8, 10, 11, 13] or binary to indicate whether the user liked or disliked the item [3, 6]. This gained popularity due to the fact that these systems acted as tools for groups of people or individuals to discover more important, accurate and closely related items based on his search query and interests. In e-commerce applications, they are used to provide information, evaluations and suggestions regarding which products are more popular and worth buying and they record the customers behavior and assist them in completing the shopping process in an effective way [12]. Recommender systems have also become an important research area in the academia and the industry due to its social and commercial impact.

The personalized queries are used to obtain dynamically generated personalized answers to users with different preferences using the user preferences stored in their individual profiles containing information such as gender, income, age, address, marital status etc. The term preference here refers to obtaining additional meaningful information from the database using co-operative query answering methods or scoring functions which assign a score to each tuple obtained from the result set [2]. The main accuracy metric used for this were precision, recall and mean absolute error and due to further advancement in research, real user experience is also considered as going beyond accuracy was

essential. The most notable problem with personalized queries is the problem with over-specialization [5], which would sometimes result in obtaining very few queries as a result or in some cases; they produce the items which only rate highly against the user's profile. For example, a user who has not been interested in classical rock but not jazz music will never get recommendations on an album which is a fusion of both, even though it is the greatest hit of all times. This was a problem in earlier recommender system used in Yahoo music website. It was also discovered that some randomness in search results did help in such cases. This led to the need for result diversification which was used to increase user satisfaction keeping the properties of recommendation lists generated with the help of collaborative filtering algorithms and intra-list similarity metrics which looked at the lists as individual entities having their own rights rather than an aggregate of independent suggestions [1,4]. For example, if a user decides to find a hotel in an unknown city, he should ideally get recommendations of hotels closest to the airport or train stations, with additional optional facilities like pick-up facility from a taxi service or suggestions for places to visit near the hotel which are more intuitive and effective in helping the user decide better, instead of suggesting hotels with similar amenities. Additionally, these types of queries are more useful when user satisfaction is preferred more than accuracy of the results. Keeping these factors in mind, a vast amount of research has been done to come up with new, efficient and improvised recommendation algorithms which perform well irrespective of the size of the database.

When it comes to choosing a recommendation algorithm from a large number of algorithms which have already been developed, offline experiments need to be conducted over the same set of real data with structural constraints. In case of new algorithms, an evaluation metric is usually chosen to rank the algorithms quantitatively using individually assigned scores and selecting the appropriate evaluation metric is crucial in deciding whether the system is good or bad [3]. In the coming sections, we will discuss about the techniques used to build recommender systems with some suitable examples.

II. Classification of Recommender systems:

In this paper, we will be looking at three mainly used techniques to build recommender systems, which are – collaborative filtering approach, content-based (or knowledge-based) approach and hybrid approach [5, 6, 8, 12].

1. Collaborative Filtering Recommender Systems

The collaborative filtering recommender systems (CFRS) use collaborative filtering methods to provide the user with recommendations based on what other users with comparable interests or preferences might have liked in the past. In essence, it leverages similarity between people to make recommendation because their preferences are sometimes correlated, making the system very useful and efficient when there are more items in the database than the users [7]. Hence, the document is represented based on the implicit or explicit ratings and evaluation of the document provided by other users in the past [4,5]. The technique which was used for this was not dependent on any machine readable representation of the objects being recommended and it worked very well with complex objects like movies, choice of cuisine and music where there may be lot of variations in each user's taste which is most probably the reason why there is a lot variation in preference and ratings. This technique was used in e-commerce sites like Amazon, Expedia and Netflix and it was called 'people-to-people correlation' [9]. They were used to generate patterns of similarity or agreement between users using the data which was collected over time.

The main advantages of this type of recommender systems is that they are more effective when it comes to customer satisfaction as they recommend the most appropriate items to users which are personalized at the same time. The collaborative filtering algorithms are designed such that the accuracy

of their prediction increases tremendously over time as and when more user preferences are added to the database, irrespective of the size of the database.

The main drawback of this method is that the system would not be very effective when user preferences change unexpectedly, as the system still focuses on past interests of the user. If a user has brought books on astrophysics for a long period of time and if he suddenly gains interest in other areas like psychology and decides to buy books related to this instead, he would still be recommended books on astrophysics, which might not be very useful to the user. It also suffers from the problems of sparsity of ratings [7] or single votes for items in the database along with the problem where a new user or new item is added to the database. This problem can be counter acted by methods like default voting, weighed majority prediction, adjusted weighed sum and case amplification [6, 21] which are used to increase the prediction accuracy. Most recently, a technique called Singular Vector Decomposition (SVD) which is used to reduce dimensionality using matrix factorization methods which provide the approximation for the best lower rank for the original matrix came into effect. SVD is extensively used to take care of the sparsity problem [5, 6] which reduces the prediction accuracy if the number of rated items in the database is less. It worked well with the vector space model and it was used for latent semantic indexing. The main idea of why this was used is to prove that the reduced space contains less noise and the latent association was captured more efficiently.

The instant recommendations used by Amazon are a classic example for CFRS [12, 20]. The system is used to recommend a variety of items like books, CDs, Apparels, and Furniture etc. An interest profile for a user is built using the details of his past purchases and ratings given by the user for a particular item. The items which are similar to the features of the items found in the interest profile are then recommended to the user.

2. Content-based Recommender Systems:

Content (or Knowledge) based recommender systems (CBRS) [3, 6, 15] will recommend the items to the user based on what he has already preferred in the past keeping in mind what contents are present in the document at the given time. In other words, it has to have the knowledge of the catalog and the user along with the functional knowledge, which is essentially the mapping between the user's need and the item which would possibly satisfy the user's need. The research on content-based methods are closely related to information retrieval and filtering research because of the early advancements of these areas and increasing importance of textual applications [8] or information, they focus more on recommending textual information keeping the user's interests, profile and preferences in mind. The content based filtering method is used where individual weights are assigned to the keywords of the content and stored in a content vector and other candidate items are compared with items which were rated higher by an individual user earlier and after establishing similarity between them, the items which have higher similarity measures will be assigned with higher utility and these articles will be recommended to the user [5, 8].

The type of information for the user's profile derived by the content-based recommender systems was largely depended on the learning method used, like Decision trees, neural networks etc. This technique used in this was named as 'item-to-item correlation' [9], where in the order size of a customer who has placed a few items in his basket can be increased by providing the customer with options to choose other complementary items. This was either done automatically or with some manual effort and it was used in sites like Reel.com which was used to match movies or in match maker present in the Moviefinder application. A classic example for CBRS is the Personal Logic System [12, 15]. The system first gathers the product requirements specified by the user and uses certain keywords from that

to consult the knowledge base. The items which match the user requirements are presented as final result.

The main advantage of this method is that it does not depend on the user ratings of items in the database and hence, even if the database does not contain user preferences, the prediction accuracy is not affected. Even if the user preferences change, it has the capacity to adjust its recommendations in a short span of time. The main drawback of this content-based recommender system is the need to know all the details of an item really well, even where the features of the item is stored in the database in a way where it can be inferred indirectly.

3. Hybrid Recommender Systems:

Hybrid recommender systems combine the methods of the earlier two methods to produce better results by involving all the advantages of the two techniques and by removing their drawbacks at the same time. Some of the hybrid recommenders mentioned in [6, 8] are:

1. Weighted recommenders

The results of all the recommendation techniques are used to compute the score of an item which is to be recommended. There assumption used in this technique is that the relative value of numerous other techniques is uniform across the space of possible items. It is easier to do credit assignment and adjust the hybrid dynamically using this method.

2. Switching recommenders

This hybrid incorporates the item level sensitivity to its strategy using which, the recommender switches between the techniques used for recommendation. The notable characteristic of this system is that it is sensitive to the strength and weakness of the constituent recommenders. However, it may add more complexity to the process of recommendation because of the need to determine the criteria for switching.

3. Mixed recommenders

These types of recommenders are used when there is a need to make large number of recommendations in a short period of time. This type of recommender takes suggestions from multiple sources to generate final results. Most of the times, items are ranked dynamically using complex combination techniques.

4. Cascaded recommenders

The recommendation process involved many stages where in the candidate set which are ranked by one technique is refined by the other technique in the next stage. The second step of the technique will never focus on items which are poorly rated and the lower-priority techniques are never used in this stage. Additionally, results in the ratings given by the higher priority techniques cannot be overturned.

5. Feature Augmentation recommender

The rating or classification of an item in the database is generated by one technique and it is fed to the next technique for further processing. In essence, the augmentation hybrid uses the features of the first recommender. It is not to be confused with the cascaded hybrid because the output of the first recommender is not used by the second one while generating rankings and the combination of the final result is done in a prioritized way.

III. Brief descriptions of some important Recommender Systems:

1. GroupLens

GroupLens [5, 8, 10, 13, 24, 26] was an architecture used for distributing ratings, which can be modified by anybody who wants to alter or improve a news client to use predicted scores or to make a news client to allow entry of evaluations. It was introduced to mine the reactions of people who read news articles and the architecture was successful in scaling up to a very large number of users and ratings. The system employed collaborative filtering methods on vast number of news articles (or content) present in NetNews in order to help users to find useful articles in a much easier way. Better Bit Bureaus (BBB) [10] were the rating servers which were used to gather and predict scores based on the heuristic that if users had already agreed on a subjective evaluation of the articles in the past, they will probably agree on it again. They used special news groups to share the ratings in order to allow collaborative filtering for users in various other sites. Ratings could range from the numbers from 1 to 5 where 1 is the lowest rating which a user can assign to an article. Additionally, if the user had spent more time on an article, the number of seconds spent on that article was recorded and used to determine the rating. GroupLens uses a Correlation algorithm which uses Pearson correlation [13, 26].

Ratings were also accepted in the form of text like ‘excellent’ for a rating of 5, from users opting for varying amounts of privacy from being totally anonymous to users having authorized signatures. Using the combined ratings, the scores were predicted by the BBB. Matrix filling techniques were used when the ratings were on the same scale, where the articles were represented by rows and the users were present in the column. This was useful when some of the articles had not been rated by the users. If multiple users have varying opinions of the article, BBB used to predict different scores to each reader by varying the weight given to the ratings.

Netnews used two mechanisms to limit the reader’s attention to the articles which interested them the most. Firstly, using content-based or cognitive filtering, the bulletin board was divided into multiple news groups with subtopics in each group. This helped the reader to identify the news better. Secondly, Usenet was made to propagate only those articles approved by the moderator if a moderator’s approval is needed for the postings on a news group. A news client was used to display both a summary of author and the corresponding subject line to the readers or to display one or more the articles in a discussion thread together. The readers could then indicate which articles interested them the most using this information. Additionally, some files called ‘kill files’ [10] were used to identify the text strings which do not interest a particular user. If a subject line or an author’s name was put into this file, articles related to them were not displayed. String search facility was provided by the news readers which used the news clients to find the articles quickly. Content-based filtering techniques are employed by the string search facility and the kill files. Strings could be combined using Boolean operators. The profile of which text strings should be included or killed is generally more complex than a collection of character strings. If the relative importance of each set of term is expressed in weights, the profile can be constructed using weight vectors instead. Social filtering techniques can also be employed to select the news articles depending on the subjective judgments of the groups of people. If the users put an author’s name in a kill file, then all the articles related to the author, including those which have

been co-authored by the author may be filtering out. So this technique was not without drawbacks. Hence, the collaborative filtering techniques were employed, where subjective evaluations on the text content were done by the readers based on the quality, interestingness, authoritativeness etc.

2. Tapestry

Tapestry [5, 8, 10, 11, 14] was a monolithic system which was designed to handle large streams of electronic documents like Netnews articles, news-wire stories, emails etc. This system used a client-server model along with correlation algorithm and it not only filtered the mail it received, it unified the ad-hoc queries over the repository of mail sent in the past. For example, if a user was interested in knowing what 'Sigmund Freud' is and if he issues an ad-hoc query using he keyword 'Sigmund', he may get too many articles or documents as a result. Searching for documents containing 'Freud' or both the keywords 'Sigmund' and 'Freud' would have resulted in better search results. This search is tested and then installed as a query filtering. The new documents which satisfy the filter will be sent to the user's mail box.

The major architectural components of Tapestry are as follows:

1. Indexer: Used to read and add the documents like emails, newswires etc to the document store. The index fields [16] obtained by parsing the document is referenced in other queries.

2. Document store: It is an append-only type of long-term storage facility which is used to maintain indexes on the documents present in the database which in-turn increase the efficiency of query searches.

3. Annotation store: It is an append-only type of worm disks used to store the annotations present in the document. The annotations are appended as additional fields and they are combined with documents into a single store.

4. Filterer: This is the computationally intensive part of the Tapestry where the queries are run by the filters. Tapestry is implemented on a database which supports SQL and because of this reason, Tapestry query language (TQL) is used to write the queries for the filter. Unlike GroupLens, Tapestry did not use aggregate queries. It was used to find the documents matching the batch of user-provided queries and it places it them in the respective little box of the owner of the query. Filtering is done in two stages. In the first stage, document is either rejected and discarded or accepted and placed in the little box assigned to a particular user. In the second stage, appraiser functions are applied on the contents of the little box. Unlike GroupLens [10], Tapestry made a sophisticated use of the subjective evaluations because it could combine them with content-based evaluations which could contain text or binary recommendations like accept or reject. For example, a reader can request articles containing the keyword 'Music' rated by a user 'Tom' with the text evaluation like 'Good'.

5. Little box: It contains documents which are removed by the document reader at a later stage. It holds limited number of messages which will be copied to the work station.

6. Remailer: If a particular user wants to access Tapestry with their mail reader, the remailer sends the contents of the little box through email.

7. Appraiser: It is used to dynamically prioritize and categorize the documents present in the user's little box after applying personalized classification on each document.

8. Browser: It is the user interface used to access the Tapestry services which includes the facility to retrieve, organize and display the documents, supplying annotations, adding, modifying or deleting filters and running ad-hoc queries.

3. Ringo

Ringo [14] is an online music recommendation system which uses social information filtering to make personalized recommendation. Using correlation algorithm (Pearson correlation technique), Ringo compares the user profiles to determine whether the users have similar taste in music and it computes the weighted average of the ratings for that particular artist, song or album using the information in them. A scale of 1 to 7 was selected for the ratings and they were not normalized since multiple users can rate the same albums in many different ways. Ringo uses these ratings to suggest new artists or albums, make predictions on them or list the albums or artists which the user might possibly hate in the future. Additionally, Ringo had offered a personalized review to each individual user, from other users who had similar taste in music. The algorithm used by Ringo is discussed in another section.

4. EntréeC

Entrée [8] is an online recommender system which employs case based reasoning technique to rank a restaurant. An entry point submitted by the user is stored and it used to find restaurants which are more similar to it. Entrée supports the feature where a user can interactively retrieve an item. If a user critiques a recommended restaurant, he is presented with an alternative and this process continues until the user finds an acceptable option. Using this method, a user is presented with fewer choices in

restaurants and he could then efficiently browse through other features like quality of the cuisine served, price etc. A user profile is not retained since Entrée is a stateless application.

A recommender system which uses case based reasoning contains the case base, similarity measure used to compare cases, vocabulary used to describe cases and knowledge required to transform the solutions which have been recalled. The vocabulary used here is obtained from the keywords for the features present in the restaurant guide. If a restaurant is determined to be 'nice' by a user, then the system determines this as a restaurant which is of a high interest to the user. The similarity is used to decompose into a set of independent attributes and a local similarity metric is defined for each attribute like price, quality of cuisine served etc. Two items are judged by the similarity with respect to that attribute. Combination of metrics is done in a cascaded way and when there is a tie between two or more restaurants which are ranked equally by a metric at a higher level, each of the subsequent metric is used to break the tie. This makes the global metric more easier to design and it's behavior can also be predicted with less complexity. Entrée uses semantic ratings, where the evaluations consists the reason behind a given rating along with the preference of each user, making the system more interactive and the algorithm employed vector similarity and correlation techniques. The main drawback of this system was the fact that even though there were cases where a large set of items which were considered to be equivalent by a similarity metric, was returned back to the user after the retrieval process, only a few of them were selected randomly and displayed back to the user. The user misses out on the opportunity to pick a better restaurant just because of the fact that it was not displayed to the user at the given time. EntreeC [8] was designed to take reduce the effects caused due to this drawback. It was a hybrid recommender which is a modified version of Entrée and it used a collaborative recommendation technique in the final cascaded step. If a user searches for a restaurant with specific features and if he critiques a restaurant or a cuisine, it is stored in the knowledge base of the system. Later, if another with the same preferences searches for this cuisine, he would not get back

the item which was critiqued earlier. Cascading the collaborative filtering technique, as well as the content-based technique in that case proved to be the most effective way to increase the efficiency and usability of the system.

5. LIBRA

Learning Intelligent Book Recommending Agent (LIBRA) [8, 16] used by Amazon, makes use of the database which holds the information of all the books extracted from the web pages at Amazon. The actual texts of the items are not used in this system. Amazon.com uses topic diversification algorithms to improve recommendations. The content information about titles consists of textual meta-data exclusively. The ratings range from 1 to 10 and the system uses Bayesian learning algorithm to learn a profile and to produce a list of titles from the system catalog after ranking them. A subject search is required to obtain the URLs containing book description of relevant titles. LIBRA downloads the files from these URL and it makes use of a simple pattern based IE (Information extraction) system [17] to extract the data for each title in order to extract structured information from unstructured text.

IE performs the task of extracting pieces of data such as keywords or a set of sub-strings or 'fillers' [16, 17] from the natural language document or web pages. The pattern matcher also uses pre-filler, post-filler for each slot [17]. A pre-filler is a text pattern that should match the textual information which immediately precedes the filler and a post-filler is a pattern which should match the text after the filler.

The author's information including the reviews and comments received from the customers, title of his work and other related titles, subject term and synopses are put into different slots which is to be used by the recommender. This information is considered as the additional information regarding

the book, unlike Amazon, which currently uses collaborative filtering technique to generate information regarding authors or titles [16]. The slots containing the information regarding date of publication, ISBN, details of the publisher, price of the book etc are not used by LIBRA. The text content from each slot is subjected to further processing into tokens and a vector of bags of words.

Tokens are ideally the unordered bag of words. The author's name is preprocessed into unique tokens of the order first and last name. The list of stop-words is also removed. Since a book is considered to be directly 'related' to itself, it is added to the list of the slots which contain information regarding the related author and title. The words related to the published reviews and comments provided by the customers, as well as the synopses are subjected to grouping and the grouped words are put into one bag called description. The user can now rate a selected title for a particular author on a scale of 1 to 10, where a rating of 1-5 is considered to be negative and the rest of the ratings are positive. LIBRA uses this information to predict the rankings of titles.

LIBRA uses an inductive learner [17]. It is essentially a bag of words Bayesian text classifier [25, 27] which is used to handle a vector of bags. A document is ordered as a sequence of words drawn from the same vocabulary. It uses the naïve Bayes assumption, where the probability of a word event is not dependent on the context and position. It is only dependant on the document class. The strength of word in a slot is calculated based on how many times it appears in a positively rated book than a book which has been rated negatively. It is then added to a profile, which is used to predict the ratings of other books based on the posterior probability of positively categorizing the book. After this, the book which holds the top-score of recommendations is displayed back to the user. This way, if a user's preference changes over time, the system could dynamically alter the recommendations in order to suit the user's needs.

6. NewsWeeder

NewsWeeder [8, 22] is a text recommender which uses the words in the texts as features. It is basically a NetNews filtering system which lets the user rate his interest level (from 1 to 5) for each article and learns their user profile based on the information obtained. This eliminates the need to depend on the user to create their profiles altogether. If a news reader enters a newsgroup to peruse through a topic, he is presented with one-line topics based on rankings. A user can randomly select a group of topics from the information visible through the recommender's interface and rates them. This is known as active feedback. The next article will be visible to the user based on which items he has read which eliminates the passive feedback methods which involve time spent in reading an article like the method used by GroupLens. The system then uses the offline data overnight to collect the ratings to create a new model for the user's interests.

The text is first parsed into tokens, which includes noun phrases, special symbols and punctuation. It uses the bag of words model where the order of tokens is not preserved with an assumption that information needed for filtering is captured appropriately. The tokens are not stemmed in order to avoid wasting additional information involved in the use of exact tokens and to find out the statistics of the rare words present in the unrated text.

The learning involves dividing the articles into training sets and test sets. The tokens from the training articles are parsed and those which appear less than three times or those which appear most frequently are thrown out. It calculates the average of the vectors in each category for ratings. The vector similarity algorithm is used to find the similarity of each rating category prototype with each training document. Further rating predictions are made using linear regression using rating category similarities.

7. BookCrossing

BookCrossing [3, 4] is recommender which allows readers to discuss, review and share their interests. Ratings are provided on the scale of 1 to 10. The main problem with this system is the sparse data set due to large number of books and less number of ratings provided by the users. Since book reading takes considerably more amount of time than watching movies or listening to songs, readers usually tend to read the books which they might like and because of this reason, the ratings are skewed mostly towards the positive side.

Since BookCrossing does not mandate the users to rate each book which they have read and also due to the fact that there are less number of users registered as readers on that site, the ratings do not represent the overall population of readers. Hence, the topic diversification algorithms are used by this in order to expand the results of the query searches in order to interest the users.

8. Jester 2.0

Jester 2.0 [23, 24] is a web based recommender system for jokes, which is used to predict the jokes which the user might interest the user. It uses principal component analysis (PCA) and cluster model based linear time algorithm to retrieve personalized information for each user. The system aids statistical pattern recognition by using a continuous rating scale. The algorithm is discussed in section V later.

IV. Commonly used metrics

This section summarizes the metrics which are used to compare various experimental results in the surveyed paper for several combinations of algorithms which have been listed in section V.

1. Item to item similarity:

Item to item similarity measure [3, 20] is use to compute the similarity between items. Most of the times, the maximum likelihood estimate (MLE) is used to calculate conditional probability of items. If I_x represents the number of users who had used item x, and the number of users who had rated both $\mathrm{i}_{1,}$ and i_2 is represented by $I_{i_{1,}i_2}$, the conditional probability of items in binary usage can be represented as:

$$\mathrm{pr}(i_{1,} \mid i_2) = \frac{|I_{i_1,i_2}|}{|I_{i_1}|}$$

If there are P_j sets of products where $p_k \in P_j$, and actual ratings of p_k is r_i (p_k) and predictions are denote by $pr_i(p_k)$, then the Mean Absolute Error is found out as:

$$|\bar{E}| = \frac{\sum_{p_k \in P_j} |r_i(p_k) - pr_i(p_k)|}{|P_j|}$$

2. Root Mean Square Error

Mean square error [4, 5] involves the concept of squaring the error before summation, which in turn highlights larger errors than the smaller ones. The error E is calculated as below:

$$\mathrm{E} = \sqrt{\frac{\sum_{x=0}^{n} (r_{ux} - pr_x)^2}{n}}$$

Here, n represents the number of ratings available, pr_x represents the predicted rating for item x and r_{ux} represents actual rating provided by the user.

3. Standard Deviation

If the standard deviation [14] of the errors is low, the scheme is considered to be consistently accurate. The error should be minimized as much as possible.

$$\sigma = \sqrt{\frac{\sum(E - \bar{E})^2}{N}}$$

4. Precision

Precision [1, 3, 4, 5, 16, 17, 18] measures the percentage of the high ratings among those which were actually predicted to be high by the recommender. It is generally represented in terms of the ration of the number of true positives to the sum of true positives and false positives.

$$\text{Precision} = \frac{\#\ of\ true\ positives}{\#\ true\ positives + \#\ of\ false\ positives}$$

5. Recall

Recall [1, 2, 3, 4, 5, 6, 16, 17, 18,] is the percentage of the correctly predicted ratings which were high among the actual ratings which were high. In general, it is the ratio of the number of true positives to the ratio of the sum of the number of true positives (TP) and number of false positives (FP).

$$\text{Recall} = \frac{TP}{TP+FN}$$

6. F-Measure

F-measure [3, 16, 17, 18] is the weighted average of precision P and recall R.

$$\text{F-measure} = \frac{2 * \text{P} * \text{R}}{P+R}$$

7. Normalized Discounted Cumulative Gain Measure:

This measure is used in IR systems where it is necessary to measure the gain of the item being at a specific place in a list. If there is a list of ordered items, the p^{th} element of the gain vector (G) of the list is computed as below:

$$\text{G [p]} = \sum_{i=1}^{n} K\ (x_p, o_i)\ (1 - \propto)^{r_{o_i}, p-1}$$

Corresponding cumulative gain vector (GC) is calculated as below:

$$\text{CG [p]} = \sum_{i=1}^{p} G[i]$$

Subsequently, since the value of CG above is weighted based on the position in the list, the discounted cumulative gain vector (DCG) [1] is calculated as follows:

$$\text{DCG[p]} = \sum_{i=1}^{p} \frac{G[i]}{\log_2(1+i)}$$

The ideal DCG is used to normalize the DCG vector for a list, but since this is an NP-complete problem it is usually approximated to some value using heuristics [1] like greedy and interchange.

V. Algorithms and their comparisons

1. Algorithm for Topic Diversification:

The detailed algorithm from for topic diversification [1, 4] is as below:

Let the set of products be denoted by P = $\{p_1, p_2, \ldots\ldots p_n\}$, users be denoted by U = $\{u_1, u_2, \ldots\ldots u_n\}$, partial rating function for each user be r_k: P -> [-1, +1]. If the ratings $r_k(p_i)$ is positive, it implies that a user has liked product p_i and if the rating is negative, it shows that he disliked it. If one user, say u_i is an active user [3], based on the similarity of the rating function, the top-M users (u_j) who are most similar to this user are selected to form the neighborhood. Cosine distance or Pearson correlation can also be used for computing the similarity values [20] of c(u_i, u_j). Based on the products which the neighbors of u_i have rated and which are relatively new to u_i, a prediction of liking is produced, which is denoted by $w_i(p_k)$. This value depends on the ratings assigned by the neighbors a_j who have rated the product p_k, which is denoted by $r_j(p_k)$ and it also depends on the similarity of users who have voted. Using the predictions of w_i, a list of top-N recommendations are produced.

The values for $w_i(p_k)$ [4] is calculated as below:

$$w_i(p_k) = \frac{\sum_{p_e \in \mathrm{P}'_\mathrm{k}} (c(p_k,p_e) \cdot r_i(p_e))}{\sum_{p_e \in \mathrm{P}'_\mathrm{k}} |c(p_k,p_e)|}$$

$$\text{where } \mathrm{P}'_\mathrm{k} := \{ p_e \mid p_e \in clique(p_k) \wedge r_i(p_k) \neq \perp \}$$

The list is denoted by D_{w_i} : {1, 2, ……, M} -> P, where D_{w_i} lists the highest predicted recommendation first and $D_{w_{i^*}}$ denotes the new list obtained after topic diversification. The topic diversification is dependent on the function which is used to quantify the similarity between two product sets. It is as follows: c^* : 2^P X 2^P -> [-1, +1] , where c^* is instantiated with the metric for taxonomy driven filtering

[4]. The similarity is dependent on the classification of the sets of products, where each product may belong to one or more classes. These classes are arranged hierarchically in large classification taxonomies like those created by Amazon for DVDs, Dresses, Books etc. Content description along with the product descriptions is associated with each of the product. The steps are summarized as below:

Begin

for each list entry y ∈ [2,M] , collect product p

Compute the similarity with the set { $D_{w_{i^*}}$(k) | k ∈ [1,y[}

for each product p

Sort all the products and compute $D_{c^*}^{rev}$

merge D_{w_i} with respect to $\emptyset_f$ to get final rank $D_{w_{i^*}}$

End

The algorithm has space and time complexity of O(n^2). Additionally, in the algorithm above, p is selected depending on the condition that they do not occur in the position o < y in $D_{w_{i^*}}$. The set { $D_{w_{i^*}}$(k) | k ∈ [1,y[} contains only the newly predicted recommendation which precede rank y. The dissimilarity rank $D_{c^*}^{rev}$ is obtained after sorting all the products in reverse order with respect to c^*(b) and it's impact is defined by the diversification factor $\emptyset_f$. Lower values of $\emptyset_f$ produces recommendations list closer to D_{w_i}, whereas higher values produces list closer to the original relevance order. If the values for $\emptyset_f$ is lesser than or equal to 0.4, the items which change with respect to the

original list were less than 3. The final rank $D_{w_{i^*}}$ is obtained after merging D_{w_i} with respect to $\emptyset_f$.

Precision and recall metrics were computed for item based (Amazon [20]) and user based collaborative filtering [3] data and for different levels of diversification. Apart from precision and recall metrics, the Intra-list similarity metric is used to determine the diversity of a list. This metric is insensitive to permutations within the list and higher values denote low diversity.

It is calculated as below:

$$\text{Intra-list similarity}(D_{w_i}) = \frac{\sum_{p_k \in \beth D_{w_i}} \sum_{p_e \in \beth D_{w_i}, p_k \neq p_e} c_o\,(p_k, p_e)}{2}$$

2. Model based algorithms and Extensions

Model based collaborative filtering algorithms [21] use a database of user votes from a large set of active users (also known as user database), to learn a model and to estimate a model and then use this for further predictions.

If votes are integer values between o and n, we can predict the votes for unobserved items for an active user as shown below:

$$P_{a,i} = \mathrm{E}(v_{a,i}) = \sum_{j=0}^{n} \Pr\left(v_{a,i} = j | v_{a,z}, z \in I_a\right)\ j$$

Here, the probability value will represent the probability that the active user will have a vote for item i, given the votes which have been observed previously.

2.1 Bayesian Networks and Bayesian Classifier:

In Bayesian network model [21], each item corresponds to a node in a given domain and the probable value of the vote for each item is represented by the state of a node. If there is no interpretation of missing value for a domain, a state corresponding to 'no vote' is assigned to it and an algorithm is applied on the training data for learning the Bayesian networks. The model determines if there are any dependencies between items and forms a network where the best predictable value for the item is indicated by the parent items. A decision tree encoding the conditional probability for the node is represents the conditional probability table. In essence, there is a decision tree for each title [21].

Bayesian text classifiers [16, 25, 27] are based on the assumption that all the word events drawn from a same vocabulary is independent of the position of the word in the document as well as the context of the word. The learning algorithm is supervised and the training set has text documents with labels. The steps given below shows the different steps involved in the algorithm:

1. Training set consists of book rated by users.

2. The training set is preprocessed and header information is discarded. The main body is tokenized and lists of words are created. Stop words are discarded since they do not contribute to text classification.

3. The ranking of titles are generated in order of preference without calculating the numerical ratings for each title. The probabilistic binary categorization technique is used to predict the probability that a document is rated as positive.

4. If there exists a vector of documents V and each document is contained in a slot S, the probability of each word given the category is calculated and posterior category of document d is calculated as follows:

$$P(c|d) = \frac{P(c)}{P(d)} \prod_{n=1}^{|d|} P(w_n|c)$$

Here, w_n is the word present in the nth position of the document, c is the class, |d| is length of the document with respect to words in it.

5. A model is created using the information for mapping the new input sample into a separate category. A model contains vocabulary tables and it is saved as a text file.

The space complexity of the algorithm is linear in the size of the training data. The time complexity of the algorithm is O(nm) where n is the number of items and m is the number of words in the dictionary.

2.2 Cluster models:

A certain category of users capture a common set of preferences and they are placed in the same group or cluster. If a user's class is known, the preferences (or votes) for the items in are assumed to be independent. In a Bayesian classifier [21], the probabilities of votes are conditionally independent if they are members of a class C (which may be unobserved), which usually consists of very less number of discrete values. The joint probability of class and votes for a probability model is expressed using naïve Bayes formula from [21] is as shown below:

$$Pr(C = c, v_1, v_2, \ldots, v_m) = Pr(C = c) \prod_{j=1}^{m} Pr(v_j|C = c)$$

$Pr(C = c, v_1, v_2, \ldots, v_m)$ represent the probability of observing a user in a given class c along with a set of vote values. Pr (C = c) represents the probability of a class membership and $Pr(v_j|C = c)$ can be calculated from the training data containing the votes provided by a user.

In Jester 2.0, clustering and PCA are done in an offline environment. The complete pairs of jokes from a predictor set are taken and correlations between them are computed over the user base. A two dimensional mapping of each user is created using PCA and the clusters of users are created using hierarchical sub divisions. Each 2D plane is divided recursively into rectangular regions. The density of these regions increases as along the direction of the origin. The ratings of jokes in each cluster are averaged and the humor preferences are calculated using this. The prediction vector for each cluster is obtained after sorting the vectors of jokes in decreasing order with respect to their mean ratings and the jokes are recommended using these vectors. The ratings are processed by Jester 2.0 after all the jokes in the predictor set are rated. Let there be n number of jokes in the database [24]. The correlation matrix (represented by Eigen components) for all the prediction set is calculated over the user base in space complexity of O(n^2). The time complexity for the algorithm is O(n).

2.3 Item to item collaborative filtering

Item based collaborative filtering [20, 25] algorithm calculates the item similarities in an offline stage and it builds a model using the computed information.

The following steps summarize the steps which makes the prediction for an product P for an active user U:

1. Retrieve all items rated by U

2. With respect to the set of retrieved items, compute the target item's similarity S. In some cases, adjusted cosine similarity [25] can be used.

3. Predict the target item using the weighted average of U's rating on I most similar items. Let P be the prediction on product p for user u. The prediction equation is as below:

$$P_{u,i} = \overline{r_u} + \frac{\sum_{n=1}^{k}(S* \hat{r}_{u,n})}{\sum_{n=1}^{k}(|S|)}$$

Where $\hat{r}_{u,n}$ is the normalized rating obtained after subtracting the user average for the rating. It is calculated as $\hat{r}_{u,n} = r_{u,n} - \bar{r}_u$.

Let I denote the set of all items in a catalog. Usually, the algorithms generate a item-item matrix or similar items table by finding the items which will are most likely to be purchased by a user (or customer). Using the following iterative algorithm, the similarity between single item and its products can be calculated:

For each i ∈ I in the catalog

    For each u ∈ U who purchased i

        For each item i' ∈ I purchased by u

            Store the information that u purchased i and i'

For each item i'

    Compute similarity between i and i'

In the worst case scenario, the space complexity is O(n^2) and the time complexity for the algorithm is O(n^2m) for any offline computation which is performed, where m is the number of vector which corresponds to a product (or item).

3. Memory based algorithms and Extensions

Memory based collaborative filtering algorithms [21] use the information in the entire user database to make predictions. The votes for the active user 'a' are predicted using the set of weights calculated from the database of users along with the user details. Let the set of votes from a user u on item (or product) i be denoted by $v_{u,i}$ and let I_u represent the set of items which user u has already voted on. The mean vote ($\bar{v}_u$) can be represented as:

$$\bar{v}_u = \frac{\sum_{i \in I_u} v_{u,i}}{|I_u|}$$

Let $p_{a,i}$ (predicted vote for the active user [21]) be the weighted sum of the votes for the other users, K denote normalizing factor, n be the number of users with finite non-zero weight. $p_{a,i}$ can be computed using:

$$p_{a,i} = \bar{v}_a + K \sum_{x=1}^{n} w(a,x)\left(v_{x,i}\right) - \bar{v}_x$$

Here, w(a, x) denotes the similarity, correlation or distance between the active user and user x. The algorithms can be distinguished using the details of the weight calculation.

3.1 Correlation algorithms

Pearson correlation [3, 4, 5, 19, 21] is used to compare the linear dependence between two variables. The formulation of statistical collaborative filtering was done in GroupLens [10, 21] and it was used in many memory based algorithms. The sequence of operations performed is as shown below:

1. Check if the user has rated an item.

2. Count the comparisons which are required to find n nearest neighbors.

3. Count the ratings that have been aggregated when the final prediction is being made.

In the context of recommender systems, it is used to compare the ratings of the items which is rated by a reader or a user (u) and the number of neighbors (n).

$$\text{Similarity (u, n)} = \frac{\sum_i (r_{u,i} - \bar{r}_u)\ \sum_i (r_{n,i} - \bar{r}_n)}{\sqrt{\sum_i (r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_i (r_{n,i} - \bar{r}_n)^2}}$$

The predicted rating for u over item j, where j has been rated by both u and n can be calculated as follows:

$$p_{u,j} = \bar{r}_u + \frac{\sum_{i=1}^{n} w\,(u,i)(r_{i,j} - \bar{r}_i)}{\sum_{i=1}^{n} w\,(u,i)}$$

The above two equations allow incremental computations (multiple passes are not done over co-rated items) and hence it takes a maximum of n steps to calculate the mean. If there are m active users who have rated an item j, we require nm comparisons to find the n neighbors in the worst case scenario. The average rating of j over n takes n steps using the prediction formula. The final cost is mn+m+n. The time complexity is O(n) and the space complexity to store the rating matrix is in the order O(mn).

Default voting [27] is an extension to the correlation algorithm explained above. The correlation algorithms did not perform efficiently when there were lesser votes for a user, because they used votes which were voted by both the active user and the matching user. Using this observation, default value was assigned to the titles which did not have explicit votes [21]. Just because an item has not been rated by a user, it should not be considered as un-interesting. So, if neither the active or matching user has voted on some item, a default value for the vote can be assigned to it, which can positive or even be a

zero or negative value if the item has not been observed by anyone for a long time. When implicit voting scheme is employed by an application like a web page, a positive value for the observed vote (or the preference) is assigned. In cases like these, if a user has not visited the web page, the default vote of 0 can be assigned to it. If a user cannot match an active user on any item, the weights are not calculated for that person.

3.2 Vector Similarity algorithms

The vector similarity or cosine similarity [1, 3, 4, 5, 19, 21] is used to measure similarity using the cosine angle formed by the frequency vectors (For eg. NewsWeeder [22]). In memory based collaborative filtering algorithms, the similarity between two documents are measured using this technique. The document is considered as a vector of frequencies of words and the cosine angle between the two vectors of frequencies are calculated to determine similarity. The dimensionality of the vector is dependent on the average number of terms between two documents.

$I_{u,i}$ is the set of items which both the users rated positively. I_i is the item which is rated positively by the user i. Only positive ratings are considered for computing the cosine similarity. The formula is as below:

$$\text{w (u,i)} = \sum_{x \in I_{u,i}} \frac{v_{u,x}}{\sqrt{\sum y \in I_u {v^2}_{u,k}}} \frac{v_{i,x}}{\sqrt{\sum y \in I_i {v^2}_{i,k}}}$$

Using this, the predicted score for a user is computed as below:

$$I_{u,i} = \text{K} \sum_{x=1}^{n} w(u,x)\, c_{x,y}$$

Using the information above, the items which are similar to each other is determined and the closest matching documents are sorted and returned back to the user. This sorting increases space complexity, but the time complexity is linear. Since only the positive ratings are computed for cosine similarity in a sequential fashion, the algorithms has a time complexity of O(n) where n is the dimensionality of the vector and with respect to space, it is O(nm) where m is the total number of documents.

There are two more extensions to the memory based algorithms which is briefly discussed in [21]. They are explained below.

Case amplification technique [5, 21] amplifies the weights closer to 1 in order to emphasize the contribution of similar users to the predicted result. A transform is applied to the estimated weights [21] as shown below, where ρ is generally 2.5:

$$w'_{a,x} = \begin{cases} w'_{a,x} & \text{if } w_{a,x} \geq 0 \\ -(-w'^{\rho}_{a,x}) & \text{if } w_{a,x} < 0 \end{cases}$$

Inverse user frequency [21, 26] is another extension which was inspired by the TF-IDF scheme [26] and Information retrieval field. Considering the fact that most commonly used words do not pose to be of much use in finding the topic in a document their weights can be reduced. According to Inverse document frequency [21], even if some words have less frequency of occurrence in a document, they might be more useful in determining the topic in a document. Using inverse user frequency, we can apply a transformation to the votes in a database. The idea behind this is the fact that if a product is liked by everyone, it is not as useful as other common products in determining similarity between users.

If n_i denotes the total number of users n, who have voted for item i in a database, the value of f_i as $\log \frac{n}{n_i}$, where f_i takes a value of zero when all the users have voted on an item. The correlation weight from [21] in this case can be calculated as below:

$$w(a,x) = \frac{\sum_i f_i \sum_i f_i v_{a,i} v_{x,i} - (\sum_i f_i v_{a,i})(\sum_i f_i v_{x,i}))}{\sqrt{UV}}$$

$$\text{where } U = \sum_i f_i \left(\sum_i f_i v_{a,i}^2 - (\sum_i f_i v_{a,i})^2\right)$$

$$\text{and } V = \sum_i f_i \left(\sum_i f_i v_{x,i}^2 - (\sum_i f_i v_{x,i})^2\right)$$

Comparisons between the algorithms and recommenders are as shown below:

Recommender System	Type	Category	Algorithm	Type of data	Rating scale
GroupLens	Hybrid (Feature Augmentation)	News	Correlation	Online data	1 to 5
Tapestry	Collaborative	News, Email	Correlation	Online data	Variable
Ringo	Collaborative	Music	Correlation	Offline data	1 to 7
EntréeC	Hybrid (Cascade)	Cuisine/Food	Correlation, Vector Similarity	Offline data	Semantic Ratings
LIBRA	Hybrid (Feature Augmentation)	Books	Bayesian Learning algorithm	Offline data	1 to 10
NewsWeeder	Content Based	News	Vector Similarity	Offline data	1 to 5
BookCrossing	Collaborative	Books	Topic Diversification	Offline data	1 to 10
Jester 2.0	Collaborative	Jokes	Cluster	Offline data	Real-valued
Amazon	Collaborative	E-commerce	Item based collaborative filtering	Offline data	1 to 5

The space and time complexity of the algorithms are as described below:

Algorithms	Space Complexity	Time Complexity
Topic Diversification	O(n^2)	O(n^2)
Bayesian networks	O(n)	O(nm)
Clustering	O(n^2)	O(n)
Correlation	O(nm)	O(n)
Vector Similarity	O(nm)	O(n)
Item to item collaborative filtering	O(n^2)	O(n^2m)

In general, the memory based algorithms are preferred by commercial recommenders due to the demand for accurate filters but model based algorithms like clustering algorithms are more efficient although not accurate. The correlation algorithms are preferred when there is sparse data due to their ability to produce accurate recommendations when compared to other model based methods operating on lesser data.

References:

[1] Marina Drosou and Evaggelia Pitoura. 2010. Search result diversification. *SIGMOD Rec.* 39, 1 (September 2010), 41-47.

[2] Georgia Koutrika and Yannis Ioannidis. 2005. Personalized Queries under a Generalized Preference Model. In *Proceedings of the 21st International Conference on Data Engineering* (ICDE '05). IEEE Computer Society, Washington, DC, USA, 841-852.

[3] Asela Gunawardana and Guy Shani. 2009. A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. Journal of Machine Learning. Volume 10 (December 2009), pp. 2935 – 2962.

[4] Cai-Nicolas Ziegler, Sean M. McNee, Joseph A. Konstan, Georg Lausen: Improving recommendation lists through topic diversification. WWW 2005: 22-32.

[5] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. *IEEE Trans. on Knowl. and Data Eng.* 17, 6 (June 2005), 734-749.

[6] Tran, T. and Cohen, R.: 2000, 'Hybrid Recommender Systems for Electronic Commerce'. In *Knowledge-Based Electronic Markets, Papers from the AAAI Workshop*, AAAI Technical Report WS-00-04. pp. 78-83. Menlo Park, CA: AAAI Press.

[7] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.* 22, 1 (January 2004), 5-53.

[8] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12, 4 (November 2002), 331-370.

[9] J. Ben Schafer, Joseph Konstan, and John Riedi. 1999. Recommender systems in e-commerce. In *Proceedings of the 1st ACM conference on Electronic commerce* (EC '99). ACM, New York, NY, USA, 158-166.

[10] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work* (CSCW '94). ACM, New York, NY, USA, 175-186.

[11] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Commun. ACM* 35, 12 (December 1992), 61-70.

[12] Yuying Jiao and Gaohui Cao. 2007. A Collaborative Tagging System for Personalized Recommendation in B2C Electronic Commerce. International Conference *on* In Wireless Communications, Networking and Mobile Computing, 2007. Shanghai (September 2007), pp. 3609 – 3612.

[13] H. Nguyen and P. Haddawy. Diva: Applying decision theory to collaborative filtering. Papers from The AAAI Workshop on Artificial Intelligence for Electronic Commerce WS-99-01, AAAI Press, Menlo Park, California, 1999.

[14] Upendra Shardanand and Pattie Maes. 1995. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems* (CHI '95), Irvin R. Katz, Robert Mack, Linn Marks, Mary Beth Rosson, and Jakob Nielsen (Eds.). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 210-217.

[15] Robert H. Guttman, Alexandros G. Moukas, and Pattie Maes. 1998. Agent-mediated electronic commerce: a survey. *Knowl. Eng. Rev.* 13, 2 (July 1998), 147-159.

[16] Raymond J. Mooney and Loriene Roy. 2000. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries* (DL '00). ACM, New York, NY, USA, 195-204.

[17] Mary Elaine Califf and Raymond J. Mooney. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence* (AAAI '99/IAAI '99). American Association for Artificial Intelligence, Menlo Park, CA, USA, 328-334.

[18] Un Yong Nahm and Raymond J. Mooney. 2000. A Mutually Beneficial Integration of Data Mining and Information Extraction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press 627-632.

[19] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, Hiroshi G. Okuno. 2006. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *Proceedings of the 7th International Conference on Music Information Retrieval (ISMIR),* pp. 296 – 301.

[20] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing* 7, 1 (January 2003), 76-80.

[21] John S. Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. *Proc of 14th Conference of Uncertainty in Artificial Intelligence*, Madison, WI, pp 43-52.

[22] K. Lang. "Newsweeder: Learning to filter netnews", *Proceedings of International Conference on Machine Learning*, pp. 331-339, July 1995.

[23] Dhruv Gupta, Mark Digiovanni, Hiro Narita, and Ken Goldberg. 1999. Jester 2.0: evaluation of an new linear time collaborative filtering algorithm. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR '99). ACM, New York, NY, USA, 291-292.

[24] Dhruv Gupta, Mark Digiovanni, Hiro Narita, and Ken Goldberg. 1999. Jester 2.0: collaborative filtering to retrieve jokes. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (SIGIR '99). ACM, New York, NY, USA, 333-334.

[25] Ghazanfar, M. and Prugel-Bennett, A. (2010) Building Switching Hybrid Recommender System Using Machine Learning Classifiers and Collaborative Filtering. IAENG International Journal of Computer Science, 37 (3).

[26] Panagiotis Symeonidis, Alexandros Nanopoulos, and Yannis Manolopoulos. 2007. Feature-Weighted User Model for Recommender Systems. *In Proceedings of the 11th international conference on User Modeling (UM '07)*, Cristina Conati, Kathleen Mccoy, and Georgios Paliouras (Eds.). Springer-Verlag, Berlin, Heidelberg, 97-106.

[27] H.Zhang and D.Li, "Naive Bayes Text Classifier", *In the proceedings of the IEEE International Conference on Granular Computing*, 2007.