Indian Journal of Science and Technology, *Vol 9(27), DOI: 10.17485/ijst/2016/v9i27/89853, July 2016*
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Enhancing Recommendation using Ranking in Multidimensional Space

R. Suganya Devi*, A. P. Chitra and D. Manjula

Department of Computer Science and Engineering, Anna University, CEG Campus, Chennai - 600025, Tamil Nadu, India; suganyawaran2001@gmail.com, chitra3593@gmail.com, dmanju62@gmail.com

Abstract

Search engines play their vital part in building ranking algorithms. Product Recommendation systems is a business activity which involves ranking to fulfill customer needs among the competitors. In our work, similar queries are extracted using Memory based Collaborative Filtering (MCF) and those individual ranked lists are combined to produce single superior ranked lists using Top-k Event Scanning (TES) approach, a rank aggregation algorithm which employs B+ trees for indexing. Experimental results shows that the performance is achieved 90% more than the other existing methods.

Keywords: Indexing, Queries, Ranking, Recommendation, Similar, Top-K, Users

1. Introduction

In day-to-day life, people rely on recommendations from other people by spoken words, reference letters, news reports and media, general surveys, travel guides and so forth. Recommender systems help the people to find the most interesting and valuable information for them. In most researches, Collaborative Filtering (CF) is one of the most favored and popular approaches for prediction process. Collaborative filtering is divided into two main approaches namely memory based and model based collaborative filtering. Memory based CF use information about the user and the item. It mainly focuses on finding similarity between the users and the items and the user rating on items.

Model-based approach uses the available information about the users to learn a model for predicting unknown ratings. The fundamental assumption of CF is that if users X and Y rate n items similarly, or have similar behaviors like buying, watching, listening and hence will rate or act on other items similarly. MCF algorithms have the ability to deal with highly sparse data and to scale with the increasing numbers of users and items, to make satisfactory recommendations in a short period of time. Recommendation of product is mainly based on similar users buying the products which are based on ratings the products. With this similar user recommendation, we are going to consider the frequency of products that is searched in the engine per year.

When user enters a query (product), the search engine returns a set of results containing thousands of links to different websites, but it is impossible for the user to go over all of them and thus, the user opens only few of them to find the answers to their query. The rest are just useless for the user. So, the search engines must provide the best results at the top so that the user can get the answers to their query within those top results. Every search engine uses some criteria to rank the pages according to their importance with respect to the user's query. The proposed work aims to consider the similar users buying the products and the frequency of product searched in the engine. It mainly focuses on retrieving the top-k product list by aggregating the ranked list of the durable query per year. For example, Google zeitgeist 2015 shows the top searched keywords in the engine. Likewise, our proposed system retrieves the top product list of the year, where the datasets contains the user rating on cars and the car queries which is collected from the UCI machine learning repository. Hence, the proposed system consider the product as cars and list the top car lists of the year by

*Author for correspondence

aggregating the ranked list of the durable queries per year.

2. Recommender System

The research contains the understanding of collaborative filtering and context aware recommender system. Anaya[1] analyze the recommender system in collaborative environment using the influence diagram, and how Collaborative filtering is used in recommender system. It mainly give recommendations based on the past user behaviors. Gong[2] analyzes the collaborative technique and this can be divided into two main approaches namely memory based and collaborative filtering. Memory based collaborative filtering mainly focuses on finding the similarity between the users and the item and then finds the similarity of user rating on items. Model based collaborative filtering mainly focuses on finding the system model and their preferences. Meghana[3] proposed the concept of mining the user reviews from the social networks to know about the product or a particular topic. It is the process of analyzing and shortening the user generated content which is helpful to the public. Existing works are focused only on data extraction and classifying the content into positive and negative reviews or based on spam or non spam whereas it doesn't consider the relevance of reviews which is related or not.

Gedikli[4] analyze the CF recommendation accuracy based on the tag preferences in a social network like twitter, face book etc. and find the user preferences and rating based on the likes and tags and form the community for similar user preferences and recommend the users in the community based on the tastes. Reddy[5] analyze the concept of recommendation based on location based social networking system which adds the location as main dimension for personalized recommendation. This system takes the input as user profile information and location model from the social data. whereas recommendation system varies over time period and the preferences of the users also varies over time, Overall group preferences are obtained by giving equal weight to all the users due to considering the social connection as the major factor in which popular ones opinion are considered.Shi[6] provide an improved collaborative filtering recommendation method based on timestamp, and in this paper, authors mainly concentrates on the preferences of the user on items based on the time series and the threshold value should be maintained to improve the performance. Al-Shamri[7] uses power coefficient as a similarity measure for memory based collaborative filtering. Baltrunas[8] introduce Group recommendation with rank aggregation and collaborative filtering based on the ranking of the products with the main consideration of the ranking of the users on the products and conveyed the advantages of group recommendation over the personalized recommendations.

2.1 Rank Aggregation

Prati[9] analyses the rank aggregation concept by combining many features of ranking algorithms, whereas rank aggregation falls into two categories, score based rank aggregation and order based rank aggregation. In the first category, aggregation function uses the scores in the ranked list in order to create the final list. Whereas in second category, aggregation function uses the order information in the ranking list. Liu[10] analyse the concept of rank aggregation in supervised and unsupervised learning. In unsupervised learning approach no training data is utilized, methods like Borda count, Median rank aggregation, Genetic algorithm, Fuzzy logic based, Markov chain based aggregation and so are proposed. One exception is Borda fuse which makes use of training data. The main advantages of supervised approach is that make use of information available in existing labelled data.

Kang[11] proposed the concept of ranking method in web services by analyzing the user queries to the web. Exploring user behavior is identified by the relevant queries which are frequently searched by the users. Geetha Rani[12] mainly focused on mining users conceptual preferences from users click through data resulted from web search which uses ranking function to retrieve the search results. Ranking the results in the search engine according to the query is submitted by the user which mainly considers the user profiling as an primary component. This system generates the Concept-based User Profile (CUP) ranking algorithm for query terms relevance search results. CongleiShi[13] proposed the concept of ranking changes over the period of time and the time series database are maintained in the database to predict the final ranked list. Queries which are submitted to the web will always differs according to the time, he proposed the concept of median ranking concept from the full or partial list. Zhang[14] proposed the concept of borda count approach for rank aggregation. Borda count is an posi-

tional method for finding the winning scores in the top list, a multivalued object based on the ranking list of the voters. Ishii[15] proposed the concept of PageRank algorithm for web aggregation in an randomized distributions. PageRank algorithm makes use of links which are connected to each webs for ranking the websites, whereas google makes use of the concept of pagerank aggregation for the relavant retrieval according the queries. Pedronette[16] proposed the concept of Clustering (CS) based rank aggregation using the contextual information from the fully or partial ranked lists. Clustering based approach combines the various ranking list using an algorithm for the prediction of final lists. Kaur[17] proposed the new concept for rank aggregation using Genetic Algorithm (GA), it is an efficient approach for aggregation mainly to overcome the NP hard problem, and it is implemented by the five steps like Initialization, Selection, Cross-over, Mutation and Convergence. The algorithm stops working when the optimal lists is obtained, it overcomes the Markov chain method[18,19] and uses transition probability matrix for rank aggregation by computing MC1, MC2, MC3 and MC4 chains, based on the comparison of probability of transitions in four chains final aggregated list is generated.

3. Proposed System

The proposed system combines the similar user recommendation using Memory based Collaborative Filtering (MCF) which considers the user rating on products and the ranking of durable queries (cars) which are frequently searched in the engine for predicting the final top-k product (cars) list for the year.

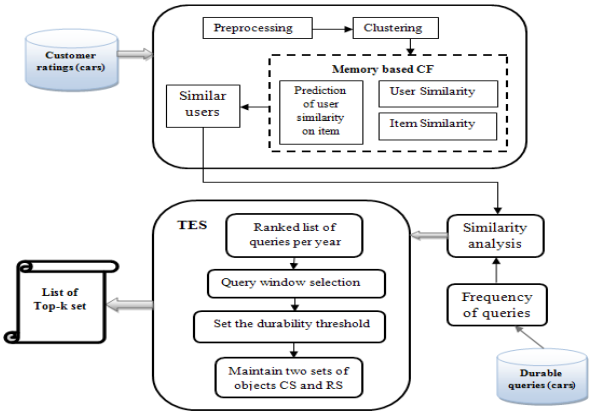

Figure 1. System architecture.

Ranking the durable queries per year is finally aggregated using the Top-k Event Scanning (TES) algorithm which maintains the two sets of objects Candidate Set (CS) and the Result Set (RS). Whenever the query q is found for the first time in the top-k sets, q is moved to CS. As soon as candidate query is found at the least minimum times in the top-k sets it is moved to RS. Finally the aggregated sets of queries are taken as a top-k retrieved products for the year. The system architecture for the proposed work is shown in the Figure 1. The following work describes the detailed description of the proposed system work flow and the algorithm and techniques used by memory based collaborative filtering and TES rank aggregation algorithm.

4. Collaborative Filtering

Collaborative Filtering (CF) method in recommender system is so far the earliest and most successful recommendation technology. It filters the information which is presented to the user by considering the information about the user preferences. It shows the good performance in using the user ratings and reviews to extract the preference information for recommendation to the user. Collaborative filtering is divided into two main methods memory based collaborative filtering and model based collaborative filtering. Memory based collaborative filtering consider the users rating, items ratings and users rating on item for the prediction of similar users preferences on items and the model based collaborative filtering uses the system information for recommendations.

4.1 Memory based Collaborative Filtering

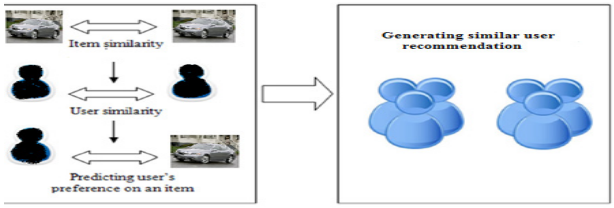

Figure 2. Workflow of memory based CF.

Complete work flow of an memory based Collaborative Filtering (MCF) is illustrated using the Figure 2. Left side of the Figure shows the similarity workflow which finally delivers the prediction of similar user's preference on an item. Basic concept of memory based collaborative filtering is to find the item similarity, user similarity and the

similarity of user on item, here item refers to the cars datasets, find the similarity between the cars based on the features and the similarity between the cars who are all having the same taste of purchasing the cars and then find the similarity of user rating on cars for recommendations. Memory based collaborative filtering is mainly used in group recommendation rather than personalized recommendation. The main advantages of memory based collaborative filtering is to scale with the co-rated items and the new data can be added easily and incrementally.

Cosine similarity is the most effective similarity measure for recommender system which is computed based on the Equation 4.1. We adopt cosine similarity to calculate the user-item preferences for the pair of users.

$$sim_{uc}(U_m, U_n) = \frac{u\vec{m} \cdot u\vec{n}}{\|u\vec{m}\| \cdot \|u\vec{n}\|} \quad (4.1)$$

Where sim_{uc} refer to similarity of user preference on cars. U_m *and* U_n refers to pair of users.

The MCF algorithm describes the prediction of similar user preferences on items based on memory based collaborative filtering which takes input as user-user similarity matrix and item-item similarity matrix by considering the ratings given to the product which is first preprocessed and clustered based on the top ratings given to the products. Whereas clustering uses k-means algorithm which generates centric value based on that clusters are formed and then construct the similarity matrix for prediction of similar user buying the same product.

Algorithm: Memory based CF.

Begin
Get target user id (tid)
Get t_pref= tid's item preference from user similarity
Get curr_sim u_{uc} of tid from t_pref
Get top k similar_users from similarity matrix
For each similar_users
Get their preferences of items from user-preference matrix
End for
Find union of all similar_user preferences
For each similar_user
Find #items from curr_sim u_{uc}
Find #items rated by the user from clustered datas

$$sim_{uc}(U_m, U_n) = \frac{u\vec{m} \cdot u\vec{n}}{\|u\vec{m}\| \cdot \|u\vec{n}\|}$$

End for
Sort the probabilities according to the preferences
For each unrated item
Read top n similar items from the similarity matrix
Display the results
End for
End

5. Ranking

The top-k query (product) is selected based on the ranking score assigned to each query. Whereas ranking score assigned is based on the frequency of searches in the search engine and the number users buying the product. We have to consider the durable queries for different time series. For example, we consider the durable queries for five years and then predict the top-k results by aggregating the results. Each year the ranking score for the query varies based on the ranking score we have assigned the value and then predict the top-k results for each year whereas the top-k results also varies for the different time series (years). In our system we consider the cars queries, for each year the rank of the car varies according to the number of searches and the number of users buying the cars.

The frequencies of durable queries are predicted using the Hash Map function it uses array in the background. Each element in the array is another data structure. The Hash Map uses a function on the key to determine where to place the key's value in the array. Hash Maps implements the interface <K,V> whereas hash map uses an inner class to store data: the Entry <K,V>.This entry is a simple key-value pair with two extra data: a reference to another Entry so that a Hash Map can store entries like singly linked lists, a hash value represents the value of the key. This hash value is stored to avoid the computation of the hash every time the Hash Map needs it.

When a user calls Map (K key, V value) or get (Object key), the function computes the index of the array in which the Entry should be. Then, the function iterates through the list to look for the Entry that has the same key (using the equals () function of the key). Then it returns the frequency of queries with the year by fixing year as an key value. After computing the frequency of queries per year, similarity measure is used to retrieve the ranked lists of queries per year, for that cosine similarity is used as an effective measure to find the similarity. Hence, the ranked lists for durable queries (product) are obtained based on the scoring values.

5.1 Rank Aggregation

Rank Aggregation is a method that is used to combine many different rank orderings on the same set of candidates, or alternatives, in order to get a better rank ordering basically used in the field of voting. The main goal of the

proposed method is to merge a number of query (product) ranked lists in order to build a single superior ranked results. The aggregated list must be the optimized list and for this purpose, TES algorithm is used in which queries are aggregated according to the two set of objects CS and RS, where the queries which are below the threshold values are eliminated in a final lists.

5.1.1 TES

Existing methods used for Top-k query rank aggregation are based on Borda count method, Kendall's tau method and Markov chain method. The existing methods perform rank aggregation in an iterative manner which is an efficient for retrieval of queries in a multidimensional space. To overcome the response time of queries and iterative rank change interval we propose the algorithm TES (Top-k Event Scanning) which provide better results by performing comparative analysis of various methods. The durability s of an object s is defined by the number of timestamps in (t^b, t^e) for which the object is in the top-k. Two sets of objects are maintained during the algorithm: a set CS of candidate objects that are not yet confirmed to be durable top-k results and a set RS of confirmed results. Whenever an object s is found for the first time in the top-k set, s is moved to CS if it is possible for s to make it to the top-k result, based on the number of remaining timestamps until t^e. As soon as a candidate object is found at least min times in the top-k set, it is moved to RS. If there are not enough timestamps for new objects to make it in the result and CS is empty, the algorithm terminates, before having to reach.

Algorithm: Top-k Event Scanning (TES).

Input $q=\{k,[t_b,t_e),r\}$

$\Delta_{min} \leftarrow [r.(t_e-t_b)]$, $t \leftarrow t_b$,CS$\leftarrow$0,RS$\leftarrow$0
while $t < t_e$ do
Retrieve from disk the snapshot top-k set S_t^* at time t
find the next timestamp t' $(t < t' < t_e)$ where $S_{t'}^* \neq S_t^*$; if no such t' exists, $t' \leftarrow t_e$
for each object $s \in S_t^*$ do
if $s \in CS \cup RS$ and $t \leq t_e - \Delta_{min}$ then
add s to CS with $\Delta_s = t' - t$
else if $s \in CS$ then add $t' - t$ to Δ_s
if $\Delta_s \geq \Delta_{min}$ then
delete s from CS and add s to RS
if $t > t_e - \Delta_{min}$ then
remove from CS every object s satisfying $\Delta_s < \Delta_{min} - t_e + t'$
if CS = 0 then break
$t \leftarrow t'$
return RS

Rank aggregation using TES is the most effective method for top-k query retrieval by using the B+ tree indexing which index and retrieves in very fast manner better than the other rank aggregation methods like Borda count and Markov method. Whereas B+ tree indexing stores all the queries in the leaf node which is very effective in indexing and retrieval of queries and it well manages the rank change interval in various years and providing the final result sets. The durability of the query (product) also varies in different time series, TES provide better results by managing the two sets of objects. Initially the two sets are assigned to zero and then it check the start and end time for each time series i.e. it set the query window in different timestamp with the durability threshold. Both the objects check their minimum threshold value before it confirms in the final sets and it is mandatory to satisfy the threshold values. Hence, TES is effective in aggregating the result and the following work describes the experimental evaluation for the proposed system.

6. Experimental Evaluation

We implemented all proposed algorithms in java using Eclipse IDE editor. The experiments were run on a window 7-64 bit with an Intel core i5 processor 2.50 GHZ and 4GB of RAM. We use two kinds of datasets:

- UCI (http://archive.ics.uci.edu/ml/machine-learning-databases/car/) datasets for Cars collected from the above link which contains the user ratings and reviews for each features of the cars like internal, external, fuel capacity, build, performance, fun, reliability, comfort and overall ratings of the cars given by the users, which is around 60k ratings of the users. It is used for finding the similar user preferences on cars.
- Queries (http://data.tc.gc.ca/extracts/vrdb_full_monthly.csv) for cars are collected from the above link which contains queries around 100 k which is searched in the engine with the corresponding year of search. It contains the collection of 20 years of car queries in year wise which is used for ranking and prediction of top-k queries. Where ranking is performed by frequency count of queries (cars) and the similar users buying the cars.

The proposed system is evaluated by changing the input parameters to the system. The accuracy percentage

of each cluster is calculated by using formula in Equation 4.2.

$$Accuracy = \left(\frac{Correctly\ predicted\ similar\ users}{total\ \#user\ preferences\ in\ the\ original\ datasets} \right) X\ 100 \quad (4.2)$$

The dataset used for evaluation contains around 10,000 unique records. Each record consists of a user_id and feature rating of cars by the user. From this data record, the item which is rated by the user can be found. The user rating of items is cut down to half and the user-user similarity matrix is generated. This data is fed into the algorithm and similar user preferences are predicted. Ten different users are randomly picked and their rating history on items is cut down to half.

Their original preference item history is stored separately for comparison. Each user's Id is given as input to the modelled system, and numbers of similar users are predicted as in Figure 3.These preferences are compared with the original user preference list of item history and the number of correctly predicted similar user preferences are noted down. Figure 3 depicts the relation between original user preference history and predicted user preferences.

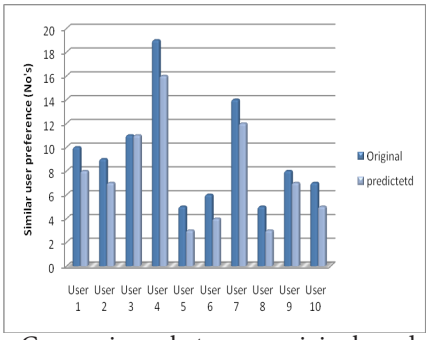

Figure 3. Comparison between original and predicted accuracy of similar user preferences.

The above graph shows that there is a high accuracy in predicting similar user preferences. Out of 10 original similar preference for user 1, 2 and 9 similar users is predicted correctly which suffices to 90.2% accuracy. Similarly percentage of accuracy for all the ten users is given in the Figure 4.

In the above experiment, similar user preferences are evaluated and the accuracy is calculated for each evaluation of results. The following evaluation shows the comparative results of performance of various rank aggregation algorithms. Whereas ranking results are evaluated using MAP in Equation 4.4 by considering the top 10 queries for five years. MAP measure is the mean of the average precision scores for each query. Where Average precision is expressed in Equation 4.3

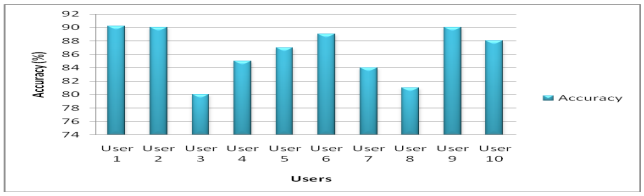

Figure 4. Percentage of accuracy for each user.

$$\text{Average Precision} = \frac{\text{Highest rank score in top k sets}}{\text{Total Rank score in top k}} \quad (4.3)$$

Where MAP is computed by the following formula in equation 4.4

$$\mathbf{MAP} = \frac{\sum_{q=1}^{Q} \mathbf{AveP(q)}}{Q} \quad (4.4)$$

Figure 5 shows the performance results of the top 10 queries per year. Map values are computed based on the score value obtained from the frequency occurrence of queries and the number of similar users for each query per year. Based on the similarity value obtained from the combination of both the functions, the queries are ranked. Instead of considering the whole queries we consider the top 10 queries which are ranked in an order for evaluation. From the results, we conclude that the average precision value for the top set queries increasingly varied per year. Whereas precision value for top queries varied periodically over the year which is shown below

Figure 5. MAP results on top 10 ranked queries.

Experimental methods for evaluating the ranked queries for each year is done using the mean average precision formula whereas rank aggregation methods are evaluated based on comparative analysis of various methods like Dynamic Adaptive algorithm, Brute force method, Borda

count method and TES algorithm. The main advantage of using TES algorithm is for effective retrieval of top-k queries within the timestamp intervals and very fast retrieval of queries by B+ tree indexing. Since, it retrieves the queries in time and it is very effective in multidimensional space for retrieval of queries. B+ tree is used because it stores all the queries in leaf nodes for easy retrieval and it maintains the rank change interval throughout the process of retrieval of top-k sets.

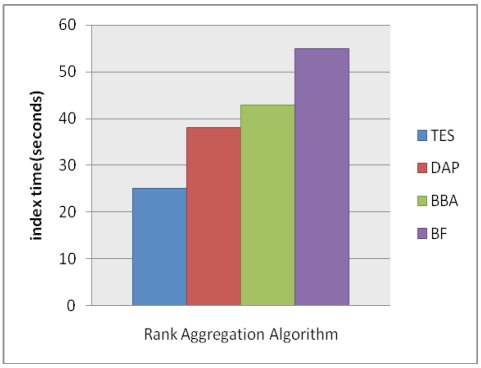

Figure 6. Comparison of various Rank aggregation methods in indexing.

Figure 6 clearly represents the indexing time taken by various algorithms and its time taken for query indexing. Above graph clearly depicts that TES algorithm with B+ tree indexing produce better results in query indexing and retrieval within 25 seconds. Brute force method performs indexing in sequential manner so it takes more time for indexing and retrieval when compared to TES. From the results we conclude that TES with B+ tree indexing is well suited for efficient indexing and retrieval in multidimensional query search engine.

7. Conclusion

Recommendation with ranking systems have become more common among the people and businesses, it altered the way people encounter the products of their interest and also other people. Recommender system makes use of data mining techniques and prediction techniques to surprise the users with suggestions of their interest. Memory based collaborative technique is used for finding the similarity between the users and the item and the user rating on item. Finding similar users based on ratings provide better recommendation. Based on the similar users and the product (query) that is searched in an engine the queries are ranked per year and finally the proposed system aggregate the ranked list using TES algorithm that produce the top-k lists of queries. The preliminary study suggested several interesting problems that were worth further exploring. Providing a framework to guide the selection and fusion of different features is one direction to work in the future.

8. References

1. Anaya AR, Luque M, Garcia-Saiz T. Recommender system in collaborative learning environment using an influence diagram. Expert Systems with Applications. 2013; 40(18):7193-202.
2. Gong SJ, Ye HW, Tan HS. Combining memory-based and model-based collaborative filtering in recommender system circuits. Pacific-Asia Conference on Circuits, Communication and Systems; 2009. p. 690-3.
3. Meghana RSJ, Subramaniyaswamy V. An effective approach to rank reviews based on relevance by weighting method. Indian Journal of Science and Technology. 2015; 8(12):1-7.
4. Gedikli F, Jannach D. Improving recommendation accuracy based on item-specific tag preferences. TIST. 2013; 4(1):11.
5. Reddy CA, Subramaniyaswamy V. An enhanced travel package recommendation system based on location dependent social data. Indian Journal of Science and Technology. 2015; 8(16):1-7.
6. Shi Y. An improved collaborative filtering recommendation method based on timestamp. 16th International Conference on Advanced Communication Technology; 2014. p. 784-8.
7. Al-Shamri MYH. Power coefficient as a similarity measure for memory-based collaborative recommender systems. Expert Systems with Applications. 2014; 41(13):5680–8.
8. Baltrunas L, Makcinskas T, Ricci F. Group recommendations with rank aggregation and collaborative filtering. Proceedings of the 4th ACM Conference on Recommender Systems; 2010. p. 119-26.
9. Prati RC. Combining feature ranking algorithms through rank aggregation. 2012 International Joint Conference on Neural Networks (IJCNN); 2012. p. 1-8.
10. Liu YT, Liu T-Y, Qin T, Ma ZM, Li H. Supervised rank aggregation. ACM Proceedings of the 16th International Conference on the WWW (WWW'07); USA. 2007. p. 481-90.
11. Kang G, Liu J, Tang M, Cao B. An effective web service ranking method via exploring user behavior. IEEE Transaction on Network and Service Management; 2015. 12(4):554-64.
12. Geetha Rani S. Sorana MM. A link-click-concept based ranking algorithm for ranking search results. Indian Journal of Science and Technology. 2014; 7(10):1712–9.
13. Shi C, Cui W, Lu S, Dittman K. Rank explorer: Visualization of ranking changes in large time series data. IEEE

Transactions on Visualization and Computer Graphics. 2012; 18(12):2669-78.
14. Zhang Y, Zhang W, Pei J, Lin X, Lin Q. Consensus-based ranking of multivalued objects: A generalized Borda Count approach. IEEE Transactions on Knowledge and Data Engineering. 2012; 26(1):83-96.
15. Ishii H, Tempo R, Bai E-W, Pei J. A web aggregation approach for distributed randomized page rank algorithms. IEEE Transactions on Automatic Control. 2012; 57(11):2703-17.
16. Pedronette DCG, Da Torres RS. Exploiting contextual information for rank aggregation. 18th International Conference on Image Processing; 2011. p. 97-100.
17. Kaur M, Kaurt P, Singh M. Rank aggregation using multi objective genetic algorithm. 1st International Conference on Next Generation Computing Technologies (NGCT); 2015. p. 836-40.
18. Wald R, Khoshgoftar TM, Dittman D, Awada W. An extensive comparison of feature ranking aggregation techniques in bioinformatics. 13th International Conference on Information Reuse and Integration (IRI); 2012. p. 377-84.
19. Hua Y, Shao J, Tian H, Zhao Z, Su F, Cai A. An output aggregation system for large scale cross-modal retrieval. IEEE International Conference on Multimedia and Expo Workshops (ICMEW); 2014. p. 1-6.