**ISSN** 1746-7659, England, UK
**Journal of Information and Computing Science**
Vol. 9, No. 3, 2014, *pp.* 181-188

# Linear Time Growth Collaborative Filtering Based on Caching Techniques

Waleed M. Al-Adrousy [1], Hesham A. Ali [2], and Taher T. Hamza [3]
[1]Computer Science Department, Computers and information sys Faculty,
Mansoura University, Egypt. Corresponding author Contacts:
waleed_m_m@mans.edu.eg , Fax (Work): +20502223754
[2]Computer Engineering and systems Department, Engineering Faculty,
Mansoura University, Egypt, h_arafat_ali@mans.edu.eg.
[3]Computer Science Department, Computers and information sys Faculty,
Mansoura University, Egypt, taher_hamza@yahoo.com

(*Received October 10, 2013, accepted March 8, 2014*)

**Abstract.** *One of social networks features is to recommend some items for users suitable and their profiles and needs. One of the used techniques of item recommendation is to calculate the average rating of other users' ratings of that item. The calculated average can be used as a rank for that item. New users' ratings are made within the lifetime of social networks. This dynamic sized set of ratings can lead to a performance problem. Ranking is not fixed since new users are constantly evaluating items in networks. However, updating ranking by traditional averaging can have a quadratic time growth. This paper aims to convert this aggregation method into linear growth instead of quadratic. The suggested algorithm needed extra storage capacity. Caching is suggested to speedup ranking with preserving storage resources. Another target is to have a high hit-ratio with better utilization of small cache size compared to the traditional Greedy- Dual-Size-Frequency algorithm. At the end of this paper an evaluation of the proposed technique performance is provided based on simulated experimental results. The experiment results show that the proposed technique has better performance rather than traditional averaging recommender systems without caching.*

**Keywords:** Social networks, collaborative filtering, recommender systems, caching techniques.

## 1. Introduction

One of the most important tools for people to communicate with each other is the Social Networks. According to the free on-line dictionary of computing (http://dictionary.reference.com/browse/social+network), a social network is “a Web site where one connects with those sharing personal or professional interests, place of origin, education at a particular school, etc.” [25]. One of users' interests is the need to ask for the best suitable service, product, person or even an article for some special purpose. Social networks have been used as a marketing way. However, not all advertisements are good as claimed or suitable to all users. So, the social networks relationships of friends have been used to guide users to select items suitable for them. This social network component is called a recommender system. Currently, there are two types of recommender systems, collaborative filtering and content-based filtering [7]. Collaborative filtering makes use of users' ratings to items, regardless of its content, while the content-based filtering does the opposite. Content-based filtering systems analyze the semantic content of profiles and pages in social network to suggest suitable items to users [5].

Recommender systems need to have a profile for each user [8]. User's profile can be static or dynamic [14]. A static profile contains user's basic information; preferences given by user explicitly such as preferences and selected friends. A dynamic profile is inferred by analyzing user's actions such as types of daily browsed pages. Both profile types have several modeling techniques, including simple databases and vector representation for user and his/her corresponding interests [7]. Vector Space Modeling (SVM) was the start of a newer technique that focuses on semantic which is named Latent Semantic Analysis [13]. Most recommender systems focus on dynamic profile because of its updated nature. Recommender systems can be evaluated according to accuracy and performance. Performance issues are the main point of interest in this

*Published by World Academic Press, World Academic Union*

research. For many Different techniques there is a common step, aggregation of several users' scores based on some criteria. The hosting servers' resources of social networks should be managed optimally to avoid wasting time or cost of hardware. One of the major resource consuming computations is the repeated ranking overtime for items in the social networks. The ranking operation depends on having an average rating for an item given by some users' ratings. The included users in the operation can be all users –in case of global rating, or a subset of users who represent the friends to a specific inquiring user. The second case is used to calculate the user based recommendation which tries to use user's profile to detect how suitable the recommended item can be [23, 9].

The contributions of this is paper can be represented in achieving two main goals: (i) suggesting an enhancement to averaging technique to have a linear order of growth, and; this will be done via a suggested incremental equation that works on a pre-calculated values that indicate the aggregation of the old non-changing ratings. Also (ii) suggesting an enhanced form of Greedy- Dual-Size-Frequency (GDSF) caching. The proposed algorithm aims to achieve better hit-ratio by using a median timestamp of items age in cache with no need for an artificial intelligence training phase. This paper is organized as follows; section 2 presents a background of research domain, section 3 presents the suggested techniques, section 4 presents the testing results, and finally the conclusion is in section 5.

## 2. Related Work

There are some common problems in researches about recommender systems [18]; prediction accuracy, testing opinions' subjectivity, recommended items suitability rather than being high ranked, effective preferences inference, trust formation of recommended items and usability of recommender systems interface layout. Those problems focus on the *value* of the recommended items rather than the *performance* of recommendation process itself. Recommender systems had many forms over the few past years. Some expert systems such as "Syskill & Webert" and Web Browser Intelligence (WBI) [8] applied intelligence to learn user preferences and patterns of recommendations and decision trees. "Syskill & Webert" system allowed user to make symbolic ratings about items using words like *cold* or *hot*. The symbolic ratings were converted later by the system into numeric ratings [12]. This technique aimed to handle the problem of difference in modeling nature between the human view and machine the view. However, this approach used explicit user likeness actions where user selects the rating for the item. Another technique in recommender systems was modeling of general interests was the analysis of user's navigation actions items pages. Knowledge based filtering was discussed in the work of Bruke et al. [4]. Their work classified knowledge into three areas: the social knowledge about users' database, the individual knowledge about a particular user and content knowledge about the items in the network. They discussed some problems in knowledge based filtering such as the distribution of users to items. In that problem, few items gain the attention of users while some other good items may not have been discovered.

Some techniques of natural language processing and semantic analysis have been used in some researches such as the work of Santos and Boticario [21]. In that research a new concept of Semantic Educational Recommender Systems "SERS" was introduced to join e-learning with semantic analysis. In [19] a survey is introduced to explore the features of some social networks that include e-learning support. One of those discussed networks is "OpenStudy" which was a large social learning community.

Semantic analysis can be complex, so, another technique was to use text mining techniques such as *term frequency* x *inverted document frequency* (TD-IDF) which was used in the work of Middleton et al. [15, 17]. This technique is simple which tries to use a set of keywords that distinguish a content category from another. Each item description is searched for those keywords and the tested to see how frequent each keyword is contained in description text. Of course, selecting the keywords set needs another set of techniques to update that set dynamically and also prevent term duplication. Term duplication can happen when many keywords are added by users but have the same meaning [24]. The focus in this research is on two of the several problems facing the collaborative filtering; *aggregation* used to predict rating for a user to a new item, and *caching* that can be used to enhance that king of computation. More discussion is presented in the following subsections. The content based filtering is beyond the scope of this study.

The first problem in collaborative filtering is the *aggregation* process. The most common case of rating aggregation is applied during averaging users' ratings. One variation of the averaging technique was

DOI https://doi.org/2024-JICS-22578 | Generated on 2025-04-09 07:03:27

discussed by Liu and Lee [10]. Their technique calculated the average rating per user, not per item. If there is a user *i*, that have rated $I_i$ set of items, then the user will have an average rating calculated by equation (eq.1).

$$\overline{V}_i = \frac{1}{|I_i|}\sum_{j\in I_i} V_{i,j} \tag{1}$$

Where $\overline{Vi}$ is the average rating for user *i*. The value of average user rating is used as an input to user similarity equation (eq. 2). This equation measures similarity between users *i* and *j*. This can be used to focus on most similar users to user *i* to include in the process of predicating ratings of user *i* for any item in the future.

$$w(a,i) = \frac{\sum_j (V_{a,j}-\overline{V}_a)(V_{j,i}-\overline{V}_i)}{\sqrt{\sum_j (V_{a,j}-\overline{V}_a)^2 \sum_j (V_{i,j}-\overline{V}_i)^2}} \tag{2}$$

Where *w(a,i)* is the similarity degree between users *a and i*. In fact this equation is called Pearson's correlation equation [10, 22] which is one of famous equations in data mining. However, Mobasher et al. had made an excellent survey about the attacks on social networks and recommendation systems including the *shilling attacks* [16]. One of those attacks is called *Push Attack*. In push attacks, the attacker tries to deceive the similarity based recommender system. The attacker mimics the target victim user known ratings. In that way, the recommender systems consider the attacker a very similar user to victim user. This gives the attacker the chance to give false ratings about items that victim hasn't yet rated or experienced. A suggested prevention technique for push attack is to calculate item overall ratings average (ranking) instead of depending on users' similarity [16].

The second problem is *caching* which aims to provide a memory copy of permanent object stored on a host like database server. This memory copy has a faster access time. The power of caching technique is to determine which objects are more important to be cached and which items are not. This is important since memory has a relative small capacity compared to permanent storage media. A suggested use of caching in recommender systems had been discussed by Ekstrand to store similar users to a target user [6]. A very good survey had been made about available caching techniques [2]. In that survey, some techniques used the intelligence methods such as neuro-fuzzy and neural networks systems. One of the systems that applied those methods is *Intelligent Client- side Web Caching Scheme Based on Least Recently Used Algorithm and Neuro-Fuzzy System* [1]. However, the artificial intelligence methods need a training phase to adjust internal weights before real testing. Greedy-Dual-Size (GDS) is one of the known techniques for caching prioritization without training phase. If we assumed that *C(p)* is the cost of fetching object *p* form host to cache and *S(p)* is the size of object *p*. Then GDS equation can be formed to calculate a priority *K(p)* as in equation (eq. 3).

$$K(p) = L + \frac{C(p)}{S(p)} \tag{3}$$

Where L is an aging factor to measure how old is the item in cache. An enhancement is made on GDS to consider frequency of use *F(p)* for object *p*. This enhancement is called *Greedy- Dual-Size-Frequency* (GDSF) which is formed in equation (eq. 4).

$$K(p) = L + F(p) * \frac{C(p)}{S(p)} \tag{4}$$

Some researchers suggested using some intelligence techniques to predict the future use for items in cache to increase a measure called Hit-Ratio [2]. Hit ratio can be calculated by dividing count of times that a needed item is found on cache over the total count of queries requests. The hit ratio is originally used as a search engine optimization measure. Similarly, in recommender systems with caching, this measure can be used to evaluate the utilization of caching capacity.

## 3. Methods and Techniques

### 3.1 Problem Formulation

DOI https://doi.org/2024-JICS-22578 | Generated on 2025-04-09 07:03:27

Social networks can be modeled as large graph *G(V,E)*, where *V* is the set of vertices (users) and *E* is the set of edges (relationships) between those vertices. Set *V* contain two types of vertices sets; set of users *U* and set of items *I*. Set *E* mainly have three possible types of edges; user-to-user relationships, user-to-item preferences and item-to-item dependency. The third type of edges may be ignored sometimes due to the complexity of its analysis and the large storage capacity that it needs. This kind of graph can be called sometimes "*bipartite*" graph since it contains two types of nodes [3]. A generalization of that kind of graph is called "k-partite" graph which includes *k* types of nodes [11]. The focus in this research is on the bipartite networks. Suppose that the function *rating (a,b)* means that user *a* has rating item *b*, and the function *path(c,d)* means that there is a bidirectional path between users *c* and *d* in the social network *G*. If there is a set of users Z such that any user *z* that belongs to *Z* has a direct or indirect relation to user y such that *path(z,y)* exits and there *rating(z,x)* exists also, then the target of the recommender system is to predict how much *rating*(*y*,*x*) can be in case that direct rating does exit so far. When the set Z equal set U, then the predicted *rating(y,x)* can be called *rank(x)* which is unified over all the social network *G*.

This research questions are: Can the aggregation of ratings be computed in a linear time? And Can the Caching techniques be enhanced to fit the suggested collaborative filtering technique? This research contribution is suggesting a simple equation to calculate the rank for an item in a linear time instead of quadratic. Another contribution is suggesting a new caching policy equation to increase hit ratio when applying the linear ranking equation with no need for a training phase needed in some other artificial intelligence methods.

### 3.2 Suggested Technique

Recommender systems mainly focus on the user-to-item preferences and sometimes user-to-user relationships. Several techniques use average calculation of a set of users' ratings for a target item. The users' set can contain all users or only a subset of neighbors, friends or similar users to the querying user. Some techniques calculate a weighted average, which multiply each individual rating by strength of relation factor [6, 20]. All those variations are based on a general simple equation of average described in equation (eq.5).

$$prediction(u,i) = \frac{\sum_{j \in N} factor(u,j) * rate(j,i)}{|N|} \tag{5}$$

Where *prediction (u,i)* is the predicted rating for user *u* to item *i*, *factor (u,j)* is a weighting factor that can have the value of 1, *N* is the set of users to aggregate, and *rate(j,i)* is the rating of user *j* to item *i* . For simplicity we can shorthand this form into equation (eq. 6).

$$prediction = \frac{\sum ratings}{|count\_of\_ratings|} \tag{6}$$

In fact, the equation is correct and simple but has a performance shortage. This equation requires that all ratings are already given and doesn't take into account the changing nature of social networks. In the real life, a new set of users can add, modify, or remove their ratings for any item at any moment. If the average rating is always updated by users' actions over time, then a quadratic growth of computation is resulted. If some batching techniques are used to buffer new ratings instead of instant updates, then some old predictions values may mislead users. In this research, a modification is suggested for equation (eq. 6) to be in the form of equation (eq. 7).

$$\text{NewAverage} = \text{OldAverage} * \text{OldCountOfItems} + \frac{\text{NewRating}}{\text{NewCountOfItems}} \tag{7}$$

This equation gives the same results always as equation (eq. 6). It has been tested on 20000 variable sized sets and always gives the correct average. If the value of *NewCountOfItems* is 1, this means that the rated item has never been rated before this new user. Equation (eq. 7) has a linear time growth compared to equation (eq. 6) in the instant updating. This equation can be used also to restore ratings to its original state in case of rollback operations performed after transactions failures.

The problem in equation (eq. 7) is the need to store old average value for target items. This is an extra overhead of storage. In fact, not all items are frequently updated. Some items are rarely accessed. So, the recommender system shouldn't waste the storage capacity by storing all current averages for all items. We suggest using a prioritization policy similar to caching techniques described previously in section 2. To enhance the GDSF technique that's presented in equation (eq. 4), we suggest a modification in equation (eq. 8).

$$K(p) = L + M(p) * \frac{C(p)}{S(p)} \tag{8}$$

Where *M(p)* is the frequency of use above the median timestamp of caching age for the item *p*. Although this is a simple modification in equation (eq. 4), it has a remarkable effect. It enhances the utilization of limited cache size compared to GDSF. In the next section we shall explore the results that confirm that. The Main Process of the suggested technique is summarized in figure 1. When new ratings are added to the social networks about any item, then the suggested technique tries to fetch the old ranking and count of raters to the target item from cache instead of re-calculating them. If those data are found in cache then the linear averaging in equation (eq. 7) can be applied, otherwise, the traditional averaging in equation (eq. 6) must be applied. In both cases, a new median of timestamps is calculated by equation (eq. 8) in order to decide which items can be in cache contents after updating ranking for any item. The suggested caching equation (eq. 8) can increase hit ratio in the future ranking updating processes.

## 4. Solution of FNFIDEs

In this research, we have tested two measures; the effect of the suggested equation (eq. 7) on average calculation time, and the effect of caching policy in the suggested equation (eq. 8) in hit ratio and computation time.

For the first measure, which is the effect of equation (eq. 7), we have generated 20000 set samples with sizes varying from 1 to 20000 items per set. Both equations (eq. 6) and (eq. 7) are applied to calculate the average. In all cases the average values in both equations are the same. Figure 2 shows the time comparison between the two techniques.

The suggested equation has a linear growth over time while the traditional averaging equation has quadratic growth. This is because the traditional averaging equation iterates over both *time* and *users* on each calculation request and never uses the historical average values of unchanged old ratings. It re-calculates its results with each time a new user adds a new rating. The suggested incremental averaging handles those problems by using historical results of old users to target item.

For the second measure, that is the usage of the suggested caching technique in equation (eq. 8). The experiments are made on a standard dataset for a real online community called "MovieLens" (http://www.movielens.org/). This dataset is collected by GroupLens Research Project at the University of Minnesota (http://www.grouplens.org/). This dataset contains 943 users, 1682 items and 100000 ratings. In this dataset each rating has its timestamp. The ratings are first ordered by timestamp to simulate the time factor. Then we sent ratings one by one to both techniques; the suggested technique in equation (eq. 8) and GDSF. A great challenge is considered, which is using a very small size of cache to test the performance of both techniques in heavy load situations. The count of items in cache was considered as the logical size measure for simplicity. The experiment started with cache sizes that can store only 10 objects. Then we increased cache sizes slightly to be 20, 50 and 100 objects. Both *C(p)* and *S(p)* are assumed to be constants. L is the time passed so far for a cache time since its first storage in it. We also applied the experiments first of small set of requests (ratings) like 20000, 25000, 30000, 40000, 6000 and 100000 ratings requests. Figures 3, 4, 5 and 6 are the comparison results.

In fact, we confirm the used cache size is so small compared to thousands of requests in real life. However, in figure 6 for example, a hit ratio is achieved when using cache size 100 objects when sample size was 100000 requests. That means that the suggested technique achieves 25% hit ratio using a cache size 0.1% of all requests compared to almost 18% hit ratio by GDSF. Due to the hardware limitations of testing environment, we used small sample sizes for testing. Of course deciding the optimal size of cache is a good research point, but it's beyond the scope of study in this research. To conclude that testing, we've made another test of time to compare both GDSF and the suggested technique and the results are shown in figure 7.

From figure 7, we can say that both techniques have almost the same time performance with a slight delay of the suggested technique. This can be justified by noticing that there is an extra step in the suggested technique to compute the median timestamp and filter out all timestamps before that median. From all above results, the suggested system can get a performance advantage over the traditional averaging recommender systems. Even the overhead of storing extra data about ratings has been handled by applying a new suggested caching technique. The combination of both suggested equations (eq.7 and 8) represents an enhancement in current recommender systems performance with high utilization of small-sized caches.

DOI https://doi.org/2024-JICS-22578 | Generated on 2025-04-09 07:03:27

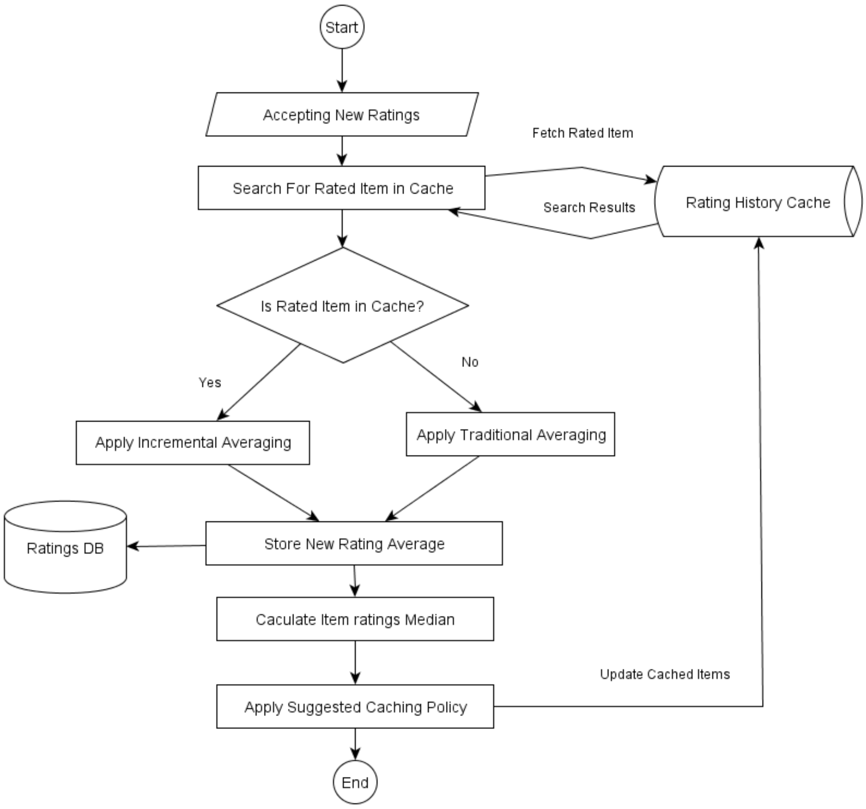

**Figure 1-Main Flow of The Suggested Recommender Systems**

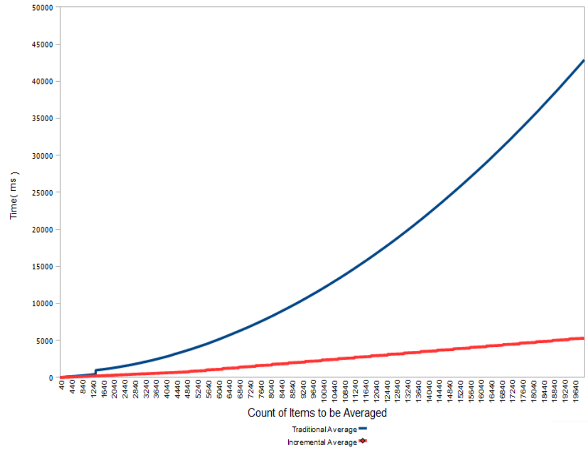

**Figure 2-Comparison between The Two Averaging Techniques in Time**

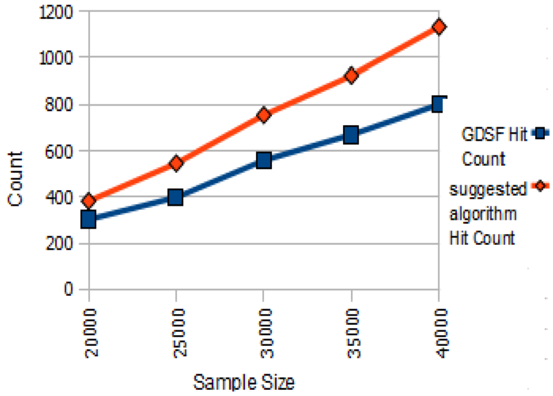

**Figure 3-Hit Counts at Cache Size 10**

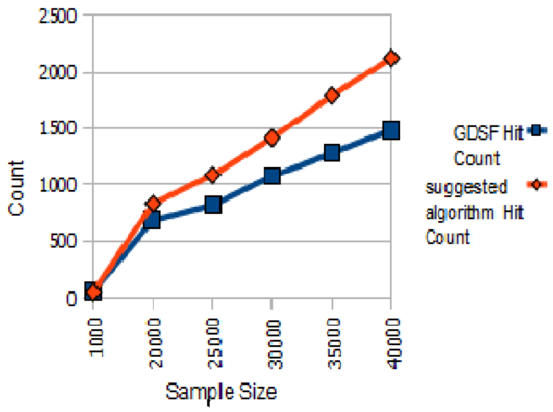

**Figure 4-Hit Counts at Cache Size 20**

DOI https://doi.org/2024-JICS-22578 | Generated on 2025-04-09 07:03:27

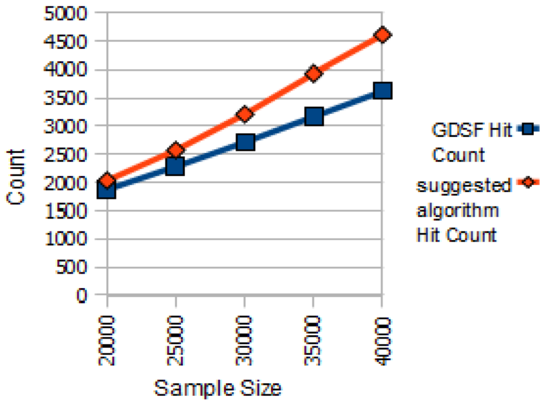

**Figure 5-Hit Counts at Cache Size 50**

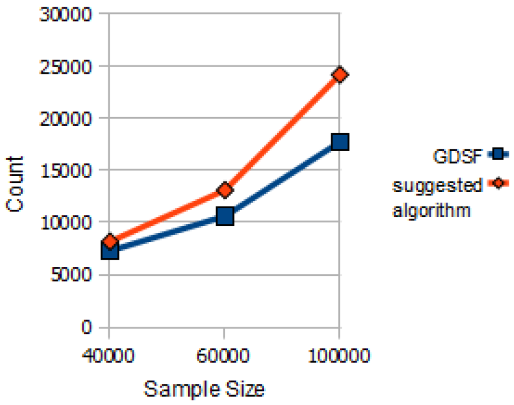

**Figure 6-Hit Counts at Cache Size 100**

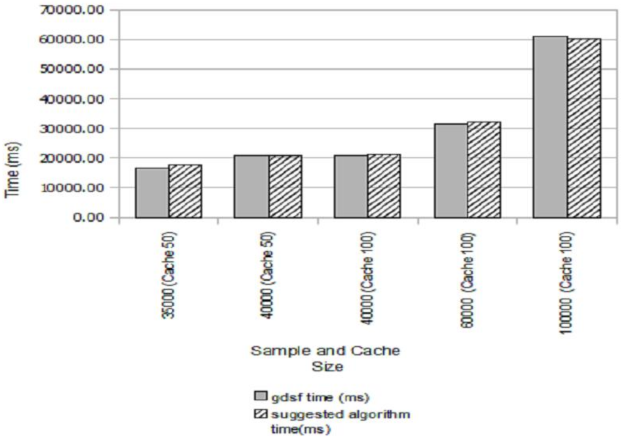

**Figure 7-Time of Computation Comparison between Suggested Caching Technique and GDSF**

## 5. Conclusion

In this research, we've studied a simple form of recommender system that can avoid push attacks by averaging a subset of users' ratings for an item. We've tried to study the time factor effect on the performance. We've presented two suggested equations; the first one is to enhance to averaging computation to be linear order of growth, and the second equation is to use caching policies to select most important items that need to cache their old averages. For the other items that aren't in cache, the traditional averaging is to be applied in that case. We could achieve a relatively high hit ratio with a very small cache size and nearly in the same time as the known GDSF technique. Our suggested caching technique – like GDSF- doesn't need an artificial intelligence training phase. Future points of research can be specifying the optimal size of cache size related to the frequency of operations. Another point of future research can be studying the effect of variable sizes of histories set from an item to another. The caching enhancement by semantic analysis of items and users profiles can be another possible research point. Another possible enhancement is to eliminate the need of cached data and preserve linear growth performance also.

## 6. References

[1] Ali, W. and Shamsuddin, S.: "Intelligent client-side web caching scheme based on least recently used algorithm and neuro-fuzzy system". Advances in Neural Networks, Springer-Verlag Berlin Heidelberg, 2009, 5552, (II), pp. 70–79.

[2] Ali, W., hamsuddin, S., and Ismail, A.: "A Survey of Web Caching and Prefetching". International Journal Advanced Soft Computing Applications, 2011, 3, (1), pp. 18-44.

[3] Asratian, A., Denley, T., and Hӓggkvist, R.: "Introduction to Bipartite Graphs" chapter in "Bipartite Graphs and their Applications", (Cambridge University press, 1998).

[4] Burke, R., Felfernig, A. and Gker, M.: "Recommender systems: An overview". AI Magazine, Association for the Advancement of Artificial Intelligence, 2011, 32, (3), pp. 13–18.

[5] Cai, X., Bain, M., Krzywicki, A. and Wobcke W.: "Collaborative filtering for people to people recommendation in social networks". Australian Joint Conference on Artificial Intelligence, AUS-AI, 2010, pp.476-485.

[6] Ekstrand, M.: “Collaborative Filtering Recommender Systems”. Foundations and Trends® in Human–Computer Interaction, 2010, 4 (2), pp. 81–173.

[7] Ghauth, K., and Abdullah, N.: "The Effect of Incorporating Good Learners' Ratings in e-Learning Content-based Recommender System", Educational Technology and Society, 2011, 14, (2), pp. 248–257.

[8] Gomah, A., Abdel-Rahman, S., Badr, A., and Farag, I.: "An Auto-Recommender Based Intelligent E-Learning System". International Journal of Computer Science and Network Security, 2011, 11, (1), pp. 67-70.

[9] Jøsangm A., Ismail, R. and Boyd, C.: "A survey of trust and reputation systems for online service provision". Decision support systems, 2007, 43, (2):pp. 618-644.

[10] Liu, F., and Lee, H.: "Use of social network information to enhance collaborative filtering performance". Expert Systems with Applications, 2010, 37, (7), pp. 4772-4778.

[11] Long, B., Wu, X., Zhang, Z., and Yu, P.: "Unsupervised Learning on K-partite Graphs". Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining. New York, USA, 2006, pp. 317-326.

[12] Lops, P., De Gemmis, M. and Semeraro, G.: "Content-based Recommender Systems: State of the Art and Trends", chapter in: *"Recommender Systems Handbook*" (Springer Science and Business Media, LLC), 2011, pp. 73–105.

[13] Luna, J., Huete, J., and Cano, J.: "User Intent Transition for Explicit Collaborative Search through Groups Recommendation". Proceedings of the 3rd international workshop on Collaborative information retrieval, ACM, New York, USA, 2011, pp. 23-28.

[14] Mezghani, M., Zayani, C., Amous, I., and Gargouri, F.: "A User Profile Modelling Using Social Annotations: A Survey". Proceedings of the 21st World Wide Web Conference, WWW 2012, April 2012, Lyon, France, pp. 969-976.

[15] Middleton, S., De Roure, D. and Shadbolt, N.: "Capturing knowledge of user preferences". Proceedings of the international conference on Knowledge capture - K-CAP, 2001, pp. 100-107.

[16] Mobasher, B., Burke R., Bhaumik, R., and Williams, C.: "Towards Trustworthy Recommender Systems: An Analysis of Attack Models and Algorithm Robustness". ACM Transactions on Internet Technology, 2007, 7, (2), Article No. 23.

[17] Pazzani, M., Muramatsu, J. and Billsus, D.: "Syskill & Webert: Identifying interesting web sites". AAAI Press, MIT Press, 1, 1996.

[18] Pu, P., Chen, L. and Hu, R.: "A user-centric evaluation framework for recommender systems". Proceedings of the ACM RecSys 2010 Workshop on User-Centric Evaluation of Recommender Systems and Their Interfaces (UCERSTI), Barcelona, Spain, Sep 30, 2010, i, pp. 14–21.

[19] Ram, A., Ai, H., Ram, P., Sahay, S.: "Open Social Learning Communities". International Conference on Web Intelligence, Mining and Semantics, WIMS-11, Sogndal, Norway, May 2011, Article No. 2.

[20] Ricci, F., Rokach, L. and Shapira, B.: "Introduction to Recommender Systems Handbook", chapter in: *"Recommender Systems Handbook*" (Boston, MA: Springer US), 2011, pp. 1–35.

[21] Santos, O., and Boticario, J.: "Requirements for Semantic Educational Recommender Systems in Formal E-Learning Scenarios". Open access Algorithms, 2011, 4, (2), pp. 131-154.

[22] Schafer, J., Frankowski, D., Herlocker, J. and Sen, S.: "Collaborative filtering recommender systems". The Adaptive Web, LNCS, Springer-Verlag Berlin Heidelberg, 2007, pp. 291–324.

[23] Su, X. and Khoshgoftaar, TM.: "A Survey of Collaborative Filtering Techniques". Advances in Artificial Intelligence, Hindawi Publishing Corporation. Volume 2009, Article ID 421425.

[24] Wu, H., Luk, R., Wong, K. and Kwok, K.: "Interpreting TF-IDF term weights as making relevance decisions". ACM Transactions on Information Systems, Jun, 2008, 26, (3), pp. 1–37.

[25] Zhou, X., Xu, Y., Li, Y., Josang, A. and Cox, C.: "The state-of-the-art in personalized recommender systems for social networking". Artificial Intelligence, 2012, 37, (2), pp. 119-132.