A scalable solution for personalized recommendations in large-scale social networks

Christos Sardianos
Harokopio University of Athens
Dept. of Informatics and Telematics
Omirou 9, Tavros
Tel: +302109549424
sardianos@hua.gr

Iraklis Varlamis
Harokopio University of Athens
Dept. of Informatics and Telematics
Omirou 9, Tavros
Tel: +302109549405
varlamis@hua.gr

ABSTRACT

The modern trend in many Web 2.0 applications is that users can interact with the applications in terms of social activity, for example they can express their trust for another user or another user's review etc. Bearing that in mind, creating recommendations for users could go one step further by reclaiming the social network information they share into these applications. This information can be exploited for creating better recommendations but also for optimizing the execution of existing algorithms in large scale datasets. In this paper, we introduce a scalable model for generating personalized user recommendations, which first uses the social information and divides the social graph into subgraphs and then applies Collaborative Filtering on the preferences information related to each subgraph. The results of our experiments show that our proposed model can perform faster than the traditional techniques while producing high level personalized recommendations. Moreover, it allows the parallel execution of the recommendation algorithm in each separate subgraph.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering*

General Terms

Algorithms, Measurement, Experimentation, Performance.

Keywords

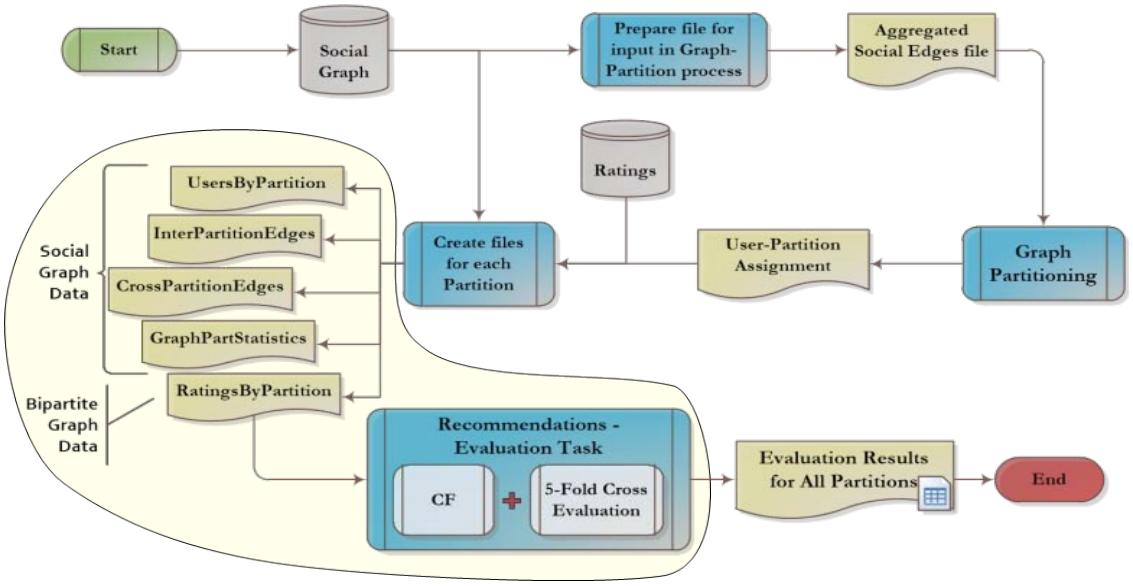
Recommender Systems, Social Networks, Content Personalization, Graph Partitioning, Collaborative Filtering.

1. INTRODUCTION

The increased popularity of social networking applications and the explosion in user provided content due to Web 2.0 applications has increased the interest to the analysis of social networks and the generation of a personalized user experience. One interesting area of research in this direction is Recommender Systems, which aims to predict the interests of the social networking application users and recommend items that may be of interest to them. In this environment, there is a large spectrum of applications, such as customer review sites, where users can express their opinions about an object (e.g. a product, a film, etc.) by providing a rating or a full-text review. What has recently changed in this is an overlay of social interaction between the users, which takes the form of mutual friendship, trust or distrust, and can be exploited to provide better recommendations [2]. This resulted in large social graphs and huge ratings datasets, which are difficult to process with existing techniques. Taking these into account, a first question that arises is if there is a way to combine the users' opinions and their social interaction information, so that the resulting recommendation can be more precise and personalized. The second question is whether it is possible to handle these large datasets in a meaningful way in order to get proper recommendations with good time performance. Unlike previous studies, in this paper we propose a model that takes into account both the social networking graph and the explicit opinions of users about items, and generates personalized recommendations by applying Collaborative Filtering (CF) techniques on subgraphs of the original graph. The subgraphs have been created based on the social information and are associated with the respective partitions of the ratings dataset. For the purpose of our study, we tested a large number of scenarios and performed the corresponding experiments using two popular datasets, Epinions and Flixster, which provide both social and rating information. The results show that partitioning the social graph and performing CF can perform faster and improve the performance of the existing CF approaches. In addition, with the proposed method we provide a solution to the Cold-Start problem that concerns the process of generating recommendations for new users.

2. RELATED WORK

With the abundance of information on the web and social media, recommender systems became critical tools for the end users. Recommender systems apply data mining techniques in the data they collect (user provided ratings, explicit user preferences, user activity data etc.) in order to generate personalized recommendations. Collaborative Filtering (CF) is a promising technique for generating personalized recommendations [6]. According to Melville and Sindhwani [3], Collaborative Filtering systems gather user preferences for objects in the form of ratings and search for similarities among these ratings, in order to find users with similar interests and this way



predict the ratings for the items the users haven't rated yet. Hence, they can recommend the items which are considered most likely preferred by the user.

Figure 1. Description of the model functionality

Over the past years, there has been a bulk of research interest regarding several Collaborative Filtering techniques and clustering methods. According to Pham et al. [5], a method that groups together users based on their social information can perform better than traditional techniques in generating recommendations. On the other side, O'Connor and Herlocker [4] proposed a Collaborative Filtering approach, which performs clustering on the graph of items and thus reduces the search space for CF algorithms. De Meo et al. [1] suggested a combination of Collaborative Filtering techniques [7] based on factorization tables and users' social friendships. Recently, Symeonidis et al. [8] implemented a multi-way spectral clustering approach, utilizing the top few eigenvectors and eigenvalues of the normalized Laplacian matrix to compute a multi-way partition of the data.

3. PROPOSED SOLUTION & DESIGN

In this section, we describe our proposed solution and the design of our approach, which tackles scalability issues of the recommendations generation process. Figure 1, illustrates the main components of our model as long as the information flow during the process of generating recommendations and the output files of the system.

More formally, we consider a social graph G_{social}, which contains edges between users and a bipartite graph $G_{bipartite}$, which contains user ratings for items:

$$G_{social} = \{V_u, E_s\}, \quad E_s: undirected, unweighted$$
$$G_{bipartite} = \{V_u, V_i, E\}, \quad E: directed, weighted \qquad (1)$$

The first step is the preprocessing of the social graph, so that it can be ready for the graph partitioning algorithm. Based on the partitions of the social graph, the partitions of the bipartite graph (ratings-graph) are created.

The result of this step is a set of **UsersByPartition** and **RatingsByPartition** files that contain all the *i* subgraphs of G_{social} and $G_{bipartite}$ respectively defined as:

$$G_{social_i}: \begin{cases} V_u = \cup_{i=1..N} V_{s_i} \\ \forall\, i,j, i \neq j \;\rightarrow\; V_{s_i} \cap V_{s_j} = \emptyset \end{cases} \qquad (2)$$

$$G_{bipartite_i}: V \in V_u \begin{cases} V \in V_{s_i} \\ e(user, item): e \in E, \rightarrow \\ user \in V_{s_i}, item \in V_i \end{cases}$$

$$G_{bipartite_i} = \left\{V_{s_i}, V_{i_{pointed\,V_{si}}}, E\right\}$$

Algorithm 1 summarizes the proposed graph partitioning.

1. **PartitionSocialGraph** (SG{V,E}, Cross, Subgraphs)
2. {SsG{V,E}} = **Partition**(SG)

```
3.      Cross = ∅
4.        For_each e in E
5.          Vi = e.from
6.          Vj = e.to
7.          SsGFrom = GetSubgrForVertex(Vi , {SsG{V,E}})
8.          SsGTo = GetSubgrForVertex(Vj , {SsG{V,E}})
9.          If (SsGFrom = SsGTo)
10.           SsGFrom = SsGFrom ∪ {e}
11.         Else
12.           InterCluster = InterClusterEdge(e,SsGFrom,SsGTo)
13.           Cross = Cross ∪ {InterCluster}
14.       End If
15.     End For_each
16.     Set Subgraphs = {SsG{V,E}}
17. End PartitionSocialGraph
18. PartitionRatingsGraph (BG{Vu,Vi,Er}, RatingSubgraphs)
19.   {BsG{Vu, ∅, ∅}} = PartitionUsers(Subgraphs)
20.   For_each e in Er
21.     Vu'=e.from
22.     BsG{ Vu',Vi', Er') = GetSubgrForVertex(Vu', Subgraphs)
23.     Vi' = Vi' ∪ e.to
24.     Er' = Er' ∪ {e}
25.   End For_each
26.   Set RatingSubgraphs = {BsG{Vu',Vi', Er'}}
27. End PartitionRatingsGraph
```

Algorithm 1. Descriptive Algorithm of Graph Partitioning

Lines 1-17 describe the social graph (SG) partitioning. Consequently, assuming social graph SG{V,E} with E{V_i,V_j}, for every edge E we get the partition that vertexes V_i and V_j belong to after the partitioning process and if V_i and V_j belong to the same partition then their social edge E is transferred to the set of edges of this partition. Otherwise, this edge is moved to a set of social edges that in fact connect different partitions, which we call Cross Partition Edges set.

Lines 18-27 describe the bipartite graph partitioning algorithm, where every item rating edge *e:*{V_u,V_i,E_r} in the bipartite graph BG, is assigned to the same subgraph with all item rating edges of user V_u and all users clustered together with V_u. The output of this process is a file with the ratings per partition, which is the input for the Recommendation-Evaluation component of our architecture. In this step we apply several CF algorithms to generate recommendations and perform a 5-fold cross validation task to evaluate performance.

4. IMPLEMENTATION

4.1 Software & Tools

The system was based on state of the art algorithms for Collaborative Filtering and Graph Partitioning glued together with java code. We used Metis[1] framework from Karypis Lab for the graph partitioning process and LensKit Recommender Toolkit[2] from GroupLens Research Team of the University of Minnesota for the recommendation generation process. Regarding Metis, our only concern was the creation of subgraphs of the same size, whereas in Lenskit we configured a script in Groovy language with parameters such as the delimiter, the precision and the scale of the ratings, the algorithms that were to be evaluated, the metrics, and some algorithm specific parameters such as Neighborhood Size for the k-Nearest Neighbors experiments, the Iterations and the Features that were to be used by the SVD algorithm and finally the ListSize for the TopNnDCG metric.

4.2 Evaluation Metrics

During our experimental procedure we evaluated a number of metrics that are widely used in all similar researches. One of the most commonly used metric is Mean Absolute Error (**MAE**), defined in equation (3).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|\,, \quad \begin{matrix} y_i : actual\ rating \\ \hat{y}_i : predicted\ rating \end{matrix} \qquad (3)$$

It is noteworthy that during implementation, the MAE metric was evaluated as MAE_ByRating (**Macro-Average**) and MAE_ByUser (**Micro-Average**).

Another common evaluation metric that was evaluated is Root Mean Square Error (**RMSE**), given in equation (4).

[1] http://glaros.dtc.umn.edu/gkhome/

[2] http://lenskit.grouplens.org/

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}, \quad \begin{matrix} y_i : actual\ rating \\ \hat{y}_i : predicted\ rating \end{matrix} \qquad (4)$$

RMSE metric was also evaluated as RMSE_ByRating (**Macro-Error**) as well as RMSE_ByUser (**Micro-Error**).

Apart from these metrics, we also evaluated normalized Discounted Cumulative Gain (**nDCG**) and normalized Discounted Cumulative Gain for Top-N items (**Top-N nDCG**). nDCG metric is defined in equation (5) and as its name suggests it is about gain, so its values range between [0.0 – 1.0].

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i - 1}}{log_2(i+1)}, \quad nDCG_k = \frac{DCG_k}{IDCG_k} \qquad (5)$$

Top-N nDCG metric is respective to nDCG but belongs to the category of metrics that are applicable to lists of recommendations, which we had to define prior to the experiments. In order to run the experiments, we had to specify the size of the item list (ListSize) that the metric had to be evaluated on.

5. Evaluation

5.1 Experimental Setup

To evaluate the performance of our proposed approach we used two datasets, available from University of Koblenz-Landau[3] (Oct. 2013) and Simon Fraser[4]. Both of the datasets contain social network information and rating information from users. The first is Epinions, one of the largest consumer product reviews site and the second is Flixster, one of the largest online movie reviews site. Epinions' dataset contains 131,828 unique users with 841,372 social relations and 13,668,320 ratings for 755,760 distinct items. Flixster dataset on the other hand, contains 786,936 users with 7,058,819 social relations and 8,196,077 ratings for 48,794 distinct items. These characteristics highlight the differences between the two datasets and are ideal cases to explore the effect of the dataset structure on our model for the different experimental scenarios, which we present in the next sections.

Table 1. Dataset Characteristics Comparison

		Dataset	
Characteristics		Epinions	Flixster
Social Graph	Num. of Distinct Users	131,828	786,936
	Num. of Social Edges	841,372	7,058,819
	Average Degree	12.765	17.94
Bipartite Graph	Num. of Distinct Users (Raters)	120,492	147,612
	Num. of Distinct Items	755,760	48,794
	Num. of Ratings	13,668,320	8,196,077
	Avg. outDegree/User	113.44	10.42
	Avg. inDegree/Item	18.09	167.97
	Evaluation scale	1 – 5	0.5 – 5
	Precision	1.0	0.5

We must notice that Flixster contains a big number of users but only less than 20% of them participate actively in the bipartite graph. This means that only their ratings influence the process of generating recommendations and the rest 80% are considered to be "Cold-Start" users.

All the scripts for the experiments were written in Java, except from the Recommendation-Evaluation script, which was written in Groovy. The platform that was used for the experiments was Okeanos, a IaaS service from EDET (National Research and Technology Network), and more specifically, there were used two Linux based 64-bit systems with 2-CPUs QEMU Virtual CPU Version 1.7.0 at 2.1GHz, 512KB cache and 6GB of RAM each.

For both of the datasets we performed a 5-Fold cross evaluation strategy and regarding to the algorithms that were examined, we evaluated User-User, Item-Item and an implementation of the SVD algorithm. At first, we run experiments on Epinions followed by experiments on Flixter to compare and contrast the results.

[3] http://konect.uni-koblenz.de/networks/
[4] http://www.cs.sfu.ca/~sja25/personal/datasets/

5.2 Effect of graph partitioning on the time complexity of recommendation algorithms

Since our approach is based on the partitioning of graphs, in the comparative evaluation we use the same recommendation generation methods applied to the original unpartitioned graphs as a baseline. In the first experiment we compare total time required for the whole process (i.e. collaborative filtering and cross fold evaluation) when applied to the unpartitioned graphs and when applied to the partitions, including the time required for the partitioning. We used exactly the same parameters in all setups: the number of features for the SVD algorithm was set to 100 and the listSize for the Top-N nDCG metric was set to 5. We tested a large number of experiments using a varying number of partitions (s) in the set of s= {1,2,4,8,16,33,65,125,250,500,1000}. In all experiments we used a single processing thread for all the partitions. Figure 2, shows the execution time of the evaluation process (in hours) for each algorithm and for various values of s. We have to mention that the horizontal axis is in base-ten logarithmic scale ($\log_{10}$).

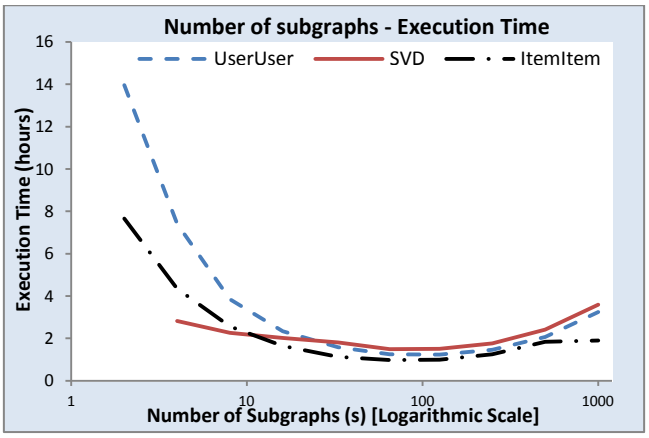


Figure 2. Execution time of the evaluation of different algorithms for different subgraph numbers (Epinions)

It is important to mention that the application of all CF algorithms on the unpartitioned graph was not possible due to memory and CPU constraints of our experimental setup. In addition, SVD algorithm was not possible to run for less than four subgraphs (s=4, ~30.123 users on average). The execution time for CF and evaluation in Epinions drops when the number of partitions increases, until the case of 65 partitions, where a maximum time performance is reached. For big values of s, the execution time increases again. However, the execution times can be quite lower, if multiple processing nodes are used in parallel, each one processing a different subgraph. Finally, as we observe from Figure 4, unlike User-User and SVD algorithms whose execution time has an upward trend above 1000 subgraphs, Item-Item appears to have a stabilized execution time at about 1 hour and 54 minutes (1h.54m).

As a conclusion from the results of these experiments we can say that using graph partitioning based on the social information graph significantly improves the execution time of Collaborative Filtering algorithms. This is very important, especially when we are able to execute Collaborative Filtering on separate subgraphs in parallel systems.

5.3 Effect of graph partitioning on algorithm performance

In the second experiment we examined the effect of graph partition on the performance of the three Collaborative Filtering algorithms. We tested all metrics described in Section 4.2. The results on the metrics that evaluate algorithm performance on the complete list of recommendations for each user of a partition (MAE, RMSE, nDCG) are almost stable for all the different numbers of s and, thus, are omitted. Results on the Top-N nDCG metric for a list of Top-5 items are depicted in Figure 3. According to the results of Figure 3, the SVD and Item-Item algorithms achieve almost ideal gain values (≅ 1). However, the performance of User-User algorithm is very low and gets worse as we go to big subgraph numbers. This means that SVD and Item-Item can be used on top of graph partitioning to produce high level of recommendations and in significantly lower execution time.

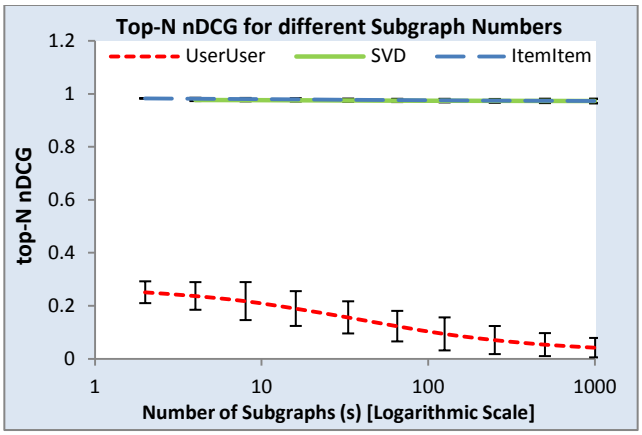


Figure 3. Top-N nDCG performance of all algorithms for different subgraph numbers in Epinions dataset

The evaluation of the User-User algorithm with the Top-N nDCG metric shows that it is not adequate for datasets, such as Epinions, that have very large number of items and in which partitioning leads to quite different ratings contained in each subgraph. One of the key facts that SVD is able to perform almost ideally in such datasets leveraging the benefits of our model, unlike User-User, is the mapping of item space into a smaller vector space, which is less sparse and therefore can produce suggestions even in this kind of datasets.

Comparing the performance of our model in the Epinions dataset to the results of the performance in the Flixster dataset (Figure 4), we notice that the results for the Flixster dataset are comparable to those for Epinions. The only difference is that in Flixster, the performance of User-User has a more stable trend as the number of subgraphs increases. It is also noteworthy that in the Flixster dataset due to graph's structure it was not possible to run the evaluation for less than eight subgraphs (s=8) and of course in the whole-unpartitioned graph as well.

As a conclusion of the above, the current experiments show that applying our proposed model for graph partitioning using the social graph for the bipartite partitioning, we can retain high quality of personalized recommendations, choosing the appropriate algorithm (except User-User in some cases, due to the reasons described in previous section), by minimizing the execution time.

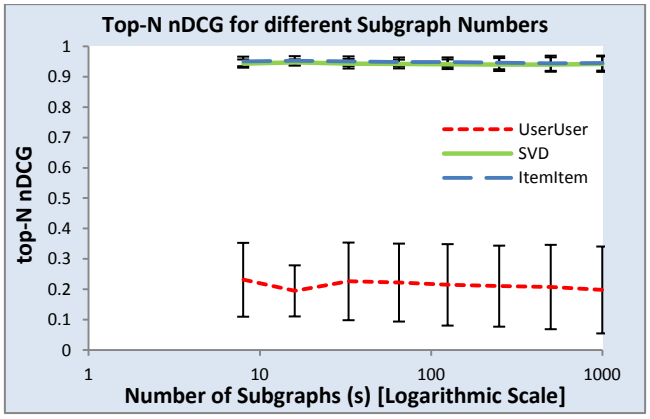


Figure 4. Top-N nDCG performance of all algorithms for different subgraph numbers in Flixster dataset

As mentioned earlier, in all of the experiments so far, we have used all the users of each subgraph as potential neighbors (Neighborhood Size k = number of users). In addition to these experiments we examined the effect of different k values on k Nearest Neighbors (knn) performance of our model. For this purpose, we tested User-User algorithm (as Item-Item and SVD are not affected by neighborhood size) for a set of subgraph values $s = \{4, 65, 1000\}$ and Neighborhood Size $k = \{1, 3, 5, 10, 25, 50, 100, 500, 1000\}$. The results from this set of experiments lead us to the conclusion that the performance of User-User algorithm for very small k values (k<<number of users) is even worse, than using the maximum possible k.

5.4 Algorithms comparison across datasets

An important effect of graph partitioning is that it affects the partitioning of item ratings and consequently the CF results. Assume two users that rate the same item in the original graph. After partitioning, the two users are assigned to different subgraphs and subsequently their ratings for this item are not available to each other anymore. In the case of datasets with few ratings per item this may result in ratings dispersion and may significantly deteriorate the quality of some collaborative filtering algorithms.

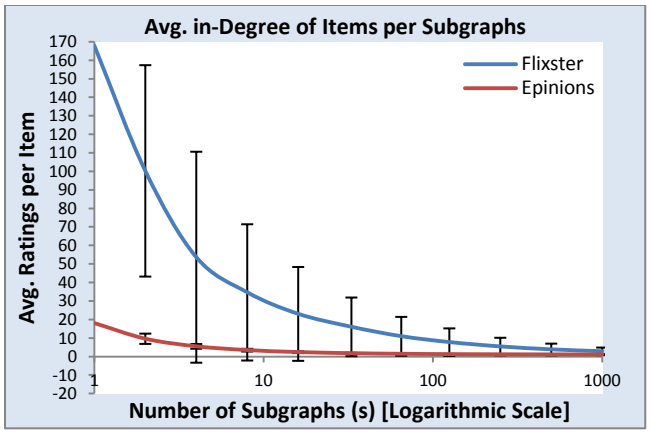


Figure 5. Average number of ratings per item (in-Degree) for different number of subgraphs (s), for the two datasets

As shown in Figure 5 and in comparison to the data in Table 1, Flixster has more users than Epinions but a comparable number of actual raters. However, the average number of ratings per item (item in-degree) in Flixster is significantly larger when few partitions are created and comparable to Epinions in the case of many partitions. In addition, as shown in Figure 6, there is a notable difference in rated items per subgraph in the case of few partitions, which is reduced as we increase the number of subgraphs created.

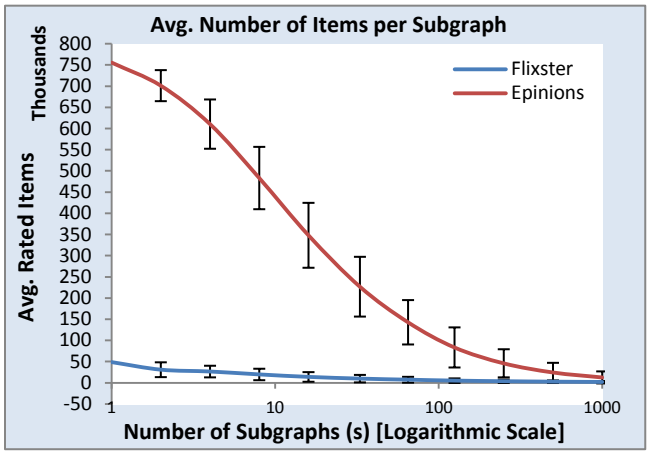


Figure 6. Average number of items for different number of subgraphs (s), for the two datasets

Although the performance of SVD is not affected, User-User has a significant reduction of performance in both datasets, as shown in Figure 7.

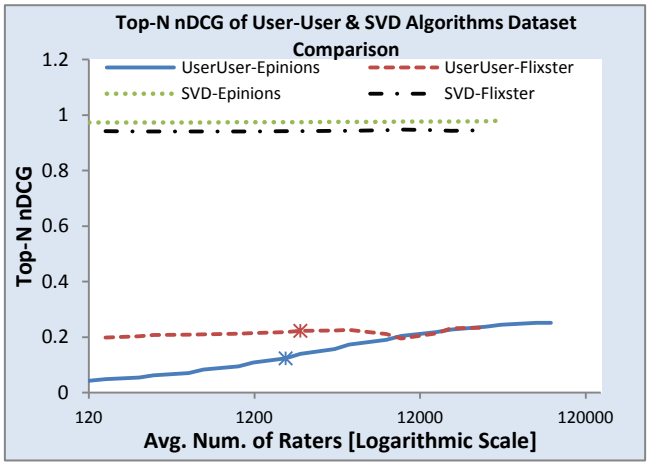


Figure 7. Top-N nDCG comparison of User-User & SVD algorithms with respect to the average number of Raters in each subgraph for the two datasets (where * denotes the point of 65 subgraphs)

As we notice in Figure 7, in the case of many subgraphs (small avg. number of raters) User-User in both datasets is not performing well, but it performs better in Flixster than in Epinions and has an almost stable trend. On the other hand, in the Epinions dataset, User-User performs inadequately when applied to many small subgraphs but has an upward trend when the number of users (raters) in the subgraph increases (fewer partitions/subgraphs created). The better performance in the Flixster dataset is justified by the fact that Epinions has a larger number of distinct items (large item space) (see Figure 6) and significantly smaller average number of ratings per item (item in-degree) (Figure 5) than Flixster. Although the prediction accuracy in both datasets is very low, when comparing the two, User-User can produce better suggestions in Flixster dataset, since it is more likely to have overlaps among user's similarity than in Epinions,.

6. CONCLUSIONS

Summarizing the results of this work, we can say that the approach and the model that is proposed in this research can perform even better than traditional techniques for producing recommendations. This mainly depends on the structure of the bipartite graph (whether this is a sparse bipartite graph or not) for which we try to produce recommendations and we can say that is the only drawback of this method. Yet, using algorithms such as SVD, we can produce high quality recommendations even in sparse graphs like that of Flixster.

A noteworthy conclusion is the reduction of the computation time of the recommendation process by performing Collaborative Filtering in segmented based on their social graph subgraphs. In this way, we can perform the recommendation process in a distributed computing system, thus minimizing the computation time whilst maintaining the same recommendation quality level.

Another important conclusion of this analysis is the ability to provide recommendations even for users, who don't have any reviews yet, also known as Cold-Start problem, by using the information of the users' social graph. This conclusion fully demonstrates the utility of exploiting social graphs in conjunction with ratings graphs to produce personalized recommendations for users, which verifies the main objective of this research.

7. FUTURE WORK

The experiments of the present study indicate the capability of scaling down the problem of generating recommendations for big graphs. In addition, results introduce a model of producing high quality recommendations by partitioning the initial big graph based on the users' social information and performing Collaborative Filtering over the subgraphs. Our future plans include the examination of performing the same experiments after removing the Cold-Start users from the beginning of the process, before even partitioning the graphs.

Further research is also needed to examine the influence of CrossCluster edge in the performance of our model, as well as applying algorithms like PageRank over the edges of every subgraph.

Finally, we intend to apply a Trust-Aware system like the one Eirinaki et al. [2] proposed on the edges of every subgraph to manage the trust over the social edges as well as testing the influence of higher N values (ex. N=10, N=20) of Top-N nDCG metric.

8. REFERENCES

[1] De Meo, P., Ferrara, E., Fiumara, G., & Provetti, A. (2011, November). Improving recommendation quality by merging collaborative filtering and social relationships. In Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on (pp. 587-592). IEEE.

[2] Eirinaki, M., Louta, M. D., & Varlamis, I. A Trust-Aware System for Personalized User Recommendations in Social Networks, in IEEE Transactions on Systems, Man, and Cybernetics: Systems (Part A) (TSMCA), (Volume:44 , Issue: 4, pp 409-241), 2013.

[3] Melville, P., & Sindhwani, V. (2010). Recommender systems. *Encyclopedia of machine learning*, 829-838.

[4] O'Connor, M., & Herlocker, J. (1999, August). Clustering items for collaborative filtering. In Proceedings of the ACM SIGIR workshop on recommender systems (Vol. 128). UC Berkeley.

[5] Pham, M. C., Cao, Y., Klamma, R., & Jarke, M. (2011). A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis. J. UCS, 17(4), 583-604.

[6] Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2001, April). Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.

[7] Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence, 2009*, 4.

[8] Symeonidis, P., Iakovidou, N., Mantas, N., & Manolopoulos, Y. (2013). From biological to social networks: Link prediction based on multi-way spectral clustering. Data & Knowledge Engineering, 87, 226-242