Recommender Systems in Light of Big Data

Khadija A. Almohsen, Huda Al-Jobori
Department of Information Technology, Ahlia University, Bahrain

Article Info

Article history:

Received Feb 13, 2015
Revised Jul 6, 2015
Accepted Jul 28, 2015

Keyword:

Apache Hadoop
Apache spark
Big data
Collaborative filtering
Recommender system
Singular value decomposition

ABSTRACT

The growth in the usage of the web, especially e-commerce website, has led to the development of recommender system (RS) which aims in personalizing the web content for each user and reducing the cognitive load of information on the user. However, as the world enters Big Data era and lives through the contemporary data explosion, the main goal of a RS becomes to provide millions of high quality recommendations in few seconds for the increasing number of users and items. One of the successful techniques of RSs is collaborative filtering (CF) which makes recommendations for users based on what other like-mind users had preferred. Despite its success, CF is facing some challenges posed by Big Data, such as: scalability, sparsity and cold start. As a consequence, new approaches of CF that overcome the existing problems have been studied such as Singular value decomposition (SVD). This paper surveys the literature of RSs and reviews the current state of RSs with the main concerns surrounding them due to Big Data. Furthermore, it investigates thoroughly SVD, one of the promising approaches expected to perform well in tackling Big Data challenges, and provides an implementation to it using some of the successful Big Data tools (i.e. Apache Hadoop and Spark). This implementation is intended to validate the applicability of, existing contributions to the field of, SVD-based RSs as well as validated the effectiveness of Hadoop and spark in developing large-scale systems. The implementation has been evaluated empirically by measuring mean absolute error which gave comparable results with other experiments conducted, previously by other researchers, on a relatively smaller data set and non-distributed environment. This proved the scalability of SVD-based RS and its applicability to Big Data.

Corresponding Author:

Khadija Atiya Almohsen,
Department of Information Technology,
Ahlia University,
Exhibitions Avenue, Manama, Bahrain
Email: kalmohsen@ahlia.edu.bh

1. INTRODUCTION

Advances in technology, the wide spread of its usage and the connectivity of everything to the Internet have made the world experience unusual rate of generating and storing data resulting in what is being called Big Data phenomenon. As a consequence, data is becoming unbelievably large in scale, scope, distribution and heterogeneity. To put it differently, Big Data is being characterized by 6Vs: Volume, Variety, Velocity, Veracity, Variability and Value [1]-[3].

As a consequence of the emerging fluid of data, normal tasks and activities become challenges. For instance, browsing the web and searching for interesting information or products is a routine and common task. However, the massive amount of data on the web is expanding the noise there making it harder and more time consuming to choose the interesting pieces of information from all this noise [4]-[5].

Likewise, the currently available systems, technologies and tools show their limitation in processing and managing this massive amount of data. This leads to the invention of new technologies, such as Map Reduce of Google, Hadoop of Yahoo! And Spark from University of California, Berkeley [6]. With this in mind, existing systems have been adapted to meet Big Data by using the newly invented tools and technologies. One of these systems is recommender system.

Recommender systems have been implemented long time ago by several Internet giants; like Amazon.com, Facebook and Google. These systems suggest new items that might be of interest to the user by analyzing user's profiles, their activities on the websites as well as their purchase history; if applicable. However, Big Data increases the cognitive load on the user, posing more challenges on recommender systems.One of these challenges is scalability in which the system should be able to deal with a bigger data set without degrading its performance. However, this is not the case with the current techniques of recommender systems as the computational time increase by increasing the number of users and items. Another challenge is to provide high quality recommendations, in a very quick manner, to gain their users satisfaction and retain them. The third challenge resulted from the sparseness of the data where each user had rated relatively small fraction of all the available items. This complicates the process of finding similarity between users as the number of commonly rated items is very small if not zero. Data Sparsity led another challenge called cold start problem in which the user does not get personalized recommendation unless s/he rates sufficient number of items [7]-[11].

This encourages more research work on new recommendation approaches that could solve the existing problems. One of the promising approaches is Singular Value Decomposition (SVD).

This research paper reviews the literature of recommender systems and provides a broad background of its different approaches. In addition, it studies the main concerns surrounding them due to big data. Furthermore, it investigates SVD approach and provides an implementation of it using Big Data tools (i.e. Apache Hadoop and Spark).

This work is intended to validate existing contributions to the field of SVD, assess the applicability of SVD to large scale recommender systems and evaluate the applicability and viability of Hadoop and Spark in building scalable system.

The reset of the paper is organized as follows: The next section provides a broad background of the theories related to RS and CF in particular. In addition, it sheds the light on applying SVD approach to RS. This will be followed by a section which details all the experiments undertaken using Apache Hadoop and Spark to implement SVD-based RS. It will also present the results of these experiments and discuss them. At the end, the conclusion and future work will be presented.

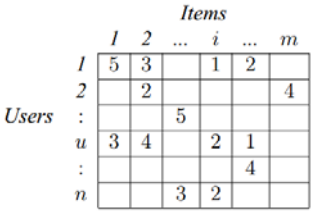
2. BACKGROUND

This section formulates the problem to be solved by RSs and provides a broad background of the different recommender algorithms and approaches, especially the contemporary one that suites scalable system. In addition, it addresses the preliminaries as well as the applicability of SVD to CF recommender systems. Furthermore, it reviews related word to give clear view of the state of the art.

2.1. Recommender System's Problem Formulation

Suppose that a Big Data set records the preferences of big number of users; denoted by n; for some or all of m items. The preference record usually takes the form of tuple (userID, itemID, rating); where rating takes a value on a numerical scale (for example from 1-5) and that expresses how much the user holding userID likes the item with itemID.

Let R be a user–item matrix of size m × n which represents the preference records such that each Rij cell either holds the rating given by user i to item j or null if the user did not rate the item yet, as shown in Figure 1. In most of the cases, this matrix is sparse because each user does not normally rate all the items in the data set.



		Items					
		1	2	...	i	...	m
	1	5	3		1	2	
	2		2				4
Users	:			5			
	u	3	4		2	1	
	:					4	
	n			3	2		

Figure 1. Sample of user-item matrix

The mission of a RS is to predict the missing ratings; i.e. predict how a user would rate an item in the future. This aids the recommender system in recommending items that are predicted to receive high rating by the user [12].

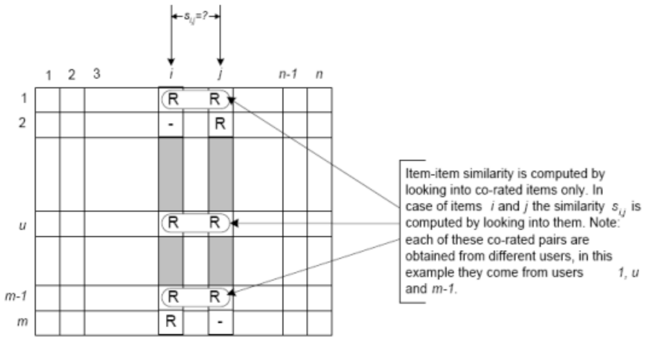
2.2. Recommender Systems and Approaches

Among the commonly used recommendation algorithms are content based recommender and collaborative filtering recommender. Content based systems analyze the user profile and his purchase history by studying the users' main attributes (also called meta-data such as: user age, gender and interest) and his previously purchased items' features (such as: price, category and description). This approach recommends items with similar attributes to the previously purchased one [8], [13]. The main problem of this approach is that it is domain specific; for example: in a movie recommendation, the system needs to consider actors and directors as attributes while making recommendation. However, such computation is not applicable for book recommendation [14]. The other approach, i.e. collaborative filtering (cf), makes recommendation based on the existing relationship between users and items. In general, it relies on other users' preferences to find items similar to what have been purchased by the user and suggest them as recommendation or to find like-minded users who have similar taste to the target user and thus recommend whatever they have purchased but not seen by the target user. [8], [14]. The two common approaches of CF are:

User-based Collaborative Filtering: it examines the entire data set of users and items to generate recommendations by identifying users that have similar interests to the target one and then recommends items that have been bought by others and not the target user. This proceeds by constructing user-item matrix which represents the interaction between users and items. After that, some statistical computations (i.e. similarity measures) will be applied on the matrix to find the nearest neighbors. These neighbors are supposed to have similar interest with the target user. This will be followed by combining the neighbors' preferences and finding the top N items that have been rated highly by neighbors and not by the target user. These N items will form the top N recommendations [8].

Despite the fact that this approach has been adapted widely, it suffers from scalability problem which was not considered a big issue few decades ago when the number of users and items was relatively small. However, as the data set size increases in big data era, computing the similarity between users is increasing exponentially because of the need for comparing each user with all the other users. Moreover, as the users interact with more items and change their preferences, the similarity needs to be recomputed; i.e. similarity pre-computation becomes useless. This is degrading the performance of RSs and that is why it is being considered as a big problem today. Furthermore, having a sparse user-item matrix, which is usually the case because users interact with relatively small set of items, also adds to the difficulty of computing user's similarity since the number of common items is relatively small if not zero [8]-[9], [14]-[15].

Item-based Collaborative Filtering: it examines the set of items rated by the target user and finds other items similar to them (which are called neighbors), by considering other users' preferences. With the hope of finding neighbors, each item will be represented by a vector of the ratings given by the different users, and then, the similarity of two items will be measured by computing the similarity between their vectors as shown in following figure.



Item-item similarity is computed by looking into co-rated items only. In case of items *i* and *j* the similarity $s_{i,j}$ is computed by looking into them. Note: each of these co-rated pairs are obtained from different users, in this example they come from users *1, u* and *m-1*.

Figure 2. Computing item-item similarity [16]

These neighbors will form the recommendations and will be ranked after predicting the preference of the target user for each one of them. The prediction $P_{u,i}$ of the target user u to one of the neighbors, item i, is given by:

$$P_{u,i} = \frac{\sum_{i=1}^{N} \mathrm{Sim}(i,j) \times R_{u,j}}{\sum_{i=1}^{N} \mathrm{Sim}(i,j)}$$

Where N is the number of neighbors, $\mathrm{sim}\ (i,j)$ is the similarity between the item j and its neighbor i, $R_{u,j}$ is the rating given by user u to item j [9], [17].

However, measuring the similarities between items takes long time and consumes lots of computer resources. This is the main pitfall of this method. Anyhow, changes in items are not as frequent as changes in users and, thus, such computations can be pre-calculated in an offline mode. Another strength of this algorithm is that it is not affected by having a sparse user-item matrix. This is because with large number of user, there will be enough number of ratings for each item which enable measuring the similarity between the different items and getting significant statistics [8]-[9], [15].

Generally speaking, CF whether it is an item-based or a user based approach, has a well-known strength in which it is not domain specific, and thus, does not rely on the items' properties and attributes. That is why it is applicable to different domains: movie recommendation, book recommendation, flowers, food and others. However, CF suffers from the following problems:

1) Scalability: RSs are being fed with massive amount of data which should be processed rapidly. However, CF algorithms computation time grows up with the continuous increase in the number of users and items [9].

2) Data Sparsity: In an e-commerce website, users usually rate small fraction of all the available items resulting in sparse data set. This degrades the accuracy of the RS because it complicates the process of finding similarities between users as the number of common items becomes relatively small [9].

3) Cold-start problems: This problem emerged as a consequence of data sparsity problem; where new users cannot get personalized recommendation unless they rate a sufficient number of items. Likewise, new items cannot be recommended before getting reasonable number or ratings [11].

4) Synonymy: different products have different names in the data set even if they are similar to each other. In this case, a standard CF RS will treat them differently and will not infer the hidden association between them. For illustration, "cartoon film" and "cartoon movie" are two phrases refereeing to the same item. However, ordinary implementations of CF algorithms had treated them differently [18].

5) Grey sheep: it addresses users whose opinions do not match with any other group of users. Consequently, CF cannot serve grey sheep since it mainly relies on the similarity between users' previous preferences [16].

The aforementioned, standard, implementation of item-based and user-based CF are following memory-based approach in which the entire data set is kept in memory while processing it and searching for similarities between users or items in order to make recommendation. The other approach of implementing CF algorithm is called model based approach in which the data set is used in an offline mode to generate a model by utilizing some data mining, machine learning or statistical techniques. This model could be used later on to predict the ratings for unseen items without the need of processing the entire data set again and again. Examples of this approach are: decision trees, clustering methods and matrix factorization models [19].

Point often overlooked is that model-based approach generates predictions with lower accuracy when compared with memory based approach. However, it has better scalability. Thus, many researchers are investigating their effort in studying and enhancing model-based CF. One of these algorithms is Singular value decomposition (SVD); which is the one implemented and validated in this work using some of Big Data Tools on a Big Data Set.

2.3. Singular Value Decomposition (SVD)

SVD is one of the famous matrix factorization techniques that decompose a matrix R of size $m \times n$ and rank = r into three matrices U, S and V as follows:

$$R = U.S.V^T$$

Where:

U: an orthonormal matrix of size $m \times r$ holding left singular vectors of R in its columns; i.e. its r columns hold eigenvectors of the r nonzero eigenvalues of RR^T.

S : a diagonal matrix of size $r \times r$ holding the singular values of R in its diagonal entries in decreasing order; i.e. $s_1 \geq s_2 \geq s_3 \geq \cdots \geq s_r$. These r values are the nonnegative square roots of eigenvalues of RR^T.

V: an orthonormal matrix of size $n \times r$ holding the right singular vectors of R in its columns; i.e. its r columns hold eigenvectors of the r nonzero eigenvalues of $R^T R$.

Furthermore, S could be reduced by taking the largest k singular values only and thus obtain S_k of size $k \times k$. Accordingly, U and V could be reduced by retaining the first k singular vectors and discarding the rest. In another word, U_k is generated by eliminating the last $r - k$ column of U and, similarly, V_k is generated by eliminating the last $r - k$ column of V. This will yield U_k of size $m \times k$ and V_k of size $n \times k$. As a consequence, $R_k = U_k . S_k . V_k^T$ and $R_k \approx R$, where R_k is the closest rank k approximation to R (9, 18,20). See Figure 3.

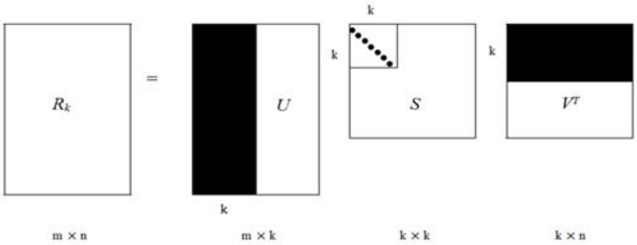


Figure 3. The reduced matrix R of rank k (20)

2.4. SVD-based Recommender Systems

Applying SVD to recommender systems assumes that the relationship between users and items as well as the similarity between users/items could be induced by some latent lower dimensional structure in the data. For illustration, the ratings given by a specific user to a particular movie, assuming that items are movies, depends on some implicit factors like the preference of that user across different movie genres. As a matter of fact, it treats users and items as unknown feature vectors to be learnt by applying SVD to user–item matrix and breaking it down into three smaller matrices: U, V and S [12]. This proceeds by constructing the, sparse, user-item matrix from the input data set and then imputing it by some values to fill the missing ratings and reduce its sparseness before computing its SVD. There are several imputation techniques and here are the most common one: impute by Zero, impute each column by its Item Average, impute each row by its User Average or impute each missing cell by the mean of User Average and item averag [21].

This will result in a filled matrix R_{filled} which could be normalized by subtracting the average rating of each user from its corresponding row resulting in R_{norm}. The last step is useful in offsetting the difference in rating scale between the different users [22].

At this point, SVD could be applied to R_{norm} to compute U_k (this holds users' features), S_k (holds the strength of the hidden features) and V_k (holds items' features) such that their inner product will give the closest rank-k approximation to R_{norm}. This lower-rank approximation of user-item matrix is better than the original one since SVD eliminate the noise in the user-item relationship by discarding the small singular values from S [18].

Henceforth, the preference of user i to item j could be predicted by the dot product of their corresponding features vectors; i.e., compute the dot product of the ith row of $(U_k . S_k)$ and jth column of V_k^T and add back the user average rating that was subtracted while normalizing R_{filled}. This could be expressed as:

$$p_{ij} = \overline{r_i} + (U_k . S_k)_{i,\_} \, . \, V^T_{\_j}$$

Where p_{ij} is the predicted rating for user i and item j, $\overline{r_i}$ is the user average rating, $V^T_{\_j}$ is the jth column of V^T and $(U_k . S_k)_{i,\_}$ is the ith row of the matrix resulting from multiplying U_k and S_k.

In point of fact, the dot product of two vectors measures the cosine similarity between them. Thus, the above formula could be interpreted as finding the similarity between user i and item j vectors and then adding the user average rating to predict the missing rating p_{ij}.

2.5. SVD Approach in Research

Sarwar, Karypis, Konstan and Riedl had studied the applicability of SVD to the field of RS by conducted two experiments on relatively small data set. In the first experiment, they did several preprocessing steps on user-item ratings matrix before finding its SVD decomposition and predicting missing users' preferences for items not seen yet. For the purpose of predicting user i preference for item j, they multiplied row i of $\mathrm{U}\,.\sqrt{\mathrm{S_k}}$ by column j of $\sqrt{\mathrm{S_k}}.\mathrm{V}^{\mathrm{t}}$. In the other experiment, they relied on SVD, instead of Pearson correlation and cosine similarity, to elicit the relationship between users and thus find user's neighbors needed to suggest top N recommendation. Their results were encouraging for the application of SVD in the field of RSs because they believed that reducing the dimensionality of the ratings matrix succeed in filtering out the noise from the data [18]. After two years, the same group of researchers proposed an incremental implementation of their previous work as a solution for the expensive computation of SVD. They gradually constructed a large scale model by relying on, previously computed, small SVD model and projecting new users/ratings to it [10]. In 2006, Netflix started a competition with 1 million dollar as a prize for the team which could improve the accuracy of their existing RS by at least 10%. This competition ended in 2009 when the grand prize was given to a team who blended SVD-based recommender with a stochastic artificial neural network technique called Restricted Boltzmann Machines [23]. Gong, Ye and Dai had proposed an algorithm which combined SVD and the traditional item-based CF approach. They computed SVD for the sparse user-item matrix and then multiply U, S, V again to get a filled matrix with approximation to the originally missing values. Then CF was applied on the new matrix to find the closest neighbors to the target item and thus provide good recommendations [17]. Zhou et. al. proposed an approximation to SVD which could provide more accurate recommendations than the standard SVD and could be computed more efficiently. Their work is summarized in sampling the rows of a user-item matrix according to sampling probabilities and constructing a smaller matrix C. Then compute SVD on the newly constructed matrix C and not the original matrix [24]. In another effort by Lee and Chang, Stochastic Singular Value Decomposition was used instead of conventional similarity measure to overcome the scalability problem of existing item-based CF recommender systems. Their work was implemented using Apache Mahout MapReduce [9].

3. EXPERIMENTS AND EVALUATIONS

3.1. Experimental Environemt

All the experiments were conducted using Scala programming language on Eclipse, running on MacBook Pro with X 10.9.3 OS, 2.4 GHz Intel Core i5 processor and 8 GB of RAM. This machine served as a single node cluster for Apache hadoop 2.4.0 which was configured in pseudo-distributed mode. In addition, Apache spark v. 1.0.2 was used as it provides fast distributed computations.

3.2. Data Set

The data set used in this work is the 1M MovieLens set collected from MovieLens website by GroupLens research lab of the Department of Computer Science and Engineering at the University of Minnesota. This data set contains 1million ratings provided by more than 6000 users to around 3900 movies in the form of tuple (userID, MovieID, rating, timestamp). Ratings take integer values in the interval [1, 5] indicating how much the user likes the movie.

The aforementioned data set was divided into training set and test set based on different ratios known as training ratios [18]. For illustration, a training ratio of 0.8 indicates that 80% of the original data set is used as training set and the other 20% are kept as test set. To put it another way, the training set is used to fill the user-item matrix R of size 6040×3900 where each cell in it holds the preference of a user to a particular item. This will be used to compute SVD, come-up with U, S, and V matrices as well as predict ratings for unrated items. On the other hand, the test set will be used to evaluate the accuracy of the predicted ratings.

3.3. Evaluation Metric

Different empirical evaluation metrics are there to assess the quality of the estimated predictions. The most common metrics are the statistical one such as Mean Absolute Error (MAE) and Root Mean Square Error (RMSE). In this work, MAE is used due to the ease of interpreting it.

The evaluation process of this work is illustrated by Figure 4, where the data set was divided into two disjoint sets; one for training and the second for testing the system as mentioned before in section 3.2. The predicted ratings will be compared with the actual ratings in the test set by measuring MAE which will compute the average of the absolute difference between each predicted value and its corresponding actual rating, [18] i.e:

$$\mathrm{MAE} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{N}} \left|\mathrm{p_{i,j}} - \mathrm{r_{i,j}}\right|}{\mathrm{N}}$$

Where N is the size of the test set, $p_{i,j}$ is the predicted rating for user i and $r_{i,j}$ is the actual ratings for user u.

A smaller value of MAE refers to a higher prediction accuracy and thus better recommendations.

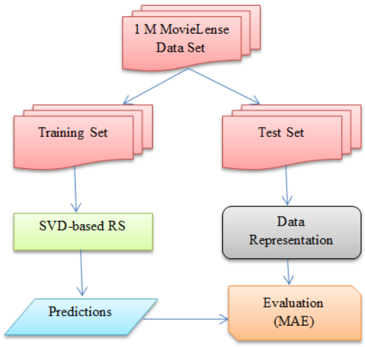


Figure 4. Prediction Evaluation Flow

Important to realize, the empirical evaluation will not address the computation time. That is because the work was done on a single machine while the algorithm is designed to run on a cluster. Thus, measuring its running time on a single machine will not reflect its real performance on a multi-node cluster. However, analytical evaluation could be used to assess the running time. As a matter of fact, the RS consists of two components: an offline component and an online one. The offline component does not affect the real-time performance of the system as all the operations will be pre-computed. Fortunately, computing the SDV, which is the most expensive operation in the whole system, is done in an offline mode. On the other hand, the online component simply generates the perdition by multiplying 2 vectors of size k. This is $O(1)$ as the value of k is constant.

3.4. Choosing the number of dimensions

Reducing the dimensions of the original matrix R is useful because it aids in eliminating the noise and focusing on the important information. With this in mind, an appropriate value of k should be selected such that it can filter out the noise but not leads to the loose of important information. In another word, the value of k should be large enough to ensure capturing the essential structure of matrix R but small enough to filter out noise and avoid overfitting [18], [20]. The best value of k will be experimentally determined by trying different values.

3.5. Experiments and Results

In the first place, 1M MovieLens data set was loaded into HDFS and then the training set, was used to fill the user-item matrix R. After that, R underwent two preprocessing operations: imputation and normalization. The imputation was done by mean of item average rating and user average rating, after experimentally proving its superiority over other imputation techniques (refer to Figure 5). Furthermore, the normalization step subtracted the average rating of each user from its corresponding row resulting in R_{norm}.

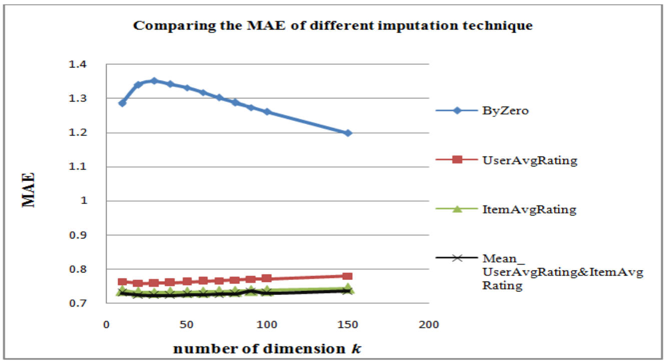


Figure 5. Comparing different imputation techniques

This was followed by using Apache Spark to compute SVD and come up with: U_k, V_k, and S_k. This is equivalent to extracting both user's and items' features from R. For that purpose, k was set to 20 after experimentally proving its superiority over other values. Refer to Figure 6.

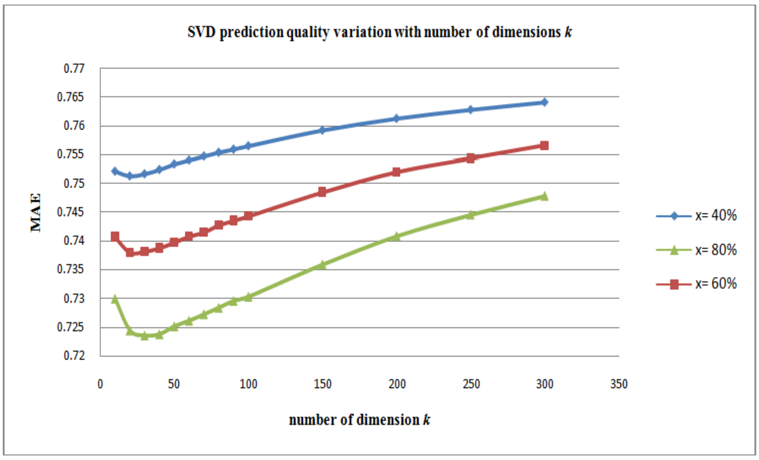


Figure 6. Determination of the optimal number of dimensions k

In order to compute a missing rating for one user, its corresponding row of $(U.S)$ was multiplied by V^T column that corresponds to the target item and then denormalized by adding the user average rating.

This workcould be expressed using the following algorithm:

```
Algorithm: Large Scale SVD-based Recommender System
//Input: 1 M MovieLense Big Data Set
//Output: A filled user-item matrix P with predictions to all the originally missing ratings
read data from HDFS in the form of tuples (userID, itemID, rating)
trainingSet←80% of the data
testSet← 20% of the data
//Construct user-item matrix from trainingSet
for  everyuserID
    Find all the ratings given by him and construct a row r with these ratings
user-item matrix R← all rows rof all the users
//Imputer R by Mean_ ItemAvgRating&UserAvgRatingavg
for every r in R
    compute average v1 of all the ratings in r
for every column c in R
    compute average v2 of all the ratings in c
for every cell $R_{ij}$ in R
    if$R_{ij}$= nil
    $R_{ij}$← average of $v1_i$ and $v2_j$
//Normalize R
for every r in R
    for every cell Rj in r
        $R_j$←$R_j$- v1
//Compute SVD
number of dimensions k←20
compute SVD of R to get U, S, V
//Predicting the missing ratings
computer the dot product of U and S to get US
find the transpose of V to get $V^T$
compute the dot product of US and $V^T$to get the predictions P
//De-normalize P
for every r in P
    for every cell Pj in r
        $P_j$←$P_j$+ v1
```

3.6. Discussion

The main computational advantage of running these experiments, which implement SVD-based recommender system, using Hadoop, Spark and Scala is its easy parallelization. Proving the powerfulness of these frameworks/APIs in implementing large-scale systems with parallelized operations in distributed mode.

The results are comparable with the results of other works, discussed previously in the literature review, such as the one conducted by Sarwar and his colleagues, by Gong and Dai as well as by Zhou and his colleagues; that were carried on significantly smaller data set (i.e. 100 K MovieLense Data Set). This proves that SVD approach is not only effective for, ordinary, small data but even for Big Data sets.

Indeed, this work resulted in better predictions when compared with Sarwar et. al. work [18] although it has been carried on much bigger data set. To put it differently, the best predictions obtained by Sarwar et. al. on 100K MovieLense data set were using training data set of 80% and $k \in [20,100]$ as they get MAE ranging from 0.748 to 0.732. However, the MAE obtained by our implementation, for the same values of k, the same training ratio and 1M MovieLense data set, were ranging between 0.724 and 0.730.

While looking for the best value of k, 20 was found as the favorable one since it gave a small value of MAE when checking it over different training ratios. This is reasonable when comparing it with previous works which found k = 14 [10], [18] or k = 15 [17] for smaller data set. Notable, increasing the volume of the data set to 1 million ratings did not, dramatically, increase the value of k which validates other researchers' opinions, reported in some research papers, in which a small number of dimensions usually give pretty good results with good approximation to the original matrix R. This is simply because a small value of k is sufficient to capture the important features of users and items and thus make good predictions. However, increasing the value of k might simply represent adding more noise to the data which does not add value to the process of making predictions.

Furthermore, trying different imputation techniques and tracking their MAE showed the importance of pre-processing steps and its effect on the prediction accuracy. As per our experiments, Mean_ItemAvgRating&UserAvgRating outperformed other imputation techniques since it gave lower MAE.

Moreover, repeating the experiments multiple times with different values of k and different values of training ratio x; revealed the sensitivity of the prediction quality to the sparsity of the data set since MAE values decrease as the training ratios increase and the sparsity decrease. Added to that, it revealed the significant effect of the value of k on the prediction quality, as well as the effectiveness of SVD in dealing with cold-start cases.

4. CONCLUSION

Recommender systems have been developed and integrated in many websites, especially e-commerce websites, long time ago. They proved their powerfulness in providing personalized, customized, web content to different users by recommending content (i.e. items in the case of e-commerce website) of interest to each user and thus mitigate the problem of information overload on the user.

Different techniques and approaches are there for recommender systems. One of the widely used techniques is collaborative filtering which mines the interaction records between users and items, purchase history, to infer user's taste and thus recommends items that match his taste.

Surprisingly, recommender systems, CF techniques in particular, have started facing some challenges with the dawn of Big Data era. This new phenomenon, Big Data, is inflaming the data volume to be processed by RS, as the number of users and content/items continue to increase, and thus raises some concerns about the sparseness of the available data, scalability of RSs as well as the quality of the predictions. With the hope of recommender systems to continue its success journey, it should process millions of items and users per seconds without degrading its prediction accuracy. For this purpose, new approaches of CF have been proposed and studied in researches after the traditional approaches showed their limitation. Among these approaches is Singular value decomposition. Furthermore, several Big Data frameworks and APIs (such as Hadoop, Mahout and Spark) have been released and tried in building large-scale recommender systems.

This research work makes a contribution to the state of the art of recommender systems in the sense that it provides an implementation of a large scale SVD-based recommender system using both Apache Hadoop and Spark. This came as a result of an intensive study to the literature as well as performing multiple experiments using Scala programming language on top of apache Hadoop and Spark. The study involved several topics which are: Big Data phenomenon, the different techniques and approaches of recommender systems together with their pros and cons, the challenges posed by big data on recommender systems and CF in particular, the applicability of SVD for recommender systems as well as its effectiveness in solving the aforementioned challenges. The experiments were conducted to determine the optimal values of two essential parameters that affect SVD-based RS which are: the imputation technique to be used in filling the user-item matrix before processing it and the number of dimensions k to be retained after decomposing the matrix. The results showed that Mean_ItemAvgRating&UserAvgRating is the best imputation technique and k=20 is the optimal number of dimensions as it gave the lowest MAE.

This work solved the scalability problem by utilizing Hadoop and its valuable features. In addition, it showed that pretty good quality could be achieved by choosing a robust imputation technique (as a preprocessing step) before applying SVD to the user-item matrix. Moreover, it asserted that Apache Spark comes with attractive merits which enable easy integration with Hadoop and easy development of parallelizable code.

This drew a conclusion that a careful implementation of a scalable SVD-based collaborative filtering recommender system is effective when choosing the right parameters and the appropriate frameworks and APIs.

5. FUTURE WORK

The obtained, promising, results are just the starting point. One might consider deploying this implementation of SVD-based RS on a multi-node cluster to evaluate its scalability, performance (its computation time in particular) and accuracy in a distributed mode.

In addition, more research to be conducted to explore Apache Spark implementation of SVD for a given matrix and find out possible ways of improving its performance in term of running time, result's quality and handling cold start problem. For this purpose, one might consider a hybrid approach that combines: stochastic version of SVD proposed by Lee and Chang, [9] incremental version of SVD proposed by Sarwar, B. et al. [10] and Expectation Maximization technique presented by Kurucz et al. This should replace the traditional implementation of SVD by an iterative process, which is the heart of Expectation Maximization, that applies a stochastic version of SVD, repeatedly, to a matrix and use the outcome of one iteration to impute the input of the next iteration. Stochastic SVD could be done in an incremental manner such that the advent of a new user will not imply re-computing the decomposition of user-item matrix; but the new user will be project to the existing SVD model.

Another research effort should be dedicated to experiment other Big Data tools and framework such as Apache Mahout and compare its performance with Apache Spark.

REFERENCES

[1] Schönberger V, and Cukier K., "A revolution that will transform how we live, work, and think", New York: Houghton Mifflin Harcourt, 2013.

[2] Chen J., Chen Y., Du X., Li C., Lu J., and Zhao S., *et al.*, "Big Data Challenge: A data management perspective", *Front. Comput. Sci.* Vol. 7, No. 2, pp. 157-164, 2013.

[3] Bizer C., Boncz P., Brodie M. L., and Erling O., "The Meaningful Use of Big Data: Four Perspectives- Four Challenges", *SIGMOD,* Vol. 4, No. 4, pp. 56-60, 2011.

[4] Villa A., "Transfering Big Data Across the Globe", Dissertation, New Hampshire (NH): University of New Hampshire Durham, 2012.

[5] Schelter S., Owen S., *Proceedings of ACM RecSys Challenge '12,* Dublin, Ireland, 2012.

[6] Schönberger V. M., Cukier K., "Big Data: A revolution that will transform how we live, work, and think". New York: Houghton Mifflin Harcourt, 2013.

[7] Chiky R., Ghisloti R., and Aoul Z. K., *Proceedings of EGC 2012,* Bordeaux, France, 2012.

[8] Thangavel S. K., and Thampi N. S., "Performance Analysis of various Recommendation Algorithm Using Apache Hadoop and Mahout", *IJSER,* Vol. 4, No. 12, pp. 279-287, 2013.

[9] Lee C. R., Chang Y. F., "Enhancing Accuracy and Performance of Collaborative Filtering Algorithm by Stochastic SDV and Its MapReduce Implementation". *In: Raś Z W, Ohsuga S, editors. IPDPSW 2013;* Cambridge. USA: IEEE computer Society, pp. 1869-1878, 2013.

[10] Sarwar B., *et al.*, "Incremental Singular Value Decomposition Algorithms for Highly Scalable Recommender Systems", *5th International Conference on Computer and Information Science,* pp. 27-28, 2002.

[11] Kabore S. C., "Design and Implementation of a Recommender System as a Module for Liferay Portal", Master thesis, UPC: University Polytechnic of Catalunya, 2012.

[12] Melville M., Sindhwani V., "Recommender Systems", In : Sammut, Claude, Webb, Geoffrey I, editors. Encyclopedia of Machine Learning, US: Springer, pp. 829-838, 2010.

[13] Rijmenam K. V., "Recommender Engines are Crucial for Positive User Experiences", 2013

[14] Owen S., Anil R., Dunning T., and Friedman E., "Mahout in Action", New York: Manning Publications, 2012.

[15] Walunj S., Sadafale K., "An online Recommendation System for E-commerce Based on Apache Mahout Framework", *2013 annual conference on Computers and peaple research,* New York, pp. 153-158, 2013.

[16] Walunj S., Sadafale K., "Priced based Recommendation System". *International Journal of Research in Computer Engineering and Information Technology,* Vol. 1, No. 1, 2013.

[17] Gong S., Ye H., and Dai Y., “Combining Singular Value Decomposition and Item-based Recommender in Collaborative Filtering”, *Second International Workshop on knowledge discovery and data mining,* USA: IEEE, pp 769-772, 2009.
[18] Sarwar B. M., Karypis G., Konstan J. A., and Riedl J. T., “Application of Dimensionality Reduction in Recommender System A Case Study”, *ACM WebKDD,* 2000.
[19] Pagare R., and Patil S. A., “Study of Collaborative Filtering Recommendation Algorithm –Scalability Issue”, *International Journal of Computer Applications,* Vol. 67, No. 25, pp 10-15, 2009.
[20] Berry M. W., Dumais S. T., and O’Brien G. W., “Using Linear Algebra for Intelligent Information Retrieval”, *SIAM Review,* Vol. 37, pp. 573-595, 1995.
[21] Ghazanfar M. A., and Bennett A. P., “The advantage of Careful Imputation Source in Sparse Data-Environment of Recommender Systems: Generating Improved SVD-based Recommendations”, *Informatica,* Vol. 37, pp 61-92, 2012.
[22] Vozalis M. G., and Margaritis K. G., “Applying SVD on Generalized Item-based Filtering”, *Internatinal Journal of Computer Science & Application,* Vol. 3, No. 3, pp. 27-51, 2006.
[23] Gower S., “Netflix Prize and SVD, 2014.
[24] Zhou X., He J., Huang G., Zhang Y., “A Personalized Recommendation Algorithms Based on Approximating the singular value decomposition (ApproSVD)”, *IEEE/WIC/ACM International Conferences on Web Intelligent Agent Technology,* USA: IEEE, 2012.

BIOGRAPHIES OF AUTHORS

Khadija Ateya Almohsen was born in 1988 in Bahrain. She received her B.Sc. in Computer Science, from University of Bahrain, Bahrain in 2011 with G.P.A 4 out of 4.

Ms. Khadija joined Ahlia University in 2011 as a Research Assistant. She is currently studying Master in Information Technology and Computer Science in Ahlia University. Her research interests are: Big Data, Machine Learning, Database and Algorithms.

Prior to joining Ahlia University, she was working in Microcenter, Bahrain as Technical Support Executive and Trainer.

Huda Kadhim AL-jobori was born in 1971 in Baghdad, Iraq. She received her Ph.D. in Computer Science and Information System, from University of Technology, Iraq in 2003.

She joined Ahlia University in 2008 as an Assistant Professor.

Dr. Huda has a B.Sc. in Computer Science from AL-Nahrain University, Iraq, and a M.Sc. in Computer Science from AL-Nahrain University, Iraq. Her research interest is Information Security, Artificial Intelligence, Computer Networks, Information Hiding, Image Processing, Database. She has published many papers in refereed journals. She participated in the review of many journal and conference papers and supervised many master and undergraduate students.