The Advantage of Careful Imputation Sources in Sparse Data-Environment of Recommender Systems: Generating Improved SVD-based Recommendations

Mustansar Ali Ghazanfar and Adam Prugel-Bennett
School of Electronics and Computer Science, University of Southampton, Highfield Campus, SO17 1BJ, United Kingdom
E-mail: eng.musi@gmail.com, adp@ecs.soton.ac.uk,
Phone: +44 (023) 80594473; fax: +44 (023) 80594498

Overview paper

Keywords: SVD, recommender systems, imputation, collaborative filtering

Received: December 2, 2012

Recommender systems apply machine learning and data mining techniques for filtering unseen information and can predict whether a user would like a given item. The main types of recommender systems namely collaborative filtering and content-based filtering suffer from scalability, data sparsity, and cold-start problems resulting in poor quality recommendations and reduced coverage. There has been some work in the literature to increase the scalability by reducing the dimensions of the recommender system dataset using singular value decomposition (SVD); however, due to sparsity it results in inaccurate recommendations. In this paper, we show how a careful selection of an imputation source in singular value decomposition based recommender system can provide potential benefits ranging from cost saving, to performance enhancement. The proposed missing value imputation methods have the ability to exploit any underlying data correlation structures and hence have been proven to exhibit much superior accuracy and performance as compared to the traditional missing value imputation strategy—item average of the user-item rating matrix—that has been the preferred approach in the literature to resolve this problem. By extensive experimental results on three different dataset, we show that the proposed approaches outperform traditional one and moreover, they provide better recommendation under new user cold-start problem, new item cold-start problem, long tail problem, and sparse conditions.

Povzetek: Opisani so priporočilni sistemi, tj. sistemi, ki filtrirajo informacije s pomočjo metod strojnega učenja.

1 Introduction

1.1 Recommender systems

There has been an exponential increase in the volume of available digital information, electronic sources, and online services in recent years. This information overload has created a potential problem, which is how to filter and efficiently deliver relevant information to a user. Furthermore, information needs to be prioritised for a user rather than just filtering the right information, which can create information overload problems. Search engines help Internet users by filtering pages to match explicit queries, but it is very difficult to specify what a user wants by using simple keywords. The Semantic Web, also provides some help to find useful information by allowing intelligent search queries; however it depends on the degree to which the web pages are annotated. These problems highlight a need for information extraction systems that can filter unseen information and can predict whether a user would like a given item. Such systems are called *recommender systems* [1, 2], and they mitigate the aforementioned problems to a great extent. Given a new item, recommender systems can predict whether a user would like this item or not, based on the user preferences (likes—positive examples, and dislikes—negative examples), observed behaviour, and information (demographic or content information) about items/users.

An example of the recommender system is the *Amazon* recommender engine [3], which can filter through millions of available items based on the preferences or past browsing behaviour of a user and can make personal recommendations. Some other well-known examples are *Youtube* (www.youtube.com) video recommender service and *MovieLens* (www.movielens.com) movie recommender system, which recommend videos and movies based on the person's opinions. Recommender systems helps E-commerce sites in increasing their sales by making useful recommendation—items a customer/user would most likely to consume [4]. In these systems, the history of user's interactions with the system is stored, which shape user's preferences. The history of the user can be gathered by explicit feedback, where the user rates some items in some scale, or by implicit feedback, where the user's interaction with the system is observed—for instance, if a user purchases an item then this is a sign that they like that item,

their browsing behaviour, etc.

There are two main types of recommender systems: collaborative filtering (CF) and content-based filtering recommender systems. Collaborative filtering recommender systems [5, 6, 7, 8, 9, 10] recommend items by taking into account the taste (in terms of preferences of items) of users, under the assumption that users will be interested in items that users similar to them have rated highly. Examples of these systems include the GroupLens system [9], and Ringo (www.ringo.com). Collaborative filtering can be classified into two sub-categories: memory-based CF and model-based CF. Memory-based approaches [6] make a prediction by taking into account the entire collection of previous rated items by a user. Examples of these systems include GroupLens recommender systems [9, 8]. Model-based approaches use rating patterns of users in the training set, group users into different classes, and use ratings of predefined classes to generate recommendation for an *active user*[1] on a *target item*[2]. Examples of these systems include item-based CF [11], Singular Value Decomposition (SVD) based models [12, 13, 14, 15], bayesian networks [16], clustering models [17, 18, 19, 20, 21, 22], and Kernel-mapping recommender [23].

Content-based filtering recommender systems [24, 25, 26] recommend items based on the content information of an item, under the assumption that users will like similar items to the ones they liked before. In these systems, an item of interest is defined by its associated features, for instance, NewsWeeder [24], a newsgroup filtering system uses the words of text as features. The textual description of items is used to build item profiles. User profiles can be constructed by building a model of the user's preferences using the descriptions and types of the items that a user is interested in, or a history of user's interactions with the system is stored (e.g. user purchase history, types of items they purchased together, etc.). The history of the user can be gathered by explicit feedback or implicit feedback. Explicit feedback is noise free but the user is unlikely to rate many items, whereas, implicit feedback is generally noisy (error prone), but can collect a lot of training data [27]. In general, a trade-off between implicit and explicit user feedback is used. Creating and learning user profiles is a form of classification problem, where training data can be divided into two categories: items liked by a user, and items disliked by a user. Furthermore, hybrid recommender systems have been proposed [28, 29, 30, 31, 32], which combine individual recommender systems to avoid certain limitations of individual recommender systems.

Recommendations can be presented to an active user in the followings two different ways: by predicting ratings of items, a user has not seen before and by constructing a list of items ordered by their preferences. The task of predicting an unknown rating is trivial, where an algorithm receives an unknown user-item pair and related users (user-based CF [8]), items (item-based CF [11]), item and user content features (content-based filtering recommender systems [33]), demographic features (demographic recommender systems [30]), or all of the aforementioned parameters (hybrid recommender systems [29]); and then predicts how much the user would like the given item in some numeric or binary scale. Different heuristics are used for producing an ordered list of items, sometimes termed as *top-N recommendations* [12]; for example, in collaborative filtering recommender system this list is produced by making the rating predictions of all items an active user has not yet rated, sorting the list, and then keeping the top-N items the active user would like the most.

[1]The user for whom the recommendations are computed.

[2]The item a system wants to recommend.

1.2 Problem statement

A Recommender system (RS) consists of two basic entities: users and items, where users provide their opinions (ratings) about items. We denote these users by $\mathcal{U} = \{u_1, u_2, \cdots, u_M\}$, where the number of users using the system is $|\mathcal{U}| = M$, and denote the set of items being recommended by $\mathcal{I} = \{ i_1,i_2, \cdots, i_N\}$, with $|\mathcal{I}| = N$. The users will have given ratings of some but not all of the items. We denote these ratings by $(r_{i,u}|(i,u) \in \mathcal{D})$, where $\mathcal{D} \subset \mathcal{I} \times \mathcal{U}$ is the set of user-item pairs that have been rated. We denote the total number of ratings made by $|\mathcal{D}| = T$. Typically each user rates only a small number of the possible items, so that $|\mathcal{D}| = T \ll |\mathcal{I} \times \mathcal{U}| = N \times M$. It is not unusual in practical systems to have $T/(N \times M) \approxeq 0.01$. The set of possible ratings made by users can be thought of as elements of an $M \times N$ rating matrix R with elements $r_{i,j}$. The recommender system's task is to infer the elements in R for which we do not have any data. As prediction accuracy of a recommender system depends heavily on the available number of examples, hence it would suffer in case where the rating data is sparse.

The continuous increase of the users and items demands the following properties in a recommender system: (1) accuracy (2) scalability (3) maximum coverage (4) robustness with *sparsity* [34, 35, 36, 37, 38]. A number of approaches have been proposed to remedy the sparsity and scalability problems associated with the CF, ranging from supervised classification techniques [39] to unsupervised clustering techniques [17], to dimensionality reduction techniques spanning a number of algorithms such as singular value decomposition [12], low rank approximation using matrix factorisation techniques [40], and principal component analysis [41]. Remaining properties can be satisfied by a hybrid approach resulting in increased coverage while eliminating the sparsity problem.

Singular value decomposition methods have proved to be successful in increasing the scalability of the CF [12, 42]; however approximating missing values by the item average, which has been heavily used in the literature, is not a reasonable approach. Although, it reduces sparsity and increases coverage, it will lead to lower recommendation accuracy. To efficiently deal with the sparsity problem, we have to address the basic question: *"Why are the data*

missing"? The following assumptions can be made about the missing data: *missing completely at random* (MCAR), *missing at random* (MAR), and *not missing at random* (NMAR) [43]. MCAR assumes that the data is missing for completely random reasons, and that the probability of observing the value of a rating does not depend on the observed or unobserved values of the dependent variable. Most of the CF algorithms assume that missing data is MAR [44], which means that probability of observing a rating does not depend on the value of the rating. NMAR occurs when the missingness is related to the unobserved dependent variable, i.e. if probability of observing the rating of an item depends on the value of the rating. NMAR assumes that there is a relationship between the missingness and what would have been observed.

To make a clear distinction between these categories, assume that $G_{i,j}$ is an indicator function, which takes a value of 1 if the subject i is observed at time j and 0 otherwise. If a study is conducted at n time-points, then the dependent variable and missing data indicator vectors can be represented by $y_i^{1\times n} = \{y_{i,1}, y_{i,2}, \cdots, y_{i,n}\}$ and $G_i^{1\times n} = \{G_{i,1}, G_{i,2}, \cdots, G_{i,n}\}$ respectively. The specific values in the missing data indicator vector, $G_{i,j}$, is equal to 1 when $y_{i,j}$ is observed and 0 otherwise. Based on the G, the dependent variable, y, can be divided into two classes: observed, y^o, and unobserved or missing, y^m. MCAR assumes that missing data indicator G_i are independent of both y_i^o and y_i^m. MAR assumes that missing data indicator G_i are independent of y_i^m; however, they are related to the observed dependent variable vector, i.e. y_i^o. An example of MAR is to drop some subjects from a study when their value falls below a certain threshold. For instance, in an election survey if a subject has age less than 18 then they are drop out of the survey. NMAR assumes that there is a relationship between the value of y_i^m and the missingness, $G_{i,j}$. MAR assumption is not true in recommender systems; for example, in MovieLens recommender system rating distribution is skewed towards the higher ratings, which indicates that users are much more likely to watch and rate movies they think they will like rather than entering rating for the movies they do not like, resulting in higher bias towards observing ratings with high value. Other researchers have claimed that this bias can results in erroneous recommendations [44].

The main claim of this work is that imputation methods to deal with the missing values, if effectively used prior to applying SVD in recommender systems, can provide potential benefits ranging from cost saving, to performance enhancement. The proposed missing value imputation methods have the ability to exploit any underlying data correlation structures and hence are proven to exhibit much superior accuracy and performance compared to the traditional missing value imputation strategy that has been the preferred approach in the literature to resolve this problem. This work, presents a comparative study of the missing values in the user-item matrix of a recommender system and their subsequent impact upon the accuracy and cost has been investigated. The empirical study has shown that the results are dataset dependent; however rather than using the traditional approach to fill the user-item rating matrix or merely ignoring the missing values, which have heavily been used in the literature, robust and advanced approaches can give considerable performance benefits in the (1) SVD based recommendations (2) iterative SVD (3) and CF applied over the reduced dataset. We evaluate our algorithms on the MovieLens (www.grouplens.org/node/73) ($100K$ ratings and $1M$ ratings) and FilmTrust (http://trust.mindswap.org/FilmTrust) datasets.

The rest of the paper has been organised as follows. Section 2 discusses the related work in detail. Section 3 presents the motivations and advantages of using the proposed algorithm. Section 4 sheds light on the background concepts related to the SVD. Section 5 outlines the proposed algorithms. Section 6 describes approaches used to approximate the missing values in the sparse user-item rating matrix. Section 7 describes the data set and metrics used in this work. Section 8 presents results comparing the performance of the proposed algorithms with the traditional one. Section 9 discusses when and how much imputation is sufficient to achieve good accuracy. Section 10 gives a detailed discussion of the work, and finally Section 11 concludes the work.

2 Related work

The Singular Value Decomposition (SVD)-based approach for solving the recommendation problem was first introduced by [39]. [12] presented a detailed analysis of the behaviour of SVD-based recommender systems. Various algorithms combining the SVD-based approach with the item-based CF have been advocated [45, 46, 47, 48]; for example, [46] combined SVD with the item-based CF and claimed that their approach outperformed the conventional item-based CF. An example of using the SVD-based approach with demographic data has been presented in [13], where the authors applied SVD over the user-item rating matrix and demographic data of users and items, and claimed that a system consisting of a linear combination of SVD-based demographic correlation and SVD-based (item-based) CF, increases the accuracy of the recommender system. [42] suggested an incremental SVD model building approach and claimed that it is more scalable than the conventional SVD-based recommender systems, while producing recommendations with same accuracy. All of the aforementioned approaches used the item average of the data matrix to approximate the missing values, which may destroy the covariance structure of the data, resulting in inaccurate recommendations.

Another way of applying SVD is presented in [49], where the authors applied Principal Component Analysis (PCA)[3] over a so-called 'gauge-set' of items—set of items

[3]PCA is a closely related concept to SVD, which reduces the dimen-

rated by every user in the system. Although it may reduce sparsity, getting this dataset is hard in real-life scenarios, and also it may lead to potential loss of information as we are ignoring ratings not in the gauge-set. Sometimes, case deletion strategy [41] is used for dealing with the missing values where all variables with missing values are omitted in the data matrix resulting in loss of information, which is not desirable. Some other well-known approaches to approximate the missing values are filling by zero and scaling the known entries as suggested by [50].

The missing values have been handled by the Expected Maximisation (EM) algorithm [51] by [52], [53], [54], [41], [55] and [14]. In this approach, the predictions generated by the current model are replaced by the previous one and the procedure is repeated until some stopping criteria are reached; for example, the error between two successive models becomes less than a threshold. The problem with this approach is that the final error and the convergence is highly dependent on the method used to approximate the initial values. [53] showed the convergence behaviour of the EM algorithm by approximating the missing entries by zeros [50] and using the gauge-set [49]. Furthermore, they proposed an approach by starting with a large rank approximation and gradually reducing the rank of SVD in each iteration of EM.

Netflix prize competition [56] has made extensive use of low rank approximation—matrix factorisation [57, 58, 59, 40, 60, 61] and SVD based scheme—for example in [62], the authors proposed a CF approach based on the global cost function. They proposed an accurate SVD based model that integrates implicit feedback. They also proposed a framework for integrating neighbourhood based and SVD based approaches and claimed that it is more accurate than other approaches. In [63], the authors removed global effect from the dataset by normalising it using different heuristics. Afterwards they simultaneously derived interpolation weights for all neighbours by solving an optimisation problem. They claimed that this approach is scalable and gives more accurate results than conventional neighbourhood based approach over Netflix dataset. In [54] the author proposed a model to maximise the log-likelihood of the available ratings using EM algorithm. These approaches often lead to *over-fitting* [64] and need extensive parameters tuning which may not be pragmatic or desired in certain cases.

Various SVD based approaches have been combined for improving the prediction accuracy; for example, in [58], the author proposed a solution for Netflix prize by blending 107 individual results, and won the Netflix 2007 prize. A similar approach is presented in [65], where the author proposed a linear combination of SVD based predictor, KMeans clustering, combining SVD with K-NN, post processing SVD with ridge regression, and others (total 72 predictors) and claimed that it gave 7.04% improvement in terms of Root Mean Square Error (RMSE) over Netflix's Cinematch[4] on Netflix prize competition. Another example is proposed in [66], where the authors used a linear combination of SVD, association rules, and other approaches for making accurate predictions. In [59], the authors combined (using ensemble methods) different variants of matrix factorisation, such as, regular matrix factorisation, and non-negative matrix factorisation, and claimed that combined approach gave 7% improvement, in terms of RMSE , over the Netflix CineMatch recommender system. Though, theoretically, we can increase accuracy of a recommender system by these methods, but practically it is not pragmatic [67]. Furthermore, these approaches completely miss other metrics, such as, coverage, and scalability proposed in [68]. In this work, we have focused on the SVD-based recommender systems [12, 13, 14, 47, 48] rather than matrix factorisation techniques.

The most similar work with ours is that undertaken by [14], where the authors used an item-item imputation technique in addition to the user-average over the Netflix dataset. Our work; however, differs from theirs in a number of areas, as follows: (1) they only used an item-item imputation while we are using 18 different approaches to analyse the behaviour of SVD and EM algorithms; (2) they only used one dataset; however, we are using three different datasets and furthermore, we find out that the results are highly dataset-dependent; (3) they claimed that the item-item imputation scheme is outperformed by the average, which is in contrast with our findings; and (4) we are applying CF over the reduced dataset; however, they did not apply it. In summary, their focus was on the efficient implementation of Lanczos, power iteration, and other algorithms rather than imputation; however, we are analysing the behaviour of the SVD-based algorithms under different recommender system conditions—cold-start, long tail, and sparsity problems.

The imputation has been used in collaborative filtering domain. The idea of using imputation in collaborative filtering domain was proposed by [16], where the authors used some default votes to decrease the sparsity of the user-item rating matrix. The author claimed that using the default votes in the user-based CF outperforms the conventional user-based CF in terms of accuracy. This idea has further been used by many researchers in various ways to approximate the missing values in the user-item rating matrix; for example, [28] used a Naive bayes classifier trained on the content profiles of users, [69] and [70] used information filtering agents or "Filterbot", [71] used a linear combination of user- and item-based CF, [72] used a recursive CF algorithm, and [73, 74] used several methods. The problem with these approaches is that they are not very scalable. Our approach is different from these because we are doing imputation in SVD domain and CF is applied over the dataset reduced by employing SVD. Furthermore, imputation has been used in other domains; for example, for Epistatic miniarray profiles [75].

Hybrid recommender systems have widely been used in

sionality by projecting high dimensional data along a smaller number of orthogonal dimensions.

[4]Cinematch is the Netflix proprietary recommender system.

the literature; for example, Pazzani [30] used a machine learning approach to build a classifier based on the demographic data about a user. The author used Winnow to extract features from user's home page to build the user model. Fab [76] employed a meta-level hybrid system [29], where first a content-based filtering using Rocchio's method [77] was applied to maintain a term vector model that describe the user's area of interest. This model was then used by CF to gather documents on basis of interest of community as a whole. LIBRA [33] a book recommender system, downloaded content information about book (meta data) from Amazon, and built user models using a Naive bayes classifier. In [31], the author proposed an on-line hybrid recommender system for news recommendation by dynamically adjusting weights to content-based filtering or CF.

Content-based recommender systems have also been combined with CF for reducing the sparsity of the dataset and producing accurate recommendations; for example in [36], the authors proposed a unique cascading hybrid recommendation approach by combining the rating, feature, and demographic information about items, and claimed that their approach outperforms other algorithms in terms of accuracy and coverage under sparse dataset. In [35], the authors combined Naive bayes classifier with CF using switching hybrid approach, and claimed that their algorithm provides better performance than other algorithms. Information filtering agents have been integrated with CF in [78, 69], where the author proposed a framework for combining the CF with content-based filtering agents. They used simple agents, such as spell-checking, which analyse a new document in the news domain and rate it to reduce the sparsity of the dataset. Again, the problem with these approaches is that, they are not very scalable.

Various hybrid recommender systems have been proposed using ontology and CF [79, 80, 81, 82, 83] to overcome the sparsity problem of the user-item rating matrix. For example in [79], the authors used domain specific ontologies to enhance the similarity between the items in the item-based CF. They linearly combine the similarities between items based on the user-item rating matrix and structure semantic knowledge about items to generate recommendations, and claimed that this semantically enhanced approach outperform item-based CF particularly given sparse dataset. The problems with these approaches in that they require laborsome knowledge engineering techniques to capture the domain specific knowledge and ontologies, which may not be pragmatic given millions of items that is common with E-commerce domains.

3 Motivation and advantages of the proposed algorithm

The motivation of our aim is to develop an efficient algorithm for producing accurate recommendations under sparse datasets at low cost. From this line of the research, we propose algorithms that satisfies the following properties:

3.1 Overcome sparsity problem

In many commercial recommender systems like Amazon; it is not unusual, even for active customer to provide ratings well under $< 1\%$ of all the available products. Furthermore, an increase in the number of items in the database will decrease the density of each user with these items. The performance—accuracy and coverage—of conventional collaborative filtering algorithms suffer the most under sparse conditions, because they rely on the similar users and items. In the worse case, we might find very few or no similar user (or item) as the correlation coefficient between two users (or items) can be defined if they have some ratings in common. If there are only a few common items, the correlation coefficient is poorly approximated, and thus less reliable recommendations are produced. The proposed algorithms do not suffer from sparsity because, they (1) use a suitable imputation source to approximate all the missing values (2) apply SVD over the dense user-item rating matrix, which captures the important latent relationship between users and items, leading to reduced sparsity and 100% coverage.

3.2 Accurate recommendations

An important task for a recommender system is to *find good items* and to ameliorate the *quality* of the recommendation for a customer. If a customer trusts and leverages a recommender system, and then discovers that they are unable to find what they want then it is unlikely that they will continue with the system. Consequently, the most important task for a recommender system is to accurately predict the rating of the non-rated user/item combination and recommend items based on these predictions. Our algorithms give more accurate recommendations as compared to the traditional SVD-based approaches and collaborative filtering.

3.3 Low on-line cost

The proposed SVD takes $O(1)$ time to generate a prediction which, is less than typical approaches. The complexity of the item-based and user-based CF, applied over the original user-item rating matrix, is $O(MN^2)$ and $O(NM^2)$ respectively[5]. When we reduce the user-item rating matrix to k dimensions using SVD, then the complexity reduces to $O(kN^2)$, and $O(kM^2)$, in case of user-based and item-based CF respectively. This reduced complexity will decrease the memory requirement and on-line processing time.

[5]We assume that SVD and the similarities between items and users are computed in off-line fashion.

3.4 Search efficiency: scalability

To engage visitors in a web-site or turn casual surfers into customer, a recommendation algorithm has to make recommendations quickly and accurately. In a database, with millions of ratings, it becomes very difficult to find the similar users or items using memory based approaches. By employing the imputed SVD, the dimension of the original user-item matrix can be reduced, and recommendations can be generated directly or CF can be applied over the reduced dimensions, making this approach suitable for high dimensional dataset.

3.5 Overcome cold-start problems

Generally, while testing recommender systems, a dataset is used where some sets of ratings are treated as unseen while the other ratings are used for learning. The unseen data are then used to test the performance of the algorithm. To obtain accurate results, datasets are usually selected with users that have made a relatively high number of ratings. However, in real applications, the datasets are often highly skewed; for example, a large number of users may have made only a small number of ratings and a large number of items may have received very few ratings. These are important scenarios in practical systems as making reasonable recommendations to new users can be crucial in attracting more users. There are two important cold-start scenarios as described below [84]:

- *New user cold-start problem:* When a new user enters the system, initially the system does not have enough data for that user, and hence the quality of the recommendations would suffer, a potential problem called the new user cold-start problem [34].
- *New item cold-start problem:* When a new item is added to a system, then initially it is not possible to get a rating for that item from a significant number of users, and consequently the CF recommender systems would not be able to recommend that item effectively. This problem is called the new item cold-start problem [34].

The Proposed imputation methods provide better recommendations than the conventional approach under the cold-start scenarios.

3.6 Overcome long tail problem

Newly introduced or unpopular items having only a few ratings can create a potential problem for a recommender system. Many recommender systems; for example, the CF ones, ignore these items or cannot produce reliable recommendations for these items. This problem is called the long tail problem [17]. As the majority of the items in a recommender system generally falls into this category [17], there is a need to develop algorithms which can filter, personalise, and accurately recommend from the huge amount of items available in the long tail. We show that the proposed imputation methods provide better recommendations than the conventional approach under the long tail scenario.

4 Background

Singular Value Decomposition (SVD) [85, 86] is a matrix factorisation technique that takes an $m \times n$ matrix A, with rank r and decomposes it into three component matrices as follows:

$$\text{SVD}(A) = U \times S \times V^{\mathsf{T}}. \tag{1}$$

U and V (V^{T} is for the transpose of V) are orthogonal matrices with dimensions $m \times m$, and $n \times n$ respectively, and S, called the *singular matrix*, is a $m \times n$ diagonal matrix consisting of non-negative real numbers. These matrices reflect the decomposition of the original matrix into linearly independent vectors (factor values). The set of initial r diagonal values of S ($s_1, s_2, \cdots, s_r$) are all positive with $s_1 \geq s_2 \geq s_3, \cdots, \geq s_r$. The first r columns of U are eigenvectors of AA^{T} and represent the left singular vectors of A. Similarly, the first r columns of V are eigenvectors of $A^{\mathsf{T}}A$ and represent the right singular vectors of A. The best low-rank approximation of matrix A is obtained by retaining the first k diagonal values of S, by removing $r - k$ columns from U, and by removing $r - k$ rows from V, which can be represented as follows:

$$A_k = U_k \times S_k \times V_k^{\mathsf{T}}. \tag{2}$$

By keeping only the k largest singular values of S, the effective dimensions of the SVD matrices U, S, and V become $m \times k$, $k \times k$, and $k \times n$ respectively. The best-k rank approximation of matrix A with respect to the Frobenius norm can be represented by:

$$||A - A_k||_F^2 = \sum_{i,u}(a_{iu} - \sum_k U_{uk} \times S_k \times V^{\mathsf{T}}{}_{ki}) \tag{3}$$

SVD can be applied over the user-item rating matrix, of dimensions $M \times N$, generated by a recommender system. It assumes that there is some latent structure—overall structure that relates to all or most items (or users)—in the matrix that is partially obscured by variability in ratings assigned to items (or assigned by users). This latent structure can be captured by transforming the matrix in low dimensions. After transformation, users and items can be represented by a vector in the k-dimensional space. The matrix product $U_k.\sqrt{S_k}^{\mathsf{T}}$ represents M (pseudo) users and $\sqrt{S_k}.V_k^{\mathsf{T}}$ represents N (pseudo) items in the k-dimensional space. For example, in a movie domain, each element of $\sqrt{S_k}.V_k^{\mathsf{T}}(i)$ $(1 \leq i \leq N)$ can be a feature of movie i, such as whether it is a horror movie, whether it is rated PG-13 or not, etc. Similarly, the corresponding element of $U_k.\sqrt{S_k}^{\mathsf{T}}(u)$ $(1 \leq u \leq M)$ shows whether the user likes these feature in movies. A rating assigned by a pseudo-users u on item i is denoted by $r'_{i,u}$. The prediction $\hat{r}_{i,u}$

for the uth user on the ith item can be computed by the following equation:

$$\hat{r}_{i,u} = U_k.\sqrt{S_k}^{\mathsf{T}}(u).\sqrt{S_k}.{V_k}^{\mathsf{T}}(i). \quad (4)$$

If we normalise the user-item rating matrix by subtracting the respective user average ($\overline{r}_i$) from a rating, then a prediction is given by the equation:

$$\hat{r}_{i,u} = \overline{r}_i + U_k.\sqrt{S_k}^{\mathsf{T}}(u).\sqrt{S_k}.{V_k}^{\mathsf{T}}(i). \quad (5)$$

5 Proposed algorithms

5.1 Imputed SVD (Algorithm 1)

We used various imputation methods, F (discussed in the next section), for approximating the missing values in the user-item rating matrix R and then applied SVD for reducing the dimensions of the matrix. The pseudo code to generate improved recommendations is given in Algorithm 1. In step 7, which serves as a pre-processing step, we fill in the missing values in the initial sparse user-item rating matrix by an imputation source. In step 8, we normalise the filled rating matrix by subtracting the respective user average from the filled rating matrix. In step 9, we reduce the dimensions of the filled normalised rating matrix by applying SVD. In the IMPUTEDERROR procedure, from steps 12 to 26, we find the optimal number of dimensions (k) by changing the dimension from 1 to 50 and observing the corresponding MAE.

5.2 Collaborative filtering applied over the reduced dataset (Algorithm 2)

We can apply the user- and item-based CF over the matrix components generated by the IMPUTE procedure. Algorithm 2 outlines the steps required to apply CF over the reduced data matrix. The similarity between two items can be found by Adjusted cosine or cosine measure [34]. We used Adjusted cosine similarity because it gave us more accurate results. The similarity between two items i_x and i_y can be found by measuring the cosine of angle computed over k users as follows:

$$\mathrm{sim}(i_x, i_y) = \frac{\sum_{u=1}^{k} {r'}_{i_x,u}.{r'}_{i_y,u}}{\sqrt{\sum_{u=1}^{k} {r'}^2_{i_x,u} \sum_{u=1}^{k} {r'}^2_{i_y,u}}}, \quad (6)$$

where ${r'}_{i_x,u}$ and ${r'}_{i_y,u}$ are the ratings assigned by user u on items i_x and i_y respectively. The ratings shown by r' are obtained from the matrix product $\sqrt{S_k}^{\mathsf{T}}.V_k$, which represents the rating given by k (pseudo) users on N items[6].

[6]We do not need to subtract the respect user average while measuring the similarity as the matrix has already been normalised prior to applying SVD.

Algorithm 1 : **ImpSvd**; Impute the matrix, compute SVD, and generate recommendations
Input: R, the user-item rating matrix; f, an imputation method
Output: $error^*$, the minimum MAE; k^*, the optimal number of dimensions for SVD

1: **procedure** SVDRECOMMENDATION(R, f)
2: (U, S, V)=IMPUTE(R, f)
3: ($error^*$, k^*)=IMPUTEDERROR(U, S, V)
4: **return** ($error^*$, k^*)
5: **end procedure**

6: **procedure** IMPUTE(R, f)
7: Fill in the missing values in the user-item rating matrix R by an imputation method f. Call the resulting dense matrix R_f.
8: Normalise the dense matrix (R_f) and call it R_N.
9: Apply SVD over the normalised matrix R_N and find three components of the matrix as shown in equation 1. Call these matrices U, S, and V.
10: **return** (U, S, V)
11: **end procedure**

12: **procedure** IMPUTEDERROR(U, S, V)
13: $error^* \leftarrow 10$
14: $k^* \leftarrow 1$
15: **for** $k \leftarrow 1, 50$ **do**
16: (U_k, S_k, V_k)=DIMREDUCE(U, S, V, k)
17: Compute $U_k.\sqrt{S_k}^{\mathsf{T}}$ and $\sqrt{S_k}.{V_k}^{\mathsf{T}}$
18: Make predictions using equation 5
19: Compute MAE for all predictions, call this $error_{new}$
20: **if** $error_{new} < error^*$ **then**
21: $error^* \leftarrow error_{new}$
22: $k^* \leftarrow k$
23: **end if**
24: **end for**
25: **return** ($error^*$, k^*)
26: **end procedure**

27: **procedure** DIMREDUCE(U, S, V, k)
Perform dimensionality reduction step:
28: Find S_k by setting $S_{i,i} = 0$ for $i > k$
29: Find U_k by removing $r - k$ columns from U
30: Find V_k by removing $r - k$ rows from V
31: **return** (U_k, S_k, V_k)
32: **end procedure**

We used the significance weighting schemes as proposed by [87] while measuring the similarity between users (or items).

The similarity between two users can be found by the Pearson correlation or the cosine of angle [88]. We used the cosine of angle, which gave us more accurate results than the Pearson correlation. The similarity between two users can be found by the cosine of angle, computed over k items, as follows[7]:

$$\text{sim}(u_a, u_b) = \frac{\sum_{i=1}^{k} {r'}_{i,u_a} . {r'}_{i,u_b}}{\sqrt{\sum_{i=1}^{k} {r'}^2_{i,u_a} \sum_{i=1}^{k} {r'}^2_{i,u_b}}}, \quad (7)$$

where ${r'}_{i,u_a}$ and ${r'}_{i,u_b}$ are the ratings assigned on item i by users u_a and u_b respectively. The ratings shown by r' are obtained from the matrix product $U_k\sqrt{S_k}^{\mathsf{T}}$, which represents the ratings given by M users on k (pseudo) items.

In the case of item-based CF, the prediction for an active user u_a on target item i_t is made by using the adjusted weighted sum formula as follows:

$$\hat{r}_{i_t,u_a} = \overline{r}_{u_a} + \frac{\sum_{i=1}^{l} sim(i, i_t) \times {r'}_{i,u_a}}{\sum_{i=1}^{l} |sim(i, i_t)|}, \quad (8)$$

where l represents the l most similar items against a target item, found after applying equation 6.

In the case of the user-based CF, the prediction for an active user u_a on target item i_t is made by using the adjusted weighted sum formula as follows:

$$\hat{r}_{i_t,u_a} = \overline{r}_{u_a} + \frac{\sum_{u=1}^{l} sim(u, u_a) \times ({r'}_{i_t,u} - \overline{r}_u)}{\sum_{i=u}^{l} |sim(u, u_a)|}, \quad (9)$$

where l represents the l most similar users against an active user, found after applying equation 7.

Individual predictions made by the user- and item-based CF can be combined linearly. We expect that combining these two approaches will result in an increase in the accuracy, as both of them focus on different kinds of relationships. Let $\hat{r}^{ub}_{i,u}$ and $\hat{r}^{ib}_{i,u}$ represent the prediction generated by the user- and item-based CF respectively. The final prediction is a linear combination of these predictions as follows:

$$\hat{r}_{i,u} = \alpha \times \hat{r}^{ub}_{i,u} + \beta \times \hat{r}^{ib}_{i,u}, \quad (10)$$

where parameters α and β can be found over the validation set. We call this algorithm $ImpSvd^{hybrid}_{CF}$.

[7]In this case, we do not normalise the user-item rating matrix prior to applying SVD.

Algorithm 2 : $\mathbf{ImpSvd_{CF}}$; Apply SVD over the reduced dataset

Input: R, the user-item rating matrix; f, an imputation method; $flag$, a variable to decide between the user- and item-based CF

Output: $error^*$, the minimum MAE; k^* and $neigh^*$, the optimal number of dimensions and neighbours for CF

procedure CFRECOMMENDATION(R, f, flag)
  (U, S, V)=IMPUTE(R,f)
  Start grid search over dimensions, k and neighbourhood size, $neigh$ to find the optimal number of dimensions, k^* and neighbourhood size, $neigh^*$
  (U_k, S_k, V_k)=DIMREDUCE(U, S, V, k)
  if flag = 1 **then**
    $\hat{r}^{ib}_{i,u}$ = $ImpSvd^{ib}_{CF}(U_k, S_k, V_k, neigh)$
  else
    $\hat{r}^{ub}_{i,u}$ = $ImpSvd^{ub}_{CF}(U_k, S_k, V_k, neigh)$
  end if
  Store the minimum MAE, $error^*$; the optimal number of dimensions, k^*; and the optimal number of neighbours, $neigh^*$
  End grid search
  return ($error^*$, k^*, $neigh^*$)
end procedure

procedure $ImpSvd^{ib}_{CF}(U_k, V_k, S_k, l)$
  Find the matrix product $\sqrt{S_k}.{V_k}^{\mathsf{T}}$
  Find the similarity between two items using equation 6
  Isolate l most similar items to the target item (neighbours of the target item) found using equation 6
  Make a prediction, $\hat{r}^{ib}_{i,u}$, using equation 8
  return $\hat{r}^{ib}_{i,u}$
end procedure

procedure $ImpSvd^{ub}_{CF}(U_k, V_k, S_k, l)$
  Find the matrix product $U_k.\sqrt{S_k}^{\mathsf{T}}$
  Find the similarity between two users using equation 7
  Isolate l most similar users to the active user (neighbours of the active user) found using equation 7
  Make a prediction, $\hat{r}^{ub}_{i,u}$, using equation 9
  return $\hat{r}^{ub}_{i,u}$
end procedure

5.3 Iterative SVD: Applying SVD combined with EM algorithm (Algorithm 3)

Algorithm 3 : **ItrSvd**; Apply SVD in EM fashion
Input: R, the user-item rating matrix; f, an imputation method; ϑ, threshold value to terminate the EM algorithm
Output: t, the number of iterations in the EM algorithm; $error^*$, the MAE observed after the EM algorithm converges

procedure ITERATIVERECOMMENDATION(R, f, ϑ)
  $t \leftarrow 0$
  $error^{(t)} \leftarrow 0$
  repeat
    (U, S, V) = IMPUTE(R, f)
    $(U_{k^*}, V_{k^*}, S_{k^*})$=DIMREDUCE($U, S, V, k^*$) ## k^* is the optimal number of dimensions learned through the validation set
    Compute $U_{k^*}\sqrt{S_{k^*}}^{\intercal}$, $\sqrt{S_{k^*}}{V_{k^*}}^{\intercal}$
    Call the current SVD model $\mathcal{M}_{k^*}$
    Make predictions using equation 5
    Compute the MAE for $\mathcal{M}_{k^*}$, call it $error_{new}$
    $t \leftarrow t + 1$
    $error^{(t)} \leftarrow error_{new}$
    $f \leftarrow \mathcal{M}_{k^*}$
  until $|error^{(t)} - error^{(t-1)}| < \vartheta$
  $error^* \leftarrow error^{(t)}$
  return t, $error^*$
end procedure

The $ItrSvd$ (Algorithm 3) uses the combination of SVD and Expected Maximisation (EM) [51] to estimate the missing values. As SVD calculations require the filled matrix, missing values are replaced by an imputation method prior to the k most effective eigenvalues being selected. In each iteration of the EM algorithm, the missing values are replaced by the corresponding values in the previous estimated model in the expectation step, i.e.

$$R_{iu}^{(t)} = \begin{cases} R_{iu} & if\ iu \in \mathcal{D}, \\ [\sum_k U_k \times S_k \times V^{\intercal}{}_k]_{iu}^{(t-1)} & otherwise, \end{cases} \quad (11)$$

and in the maximisation step the aim is to find the model ($\mathcal{M}^{(t)}$) parameters that minimises

$$\sum_{iu}(R_{iu}^{(t)} - \mathcal{M}_{iu})^2, \quad (12)$$

where $\mathcal{M}_{iu} = [\sum_k U_k \times S_k \times V^{\intercal}{}_k]_{iu}$. The algorithm keeps alternating between expectation and maximisation (SVD computation) steps, until it converges (the change in the MAE between two iterations becomes less than a pre-determined threshold (0.001)). This algorithm usually gives more accurate results after convergence; however, its drawback is that it is highly sensitive to the noise in the dataset and it only considers the global data correlation, which means that in a locally correlated dataset, it will lead to higher estimation error.

6 Proposed approaches to approximate the missing values in the user-item rating matrix

Our main focus is on careful selection of imputation sources for approximating the missing values in the user-item rating matrix that can lead to different results. Our claim is that, sensible approaches used for filling the missing values, prior applying the SVD, can lead to significant performance increase. The imputation approaches used are discussed below:

- *Filling by zero (zeros)*: We fill the missing values in the user-item rating matrix by zero. This approach is very simple, and computational efficient, which make it attractive. This approach does not take into account the underlying correlation structure of the data affecting the data variance that is generally high. Subsequently, if we have a large number of missing values, then this imputation approach can results in inaccurate recommendations.

- *Filling by random number (Rand)*: We fill the missing values in the user-item rating matrix by a random number generator function that generates a random number in the range of 1 to 5 in the case of MovieLens and 1 to 10 in the case of FilmTrust dataset. Its advantages and disadvantages are the same as those of Zero.

- *Filling by normal distribution (Nor_U, Nor_I)*: We fill the missing values in the user-item rating matrix by normal distribution $\mathcal{N}(\mu, \sigma^2)$. Here we denote Nor_U to represent the case where the corresponding user average and standard deviation of ratings are used as μ and σ respectively. Similarly, we denote Nor_I to represent the case where the corresponding item average and standard deviation of ratings (given by other users) are used as μ and σ respectively.

- *Filling by uniform distribution (UniformDist)*: We fill the missing values in the user-item rating matrix by uniform distribution $\mathcal{U}(a, b)$, where $a = 1, b = 5$ and $a = 1, b = 10$ for the MovieLens and FilmTrust dataset respectively.

- *Filling by item average (ItemAvg)*: In this approach an unknown rating is replaced by the average rating given by all the users in the training set. If no one has rated that item it is replaced by zero. This approach serves as a **baseline** for our experimental evaluation, as it has been the preferred approach to resolve this problem; for example it has been used in [12, 13, 46].

- *Filling by user average (UserAvg)*: In this approach an unknown rating for an active user is replaced by the average rating given by the active user in the training set. If the active user has rated no item, then it is

replaced by zero. This approach is very simple however, it can distort the shape of the distribution and can reduce the variance of the data. We use the term **conventional methods** for the ItemAvg and UserAvg imputation sources.

- *Filling by the average of user and item averages (UserItemAvg)*: In this approach an unknown rating is replaced by averaging the user's average rating and item's average rating.

- *Filling by user-based CF (UBCF)*: In this approach an unknown rating is predicted by using user-based CF[8]. In user-based CF there are three main steps to make a prediction: (1) find all the users who have rated the target item, (2) find the similarity of the users with the active user, and isolate these users also called *neighbours* of the active user, (3) make prediction by adjusted weighted sum of the ratings provided by neighbours. Despite their simplicity they give accurate predictions.

- *Filling by item-based CF (IBCF)*: In this approach, an unknown rating is predicted by using item-based CF. In Item-based CF there are three main steps to make a prediction: (1) find all the items rated by the active user, (2) find the similarity of these items with the target item, and isolate the items also called *neighbours* of the target item, (3) make prediction by adjusted weighted sum of the ratings provided by the active user on the neighbouring items. It has been argued [11] that item-based CF outperforms user-based CF.

- *Filling by the average of user- and item-based CF (UBIBCF)*: In this approach, an unknown rating is replaced by averaging the predictions generated by user-based and item-based CF.

- *Filling by SVM classifier (SVMClass)*: In this approach an unknown rating is replaced by using the results obtained by applying the SVM classifier over the training set. We normalise the data in scale of $0-1$ and used LibSVM [89] for binary classification. We used linear kernel rather than radial basis function (RBF), as other researchers have found that if the number of features are very large compared to the number of instances, there is no significant benefit of using RBF over linear kernel [90]. Furthermore, tuning parameters in RBF and polynomial kernels is very computation intensive given a large feature size. For multi class problem, several methods have been proposed, such as one-verse-one (1v1), one-verse-all (1vR), Directed Acyclic Graph (DAG), and unbalanced decision tree (UDT) [91]. We did not found any significant difference among the results obtained by these methods, hence we show results in case of 1v1 only. More details to use SVM for recommender systems can be found in our previous work [92]. We have chosen SVM, as they give good performance for text categorisation problems; for example, in [93], the authors claimed that they outperform KNN, Naive bayes, and other classifiers in text categorisation tasks.

- *Filling by Naive bayes classifier (NBClass)*: In this approach an unknown rating is replaced by using the results obtained by applying the Naive bayes classifier over the training set. The details of building and using Naive bayes classifier for recommender system can be obtained from our previous work [35, 92].

- *K nearest neighbours (KNN)*: In this approach an unknown rating is replaced by using the results obtained by applying the K Nearest Neighbours (KNN) using the Weka collection of machine learning algorithms [94]. KNN estimates missing values by searching for the K nearest neighbours (users) and then taking the weighted average of these K neighbours' ratings. In our work, the proposed scheme is similar to the KNN; however, it differs in that the contribution of each neighbour is weighted by its similarity to the active user. As the degree of contribution will be determined by the choice of weighting system, hence we tested our scheme with two weighting systems. In the first approach (shown by KNN in the results) we weight neighbours by $1 - dist$, where $dist$ is the distance between two neighbours. In the second approach (shown by WKNN in the results), we weight neighbours according to the following scheme employed by [75]:

$$weight(i,j) = \Big(\frac{dist^2}{1 - dist^2 + \epsilon}\Big)^2,$$

where $\epsilon = 10^{-6}$ is added to avoid dividing by zero. This function is similar to the Gaussian kernel function, which gives more weight to closer neighbours than distant neighbours. WKNN has proven to give good results in [75].

- *Filling by decision tree (C4.5)*: In this approach, an unknown rating is replaced by using the results obtained by applying the Decision Tree (C4.5) using the Weka library. Although the process of constructing the tree tries to minimise the error rate using the training data for evaluation, it will probably not perform well while classifying the test data. The reason is that it can easily be *over-fitted* to the training data [64]. Therefore, in order to generalise its performance, we pruned the tree by learning the pruning confidence over the training set. We used Laplace smoothing for predicted probabilities and kept the minimum number of instances per leaf to 2.

- *Filling by SVM regression (SVMReg)*: In this approach, an unknown rating is replaced by using the results obtained by applying the SVM regression over the training set. We used linear kernel and trained the

[8]If algorithm fails to predict a rating, then it is replaced by the approach given in 6. The same is true for other algorithms

cost parameters. We used nu-SVR version of the SVM regression using the LibSVM [89] library.

– *Filling by linear regression (LinearReg)*: In this approach, an unknown rating is replaced by using the results obtained by applying the linear regression (with default parameters) using the Weka library. This method tries to lower the data variance of missing value estimates, by exploiting the underlying localised or global correlation structure of the data.

– *Filling by logistic regression (LogisticReg)*: In this approach, an unknown rating is replaced by using the results obtained by applying the logistic regression (with default parameters) using the Weka library.

– *AdaBoost (AdaBoost)*: In this approach, an unknown rating is replaced by using the results obtained by applying the Ada Boost over C4.5 decision tree approach, using the Weka library. It can be less susceptible to the over-fitting than most learning algorithms; however it is sensitive to noisy data.

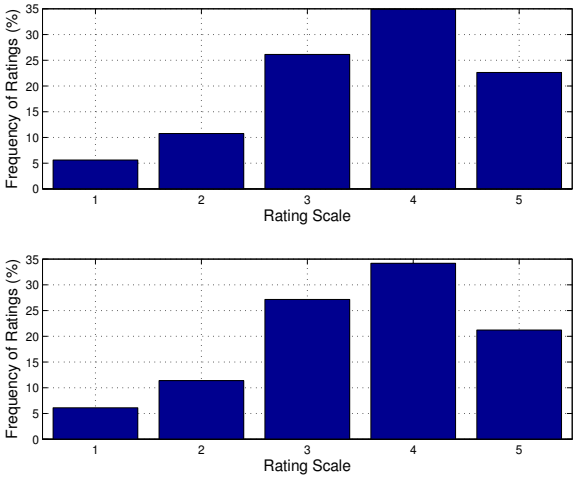


Figure 1: Rating distribution of the MovieLens datasets. The upper plot is for ML dataset and the lower plot is for SML dataset. We observe that, the rating distribution is skewed towards rating of 4.

7 Experimental evaluation

7.1 Datasets

We used the MovieLens (consisting of 100K ratings shown by SML and 1M ratings shown by ML) and FilmTrust (FT) datasets for evaluating our algorithm. The MovieLens data set has been used in many research projects [11, 13, 95, 92, 87]. We created the FilmTrust dataset by crawling (on 10th of March 2009) the FilmTrust website. The FilmTrust dataset is shown by FT1. We removed all movies and user from the FT1 dataset with less than 5 ratings and the resulting dataset is shown by FT5. The characteristics of the MovieLens and FilmTrust dataset are shown in Table 1. The sparsity of a dataset is calculated as follows: $\left(1 - \frac{\textit{non zero entries}}{\textit{all possible entries}}\right)$. Figure 1 and 2 show the distribution of ratings in datasets. We observe that in the MovieLens dataset, the rating distribution is skewed towards rating of 4, whereas, in the FilmTrust dataset it is skewed towards ratings between 9 to 10.

7.2 Feature extraction and selection

We downloaded information about each movie in the SML and FT datasets, from IMDB[9]. After stop word (frequently occurring words that carry little information [10]) removal [96] and stemming (removing the case and inflections information from a word and mapping it to the same stem[11]), we constructed a vector of ratings, keywords, tags, genres, directors, actors/actresses, producers, writers, and user reviews given to a movie in IMDB. We used TF-IDF (Term Frequency-Inverse Document Frequency) approach for determining the weights of words in a document (i.e. movie). In our case, we have 5 classes for the MovieLens and 10 classes for the FilmTrust dataset. These vector of features are used to train the classification and regression approaches discussed in Section 6. We used DF-Thresholding feature selection technique to reduce the feature space by eliminating useless noise words—words having little (or no) discriminating power in a classifier, or having low signal-to-noise ratio. We leverage WordNet using Java WordNet Interface (http://projects.csail.mit.edu/jwi/) for overcoming the *synonym* problem between features while finding the similarities among features. The details of training a classifier using these features can be found in our previous work [92].

It must be noted that the text categorisation and recommender system share a number of characteristics. A *Vector Space Model* is the most commonly used document representation technique, in which documents are represented by vectors of words. Each vector component represents a word feature and approaches, such as Boolean weights, $TF - IDF$, normalised $TF - IDF$ [97] etc. can be used for determining the weight of a word in document. The feature space in a typical attribute-value representation can be very large (e.g. 10 000 dimensions and more). A *word-by-document matrix* is used to represent a collection of documents, where each entry symbolises the occurrence of a word in a document. This matrix is typically very sparse, as not every word appears in every document. The recommender systems share the same characteristic. In [98] the authors argue that each user can be viewed as a document and each item rated by a user can be represented by a word

[9] We matched the movie titles, provided by the SML and FT datasets, against the titles in the IMDB (www.imdb.com). The details of the matching algorithm are beyond the scope of this paper.

[10] We used Google's stop word list www.ranks.nl/resources/stopwords.html.

[11] We used Porter Stemmer [27] algorithm for stemming.

Table 1: Characteristics of the datasets used in this work. The MovieLens dataset is shown by SML (100K ratings) and ML (1M ratings), and the FilmTrust dataset is shown by FT1 (original FilmTrust dataset) and FT5 (containing users and movies with at least 5 ratings). Average rating represents the average rating given by all users in the dataset.

Characteristics	**Dataset**			
	ML	**SML**	**FT**1	**TF**5
Number of users	6040	943	1214	1016
Number of movies	3706	1682	1922	314
Number of ratings	1000209	100000	28645	25730
Rating scale	1 (bad)-5 (excellent) (Integer scale)	Same as ML (Integer scale)	1.0 (bad)-10.0 (excellent) (Floating point scale)	−− −−
Sparsity	0.955	0.934	0.988	0.919
Max number of ratings given by a user	2314	737	244	133
Max number of ratings given to a movie	3428	583	880	842
Average rating	3.581	3.529	7.607	7.601

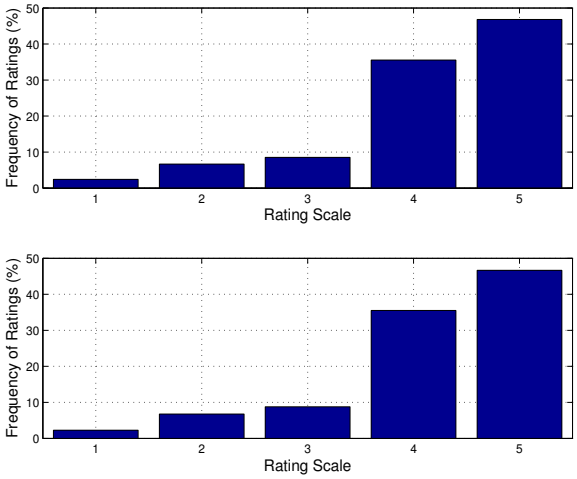


Figure 2: Rating distribution of the FilmTrust datasets. The upper plot is for FT1 dataset and the lower plot is for FT5 dataset. The rating scale is digitised as follows: a rating between 0 to 2.0 is represented by 1, a rating between 2.1 to 4.0 is represented by 2, a rating between 4.1 to 6.0 is represented by 3, a rating between 6.1 to 8.0 is represented by 4 , a rating between 8.1 to 10.0 is represented by 5. We observe that the rating distribution is skewed towards ratings between 8.1 to 10

appearing in a document. Our assumption is slightly different from the one used in [98], we view each user as a document; however we get (content) features against each item rated by a user. Each item is represented by a vector of bags of words and the user profile is represented by a big vector obtained by concatenating the vectors of bags of words of each item rated by a user. In this way, the user profile captured by a recommender system is very similar to the vector space model in text categorisation. Hence, our assumption is that the basic text categorisation algorithms can be applied to recommender system problem and that the results should be comparable.

7.3 Metrics

Several metrics have been used for evaluating recommender systems, which can broadly be categorised into *predictive accuracy metrics*, *classification accuracy metrics*, and *rank accuracy metrics* [68]. The predictive accuracy metrics measure how close is the recommender system's predicted value of a rating, with the true value of that rating assigned by the user. These metrics include mean absolute error, root mean squared error, and normalised mean absolute error, and have been used in research projects such as [19, 16, 18, 12, 11, 99]. The classification accuracy metrics determine the frequency of decisions made by a recommender system, for finding and recommending a good item to a user. These metrics include precision, recall, and $F1$ measure, and have been used in [12, 99]. The last category of metrics, rank accuracy metrics measure the proximity between the ordering predicted by a recommender system to the ordering given by the actual user, for the same set of items. These metrics include half-life utility metric proposed by Brease [16].

Our specific task in this paper is to predict scores for items that have already been rated by actual users, and to check how well this prediction helps users in selecting high quality items. considering this, we have used *Mean Absolute Error (MAE)*, *Receiver Operating Characteristic (ROC) sensitivity*, *precision*, *recall*, and $F1$ *measure*.

7.3.1 Mean absolute error (MAE)

The Mean Absolute Error (MAE) measures the average absolute deviation between the rating predicted by a recommendation algorithm and the true rating assigned by the user. It is computed as follows:

$$MAE = \frac{1}{|\mathcal{D}^{test}|} \sum_{r_{i,u} \in \mathcal{D}^{test}} |\hat{r}_{i,u} - r_{i,u}|,$$

where $r_{i,u}$ and $\hat{r}_{i,u}$ are the actual and predicted values of a rating respectively, and $\mathcal{D}^{test}$ is the set of rating records

Table 2: Classification of items in a document

	Selected	Not Selected	Total
Relevant	I_{rs}	I_{rn}	I_r
Irrelevant	I_{is}	I_{in}	I_i
Total	I_s	I_n	I

in the test set. A rating record is a tuple consisting user ID (Identifier), movie ID, and rating, $< uid, mid,$ where r is the rating a recommender system has to pre It has been used in [16], [12], [11], [42], [46], [71], [13], [36] and [35]. The aim of a recommender syste to minimise the MAE score.

7.3.2 Receiver operating characteristic (ROC) sensitivity

ROC is the extent to which an information filtering system can distinguish between good and bad items. *ROC sensitivity* measures the probability with which a system accept a good item. The ROC sensitivity ranges from 1 (perfect) to 0 (imperfect) with 0.5 for random. To use this metric for recommender systems, we must first determine which items are good (*signal*) and which are bad (*noise*). The guidelines of using this metric can be found in our previous work [92].

7.3.3 Precision, recall, and F1 measure

Precision, recall, and F1 measure evaluate the effectiveness of a recommender system by measuring the frequency with which it helps users selecting/recommending a good item [68]. The most appropriate way to measure the precision and recall in the context of recommender systems, is to predict the top-N items for the known ratings, which can be done by splitting each user's ratings into training and test set, training the model on the training set, and then predicting the top-N items from the test set. Here the underlying assumption is that, the distribution of relevant and irrelevant items in each user's test set, is the same as the true distribution for that user across all items. This has been used in [39].

Information retrieval [85] area, defines "objective" measure for precision, recall and related metric, where the relevance is independent to the user and is only associated with the query. However in context of recommender systems, the term "objective relevance" does not fit well—as every user have different taste, opinions, and reason to rate an item, hence, relevance is inherently "subjective" in recommender systems. The first step in computing the precision and recall is to divide items into two classes: relevant and irrelevant, which is the same as in ROC-sensitivity.

Precision gives us the probability that a selected item is relevant. A precision of 60% means that 6 of of every 10 recommendations for a user will be relevant. A user is more likely to understand the meaning of $x\%$ difference in

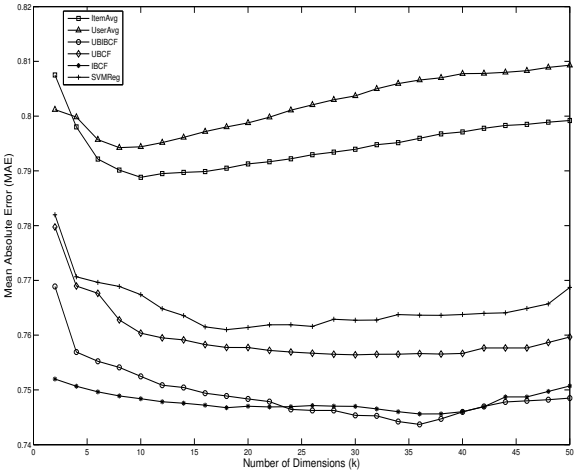


Figure 3: Determining the optimal number of dimensions in the imputed SVD over the training set (SML dataset). The error bars, lying between 0.001 and 0.004 for all approaches, are not shown for reasons of clarity.

precision rather comprehending 0.05%-point difference in the MAE [68]. Mathematically, it is defined as follows:

$$Precision = \frac{I_{rs}}{I_s}.$$

Recall gives us the probability that a relevant item is selected [68]. Mathematically, it is defined as follows:

$$Recall = \frac{I_{rs}}{I_r}.$$

Precision and recall should be measure together, as it has been claimed that they are inversely proportional to each other, and furthermore, they depend on the size of the resultant vector returned to the user. $F1$ measure [68] combines the precision and recall into a single metric and has been used in many research projects [12, 99]. F1 is computed as follows:

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

We calculated precision, recall, and F1 measures for each user, and reported the average results over all users.

7.4 Evaluation methodology

We performed the striated 5 fold cross validation and reported the average results with standard deviation. Each distinct fold contains 20% random ratings of each user as the test set and the remaining 80% as the training set. We further subdivided our training set into a validation set and training set for measuring the parameters sensitivity. For learning the parameters, we conducted 5-fold cross validation on the 80% training set, by selecting the different test and training set each time, and taking the average of results.

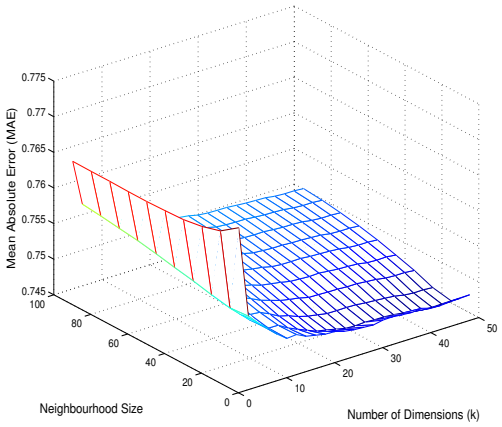


Figure 4: Determining the optimal parameters in user-based CF (for SML dataset with IBUBCF imputation source), through grid search over the training set. The "Number of Dimensions (k)" represents the number of dimensions in the reduced space (representing the k pseudo items) and "Neighbourhood Size" represents the number of most similar users against the active user.

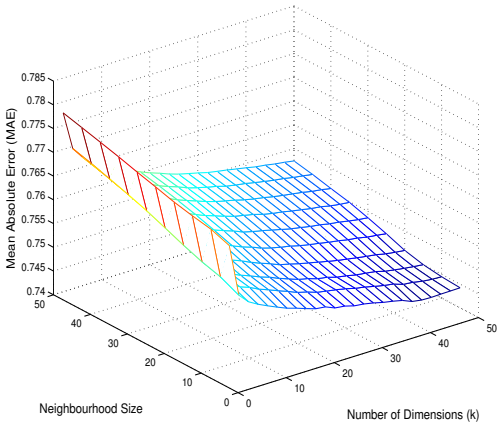


Figure 5: Determining the optimal parameters in item-based CF (for SML dataset with IBUBCF imputation source), through grid search over the training set. The "Number of Dimensions (k)" represents the number of dimensions in the reduced space (representing the k pseudo users) and "Neighbourhood Size" represents the number of most similar items against the target item.

8 Results and discussion

8.1 Learning the optimal parameters

The purpose of these experiments is to determine, which of the parameters affect the prediction quality of the proposed algorithms, and to determine their optimal values.

8.1.1 Finding the optimal number of dimensions

Two factors are important while finding the optimal number of dimensions. First, the number of dimensions must be small enough to make the resulting system scalable and second it must be big enough to capture the important latent information between the users or items. Figure 3 shows how the MAE changes as a function of the number of dimensions (k) in the case of SML dataset. We show results only for the conventional approaches and the ones giving us good results. We observe that, in the case of UBCF, IBCF, and UBIBCF, the MAE keeps on decreasing, reaches its minimum between $k = \{30 - 40\}$, and then starts increasing again. We choose $k = 36$ for these imputation methods. We further observe that the MAE is minimum at $k = 18$, $k = 8$, and $k = 10$ in the case of SVMReg, UserAvg, and ItemAvg respectively. Similarly, we tuned all approaches for the optimal dimensions for other datasets.

8.1.2 Finding the optimal number of neighbours and dimensions for user-based CF

The neighbourhood size is dataset dependent and furthermore change in the distribution and sparsity of the dataset will change the neighbourhood size. The work in [13] finds the optimal number of dimensions by keeping the neighbourhood size fixed to a value, changing the dimensions, and observing the corresponding MAE. The dimension that gives the minimum MAE is recorded to be the optimal one. Then the optimal neighbourhood size can be found by keeping the dimension parameter fixed to the optimal one. This sound a reasonable strategy; however, it does not show how the MAE changes with all possible combinations of both parameters—number of neighbours and dimension. We claim that *grid search* can effectively be used to find the optimal value of neighbourhood size and dimensions.

We performed a series of experiments by changing the dimension each time from 2 to 50 with a difference of 2. For each experiment, we changed the neighbourhood size from 5 to 100 with difference of 10, keeping the dimension parameter fixed, and observed the corresponding MAE. Figure 4 shows that the MAE is minimum at the neighbourhood size of 15. This is in contrast with the neighbourhood size in the classical user-based CF [11], where the MAE decreases with the increase in the neighbourhood size, reaches at its minimum for a specific neighbourhood size ranging from $\{50 - 70\}$, and then starts increasing again. The reason can be that filling the user-item rating matrix with an imputation source and then applying SVD may change the sparsity and distribution of the dataset. We observe in the dimension scale, keeping the neighbourhood size fixed to 15, that the MAE decreases with the increase in the rank of the lower dimension space, reaches at its peak at $k = 46$, and after that it either increases or stays constant. The grid coordinates, which gave the lowest MAE, are recorded to be the best parameter. In the case of UBIBCF imputation source, they found to be 15 for the neighbourhood size and 46 for the dimension. Similarly, we tuned the parameters for all approaches for other datasets.

Table 3: Learning parameter sets α and β over the training set through cross validation. α and β show the relative impact of user-based and item-based CF in a prediction respectively.

Params		**MAE**			
α	β	**ML**	**SML**	**FT**1	**FT**5
0.1	0.9	0.710	0.740	1.483	1.449
0.2	0.8	0.709	0.739	1.476	1.440
0.3	0.7	0.708	0.738	1.471	1.434
0.4	0.6	**0.706**	**0.736**	**1.470**	1.430
0.5	0.5	0.707	0.737	1.471	**1.429**
0.6	0.4	0.708	0.737	1.475	1.431
0.7	0.3	0.707	0.738	1.481	1.435
0.8	0.2	0.709	0.739	1.489	1.442
0.9	0.1	0.712	0.741	1.499	1.452

8.1.3 Finding the optimal number of neighbours and dimensions for item-based CF

We varied the number of dimensions from 2 to 50 with a difference of 2, and the number of neighbours from 5 to 50 with a difference of 5. Figure 5 shows that the MAE is minimum for the neighbourhood size of 5. After that, an increase in the neighbourhood size increases the MAE. In the dimension scale, keeping the neighbourhood size fixed to 5, we note that the MAE decreases with the increase in the rank of the lower dimension space, reaches at its peak at $k = 44$ and after that it either increases or stay constant. In the case of UBIBCF imputation source, the optimal parameters are found to be 5 for neighbourhood size and 44 for dimension. Similarly, we tuned the parameters for all approaches for other datasets.

8.1.4 Finding the optimal values of parameters α and β

Parameters α and β (refer to Section 5.2) determine the relative weights of user-based and item-based CF in the final prediction. The 9 parameter sets were generated by producing all possible combination of parameters values, ranging from 0.1 to 1.0 with differences of 0.1[12]. Table 3 present the parameter sets learned. The parameters sets $\alpha = 0.6, \beta = 0.4$; $\alpha = 0.6, \beta = 0.4$; $\alpha = 0.6, \beta = 0.4$; and $\alpha = 0.5, \beta = 0.5$ gave the lowest *MAE* in the case of ML, SML, FT1 and FT5 dataset respectively. It is worth noting that the values of parameters are found different for the MovieLens and FilmTrust dataset. We note that the item-based CF has more weight in the final prediction.

[12]We assume that $\alpha + \beta = 1$ without the loss of generalisation.

8.2 Performance evaluation of different imputation sources (Algorithm 1, $ImpSvd$)

The results obtained by *ImputedSVD* (Algorithm 1) under different imputation sources are shown in Table 4. Note that, we only show the best results obtained by varying k from 1 to 50. The table shows that the imputation methods SVM regression, UBCF, IBCF, and UBIBCF give more accurate results than others. The % decrease in MAE over the baseline method ItemAvg is found to be: (1) 4.79%, 5.61%, and 6.57% in case of UBCF, IBCF, and UBIBCF respectively for ML dataset (2) 5.16%, 5.55%, 5.94%, and 7.23% in case of SVM regression, UBCF, IBCF, and UBIBCF respectively for SML dataset (3) 17.0%, 14.70%, 14.52%, and 15.52% in case of SVM regression, UBCF, IBCF, and UBIBCF respectively for FT1 dataset (4) 5.86%, 2.70%, 2.56%, and 4.58% in case of SVM regression, UBCF, IBCF, and UBIBCF respectively for FT5 dataset. The ranking of the algorithms (with respect to the MAE) with the respective *p*-value in case of pair t test is found to be: (1) **UBIBCF** ($p < 0.001$) > **IBCF** ($p < 0.05$) > **UBCF** ($p < 0.05$) for ML dataset (2) **UBIBCF** ($p < 0.001$) > **IBCF** ($p < 0.001$) > **UBCF** ($p < 0.001$) >**SVMReg** ($p < 0.05$) for SML dataset (3) **SVMReg** ($p < 0.001$) > **UBIBCF** ($p < 0.001$) > **IBCF** ($p < 0.001$) >**UBCF** ($p < 0.001$) for FT1 dataset (4) **SVMReg** ($p < 0.001$) > **UBIBCF** ($p < 0.005$) > **IBCF** ($p < 0.001$) >**UBCF** ($p < 0.005$) for FT5 dataset. Furthermore, the proposed imputation sources give 5% to 10% improvement over the baseline approach, in terms of ROC-sensitivity, precision, recall, and F1 scores (results not shown due to space limits). These results indicate that proposed approaches accurately (1) predict an unknown rating for a user, (2) recommend the top-N items a user would like the most.

The FilmTrust dataset is a good example of the real world recommender system's characteristics. It has imbalanced data, i.e. a user may have 1 rating and other may have more than 100 ratings and the same is true for items as well. It captures well the *new user cold-start* and *new item cold-start* problems. What is evident from table 4 is that approximating missing values in the user-item rating matrix with the baseline approach gives the worst results. We observe that the SVMReg approach outperform others in FilmTrust dataset. The reason is that, the FilmTrust dataset is well suited to regression algorithms, as it has floating point scale (refer to Section 7.1). It is worth noting that, in the case of FT1 dataset, the UserAvg imputation approach gives accurate (or comparable) results as compared to the other approaches. We believe that it is due to the distribution of the dataset—in the FilmTrsut dataset, majority of the users have rated the popular set of movies and their rating tends to match the average user rating. In the case of FT5 dataset, the performance of approaches, even conventional ones, improves simply because we have removed users and items with less clear profiles. We note that, in FT5 case, again the baseline approach gives the worse re-

Table 4: Best MAE observed in different imputation sources. The best results have been shown in bold. K represents the number of dimensions, which gave the most accurate results.

Imp. Sr.	Best *MAE*				Dimension (k)		
	ML	**SML**	**FT**1	**FT**5	**ML**	**SML**	**FT**
Zeros	2.425 ± 0.002	2.321 ± 0.002	4.354 ± 0.020	3.898 ± 0.031	26	12	2
Rand	1.092 ± 0.002	1.072 ± 0.005	2.214 ± 0.019	2.064 ± 0.021	14	4	2
ItemAvg	0.730 ± 0.002	0.774 ± 0.002	1.700 ± 0.011	1.483 ± 0.012	22	10	10
UserAvg	0.759 ± 0.002	0.778 ± 0.002	1.452 ± 0.016	1.433 ± 0.005	22	8	4
UserItem Avg	0.724 ± 0.002	0.754 ± 0.003	1.527 ± 0.016	1.442 ± 0.014	30	12	14
Uniform Dist	0.911 ± 0.002	0.905 ± 0.002	2.061 ± 0.031	1.933 ± 0.023	10	4	2
Nor_U	0.790 ± 0.002	0.810 ± 0.002	1.505 ± 0.018	1.491 ± 0.010	4	2	2
Nor_I	0.766 ± 0.002	0.800 ± 0.002	1.796 ± 0.019	1.562 ± 0.020	2	2	2
UBCF	0.695 ± 0.002	0.731 ± 0.001	1.450 ± 0.016	1.442 ± 0.015	40	36	4
IBCF	0.689 ± 0.002	0.728 ± 0.003	1.453 ± 0.010	1.445 ± 0.013	40	36	6
UBIBCF	$\mathbf{0.682 \pm 0.002}$	$\mathbf{0.718 \pm 0.002}$	1.436 ± 0.014	1.415 ± 0.017	40	36	6
KNN	$--$	0.804 ± 0.005	1.485 ± 0.017	1.479 ± 0.019	$--$	18	4
WKNN	$--$	0.793 ± 0.002	1.481 ± 0.007	1.474 ± 0.008	$--$	18	4
NBClass	$--$	0.775 ± 0.005	1.475 ± 0.016	1.468 ± 0.017	$--$	26	8
SVMClass	$--$	0.763 ± 0.004	1.455 ± 0.014	1.445 ± 0.016	$--$	18	6
C4.5	$--$	0.781 ± 0.003	1.495 ± 0.012	1.485 ± 0.015	$--$	22	10
SVMReg	$--$	0.734 ± 0.004	$\mathbf{1.411 \pm 0.015}$	$\mathbf{1.396 \pm 0.019}$	$--$	18	6
Linear Reg	$--$	0.783 ± 0.003	1.447 ± 0.014	1.437 ± 0.018	$--$	16	4
Logistic Reg	$--$	0.781 ± 0.004	1.443 ± 0.015	1.434 ± 0.017	$--$	14	4
AdaBoost	$--$	0.772 ± 0.006	1.476 ± 0.014	1.468 ± 0.018	$--$	26	8

Table 5: The best MAE observed in different imputation sources in the case of item-based CF applied over the reduced dataset. The best results have been shown in bold.

Imputation Source	Best *MAE*			
	ML	**SML**	**FT**1	**FT**5
ItemAvg	0.741 ± 0.002	0.781 ± 0.0018	1.702 ± 0.017	1.475 ± 0.018
UserAvg	0.767 ± 0.002	0.788 ± 0.004	1.496 ± 0.015	1.442 ± 0.016
UBCF	0.721 ± 0.002	0.739 ± 0.003	1.483 ± 0.018	1.434 ± 0.011
IBCF	0.701 ± 0.002	0.738 ± 0.002	1.459 ± 0.013	1.462 ± 0.014
UBIBCF	$\mathbf{0.691 \pm 0.002}$	$\mathbf{0.723 \pm 0.004}$	1.432 ± 0.017	1.418 ± 0.018
SVMReg	$--$	0.744 ± 0.002	$\mathbf{1.417 \pm 0.019}$	$\mathbf{1.404 \pm 0.019}$

Table 6: The best MAE observed in different imputation sources in the case of user-based CF applied over the reduced dataset. The best results have been shown in bold.

Imputation Source	Best *MAE*			
	ML	**SML**	**FT**1	**FT**5
ItemAvg	0.742 ± 0.002	0.776 ± 0.002	1.731 ± 0.017	1.465 ± 0.018
UserAvg	0.773 ± 0.002	0.786 ± 0.002	1.483 ± 0.014	1.439 ± 0.015
UBCF	0.709 ± 0.002	0.734 ± 0.003	1.465 ± 0.015	1.422 ± 0.017
IBCF	0.706 ± 0.002	0.732 ± 0.002	1.446 ± 0.017	1.445 ± 0.018
UBIBCF	$\mathbf{0.692 \pm 0.002}$	$\mathbf{0.722 \pm 0.003}$	1.445 ± 0.019	1.419 ± 0.019
SVMReg	$--$	0.743 ± 0.002	$\mathbf{1.416 \pm 0.018}$	$\mathbf{1.401 \pm 0.019}$

Table 7: Best MAE, ROC-Sensitivity, Precision, Recall, and F1 observed in the case of hybrid recommender systems proposed in algorithm 3. The SMVReg is used for FilmTrust dataset and the UBIBCF is used for the remaining datasets as imputation source, prior applying the SVD. The optimal parameters are learned over the training set using grid search. Precision, Recall, and F1 have been measured over top 20 recommendations.

DataSet	MAE	ROC	Precision	Recall	F1
ML	0.684 ± 0.002	0.790 ± 0.002	0.518 ± 0.005	0.595 ± 0.003	0.524 ± 0.004
SML	0.717 ± 0.002	0.695 ± 0.006	0.543 ± 0.005	0.555 ± 0.003	0.513 ± 0.005
FT1	1.409 ± 0.013	0.566 ± 0.008	0.591 ± 0.008	0.568 ± 0.007	0.549 ± 0.007
FT5	1.394 ± 0.016	0.578 ± 0.008	0.598 ± 0.011	0.574 ± 0.008	0.556 ± 0.012

sults.

8.3 Performance evaluation of CF applied over the reduced dataset (Algorithm 2, $ImpSvd_{CF}$)

Table 5 and 6 show that the proposed approaches give more accurate results than the conventional ones, when we apply CF over the reduced dataset.

It is worth noting that the results (in general) obtained by applying CF over the reduced dataset do not give any advantage over the results obtained by applying the SVD. However, in the case of FilmTrust dataset, some of the proposed approaches (UBCF, IBCF, UBIBCF) give (insignificantly) better results when CF is applied over the reduced dataset. It might be due to the reason that the FilmTrust dataset is very sparse, which implies the the latent structure between movies and users might not be captured by applying the SVD, and can be found by applying CF over the reduced dataset. Another thing to note is that, the results obtained in the case of proposed approaches are (almost) equivalent to the ones obtained in the proposed Imputed SVD. Furthermore, in general, user-based CF performs better than the item-based CF in case of FT1, FT5, and SML dataset, whereas, item-based CF performs better than user-based CF in case of ML dataset.

8.4 Performance evaluation of hybrid recommender system (Algorithm 2, $ImpSvd_{CF}^{hybrid}$)

User-based and item-based CF can be combined linearly. Table 7 shows that by linearly combining the UBCF and IBCF, in case of UBIBCF imputation source, gives the improved results with MAE less than 0.684, 0.717, 1.409, and 1.394 in case of SML, ML, FT1, and FT5 datasets respectively. The reason of improvements in the results is that user-based and item-based CF focus on different kind of relationship in the dataset. In certain cases, user-based CF may be useful in identifying different kind of relationship that item-based CF will fail to recognize; for example, if none of the items rated by an active user are closely related to the target item i_t, then it is beneficiary to switch to user-oriented perspective that may find set of users very similar to the active user, who rated i_t.

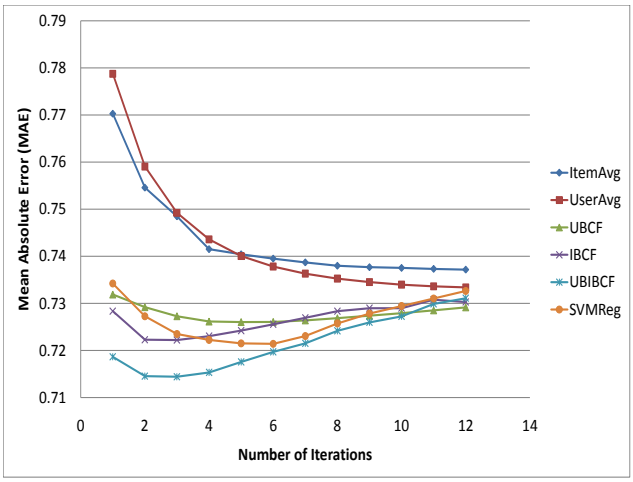


Figure 6: Comparing the proposed approaches with others in the case of iterative SVD (fixed dimension case), over SML dataset. X-axis shows the number of iterations and y-axis shows the corresponding MAE observed. The proposed approaches converge much quicker as compared to the conventional ones. The error bars (< 0.001 for all approaches) are not shown for reasons of clarity.

8.5 Performance evaluation of iterative SVD (Algorithm 3, $ItrSvd$)

There are two options to find the optimal number of dimensions in the $ItrSvd$ algorithm; (1) learning the optimal number of dimensions in the first iteration using the validation set and keeping them fixed for all the iterations, and (2) learning the optimal number of dimensions in each iteration using the validation. In the following, we represent the former case with *fixed dimension* and the latter one with *variable dimension*. We first show results for the fixed dimension and then proceed to the variable dimension case.

Figure 6 shows how the MAE changes with the number of iterations in SML dataset. We observe that the conventional approaches converge much slower as compared to the proposed ones. Figure 6 shows that in case of the baseline approach the MAE keeps on decreasing until it converges after 10 iterations. The minimum MAE observed after 10 iterations is 0.738. The MAE in case of the proposed approaches is shown at the lower plot of the figure. We observe that the MAE is much lower as com-

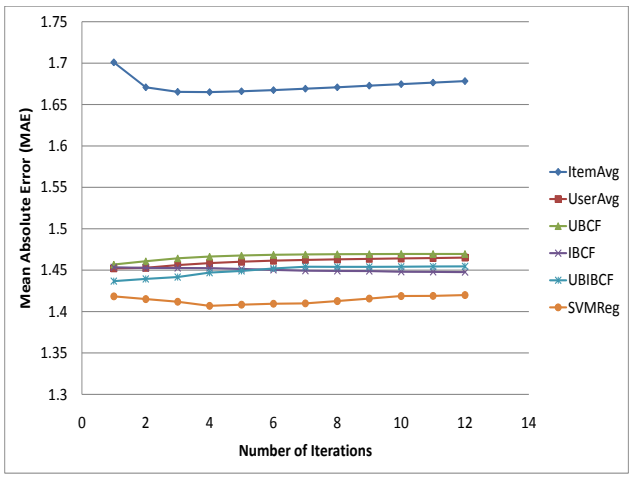


Figure 7: Comparing the proposed approaches with others in the case of iterative SVD (fixed dimension case), over FT1 dataset. X-axis shows the number of iterations and y-axis shows the corresponding MAE observed. The conventional approach converges much quicker than the others; however, the MAE observed after convergence is much higher than the proposed ones. The error bars (lying between 0.001 and 0.004 for all approaches) are not shown for reasons of clarity.

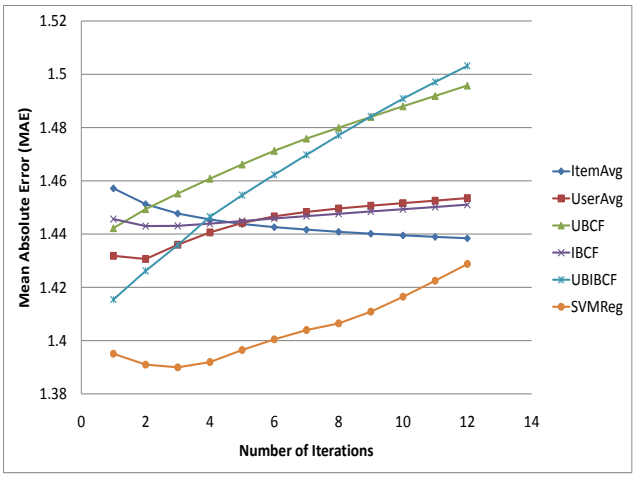


Figure 8: Comparing the proposed approaches with others in the case of iterative SVD (fixed dimension case), over FT5 dataset. X-axis shows the number of iterations and y-axis shows the corresponding MAE observed. The proposed approaches converge much quicker as compared to the conventional ones. The error bars (lying between 0.001 and 0.004 for all approaches) are not shown for reasons of clarity.

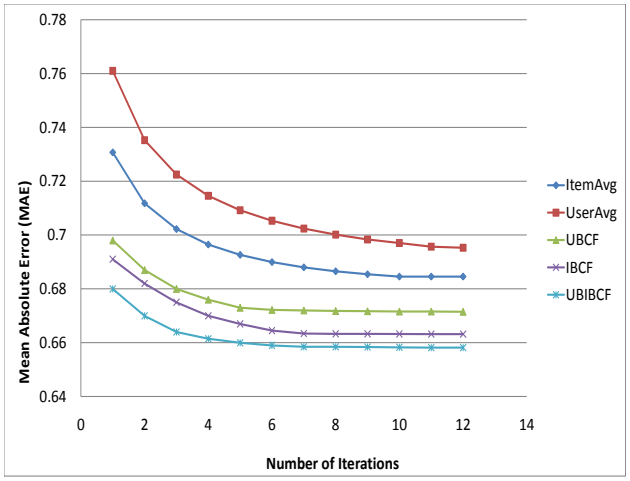


Figure 9: Comparing the proposed approaches with others in the case of iterative SVD (fixed dimension case), over ML dataset. X-axis shows the number of iterations and y-axis shows the corresponding MAE observed. The proposed approaches converges much quicker as compared to the conventional ones. The error bars (< 0.001 for all approaches) are not shown for reasons of clarity.

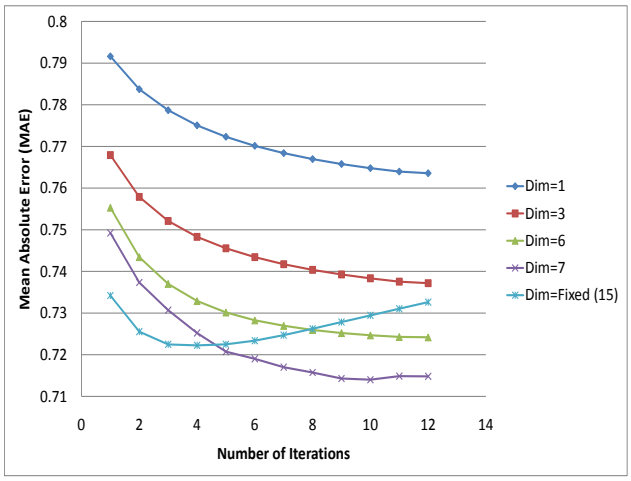


Figure 10: how the MAE changes with the increase in the number of dimensions for SVM regression approach, over SML dataset. X-axis shows the number of iterations and y-axis shows the corresponding MAE observed. Fixed dimensions represent the case, where the optimal numbers of dimensions are learned in the first iteration through the training set and kept fixed for all iterations. We observe that the results are highly dependent on the dimension parameter. The error bars (< 0.001 for all approaches) are not shown for reasons of clarity.

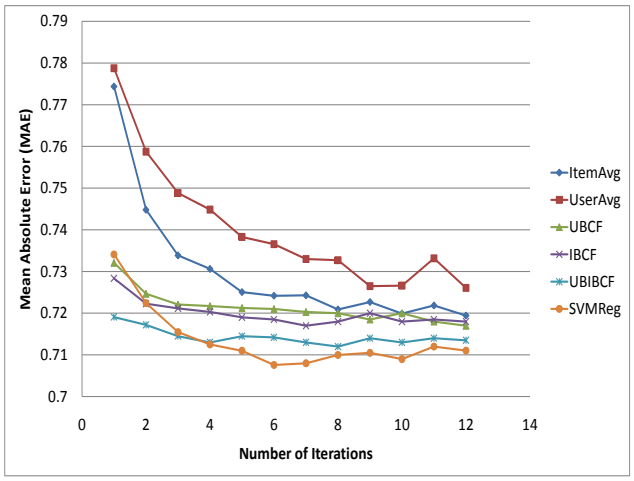


Figure 11: Comparing the proposed approaches with others in the case of iterative SVD (variable dimension case) over SML dataset. X-axis shows the number of iterations and y-axis shows the corresponding MAE observed. The proposed approaches converge much quicker as compared to the conventional ones. The error bars (< 0.001 for all approaches) are not shown for reasons of clarity.

pared to the conventional approaches, even in the first iteration and it converges much faster than conventional approaches. The IBCF and UBIBCF converges after $2-3$ whereas the UBCF and SVMReg converges after $5-6$ iterations, and then the MAE starts increasing, which may be due to the over-fitting. We further observe that approximating the missing values using CF gives better results as compared to the SVM regression.

The performance of the baseline approach is even worse for FT1 dataset. Figure 7 shows that in the case of ItemAvg, the MAE is 1.7 at first iteration, keeps on decreasing until it reaches at its minimum to 1.66 after $3-4$ iterations. The algorithm converges after 3 to 4 iterations and then the MAE starts increasing again. IBCF and SVMReg approaches show the similar behaviour, where the MAE reaches at its minimum after $4-5$ iterations, and then starts increasing again. The remaining approaches do not show any improvement in the MAE with an increase in the number of iterations. We note that the SVMReg imputation approach gives more accurate results with MAE = 1.40.

The results in the case of FT5 dataset are shown in Figure 8. The results of the baseline approach are surprisingly good where the MAE keeps on decreasing, until it converges after $10-12$ iterations. The lowest MAE observed after 12 iterations is still higher than the ones obtained in first iteration of the proposed approaches. The MAE in the case of UBCF and UBIBCF imputation approaches, increases with the increase in the number of iterations, which might be due to over-fitting. The MAE in the case of remaining imputation approaches decreases with the increase in number of iterations, reaches at its minimum at $2-3$ iterations and then starts increasing. Again, the SVMReg imputation approach gives more accurate results.

The results in case of ML dataset are shown in Figure 9. We observe that they show the similar behaviour as in the case of SML dataset. Based on the experimental results, we can conclude that the proposed approaches produce anytime [100] recommendations and converge much faster than the conventional ones. It is worth noting that computing SVD is very computation expensive (regardless it is done off-line), which implies finding the solution in the iterative SVD using the baseline approach is not pragmatic, and hence proposed approaches should be used to save time and memory.

The optimal number of dimensions can be learned at each iteration using the training set, though it is very expensive however, it may increase accuracy. To check how the MAE changes with the dimension parameter, we show results in the case of SVMReg imputation approach over SML dataset for different number of dimensions. *"Dim=Fixed (15)"* represents the case, where we learn the optimal number of dimensions through the training set and then these dimensions are kept fixed for all iterations. We also consider other cases, where we (randomly) choose dimension parameter to be 1, 3, 5, and 7. Figure 10 shows that the MAE is highly dependent on the dimension parameter. To further investigate the results, we perform experiments where the optimal numbers of dimensions are learned at each iteration. We only did experiments with SML and FT datasets, as we found it very expensive for ML dataset, both in terms of memory requirements and computation cost.

There were no clear improvement (and patterns) in results to be discussed in the case of FT5 dataset, may be due to the reason that we do not have enough data to learn the optimal parameters. We discuss the results in the case of SML data set, though they show the similar behaviour for FT1 dataset. We choose SML dataset as it has heavily been used in the literature and it is easy to reproduce the results. The results[13] are shown in Figure 11.

Figure 11 shows that learning the optimal number of dimensions at each iteration decreases the MAE of all approaches in general. Furthermore, all approaches except the SVMReg show the same behaviour as shown by the fixed iteration case. The MAE in the case of SVMReg approach keeps on decreasing with the increase in the number of iterations, reaches at its minimum at iteration 6, and then either stays stable or increases again. We further observe that the SVMReg outperform others, which is not true in fixed iteration case.

Table 8 compares the performance—in terms of MAE—of different approaches under the iterative SVD. The optimal number of dimensions are learned in the first iterations and kept fixed. We observe that the proposed approaches produce better results than the baseline approach. We further observe that, in the case of MovieLens dataset, the

[13]Note that we used the training data to estimate the best parameters and used an independent test set to give the unbiased estimate of the generalisation error.

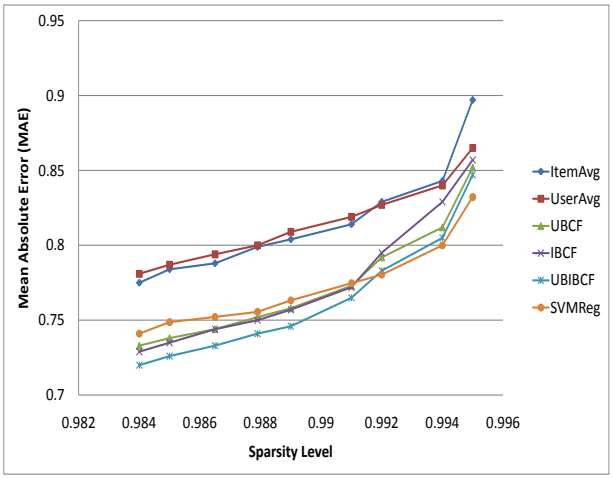


Figure 12: How sparsity affects the performance of different approaches, for SML dataset. Algorithm 1 was used to make recommendations.

UBIBCF and IBCF approaches outperform others; whereas in the case of FilmTrust dataset, the SVMReg and UBCF perform the best. The performance comparisons in terms of ROC-sensitivity and F1 measure are given in Appendix.

8.6 Performance evaluation under different sparsity levels

To check the performance of the proposed approaches under sparsity, we increased the sparsity level of the training set by dropping some randomly selected rating records. Whereas, we kept the test set same for each sparse training set. We used algorithm 1 to make recommendations. Figure 12 shows how different approaches perform under sparse conditions. The figure shows that the performance of the conventional approaches suffer more than the proposed ones. The reason is that under sparse conditions, the item and user averages might be misleading resulting in erroneous recommendations.

It must be noted that under very sparse conditions (sparsity ≥ 0.994), SVMReg outperforms the rest. It is because, with an increase in the sparsity, we do not have comprehensive user/item rating profile that can be used to make predictions for other unknown items. However, we can capture user profile in terms of the important features in which a user is interested, resulting in improved user profile and predictions. We also note that, the remaining approaches give the equivalent results. Hence, under very sparse condition, the SVMReg can be used provided enough resources are available, and the conventional approaches can be used otherwise.

8.7 Performance evaluation under cold-start scenarios

8.7.1 New user cold-start scenario

For testing the performance of approaches under new user cold-start scenario, we selected 50 random users, and kept their number of ratings in the training set to 2, 5, 10, 15, and 20. The corresponding *MAE*; represented by *MAE*2, *MAE*5, *MAE*10, *MAE*15, and *MAE*20 is shown in Table 9. Table 9 shows that the conventional approaches suffer the most under this scenario. It is worth noting that, when a user has rated less than (or equal to) 10 movies, then UserItemAvg gives the best results; however, as a user rates more items, the CF and SVMReg give reliable recommendation as shown by the table.

8.7.2 New item cold-start scenario

For testing the performance of approaches under new item cold-start scenario, we selected 50 random items, and kept the number of users in the training set who have rated the these item to 2, 5, 10, 15, and 20. The corresponding *MAE*; represented by *MAE*2, *MAE*5, *MAE*10, *MAE*15, and *MAE*20 is shown in Table 10. Table 10 shows that the SVMReg gives the best performance when an item has been rated by less than (or equal) to 10 items, and UBCF gives the best performance otherwise. We note that the IBCF does not perform very well as compared to the UBCF, the reason is we do not have comprehensive item rating profiles for finding other similar items. The reason for the good results in the case of SVMReg is the same as discussed in Section 8.6.

8.8 Performance evaluation under long tail scenario

To test the performance of the proposed algorithms under long tail scenario, we created the artificial long tail scenario by randomly selecting the 80% of items in the tail. The number of ratings given in the tail part were varied between 2, 4, 6, 8, and 10. The results, shown in Table 11, demonstrated the similar behaviour as in the case of new item case.

8.9 Performance evaluation under different training and test sizes

We performed experiments with different sizes of the test and train set by randomly dividing the rating records into $X\%$ training set and a $(100-X)\%$ test set. A value of $X = 20\%$ for SML dataset indicates that 100 000 ratings have been divided into 20 000 train cases and 80 000 test cases. Table 12 shows that the proposed approaches outperform others at each value of X. We note that the SVMReg gives the best performance for smaller training set sizes. The reason is the same as discussed in Section 8.6.

Table 8: Comparing the MAE observed in different imputation approaches under the iterative SVD (fixed dimension case). Only the conventional approaches and the approaches which gave the best results are compared. The best results have been shown in bold.

Imputation Source	**Best MAE**			
	ML	**SML**	**FT**1	**FT**5
ItemAvg	0.685 ± 0.002	0.738 ± 0.002	1.661 ± 0.003	1.438 ± 0.012
UserAvg	0.697 ± 0.002	0.734 ± 0.002	1.451 ± 0.003	1.430 ± 0.005
UBCF	0.672 ± 0.002	0.723 ± 0.002	1.450 ± 0.003	1.442 ± 0.013
IBCF	0.664 ± 0.002	0.722 ± 0.003	1.448 ± 0.003	1.442 ± 0.017
UBIBCF	$\mathbf{0.659 \pm 0.002}$	$\mathbf{0.715 \pm 0.002}$	1.436 ± 0.003	1.418 ± 0.013
SVMReg	$--$	0.721 ± 0.003	$\mathbf{1.401 \pm 0.003}$	$\mathbf{1.390 \pm 0.015}$

Table 9: Comparing MAE observed in different imputation approaches under **new-user cold start scenario**, for SML dataset. Only the conventional approaches and the approaches, which gave the best results are compared. The best results have been shown in bold.

Imp. Sr.	**Best MAE**				
	MAE2	**MAE**5	**MAE**10	**MAE**15	**MAE**20
ItemAvg	0.908 ± 0.041	0.887 ± 0.042	0.885 ± 0.041	0.883 ± 0.041	0.882 ± 0.041
UserAvg	1.087 ± 0.049	0.928 ± 0.044	0.903 ± 0.043	0.878 ± 0.043	0.877 ± 0.042
UserItemAvg	$\mathbf{0.901 \pm 0.041}$	$\mathbf{0.855 \pm 0.040}$	$\mathbf{0.850 \pm 0.040}$	0.843 ± 0.040	0.839 ± 0.040
UBCF	1.080 ± 0.049	0.886 ± 0.043	0.865 ± 0.043	0.841 ± 0.042	0.825 ± 0.041
IBCF	1.082 ± 0.050	0.896 ± 0.043	0.868 ± 0.043	0.844 ± 0.043	0.817 ± 0.042
UBIBCF	1.071 ± 0.049	0.891 ± 0.043	0.862 ± 0.042	$\mathbf{0.837 \pm 0.043}$	$\mathbf{0.816 \pm 0.041}$
SVMReg	0.962 ± 0.050	0.912 ± 0.044	0.873 ± 0.043	0.841 ± 0.042	0.836 ± 0.042

Table 10: Comparing MAE observed in different imputation approaches under **new-item cold start scenario**, for SML dataset. Only the conventional approaches and the approaches which gave the best results are compared. The best results have been shown in bold.

Imp. Sr.	**Best MAE**				
	MAE2	**MAE**5	**MAE**10	**MAE**15	**MAE**20
ItemAvg	1.010 ± 0.064	0.876 ± 0.052	0.854 ± 0.054	0.840 ± 0.056	0.838 ± 0.056
UserAvg	0.876 ± 0.059	0.874 ± 0.057	0.872 ± 0.058	0.870 ± 0.058	0.867 ± 0.057
UserItemAvg	0.865 ± 0.056	0.833 ± 0.054	0.832 ± 0.052	0.824 ± 0.055	0.822 ± 0.054
UBCF	0.911 ± 0.061	0.829 ± 0.052	0.810 ± 0.053	$\mathbf{0.800 \pm 0.059}$	$\mathbf{0.790 \pm 0.053}$
IBCF	0.840 ± 0.059	0.834 ± 0.059	0.829 ± 0.060	0.818 ± 0.056	0.813 ± 0.058
UBIBCF	0.858 ± 0.060	0.824 ± 0.053	0.812 ± 0.057	0.802 ± 0.0056	0.795 ± 0.055
SVMReg	$\mathbf{0.822 \pm 0.062}$	$\mathbf{0.815 \pm 0.052}$	$\mathbf{0.809 \pm 0.056}$	0.804 ± 0.057	0.802 ± 0.057

Table 11: Comparing the MAE observed in different imputation methods under the **long tail scenario**, for the SML dataset. The best results are shown in bold font.

Imp. Sr.	**Best MAE**				
	MAE2	**MAE**4	**MAE**6	**MAE**8	**MAE**10
ItemAvg	1.090 ± 0.003	0.891 ± 0.003	0.879 ± 0.003	0.867 ± 0.003	0.861 ± 0.003
UserAvg	0.881 ± 0.003	0.878 ± 0.003	0.869 ± 0.003	0.866 ± 0.003	0.865 ± 0.002
UserItemAvg	0.884 ± 0.003	0.882 ± 0.003	0.871 ± 0.003	0.863 ± 0.003	0.861 ± 0.003
UBCF	0.881 ± 0.003	0.874 ± 0.003	0.847 ± 0.003	0.838 ± 0.003	0.819 ± 0.002
IBCF	0.886 ± 0.003	0.875 ± 0.003	0.861 ± 0.003	0.860 ± 0.003	0.856 ± 0.003
UBIBCF	0.882 ± 0.003	0.869 ± 0.003	0.844 ± 0.003	0.836 ± 0.003	0.824 ± 0.002
SVMReg	$\mathbf{0.879 \pm 0.002}$	$\mathbf{0.865 \pm 0.002}$	$\mathbf{0.842 \pm 0.002}$	$\mathbf{0.833 \pm 0.002}$	$\mathbf{0.817 \pm 0.002}$

Table 12: Comparing MAE observed in different imputation approaches under **varying training set sizes**, for SML dataset. Only the conventional approaches and the approaches which gave the best results are compared. The best results have been shown in bold.

Imp. Sr.	**Best MAE**			
	$X = 20\%$	$X = 40\%$	$X = 60\%$	$X = 80\%$
ItemAvg	0.838 ± 0.004	0.809 ± 0.005	0.788 ± 0.005	0.774 ± 0.002
UserAvg	0.839 ± 0.003	0.818 ± 0.005	0.792 ± 0.005	0.778 ± 0.002
UserItemAvg	0.798 ± 0.003	0.784 ± 0.005	0.767 ± 0.005	0.754 ± 0.002
UBCF	0.807 ± 0.004	0.766 ± 0.005	0.746 ± 0.006	0.732 ± 0.003
IBCF	0.804 ± 0.004	0.762 ± 0.005	0.740 ± 0.006	0.730 ± 0.003
UBIBCF	0.802 ± 0.005	0.760 ± 0.005	$\mathbf{0.733 \pm 0.006}$	$\mathbf{0.721 \pm 0.003}$
SVMReg	$\mathbf{0.796 \pm 0.005}$	$\mathbf{0.756 \pm 0.006}$	0.748 ± 0.007	0.736 ± 0.003

Table 13: A comparison of the proposed algorithm with existing in terms of cost (based on [13]) and accuracy metrics. The SMVReg is used for FilmTrust dataset and the UBIBCF is used for the remaining datasets as imputation source, prior applying the SVD. t represents the number of iterations in the EM algorithm.

Algorithm	**Off-line Cost**	**On-line Cost**	**Best *MAE***			
			ML	**SML**	**FT**1	**FT**5
User-based CF with DV	$O(NM^2)$	$O(NM)$	0.706	0.746	1.442	1.416
Item-based CF	$O(N^2M)$	$O(N^2)$	0.705	0.744	1.433	0.418
Baseline SVD	$O(M)$	$O(1)$	0.730	0.774	1.700	1.483
Baseline Item -Based SVD	$O(M)$	$O(N^2)$	0.741	0.781	1.702	1.522
$ImpSvd$	$O(M^2N)+$ $O(N^2M)+O(N^3)$	$O(1)$	0.682	0.718	1.411	1.396
$ImpSvd^{ib}_{CF}$	$O(M^2N)+$ $O(N^2M)+O(N^3)$	$O(N^2)$	0.691	0.723	1.417	0.404
$ImpSvd^{ub}_{CF}$	$O(M^2N)+$ $O(N^2M)+O(N^3)$	$O(NM)$	0.692	0.722	1.416	0.401
$ImpSvd^{hybrid}_{CF}$	$O(M^2N)+$ $O(N^2M)+O(N^3)$	$O(NM)$	0.684	0.717	1.409	1.394
$ItrSvd$	$t*O(M^2N)+$ $O(N^2M)+O(N^3)$	$O(1)$	**0.659**	**0.715**	**1.401**	**1.390**

8.10 A comparison of the proposed algorithms with others

8.10.1 Direct comparison

We compared our algorithms with four different algorithms: user-based CF with default voting propose in [16] (shown by *User-Based CF with DV* in Table 13), item-based CF propose in [11] (shown by *Item-Based CF* in Table 13), a simple SVD based approach proposed in [12] (shown by *Baseline SVD* in Table 13), an item-based CF approach applied over the reduced user-item rating matrix, proposed in [46] (shown by Baseline Item-Based SVD in Table 13). Furthermore, we tuned all algorithms for the best mentioning parameters. For the proposed algorithms, we used UBIBCF and SVMReg as imputation sources in MovieLens and FilmTrust dataset respectively.

Table 13 shows the cost of the proposed algorithms and others with the corresponding lowest MAE. The table shows that the proposed algorithms are scalable and practical as they have on-line cost less than or equal to the cost of other algorithms; however they give much lower MAE. It must be noted that, the baseline algorithms do not perform very well as compared to the user-based and item-based CF applied over the original user-item rating matrix, which is in contrast with the work proposed in [13][14]. The proposed $ItrSvd$ algorithm performs the best out of all of them; however, it would incur the biggest off-line cost (depending on the number of iterations required to converge), and must be used given the availability of sufficient resources. The same is true for the $ImpSvd_{CF}^{hybrid}$, which gives more accurate results compared to baseline or simple CF; however, it would incur the greater cost. The $ImpSvd$ algorithm comes the next, and can be used if we want the lowest off-line cost (as SVD is applied only once), fast on-line performance, and prefer (good) accuracy.

8.10.2 Indirect comparison

In this section, we compare our results with other algorithms indirectly, i.e. we take the results[15] from the respective papers without re-implementing them, which might make the comparison less than ideal. The test procedures used in these papers are different from ours. We used a standard approach of testing using 5 fold-cross validation.

A comparison in terms of Normalised MAE (NMAE)[16] of the algorithms is given in Table 14. In Table 14, the URP represents the algorithm proposed in [101], Attitude represents the algorithm proposed in [102], MatchBox is proposed in [103], MMMF represents the maximum margin matrix factorisation algorithm proposed in [104], Item has been proposed in [105], E-MMF represents the ensemble maximum margin matrix factorisation technique proposed in [106], and NLMF represents the non-linear matrix factorisation technique (with linear and RBF versions) as proposed in [107].

Table 14: Comparing the NMAE (Normalized MAE) observed in different algorithms for the ML dataset. The proposed algorithms outperforms URP [101], Attitude [102], MatchBox [103], and MMMF [104]. They give the comparable results to Item [105], E-MMF [106], and NLMF [107]. Our results and the best results have been shown in bold.

Algorithm	**NMAE**
URP	0.4341 ± 0.0023
Attitude	0.4320 ± 0.0055
MatchBox	0.4206 ± 0.0055
MMMF	0.4156 ± 0.0037
$ItrSvd$	$\mathbf{0.4118 \pm 0.0025}$
Item	0.4096 ± 0.0029
E-MMF	0.4029 ± 0.0027
NLMF Linear	0.4052 ± 0.0011
NLMF RBF	$\mathbf{0.4026 \pm 0.0020}$

Table 14 shows that the NLMF, E-MMF, and Item perform better than the rest. The proposed hybrid algorithm gives comparable results to them with NMAE = 0.4118. It is worth mentioning that Item [105], E-MMF [106], and NLMF [107] employ extensive parameters learning, for instance the E-MMF is an ensemble of about 100 predictors, which makes this algorithms unattractive. From this table, we may conclude that the proposed algorithm is comparable to the state-of-the-art algorithm for the ML dataset.

9 When and how much imputation is required

As it is costly to do imputation by the proposed approaches, hence we investigate when it is beneficial to switch to the conventional approaches, which are cheap to compute. Next, we shed light on the following two questions: (1) when is imputation required? and (2) how much imputation is required?

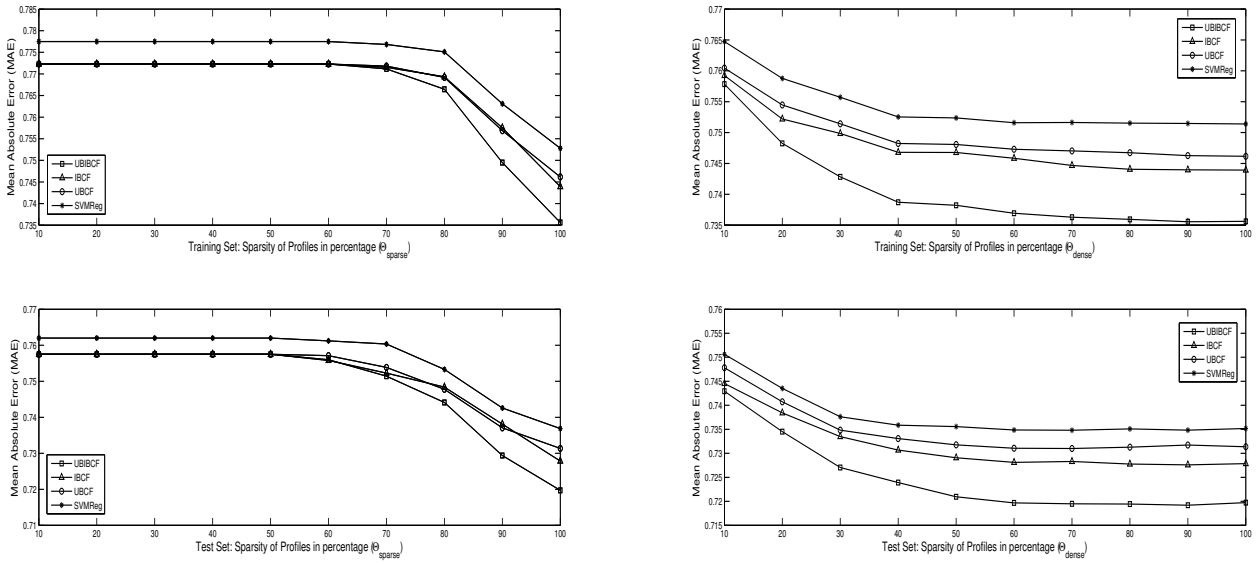
9.1 When to do imputation by the proposed approaches

To answer this question, we look into the sparsity of users' and items' profiles. We only do imputation by the proposed approaches when a user's (or item's) profile is $\Theta_{sparse}\%$ sparse, where $\Theta_{sparse} = \{10, 20, \cdots, 100\}$. A value of $\Theta_{Sparse} = 10$ shows that the proposed approaches are used to fill in the missing values if the sparsity of a profile is less than $10\% = 0.1$, and the UserItemAvg approach is used

[14] It might be due to the reason that, the author in [13] did not use any significance weighting schemes, and used weighted sum prediction formula [87] in the item-based CF.

[15] The results for the *weak generalisation* case were taken, as this procedure resembles with our test procedure.

[16] NMAE has been used in [102, 107], and is computed by normalizing the MAE by a factor. The value of the factor depends on the range of the ratings; for example for the MovieLens dataset, it is 1.6. For further information, refer to [107].



(a) When is imputation required?

(b) How much imputation is required?

Figure 13: Figures showing when and how much imputation is required for the SML dataset. Θ_{sparse} shows the sparsity of users' (or items') profiles in percentage. Θ_{dense} shows the percentage up to which users' (or items') profiles are filled using the proposed approaches. The optimal number of dimensions have been kept the same as shown in Table 4.

otherwise. Figure 13(a) shows that the MAE is minimum at $\Theta_{sparse} = 100$. For the subsequent experiments, we choose to do imputation when $\Theta_{sparse} = 100$.

9.2 How much imputation is required

Users' (or items') profiles can be filled up to $\Theta_{dense}\%$ of the missing values in their profiles. To investigate how much imputation is necessary, we performed experiments with different values of Θ_{dense} and observed the corresponding MAE. A value of $\Theta_{dense} = 10$ shows that 10% missing values of a profile are filled using the proposed approaches and UserItemAvg is used for the remaining 90% missing values. Figure 13(b) shows that after $\Theta_{desne} = 60$, the change in the MAE becomes very small. Hence, 60% imputation is sufficient to achieve good accuracy.

10 Discussion

What is evident from the experimental results is that the approximation of missing values in the sparse user-item rating matrix has an important role in SVD based recommendations. The literature proposes using item average to approximate the missing values in the sparse user-item rating matrix. We show that this is not a feasible solution in terms of accuracy. Moreover, the convergence is very slow in the case of conventional iterative SVD. Approaches such as CF or SVM should be used for this purpose.

We note that the imputation approaches based on the content-based filtering are not very accurate as compared to the collaborative filtering ones in the recommender system domain, though content-based filtering has successfully been applied to text categorisation and it gives accurate results as well. The reason is that the text categorisation and recommender system problems are quite different from each other. First, a user rates the same item differently under different context [108] and the reason of rating might be complex. Similarly, the positive feedback [109] given by a user, e.g. *purchased an item* is dependent on the context; for example, a user might purchase an item as a gift, hence we cannot predict that they will purchase other similar items. second, the user feedback [26] in a recommender system is noisy, the observations, *did not buy an item*, or *did not watch a movie* do not necessarily mean that the user is not interested in that item or movie. It can be the case, that user like that item or movie but has not purchased or watched it. Third, the evaluation criteria for both is different, recommender system usually provides a list of top items a user would like to consume, whereas text categorisation classify a given document to set of pre-defined categorisation. Furthermore, in text categorisation a document belongs to a single or a very few categories, whereas a user in recommender system might be interested in a large number of different items. Fourth, a user might change their taste over time and this temporal change in profile is not shared by the text categorisation tasks. Making accurate recommendation given the noisy input is different and more difficult as compared to the text categorisation task.

We captured the user profiles in terms of the important

features, i.e. a user likes a certain types of movies such as horror, romantic etc. However, this assumption was not very correct as well. As most of the users love to watch movies that makes them simply feel good rather than strictly following the the same types of movies. Furthermore, taking into account the temporal property of datasets might improve the results.

It is worth noting that the SVM regression gives much better performance than other classification and regression approaches, though we have imbalanced dataset. Schemes dealing with the imbalanced data may increase the performance of SVM regression even further. In the case of SVM classification, we overcome this problem by assigning different penalties to classes according to the prior knowledge of users. The prior knowledge of a user is the fraction of the total number of ratings belonging to a class to the total number of ratings provided by the user in the training set. Over sampling and under sampling [64] can be used to check the performance of SVM regressions, which is a subject of future research.

Based on the experimental results, we can underline five interesting points: (1) the results of different approaches are dataset dependent and no approach is a panacea. Due to dataset characteristic—data distribution, scale, and sparsity—one approach might be very good for one dataset while might fail to produce good results for the second dataset, (2) collaborative filtering and SVM provide more accurate and much more computationally tractable results under all experiments: the iterative SVD, simple imputed SVD, CF applied over reduced dataset, and in SVD under sparse settings (3) although, conventional approaches are straight forward to implement, they do not provide good results. The same is true for many classification and regression approaches, (4) the hybrid recommender system algorithms provide more accurate recommendations than the individual ones. Different recommendation algorithms, if combined in a systemic way, have complementary role for recommendation generation, (5) different imputation schemes can be chosen depending the different circumstances and priorities—time and frequency of running the off-line computation, required accuracy, required recommendation time, and available resources (for example, the content features and memory).

11 Conclusion and future work

Recommender systems play an important role in identifying the interesting items for users and try to solve the problem of information overload. This paper makes significant contributions to the state-of-the-art in two areas of recommender systems, namely, SVD based recommendation algorithms, and the hybrid recommendation algorithms.

There has been some work, in the literature to overcome the scalability problem of recommender system using SVD; however due to sparsity it leads to poor quality recommendation. We show how both scalability and accuracy problems can be eliminated by using a suitable imputation source, as a pre-processing step, with SVD. We have shown by empirical study that rather than merely using the item average of user-item rating matrix as imputation, or ignoring the missing values, which have been the preferred approaches in the literature, flexible and robust imputation approaches gives considerable benefits ranging from cost saving to performance enhancement, and therefore, should be used prior to applying SVD over user-item rating matrix. We further show how the results of CF, when applied over the dataset reduced by SVD, change with the imputation source.

An important research issue in recommender system is that the recommendations should be tailored to the user's current information seeking task [110]. In this paper, we consider the two-dimensional Users $\times$ Items space, by recommending items to users based on the information only about users and items. It has been claimed that taking the additional context information (such as time, place, and the company of a user) into account, either by extending the user-item rating matrix into multiple dimensions or using reduction-based recommendation approach, might increase the performance of the recommender systems [111]. This multi-criteria ratings data would be very sparse as compared to the traditional user-item rating matrix, and drawing supplementary information from the content or demographic data of user/item might help reducing the sparsity of data matrix, which makes our imputation sources even more attractive in these scenarios. Keeping these promising results as starting point, we are focusing on the multi-dimensional dataset, where clustering algorithms can be used for partitioning the data and imputed SVD can be applied to reduce the sparsity and dimensional of the resulting partition.

Another avenue for future work would be to incorporate external sources of information, such as ontology of items, which may improve the results, particularly under sparse conditions. Furthermore, we would like to explore in detail the questions discussed in Section 9.

Acknowledgment

The work reported in this paper has formed part of the Instant Knowledge Research project which is jointly funded by Mobile VCE, (the Virtual Centre of Excellence in Mobile & Personal Communications, www.mobilevce.com), UK Technology Strategy Board (TSB), and EPSRC (Engineering and Physical Sciences Research Council).

12 Performance comparison of different imputation approaches

Tables 15 and 16 compare the performance (in terms of ROC-sensitivity and F1 measure respectively) of different approaches under the iterative SVD. We observe that

Table 15: Comparing the ROC observed in different imputation approaches under the iterative SVD (fixed dimension case). Only the conventional approaches and the approaches which gave the best results are compared. The best results have been shown in bold.

Imputation Source	Best ROC-sensitivity			
	ML	**SML**	**FT**1	**FT**5
ItemAvg	0.685 ± 0.003	0.651 ± 0.005	0.504 ± 0.012	0.569 ± 0.011
UserAvg	0.721 ± 0.002	0.683 ± 0.009	0.572 ± 0.011	0.571 ± 0.013
UBCF	0.724 ± 0.002	0.691 ± 0.005	0.546 ± 0.011	0.530 ± 0.012
IBCF	$\mathbf{0.759 \pm 0.002}$	$\mathbf{0.724 \pm 0.012}$	0.534 ± 0.011	0.544 ± 0.016
UBIBCF	0.747 ± 0.002	0.711 ± 0.004	0.517 ± 0.011	0.563 ± 0.009
SVMReg	$--$	0.695 ± 0.007	$\mathbf{0.574 \pm 0.013}$	$\mathbf{0.583 \pm 0.014}$

Table 16: Comparing the Top-N F1 (computed over top-20 recommendations) observed in different imputation approaches under the iterative SVD (fixed dimension case). Only the conventional approaches and the approaches which gave the best results are compared. The best results have been shown in bold.

Imputation Source	Best **F**1			
	ML	**SML**	**FT**1	**FT**5
ItemAvg	0.445 ± 0.004	0.481 ± 0.005	0.486 ± 0.012	0.547 ± 0.008
UserAvg	0.463 ± 0.004	0.503 ± 0.007	0.540 ± 0.011	0.538 ± 0.014
UBCF	0.468 ± 0.004	0.514 ± 0.004	0.531 ± 0.012	0.520 ± 0.010
IBCF	$\mathbf{0.487 \pm 0.004}$	$\mathbf{0.531 \pm 0.009}$	0.505 ± 0.012	0.519 ± 0.012
UBIBCF	0.481 ± 0.004	0.528 ± 0.002	0.507 ± 0.012	0.534 ± 0.010
SVMReg	$--$	0.508 ± 0.005	$\mathbf{0.556 \pm 0.014}$	$\mathbf{0.563 \pm 0.014}$

the proposed approaches give better results than traditional ones.

References

[1] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[2] B. Mobasher, "Recommender systems," *KI*, vol. 21, no. 3, pp. 41–43, 2007.

[3] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: Item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, pp. 76–80, January 2003.

[4] J. B. Schafer, J. Konstan, and J. Riedi, "Recommender systems in e-commerce," in *Proceedings of the 1st ACM conference on Electronic commerce*, ser. EC '99. New York, NY, USA: ACM, 1999, pp. 158–166. [Online]. Available: http://doi.acm.org/10.1145/336992.337035

[5] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Commun. ACM*, vol. 35, pp. 61–70, December 1992. [Online]. Available: http://doi.acm.org/10.1145/138859.138867

[6] U. Shardanand and P. Maes, "Social information filtering: algorithms for automating word of mouth," in *Proceedings of the SIGCHI conference on Human factors in computing systems*, ser. CHI '95. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 210–217. [Online]. Available: http://dx.doi.org/10.1145/223904.223931

[7] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter, "Phoaks: a system for sharing recommendations," *Commun. ACM*, vol. 40, pp. 59–62, March 1997. [Online]. Available: http://doi.acm.org/10.1145/245108.245122

[8] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, ser. CSCW '94. New York, NY, USA: ACM, 1994, pp. 175–186. [Online]. Available: http://doi.acm.org/10.1145/192844.192905

[9] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl, "Grouplens: applying collaborative filtering to usenet news," *Commun. ACM*, vol. 40, pp. 77–87, March 1997. [Online]. Available: http://doi.acm.org/10.1145/245108.245126

[10] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory and model-based approach," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, ser. UAI '00.

San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 473–480.

[11] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the 10th international conference on World Wide Web*, ser. WWW '01. New York, NY, USA: ACM, 2001, pp. 285–295. [Online]. Available: http://doi.acm.org/10.1145/371920.372071

[12] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Application of dimensionality reduction in recommender system–a case study," in *IN ACM WEBKDD WORKSHOP*. Citeseer, 2000.

[13] M. Vozalis and K. G. Margaritis, "Using svd and demographic data for the enhancement of generalized collaborative filtering," *Information Sciences*, vol. 177, pp. 3017–3037, August 2007.

[14] M. Kurucz, A. Benczúr, and K. Csalogány, "Methods for large scale SVD with missing values," in *Proceedings of KDD Cup and Workshop*. Citeseer, 2007.

[15] M. A. Ghazanfar and A. Prügel-Bennett, "The advantage of careful imputation sources in sparse data-environment of recommender systems: Generating improved svd-based recommendations," in *IADIS European Conference on Data Mining*, July 2011.

[16] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, ser. UAI'98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 43–52.

[17] Y.-J. Park and A. Tuzhilin, "The long tail of recommender systems and how to leverage it," in *Proceedings of the 2008 ACM conference on Recommender systems*, ser. RecSys '08. New York, NY, USA: ACM, 2008, pp. 11–18. [Online]. Available: http://doi.acm.org/10.1145/1454008.1454012

[18] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering," in *Proceedings of the Fifth International Conference on Computer and Information Technology*, 2002.

[19] G.-R. Xue, C. Lin, Q. Yang, W. Xi, H.-J. Zeng, Y. Yu, and Z. Chen, "Scalable collaborative filtering using cluster-based smoothing," in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '05. New York, NY, USA: ACM, 2005, pp. 114–121. [Online]. Available: http://doi.acm.org/10.1145/1076034.1076056

[20] A. M. Rashid, S. K. Lam, G. Karypis, and J. Riedl, "Clustknn: a highly scalable hybrid model-& memory-based cf algorithm," in *Proc. of WebKDD 2006: KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2006), August 20-23 2006, Philadelphia, PA*. Citeseer, 2006.

[21] F. Wang, T. Li, and C. Zhang, "Semi-supervised clustering via matrix factorization," in *SDM*, 2008, pp. 1–12.

[22] M. A. Ghazanfar and A. Prügel-Bennett, "Fulfilling the needs of gray-sheep users in recommender systems, a clustering solution," in *2011 International Conference on Information Systems and Computational Intelligence*, January 2011. [Online]. Available: http://eprints.ecs.soton.ac.uk/21770/

[23] M. A. Ghazanfar, S. Szedmák, and A. Prügel-Bennett, "Incremental kernel mapping algorithms for scalable recommender systems," in *IEEE ICTAI*, 2011, pp. 1077–1084.

[24] K. Lang, "NewsWeeder: learning to filter netnews," in *Proceedings of the 12th International Conference on Machine Learning*. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1995, pp. 331–339.

[25] R. van Meteren and M. van Someren, "Using content-based filtering for recommendation," in *Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop*. Citeseer, 2000.

[26] M. J. Pazzani and D. Billsus, "The adaptive web," P. Brusilovsky, A. Kobsa, and W. Nejdl, Eds. Berlin, Heidelberg: Springer-Verlag, 2007, ch. Content-based recommendation systems, pp. 325–341. [Online]. Available: http://dl.acm.org/citation.cfm?id=1768197.1768209

[27] S. Alag, *Collective Intelligence in Action*. Manning Publications, October, 2008.

[28] P. Melville, R. J. Mooney, and R. Nagarajan, "Content-boosted collaborative filtering for improved recommendations," in *Eighteenth national conference on Artificial intelligence*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 2002, pp. 187–192.

[29] R. Burke, "Hybrid recommender systems: Survey and experiments," *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, November 2002.

[30] M. J. Pazzani, "A framework for collaborative, content-based and demographic filtering," *Artif. Intell. Rev.*, vol. 13, pp. 393–408, December 1999.

[Online]. Available: http://dx.doi.org/10.1023/A:1006544522159

[31] M. Claypool, A. Gokhale, T. Mir, P. Murnikov, D. Netes, and M. Sartin, "Combining content-based and collaborative filters in an online newspaper," in *In Proceedings of ACM SIGIR Workshop on Recommender Systems*. Berkeley, California: ACM, 1999.

[32] R. Burke, "Integrating knowledge-based and collaborative-filtering recommender systems," in *In AAAI Workshop on AI in Electronic Commerce*. AAAI, 1999, pp. 69–72.

[33] R. J. Mooney and L. Roy, "Content-based book recommending using learning for text categorization," in *Proceedings of the fifth ACM conference on Digital libraries*, ser. DL '00. New York, NY, USA: ACM, 2000, pp. 195–204. [Online]. Available: http://doi.acm.org/10.1145/336597.336662

[34] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, pp. 734–749, June 2005. [Online]. Available: http://dx.doi.org/10.1109/TKDE.2005.99

[35] M. A. Ghazanfar and A. Prügel-Bennett, "An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering," in *Lecture Notes in Engineering and Computer Science: Proceedings of The International Multi Conference of Engineers and Computer Scientists 2010*. IMECS 2010, 17–19 March, 2010, Hong Kong, 2010, pp. 493–502. [Online]. Available: http://eprints.ecs.soton.ac.uk/18483/

[36] ——, "A scalable, accurate hybrid recommender system," in *Proceedings of the 2010 Third International Conference on Knowledge Discovery and Data Mining*, ser. WKDD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 94–98. [Online]. Available: http://dx.doi.org/10.1109/WKDD.2010.117

[37] M. A. Ghazanfar, S. Szedmak, and A. Prugel-Bennett, "Incremental kernel mapping algorithms for scalable recommender systems," in *Proceedings of the 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, ser. ICTAI '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 1077–1084. [Online]. Available: http://dx.doi.org/10.1109/ICTAI.2011.183

[38] M. A. Ghazanfar, A. PrüGel-Bennett, and S. Szedmak, "Kernel-mapping recommender system algorithms," *Inf. Sci.*, vol. 208, pp. 81–104, Nov. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2012.04.012

[39] D. Billsus and M. J. Pazzani, "Learning collaborative information filters," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1998, pp. 46–54.

[40] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Investigation of various matrix factorization methods for large recommender systems," in *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, ser. NETFLIX '08. New York, NY, USA: ACM, 2008, pp. 6:1–6:8. [Online]. Available: http://doi.acm.org/10.1145/1722149.1722155

[41] D. Kim and B.-J. Yum, "Collaborative filtering based on iterative principal component analysis," *Expert Syst. Appl.*, vol. 28, pp. 823–830, May 2005. [Online]. Available: http://dx.doi.org/10.1016/j.eswa.2004.12.037

[42] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Incremental singular value decomposition algorithms for highly scalable recommender systems," in *Proceedings of the 5th International Conference in Computers and Information Technology*. Citeseer, 2002, pp. 27–28.

[43] R. J. A. Little and D. B. Rubin, "Statistical analysis with missing data," 1987.

[44] B. Marlin, R. Zemel, S. Roweis, and M. Slaney, "Collaborative filtering and the missing at random assumption," in *Uncertainty in Artificial Intelligence: Proceedings of the 23rd Conference (Submitted)*, vol. 47. Citeseer, 2007, pp. 50–54.

[45] M. G. Vozalis and K. G. Margaritis, "Applying svd on item-based filtering," in *Proceedings of the 5th International Conference on Intelligent Systems Design and Applications*, ser. ISDA '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 464–469. [Online]. Available: http://dx.doi.org/10.1109/ISDA.2005.25

[46] M. Vozalis and K. G. Margaritis, "Applying SVD on generalized item-based filtering," *International Journal of Computer Science and Applications*, vol. 3, no. 3, pp. 27–51, 2006.

[47] A. Martinez, J. Arias, A. Vilas, J. Garcia Duque, and M. Lopez Nores, "What's on tv tonight? an efficient and effective personalized recommender system of tv programs," *Consumer Electronics, IEEE Transactions on*, vol. 55, no. 1, pp. 286–294, 2009.

[48] A. B. Barragáns-Martínez, E. Costa-Montenegro, J. C. Burguillo, M. Rey-López, F. A. Mikic-Fonte, and A. Peleteiro, "A hybrid content-based and item-based collaborative filtering approach to

recommend tv programs enhanced with singular value decomposition," *Inf. Sci.*, vol. 180, pp. 4290–4311, November 2010. [Online]. Available: http://dx.doi.org/10.1016/j.ins.2010.07.024

[49] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, pp. 133–151, July 2001.

[50] Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia, "Spectral analysis of data," in *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, ser. STOC '01. New York, NY, USA: ACM, 2001, pp. 619–626. [Online]. Available: http://doi.acm.org/10.1145/380752.380859

[51] C. Do and S. Batzoglou, "What is the expectation maximization algorithm?" *Nature biotechnology*, vol. 26, no. 8, pp. 897–899, 2008.

[52] J. Canny, "Collaborative filtering with privacy via factor analysis," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '02. New York, NY, USA: ACM, 2002, pp. 238–245. [Online]. Available: http://doi.acm.org/10.1145/564376.564419

[53] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *ICML*, vol. 20, no. 2, 2003, p. 720.

[54] S. Zhang, W. Wang, J. Ford, F. Makedon, and J. Pearlman, "Using singular value decomposition approximation for collaborative filtering," in *Proceedings of the Seventh IEEE International Conference on E-Commerce Technology*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 257–264.

[55] S. Zhang, W. Wang, J. Ford, and F. Makedon, "Learning from incomplete ratings using non-negative matrix factorization," in *6th SIAM Conference on Data Mining (SDM)*. Citeseer, 2006, pp. 548–552.

[56] J. Bennett and S. Lanning, "The netflix prize," in *Proceedings of KDD Cup and Workshop*, vol. 2007. Citeseer, 2007.

[57] N. Srebro, J. D. M. Rennie, and T. Jaakkola, "Maximum-margin matrix factorization," *Advances in neural information processing systems*, vol. 17, pp. 1329–1336, 2005.

[58] R. Bell, Y. Koren, and C. Volinsky, "The BellKor solution to the Netflix prize, in: AT&T Labs–Research: Technical report November," 2007.

[59] M. Wu, "Collaborative filtering via ensembles of matrix factorizations," in *Proceedings of KDD Cup and Workshop*. Citeseer, 2007.

[60] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," *Advances in Neural Information Processing Systems*, vol. 20, pp. 1257–1264, 2008.

[61] G. Takács, I. Pilászy, B. Németh, and D. Tikk, "Scalable collaborative filtering approaches for large recommender systems," *J. Mach. Learn. Res.*, vol. 10, pp. 623–656, June 2009.

[62] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model," in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, ser. KDD '08. New York, NY, USA: ACM, 2008, pp. 426–434. [Online]. Available: http://doi.acm.org/10.1145/1401890.1401944

[63] R. M. Bell and Y. Koren, "Scalable collaborative filtering with jointly derived neighborhood interpolation weights," in *Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 43–52.

[64] I. H. W. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, October 1999.

[65] A. Paterek, "Improving regularized singular value decomposition for collaborative filtering," in *Proc. KDD Cup and Workshop*. Citeseer, 2007.

[66] M. Kurucz, A. Benczúr, T. Kiss, I. Nagy, A. Szabó, and B. Torma, "Who rated what: a combination of SVD, correlation and frequent sequence mining," in *Proc. KDD Cup and Workshop*, vol. 23. Citeseer, 2007, pp. 720–727.

[67] M. Piotte and M. Chabbert, "The pragmatic theory solution to the netflix grand prize, in: Netflix prize documentation," 2009.

[68] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, "Evaluating collaborative filtering recommender systems," *ACM Trans. Inf. Syst.*, vol. 22, pp. 5–53, January 2004. [Online]. Available: http://doi.acm.org/10.1145/963770.963772

[69] N. Good, J. B. Schafer, J. A. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, "Combining collaborative filtering with personal agents for better recommendations," in *Proceedings of the sixteenth national conference on Artificial intelligence and the eleventh Innovative applications of artificial intelligence conference innovative applications of artificial intelligence*, ser. AAAI '99/IAAI '99. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1999, pp. 439–446.

[70] S.-T. Park, D. Pennock, O. Madani, N. Good, and D. DeCoste, "Naive filterbots for robust cold-start recommendations," ser. KDD '06. New York, NY, USA: ACM, 2006, pp. 699–705. [Online]. Available: http://doi.acm.org/10.1145/1150402.1150490

[71] H. Ma, I. King, and M. R. Lyu, "Effective missing data prediction for collaborative filtering," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 39–46. [Online]. Available: http://doi.acm.org/10.1145/1277741.1277751

[72] J. Zhang and P. Pu, "A recursive prediction algorithm for collaborative filtering recommender systems," in *Proceedings of the 2007 ACM conference on Recommender systems*, ser. RecSys '07. New York, NY, USA: ACM, 2007, pp. 57–64. [Online]. Available: http://doi.acm.org/10.1145/1297231.1297241

[73] X. Su, T. M. Khoshgoftaar, X. Zhu, and R. Greiner, "Imputation-boosted collaborative filtering using machine learning classifiers," in *Proceedings of the 2008 ACM symposium on Applied computing*, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 949–950. [Online]. Available: http://doi.acm.org/10.1145/1363686.1363903

[74] X. Su, T. M. Khoshgoftaar, and R. Greiner, "A mixture imputation-boosted collaborative filter," in *Proceedings of the 21th International Florida Artificial Intelligence Research Society Conference (FLAIRS'08)*, 2008, pp. 312–317.

[75] C. Ryan, D. Greene, G. Cagney, and P. Cunningham, "Missing value imputation for epistatic MAPs," *BMC Bioinformatics*, vol. 11, no. 1, pp. 197+, 2010.

[76] M. Balabanović and Y. Shoham, "Fab: content-based, collaborative recommendation," *Commun. ACM*, vol. 40, pp. 66–72, March 1997. [Online]. Available: http://doi.acm.org/10.1145/245108.245124

[77] T. Joachims, "A probabilistic analysis of the rocchio algorithm with tfidf for text categorization," in *Proceedings of the Fourteenth International Conference on Machine Learning*, ser. ICML '97. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 143–151.

[78] B. M. Sarwar, J. A. Konstan, A. Borchers, J. Herlocker, B. Miller, and J. Riedl, "Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system," in *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, ser. CSCW '98. New York, NY, USA: ACM, 1998, pp. 345–354. [Online]. Available: http://doi.acm.org/10.1145/289444.289509

[79] B. Mobasher, X. Jin, and Y. Zhou, "Semantically Enhanced Collaborative Filtering on the Web," vol. 3209, pp. 57–76, Sep. 2003.

[80] S. M. David, D. C. D. Roure, and N. R. Shadbolt, "Capturing knowledge of user preferences: Ontologies in recommender systems," in *In Proceedings of the First International Conference on Knowledge Capture (K-CAP 2001), Oct 2001*. ACM Press, pp. 100–107.

[81] M. Szomszor, C. Cattuto, H. Alani, K. O'äôhara, A. Baldassarri, V. Loreto, and V. D. P. Servedio, "Folksonomies, the semantic web, and movie recommendation," in *Bridging the Gep between Semantic Web and Web 2.0 (SemNet 2007)*, 2007, pp. 71–84.

[82] I. Cantador, A. Bellogín, and P. Castells, "A multilayer ontology-based hybrid recommendation model," *AI Commun.*, vol. 21, pp. 203–210, April 2008.

[83] S.-S. Weng and H.-L. Chang, "Using ontology network analysis for research document recommendation," *Expert Syst. Appl.*, vol. 34, pp. 1857–1869, April 2008.

[84] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, ser. SIGIR '02. New York, NY, USA: ACM, 2002, pp. 253–260. [Online]. Available: http://doi.acm.org/10.1145/564376.564421

[85] M. W. Berry, S. T. Dumais, and G. W. O'Brien, "Using linear algebra for intelligent information retrieval," *SIAM Rev.*, vol. 37, pp. 573–595, December 1995.

[86] D. Scott C., D. Susan T., L. Thomas K., F. George W., and H. Richard A., "Indexing by latent semantic analysis," *Journal of the American society for information science*, vol. 41, no. 6, pp. 391–407, 1990.

[87] M. A. Ghazanfar and A. Prügel-Bennett, "Novel significance weighting schemes for collaborative filtering: Generating improved recommendations in sparse environments." in *DMIN*. CSREA Press, 2010, pp. 334–342.

[88] J. Herlocker, J. A. Konstan, and J. Riedl, "An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms," *Inf. Retr.*, vol. 5, pp. 287–310, October 2002.

[89] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May 2011. [Online]. Available: http://doi.acm.org/10.1145/1961189.1961199

[90] C. Hsu, C. Chang, C. Lin *et al.*, "A practical guide to support vector classification," 2003.

[91] A. Ramanan, S. Suppharangsan, and M. Niranjan, "Unbalanced decision trees for multi-class classification," in *IEEE - Second International Conference on Industrial and Information Systems, ICIIS 2007*. IEEE, August 2007, pp. 291–294. [Online]. Available: http://eprints.ecs.soton.ac.uk/21490/

[92] M. A. Ghazanfar and A. Prügel-Bennett, "Building Switching Hybrid Recommender System Using Machine Learning Classifiers and Collaborative Filtering," *IAENG International Journal of Computer Science*, vol. 37, no. 3, pp. 272–287, 2010.

[93] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," in *Proceedings of the 10th European Conference on Machine Learning*. London, UK: Springer-Verlag, 1998, pp. 137–142.

[94] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, November 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[95] M. Vozalis and K. G. Margaritis, "On the enhancement of collaborative filtering by demographic data," *Web Intelli. and Agent Sys.*, vol. 4, pp. 117–138, April 2006.

[96] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning, "Kea: practical automatic keyphrase extraction," in *Proceedings of the fourth ACM conference on Digital libraries*, ser. DL '99. New York, NY, USA: ACM, 1999, pp. 254–255. [Online]. Available: http://doi.acm.org/10.1145/313238.313437

[97] K. Aas and L. Eikvil, "Text categorisation: A survey." 1999.

[98] T. Zhang and V. S. Iyengar, "Recommender systems using linear classifiers," *J. Mach. Learn. Res.*, vol. 2, pp. 313–334, March 2002.

[99] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Analysis of recommendation algorithms for e-commerce," in *Proceedings of the 2nd ACM conference on Electronic commerce*, ser. EC '00. New York, NY, USA: ACM, 2000, pp. 158–167. [Online]. Available: http://doi.acm.org/10.1145/352871.352887

[100] M. A. Ghazanfar and A. Prügel-Bennett, "Novel heuristics for coalition structure generation in multi-agent systems," in *The 2010 International Conference of Computational Intelligence and Intelligent Systems*. ICCIIS'10, 30 June–2 July 2010, London, U.K., 2010. [Online]. Available: http://eprints.ecs.soton.ac.uk/18788/

[101] B. Marlin, "Modeling user rating profiles for collaborative filtering," *Advances in neural information processing systems*, vol. 16, pp. 627–634, 2004.

[102] ——, "Collaborative Filtering: A Machine Learning Perspective," Master's thesis, University of Toronto, 2004.

[103] D. H. Stern, R. Herbrich, and T. Graepel, "Matchbox: large scale online bayesian recommendations," in *Proceedings of the 18th international conference on World wide web*, ser. WWW '09. New York, NY, USA: ACM, 2009, pp. 111–120. [Online]. Available: http://doi.acm.org/10.1145/1526709.1526725

[104] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proceedings of the 22nd international conference on Machine learning*, ser. ICML '05. New York, NY, USA: ACM, 2005, pp. 713–719. [Online]. Available: http://doi.acm.org/10.1145/1102351.1102441

[105] S. Park and D. Pennock, "Applying collaborative filtering techniques to movie search for better ranking and browsing," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2007, pp. 550–559.

[106] D. DeCoste, "Collaborative prediction using ensembles of maximum margin matrix factorizations," in *Proceedings of the 23rd international conference on Machine learning*, ser. ICML '06. New York, NY, USA: ACM, 2006, pp. 249–256. [Online]. Available: http://doi.acm.org/10.1145/1143844.1143876

[107] N. D. Lawrence and R. Urtasun, "Non-linear matrix factorization with gaussian processes," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 601–608. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553452

[108] L. Baltrunas, "Exploiting contextual information in recommender systems," in *Proceedings of the 2008 ACM conference on Recommender systems*, ser. RecSys '08. New York, NY, USA: ACM, 2008, pp. 295–298. [Online]. Available: http://doi.acm.org/10.1145/1454008.1454056

[109] D. Oard and J. Kim, "Implicit feedback for recommender systems," in *Proceedings of the AAAI Workshop on Recommender Systems*, 1998, pp. 81–83.

[110] S. M. McNee, "Meeting user information needs in recommender systems," Ph.D. dissertation, UNIVERSITY OF MINNESOTA, USA, 2006.

[111] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, pp. 103–145, January 2005. [Online]. Available: http://doi.acm.org/10.1145/1055709.1055714