Expert Systems With Applications 115 (2019) 427–441

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# Boosting collaborative filtering with an ensemble of co-trained recommenders

Arthur F. da Costa[a,*], Marcelo G. Manzato[a], Ricardo J.G.B. Campello[b,a]

[a] Institute of Mathematical and Computer Science, University of São Paulo, Brazil
[b] School of Mathematical & Physical Sciences, University of Newcastle, Australia

ARTICLE INFO

*Article history:*
Received 14 December 2017
Revised 18 July 2018
Accepted 11 August 2018
Available online 13 August 2018

*Keywords:*
Co-training
Ensembles
Recommender systems
Semi-supervised learning

ABSTRACT

Collaborative Filtering (CF) is one of the best performing and most widely used approaches for recommender systems. Although significant progress has been made in this area, current CF methods still suffer from cold-start and sparsity problems. A primary issue is that the fraction of users willing to rate items tends to be very small in most practical applications, which causes the number of users and/or items with few or no interactions in recommendation databases to be large. As a direct consequence of ratings sparsity, recommender algorithms may provide poor recommendations (reducing accuracy) or decline recommendations (reducing coverage). This paper proposes an ensemble scheme based on a co-training approach, named ECoRec, that drives two or more recommenders to agree with each others' predictions to generate their own. The experiments on eight real-life public databases show that better accuracy can be obtained when recommender algorithms are simultaneously trained from multiple views and combined into an ensemble to make predictions.

## 1. Introduction

In recent years, the interest in recommender systems has drastically increased, from both researchers and practitioners, with the explosive growth and variety of information available on the Web (Aggarwal, 2016; Bobadilla, Ortega, Hernando, & Gutiérrez, 2013; Ricci, Rokach, & Shapira, 2015). In this field, Collaborative Filtering (CF) approaches, especially matrix factorization and neighborhood models, have achieved significant improvements (Aggarwal, 2016; Bobadilla et al., 2013). These approaches operate based on users' previous interests encoded by the ratings matrix reflecting the similarities of users or items. However, they underperform when little collaborative information is available, which is related to known traditional recommender problems, such as cold-start and sparsity.

In many real-life applications, both the number of items and the number of consumers are large. In such cases, even when many events have been recorded, the consumer-product interaction matrix can still be extremely sparse, that is, there are very few elements in the matrix whose value is known. This problem, commonly referred to as the sparsity problem, has a major negative impact on the effectiveness of collaborative filtering approaches (Bobadilla et al., 2013). Because of sparsity, it is very likely that the similarity (or correlation) between two given users cannot be properly measured or may not be reliable, rendering poor collaborative filtering-based recommendations (Aggarwal, 2016; Bobadilla et al., 2013).

Another problem, known as cold-start, regards recommendations for new (or rarely rated) items or new (or inactive) users (Aggarwal, 2016; Bobadilla et al., 2013). For instance, as opposed to popular items, it is difficult for standard recommendation approaches to provide high-quality recommendations involving newly released items or items that are not so popular, for which very few ratings are available.

From the machine learning perspective, the cold-start and sparsity problems have a common root in the shortage of labeled data to train recommender algorithms in a traditional, fully supervised (i.e., informed/guided) way. In the realm of collaborative filtering, "labeled" data means known user/item interactions, such as ratings. However, very often labeled data are limited and expensive to obtain, since labeling typically requires users' feedback or human expertise (e.g. conducting user study). It can be even more critical if one wants to train personalized models, which requires large amounts of labeled data from each individual.

In order to minimize these problems, semi-supervised learning (SSL) has been adopted in recommender systems to boost learning performance by simultaneously using labeled and unlabeled data (Karimi, Freudenthaler, Nanopoulos, & Schmidt-Thieme, 2011;

[*] Corresponding author.
*E-mail addresses:* fortes@icmc.usp.br (A.F. da Costa), mmanzato@icmc.usp.br (M.G. Manzato), ricardo.campello@newcastle.edu.au (R.J.G.B. Campello).

https://doi.org/10.1016/j.eswa.2018.08.020
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

Zhang, Tang, Zhang, & Xue, 2014). The goal of semi-supervised learning is to take advantage of the combination of labeled and unlabeled data to improve on learning behavior. SSL is of particular interest in big data scenarios as it can make use of abundant unlabeled data to enhance supervised learning tasks when labeled data are difficult or expensive to obtain (Zhu, Hu, Yan, & Li, 2010). Basically it takes advantage of the relationship among labeled and unlabeled data to make more accurate predictions than purely using labeled data for training (Zhang et al., 2014; Zhu et al., 2010). However, many SSL approaches achieve improvements by using extra content/information that may not be available in practical applications, such as independent types of *metadata* required to build different training sets; e.g., items' genres and ratings. Methods that follow this scheme typically enrich the original matrix with extra information that is difficult to obtain or requires an expert, or using over-simplistic heuristics instead (e.g. the average of ratings or the most common rating of the user) (Karimi et al., 2011; Ristoski, Mencia, & Paulheim1, 2014; Zhang et al., 2014).

An important research field of SSL is *co-training*, a multi-view learning approach originally proposed by Blum and Mitchell (1998), where the features of the domain can be partitioned into disjoint subsets (views) that would presumably suffice to learn the target concept if enough labeled data was available (Xu, Tao, & Xu, 2013). In a broader sense, co-training algorithms can use different views of the data other than disjoint subsets of features. The only assumption is that the different views are expected to be uncorrelated and compatible, i.e., a problem has uncorrelated views if, given the label of any example, its descriptions in each view are independent and two views are compatible if all examples are labeled identically by the target concepts in each view  (Sun et al., 2013; Xu et al., 2013). "*Given this assumption, the unlabeled examples can be used to eliminate functions that are not "compatible", hence yielding information that can help to learn the target function, the "uncorrelated" assumption means that given the label of anyexample, its descriptions in different views are conditionally independent. This assumption ensures that the labels propagate throughout the domain through the unlabeled examples instead of being confined to part of the domain*" (Zhang & Lee, 2005).

In a recent, very preliminary conference publication (Da Costa, Manzato, & Campello, 2018), we showed that co-training can be used to tackle traditional recommendation problems such as sparsity and cold-start by enriching the databases as a pre-processing step for the recommender algorithms, thus reducing the amount of labeled data required for learning. The proposed co-training technique for recommendation, called CoRec, was shown to be able to simultaneously construct different recommenders as different views of the data, making use of cheap and abundant unlabeled data in addition to the original labeled data available for training. However, CoRec produces two distinct models and predictions of the same data, each of which from a new, enriched user-item matrix generated from one of its two independent views. This imposes the additional overhead of assessing and choosing between different models for the final recommendation. In addition, CoRec was designed and tested to operate with two distinct recommender algorithms (views) only.

In this paper, we propose a thorough extension of our preliminary work (Da Costa et al., 2018). Here, CoRec is not only described and tested much more comprehensively, but it is also conceptually extended. The proposed extensions are twofold. First, CoRec is generalized to operate with any number of recommender algorithms as multiple views, rather than only two. Second, and more importantly, we incorporated into the method an aggregation scheme that combines the multiple models generated during co-training into an ensemble. The resulting, extended algorithm is named ECoRec. Like CoRec, ECoRec is able to construct multiple recommenders by incorporating different views of the data, which ideally helps capture their characteristics from different and complementary viewpoints. In order to combine and improve the individual models generated by CoRec as they are simultaneously co-trained, ECoRec incorporates an in-built ensemble module based on a confidence criterion, to generate only one enriched and confident user-item matrix, which can be used by any recommender algorithm at the end of the process. In addition, our approach allows different types of explicit feedback and recommender algorithms to be integrated into the model, as opposed to other methods in the literature that are hooked to particular types of feedback and recommender algorithms. As a result, ECoRec is able to significantly reduce well-known problems such as sparsity and cold-start, allowing one to build better training models than using standalone recommenders.

Unlike our previous work (Da Costa et al., 2018), in this paper we evaluate our approach with more than two different collaborative recommenders, and we compare different types of predictions generated by ECoRec to demonstrate the generality of the proposal. We have also run experiments with a larger collection of datasets. In fact, the experiments were run with eight real databases (in contrast to only three in the preliminary, conference paper) from different domains, namely, movies, books, jokes and songs. Our study shows that the proposed ensemble co-training approach is able to provide better accuracy than individual predictions by combining results from multiple co-trained recommenders.

This article is structured as follows: in Section 2 we overview research results related to multi-view learning and ensemble approaches for recommender systems. In Section 3 we review the recommender algorithms that we use in our study and the co-training process in a broad sense. We then present our approach in Section 4 and discuss the experimental results in Section 5. In Section 6 we present our conclusions and future work.

## 2. Related work

One of the most straightforward techniques to tackle the sparsity and cold-start problems in recommender systems is filling the missing ratings in the user-item matrix with default values (e.g. averages). This method, however, lacks personalization, is based on ad-hoc heuristics, and may include noisy data into the process, besides being time and memory consuming (Bobadilla et al., 2013; Ricci et al., 2015). For this reason, a variety of other, more sophisticated techniques have been developed.

In the following we discuss semi-supervised algorithms and ensemble methods applied to recommender systems, since these are the two topics mostly related to our proposal.

### 2.1. Semi-supervised learning in recommender systems

The recommender system domain is very challenging due to its idiosyncrasies, such as, for instance, dealing with large matrices that are typically up to 99% empty. In face of these challenges, only little work has been done on semi-supervised learning for recommender systems in order to enrich the original user-item matrix.

One approach involves active learning techniques, where an algorithm chooses which label (rating) to request from a user in order to maximize the performance gain from a given labeling budget (Karimi et al., 2011). For example, in Sun's work (Sun et al., 2013), a recommender approach was proposed that conducts an interview process as guided by a decision tree with multiple questions at each split, to learn users' behavior in order to improve the power of prediction of the recommender algorithm. Active learning techniques, however, are based on the assumption that a user is able and willing to provide any requested labels, which is often not the case in real applications of recommender systems.

Another alternative is co-training. The enrichment of user-item matrix using co-training technique is proposed by some works fully be integrated ratings of collaborative filtering, user profiles, item profiles, relationships between user profiles and item profiles of content-based filtering. Zhang et al. (2014) proposed a collaborative filtering co-training method for stationary, batch-based recommender systems that makes use of items' metadata in order to enrich the traditional training model. Zhang and Wang (2015) proposed a multi-view learning framework with the ability of knowledge transfer for recommendation. Their framework was developed for item recommendation and can learn multi-view representations automatically from data. However, both techniques achieve improvements by using extra content/information. In fact, two or more different and independent types of metadata are required to build different training sets — e.g., items' genres and ratings.

In more recent works, Quang, Lien, and Phuong (2015) proposed a collaborative filtering method based on co-training that iteratively expands the training set by switching between two different feature sets. In the collaborative filtering settings, their co-training based method used users and items as two different feature sets. Their approach leads to improved prediction accuracy than isolated recommenders and reduces the sparsity problem. Matuszyk and Spiliopoulou (2017) proposed a semi-supervised framework for stream-based recommendations based on co-training, which uses abundant unlabeled information to improve the quality of recommendations and one recommender algorithm based on matrix factorization. In their approach a single learner provides reliable predictions to itself and uses them as labels.

Despite the presented collaborative filtering methods based on memory (Karimi et al., 2011; Matuszyk & Spiliopoulou, 2017; Zhang et al., 2014) or based on model (Quang et al., 2015; Zhang & Wang, 2015) to enrich the original user-item matrix using extra information, in real scenarios, a variety of such information may not be available for recommendation or they require accurate input from a domain expert. Moreover, the computational cost to extract, treat and process the extra information within the recommender algorithms is high, which can make these algorithms extremely slow.

Our proposed approach lifts the aforementioned limitations by using only a training set composed by only one type of information and a number of recommenders for multi-view learning; thus we use different recommenders as multiple viewpoints, rather than varied input data that may not be available. Moreover, ECoRec allows different recommender algorithms to be used during the co-training process, as opposed to other approaches, which are dependent on both recommendation algorithms and external metadata.

### 2.2. Ensembles in recommender systems

Ensemble is a machine learning approach that uses a combination of various models in order to improve on the results obtained by each model individually. Several recent studies, e.g. Jahrer, Töscher, and Legenstein (2010), have demonstrated the effectiveness of an ensemble of simpler techniques, showing that they can outperform more complex algorithms operating standalone.

Su, Khoshgoftaar, Zhu, and Greiner (2008) alleviated the data sparsity problem in their work by inputting the matrix with artificial ratings, prior to using the collaborative filtering algorithm. The authors used ten different machine learning models to be evaluated for the data imputing task including an ensemble classifier. Conceptually, their proposed ensemble is a sort of hybridization approach. In (Lee & Olafsson, 2009), the recommendations of several k-NN models were combined into an ensemble. The suggested model was a fusion between the User-KNN and Item-KNN algorithms. In addition, the authors suggested the lazy bagging learning approach for computing the user-user or item-item similarities. Their experiments improved the accuracy of predictions in the neighborhood models. Schclar, Tsikinovsky, Rokach, Meisels, and Antwarg (2009) proposed a modified version of the AdaBoost ensemble regressor in order to improve the RMSE measure of a hybrid approach (matrix factorization and a neighborhood techniques). The authors demonstrated that adding more recommender algorithms to the ensemble reduces the RMSE.

Lastly, Bar, Rokach, Shani, Shapira, and Schclar (2013) proposed a systematic framework for applying ensembles to recommender systems. They employ automatic methods for generating an ensemble of collaborative filtering models based on a single CF algorithm (homogeneous ensemble). The authors experimentally showed the usefulness of this framework by applying several ensemble methods to various base CF algorithms. In Ristoski et al. (2014), the authors discussed the development of a hybrid multi-strategy book recommendation system using Linked Open Data. Their approach builds on training individual base recommenders and using global popularity scores as generic recommenders. The results of the individual recommenders are combined using an ensemble method and rank aggregation techniques. The authors showed that their approach delivers very good results in different recommendation settings and also allows incorporating diversity of recommendations. However, their work requires several types of user interactions, which is not commonly available in real databases.

Our proposal is based on a heterogeneous (rather than homogeneous) ensemble, as it combines multiple predictions generated by different recommenders. It also differs from the above related work on ensembles in that we perform exchange of information between recommenders in a co-training fashion, in order to generate more accurate predictions. Our contribution, thus, can be considered a pre-processing approach to generate augmented/enriched user-item matrices based on multiple view types and a confidence measure, but it also uses an ensemble technique to combine and enhance the results of individual views in a more robust user-item matrix.

## 3. Background

In this section, we review concepts related to our approach. Specifically, prior to describing our proposal, in the following we revise the recommender algorithms, confidence measures and the co-training method that will be used later on. It is worth remarking that there are different variants in the literature of some of the concepts reviewed in this section, particularly the recommender algorithms, so we will restrict ourselves to the specific implementations used in our experiments.

### 3.1. Notation

We use special indexing letters to distinguish users and items: a user is indicated as $u$, a known item is referred to as $i$; and $r_{ui}$ is used to refer to a rating from a user $u$ to an item $i$. The final prediction and the confidence estimate of the system about the preference of user $u$ to item $i$ are represented by $\hat{r}_{ui}$ and $\mathcal{C}_{ui}$, respectively, which are a floating point values produced by the recommender algorithm. The set of pairs $(u, i)$, for which $r_{ui}$ is known, is represented by the set $D = \{(u,i)|r_{ui} \text{ is known}\}$.

### 3.2. User k-nearest neighbors (User-KNN)

The first recommendation algorithm is the well-known User-KNN (Koren, 2008; Ricci et al., 2015), which produces recommendations based on similar users. The main goal of the algorithm is to find similar users and predict a rating based on their rating assignments.

In detail, a rating $\hat{r}_{ui}$ is predicted for an unknown user-item pair, $(u, i)$, considering the ratings that other users with similar preferences to $u$ have assigned to item $i$. To find similar users, a measure of proximity $p_{uv}$ is employed between the feature vectors describing the users (i.e., the respective rows of the user-item matrix). The proximity measure can be based, e.g., on the Pearson correlation coefficient, or the cosine similarity, among others. The final similarity measure is a retracted coefficient, defined as:

$$s_{uv} = \frac{n_{uv}}{n_{uv} + \lambda_1} p_{uv} \tag{1}$$

where $n_{uv}$ is the number of items that users $u$ and $v$ have in common, and $\lambda_1$ is a regularization constant, set to 100 according to the literature (Koren, 2008). The idea is to assign higher (lower) weights to proximity values associated with pairs of users sharing larger (smaller) numbers of rated items.

In order to predict a rating $\hat{r}_{ui}$ for an unknown user-item pair, $(u, i)$, the algorithm identifies the $k$ users that are the most similar to $u$ using the similarity measure between users described above. The algorithm then selects the subset of those users that have evaluated item $i$. This subset is denoted as $S^k(i; u)$. The final rating is then predicted using Eq. (2):

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in S^k(i;u)} s_{uv}(r_{vi} - b_{vi})}{\sum_{v \in S^k(i;u)} s_{uv}} \tag{2}$$

where $b_{ui}$ and $b_{vi}$ are baseline estimates based on the ratings provided by users, proposed by Koren (2008) for neighborhood models, and defined as:

$$b_{ui} = \mu + b_u + b_i \tag{3}$$

where $\mu$ is the global average rating and $b_u$ and $b_i$ indicate the observed deviations of user $u$ and item $i$, respectively, from the average, which are based on their interactions. The main purpose of $b_{ui}$ is to capture systematic tendencies of certain users to give higher/lower ratings than others, and certain items to receive higher/lower ratings than others. For example, suppose that we need to calculate a baseline predictor for the rating of the song Yellow Submarine by user Paul, in a music database where the overall average ($\mu$) of all ratings is 3.5 stars. However, Yellow Submarine is deemed better than an average song, so it tends to be rated 1 star above the average ($b_i = +1$), whereas Paul is a critical user, who tends to rate 0.5 stars lower than the average ($b_u = -0.5$). Hence, the baseline predictor ($b_{ui}$) for Yellow Submarine's rating by Paul would be 4 stars ($3.5 - 0.5 + 1$). The estimates $b_u$ and $b_i$ can be calculated by iterating $n$ times the following equations:

$$b_i = \frac{\sum_{u:(u,i) \in D}(r_{ui} + \mu - b_u)}{\lambda_2 + |\{u : (u, i) \in D\}|} \tag{4}$$

$$b_u = \frac{\sum_{u:(u,i) \in D}(r_{ui} + \mu - b_i)}{\lambda_3 + |\{u : (u, i) \in D\}|} \tag{5}$$

where $\lambda_2$ and $\lambda_3$ are constants set as 10 and 15, respectively, according to Koren (2008).

### 3.3. Item k-nearest neighbors (Item-KNN)

The second recommender algorithm used in our experiments is Item-KNN (Aggarwal, 2016; Koren, 2008; Ricci et al., 2015), which also uses the concept of nearest neighbors. The main difference between User-KNN and Item-KNN is that the latter predicts the unknown rating $r_{ui}$ of an item $i$ by a user $u$ based on $u$'s ratings of the $k$ items most similar to $i$. In order to find similar items, a similarity measure is employed between items, as shown in Eq. (6):

$$s_{ij} = \frac{n_{ij}}{n_{ij} + \lambda_1} p_{ij} \tag{6}$$

where $p_{ij}$ is a measure of proximity (e.g. Pearson or cosine) employed between the feature vectors describing the items (i.e., the respective columns of the user-item matrix) and $n_{ij}$ is the number of shared features that describe items $i$ and $j$ (i.e., number of users that rated both items). The value of $\lambda_1$ is the same used in Eq. (1).

Using the similarity measure in Eq. (6), we identify the set of $k$ items that are most similar to $i$, the so-called $k$-nearest neighbors, and select the subset of those items that have been rated by $u$. Using this subset, denoted as $Si^k(i; u)$, the final predicted rating is an average of such similar items' ratings, adjusted to their baseline estimate:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in Si^k(i;u)} s_{ij}(r_{uj} - b_{uj})}{\sum_{j \in Si^k(i;u)} s_{ij}} \tag{7}$$

where $b_{ui}$ and $b_{uj}$ are computed by Eq. (3).

### 3.4. Singular Value Decomposition (SVD)

A matrix factorization technique, called Singular Value Decomposition for recommender systems (SVD) (Koren, Bell, & Volinsky, 2009), stands out from other recommender systems for allowing the discovery of underlying latent characteristics to the interactions between users and items. In order to apply matrix factorization, initially we have a set $U$ of users and a set $I$ of items, as well as a sparse matrix $\mathbf{R}$ ($|U| \times |I|$) that corresponds to the ratings given by the users to the items. To discover the latent features of dimensionality $F$, we have to find two matrices $\mathbf{P}$ and $\mathbf{Q}$ of dimensions $|U| \times |F|$ and $|I| \times |F|$, respectively, such that their product approximates $\mathbf{R}$:

$$\mathbf{R} \approx \mathbf{P} \times \mathbf{Q}^T = \hat{\mathbf{R}} \tag{8}$$

Each row of $\mathbf{P}$ carries the strength of the associations between a user and the resources, whereas each row of $\mathbf{Q}$ carries the strength of the associations between an item and resources (Koren et al., 2009). For predicting a rating of an item $i$ by $u$, we can calculate the dot product of two vectors corresponding to $u$ and $i$:

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i = b_{ui} + \sum_{f=1}^{F} p_{uf} q_{if}. \tag{9}$$

where the baseline $b_{ui}$ is defined as $b_{ui} = \mu + b_u + b_i$ and accounts for differences of users and items in comparison to the overall rating average $\mu$. All parameters are estimated by minimizing the associated squared error function:

$$\min_{p_*, q_*, b_*} \sum_{(u,i) \in K} (r_{ui} - \mu - b_u - b_i - p_u^T q_i)^2 + \lambda (b_u^2 + b_i^2 + ||p_u||^2 + ||q_i||^2) \quad . \tag{10}$$

After estimating $\mathbf{P}$ and $\mathbf{Q}$ by minimizing the regularized squared error, one can predict the unknown ratings by taking the dot product of the latent features for users and items. To minimize the loss function, it is possible to use stochastic gradient descent (SGD) or alternating least square (ALS), which can also be used for on-line learning, i.e., updating the model incrementally each time a new rating comes in Koren et al. (2009) and Bobadilla et al. (2013).

### 3.5. Confidence measures

Collaborative filtering approaches should provide confidence values associated with accurate values of prediction, in order to distinguish individual predictions that can be trusted from those that cannot. Research about confidence measures in RS requires the existence of simple, plausible and universally reliable quality measures, usually focused on accuracy (Bobadilla, Gutirrez, Ortega,

& Zhu, 2018). Novelty, serendipity and diversity have been studied; nevertheless, there is a noticeable lack of systematic and experimental analysis in confidence quality measures (Hernando, Bobadilla, Ortega, & Tejedor, 2013; Mazurowski, 2013).

Along with the well-known and traditional collaborative filtering algorithms (e.g. neighborhood and matrix factorization models), the literature reports some confidence measures, such as:

- **Support for user (SU)**: the confidence of an individual prediction is simply the number of ratings that the user assigned previously (i.e., if a user assigned 30 ratings in the database, the confidence estimate $\mathcal{C}_{ui}$ will be 30, for any rated item $i$) (Mazurowski, 2013).
- **Support for item (SI)**: the confidence $\mathcal{C}_{ui}$ of prediction $\hat{r}_{ui}$ is simply the number of ratings that were assigned to the item $i$ within the available database (Mazurowski, 2013).
- **Variability for item (VI)**: the confidence $\mathcal{C}_{ui}$ of an individual prediction is the standard deviation of ratings that were assigned to item $i$ for all users in the dataset (Adomavicius, Kamireddy, & Kwon, 2007).
- **Resample**: repeats the entire prediction process of the CF algorithm using different subsets of the entire set of ratings available for system development (Mazurowski, 2013). The variability is measured by the standard deviation of the predictions. Since a higher standard deviation is expected to correspond to lower confidence, the inverse of the standard deviation can be taken as the confidence prediction $\mathcal{C}_{ui}$. This measure also contains two extensions: *Fast*, which reduces the computational time and uses only the prediction step to calculate confidence; and *Noise*, that instead of re-sampling the subset, it replaces the ratings with a value randomly sampled from a Gaussian distribution.

The first three measures have low computational cost and complexity since they involve simple calculations (e.g. standard deviation and mean), however they only consider information about items or users in a separated way. On the other hand, Resample measures are more robust and accurate in estimating confidence of individual predictions, but they have high computational cost, since they require the execution of the recommender algorithm many times as well as the re-calculation of the similarity matrix.

### 3.6. Co-training

Co-training is concerned with the problem of learning from data represented by multiple distinct views (Blum & Mitchell, 1998). The emergence of this learning scheme is largely motivated by the fact that data in a variety of real applications are (or can be) described by different feature sets or different views of the same feature set (Sun et al., 2013; Xu et al., 2013). Generally, co-training is used when there are only small amounts of labeled data yet large amounts of unlabeled data described by two or more views that are ideally conditionally independent and sufficient to represent the target concept (Blum & Mitchell, 1998). Co-training first learns a separate prediction model (e.g. a classifier) for each view using any labeled examples. The most confident predictions of each model on the unlabeled data are then used to iteratively obtain additional (augmented) labeled training sets for further refinement of the other model(s).

This method assumes that each example is described using at least two different views that provide complementary information. Typically, different views can be obtained from disjoint feature subsets of different natures. For example, in multimedia content understanding, multimedia segments can be simultaneously described by their video and audio signals. In webpage classification, a webpage can be described by the document text itself and also by the anchor text attached to hyperlinks pointing to this page (Blum & Mitchell, 1998).

However, co-training can also be performed when there is only one representation of the data, if such a single representation is processed by independent prediction models, such as two different classifiers (Xu et al., 2013). In this case, the method exploits two views of the same, single data representation by co-training two classifiers using the available training examples. Iteratively, each classifier labels the unlabeled examples and the most confident predictions are provided as additional training examples to improve the accuracy of the other classifier.

In this paper we extend this idea to the ratings prediction problem in recommender systems in order to tackle sparsity and cold start issues by simultaneously considering multiple views of the same data. Our method couples the training of different, independent recommender systems in order to mutually and iteratively improve their accuracy by making each recommender provide the other(s) with its most confident predictions, in a co-training fashion, which is further enhanced by an ensemble of the resulting recommenders. The method is discussed in more detail in the next section.

## 4. Proposed approach

We propose an Ensemble Co-training Recommender system (ECoRec) that uses two or more recommender algorithms as viewpoints, such as those presented in Section 3, in order to provide more accurate predictions and to significantly reduce the sparsity problem by enriching the original user-item ratings matrix. ECoRec uses only a single training set as input data and two or more distinct recommender algorithms as different views, and combine the output of each recommender into an ensemble.

In the co-training stage, each recommender is trained aiming to agree with other recommenders' predictions, such that they learn not only from the original training set of labeled data, but also from each other's most confident predictions of the unlabeled data. In a rating prediction scenario, a “label” means a rating $r_{ui}$ of an item $i$ by a user $u$, and “labeling” unlabeled data means assigning predicted ratings $\hat{r}_{ui}$ for unseen user-item pairs $(u, i)$.

At each iteration of the co-training process, the recommender representing one view labels the unlabeled data, the most confident predicted ratings are then added to the training pool of the other recommender(s), and vice-versa. Thus, the information underlying two or more different views can be exchanged. Specifically, each recommender predicts ratings for all unobserved user-item pairs (unlabeled set), and the most confident predictions are used to populate the training set (i.e., to augment/enrich the labeled set) of the other recommender(s), in an iterative fashion. In addition, at each iteration the predictions of each recommender are combined into an ensemble, weighted by their confidence. The overall process is shown in Fig. 1, where it can be seen that, in contrast to single recommenders, our approach simultaneously learns two or more models iteratively, in order to exploit different viewpoints of the same input data and boost the learning performance.

There are four phases in our technique, namely, data pre-processing, co-training, ensemble, and recommendation. In the following subsections, we describe these procedures in more detail.

### 4.1. Data pre-processing

In this stage, ECoRec splits the original data set into a labeled set ($T_L^\eta$), which contains all triples $(u, i, r_{ui})$ of the user-item matrix, and an unlabeled set ($T_{UL}^\eta$), which contains randomly selected unobserved pairs $(u, i)$ per user. These are the sets used as input to co-training the pool of recommender algorithms ($\eta = 1, 2, \ldots, N$).

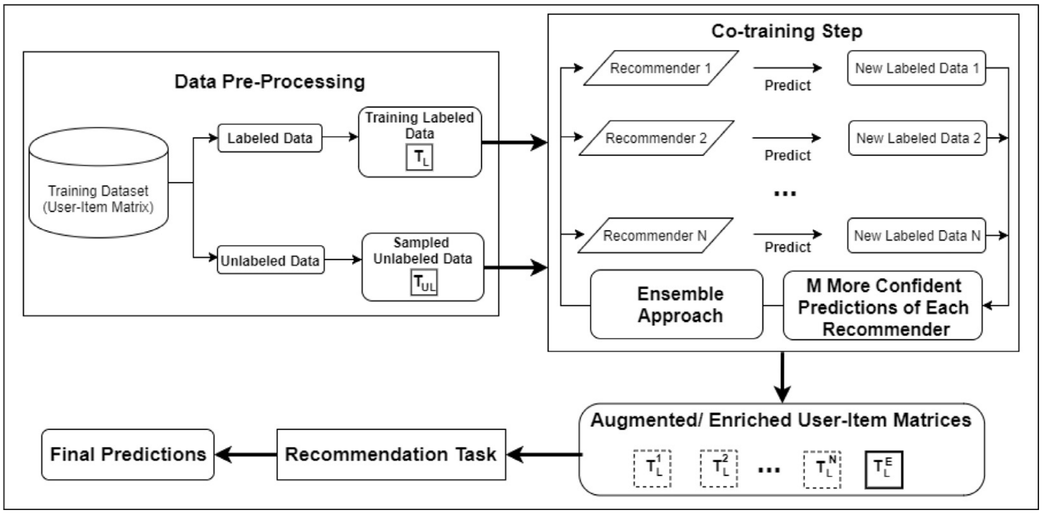

**Fig. 1.** Schematic view of the proposed system.

The initial labeled and unlabeled sets are common to all recommenders, i.e., $T_L^1 = T_L^2 = \ldots = T_L^N$ and $T_{UL}^1 = T_{UL}^2 = \ldots = T_{UL}^N$. The initial labeled set, $T_L^\eta$, is constituted by all observed user-item-ratings triples, i.e., all entries of the user-item matrix for which ratings are available (non-empty entries). Since the user-item matrix is sparse, the amount of unobserved (empty) entries can be enormous. Instead of including all the unrated user-item pairs into the initial unlabeled set $T_{UL}^\eta$ that is provided to the recommenders, which would impose an unnecessary additional computational burden and slow down convergence of the co-training process, we instead randomly select $X$ unlabeled pairs ($u$, $i$) per user. For example, if $X = 40$, for each user in the original user-item matrix, 40 items unknown to that user will be randomly chosen to be included into the unlabeled set, $T_{UL}^\eta$. We used random subsampling to incorporate novelty and aggregated diversity into the original datasets and thus soften the problems of new users and items (Castells, Hurley, & Vargas, 2015). Since all pairs that are randomly chosen are unlabeled, all the $X$ new items chosen to be labeled for each user will be novel to that user. As these items are selected randomly, we have a chance to pick up both good and bad items for users based on their behavior. The task of the recommenders during co-training is to jointly learn how to predict the ratings that users would give to the pairs in this unlabeled set.

### 4.2. Co-training step

Having the recommenders as well as the labeled and unlabeled sets in hand, the next step is to apply the co-training on these recommender algorithms. Algorithm 1 shows how the method of co-training works in ECoRec.

Our co-training process consists of evaluating the same training set in different views, driving them to agree and complement the predictions made by each other. To this end, different rating prediction algorithms can be used, provided that they represent different views of the data. For example, we can use Item-KNN and User-KNN as views, where the first favors similarity between users, whereas the second favors similarity between items.

For every iteration of the ECoRec algorithm, each recommender $\eta$ ($\eta = 1, 2, \ldots, N$) is responsible for predicting ratings for its unlabeled set, $T_{UL}^\eta$, based on its labeled set, $T_L^\eta$ (lines 11 to 14). Then, ECoRec calculates the confidence of each predicted rating, choosing the $M$ most confident ones (lines 15–16) and removing them from the set of unlabeled data for all recommenders (lines 18 to 21). Notice that the (positive integer) parameter $M$ controls the learning rate of the co-training process. As usual in statistical and machine learning algorithms, slow learning (corresponding to small values of $M$) tends to produce more accurate models, at a higher computational cost though (larger number of iterations required for convergence).

Once this procedure is completed for all the recommenders, the algorithm updates the labeled set of each recommender by adding its $M$ most confident examples to the other recommenders' labeled sets (lines 22 to 33), since we expect each model to learn only from the newly labeled examples provided by the other views, besides the original labeled examples. If an unlabeled pair ($u$, $i$) belongs to the confident sets ($T_\eta$) of two or more views, then the weighted average of the corresponding predicted ratings ($\hat{r}_{ui}^\eta$) is taken as the final rating $\hat{r}_{ui}^\eta$, and the triple $(u, i, \hat{r}_{ui}^\eta)$ is added to the designated labeled sets. The average is weighted by the confidence of the predicted ratings in question. The intuition behind our approach is that one recommender adds confident new examples to the labeled sets that the other recommender(s) will then use for learning, but from another perspective (view). In our approach the predictions made by each recommender never feed their own labeled set, as usual in co-training. For example, using two recommenders in ECoRec, the predictions made by recommender 1 will only feed the labeled set $T_2^L$, but never its own ($T_1^L$) (line 25). The algorithm only stops when all unlabeled data of all views have been labeled, which is guaranteed by construction.

The outputs of the co-training process are new augmented/enriched matrices for each recommender used ($T_L^1$, $T_L^2$, ... , $T_L^N$), besides a $T_L^E$ set that is generated by an ensemble using the confident sets of all individual recommenders ($T_F$) (lines 34 to 36), as detailed in Section 4.3.

From the recommender systems perspective, the recommender algorithms tend to improve their prediction power as the data increase (Aggarwal, 2016; Hernando et al., 2013; Mazurowski, 2013). In fact, how well the recommenders work depends directly on the number of labeled examples and their associated confidence measure. Both factors contribute to have an aggregated impact on the final results and convergence during the co-training approach. The idea behind the confidence measure is that examples labeled with high confidence are expected to decrease the most the error of the recommender on the labeled example set, if they are included. Our proposed confidence criterion is discussed next.

**Confidence criterion**: One challenge in our method, which is inherent to co-training approaches, is how to select a subset of predicted examples from each recommender's unlabeled set (line

```
    Input: User-item matrix, positive integers $X$ and $M$
1   Task 1: Data Pre-Processing$T_L \leftarrow$ all labeled triples $(u, i, r_{ui})$ of the user-item matrix$T_{UL} \leftarrow X$ randomly selected unlabeled pairs $(u, i)$ per userfor $\eta$=1 to N do
2       $T_L^\eta$, $T_{UL}^\eta \leftarrow T_L$, $T_{UL}$
3   end
4   $T_L^E \leftarrow T_L$Task 2 : Co-training and Ensemble Techniquewhile $T_{UL}^1 \neq \emptyset$, $T_{UL}^2 \neq \emptyset$, ... , $T_{UL}^N \neq \emptyset$ do
5       for $\eta$=1 to N do
6           foreach user-item pair $(u, i) \in T_{UL}^\eta$ do
7               $\hat{r}_{ui}^\eta \leftarrow$ prediction made by recommender $\eta C_{ui}^\eta \leftarrow$ confidence of prediction $\hat{r}_{ui}^\eta$ by Eq.~(11)
8           end
9           $R_\eta \leftarrow M$ most confident triples $(u, i, \hat{r}_{ui}^\eta)$ according to $C_{ui}^\eta T_\eta \leftarrow M$ pairs $(u, i)$ corresponding to the triples in $R_\eta$
10      end
11      $T_F \leftarrow T_1 \cup T_2 \cup \ldots \cup T_N$for $\eta$=1 to N do
12          $T_{UL}^\eta \leftarrow T_{UL}^\eta - T_F$
13      end
14      for $\eta$=1 to N do
15          $L_\eta \leftarrow \emptyset$for $\omega$=1 to N do
16              if $\eta$ != $\omega$ then
17                  foreach user-item-rating triple $(u, i, \hat{r}_{ui}^\omega) \in R_\omega$ do
18                      $L_\eta \leftarrow L_\eta \cup (u, i, \hat{r}_{ui}^\omega)$
19                  end
20              end
21          end
22          Combine triples $(u, i, \hat{r}_{ui}^{(\cdot)}) \in L_\eta$ sharing a common $(u, i)$ into a single entry, by Eq.~(13)$T_L^\eta \leftarrow T_L^\eta \cup L_\eta$
23      end
24      $R_F \leftarrow R_1 \cup R_2 \cup \ldots \cup R_N$Combine triples $(u, i, \hat{r}_{ui}^{(\cdot)}) \in R_F$ sharing a common $(u, i)$ into a single entry, by Eq.~(13)$T_L^E \leftarrow T_L^E \cup R_F$
25  end
26  Task 3: Recommendation Taskfor $\eta$=1 to N do
27      Train the Recommender $\eta$ with $T_L^E$ to predict new ratings for unknown items for each user
28  end
    Output: Sets of unknown pairs $(u, i)$ with their respective predicted ratings $(\hat{r}_{ui})$.
```

**Algorithm 1:** ECoRec Algorithm.

15) so as to enhance the labeled set of the other recommenders. The selection criterion we used in this work is the recommender algorithm's *confidence*. Every candidate prediction must have a confidence value associated with it.

Confidence in recommender systems can be defined as the system's trustworthiness of its recommendations or predictions (Ricci et al., 2015). When a confidence measure is available, the user can benefit from observing the confidence scores along with the recommendations (Guo, 2013; Zhou & Li, 2005). For instance, when the system reports a low confidence in a recommended item, the user may tend to further inspect the item before making a decision. However, not every recommender algorithm generates a confidence score, or there may be a high cost associated to computing one.

The confidence criterion we propose and use in this work is based on the fact that collaborative filtering recommenders tend to improve their accuracy as the amount of data over items/users grows. Based on our previous work (Da Costa et al., 2018), our confidence measure for a prediction $\hat{r}_{ui}^\eta$ made by recommender algorithm $\eta$ for the rating of pair $(u, i)$ is as follows:

$$C_{ui}^\eta = \sigma_{ui}^\eta N_u N_i \tag{11}$$

where $N_u$ and $N_i$ are the popularity of user $u$ and item $i$, respectively, given by the number of ratings applied/received by user $u$ and item $i$ in the labeled set associated with recommender $\eta$, $T_L^\eta$, and $\sigma_{ui}^\eta$ represents a measure of trustworthiness of the prediction made by recommender $\eta$, defined as:

$$\sigma_{ui}^\eta = \frac{1}{|b_{ui} - \hat{r}_{ui}^\eta|} \tag{12}$$

where $b_{ui}$ is a baseline estimate of the rating for the pair $(u, i)$ in question, calculated based on the global mean and the mean of interactions of users and items in $T_L^\eta$, using Eq. (3). The purpose of using $\sigma_{ui}^\eta$ is to weight the users' and items' popularities by the inverse of the error between the baseline estimate and the rating predicted by the recommender, since a small error means that the predicted rating is close to the behavior of that user and item in the dataset. For example, a user has a $b_{ui} = 4.5$ and the recommender $\eta$ has generated $\hat{r}_{ui}^\eta = 4$, so the difference between them will be 0.5. Since we want to weight popularities so that the smaller the difference the greater the weighting, we take the inverse of the difference, i.e., $\sigma_{ui}^\eta = 2$.

The main rationale behind our proposed confidence measure is that the more popular users and items are (i.e. the more ratings are associated with them), the more accurate the predictions calculated based on these users and items tend to be. In addition, the confidence on a prediction deviating largely from the baseline rating $b_{ui}$, which takes into account the overall ratings given/received by the user $u$ and the item $i$ in question, tends to be penalized. Unlike several approaches presented in Section 3.5, our confidence measure considers both user and item interactions, as well as the quality of the prediction generated by each recommender in each epoch of the co-training approach.

### 4.3. Ensemble

As a result of each iteration of the co-training process, we get multiple recommenders, which may provide different recommendations from their different views. In order to get a single, more robust reconciled prediction, we can ensemble the individual predictions of each recommender during the co-training process. A straightforward alternative is to take the average of the ratings over all recommenders, which results in a *bagging-like* strategy that is known to be able to simultaneously reduce bias and variance of prediction models. However, this method does not consider the confidence associated with the predictions from different recommenders. To take confidence into account, we ensemble the results by a weighted average of the predictions generated by individual recommenders at each iteration of the co-training process. The confidence measure in Eq. (11) is used for weighting:

$$\hat{r}_{ui} = \frac{\sum_{\eta \in S} C_{ui}^\eta \hat{r}_{ui}^\eta}{\sum_{\eta \in S} C_{ui}^\eta} \tag{13}$$

In Eq. (13), $\hat{r}_{u,i}^\eta$ and $C_{ui}^\eta$ stand for the predicted rating and its confidence, respectively, provided by recommender $\eta$ for user-item $(u, i)$. Set $S \subseteq \{1, \cdots, N\}$ is the subset of recommenders for which the predicted rating $\hat{r}_{u,i}^\eta$ should be averaged. In the case of the ensemble performed at the end of each iteration of the ECoRec algorithm (line 35 of Algorithm 1), $S$ is the subset of recommenders $\eta$ for which $\hat{r}_{u,i}^\eta$ is one of the $M$ most confident predictions of that recommender at that iteration, i.e., $(u, i, \hat{r}_{u,i}^\eta) \in R_\eta$. Formally, $S = \{\eta : (u, i, \hat{r}_{u,i}^\eta) \in R_\eta\}$. This way, multiple confident predictions for the same entry $(u, i)$ can be combined into a single prediction, $\hat{r}_{ui}$.

**Table 1**
Comparison between our proposed confidence measure against three traditional measures in terms of RMSE and F-measure. Bold typeset indicates the best performance.

| Datasets | Measures/Confidence | SI | SU | VI | PC |
|---|---|---|---|---|---|
| **Yahoo Movies** | RMSE | 0.9766 | 0.9753 | 0.9747 | **0.9489** |
| | F-Measure | 0.8311 | 0.8151 | 0.8322 | **0.8461** |
| **FilmTrust** | RMSE | 0.8673 | 0.8657 | 0.8645 | **0.8011** |
| | F-Measure | 0.6111 | 0.6251 | 0.6372 | **0.6578** |
| **Ciao DVD** | RMSE | 1.1628 | 1.1697 | 1.1644 | **1.0288** |
| | F-Measure | 0.7799 | 0.7987 | 0.8096 | **0.8210** |
| **MovieLens 2k** | RMSE | 0.7911 | 0.79085 | 0.79097 | **0.7401** |
| | F-Measure | 0.7757 | 0.7736 | 0.7836 | **0.8135** |
| **Jester** | RMSE | 0.9955 | 0.9987 | 0.9986 | **0.9568** |
| | F-Measure | 0.5566 | 0.5551 | 0.5624 | **0.5882** |
| **Booking Crossing** | RMSE | 0.8526 | 0.8518 | 0.8526 | **0.7693** |
| | F-Measure | 0.8912 | 0.8843 | 0.8942 | **0.9165** |
| **Amazon Digital Music** | RMSE | 0.9376 | 0.9318 | 0.9345 | **0.8798** |
| | F-Measure | 0.8597 | 0.8606 | 0.8602 | **0.8901** |
| **Yahoo Music** | RMSE | 1.1574 | 1.1564 | 1.1559 | **1.1229** |
| | F-Measure | 0.4308 | 0.4311 | 0.4397 | **0.5261** |

**Table 2**
Comparison between ECoRec with two recommenders and standalone KNN-based recommenders in terms of RMSE.

| Recommenders | User-KNN | | | | Item-KNN | | | |
|---|---|---|---|---|---|---|---|---|
| User-Item Matrix | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^E$ | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^E$ |
| | | | | **Yahoo Movies** | | | | |
| 20 new ratings | 0.9834 | 0.9786 | 0.9821 | 0.9778 | 0.9637 | 0.9621 | 0.9581 | 0.9530 |
| 30 new ratings | | 0.9771 | 0.9789 | 0.9767 | | 0.9520 | 0.9495 | 0.9498 |
| 40 new ratings | | 0.9737 | 0.9780 | **0.9716** | | 0.9508 | 0.9499 | **0.9489*** |
| | | | | **FilmTrust** | | | | |
| 20 new ratings | 0.8584 | 0.8369 | 0.8465 | 0.8410 | 0.8336 | 0.8297 | 0.8325 | 0.8274 |
| 30 new ratings | | 0.8278 | 0.8417 | 0.8207 | | 0.8204 | 0.8198 | 0.8118 |
| 40 new ratings | | 0.8287 | 0.8384 | **0.8198** | | 0.8160 | 0.8172 | **0.8011*** |
| | | | | **CiaoDVD** | | | | |
| 20 new ratings | 1.0761 | 1.0815 | 1.0810 | 1.0769 | 1.0664 | 1.0634 | 1.0616 | 1.0585 |
| 30 new ratings | | **1.0711** | 1.0750 | 1.0716 | | 1.0569 | 1.0611 | 1.0502 |
| 40 new ratings | | 1.0897 | 1.0886 | 1.0876 | | 1.0472 | 1.0409 | **1.0288*** |
| | | | | **MovieLens 2k** | | | | |
| 20 new ratings | 0.7908 | 0.7897 | 0.7901 | 0.7834 | 0.7460 | 0.7458 | 0.7456 | 0.7437 |
| 30 new ratings | | 0.7894 | 0.7906 | 0.7811 | | 0.7452 | 0.7448 | 0.7413 |
| 40 new ratings | | 0.7814 | 0.7893 | **0.7803** | | 0.7414 | 0.7429 | **0.7401*** |
| | | | | **Jester** | | | | |
| 20 new ratings | 1.0208 | 1.0045 | 1.0098 | 1.0011 | 0.9679 | 0.9653 | 0.9644 | 0.9639 |
| 30 new ratings | | 0.9955 | 0.9985 | 0.9904 | | 0.9627 | 0.9625 | 0.9615 |
| 40 new ratings | | 0.9761 | 0.9854 | **0.9702** | | 0.9592 | 0.9579 | **0.9568*** |
| | | | | **Booking Crossing** | | | | |
| 20 new ratings | 0.9935 | 1.0677 | 1.0824 | 1.0524 | 0.8659 | 0.9146 | 0.9151 | 0.9054 |
| 30 new ratings | | 0.9928 | 0.9916 | 0.9910 | | 0.8637 | 0.8781 | 0.8386 |
| 40 new ratings | | 0.8078 | 0.8308 | **0.8074** | | 0.7738 | 0.7820 | **0.7693*** |
| | | | | **Amazon Digital Music** | | | | |
| 20 new ratings | 0.9715 | 1.0415 | 1.0505 | 1.0148 | 0.9057 | 0.9952 | 1.0074 | 0.9889 |
| 30 new ratings | | 0.9700 | 0.9684 | 0.9695 | | 0.9001 | 0.9121 | 0.8989 |
| 40 new ratings | | 0.8973 | 0.9088 | **0.8918** | | 0.8890 | 0.8905 | **0.8798*** |
| | | | | **Yahoo Music** | | | | |
| 20 new ratings | 1.1476 | 1.1466 | 1.1463 | 1.1460 | 1.1451 | 1.1297 | 1.1280 | 1.1276 |
| 30 new ratings | | 1.1458 | 1.1453 | **1.1441** | | 1.1318 | 1.1278 | **1.1229*** |
| 40 new ratings | | 1.1539 | 1.1628 | 1.1572 | | 1.1356 | 1.1327 | 1.1328 |

Bold typeset indicates the best performance in each recommender algorithm. * indicates statistical significance with *p*-value <0.01.

From Eq. (13) we obtain, at each iteration of the co-training process, a single rating prediction that takes into consideration the confidence of the individual constituent predictions. For example, let us consider 3 recommenders in ECoRec ($N = 3$). Let's suppose that, at a given iteration of the algorithm, Recommenders 1 and 3 both have entry (user *k*, item *j*) within their most confident predictions ($r^1_{kj} = 5$ with $\mathcal{C}^1_{kj} = 0.5$ and $r^3_{kj} = 3.5$ with $\mathcal{C}^3_{kj} = 0.8$). Assuming that entry (user *k*, item *j*) is not within the most confident set of Recommender 2, the computation of the final prediction for this entry according to the ensemble is $\hat{r}_{kj} = (0.5 \cdot 5 + 0.8 \cdot 3.5)/(0.5 + 0.8) = 4.07$.

Notice that we perform the ensemble within the co-training loop (lines 34 to 36 of Algorithm 1), because we need the confidence values generated at each iteration of the co-training process to capture the confidence information as it evolves throughout iterations.

### *4.4. Recommendation task*

Algorithm 1outputs a number of augmented/enriched user-item matrices, $T_L^1$, $T_L^2$, ... , $T_L^N$, besides $T_L^E$ generated by the ensemble. These matrices can now be used by different recommenders to calculate predictions for new ratings. In our experiments, we applied only the user-item matrix generated by our ensemble approach ($T_L^E$) in all individual recommender algorithms to predict new ratings, such that we obtain recommendations from multiple recommender algorithms but using only a single augmented/enriched user-item matrix. Optionally, the resulting ratings from the different recommenders can also be combined into a single rating using Eq. (13), as long as the confidence values associated with each pair ($u$, $i$) are stored during the co-training process.

## 5. Evaluation

To evaluate our proposal, we compare ECoRec against each individual collaborative recommender presented in Section 3. In our experiments, we applied our algorithm in two different settings: using two and three recommenders as different views. In the first case, we used Item-KNN and User-KNN as recommenders (with Pearson Correlation similarity). These algorithms were chosen because they make recommendations based on different views of the data: User-KNN uses the relations between users, while Item-KNN uses the similarity between the items to make the prediction. In the second case, we used the SVD recommender (Koren, 2008) in addition to the two previous neighborhood models. The latter generates predictions using a matrix factorization approach, which differs largely from the neighborhood models in that it uses latent factors to increase its predictive power, thus representing an alternative, complementary view of the data. These algorithms are very effective and widely used both in the literature as well as in practical applications of recommender systems (Bobadilla et al., 2013; Ricci et al., 2015).

We also compare ECoRec against the semi-supervised learning method CSEL (Zhang et al., 2014), which is a co-training framework for processing different types of information, including item's metadata. We adapted their algorithm to accept ratings only, so a fair comparison between the results could be achieved. Similar to ECoRec, this algorithm was proposed to incorporate unlabeled examples into the original user-item matrix using a co-training process. Its output also generates $N$ enriched matrices, where $N$ is the number of recommenders used in the process. We report the results corresponding to the best enriched matrix generated by CSEL. In addition, we compare our approach with one additional co-training approach, called Co-CF (Quang et al., 2015), and also with AdaBoost.RT (AB.RT), a traditional ensemble approach presented in the work of Bar et al. (2013).

### 5.1. Databases

We evaluate our approach on eight publicly available databases from different domains, which contain explicit user's interactions.

- **Yahoo Movies:** this database contains a small sample of the Yahoo! Movies community's preferences for various movies, rated on a scale from A+ to F. Developed by the Yahoo! Research Alliance Webscope program,[1] this database consists of 169,767 ratings given by 4385 users to 4339 movies. For our experiments we normalized the original ratings in the range from 1 to 5.
- **FilmTrust:** consists of 35,497 ratings given by 1508 users to 2071 movies developed by Guo, Zhang, and Yorke-Smith (2013). The ratings used as explicit feedback in our study are in the range between 0.5 and 5.
- **CiaoDVD:** is a DVD movie database proposed by Guo, Zhang, Thalmann, and Yorke-Smith (2014), and consists of 280,391 ratings given by 7375 users to 99,746 items in a range from 1 to 5.
- **MovieLens** 2$k$**:** consists of 800,000 ratings (from 0.5 to 5) and 10,000 interaction tags applied to 2113 users and 10,197 movies. As explicit information, we used the ratings users assigned to items. This database was developed by Cantador, Brusilovsky, and Kuflik (2011) and has not been modified for the experiments.
- **Jester:** consists of 1,810,455 ratings applied by 23,500 users to 100 jokes. As explicit information, we used the normalized ratings users assigned to items. For our experiments we used only a subsample of the original database proposed by Goldberg, Roeder, Gupta, and Perkins (2001), resulting in 360,917 interactions made by 5000 users to 100 items. Original ratings ($-10$ to 10) were rescaled into the range from 0.5 to 5.
- **Booking Crossing:** this database was developed in the work of Ziegler, McNee, Konstan, and Lausen (2005) and contains 278,858 users (anonymized, but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books. For our experiments, we used only users' ratings, filtering out all items with less than 5 interactions in the database, resulting in 102,963 interactions made by 6628 users to 7164 books. We also rescaled the original ratings (0 to 10) into a new range (0.5 to 5).
- **Amazon Digital Music:** consists of 206,282 ratings applied by 16,396 users to 101,708 music albums. For our experiments we used only a sample of the original data set, proposed by McAuley, Pandey, and Leskovec (2015), containing 59,793 interactions made by 6337 users to 4744 items.
- **Yahoo Music:** this database was also developed by Yahoo! Research Alliance Webscope program[1] and contains ratings for songs collected from two different sources. In this work we use a source that consists of ratings supplied by users during normal interaction with Yahoo! Music services. The rating data includes 196,150 interactions made by 15,400 users to 1000 songs.

These databases have been made available in Github[2] with their respective terms of use and licenses.

### 5.2. Experimental setup and evaluation metric

Recall from Section 4 that ECoRec generates augmented labeled sets, $T_L^1$, $T_L^2$, ... , $T_L^N$, $T_L^E$, which are then used to populate the original user-item matrix ($T_L$) with predictions made during the co-training process. The enriched user-item matrices resulting from $T_L^1$, $T_L^2$, ... , $T_L^N$, $T_L^E$ are used in order to evaluate the accuracy of the results. In particular, for the two different settings considered in our experiments (User-KNN/Item-KNN and User-KNN/Item-KNN/SVD), the algorithm then produces three and four new enriched data sets ($T_L^1/T_L^2/T_L^E$ and $T_L^1/T_L^2/T_L^3/T_L^E$), respectively. In order to evaluate the results, we apply all base recommenders to all resulting enriched sets. The same recommender algorithms used to generate enriched matrices in ECoRec (User-KNN, Item-KNN, SVD) are also used in our closest competitor, CSEL.

To measure the predictive ability of the different methods, we used root mean square error measure (RMSE) and $F$-measure (Aggarwal, 2016; Bobadilla et al., 2013; Ravi & Vairavasundaram, 2016), with 10-fold cross-validation. We computed the mean across the 10-folds and compared our method against the competitors using a two-sided paired $t$-test with a 99% confidence level (Aggarwal, 2016; Box, 1987). We compute the root mean

[1] https://webscope.sandbox.yahoo.com/

[2] https://github.com/ArthurFortes/Datasets-for-Recommneder-Systems

**Table 3**
Comparison between ECoRec with two recommenders and standalone KNN-based recommenders in terms of F-Measure.

| Recommenders | User-KNN | | | | Item-KNN | | | |
|---|---|---|---|---|---|---|---|---|
| User-Item Matrix | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^E$ | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^E$ |
| | | | | **Yahoo Movies** | | | | |
| 20 new ratings | 0.8356 | 0.8402 | 0.8410 | 0.8415 | 0.8284 | 0.8325 | 0.8323 | 0.8365 |
| 30 new ratings | | 0.8412 | 0.8420 | 0.8427 | | 0.8325 | 0.8323 | 0.8367 |
| 40 new ratings | | 0.8417 | 0.8424 | **0.8461**$^*$ | | 0.8334 | 0.8334 | **0.8374**$^*$ |
| | | | | **FilmTrust** | | | | |
| 20 new ratings | 0.6084 | 0.6307 | 0.6276 | 0.6429 | 0.6275 | 0.6312 | 0.6307 | 0.6432 |
| 30 new ratings | | 0.6364 | 0.6303 | 0.6487 | | 0.6320 | 0.6297 | 0.6482 |
| 40 new ratings | | 0.6428 | 0.6323 | **0.6578**$^*$ | | 0.6345 | 0.6327 | **0.6570**$^*$ |
| | | | | **CiaoDVD** | | | | |
| 20 new ratings | 0.7972 | 0.8042 | 0.8056 | 0.8092 | 0.7961 | 0.8022 | 0.8013 | 0.8058 |
| 30 new ratings | | 0.8067 | 0.8092 | 0.8102 | | 0.8029 | 0.8025 | 0.8070 |
| 40 new ratings | | 0.8110 | 0.8129 | **0.8210**$^*$ | | 0.8033 | 0.8035 | **0.8110**$^*$ |
| | | | | **MovieLens 2k** | | | | |
| 20 new ratings | 0.8040 | 0.8042 | 0.8048 | 0.8052 | 0.7988 | 0.8095 | 0.8102 | 0.8112 |
| 30 new ratings | | 0.8039 | 0.8049 | 0.8047 | | 0.8102 | 0.8109 | 0.8115 |
| 40 new ratings | | 0.8045 | 0.8053 | **0.8103**$^*$ | | 0.8108 | 0.8115 | **0.8135**$^*$ |
| | | | | **Jester** | | | | |
| 20 new ratings | 0.5269 | 0.5853 | 0.5861 | **0.5882**$^*$ | 0.5666 | 0.5638 | 0.5673 | 0.5692 |
| 30 new ratings | | 0.5853 | 0.5845 | 0.5860 | | 0.5670 | 0.5672 | **0.5701** |
| 40 new ratings | | 0.5831 | 0.5868 | 0.5877 | | 0.5659 | 0.5662 | 0.5687 |
| | | | | **Booking Crossing** | | | | |
| 20 new ratings | 0.8770 | 0.8946 | 0.8952 | 0.8995 | 0.8725 | 0.8754 | 0.8754 | 0.8802 |
| 30 new ratings | | 0.8967 | 0.8973 | 0.9017 | | 0.8781 | 0.8774 | 0.8844 |
| 40 new ratings | | 0.8973 | 0.8977 | **0.9165**$^*$ | | 0.8798 | 0.8796 | **0.8954**$^*$ |
| | | | | **Amazon Digital Music** | | | | |
| 20 new ratings | 0.8434 | 0.8683 | 0.8684 | 0.8741 | 0.8177 | 0.8440 | 0.8438 | 0.8572 |
| 30 new ratings | | 0.8693 | 0.8702 | 0.8809 | | 0.8458 | 0.8453 | 0.8589 |
| 40 new ratings | | 0.8705 | 0.8710 | **0.8901**$^*$ | | 0.8464 | 0.8461 | **0.8610**$^*$ |
| | | | | **Yahoo Music** | | | | |
| 20 new ratings | 0.4571 | 0.4756 | 0.4780 | **0.4866**$^*$ | 0.5028 | 0.5187 | 0.5200 | **0.5261**$^*$ |
| 30 new ratings | | 0.4742 | 0.4773 | 0.4817 | | 0.5153 | 0.5181 | 0.5209 |
| 40 new ratings | | 0.4718 | 0.4756 | 0.4805 | | 0.5101 | 0.5129 | 0.5193 |

Bold typeset indicates the best performance in each recommender algorithm. $^*$ indicates statistical significance with $p$-value $<0.01$.

**Table 4**
Comparison between KNN-based ECoRec and baselines in terms of RMSE and F-Measure. Bold typeset indicates the best performance. $^*$ indicates statistical significance with $p$-value $<0.01$.

| Database | Measure / Approach | ECoRec | CSEL | Co-CF | AB.RT |
|---|---|---|---|---|---|
| **Yahoo Movies** | *RMSE* | **0.9489** | 0.9501 | 0.9394 | 0.9477 |
| | *F-Measure* | **0.8461**$^*$ | 0.8114 | 0.8323 | 0.8371 |
| **FilmTrust** | *RMSE* | **0.8011**$^*$ | 0.8432 | 0.8385 | 0.8184 |
| | *F-Measure* | **0.6578**$^*$ | 0.5987 | 0.6356 | 0.6296 |
| **CiaoDVD** | *RMSE* | **1.0288**$^*$ | 1.0487 | 1.0452 | 1.0357 |
| | *F-Measure* | **0.8210**$^*$ | 0.8012 | 0.7798 | 0.8049 |
| **MovieLens 2*k*** | *RMSE* | **0.7401**$^*$ | 0.7498 | 0.7492 | 0.7531 |
| | *F-Measure* | **0.8103**$^*$ | 0.7921 | 0.7852 | 0.7772 |
| **Jester** | *RMSE* | 0.9568 | 0.9598 | **0.9520** | 0.9622 |
| | *F-Measure* | **0.5882**$^*$ | 0.5791 | 0.5722 | 0.5741 |
| **Booking Crossing** | *RMSE* | **0.7693**$^*$ | 0.7864 | 0.7739 | 0.7761 |
| | *F-Measure* | **0.9165**$^*$ | 0.8972 | 0.8931 | 0.8987 |
| **Amazon Digital Music** | *RMSE* | **0.8798** | 0.8906 | 0.8845 | 0.8807 |
| | *F-Measure* | **0.8901**$^*$ | 0.8721 | 0.8778 | 0.8732 |
| **Yahoo Music** | *RMSE* | **1.1229** | 1.1443 | 1.1289 | 1.1333 |
| | *F-Measure* | **0.5261**$^*$ | 0.5073 | 0.5177 | 0.5170 |

square error (RMSE) of every predicted rating $\hat{r}_{ui}$ in relation to its real rating $r_{ui}$ found in the test set:

$$\text{RMSE} = \frac{1}{|U|}\sum_{u\in U}\sqrt{\frac{1}{|O_u|}\sum_{i\in O_u}(\hat{r}_{ui} - r_{ui})^2}, \qquad (14)$$

where $O_u$ is the set of items evaluated by user $u$.

Due to the high sparsity of the datasets used, some recommender algorithms may not be capable of predicting the ratings (Ravi & Vairavasundaram, 2016), so the coverage measure is also adopted. If the recommender algorithm is not capable of predicting a rating for a $(u,\ i)$ pair, then it is said that the algorithm did not cover the particular pair of user and item:

$$coverage = \frac{CS}{N_{ratings}} \quad , \qquad (15)$$

where $CS$ denotes the number of ratings predicted and $N_{ratings}$ denotes the number of ratings tested. RMSE and coverage can be combined into a single measure as $F$-measure (Ravi & Vairavasundaram, 2016). To calculate $F$-measure, it is necessary to determine precision values. Precision can be obtained from RMSE as:

$$precision = 1 - \frac{RMSE}{Maximum\ possible\ error} \quad , \qquad (16)$$

**Table 5**
Comparison between ECoRec with three recommenders and standalone KNN-based and Matrix Factorization-based recommenders in terms of RMSE.

| Recommenders | User-KNN | | | | | Item-KNN | | | | | SVD | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| User-Item Matrix | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^3$ | $T_L^E$ | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^3$ | $T_L^E$ | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^3$ | $T_L^E$ |
| **Yahoo Movies** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.972 | 0.973 | 0.975 | 0.972 | | 0.957 | 0.955 | 0.953 | 0.952 | | 0.974 | 0.974 | 0.975 | 0.972 |
| 30 new ratings | 0.983 | 0.973 | 0.974 | 0.972 | 0.971 | 0.963 | 0.951 | 0.951 | 0.950 | 0.949 | 0.984 | 0.976 | 0.976 | 0.976 | 0.973 |
| 40 new ratings | | 0.971 | 0.972 | 0.973 | **0.970** | | 0.949 | 0.951 | 0.950 | **0.946*** | | 0.975 | 0.973 | 0.970 | **0.968** |
| **FilmTrust** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.838 | 0.836 | 0.838 | 0.835 | | 0.827 | 0.821 | 0.821 | 0.820 | | 0.819 | 0.817 | 0.818 | 0.817 |
| 30 new ratings | 0.858 | 0.843 | 0.828 | 0.838 | 0.828 | 0.833 | 0.818 | 0.817 | 0.825 | 0.817 | 0.819 | 0.819 | 0.818 | 0.818 | 0.816 |
| 40 new ratings | | 0.831 | 0.825 | 0.834 | **0.824** | | 0.814 | 0.818 | 0.816 | **0.813** | | 0.816 | 0.811 | 0.814 | **0.807*** |
| **CiaoDVD** | | | | | | | | | | | | | | | |
| 20 new ratings | | 1.071 | 1.081 | 1.072 | **1.070** | | 1.085 | 1.064 | 1.085 | 1.064 | | 0.973 | 0.971 | 0.975 | 0.970 |
| 30 new ratings | 1.076 | 1.108 | 1.113 | 1.108 | 1.106 | 1.066 | 1.070 | 1.063 | 1.076 | 1.061 | 0.981 | 0.967 | 0.970 | 0.970 | 0.969 |
| 40 new ratings | | 1.1072 | 1.1046 | 1.1332 | 1.0902 | | 1.040 | 1.052 | 1.048 | **1.036** | | 0.965 | 0.968 | 0.970 | **0.962*** |
| **MovieLens 2*k*** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.790 | 0.789 | 0.789 | **0.787** | | 0.746 | 0.746 | 0.746 | 0.745 | | 0.781 | 0.781 | 0.781 | **0.778** |
| 30 new ratings | 0.790 | 0.789 | 0.789 | 0.789 | 0.789 | 0.746 | 0.746 | 0.746 | 0.745 | **0.744*** | 0.787 | 0.785 | 0.786 | 0.785 | 0.785 |
| 40 new ratings | | 0.790 | 0.789 | 0.789 | 0.789 | | 0.746 | 0.747 | 0.747 | 0.747 | | 0.785 | 0.786 | 0.785 | 0.784 |
| **Jester** | | | | | | | | | | | | | | | |
| 20 new ratings | | 1.002 | 0.996 | 0.994 | 0.986 | | 0.962 | 0.961 | 0.963 | 0.962 | | 0.952 | 0.951 | 0.952 | 0.951 |
| 30 new ratings | 1.020 | 0.996 | 0.990 | 0.998 | **0.979** | 0.967 | 0.962 | 0.960 | 0.961 | 0.959 | 0.957 | 0.952 | 0.951 | 0.956 | **0.949*** |
| 40 new ratings | | 1.000 | 0.986 | 0.995 | 0.980 | | 0.961 | 0.957 | 0.960 | **0.955** | | 0.955 | 0.954 | 0.959 | 0.954 |
| **Booking Crossing** | | | | | | | | | | | | | | | |
| 20 new ratings | | 1.002 | 0.994 | 1.023 | 0.978 | | 0.842 | 0.861 | 0.872 | 0.839 | | 0.767 | 0.763 | 0.766 | **0.761** |
| 30 new ratings | 0.993 | 0.941 | 0.991 | 0.994 | 0.890 | 0.865 | 0.812 | 0.855 | 0.853 | 0.789 | 0.769 | 0.765 | 0.764 | 0.765 | 0.763 |
| 40 new ratings | | 0.853 | 0.872 | 0.888 | **0.814** | | 0.779 | 0.771 | 0.792 | **0.757*** | | 0.766 | 0.764 | 0.766 | 0.764 |
| **Amazon Digital Music** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.987 | 0.988 | 1.00 | 0.963 | | 0.943 | 0.947 | 0.957 | 0.942 | | 0.874 | 0.873 | 0.877 | 0.870 |
| 30 new ratings | 0.971 | 0.951 | 0.976 | 0.989 | 0.915 | 0.905 | 0.931 | 0.973 | 0.967 | 0.911 | 0.871 | 0.870 | 0.871 | 0.872 | **0.867** |
| 40 new ratings | | 0.902 | 0.929 | 0.934 | **0.883** | | 0.889 | 0.902 | 0.933 | **0.866*** | | 0.870 | 0.872 | 0.871 | 0.870 |
| **Yahoo Music** | | | | | | | | | | | | | | | |
| 20 new ratings | | 1.141 | 1.149 | 1.142 | **1.140** | | 1.124 | 1.125 | 1.126 | **1.118*** | | 1.171 | 1.170 | 1.189 | **1.169** |
| 30 new ratings | 1.147 | 1.147 | 1.143 | 1.144 | 1.142 | 1.145 | 1.128 | 1.131 | 1.129 | 1.127 | 1.197 | 1.189 | 1.192 | 1.192 | 1.188 |
| 40 new ratings | | 1.149 | 1.148 | 1.150 | 1.147 | | 1.134 | 1.134 | 1.130 | 1.129 | | 1.194 | 1.191 | 1.193 | 1.190 |

Bold typeset indicates the best performance in each recommender algorithm. * indicates statistical significance with $p$-value $<0.01$.

**Table 6**
Comparison between ECoRec with three recommenders and standalone KNN-based and Matrix Factorization-based recommenders in terms of F-Measure.

| Recommenders | User-KNN | | | | | Item-KNN | | | | | SVD | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| User-Item Matrix | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^3$ | $T_L^E$ | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^3$ | $T_L^E$ | $T_L$ | $T_L^1$ | $T_L^2$ | $T_L^3$ | $T_L^E$ |
| **Yahoo Movies** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.837 | 0.840 | 0.839 | 0.842 | | 0.831 | 0.832 | 0.831 | 0.838 | | 0.831 | 0.832 | 0.832 | 0.837 |
| 30 new ratings | 0.834 | 0.838 | 0.841 | 0.840 | 0.843 | 0.827 | 0.830 | 0.831 | 0.831 | 0.838 | 0.825 | 0.830 | 0.832 | 0.831 | 0.839 |
| 40 new ratings | | 0.838 | 0.844 | 0.839 | **0.850*** | | 0.833 | 0.837 | 0.836 | **0.841** | | 0.831 | 0.838 | 0.839 | **0.845*** |
| **FilmTrust** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.623 | 0.615 | 0.628 | 0.630 | | 0.633 | 0.632 | 0.631 | 0.638 | | 0.643 | 0.642 | 0.642 | 0.649 |
| 30 new ratings | 0.608 | 0.622 | 0.625 | 0.637 | 0.641 | 0.617 | 0.633 | 0.633 | 0.632 | 0.637 | 0.630 | 0.643 | 0.647 | 0.6422 | 0.647 |
| 40 new ratings | | 0.638 | 0.627 | 0.637 | **0.651*** | | 0.639 | 0.635 | 0.632 | **0.645*** | | 0.644 | 0.648 | 0.645 | **0.652*** |
| **CiaoDVD** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.805 | 0.807 | 0.802 | 0.815 | | 0.801 | 0.802 | 0.800 | 0.808 | | 0.831 | 0.837 | 0.841 | 0.847 |
| 30 new ratings | 0.797 | 0.806 | 0.809 | 0.807 | 0.819 | 0.796 | 0.802 | 0.803 | 0.804 | 0.811 | 0.828 | 0.832 | 0.833 | 0.833 | 0.839 |
| 40 new ratings | | 0.806 | 0.814 | 0.819 | **0.825** | | 0.812 | 0.813 | 0.814 | **0.827*** | | 0.835 | 0.837 | 0.839 | **0.848*** |
| **MovieLens 2*k*** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.804 | 0.812 | 0.809 | **0.819*** | | 0.809 | 0.816 | 0.809 | **0.821*** | | 0.809 | 0.810 | 0.810 | **0.819** |
| 30 new ratings | 0.783 | 0.804 | 0.804 | 0.803 | 0.808 | 0.803 | 0.810 | 0.810 | 0.809 | 0.812 | 0.805 | 0.810 | 0.810 | 0.810 | 0.816 |
| 40 new ratings | | 0.804 | 0.805 | 0.804 | 0.810 | | 0.810 | 0.811 | 0.810 | 0.818 | | 0.810 | 0.811 | 0.811 | 0.817 |
| **Jester** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.580 | 0.586 | 0.581 | 0.598 | | 0.563 | 0.567 | 0.560 | 0.570 | | 0.543 | 0.555 | 0.556 | 0.562 |
| 30 new ratings | 0.526 | 0.576 | 0.585 | 0.572 | 0.581 | 0.556 | 0.564 | 0.566 | 0.557 | 0.568 | 0.544 | 0.564 | 0.566 | 0.566 | 0.572 |
| 40 new ratings | | 0.589 | 0.595 | 0.596 | **0.611*** | | 0.576 | 0.577 | 0.573 | **0.586*** | | 0.558 | 0.567 | 0.567 | **0.579** |
| **Booking Crossing** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.893 | 0.894 | 0.891 | 0.901 | | 0.874 | 0.875 | 0.873 | 0.882 | | 0.884 | 0.891 | 0.895 | 0.899 |
| 30 new ratings | 0.877 | 0.892 | 0.897 | 0.895 | 0.915 | 0.852 | 0.873 | 0.877 | 0.875 | 0.889 | 0.887 | 0.893 | 0.897 | 0.897 | 0.900 |
| 40 new ratings | | 0.904 | 0.899 | 0.907 | **0.927*** | | 0.886 | 0.888 | 0.886 | **0.893** | | 0.910 | 0.901 | 0.915 | **0.932*** |
| **Amazon Digital Music** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.856 | 0.858 | 0.856 | 0.866 | | 0.841 | 0.843 | 0.842 | 0.848 | | 0.851 | 0.856 | 0.854 | 0.862 |
| 30 new ratings | 0.833 | 0.857 | 0.863 | 0.860 | 0.869 | 0.827 | 0.844 | 0.844 | 0.844 | 0.849 | 0.849 | 0.858 | 0.857 | 0.857 | 0.860 |
| 40 new ratings | | 0.860 | 0.868 | 0.864 | **0.878*** | | 0.849 | 0.857 | 0.854 | **0.859*** | | 0.859 | 0.860 | 0.866 | **0.874*** |
| **Yahoo Music** | | | | | | | | | | | | | | | |
| 20 new ratings | | 0.471 | 0.475 | 0.473 | 0.479 | | 0.515 | 0.517 | 0.516 | 0.520 | | 0.517 | 0.518 | 0.517 | 0.522 |
| 30 new ratings | 0.457 | 0.470 | 0.474 | 0.471 | 0.477 | 0.502 | 0.511 | 0.514 | 0.512 | 0.517 | 0.465 | 0.516 | 0.514 | 0.514 | 0.520 |
| 40 new ratings | | 0.487 | 0.488 | 0.491 | **0.498*** | | 0.526 | 0.527 | 0.528 | **0.533*** | | 0.526 | 0.527 | 0.527 | **0.531*** |

Bold typeset indicates the best performance in each recommender algorithm. * indicates statistical significance with $p$-value $<0.01$.

where *Maximum possible error* is the difference between the highest and the lowest rating value in each database. In this way, *F-measure* is determined by using coverage and precision as follows:

$$\textit{F-measure} = \frac{2 \cdot \textit{precision} \cdot \textit{coverage}}{\textit{precision} + \textit{coverage}} \quad , \tag{17}$$

which reaches its best value at 1 (perfect precision and coverage) and worst at 0.

Regarding the parameters of the algorithms, we determined a collection of values which performed well for all databases. In particular, we ran experiments for $X$ (new ratings per user) equal to 10, 20, 30, 40, 50 and 100 ratings, and chose 20, 30 and 40 as they provided the best results. Besides, we found that the confident set size $M$, as expected, has an impact on the learning rate for ECoRec and CSEL. We experimented with several values of $M$ from 1000 up to 20,000 and found that 10% of the unlabeled set size ($|T_{UL}|$) favors both CSEL and ECoRec, leading to a good trade-off between convergence rate (computational cost) and accuracy across all datasets. For User-KNN and Item-KNN, we used Pearson Correlation to generated the similarity matrices, once the results indicate that this similarity is superior to other relevant measures that we tested, such as Cosine and Jaccard. For the SVD algorithm, we ran experiments for factors equal to 10, 30, 50, and chose 10 as it provided the best results in all databases. For other parameters, we used the default values of each recommender tool and library chosen.

Our approach (ECoRec) was developed in Python version 3.6,[3] and the source-code is freely available in Github.[4] The baseline competitors belong to the Case Recommender Framework version 0.0.20,[5] an open source tool developed in *Python*. We used the default framework settings for recommender algorithms.

## 5.3. Results

In the following we discuss the results in different experimental setups.

### 5.3.1. Performance of the confidence measure

This section shows a comparative evaluation of the confidence measures presented in Section 3.5 against our proposed measure (PC) using ECoRec with User-KNN and Item-KNN algorithms. The Resample measure is not considered in this evaluation since it requires running each algorithm in each iteration of co-training many times to converge, which considerably increases the computational cost of our approach. In this experiment, we used the best enriched matrix and recommender for each database using 40 new ratings per user based on a preliminary analysis. The results are summarized in Table 1.

It is clear that the proposed confidence measure provides superior performance when compared against the support for user, support for item and variability for item measures. Our confidence measure emerges as the best for all databases with a considerable accuracy gain.

### 5.3.2. ECoRec with two recommenders

The overall performances of the algorithms on the eight databases are given in Table 2. By using the enhanced training sets generated by ECoRec ($T_L^1$, $T_L^2$ and $T_L^E$), the User-KNN and Item-KNN results were improved over the traditional training set ($T_L$) in terms of overall RMSE in all experiments.

We note that adding more users' ratings using the proposed ECoRec algorithm with neighborhood-based models yielded better results than the baselines using the original database. This is because the enriched data generated by ECoRec are populated with examples confidently predicted by different recommender algorithms as different views of the users' preferences. In addition, we note that the enriched set generated by the ensemble ($T_L^E$) outperforms both baselines and their results from the individual enriched sets ($T_L^1$ and $T_L^2$) in almost all cases, showing the effectiveness of the ensemble process. The results obtained by Item-KNN using $T_L^E$ were the best overall for all experiments.

Table 4shows the results comparing the baseline with our approach, using the best enriched matrix and recommender for each database from Table 2. For the baselines, we used the best final result generated by each approach. In all eight databases, ECoRec achieved better results than the other baselines.

### 5.3.3. ECoRec with three recommenders

Table 5shows the results of the generated enriched sets ($T_L^1$, $T_L^2$, $T_L^3$ and $T_L^E$) using both neighborhood models (User-KNN and Item-KNN) as well as SVD. The results show that even though individual recommenders may perform worse than others, they still contribute positively to the ensemble, which considerably outperforms the baselines.

The comparison between our approach and CSEL in this experiment is depicted in Table 7, once again using the best enriched matrix and recommender for each database from Table 5 in the case of ECoRec, and the best among all the results obtained from CSEL. Overall, ECoRec achieved better results than CSEL also in this scenario.

### 5.3.4. Cold-start problem

In order to explore the effectiveness of our approach to mitigate the cold-start problem, we report the recommendation performance for a subset of users according to their popularity. We estimate the popularity of each user based on the number of their ratings, and select a subset of users with less than 20 interactions in the training set. We then average the RMSE scores obtained for those users in our first experiment in Section 5.3.2. The results are displayed in Table 8 for the KNN-based recommenders on the augmented $T_L^E$ dataset generated by ECoRec. We evaluate the gains when we gradually added 20, 30 and 40 new ratings per user.

Table 8shows how the cold-start problem can be addressed by the proposed ECoRec approach. Compared to the baseline algorithms, our ensemble approach successfully tackles the cold-start problem by largely improving the performance of the unpopular users in most of the experiments.

### 5.3.5. Sparsity problem

The sparsity of a rating matrix is defined as (Somboonviwat & Aoyama, 2016):

$$\textit{Sparsity} = \left(1 - \frac{\textit{Number of Ratings}}{\textit{Number of users} \times \textit{Number of items}}\right)100\% \tag{18}$$

ECoRec alleviates sparsity by labeling new examples and thereby expanding the original dataset used to train recommender systems. In order to quantify the achieved label expansion, Table 9 displays the databases before and after the co-training process with two and three views when using 20, 30 and 40 new unlabeled items per user.

We note that the reduction of sparsity is directly related to the improvement of the accuracy of the results, since the more labeled examples the better the ability of the recommender algorithms to learn the users' behavior. In some databases, such as CiaoDVD, Booking Crossing and Amazon Digital Music, despite the small percentage reduction in sparsity, the results exhibited considerable accuracy improvements when compared to the original training set.

[3] http://www.python.org/

[4] https://github.com/caserec/CaseRecommender

[5] https://pypi.python.org/pypi/CaseRecommender

**Table 7**
Comparison between ECoRec and CSEL with three recommenders in terms of RMSE. Bold typeset indicates the best performance. * indicates statistical significance with $p$-value <0.01.

| Database | Measure / Approach | ECoRec | CSEL | Co-CF | AB.RT |
|---|---|---|---|---|---|
| **Yahoo Movies** | *RMSE* | **0.9467*** | 0.9892 | 0.9578 | 0.9704 |
| | *F-Measure* | **0.8507*** | 0.8254 | 0.8323 | 0.8260 |
| **FilmTrust** | *RMSE* | **0.8079*** | 0.8436 | 0.8481 | 0.8327 |
| | *F-Measure* | **0.6529*** | 0.6174 | 0.6081 | 0.6148 |
| **CiaoDVD** | *RMSE* | **0.9626*** | 0.9867 | 0.9988 | 0.9761 |
| | *F-Measure* | **0.8486** | 0.8254 | 0.8019 | 0.8332 |
| **MovieLens 2*k*** | *RMSE* | **0.7446** | 0.7689 | 0.7772 | 0.7507 |
| | *F-Measure* | **0.8218*** | 0.7982 | 0.7873 | 0.8023 |
| **Jester** | *RMSE* | **0.9494*** | 0.9695 | 0.9623 | 0.9573 |
| | *F-Measure* | **0.6118*** | 0.5782 | 0.5691 | 0.5820 |
| **Booking Crossing** | *RMSE* | **0.7572*** | 0.8416 | 0.7920 | 0.8045 |
| | *F-Measure* | **0.9325*** | 0.8927 | 0.8843 | 0.8906 |
| **Amazon Digital Music** | *RMSE* | 0.8661 | 0.8857 | **0.8575*** | 0.8738 |
| | *F-Measure* | **0.8784*** | 0.8675 | 0.8476 | 0.8532 |
| **Yahoo Music** | *RMSE* | **1.1186*** | 1.2132 | 1.3645 | 1.2472 |
| | *F-Measure* | **0.5339*** | 0.5124 | 0.4950 | 0.5173 |

**Table 8**
RMSE for KNN-based recommenders before ($T_L$) and after ECoRec ($T_L^E$) with 20, 30 and 40 new ratings per user. The results displayed are only for those users with less than 20 interactions in the training set.

| Measures | User-KNN | | | | Item-KNN | | | |
|---|---|---|---|---|---|---|---|---|
| | $T_L$ | ECoRec ($T_L^E$) | | | $T_L$ | ECoRec ($T_L^E$) | | |
| | | Unlabeled item per user ($X$) | | | | Unlabeled item per user ($X$) | | |
| | | 20 | 30 | 40 | | 20 | 30 | 40 |
| **Yahoo Movies** (***484 new users***) | | | | | | | | |
| RMSE | 1.1929 | 1.1419 | 1.1389 | **1.1361*** | 1.1635 | 1.1570 | 1.1570 | 1.1434 |
| F-Measure | 0.8461 | 0.8589 | 0.8590 | **0.8667*** | 0.8405 | 0.8542 | 0.8535 | 0.8587 |
| **FilmTrust** (***889 new users***) | | | | | | | | |
| RMSE | 0.8030 | 0.8072 | 0.7924 | 0.7710 | 0.8100 | 0.7942 | 0.7789 | **0.7612*** |
| F-Measure | 0.8161 | 0.8539 | 0.8552 | 0.8501 | 0.8064 | 0.8459 | 0.8461 | **0.8622*** |
| **CiaoDVD** (***1,658 new users***) | | | | | | | | |
| RMSE | 1.0933 | **0.9575*** | 1.1668 | 1.2148 | 1.1350 | 1.1308 | 1.0954 | 0.9988 |
| F-Measure | 0.8018 | 0.8331 | 0.8338 | **0.8453*** | 0.7939 | 0.8253 | 0.8258 | 0.8313 |
| **MovieLens** 2*k*(***113 new users***) | | | | | | | | |
| RMSE | 1.0055 | 0.9975 | 0.9827 | **0.9554** | 0.9655 | 0.9640 | 0.9641 | 0.9585 |
| F-Measure | 0.8848 | 0.8952 | 0.8957 | 0.9097 | 0.8960 | 0.9076 | 0.9076 | **0.9249*** |
| **Jester** (***43 new users***) | | | | | | | | |
| RMSE | 1.0027 | 1.0001 | 1.0019 | 0.9935 | 1.0027 | 0.9590 | **0.9584** | 0.9650 |
| F-Measure | 0.8043 | 0.8489 | 0.8464 | **0.8654*** | 0.8167 | 0.8268 | 0.8231 | 0.8354 |
| **Booking Crossing** (***2,675 new users***) | | | | | | | | |
| RMSE | 1.2438 | 1.1069 | 0.8655 | **0.7968*** | 1.0844 | 0.9851 | 0.8183 | 0.8143 |
| F-Measure | 0.8924 | 0.9145 | 0.9149 | **0.9286*** | 0.8981 | 0.9006 | 0.9007 | 0.8986 |
| **Amazon Digital Music** (***5,426 new users***) | | | | | | | | |
| RMSE | 1.0817 | 1.0758 | 0.9439 | 0.8974 | 1.0637 | 1.0566 | 0.9371 | **0.8326*** |
| F-Measure | 0.8470 | 0.8678 | 0.8681 | 0.8728 | 0.8562 | 0.8619 | 0.8618 | **0.8748** |
| **Yahoo Music** (***664 new users***) | | | | | | | | |
| RMSE | 1.2313 | 1.1926 | 1.1962 | 1.2084 | 1.2313 | **1.1731*** | 1.1769 | 1.1797 |
| F-Measure | 0.7062 | 0.7378 | 0.7349 | **0.7469*** | 0.6899 | 0.7086 | 0.7040 | 0.7369 |

Bold typeset indicates the best performance. * indicates statistical significance with $p$-value <0.01.

### *5.4. Discussion*

The experiments show that, by using ECoRec, we obtain better results than by using recommenders trained with the original data in all databases. This is due to the fact that we can infer new users' feedback, reducing the sparseness and cold-start of the original database with a certain confidence based on the multiple views of different recommender algorithms.

Our co-training approach was able to successfully increase the number of ratings in the training sets, leading to lower RMSE and higher F-Measure using two or more recommenders, as shown in Tables 2, 3, 5 and 6. This improvement can be explained by the following two reasons. First, ECoRec ranks all ratings using the confidence measure, giving less importance to unreliable ratings, such as pairs $(u, i)$ with few interactions and largely deviating from the average scores. The other reason is that it combines the individual results generated by each recommender in the final recommendation, driving them to agree with each other and improving the accuracy of predictions throughout iterations and covering as many pairs as possible.

The results presented in Section 5.3 show that ECoRec, especially with the ensemble approach, was able to achieve better accuracy in databases from four different domains (books, movies, music and jokes), with statistical significance. Although some recommenders provided worse predictions than others individually, during the co-training process they complement each other and generate better results than the baselines. Most of the results showed improvement in the accuracy of recommendation, especially when using more than ten new ratings per user in the ensemble co-training process.

**Table 9**
Sparsity of databases before and after ECoRec with 20, 30 and 40 new ratings per user using two and three recommenders (views).

| | | Unlabeled items per user ($X$) | | |
|---|---|---|---|---|
| Database/approach | Original ($T_L$) | 20 | 30 | 40 |
| **Yahoo Movies** | 99.85% | 98.46% | 98.06% | 97.66% |
| **FilmTrust** | 98.93% | 97.18% | 96.28% | 95.40% |
| **CiaoDVD** | 99.84% | 99.57% | 99.43% | 99.29% |
| **MovieLens 2*K*** | 96.37% | 96.27% | 96.07% | 95.87% |
| **Jester** | 33.65% | 23.48% | 16.97% | 12.17% |
| **Booking Crossing** | 99.66% | 98.93% | 98.55% | 98.17% |
| **Amazon Digital Music** | 99.81% | 99.06% | 98.67% | 98.29% |
| **Yahoo Music** | 96.28% | 93.00% | 91.16% | 89.31% |

The number of new ratings per user ($X$) affects performance in a reasonably predictable way. Typically, increasing $X$ will tend to increase performance, but only to a certain data-dependent extent, beyond which performance may plateau or start degrading. The reason for this behavior is that, as $X$ increases, the proportion of estimated ratings increases as compared to the proportion of real ratings in the augmented user-item matrix. Since the estimated ratings are subject to estimation errors, there is a point beyond which the gains from reducing sparsity no longer overweigh the increasing uncertainty in the data. In our experiments, performance dropped in most databases when $X > 50$, due to the fact that samples become less reliable (ratings tend to be more global rather than reflect user's behavior) and the method thus becomes more sensitive to noise.

In CSEL (Zhang et al., 2014) and in Co-CF (Quang et al., 2015), the authors chose the best sample of the unlabeled set based on their confidence measure, whereas we randomly select $X$ new items for each user from the database. We can see in Tables 4 and 7 that ECoRec outperforms CSEL and Co-CF with two or three recommenders in the most cases. Our approach, in which the distribution of new data is the same for all users, has thus been shown to be more effective to mitigate cold-start and sparsity. Moreover, our approach also outperforms AdaBoost.RT (Bar et al., 2013).

The computational cost of the algorithm is approximately $t$ times the cost of the recommender algorithms used, since the individual models are re-trained $t$ times during the co-training process. This additional burden can be lessened by using task parallel libraries and computer clusters, as the multiple recommenders can be trained independently in each iteration of the algorithm. In our experiments, we parallelized the recommenders, confidence calculation and ensemble steps, using a native Python library.[6] It is worth noticing that the entire co-training processes can be performed off-line, so it is by no means a bottleneck for real-world applications.

## 6. Final remarks

This paper proposed an ensemble-based co-training approach, named ECoRec, to process data from two or more different views in order to create a more precise and robust model for recommendation. In our experiments, these views were accomplished by different recommender algorithms, namely, User-KNN, Item-KNN, and SVD.

We conducted experiments in eight databases from different domains (movies, books, jokes and music) and the results show that our strategy improves the overall system's performance in a variety of experimental setups. The main advantage of our approach is to provide an enriched user-item matrix that helps mitigate the sparsity and cold-start problems. The proposed ECoRec approach is extensible and flexible, as it enables developers to use different recommender algorithms, confidence measures, and ensemble strategies.

As future work, we aim to assess some of these possible extensions. We also intend to investigate how to incorporate different types of feedback, for example ratings and sentiment review, so they can improve the accuracy of the predictions when they are available. Finally, we intend to investigate an active learning technique for linear regression as part of the ensemble step, in order to weight each recommenders' estimated accuracy as evaluated during the co-training process.

## Acknowledgments

This work was supported by the Coordination for the Improvement of Higher Education Personnel (CAPES), the São Paulo Research Foundation, FAPESP [2016/20280-6], and the Brazilian National Research Council, CNPq [304137/2013-8].

## References

Adomavicius, G., Kamireddy, S., & Kwon, Y. (2007). Towards more confident recommendations: Improving recommender systems using filtering approach based on rating variance. In *WITS 2007 - proceedings, 17th annual workshop on information technologies and systems* (pp. 152–157). Social Science Research Network.

Aggarwal, C. C. (2016). *Recommender systems: The textbook* (1st). Springer Publishing Company, Incorporated.

Bar, A., Rokach, L., Shani, G., Shapira, B., & Schclar, A. (2013). Improving simple collaborative filtering models using ensemble methods. In *Multiple classifier systems: 11th international workshop*. In *MCS '13: 7872* (pp. 1–12). Springer Berlin Heidelberg. doi:10.1007/978-3-642-38067-9_1.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on computational learning theory*. In *COLT' 98* (pp. 92–100). New York, NY, USA: ACM. doi:10.1145/279943.279962.

Bobadilla, J., Gutirrez, A., Ortega, F., & Zhu, B. (2018). Reliability quality measures for recommender systems. *Information Sciences, 442–443*, 145–157. doi:10.1016/j.ins.2018.02.030.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems, 46*, 109–132. doi:10.1016/j.knosys.2013.03.012.

Box, J. F. (1987). Guinness, gosset, fisher, and small samples. *Statistical Science, 2*(1), 45–52. doi:10.1214/ss/1177013437.

Cantador, I., Brusilovsky, P., & Kuflik, T. (2011). 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In *RecSys 2011* (pp. 387–388). New York, NY, USA: ACM.

Castells, P., Hurley, N. J., & Vargas, S. (2015). Novelty and diversity in recommender systems. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (pp. 881–918)). Boston, MA: Springer US. doi:10.1007/978-1-4899-7637-6_26.

Da Costa, A. F., Manzato, M. G., & Campello, R. J. G. B. (2018). Corec: A co-training approach for recommender systems. *SAC '18*. New York, NY, USA: ACM.

Goldberg, K., Roeder, T., Gupta, D., & Perkins, C. (2001). Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval, 4*(2), 133–151. doi:10.1023/A:1011419012209.

Guo, G. (2013). Improving the performance of recommender systems by alleviating the data sparsity and cold start problems. In *Proceedings of the 23rd international joint conference on artificial intelligence*. In *IJCAI '13* (pp. 3217–3218). AAAI Press.

Guo, G., Zhang, J., Thalmann, D., & Yorke-Smith, N. (2014). Etaf: An extended trust antecedents framework for trust prediction. In *Proceedings of the 2014 international conference on advances in social networks analysis and mining*. In *ASONAM '14* (pp. 540–547).

Guo, G., Zhang, J., & Yorke-Smith, N. (2013). A novel bayesian similarity measure for recommender systems. In *Proceedings of the 23rd international joint conference on artificial intelligence*. In *IJCAI '13* (pp. 2619–2625).

Hernando, A., Bobadilla, J., Ortega, F., & Tejedor, J. (2013). Incorporating reliability measurements into the predictions of a recommender system. *Information Sciences, 218*, 1–16. doi:10.1016/j.ins.2012.06.027.

Jahrer, M., Töscher, A., & Legenstein, R. (2010). Combining predictions for accurate recommender systems. In *Proceedings of the 16th ACM SIGKDDinternational conference on knowledge discovery and data mining*. In *KDD '10* (pp. 693–702). New York, NY, USA: ACM. doi:10.1145/1835804.1835893.

Karimi, R., Freudenthaler, C., Nanopoulos, A., & Schmidt-Thieme, L. (2011). Towards optimal active learning for matrix factorization in recommender systems. In *2011 IEEE 23rd international conference on tools with artificial intelligence* (pp. 1069–1076). Washington, DC, USA: IEEE Computer Society. doi:10.1109/ICTAI.2011.182.

Koren, Y. (2008). Factorization meets the neighborhood. In *KDD '08* (pp. 426–434). New York, NY, USA: ACM. doi:10.1145/1401890.1401944.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8), 30–37. doi:10.1109/MC.2009.263.

[6] https://docs.python.org/3/library/multiprocessing.html

Lee, J.-S., & Olafsson, S. (2009). Two-way cooperative prediction for collaborative filtering recommendations. *Expert Systems with Applications, 36*(3, Part 1), 5353–5361. doi:10.1016/j.eswa.2008.06.106.

Matuszyk, P., & Spiliopoulou, M. (2017). Stream-based semi-supervised learning for recommender systems. *Machine Learning, 106*(6), 771–798. doi:10.1007/s10994-016-5614-4.

Mazurowski, M. A. (2013). Estimating confidence of individual rating predictions in collaborative filtering recommender systems. *Expert Systems with Applications, 40*(10), 3847–3857. doi:10.1016/j.eswa.2012.12.102.

McAuley, J., Pandey, R., & Leskovec, J. (2015). Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. In *KDD '15* (pp. 785–794). New York, NY, USA: ACM. doi:10.1145/2783258.2783381.

Quang, T. N., Lien, D. T., & Phuong, N. D. (2015). Collaborative filtering by co-training method. In V.-H. Nguyen, A.-C. Le, & V.-N. Huynh (Eds.), *Knowledge and systems engineering* (pp. 273–285). Cham: Springer International Publishing.

Ravi, L., & Vairavasundaram, S. (2016). A collaborative location based travel recommendation system through enhanced rating prediction for the group of users. *Intelligence and Neuroscience, 2016*, 7. doi:10.1155/2016/1291358.

Ricci, F., Rokach, L., & Shapira, B. (2015). *Recommender systems handbook* (2nd). New York, NY, USA: Springer Publishing Company, Incorporated.

Ristoski, P., Mencia, E. L., & Paulheim1, H. (2014). A hybrid multi-strategy recommender system using linked open data. In *Semantic web evaluation challenge: Semwebeval 2014*. In *ESWC '14* (pp. 150–156). Cham

Schclar, A., Tsikinovsky, A., Rokach, L., Meisels, A., & Antwarg, L. (2009). Ensemble methods for improving the performance of neighborhood-based collaborative filtering. In *Proceedings of the third ACM conference on recommender systems*. In *RecSys '09* (pp. 261–264). New York, NY, USA: ACM. doi:10.1145/1639714.1639763.

Somboonviwat, K., & Aoyama, H. (2016). Empirical analysis of the relationship between trust and ratings in recommender systems. In N. T. Nguyen, B. Trawiński, H. Fujita, & T.-P. Hong (Eds.), *Intelligent information and database systems: 8th Asian conference*. In *ACIIDS '16* (pp. 116–126). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:10.1007/978-3-662-49381-6_12.

Su, X., Khoshgoftaar, T. M., Zhu, X., & Greiner, R. (2008). Imputation-boosted collaborative filtering using machine learning classifiers. In *Proceedings of the 2008 ACM symposium on applied computing*. In *SAC '08* (pp. 949–950). New York, NY, USA: ACM. doi:10.1145/1363686.1363903.

Sun, M., Li, F., Lee, J., Zhou, K., Lebanon, G., & Zha, H. (2013). Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the sixth ACM international conference on web search and data mining*. In *WSDM '13* (pp. 445–454). New York, NY, USA: ACM. doi:10.1145/2433396.2433451.

Xu, C., Tao, D., & Xu, C. (2013). A survey on multi-view learning. *Computing Research Repository - CoRR*. abs/1304.5634

Zhang, D., & Lee, W. S. (2005). Validating co-training models for web image classification. *Computer Science (CS); Singapore-MIT Alliance (SMA)*.

Zhang, M., Tang, J., Zhang, X., & Xue, X. (2014). Addressing cold start in recommender systems: A semi-supervised co-training algorithm. In *Proceedings of the 37th international ACM SIGIR conference on research and development in information retrieval*. In *SIGIR '14* (pp. 73–82). New York, NY, USA: ACM. doi:10.1145/2600428.2609599.

Zhang, Q., & Wang, H. (2015). Collaborative multi-view learning with active discriminative prior for recommendation. In *Advances in knowledge discovery and data mining: 19th Pacific-Asia conference, PAKDD 2015, Ho Chi Minh City, Vietnam, may 19–22, 2015, proceedings, Part I* (pp. 355–368). Cham: Springer International Publishing. doi:10.1007/978-3-319-18038-0_28.

Zhou, Z.-H., & Li, M. (2005). Semi-supervised regression with co-training. In *Proceedings of the 19th international joint conference on artificial intelligence*. In *IJCAI'05* (pp. 908–913). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Zhu, T., Hu, B., Yan, J., & Li, X. (2010). Semi-supervised learning for personalized web recommender system. *Computing and Informatics, 29*, 617–627.

Ziegler, C.-N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on world wide web*. In *WWW '05* (pp. 22–32). New York, NY, USA: ACM. doi:10.1145/1060745.1060754.