Complex Systems Informatics and Modeling Quarterly
CSIMQ, Article 75, Issue 13, December 2017/January 2018, Pages 22–42
Published online by RTU Press, https://csimq-journals.rtu.lv
https://doi.org/10.7250/csimq.2017-13.02
ISSN: 2255-9922 online

Using Time Clusters for Following Users' Shifts in Rating Practices

Dionisis Margaris[1] and Costas Vassilakis[2*]

[1] Department of Informatics and Telecommunications, University of Athens,
Panepistimioupoli, Athens, 15784, Athens, Greece
[2] Department of Informatics and Telecommunications, University of the Peloponnese,
Akadimaikou G.K. Vlachou, Tripoli, 22100, Greece

margaris@di.uoa.gr, costas@uop.gr

Abstract. Users that enter ratings for items follow different rating practices, in the sense that, when rating items, some users are more lenient, while others are stricter. This aspect is taken into account by the most widely used similarity metric in user-user collaborative filtering, namely, the Pearson Correlation, which adjusts each individual user rating by the mean value of the ratings entered by the specific user, when computing similarities. However, a user's rating practices change over time, i.e. a user could start as strict and subsequently become lenient or vice versa. In that sense, the practice of using a single mean value for adjusting users' ratings is inadequate, since it fails to follow such shifts in users' rating practices, leading to decreased rating prediction accuracy. In this work, we address this issue by using the concept of dynamic averages introduced earlier and we extend earlier work by (1) introducing the concept of rating time clusters and (2) presenting a novel algorithm for calculating dynamic user averages and exploiting them in user-user collaborative, filtering implementations. The proposed algorithm incorporates the aforementioned concept and is able to follow more successfully shifts in users' rating practices. It has been evaluated using numerous datasets, and has been found to introduce significant gains in rating prediction accuracy, while outperforming the dynamic average computation approaches that are presented earlier.

Keywords: Recommender systems, collaborative filtering, rating time clusters, dynamic average, rating abstention interval, ratings' timestamps.

1 Introduction

Collaborative filtering (CF), which is the most successful and most applied technique in the design of recommender systems [1], computes personalized recommendations, by taking into account the users' past likings and tastes, in the form of ratings entered in the CF rating database.

* Corresponding author

Reference: D. Margaris and C. Vassilakis, "Using Time Clusters for Following Users' Shifts in Rating Practices," *Complex Systems Informatics and Modeling Quarterly*, CSIMQ, no. 13, pp. 22–42, 2017. Available: https://doi.org/10.7250/csimq.2017-13.02

Additional information. Author's ORCID iD: D. Margaris – orcid.org/0000-0002-7487-374X, C. Vassilakis – orcid.org/0000-0001-9940-1821. Article PII S225599221700075X. Article received: 2017 October 01. Accepted: 2017 December 21. Available online: 2017 December 29.

User-user CF algorithms firstly identify people having similar tastes, by examining the resemblance of already entered ratings; for each user *u*, other users having highly similar tastes with *u* are designated as u's nearest neighbors (NNs). Afterwards, in order to predict the rating that *u* would give to an item i, that u has not reviewed yet, the ratings assigned to item *i* by *u*'s NNs are combined [2], under the assumption that users are highly likely to exhibit similar tastes in the future, if they have done so in the past as well [3], [4]. To measure similarity between users, the Pearson Correlation Coefficient is the most commonly used formula in CF recommender systems. In this context, the Pearson correlation coefficient adjusts the ratings of a user *u* by the mean value of all ratings entered by *u*, so as to tackle the issue that some users may rate items either higher or lower than others. However, relying on a single mean value presumes that the users' marking practices remain constant over time; in practice though, a user's marking practices may change over time, i.e. a user could start off being lenient and subsequently change to being strict, or vice versa.

For instance, consider that a user watches 8 movies, day after day and rates them from 4/10 to 7/10 (assuming a uniform distribution, starting with 4/10 and ending with 7/10); hence a movie that has been rated with 4/10 is considered as a relatively bad one. Afterwards, the user abstains from watching/rating movies for 5 days and then she watches 4 movies, one every two days, rating them from 7/10 to 10/10 (starting with 10/10 and ending with 7/10, this time). We can understand that the first movie of the second rating set, that has been rated with 10/10 is considered as an excellent one, but we cannot be certain how the first period movie that has been rated with 7/10 compares with the second period movie that has been rated with the same mark: it may be the case that after the user has watched the movie that she considered excellent and rated with 10/10, her standards have risen, therefore in reality the second period movie that was rated with 7/10 is actually better than the equally ranked first period movie. It is also possible that the ratings entered by the user during the two periods were affected by a change in her mood [5]. In any case, the user's 5-day abstention signifies a change in her rating practices. The Pearson Correlation, which is predominantly used for computing user similarity in CF algorithms, does not consider such changes in rating practices, and therefore inaccuracies may be introduced in the user neighborhood computation process.

Insofar, while many efforts have been made to improve the CF prediction accuracy, and the aspect of changes in users' interests has been extensively studied ([6] provides a comprehensive review), the issue of shifts in rating practices has not received adequate attention. Margaris and Vassilakis [7] introduce the concept of dynamic user rating averages which follow the users' marking practices shifts and present two alternative algorithms, namely the $DA_{vicinity}$ and $DA_{previous}$ algorithms, for computing a user's dynamic averages. These algorithms are validated in the context of user-user CF, and have been found to achieve better rating prediction accuracy than the plain CF algorithm.

In this article, we extend the work in [7] as follows:

(1) We introduce the concept of *rating time clusters*; a rating time cluster corresponds to a group of ratings which have some temporal cohesion and for which the user is assumed to follow the same rating practice

(2) We present a novel algorithm, namely $DA_{clusters}$, for computing rating time clusters. After formulating rating time clusters, $DA_{clusters}$ computes one dynamic average per cluster; this dynamic average is then used in the rating prediction process. To validate our approach, we present an extensive evaluation, comparing the presented algorithm against the $DA_{previous}$ and $DA_{vicinity}$ algorithms proposed in [7] and the plain CF algorithm, which is used as a yardstick. Experiments have shown that using the *per cluster* dynamic averages, which are computed by the $DA_{clusters}$ algorithm leads to more accurate predictions as compared to the *per-rating* dynamic averages that are computed by the $DA_{previous}$ and $DA_{vicinity}$ algorithms proposed in [7]. Improvement in accuracy indicates that the $DA_{clusters}$ algorithm is able to follow more accurately shifts in rating prediction practices. The $DA_{clusters}$ algorithm also

introduces significant space savings in comparison to the $DA_{previous}$ and $DA_{vicinity}$ algorithms [7].

The proposed algorithm, as well as the two algorithms presented in [7], are based on the exploitation of timestamp information which is associated with ratings; hence in this work, we use the Amazon datasets [8], [9], the MovieLens datasets [10], [11] and the Netflix dataset [12], which include the ratings' timestamps. It is worth noting that the proposed algorithm can be combined with other techniques that have been proposed for either improving prediction accuracy in CF-based systems, including consideration of social network data (e.g. [13], [14], [15]), location data [16], [17] and pruning of old user ratings [18], [19], or techniques for speeding up prediction computation time, such as clustering [20], [21], [22]. The rest of the article is structured as follows: Section 2 overviews related work, while Section 3 introduces the proposed algorithm and briefly describes the dynamic average-based algorithms presented in [7] for self-containment purposes. Section 4 evaluates the proposed algorithm using the aforementioned datasets and finally, Section 5 concludes the article and outlines future work.

2 Related Work

The accuracy of CF-based systems is a topic that has attracted considerable research efforts. Koren [23] proposes a new neighborhood-based model, which is based on formally optimizing a global cost function and leads to improved prediction accuracy, while maintaining merits of the neighborhood approach such as explainability of predictions and ability to handle new ratings (or new users) without retraining the model. In addition, it suggests a factorized version of the neighborhood model, which improves its computational complexity while retaining prediction accuracy. Liu et al. [24] present a new user similarity model to improve the recommendation performance when only few ratings are available to calculate the similarities for each user. The model considers the local context information of user ratings, as well as the global preference of user behavior. Ramezani et al. [25] propose a method to find the neighbor users based on the users' interest patterns in order to overcome challenges like sparsity and computational issues, following the idea that users who are interested in the same set of items share similar interest patterns, therefore, the non-redundant item subspaces are extracted to indicate the different patterns of interest and then, a user's tree structure is created based on the patterns she has in common with the active user.

Research has shown that exploiting time in the rating prediction computation can improve prediction accuracy, due to concept drift; concept drift is the phenomenon when the relation between the input data and the target variable changes over time [6]. Change of interests [26], [6], is a typical example of concept drift. To this end, Zliobaite et al. [27] develop an intelligent approach for sales prediction, which uses a mechanism for model switching, depending on the sales behavior of a product. This research presents an intelligent two level sales prediction approach that switches the predictors depending on the properties of the historical sales, such as product moving average sales, cumulative sales, holidays and seasonal sales. This approach is shown to achieve better results as compared to both (a) a baseline predictor and (b) an ensemble of predictors. Ang et al. [28] address the problem of adaptation when external changes are asynchronous, by developing an ensemble approach, called PINE, which combines reactive adaptation via drift detection, and proactive handling of upcoming changes via early warning and adaptation across the peers. In addition, PINE is parameter-insensitive and incurs less communication cost while achieving better accuracy. Elwell and Polikar [29] tackle the issue of concept drift in the context of online learning, introducing a batch-based ensemble of classifiers, called Learn++.NSE, where NSE stands for Non-Stationary Environments. Learn++.NSE learns from consecutive batches of data without making any assumptions on the nature or rate of drift; it can learn from such environments that experience constant or variable rate of drift, addition or deletion of concept classes, as well as cyclical drift. The algorithm learns incrementally, as other members of the Learn++ family of algorithms, that is, without requiring access to previously seen data. Learn++.NSE trains one new classifier for each batch of data it receives, and combines these classifiers using a dynamically weighted majority voting. The algorithm is evaluated on several synthetic datasets designed to simulate a variety of nonstationary environments, as well as a real-world weather prediction dataset. Minku et al. [30] present a new categorization for concept drift, separating drifts according to different criteria into mutually exclusive and non-heterogeneous categories. Moreover, they present a diversity analysis in the presence of different types of drifts and it shows that, before the drift, ensembles with less diversity obtain lower test errors. Nishida and Yamauchi [31] have developed a detection method that includes an online classifier and monitors its prediction errors during the learning, which uses a statistical test of equal proportions. Experimental results showed that this method performed well in detecting the concept drift in five synthetic datasets that contained various types of concept drift. Vaz et al. [32] propose an adaptation of the item-based CF algorithm to incorporate rating age influence in predictions. It considers ratings in two dimensions: the active user ratings and the community ratings and it inserts a time weight, which gave more relevance to more recent ratings than to older ones, both in the similarity calculation and in the rating prediction equation. Koenigstein et al. [33] consider the temporal dimension in the context of recommender systems by capturing different temporal dynamics of music ratings, along with information from the taxonomy of music-related items; both these dimensions are exploited by a rich bias model. The method proposed in this work is applied on a sparse, large-scale dataset, and the particular characteristics of the dataset are extracted and utilized. Liu et al. [34] present a social temporal collaborative ranking model that can simultaneously achieve three objectives: (1) combines both explicit and implicit user feedback, (2) supports time awareness using an expressive sequential matrix factorization model and a temporal smoothness regularization function to tackle overfitting, and (3) supports social network awareness by incorporating a network regularization term. Dias and Fonseca [35] explore the usage of temporal context and session diversity in session-based CF techniques for music recommendation. They compare two techniques to capture the users' listening patterns over time: one explicitly extracts temporal properties and session diversity, to group and compare the similarity of sessions, the other uses a generative topic modeling algorithm, which is able to implicitly model temporal patterns. Results reveal that the inclusion of temporal information, either explicitly or implicitly, increases significantly the accuracy of the recommendation, as compared to the traditional session-based CF.

Li et al. [36] study the problem of predicting the popularity of social multimedia content embedded in short microblog messages, exploiting the idea of concept drift to capture the phenomenon that through the social networks' "re-share" feature, the popularity of a multimedia item may revive or evolve. They model the social multimedia item popularity prediction problem using a classification-based approach which is used for two sub-tasks, namely re-share classification and popularity score classification. Furthermore, they develop a concept drift-based popularity predictor by ensembling multiple trained classifiers from social multimedia instances in different time intervals.

Lu et al. [37] present a novel evolutionary view of user's profile by proposing a Collaborative Evolution (CE) model, which learns the evolution of user's profiles through the sparse historical data in recommender systems and outputs the prospective user profile of the future.

Kangasrääsiö et al. [38] formulate a Bayesian regression model for predicting the accuracy of each individual user feedback and thus find outliers in the feedback data set. Additionally, they introduce a timeline interface that visualizes the feedback history to the user and provides her with suggestions on which past feedback is likely in need of adjustment. This interface also allows the user to adjust the feedback accuracy inferences made by the model. The proposed modeling technique, combined with the timeline interface, makes it easier for the users to notice and correct mistakes in their feedback, and to discover new items.

However, none of the above mentioned works considers the issue of shifts in the users' rating practices. This issue has only recently received some attention: Margaris and Vassilakis [7] introduce and exploit the concept of dynamic user rating averages which follow the users' marking practices shifts. Furthermore, they present two alternative algorithms, namely the $DA_{vicinity}$ and the $DA_{previous}$, for computing a user's dynamic averages and perform a comparative evaluation in the context of a user-user CF implementation. The results of this evaluation show that the dynamic average-based algorithms exhibit better performance than the plain CF algorithm in terms of rating prediction accuracy, at the expense of a small to tolerable drop in coverage.

This article extends the work presented in [7] by (1) introducing a more successful dynamic average computation algorithm, based on user-level rating clusters, which is able to better follow the variations of user rating practices and (2) validating its performance against widely used datasets with diverse characteristics. The newly introduced algorithm has been found to provide more accurate rating predictions by better capturing the shifts in users' rating practices. It is worth noting that the proposed algorithm is agnostic to the reasons that have led to the shifts in users' rating practices, such as adoption of different standards or changes in mood: the proposed algorithm focuses on identifying periods with distinct rating practices, and not on analyzing the reasons behind these shifts.

3 Exploiting Ratings' Timestamps in Users Dynamic Average Configuration

In CF, predictions for a user *X* are computed based on a set of users which have rated items similarly with *X*; this set of users is termed "near neighbors of *X*" (X's NNs). The similarity metric for ratings is typically based on the Pearson correlation metric, which is expressed as:

$$Pearson\_sim(X,Y) = \frac{\sum_{i \in I_X \cap I_Y} \left(R_{X,i} - \overline{R_X}\right) * \left(R_{Y,i} - \overline{R_Y}\right)}{\sqrt{\sum_{i \in I_X \cap I_Y} \left(R_{X,i} - \overline{R_X}\right)^2} * \sqrt{\sum_{i \in I_X \cap I_Y} \left(R_{Y,i} - \overline{R_Y}\right)^2}} \quad (1)$$

where *i* ranges over items that have been rated by both *X* and *Y*. The algorithms presented in this section target the computation of $\overline{R_X}$ (resp. $\overline{R_Y}$), aiming to substitute the global average, which is insensitive to shifts in rating practices, by an average that is tailored to the time period that $R_{X,i}$ (resp. $R_{Y,i}$) was entered. When a dynamic average computation algorithm DA_{Alg} is employed, the above formula is modified as:

$$Pearson\_sim(X,Y) = \frac{\sum_{i \in I_X \cap I_Y} \left(R_{X,i} - DA_{Alg}(R_{X,i})\right) * \left(R_{Y,i} - DA_{Alg}(R_{Y,i})\right)}{\sqrt{\sum_{i \in I_X \cap I_Y} \left(R_{X,i} - DA_{Alg}(R_{X,i})\right)^2} * \sqrt{\sum_{i \in I_X \cap I_Y} \left(R_{Y,i} - DA_{Alg}(R_{Y,i})\right)^2}} \quad (2)$$

3.1 Existing Dynamic Average Algorithms

In [7], two algorithms for computing dynamic user averages were proposed:

- The dynamic average based on the temporal vicinity of the ratings, which will be denoted as $DA_{vicinity}$, which follows a weighted average approach: for a rating *r*, each user rating *r'* posted by the same user is assigned a weight on the basis of its temporal vicinity to *r* (ratings that have been entered temporally close to *r* are assigned higher weights, and as temporal distance increases, the weights decrease), and finally the weighted average involving all ratings entered by the particular user is computed. The dynamic average for a rating $r_{u,i}$ on item *i* entered by user *u* is denoted as $DA_{vicinity}(r_{u,i})$ and formally is computed as

$$DA_{vicinity}\left(r_{u,i}\right) = \frac{\sum_{r \in Ratings(u)} w_{u,i}(r) * r}{\sum_{r \in Ratings(u)} w_{u,i}(r)} \quad (3)$$

  where $w_{u,i}(r)$ is the weight of rating *r* with respect to its temporal vicinity to rating $r_{u,i}$ and is calculated using formula (4):

$$w_{u,i}(\mathrm{r}) = 1 - \frac{|\, t(r_{u,x}) - t(\mathrm{r})|}{\max\limits_{r' \in Ratings(u)} (\mathrm{t}(r')) - \min\limits_{r' \in Ratings(u)} (\mathrm{t}(r'))} \tag{4}$$

In formula (4), $t(x)$ denotes the timestamp of rating x, whereas $\max\limits_{r' \in Ratings(u)} (\mathrm{t}(r'))$ and $\min\limits_{r' \in Ratings(u)} (\mathrm{t}(r'))$ denote the maximum timestamp and minimum timestamp, respectively, among the ratings entered by user u; this weight computation formula follows the standard normalization function presented in [39].

- The dynamic average based only on previous ratings, which will be denoted as $DA_{previous}$, where again each user rating $r_{u,i}$ is coupled with its own average $DA_{previous}(r_{u,i})$, however when computing this average, only ratings entered by the same user (u) prior to $r_{u,i}$ are taken into account. Formally, this is computed as

$$DA_{previous}(r_{u,i}) = \frac{\sum_{r \in Ratings(u)\ AND\ t(r) < t(r_{u,i})} r}{|r: r \in Ratings(u)\ AND\ t(r) < t(r_{u,i})|} \tag{5}$$

Results of [7] assert that both these approaches improve the CF predictions, however they both suffer from the problem of exhibiting increased storage requirements, since for each rating the dynamic average associated to the particular rating must be available, in order to compute predictions.

3.2 The Proposed Algorithm

Under the proposed approach for computing dynamic averages, instead of computing and storing a separate average for each particular rating, the ratings of each user u are partitioned into clusters with respect to their timestamps, and a single dynamic average is computed and stored for each time cluster. This approach drastically reduces the space requirements for storing the dynamic averages.

In order to group a user's u ratings into clusters, the proposed algorithm iterates over the ratings entered by u in ascending time order, following a greedy approach, so as to reduce the computational complexity of cluster formulation. More specifically, the algorithm initially considers a cluster including the two first ratings of the user and computes the average rating abstention interval between the elements of the cluster (which is initially equal to the difference of the timestamps of the two cluster elements). Subsequently, it examines if the next rating can be incorporated into the current cluster: if the abstention interval between the timestamp of the next rating and the timestamp of the latest rating within the cluster is less than the average abstention interval between consecutive ratings within the cluster, then the new rating is appended to the current cluster; in the opposite case, the current cluster is finalized and the clustering procedure is executed anew from the next rating onwards.

Effectively, this technique locates, for each user, a point in time when she has abstained from submitting ratings for a period of time which is longer than she had usually done recently (i.e. at that particular time period) and assumes that this abstention may signify a change of user's rating strictness. It is worth noting that while a number of clustering algorithms exist (e.g. k-means, k-medoids, CLARA [40]), all these algorithms require that the number of clusters is known a priori, a condition that is not met by the user rating datasets. Various techniques are presented in the literature for computing the number of clusters that will deliver the optimal clustering (e.g. [41]); the investigation of these techniques and the use of different clustering algorithms will be part of our future work.

The proposed algorithm is illustrated below in pseudocode; besides formulating clusters, the pseudocode also computes the dynamic average for each cluster.

```
// INPUT: rating database
// Output: clusters array, containing for each user the computed set of clusters
//         each cluster contains the ratings and the respective dynamic average
clusters = ∅
FOREACH user u ∈ RatingsDB
  clusters[u] = ∅
  ru = retrieveAllUserRatings(u, RatingsDB)
  sort ru on timestamp with ascending order
  cluster_id = 1
  cluster_id_set = {ru[1], ru[2]}
  // must include the first two ratings, in order to have an interval to compare with
  cluster_ratings_counter = 2
  cluster_ratings_sum = rating(ru[1]) + rating(ru[2])
  cluster_abstention_avg = timestamp(ru[2]) – timestamp(ru[1])
  FOR i = 3 TO count(ru)-1
  // we cannot have a single rating in a cluster; hence the last rating belongs by
  // default in the user’s last cluster
      current_interval = timestamp(ru[i]) – timestamp(ru[i-1])
      IF (current_interval > (FACTOR * cluster_abstention_avg) )
            // start a new cluster
            clusters[u][cluster_id].ratings = cluster_id_set
            clusters[u][cluster_id].dynamic_average = cluster_ratings_sum /
                    cluster_ratings_counter

            cluster_id++
            cluster_id_set = {ru[i], ru[i+1]}
            cluster_ratings_counter = 2
            cluster_id_dynamic_average = rating(ru[i]) + rating(ru[i+1])
            cluster_abstention_avg = timestamp(ru[i+1]) – timestamp(ru[i])

            i++ // in the next loop we consider the first unallocated rating
      ELSE
            // add rating to current cluster
            cluster_id_set = cluster_id_set ∪ {ru[i]}
            cluster_abstention_avg = (cluster_abstention_avg *
      (cluster_ratings_counter–1) + current_interval) / cluster_ratings_counter
            cluster_ratings_counter++
            cluster_ratings_sum = cluster_ratings_sum + rating(ru[i])
      END IF
  NEXT i
  // add last rating to the last computed cluster and update its dynamic average
  clusters[u][cluster_id].ratings= clusters[u][cluster_id].ratings ∪ {ru[count(ru)]}
  clusters[u][cluster_id].ratings.dynamic_average = (cluster_ratings_counter *
          clusters[u][cluster_id].ratings.dynamic_average + rating(ru[count(ru)])) /
          (cluster_ratings_counter + 1)
NEXT u
```

Due to the fact that the algorithm operates in a greedy fashion, it is prone to the formulation of an excessive number of clusters: if the first elements that are added to the cluster have small differences in their timestamps, then trivial clusters containing only two (or very few) ratings will be created. To ameliorate this effect, the constant *FACTOR* is used in the pseudocode, which adjusts the new ratings cluster detection threshold: setting *FACTOR* to values higher than 1 relaxes the criterion for incorporating each next rating into the current cluster, decreasing thus the probability that trivial clusters will be created. In our experiments, reported in Section 4, we explored different candidate values for the FACTOR parameter. More specifically, we used the following candidate *FACTOR* values: 1, 1.25, 1.5, 1.75, 2, 2.5, 3, 3.5, 4, 4.5, 5, 7.5 and 10. While, theoretically, *FACTOR* can be set to values lower than 1, this worsens the problem of trivial cluster creation, and therefore it is not considered in our evaluation.

4 Performance Evaluation

In this section, we report on our experiments through which we compared the proposed algorithm, $DA_{clusters}$, against the dynamic average-based algorithms introduced in [7], as well as the plain CF algorithm (which is used as a yardstick).

In this comparison we consider the following aspects:

1. Prediction accuracy; for this comparison, we used two well-established error metrics, namely the mean absolute error (MAE) metric, as well as the Root Mean Squared Error (RMSE) that 'punishes' big mistakes more severely. RMSE was used in the Netflix competition [12].
2. The *coverage* of the algorithm, i.e. the percentage of the cases for which a prediction can be computed [42].
3. The probability that an algorithm computes the correct user rating. Since user ratings are typically integer numbers, while predictions are calculated as real numbers, for comparing the prediction to the actual user rating we round the prediction to the nearest integer. This is analogous to the practice used in the Netflix Competition [12].
4. The number of dynamic averages the algorithm stores, in order to perform prediction computation [7].

To compute the MAE, the RMSE and the probability to compute the correct prediction, we employed the standard "hide one" technique [4]: each time, we hid a random rating in the database and then predicted its value based on the ratings of other non-hidden items. For each user, this procedure was executed for 10 randomly selected ratings entered by the particular user; therefore the computation of the MAE, the RMSE and the correct prediction probability was performed considering all users in the database.

The algorithms employing dynamic averages may exhibit different coverage, since the introduction of dynamic averages modifies the similarity metrics, and henceforth users that are deemed "similar" when using the plain CF algorithm (i.e. their standard Pearson o similarity surpasses a threshold) may be deemed "not similar" when using the dynamic average-aware Pearson similarity, or vice versa. Under this condition, some users that are characterized as "grey sheep" [42] when using the plain CF algorithm (i.e. did not have enough near neighbours for a recommendation to be computed) may gain enough neighbours when using a dynamic average-based algorithm, thus increasing coverage; conversely some users for which a recommendation was computed using the plain CF algorithm may become "grey sheep" when using a dynamic average-based algorithm because they lost some near neighbors, in which case coverage decreases.

For our experiments we used a machine equipped with six Intel Xeon E7 - 4830 @ 2.13GHz CPUs, 256GB of RAM and one 900GB HDD with a transfer rate of 200MBps, which hosted the datasets and ran the rating prediction algorithms.

In the following paragraphs, we report on our experiments regarding eight datasets. Four of these datasets are obtained from Amazon [8], [9], three from MovieLens [10], [11], and one from Netflix [12]. These eight datasets used in our experiments (a) contain reliable timestamps (most of the ratings within each dataset have been entered in real rating time and not in a batch mode), (b) are up to date (published between 1998 and 2016), (c) are widely used as benchmarking datasets in CF research and (d) vary with respect to type of dataset (movies, music, videogames and books) and size (from 2MB, up to 4.7GB). The basic properties of these datasets are summarized in Table 1.

In each dataset, users initially having less than 10 ratings were dropped, since users with few ratings are known to exhibit low accuracy in predictions computed for them [3]. This procedure did not affect the three MovieLens and the one NetFlix datasets, because these four datasets contain only users that have rated 20 items or more. Furthermore, we detected cases where for a particular user, all her ratings' timestamps were almost identical (i.e. the difference between the minimum and maximum timestamps was less than 30 seconds). These users were dropped as well, since this timestamp distribution indicated that the ratings were entered in a batch mode, hence the assigned timestamps are not representative of the actual time that these ratings were given by the users.

Table 1. Datasets Summary

Dataset name	**#users**	**#ratings**	**#items**	**Avg. #ratings / user**	**DB Size (in text format)**
Amazon "Videogames" [8], [9]	8.1K	157K	50K	19.6	3.8MB
Amazon "CDs and Vinyl" [8], [9]	41K	1.3M	486K	31.5	32MB
Amazon "Movies and TV" [8], [9]	46K	1.3M	134K	29.0	31MB
Amazon "Books" [8], [9]	295K	8.7M	2.33M	29.4	227MB
MovieLens "Old 100K" dataset [10], [11]	943	100K	1.7K	106.0	2.04MB
MovieLens "Latest-20M, recommended for new research"[10], [11]	138K	20M	27K	145	486MB
MovieLens "Latest 100K, Recommended for education and development" (small) [10], [11]	700	100K	9K	143	2.19MB
NetFlix Competition [12]	480K	96M	17.7K	200	4.7GB

In the following paragraphs, we report on our findings regarding the performance of the algorithm proposed in this work, versus the $DA_{previous}$ and the $DA_{vicinity}$ algorithms reported in [7], and the plain CF algorithm, which uses the standard Pearson correlation coefficient.

4.1 The Amazon "Videogames" Dataset

For the Amazon Videogames dataset, when using the plain CF algorithm, predictions could be formulated for 72.15 % of the cases; in the rest of them, the respective users had no neighbors with a positive Pearson coefficient, i.e. no candidate recommenders, and therefore no prediction could be computed for them.

We can observe that in the $DA_{previous}$ algorithm, which was the winner in the respective experiment presented in [7], this percentage drops by 4.29 %, while, on the other hand, the $DA_{previous}$ algorithm improves the percentage of correct predictions by 0.76 %, while it also reduces the MAE by 7.9 % and the RMSE by 5.9 %.

As far as the proposed algorithm is concerned, all tested variants (a variant corresponds to a particular setting of the *FACTOR* parameter) reduce the MAE from 3.39 % (*FACTOR* = 10) to 11.9 % (*FACTOR* = 1) in comparison to the plain CF algorithm, at the expense of a coverage reduction, which ranges from 1.15 % (*FACTOR* = 10) to 5.87 % (*FACTOR* = 1.0), again in comparison to the plain CF algorithm. Works such as [18], [19] and [43] also assert that a tradeoff between coverage and accuracy exists, and in order to obtain a single measure for rating the suitability of each algorithm, the harmonic mean (HM) of these measures can be adopted; this is analogous to the goal of maximizing the HM of precision and recall – termed the *F1 measure* – in information retrieval [44]). Towards this direction, we adopt the following formula introduced in [43]:

$$\mathrm{HM(a)} = \left(2 * \frac{normCov(a) * normAcc(a)}{normCov(a) + normAcc(a)}\right) \qquad (6)$$

where $a \in Alg$ is a rating prediction algorithm and *Alg* is the set of all algorithms participating in the evaluation. For the $DA_{clusters}$ algorithm, different settings for the *FACTOR* parameter are considered as different algorithms in the context of the evaluation. *normCov(a)* and *normAcc(a)* denote the normalized coverage and normalized accuracy, respectively, of algorithm *a*. These are computed according to the following formulas:

$$normCov(a) = \frac{coverage(a) - \min_{a' \in Alg}(coverage(a'))}{\max_{a' \in Alg}(coverage(a')) - \min_{a' \in Alg}(coverage(a'))} \tag{7}$$

$$normAcc(a) = \frac{\max_{a' \in Alg}(MAE(a')) - MAE(a)}{\max_{a' \in Alg}(MAE(a')) - \min_{a' \in Alg}(MAE(a'))} \tag{8}$$

The results obtained from the Amazon "Videogames" dataset, are depicted in Table 2. For conciseness purposes, only the four best performing variants of the $DA_{clusters}$ algorithm are reported, i.e. the four variants achieving the highest *HM* value. This practice is followed in the presentation of the results for all datasets.

Column *% coverage* corresponds to the percentage of cases for which the algorithm could compute predictions, or – equivalently – when the number of near neighbors computed using the algorithm's similarity metric was adequate [1], [2], to formulate a rating prediction. Columns *MAE* and *RMSE* illustrate the mean absolute error and the root mean square error, respectively, while column *HM (coverage, accuracy)* depicts the harmonic mean measure. Column *Correct predictions %* illustrates the percentage of the cases for which the algorithm achieved to compute the exact rating given by the user. Finally, column *avgs reduction %* shows the reduction in storage space requirements against the dynamic average-based algorithms presented in [7], which is achieved by the $DA_{clusters}$ algorithm, due to the fact that only one dynamic average is computed and stored per cluster, instead of computing one dynamic average per rating; this metric is computed only for the variants of $DA_{clusters}$.

In Table 2 we can observe that the highest harmonic mean is achieved by $DA_{clusters}$*@2.0*, i.e. the variant of $DA_{clusters}$ where the *FACTOR* parameter has been set to 2.0. This variant achieves an overall reduction in the MAE equal to 9.2 % against the plain CF algorithm, surpassing the respective reduction atained by the $DA_{previous}$ algorithm by 1.3 %. The $DA_{clusters}$ algorithm also reduces the RMSE metric by an additional 2.01 % in comparison to the performance of the $DA_{previous}$ algorithm, and at the same time increases both the coverage by 1.52 % and the percentage of correct predictions by 0.36 %.

Furthermore, when applying the proposed technique in this dataset, the dynamic averages stored are reduced by 78 % in comparison to the dynamic averages stored by the algorithms presented in [7], thus introducing significant space gains (recall that in both dynamic average-based algorithms, presented in [7], every rating is coupled with its own particular average).

4.2 The Amazon "CDs and Vinyl" Dataset

The results obtained from the Amazon "CDs and Vinyl" dataset, are depicted in Table 3. Again, when using the plain CF algorithm, predictions could be formulated for 59.3 % of the cases; in the rest of the cases, the respective users had no neighbor with a positive Pearson coefficient, i.e. no candidate recommenders, and therefore no prediction could be computed for them.

In comparison to the plain CF algorithm, the $DA_{previous}$ ratings algorithm, which was the winner in the respective test presented in [7], reduces the MAE by 6.35 %, the RMSE by 5.28 % and increases the percentage of correct predictions by 0.33 %, at the expense of reducing coverage by 4.22 %.

Table 2. Amazon “Videogames” dataset results

Method	**MAE (out of 4)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	0.826	1.142	72.15	0.000	30.38	–
$DA_{previous}$ [7]	0.761	1.074	67.86	0.381	31.14	–
$DA_{vicinity}$ [7]	0.799	1.106	69.96	0.380	30.54	–
$DA_{clusters}$@2.0	**0.750**	**1.051**	**69.38**	**0.626**	**31.50**	**78**
$DA_{clusters}$@2.5	0.760	1.062	69.42	0.594	31.21	80
$DA_{clusters}$@3.5	0.766	1.069	69.64	0.589	31.23	81
$DA_{clusters}$@3.5	0.770	1.074	69.79	0.581	31.43	83

Experiment statistics:	min(MAE) = 0.727 ($DA_{clusters}$@1.0); max(MAE) = 0.826 (plain CF)
	min(coverage) = 66.28 ($DA_{clusters}$@1.0); max(coverage) = 72.15 (plain CF)					

However, $DA_{clusters}$@3.5, which achieves the best HM across all tested algorithms, further reduces the MAE by 0.68 % and the RMSE by 0.28 %, while at the same time increases the coverage by 1.87 % and the percentage of correct predictions by 0.44 % (all differences are reported in comparison with $DA_{previous}$). Space-wise, $DA_{clusters}$@3.5 necessitates the storage of 86 % less dynamic averages than the dynamic average techniques presented in [7].

Table 3. Amazon “CDs and Vinyl” dataset results

Method	**MAE (out of 4)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	0.740	1.060	59.30	0.000	28.07	–
$DA_{previous}$ [7]	0.693	1.004	55.08	0.437	28.40	–
$DA_{vicinity}$ [7]	0.718	1.032	58.66	0.410	28.32	–
$DA_{clusters}$@3.0	0.683	1.001	56.40	0.615	28.69	83
$DA_{clusters}$@3.5	**0.688**	**1.001**	**56.95**	**0.634**	**28.84**	**86**
$DA_{clusters}$@4.0	0.691	1.010	57.15	0.628	28.93	86
$DA_{clusters}$@4.5	0.703	1.013	57.68	0.560	28.06	87

Experiment statistics:	min(MAE) = 0.657 ($DA_{clusters}$@1.0); max(MAE) = 0.74 (plain CF)
	min(coverage) = 52.75 ($DA_{clusters}$@1.0); max(coverage) = 59.30 (plain CF)					

4.3 The Amazon “Movies and TV” Dataset

The results obtained from the Amazon “Movies and TV” dataset, are depicted in Table 4. When using the plain CF algorithm, predictions could be formulated for 78.50 % of the cases. In the $DA_{previous}$ ratings algorithm (which was the winner in the respective test presented in [7]) this percentage drops by 3.62 %, while the percentage of correct predictions increases by 0.88 %, the MAE decreases by 7.54 % and the RMSE drops by 5.80 %.

The proposed technique achieves to further increase prediction quality, while at the same time limits the loss in coverage. More specifically, the *DAclusters@2.0* variant (which achieves the highest HM), exhibits smaller MAE and RMSE in relation to $DA_{previous}$ by 1.38 % and 2.02 %, respectively, while it also increases the percentage of correct predictions by 0.42 % and coverage by 1.35 %. It additionally reduces the number of dynamic averages that must be stored by 80 %.

Table 4. Amazon "Movies and TV" dataset results

Method	**MAE (out of 4)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	0.782	1.103	78.50	0.000	35.45	–
$DA_{previous}$ [7]	0.723	1.039	74.88	0.309	36.33	–
$DA_{vicinity}$ [7]	0.755	1.068	77.26	0.406	35.88	–
$DA_{clusters}$*@1.75*	0.707	1.011	76.09	0.588	36.72	79
$DA_{clusters}$@2.0	**0.713**	**1.018**	**76.23**	**0.591**	**36.75**	**80**
$DA_{clusters}$*@2.5*	0.721	1.028	76.40	0.584	36.76	83
$DA_{clusters}$*@3.0*	0.727	1.036	76.72	0.590	36.79	85

Experiment statistics:	min(MAE) = 0.686 ($DA_{clusters}$*@1.0*); max(MAE) = 0.782 (plain CF)
	min(coverage) = 73.94 ($DA_{clusters}$*@1.0*); max(coverage) = 78.50 (plain CF)					

4.4 The Amazon "Books" Dataset

The results obtained from the Amazon "Books" dataset, which is the largest Amazon dataset, are depicted in Table 5. When using the plain CF algorithm, predictions could be formulated for 53.76 % of the cases. In the $DA_{previous}$ ratings algorithm (which was the winner of the respective experiment reported in [7]), this percentage drops by 1.51 %, while the percentage of correct predictions increases by 0.36 %, the MAE reduces by 2.2 % and the RMSE drops by 1.9 %.

Table 5. Amazon "Books" dataset results

Method	**MAE (out of 4)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	0.631	0.891	53.76	0.000	43.38	–
$DA_{previous}$ [7]	0.617	0.874	52.25	0.273	43.74	–
$DA_{vicinity}$ [7]	0.625	0.884	52.75	0.136	43.68	–
$DA_{clusters}$*@1.5*	0.567	0.810	51.34	0.522	43.89	80
$DA_{clusters}$*@1.75*	0.574	0.819	51.87	0.602	43.88	82
$DA_{clusters}$@2.0	**0.581**	**0.828**	**52.24**	**0.620**	**44.02**	**84**
$DA_{clusters}$*@2.5*	0.59	0.84	52.35	0.570	43.9	87

Experiment statistics:	min(MAE) = 0.551 ($DA_{clusters}$*@1.0*); max(MAE) = 0.631 (plain CF)
	min(coverage) = 49.81 ($DA_{clusters}$*@1.0*); max(coverage) = 53.76 (plain CF)					

In Table 5 we can observe that the variant $DA_{clusters}$@2.0 is the one achieving the highest HM. In comparison to the $DA_{previous}$ ratings algorithm, the $DA_{clusters}$*@2.0* variant reduces the MAE and the RMSE by 5.71 % and 5.16 %, respectively and increases the percentage of correct predictions by 0.28 %, while practically leaving coverage unaffected. Finally, it necessitates the storage of 84 % less dynamic averages in comparison to both algorithms introduced in [7] ($DA_{previous}$ and $DA_{vicinity}$).

4.5 The MovieLens "Old 100K" Dataset

The results obtained from the MovieLens "Old 100K" dataset are depicted in Table 6. When using the plain CF algorithm, predictions could be formulated for 99.82 % of the cases (due to the dataset's high density). In the $DA_{previous}$ ratings algorithm (which was ranked first in the respective experiment presented in [7]), the coverage is practically not affected, while the percentage of correct predictions increases by 1.19 %, the MAE and the RMSE fall by 3.47 % and 3.03 %, respectively.

In table 6 we can observe that the variant *$DA_{clusters}$@1.75* is the one achieving the highest HM. In comparison to *$DA_{previous}$*, the *$DA_{clusters}$@1.75* variant exhibits lower MAE and RMSE by 14.40 % and 13.36 %, respectively, while it increases the percentage of correct predictions by 7.44 % and maintains the same high coverage. Space-wise, *$DA_{clusters}$@1.75* variant necessitates the storage of 76 % less dynamic averages than the dynamic average techniques presented in [7].

Table 6. MovieLens “Old 100K” dataset results

Method	**MAE (out of 4)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	0.750	0.958	99.82	0.000	41.43	–
$DA_{previous}$ [7]	0.724	0.929	99.83	0.285	42.62	–
$DA_{vicinity}$ [7]	0.729	0.934	99.83	0.238	42.64	–
$DA_{clusters}$@1.5	0.612	0.807	99.83	0.872	50.45	75
$DA_{clusters}$@1.75	**0.616**	**0.809**	**99.84**	**0.940**	**50.06**	**76**
$DA_{clusters}$@2.0	0.623	0.818	99.84	0.914	49.22	78
$DA_{clusters}$@2.5	0.641	0.839	99.83	0.774	48.53	81
Experiment statistics:	min(MAE) = 0.599 (*$DA_{clusters}$@1.0*); max(MAE) = 0.750 (plain CF) min(coverage) = 99.78 (*$DA_{clusters}$@5.0*); max(coverage) = 99.84 ($DA_{clusters}$@1.75)					

4.6 The MovieLens “Latest-20M, Recommended for New Research” Dataset

The results obtained from the MovieLens “Latest-20M, recommended for new research” dataset, are depicted in Table 7. Under the plain CF algorithm, predictions could be formulated for 99.96 % of the cases (again, due to the dataset’s high density). In the *$DA_{previous}$* ratings algorithm this percentage remains practically unaffected, dropping by 0.04 %, while the percentage of correct predictions increases by 1.48 %, the MAE reduces by 3.26 % and the RMSE declines by 3.05 %.

In Table 7 we can observe that the variant *$DA_{clusters}$@1.75* is the one scoring the highest HM. In comparison to the *$DA_{previous}$* variant, the *$DA_{clusters}$@1.75* variant reduces the MAE and the RMSE by 11.31 % and 9.71 % respectively, and increases the percentage of correct predictions by 4.92 % with no change in coverage. Additionally, the *$DA_{clusters}$@1.75* reduces the requirements for storage of dynamic averages by 79 %.

Table 7. MovieLens “Latest-20M, recommended for new research” dataset results

Method	**MAE (out of 9)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	1.379	1.802	99.96	0.000	23.68	–
$DA_{previous}$ [7]	1.334	1.747	99.92	0.245	25.16	–
$DA_{vicinity}$ [7]	1.352	1.763	99.90	0.000	24.42	–
$DA_{clusters}$@1.75	**1.178**	**1.576**	**99.92**	**0.481**	**30.08**	**79**
$DA_{clusters}$@2.0	1.197	1.596	99.92	0.467	29.68	82
$DA_{clusters}$@2.5	1.212	1.614	99.92	0.455	29.38	84
$DA_{clusters}$@3.0	1.230	1.633	99.92	0.438	28.50	86
Experiment statistics:	min(MAE) = 1.146 (*$DA_{clusters}$@1.0*); max(MAE) = 1.379 (plain CF) min(coverage) = 99.9 (*$DA_{vicinity}$*); max(coverage) = 99.96 (plain CF)					

4.7 The MovieLens "Latest 100K, Recommended for Education and Development (small)" Dataset

The results obtained from the MovieLens "Latest 100K, Recommended for education and development (small)" dataset, are depicted in Table 8. When using the plain CF algorithm, predictions could be formulated for 99.57 % of the cases (again, due to the dataset's high density). In the $DA_{previous}$ ratings algorithm (which was the winner in the respective experiment reported in [7]) this percentage drops by 0.21 %, while the percentage of correct predictions increases by 0.12 %, the MAE reduces by 3.36 % and the RMSE drops by 3.75 %.

Table 8. MovieLens "Latest 100K, Recommended for education and development (small)" dataset results

Method	**MAE (out of 9)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	1.430	1.893	99.57	0.000	23.90	–
$DA_{previous}$ [7]	1.382	1.822	99.36	0.256	24.02	–
$DA_{vicinity}$ [7]	1.402	1.848	99.60	0.187	23.99	–
$DA_{clusters}@1.25$	1.179	1.573	99.32	0.536	29.39	76
$DA_{clusters}@1.5$	**1.194**	**1.599**	**99.34**	**0.568**	**29.40**	**78**
$DA_{clusters}@1.75$	1.203	1.61	99.31	0.499	29.36	79
$DA_{clusters}@2.0$	1.222	1.631	99.31	0.485	28.57	81
Experiment statistics:	min(MAE) = 1.158 ($DA_{clusters}@1.0$); max(MAE) = 1.43 (plain CF) min(coverage) = 99.15 ($DA_{clusters}@2.5$); max(coverage) = 99.6 ($DA_{vicinity}$)					

As shown in Table 8, the $DA_{clusters}@1.5$ variant is the one exhibiting the highest HM. As compared to the $DA_{previous}$ algorithm, the $DA_{clusters}@1.5$ variant reduces the MAE and the RMSE by 13.15 % and 11.78 %, respectively, while it increases the correct prediction percentage by 5.38 % and achieves practically the same coverage. In terms of dynamic averages storage, the $DA_{clusters}@1.5$ variant reduces the respective requirements by 78 %, in comparison to the dynamic average-based algorithms presented in [7].

4.8 The "NetFlix Competition" Dataset

The results obtained from the "NetFlix Competition" dataset, are depicted in Table 9. Again, using the plain CF algorithm, predictions could be formulated for 99.12 % of the cases. In the respective experiments reported in [7], the $DA_{vicinity}$ algorithm was the winner, and our measurements verify that it surpasses the $DA_{previous}$ algorithm in this dataset. In comparison to the plain CF algorithm, the $DA_{vicinity}$ algorithm reduces the MAE and the RMSE by 4.87 % and 5.77 %, respectively, and achieves an increase of the correct predictions percentage by 4.18 %, at the expense of reducing coverage by 0.29 %.

Table 9 indicates that the variant $DA_{clusters}@1.75$ exhibits the highest HM. In comparison to the $DA_{vicinity}$ algorithm, the $DA_{clusters}@1.75$ variant decreases the MAE and the RMSE by 5.0 % and 2.43 % respectively, while it also increases the percentage of correct predictions by 1.38 % and coverage by 0.14 %. Furthermore, the $DA_{clusters}@1.75$ variant reduces the number of dynamic averages that must be stored by 90 %, in comparison to the dynamic average-based algorithms presented in [7].

Table 9. "NetFlix Competition" dataset results

Method	**MAE (out of 4)**	**RMSE**	**Coverage, %**	**HM (coverage, accuracy)**	**Correct predictions, %**	**avgs reduction, %**
Plain CF	0.780	0.988	99.12	0.000	40.18	–
$DA_{previous}$ [7]	0.750	0.966	98.81	0.291	45.21	–
$DA_{vicinity}$ [7]	0.742	0.931	98.83	0.357	44.36	–
$DA_{clusters}@1.5$	0.701	0.905	98.91	0.645	45.82	90
$DA_{clusters}@1.75$	**0.703**	**0.907**	**98.97**	**0.753**	**45.74**	**90**
$DA_{clusters}@2.0$	0.709	0.909	98.95	0.695	44.22	91
$DA_{clusters}@2.5$	0.713	0.918	98.94	0.662	43.91	91

Experiment statistics:	min(MAE) = 0.697 ($DA_{clusters}@1.0$); max(MAE) = 0.780 (plain CF)
	min(coverage) = 98.71 ($DA_{clusters}@2.5$); max(coverage) = 99.6 ($DA_{vicinity}$)					

4.9 Algorithm Comparison

Figure 1 depicts the improvement in MAE achieved by the dynamic average-based algorithms; for each dataset, the performance of the $DA_{previous}$ and $DA_{vicinity}$ algorithms is presented, together with the performance of the $DA_{clusters}$ variant that achieved the highest HM in the particular dataset. In all cases, the performance of the plain CF algorithm is used as a baseline.

The graph shows that the $DA_{clusters}$ algorithm introduced in this article achieves a mean improvement of 11.5 % against the plain CF algorithm across all datasets, ranging from 7.0 % to 17.9 %. This surpasses the performance of the $DA_{previous}$ and $DA_{vicinity}$ algorithms, for which the improvements in the average MAE reduction across all datasets are 4.7 % and 2.8 %, respectively. It is worth noting that the $DA_{clusters}$ algorithm is consistently ranked first in all eight datasets.

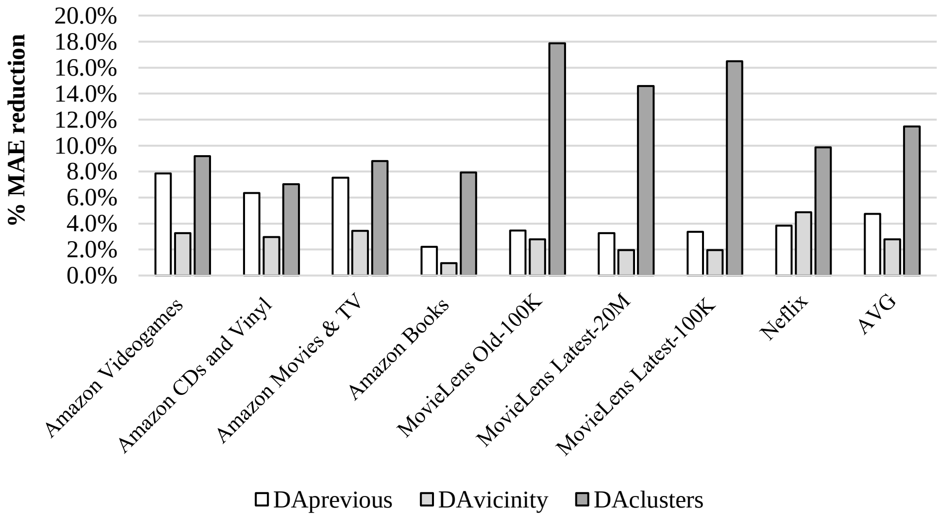

Figure 1. MAE improvement achieved by the $DA_{clusters}$ and the algorithms proposed in [7]

Figure 2 presents the respective improvements regarding the RMSE metric. In all cases, the $DA_{clusters}$ algorithm, presented in this article, is ranked first, scoring an improvement of 10.0 % on average, while the $DA_{previous}$ and $DA_{vicinity}$ algorithms proposed in [7] achieve corresponding improvements of 3.9 % and 2.8 %, respectively. Moreover, the improvements are very similar to those of the MAE metric shown in Figure 1, indicating that prediction improvements are spread uniformly among predictions with high and low errors (recall that the RMSE metric strongly penalizes predictions with high errors).

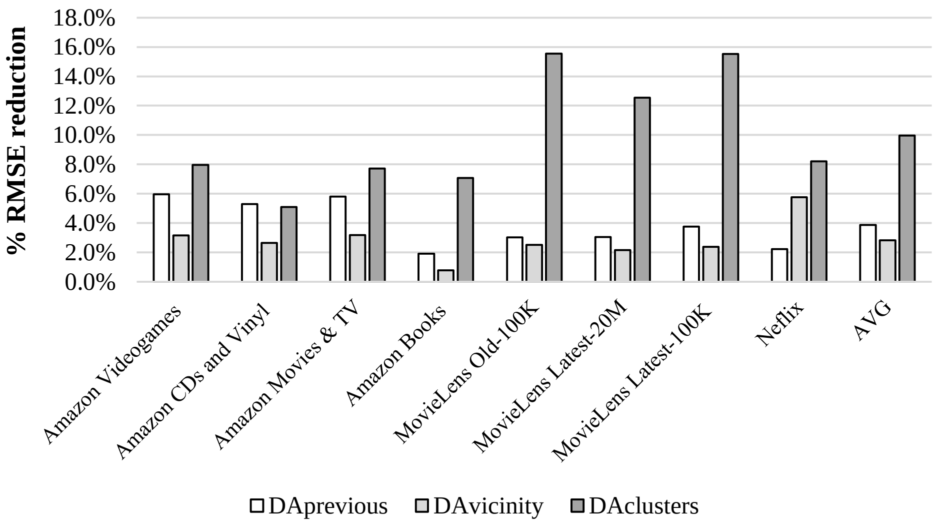

Figure 2. RMSE improvement achieved by the *DA*$_{clusters}$ and the algorithms proposed in [7]

Figure 3 illustrates the improvements regarding the correct prediction percentage. The *DA*$_{clusters}$ algorithm is ranked first across all datasets, achieving improvements ranging from 0.6 % to 8.6 % against the baseline algorithm (which is the plain CF algorithm), scoring an average improvement of 3.7 %. We can observe that in the denser datasets (all MovieLens and the Netflix dataset), the improvements in correct prediction percentage achieved by the *DA*$_{clusters}$ algorithm are more substantial. In comparison to the algorithms proposed in [7], the *DA*$_{clusters}$ algorithm outperforms both the *DA*$_{previous}$ and the *DA*$_{vicinity}$ algorithms, which achieve an average improvement equal to 1.3 % and 0.9 %, respectively.

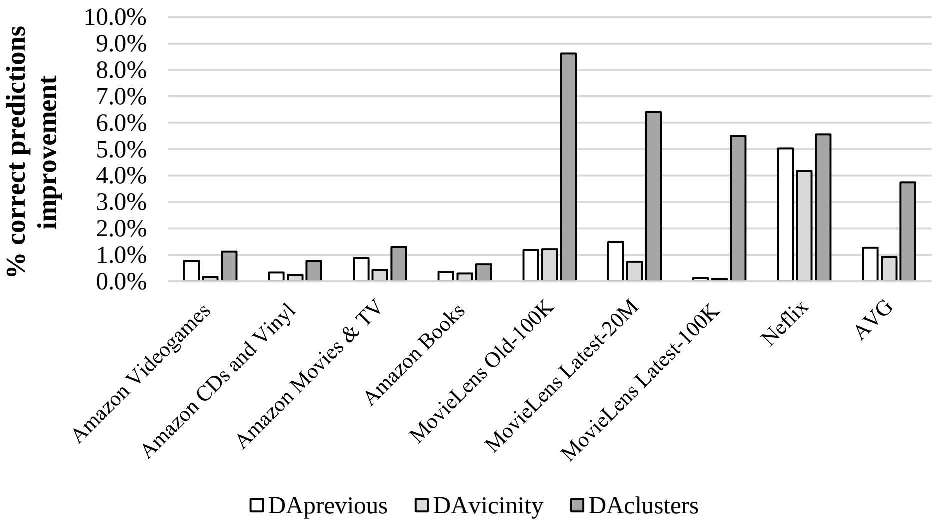

Figure 3. Correct predictions percentage improvement achieved by the *DA*$_{clusters}$ and the algorithms proposed in [7]

Figure 4 illustrates the reduction in coverage sustained when using the dynamic average-based algorithms, against the baseline algorithm (plain CF). In this case, the *DA*$_{clusters}$ algorithm is ranked second regarding its average performance, losing 1.2 % of coverage on average, with its coverage loss varying from -0.02 % (i.e. it achieves a marginal coverage increase in one dataset, namely the “MovieLens Old-100K” dataset) to 2.77 %. The winner algorithm regarding this metric is *DA*$_{vicinity}$, for which the average coverage drop is 0.7 %, while the *DA*$_{previous}$ algorithm is ranked third, with an average coverage reduction of 1.8 %. In Figure 4 we can observe that coverage drop is considerably higher in the Amazon datasets, which are sparser, while in the denser datasets (all MovieLens datasets and the Netflix dataset), coverage loss is negligible (less than 0.29 % in all cases).

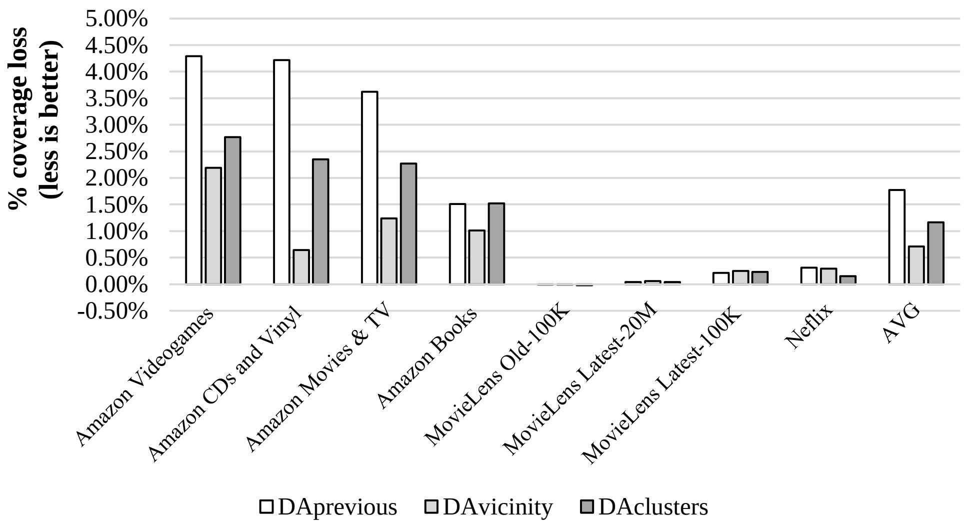

Figure 4. Predictions (cases) lost when applying the $DA_{clusters}$ algorithm and the algorithms proposed in [7] (less is better)

Finally, Figure 5 illustrates the reduction in storage needs for dynamic averages that is achieved by the $DA_{clusters}$ algorithm, as compared to the dynamic average-based algorithms proposed in [7]; since both of these algorithms require the same number of dynamic averages, only one set of measurements is presented in Figure 5. For each dataset, we consider the variant of $DA_{clusters}$ that achieves the highest HM. We can observe that the $DA_{clusters}$ algorithm substantially reduces the need for storing dynamic averages, with the respective gains ranging from 76 % to 90 %, having an average of 81 % across all datasets.

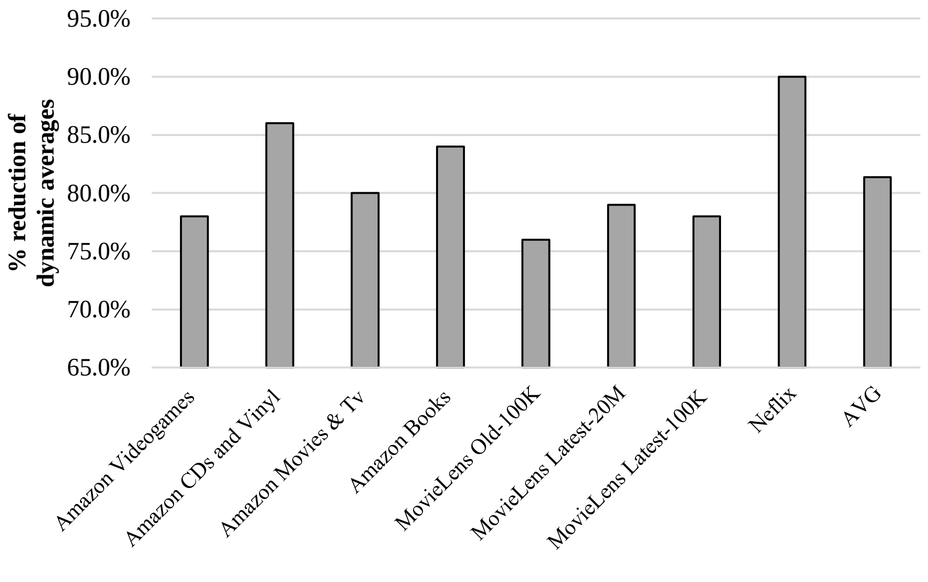

Figure 5. Storage needs reductions achieved by the $DA_{clusters}$ algorithm against the algorithms proposed in [7]

Summarizing, we can see that in all datasets the $DA_{clusters}$ algorithm is ranked first regarding the MAE reduction, the RMSE reduction and the correct predictions' percentage improvement. The only metric for which the $DA_{clusters}$ algorithm is ranked second is coverage, where it lags behind the $DA_{vicinity}$ algorithm; however, the $DA_{clusters}$ algorithm performs substantially better than $DA_{vicinity}$ regarding the MAE reduction, therefore the $DA_{clusters}$ algorithm is clearly ranked first when jointly examining the coverage and accuracy metric, as shown in the discussions presented in Sections 4.1-4.8. Finally, the $DA_{clusters}$ algorithm introduces significant savings in space requirements for storing the dynamic averages, necessitating the storage of 81 % less dynamic averages than the algorithms presented in [7].

4.10 Discussion

From the experiments above, it is clear that the $DA_{clusters}$ algorithm leads to more accurate predictions than the algorithms presented in [7], namely $DA_{vicinity}$ and $DA_{previous}$. This is owing to the fact that the $DA_{clusters}$ algorithm computes local averages taking into account only ratings that are temporally "close" together, taking into account the expected user rating frequency through the examination of rating abstention periods. On the contrary, when computing the dynamic average for some rating *r*, the $DA_{previous}$ algorithm takes into account all previous ratings, regardless of whether these are temporally close or distant to *r*. Consequently, the value of the dynamic average for *r* is affected by ratings that are temporally distant (where "distant" should be interpreted in relation to the nominal user rating frequency) and are thus highly likely to belong to periods during which the user used to follow different rating practices. Conversely, ratings that have been entered shortly after *r* (where "shortly" should be again interpreted in relation to the nominal user rating frequency), which should be grouped in the same rating practice period with *r* and be taken into account in the computation of the dynamic average of *r*, are disregarded. Notably, in dense datasets, where user rating frequencies are higher, and more clusters are formulated per user, the gains in accuracy are considerably higher.

Considering the $DA_{vicinity}$ algorithm, when the dynamic average for some rating *r* is computed, all ratings are taken into account, albeit smaller weights are used for temporally distant ones. However, for periods where the user's rating frequency is low, this practice reduces the weight of ratings that should be grouped in the same rating practice period with *r*, introducing a source of inaccuracy in the dynamic average computation. Additionally, ratings that are temporally very distant from *r* are included in the computation of the dynamic average related to *r*, and these ratings effectively add noise to the computation of the dynamic average.

5 Conclusion and Future Work

In this article we have introduced the concept of *rating time clusters*, i.e. groups of ratings with temporal cohesion, and for which the user is assumed to follow the same rating practice. Furthermore, we have presented $DA_{clusters}$, a novel algorithm which computes rating time clusters and one dynamic average per cluster for each cluster; the computed dynamic averages are then utilized in the rating prediction process. The proposed algorithm has been experimentally verified using 8 datasets and compared to the algorithms proposed in [7] ($DA_{vicinity}$ and $DA_{previous}$), and has been found to consistently outperform both of them, in terms of prediction accuracy and overall performance, i.e. their performance taking into account the accuracy and coverage metrics. Improvement in rating prediction accuracy indicates that the $DA_{clusters}$ algorithm is able to follow more closely shifts in rating prediction practices. In particular, the average MAE reduction compared to the $DA_{previous}$ and the $DA_{vicinity}$ algorithms is 6.7 % and 8.7 % respectively, whereas the corresponding improvements regarding the RMSE metric are 6.1 % and 7.1 %. Considering the correct prediction metric, the proposed algorithm outperforms the $DA_{previous}$ and $DA_{vicinity}$ algorithms by 2.5 % and 2.8 %, respectively. Finally, the percentage of the dynamic averages stored is reduced by 81 %, compared to the dynamic average-based algorithms presented in [7].

Our future work will focus on adapting the proposed approach for use with matrix factorization techniques (an alternative method to CF, used in recommender systems, based on matrix decomposition) [45], as well as comparing it with time-aware matrix factorization models

[46], [47]. We also plan to explore methods for further decreasing the space overhead for the implementation of dynamic averages and, finally, using more elaborate clustering techniques, such as those described in [40] and [41], for identifying periods in which each user's rating practices remain stable.

References

[1] J.B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative Filtering Recommender Systems," *The Adaptive Web, Lecture Notes in Computer Science*, vol. 4321, pp. 291–324, 2007. Available: https://doi.org/10.1007/978-3-540-72079-9_9

[2] M. Balabanovic and Y. Shoham, "Fab: content-based, collaborative recommendation," *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997. Available: https://doi.org/10.1145/245108.245124

[3] M.D. Ekstrand, J.T. Riedl, and J.A. Konstan, "Collaborative Filtering Recommender Systems," *Foundations and Trends in Human-Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011. Available: https://doi.org/10.1561/1100000009

[4] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.P. Kriegel, "Probabilistic Memory-Based Collaborative Filtering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 1, pp. 56–69, 2004. Available: https://doi.org/10.1109/TKDE.2004.1264822

[5] P. Winoto and T.Y. Tang, "The role of user mood in movie recommendations," *Expert Systems with Applications*, vol. 37, pp. 6086–6092, 2010, Available: https://doi.org/10.1016/j.eswa.2010.02.117

[6] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A Survey on Concept Drift Adaptation," *ACM Computing Surveys*, vol. 1, no. 1, Article 1, 37 pages, 2013. Available: https://doi.org/10.1145/2523813

[7] D. Margaris and C. Vassilakis, "Improving Collaborative Filtering's Rating Prediction Quality by Considering Shifts in Rating Practices," in *Proc. the 19th IEEE International Conference on Business Informatics*, CBI '17 Thessaloniki, Greece, July 24–27, vol. 01, pp. 158–166, 2017. Available: https://doi.org/10.1109/CBI.2017.24

[8] J.J. McAuley, R. Pandey, and J. Leskovec, "Inferring Networks of Substitutable and Complementary Products," in *Proc. the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* KDD '15, Sydney, Australia, August 10–13, pp. 785–794, 2015. Available: https://doi.org/10.1145/2783258.2783381

[9] J. McAuley, C. Targett, J. Shi, and A. van den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. the 38th international ACM SIGIR conference*, SIGIR 2015, Santiago, Chile, August 09–13, pp. 43–52, 2015. Available: https://doi.org/ 10.1145/2766462.2767755

[10] MovieLens, "MovieLens datasets", 2017. Available: https://grouplens.org/datasets/movielens/ (accessed 18.09.2017)

[11] F. Maxwell Harper and J.A. Konstan, "The MovieLens Datasets: History and Context," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 4, Article No. 19, 19 pages, 2016. Available: https://doi.org/10.1145/2827872

[12] Y. Zhou, D. Wilkinson, R. Schreiber, and R. Pan, "Large-Scale Parallel Collaborative Filtering for the Netflix Prize," in *Proc. the 4th international conference on Algorithmic Aspects in Information and Management* AAIM '08, Shanghai, China, June 23–25, pp. 337–348, 2008. Available: https://doi.org/10.1007/978-3-540-68880-8_32

[13] E. Bakshy, I. Rosenn, C. Marlow, and L. Adamic, "The role of social networks in information diffusion," in *Proc. the 21st international conference on World Wide Web*, WWW '12, Lyon, France, April 16–20, pp. 519–528, 2012. Available: https://doi.org/10.1145/2187836.2187907

[14] D. Margaris, C. Vassilakis, and P. Georgiadis, "Recommendation Information Diffusion in Social Networks Considering User Influence and Semantics," *Social Network Analysis and Mining*, vol. 6, no. 1, pp. 1–22, 2016. Available: https://doi.org/10.1007/s13278-016-0416-z

[15] D. Margaris, C. Vassilakis, and P. Georgiadis, "Knowledge-Based Leisure Time Recommendations in Social Networks," *Current Trends on Knowledge-Based Systems: Theory and Applications*, Alor-Hernández G., Valencia-García R. (Eds), pp. 23–48, 2017. Available: https://doi.org/10.1007/978-3-319-51905-0_2

[16] J. Bao, Y. Zheng, and M. Mokbel, "Location-based and Preference-Aware Recommendation Using Sparse Geo-Social Networking Data," in Proc. the 20th International Conferences on Advances in Geographic

Information Systems, SIGSPATIAL ’12, Redondo Beach, California, November 06–09, pp. 199–208, 2012. Available: https://doi.org/10.1145/2424321.2424348

[17] Y. Zheng and X. Xing, “Learning travel recommendations from user-generated GPS traces,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 1, 2011. Available: https://doi.org/10.1145/1889681.1889683

[18] D. Margaris and C. Vassilakis, “Pruning and aging for user histories in collaborative filtering,” in *Proc. the 2016 IEEE Symposium Series on Computational Intelligence*, SSCI 2016, Athens, Greece, Dec. 6–9, pp 1–8, 2016. Available: https://doi.org/10.1109/SSCI.2016.7849920

[19] D. Margaris and C. Vassilakis, “Enhancing User Rating Database Consistency through Pruning,” *Special issue on Consistency and Inconsistency in Data-centric Applications, Transactions on Large-Scale Data and Knowledge-Centered Systems*, Springer, vol. XXXIV, pp. 33–64, 2017. Available: https://doi.org/10.1007/978-3-662-55947-5_3

[20] S. Gong, “A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering,” *Journal of Software*, vol. 5, no. 7, pp. 745–752, 2010. Available: https://doi.org/10.4304/jsw.5.7.745-752

[21] A. Das, M. Datar, A. Garg, and S. Rajaram, “Google News Personalization: Scalable Online Collaborative Filtering,” in *Proc. the 16th international conference on World Wide Web*, WWW ‘2007, Banff, Alberta, Canada, May 08–12, pp. 271–280, 2007. Available: https://doi.org/10.1145/1242572.1242610

[22] M.K. Najafabadi, M.N. Mahrin, S. Chuprat, and H.M. Sarkan, “Improving the accuracy of collaborative filtering recommendations using clustering and association rules mining on implicit data,” *Computers in Human Behavior*, vol. 67, no. C, pp. 113–128, 2017. Available: https://doi.org/10.1016/j.chb.2016.11.010

[23] Y. Koren, “Factor in the neighbors: Scalable and accurate collaborative filtering,” *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 4, no. 1, Article 1, 24 pages, 2010. Available: https://doi.org/10.1145/1644873.1644874

[24] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, “A new user similarity model to improve the accuracy of collaborative filtering,” *Knowledge-Based Systems*, vol. 56, pp. 156–166, 2014, Available: https://doi.org/ 10.1016/j.knosys.2013.11.006

[25] M. Ramezani, P. Moradi, and F. Akhlaghian, “A pattern mining approach to enhance the accuracy of collaborative filtering in sparse data domains,” Physica A: Statistical Mechanics and its Applications, vol. 408, pp. 72–84, 2014. Available: https://doi.org/10.1016/j.physa.2014.04.002

[26] L. Li, L. Zheng, F. Yang, and T. Li, “Modeling and broadening temporal user interest in personalized news recommendation,” *Expert Systems with Applications*, vol. 41, no. 7, pp. 3168–3177, 2014. Available: https://doi.org/10.1016/j.eswa.2013.11.020

[27] I. Zliobaite, J.Bakker, and M. Pechenizkiy, “Beating the baseline prediction in food sales: How intelligent an intelligent predictor is?” *Expert Systems with Applications*, vol. 39, no. 1, pp. 806–815, 2012. Available: https://doi.org/10.1016/j.eswa.2011.07.078

[28] H.H. Ang, V. Gopalkrishnan, I. Zliobaite, M. Pechenizkiy, and S. C. H. Hoi, “Predictive Handling of Asynchronous Concept Drifts in Distributed Environments,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2343–2355, 2013. Available: https://doi.org/10.1109/TKDE.2012.172

[29] R. Elwell and R. Polikar, “Incremental Learning of Concept Drift in Nonstationary Environments,” *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011. Available: https://doi.org/10.1109/TNN.2011.2160459

[30] L.L. Minku, A.P. White, and X. Yao, “The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2010. Available: https://doi.org/ 10.1109/TKDE.2009.156

[31] K. Nishida and K. Yamauchi, “Detecting concept drift using statistical testing,” in *Proc. the 10th international conference on Discovery science*, DS '07, Sendai, Japan, October 01–04, pp. 264–269, 2007. Available: https://doi.org/10.1007/978-3-540-75488-6_27

[32] P.C. Vaz, R. Ribeiro, and D.M. de Matos, “Understanding temporal dynamics of ratings in the book recommendation scenario,” in *Proc. the 2013 ACM International Conference on Information Systems and Design of Communication*, ISDOC ‘13, Lisboa, Portugal, July 11–12, pp. 11–15, 2013. Available: https://doi.org/10.1145/2503859.2503862

[33] N. Koenigstein, G. Dror, and Y. Koren, “Yahoo! Music recommendations: modeling music ratings with temporal dynamics and item taxonomy,” in *Proc. the 5th ACM conference on Recommender systems*, RecSys '11, New York, NY, USA, pp. 165–172, 2011. Available: https://doi.org/10.1145/2043932.2043964

[34] N.N. Liu, L. He, and M. Zhao, “Social temporal collaborative ranking for context aware movie recommendation,” *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 4, no. 1, Article No. 15, 26 pages, 2013. Available: https://doi.org/10.1145/2414425.2414440

[35] R. Dias and M. J. Fonseca, “Improving Music Recommendation in Session-Based Collaborative Filtering by Using Temporal Context,” in *Proc. IEEE 25th International Conference on Tools with Artificial Intelligence*, Herndon, Virginia, USA, Nov 4–6, pp. 783–788, 2013. Available: https://doi.org/10.1109/ICTAI.2013.120

[36] C.T. Li, M.K. Shan, S.H. Jheng, and K.C. Chou, “Exploiting concept drift to predict popularity of social multimedia in microblogs,” *Information Sciences*, vol. 339, pp. 310–331, 2016, Available: https://doi.org/10.1016/j.ins.2016.01.009

[37] Z. Lu, S.J. Pan, Y. Li, J. Jiang, and Q. Yang, “Collaborative evolution for user profiling in recommender systems,” in *Proc. the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI '16, New York, USA, July 09–15, pp. 3804–3810, 2016. Available: https://doi.org/1073-0516/01/0300-0034

[38] A. Kangasrääsiö, Y. Chen, D. Głowacka, and S. Kaski, “Interactive Modeling of Concept Drift and Errors in Relevance Feedback,” in *Proc. the 2016 ACM Conference on User Modeling Adaptation and Personalization*, UMAP '16, Halifax, Nova Scotia, Canada, July 13–17, pp. 185–193, 2016. Available: https://doi.org/10.1145/2930238.2930243

[39] D. He and D. Wu, “Toward a robust data fusion for document retrieval,” in *Proc. of the IEEE 4th International Conference on Natural Language Processing and Knowledge Engineering*, NLP-KE 2008, Beijing, China, October 19–22, 2008. Available: https://doi.org/10.1109/NLPKE.2008.4906754

[40] L. Kaufman and P.J. Rousseeuw, *Finding Groups in Data: an Introduction to Cluster Analysis*, John Wiley & Sons, 2008, ISBN: 0471735787. Available: https://doi.org/10.1002/9780470316801

[41] D. Margaris, P. Georgiadis, and C. Vassilakis, “A Collaborative Filtering Algorithm with Clustering for Personalized Web Service Selection in Business Processes,” in *Proc. The 9th IEEE international conference on Research Challenges in Information Science*, RCIS 2015, Athens, Greece, May 13–15, pp. 169–180, 2015. Available: https://doi.org/10.1109/RCIS.2015.7128877

[42] R. Burke, “Hybrid recommender systems: Survey and experiments,” *User Modeling and User-Adapted Interaction*, vol. 12, no. 4, pp. 331–370, 2002. Available: https://doi.org/10.1023/A:1021240730564

[43] D. Margaris and C. Vassilakis, “Improving Collaborative Filtering’s Rating Prediction Quality in Dense Datasets, by Pruning Old Ratings,” in *Proc. the 2017 IEEE Symposium on Computers and Communications*, ISCC 2017, pp. 1168–1174, Heraklion, Greece, July 3–6, pp. 158–166, 2017. Available: https://doi.org/10.1109/ISCC.2017.8024683

[44] Z.C. Lipton, C.Elkan, and B. Naryanaswamy, “Optimal Thresholding of Classifiers to Maximize F1 Measure,”. in *Proc. the the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases*, ECMLPKDD'14, Nancy, France, September 15–19, vol. 2, pp. 225–239, 2014. Available: https://doi.org/10.1007/978-3-662-44851-9_15

[45] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” *IEEE Computer*, vol. 42, no. 8, pp. 30–37, 2009. Available: https://doi.org/10.1109/MC.2009.263

[46] Z. Gantner, S. Rendle, and L. Schmidt-Thieme, “Factorization models for context-/time-aware movie recommendations,” in *Proc. the ACM Workshop on Context-Aware Movie Recommendation*, CAMRa '10, Barcelona, Spain, September 30, pp. 14–19, 2010. Available: https://doi.org/10.1145/1869652.1869654

[47] J.D. Zhang and C.Y. Chow, “TICRec: A probabilistic framework to utilize temporal influence correlations for time-aware location recommendations,” *IEEE Transactions on Services Computing*, vol. 9, no. 4, pp. 633–646, 2016. Available: https://doi.org/10.1109/TSC.2015.2413783