Pruning and Aging for User Histories in Collaborative Filtering

Dionisis Margaris
Department of Informatics and Telecommunications
University of Athens
Athens, Greece
margaris@di.uoa.gr

Costas Vassilakis
Department of Informatics and Telecommunications
University of the Peloponnese
Tripoli, Greece
costas@uop.gr

***Abstract*—In this paper, we introduce algorithms for pruning and aging user ratings in collaborative filtering systems, based on their oldness, under the rationale that aged user ratings may not accurately reflect the current state of users regarding their preferences. The aging algorithm reduces the importance of aged ratings, while the pruning algorithm removes them from the database. The algorithms are evaluated against various types of datasets. The pruning algorithm has been found to present a number of advantages, namely (1) reducingthe rating database size, (2) achievingbetter predictiongeneration times and (3) improving prediction quality by cutting off predictions with high error. The algorithm can be used in all rating databases that include a timestamp and has been proved to be effective in any type of dataset, from movies and music, to videogames and books.**

Keywords—Pruning; Aging;Collaborative filtering; User rating databases; Evaluation; Performance

I. INTRODUCTION

Collaborative filtering (CF) formulates personalized recommendations on the basis of ratings expressed by people having similar tastes to the user for whom the recommendation is generated; taste similarity is computed by examining the resemblance of already entered ratings [1]. CF works on the assumption that if users have similar taste regarding choosing or liking an item in the past then are likely to have similar interest in the future too. Typically, for each user a set of "nearest neighbor" users is found, i.e. those users that display the strongest correlation to the target user. Scores for unseen items are predicted based on a combination of the scores known from the nearest neighbors [14].

Research has proven that the CF-based recommendation approach is the most successful and widely used for implementing recommender systems [2]. When CF is employed, typical recommender systems assume that time is not relevant and ignore how old each user-item rating is. However, rating age can be exploited to substantially enhance recommendation quality, due to phenomena like shift of interest [3][4]. For instance, in domains like music, users continuously listen to and rate songs. In this setting, user neighborhoods based on song ratings rapidly lose their validity; indeed the time span in which a user-based neighborhood remains valid is too short because new users –who are potential neighbor candidates- continuously join the system, and new user preferences are constantly added to the user rating database [11]. This necessitates the re-computation of the user neighborhood, which may be an excessively costly action if all rating data are retained and matched in the neighborhood re-computation process.

Another major issue that collaborative filtering systems front is the storing and management of rating data (user ratings). On one hand, maintaining all ratings in an extensive database offers the advantage of reducing the "grey sheep" probability [15], i.e. the probability that some user's recorded preferences are unusual, as compared to the rest of the community, which leads to poor recommendations. However, maintaining and processing the whole data bulk increases storage demands and the time needed to generate recommendations (since computing the neighborhoods for all data and all users is costly).

In this paper, we introduce algorithms for pruning and aging user ratings, based on their oldness. The proposed algorithms exploit user rating timestamps in the recommendation process; user rating timestamps exist in many datasets, such as Amazon [5][6] and Movielens[7][23]. The first of the algorithms decreases the significance of old-aged ratings, while the second one prunes user rating histories based on these timestamps; in this sense, the second algorithm effectively applies the p-core pruning strategy [24][25], exploiting the age of ratings to select the ratings to be pruned. The proposed algorithms are evaluated in terms of (a) prediction accuracy, (b) storage size gains and (c) execution performance. The results show that the pruning-based algorithm achieves very satisfactory results in the Amazon [5][6] and Movielens [7][23] datasets, leading to reduced prediction formulation times and, in parallel, to increased accuracy due to the removal of aged ratings, which have been found to contribute to the computation of predictions with high absolute errors; considerable database size gains are also achieved. On the contrary, aging of ratings in some cases marginally improves prediction quality, and in some others degrades it, without introducing any savings in prediction formulation time or database size gains. Additionally, the suggested pruning technique can be applied as a preprocessing step to any CF-based algorithm, including algorithms that consider social network data (e.g. [13][19]).

The rest of the paper is structured as follows: section 2 overviews related work, while section 3 presents the proposed algorithms. Section 4 evaluates the proposed algorithm and finally, section 5 concludes the paper and outlines future work.

II. Related Work

The accuracy of CF systems is a topic that has attracted considerable research efforts. [8] 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. [9] proposes a filtering technique that applies to both unfairly positive and unfairly negative ratings in Bayesian reputation system. It is based on a reputation system and integrates a reputation systems filtering method, under the assumption that ratings provided by different raters on a given agent will follow more or less the same probability distribution. [10] presents two matrix completion methods named *Kmf* and *Mkmf*, which can both exploit the underlying nonlinear correlations among rows (users) and columns (items) of the rating matrix simultaneously. *Kmf* incorporates kernel methods for matrix factorization, which embeds the low-rank feature matrices into a much higher dimensional space, enabling the ability to learn nonlinear correlations upon the rating data in original space. *Mkmf* further extends *Kmf* to combine multiple kernels by learning the set of weights for each kernel function based on the observed data in rating matrix. [3] and [4] explore the issue of temporal dynamics in user preferences, constructing time-sensitive user models and profiles respectively, and exploiting these models in the recommendation process.

Recently, social network data have been identified as an important means for increasing recommendation accuracy. [19] examines the role of social networks (SNs) in the recommendation process within a large-scalefield experiment that randomizes exposure to signals about friends' information and the relative role of strong and weak ties.[13] develops a way to increase recommendation effectiveness by combining SN information and CF methods, by collecting data about users' preference ratings and their SN relationships from a SN Web site and by developing approaches for selecting neighbors and amplifying friends' data.

Regarding the performance of CF-based systems, these have been proved toexhibit degraded performance in the presence of large volumesof data: clustering schemes have been proposed as a method to alleviate this problem [16][17][18]. Clustering schemes organize users and/oritems into clusters based on appropriate characteristics, and in the recommendation formulation process firstly the similarity between the target user or item and clusters is computed, allowing forrapidly locating similar elements and limiting processing to items only within the identified clusters.

[11] proposes a strategy to arbitrarily split the computation of user-to-user distances and neighbors into an off-line and on-line task which can temporarily satisfy the on-line time requirements, until a significant number of new users and ratings join the system. Additionally, in order to alleviate the scalability problem and to obtain a performance similar to that observed in ranking tasks, inverted indexes to evaluate queries efficiently were used. [12] proposes a multi-core CPU and a GPU implementation for the alternating least-squares algorithm to compute recommendations based on implicit feedback datasets. One central feature of the reported implementation is an algorithm-specific kernel achieving compute-bound performance for the multiplication of two dense matrices scaled by a sparse diagonal matrix. Furthermore, it proposes to reorder the sequential system generation and system solve into a batched system generation and a batched solve, to compute many systems simultaneously.

III. The Aging and Pruning Algorithms

A. The aging algorithm

The aging algorithm operates similarly to a standard user-user CF algorithm [22], except for the fact that when computing user similarity, each rating is assigned a weight based on its age: recent ratings are assigned higher weights and older ratings are assigned smaller ones. This approach is based on the rationale that aged user ratings may not accurately reflect the current state of users regarding their preferences, thus they are assigned a smaller weight.

In more detail, in order to predict the rating $p_{u,x}$ of item x for user u the aging algorithm proceeds as follows:

1. The K nearest-neighbors for user u are computed. The similarity between users u and v is computed using the formula

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} \left(r_{u,i} - \overline{r_u}\right) * \left(r_{v,i} - \overline{r_v}\right) * w\left(r_{u,i}\right) * w(r_{v,i})}{\sqrt{\sum_{i \in I_{uv}} \left(r_{u,i} - \overline{r_u}\right)^2 * \sum_{i \in I_{uv}} \left(r_{v,i} - \overline{r_v}\right)^2}}$$

   In the above formula, I_{uv} denotes the set of items that have been rated by both u and v, $r_{u,i}$ represents the rating assigned by user u to item i and $\overline{r_u}$ denotes the mean value of ratings given by user u. Finally, $w(r_{u,i})$ represents the weight assigned to rating $r_{u,i}$. Details on the weight calculation functions used in this paper are given below. The set of K users having the highest similarity values will be denoted as U.

2. Afterwards, the prediction of the rating $p_{u,x}$ is computed as

$$p_{u,x} = \overline{r_u} + \frac{\sum_{u' \in U} sim(u,u') * (r_{u',i} - \overline{r_{u'}})}{\sum_{u' \in U} |sim(u,u')|}$$

With prediction values available, typically the R items with the highest predictions are chosen and recommended to the user.

Regarding the value of the nearest neighborhood maximum size K, we have used the value 20, suggested in [28].

In this paper, we have explored two different methods of weighting old ratings, i.e. computing the age-based weight $w(r_{u,i})$ for rating $r_{u,i}$ entered by user u. The first method computes the weight of a rating $r_{u,I}$ using the standard normalization function presented in [27] i.e.

$$w(r_{u,i}) = \frac{t\left(r_{u,i}\right) - \min\limits_{x \in Ratings(u)} (\mathrm{t}\left(r_{u,x}\right))}{\max\limits_{x \in Ratings(u)} (\mathrm{t}\left(r_{u,x}\right)) - \min\limits_{x \in Ratings(u)} (\mathrm{t}\left(r_{u,x}\right))}$$

where $t(r_{u,i})$ is the timestamp of rating $r_{u,i}$; $\min_{x \in Ratings(u)}(t(r_{u,x}))$ and $\max_{x \in Ratings(u)}(t(r_{u,x}))$ denote the minimum and the maximum timestamp in the database among ratings entered by user *u*, respectively. This method will be denoted as *aging-N*. The second method applies a sigmoid function [21] to the normalized weight computed using the above formula, so as to further favor more recent ratings against older ones. In more detail, the weight of a rating *r* using the sigmoid function is

$$w^S(r) = \frac{1}{1 + e^{-12*(w(r)-0.5)}}$$

In this formula, the exponent of *e* is scaled to the range [6, 6], since [21] shows that this is the optimal domain for the sigmoid function. For values less than -6, the sigmoid function produces results very close to 0, and thus if timestamps were mapped to values -6, the sigmoid function would effectively map them to almost identical values, failing to discriminate among different timestamp values (and similarly for values greater than 6, which would be mapped to results very close to 1). This method will be denoted as *aging-S*.

All timestamps are represented in UNIX format (number of seconds elapsed since Jan 1 1970 00:00:00 UTC)

B. The pruning algorithm

The pruning algorithm operates similarly to a standard user-user CF algorithm [22], however it applies a preprocessing step to the dataset: this step retains only the *N* more recent ratings per user, dropping all older ones. Then, prediction computation and recommendation formulation proceeds as described in the case of the aging algorithm, however without using weights in the computation of user similarities.

In our experiments, reported in section IV, we explored different values for the parameter *N*. In each dataset *D* used in the experiment, we started off with a value of N_D for which 70% of the dataset users had N_D or less ratings, and then worked downwards.

IV. Performance Evaluation

In this section, we report on our experiments through which we compared the algorithms presented in section 3 and the plain collaborative filtering algorithm using the full dataset, taking into account:

1. the quality of predictions; for this comparison, we used the mean absolute error (MAE) metric. To compute the MAE, we employed the standard "hide one" technique [26]: for each user in the database we hid her last rating, and then predicted its value based on theratings on other non-hidden items. The MAE was therefore computed by considering all users in the database. The reason that only the last rating was hidden and predicted is owing to the fact that the value of each prediction should be computed using only ratings that existed in the database at the time the rating to be predicted was submitted. To further validate our results, we conducted additional experiments where the last *L* ratings of each user was dropped, with *L* varying from 1 to 5, and the algorithms listed in section 3 were run. The results obtained from these experiments were practically identical to those found when the last rating of each user was hidden, so in the following we limit our discussion to the latter case.
2. the database size.
3. the time needed to compute predictions.

For our experiments we used a laptop equipped with one dual core Intel CeleronN2840@2.16GHz CPU, 4 GB of RAM and one 240GB SSD with a transfer rate of 370MBps, which hosted the datasets and ran the recommendation algorithms. Regarding the time measurements reported below, we note that the CF implementation used in the experiments did not employ indexing or clustering techniques, therefore, the time gains reported may differ in settings where such techniques are employed. In all cases, the reported time gains provide a good insight about the speedup potential of the presented solutions.

The six datasets used in our experiments are briefly summarized in Table I. In each dataset, users having less than 10 ratings were dropped, since users with few ratings are known to exhibit low accuracy in predictionscomputed for them [22]. 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 timestamp was less than 5 seconds). These users were dropped too, 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 I. Dataset Summary

Dataset name	#users (total / retained)	#ratings (total / retained)	#items (total)	Avg. #ratings / user (retained)
Amazon "Video-games" [5][6]	826,767/ 8,057	1,324,753 / 157,495	50,210	19.6
Amazon "CDs and Vinyl" [5][6]	1.578.597/ 41,247	3,749,004 / 1,297,632	486,356	31.5
Amazon "Movies and TV" [5][6]	2,088,620/ 46,483	4,607,047 / 1,349,341	134,197	29.0
Amazon "Books" [5][6]	8,026,324/ 294,739	22,507,155 / 8,654,634	2,330,065	29.4
MovieLens "100K Dataset" [7][23]	943 / 943	100,000 / 100,000	1,682	106.0
MovieLens "ml-latest-small" [7][23]	668 / 668	105,340 / 105,340	10,330	157.7

No users were dropped from the MovieLens datasets, since these datasets contain only users that have rated 20 items or more. In the following paragraphs, we report on our findings regarding the application of the algorithms described in section III, and the setting of parameter *N* in the pruning algorithm.

A. The Amazon "Videogames" dataset

The results obtained from the Amazon Videogames dataset, are depicted in Table II. Column "% of cases" corresponds to the percentage of casesfor which the CF algorithm could computepredictions. Even when using the full dataset (the case denoted as *full*), predictions could be formulated for 88.9% of the cases; in the rest of them, the respective users had no

neighbors with a positive Pearson coefficient, i.e. no candidate recommenders [29], and therefore no prediction could be computed for them. When pruning was employed, the average number of nearest neighbors (column "avg. NNs") expectedly dropped, and so did the percentage of cases for which a prediction could be computed . The aging-based algorithms did not drop any portions of the dataset, hence both the percentage of cases for which a prediction could be computed and the average nearest neighbors remain the same with the case of the plain CF algorithm using the full dataset.

TABLE II. AMAZON VIDEOGAMES DATASET RESULTS

Method	%of cases	avg NNs	MAE CF (out of 5)	MAE CF of non-rec	% DB size reduction	% speedup
full	88.9	23.8	0.43	-	-	-
aging-N	88.9	23.8	0.44	-	0	0
aging-S	88.9	23.8	0.44	-	0	0
keep-20	87.1	20.9	0.42	0.7	24	24
keep-15	86.4	19.6	0.41	0.69	32	39
keep-10	84.3	17.3	0.41	0.65	47	57
keep-5	78.1	12.8	0.37	0.57	73	70

Column "MAE CF" corresponds to the mean absolute error of the predictions that each algorithm was able to formulate. We can observe that this metric remains fairly stable in all algorithm variants, except for the keep-5 one where it significantly drops, at the expense however of reducing the percentage of cases for which a prediction could be formulated by 10.8%. Column "MAE CF of non-rec" corresponds to the MAE of the predictions that *were computed* using the full dataset but *could not be computed* using the reduced dataset in the *keep-N* algorithm variants. Here we can observe that the predictions dropped by the *keep-N* algorithms are actually those having the highest errors; this indicates that the pruning strategy contributes towards offering higher quality predictions.

Finally, columns "% DB size reduction" and "%speedup" correspond to the savings in space and time, computed against the *full* algorithm. The two aging-based variants use the full dataset and perform minimally more computations (to calculate and use the rating weights), so their performance both space-wise and time-wise is identical to the *full* version. Contrary, the pruning-based variants introduce significant savings in both the DB-size and the time needed to compute the predictions.

Fig. 1 compares the quality of the predictions generated by the different algorithm variants, against the quality of the respective predictions generated by the *full* version. The improvement of rating *r* generated by a variant of the proposed algorithms against its counterpart r_f of the full version is defined as $\frac{absolute\_error(r_f) - absolute\_error(r)}{absolute\_error(r_f)}$. In Fig. 1 we can observe that *aging-N* and *aging-S* produce predictions of almost identical quality, which is very close to the quality of the predictions produced by the *full* version. In more detail, 82% of the *aging-N* and *aging-S* predictions are identical to those produced by the *full* version, while 97.8% of the predictions of the *aging-N* and *aging-S* predictions are "very close" (i.e. from 10% worse to 10% better) to the respective *full* version's predictions.

Regarding the pruning-based variants, we can observe that 69% of the predictions produced by the *keep-20* variant coincide with those produced by the *full* version, while the percentage of *keep-20's* predictions with quality "very close" (i.e. from 10% worse to 10% better) to those produced by the *full* version rises up to 95.6%. When the number *N* of ratings retained per user drops, we notice that the predictions produced by the pruning-based variants increasingly deviate from those generated by the *full* version. In all cases, the positive deviations (i.e. improvements against the *full* version) outnumber the negative ones (i.e. deteriorations against the *full* version), as reflected in the reduction of the MAE metric depicted in Table II.

Overall, it appears that opting for either the keep-20 or the keep-15 variant offers a speedup from 24% to 39% with virtually no effect on the MAE and with a small (from 1.8% to 2.5%) reduction in the amount of cases for which a prediction can be offered. As noted above, the predictions missed are those exhibiting high absolute error values, and thus are of lower quality.

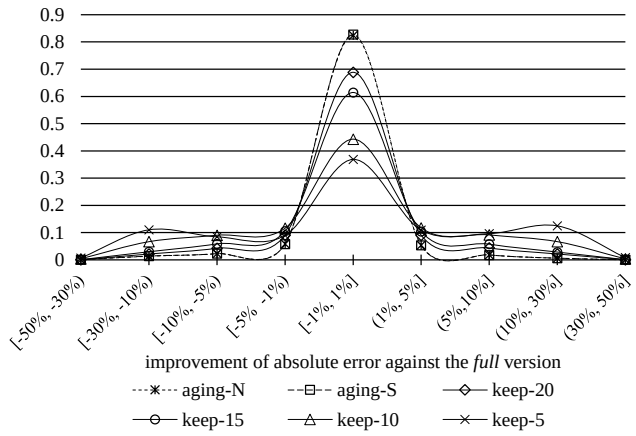

Fig. 1. Prediction quality for the Amazon videogame dataset

B. The Amazon "CDs &vinyl" dataset

Table III illustrates the results obtained from the Amazon "CDs & Vinyl" dataset. The full version could compute predictions for the 81% of the cases, with this percentage decreasing in the pruning-based versions from 2.3% (*N*=100) up to 7.1% (*N*=20). In all cases, the MAE for the computed predictions remains fairly stable. The MAE for predictions that could not be computed due to pruning is 47.2% higher than the overall MAE, showing that pruning suppresses predictions of lower quality. Finally, in the pruning-based versions DB space savings range from 21% (*N*=100) to 49% (*N*=20), with the respective time savings ranging from 16% to 53%.

Fig. 2 compares the quality of the predictions computed by the different algorithm variants, against the quality of the respective predictions computed by the *full* version. We notice that *aging-N* and *aging-S* have almost identical performance – and very similar to that of the *full* version-, but none of them achieving a MAE improvement.

TABLE III. AMAZON CDS & VINYL DATASET RESULTS

Method	%of cases	avg NNs	MAE CF (out of 5)	MAE CF of non-rec	% DB size reduction	% speedup
full	81.0	22.1	0.36	-	0	0
aging-N	81.0	22.1	0.37	-	0	0
aging-S	81.0	22.1	0.37	-	0	0
keep-100	78.7	19.6	0.36	0.53	21	16
keep-80	78.2	19.6	0.36	0.53	24	20
keep-50	77.2	17.9	0.36	0.52	31	27
keep-30	75.6	16.3	0.36	0.50	41	42
keep-20	73.9	14.6	0.36	0.49	49	53

Regarding the pruning-based variants, we can observe that 84.6% of the predictions computed by the *keep-100* variant coincide with those produced by the *full* version, while the percentage of *keep-100's* predictions with quality "very close" (i.e. from 10% worse to 10% better) to those produced by the *full* version rises up to 98.5%. When *N* drops, we notice that the predictions computed by the pruning-based variants increasingly deviate from those generated by the *full* version, but this time at a slower rate, as compared to the videogame dataset. In all cases, the positive deviations (i.e. improvements against the *full* version) are almost equal in number to the negative ones (i.e. deteriorations against the *full* version), as reflected in the almost constant value of the MAE metric depicted in Table III.

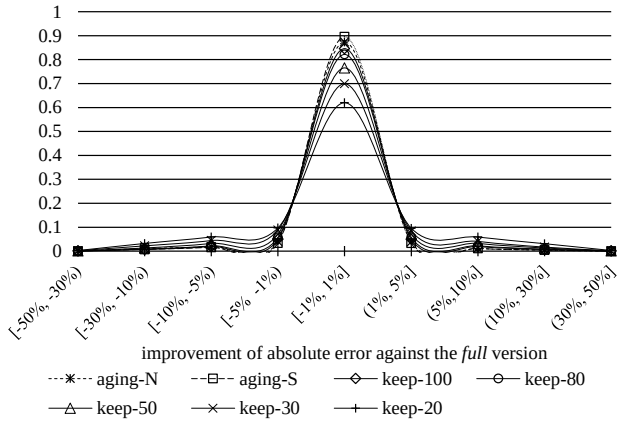

Fig. 2. Prediction quality for the Amazon CDs & Vinyl dataset

Overall, it appears that the *keep-100*, *keep-80* and *keep-50* are prominent candidates for this case, offering speedups from 16% to 27% with a small reduction (ranging from 2.3% to 3.8%) in the amount of cases for which a prediction can be computed. The predictions missed are those exhibiting high absolute error values, and thus are of lower quality.

C. The Amazon "Movies & TV"dataset

Table IV illustrates the results obtained from the Amazon "Movies& TV" dataset. The full version could compute predictions for 94.6% of the cases, with this percentage dropping in the pruning-based versions from 1.8% (*N*=30) up to 8.8% (*N*=5). In the two aging variants, the MAE increases from 5.4% (*aging-S*) to 8.1% (*aging-N*), while in the pruning-based variants the MAE remains fairly stable, except for the case of the *keep-5* version where it decreases by 8.1%, at the expense of the algorithm being unable to compute predictions for an additional 8.8% of the cases.Finally, in the pruning-based versions DB space savings range from 37% (*N*=30) to 81% (*N*=5), with the respective time savings ranging from 32% to 85%.

Fig. 3 compares the quality of the predictions computed by the different algorithm variants, against the quality of the respective predictions computed by the full version. We notice that *aging-N* and *aging-S* have very similar performance; they both however increase the MAE, since the number of deteriorations is higher than the number of improvements, especially in the ranges [-30%, -10%) and [-10, -5%) where the absolute number of predictions are approximately three times greater than the number of predictions at the corresponding positive side of the improvement axis.

TABLE IV. AMAZON MOVIES & TV DATASET RESULTS

Method	%of cases	avg NNs	MAE CF (out of 5)	MAE CF of non-rec	% DB size reduction	% speedup
full	94.6	91.3	0.37	-	0	0
aging-N	94.6	91.3	0.40	-	0	0
aging-S	94.6	91.3	0.39	-	0	0
keep-30	92.8	69.4	0.37	0.68	37	32
keep-20	92.2	63.6	0.37	0.65	45	43
keep-10	90	50	0.36	0.59	64	71
keep-5	85.8	35.1	0.34	0.53	81	85

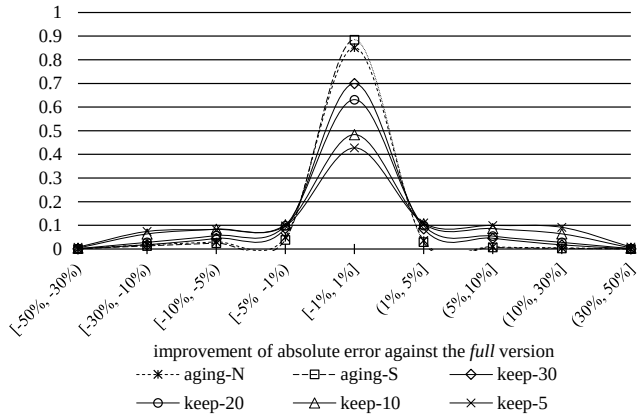

Fig. 3. Prediction quality for the Amazon Movies & TV dataset

Regarding the pruning-based variants, we can observe that 70% of the predictions computed by the *keep-30* variant coincide with those produced by the *full* version, while the percentage of *keep-30's* predictions with quality "very close" (i.e. ranging from 10% worse to 10% better) to those produced by the *full* version rises up to 96.25%. When *N* drops, the predictions computed by the pruning-based variants increasingly deviate from those generated by the *full* version, at a rate comparable to that of the Videogames dataset. In all cases, the positive deviations (i.e. improvements against the *full* version) are almost equal in number to the negative ones (i.e. deteriorations against the *full* version), as reflected in the almost constant value of the MAE metric depicted in Table IV (except for the case of *keep-5*, where the positive deviations outnumber the negative ones).

Summarizing this case, it appears that *keep-30* and *keep-20* are prominent candidates for this case, offering speedups from 32% to 43% with a small reduction (ranging from 1.8% to 2.4%) in the amount of cases for which a prediction can be computed.

The predictions missed are those exhibiting high absolute error values (72% higher than the MAE), and thus are of lower quality.

D. The Amazon "Books" dataset

Table V illustrates the results obtained from the Amazon "Books" dataset. The full version could compute predictions for 87.7% of the cases, with this percentage dropping in the pruning-based versions from 2.6% (*N*=80) up to 6.3% (*N*=25). In the two aging variants, the MAE remains almost equal to that of the *full* version, while in the pruning-based variants the MAE presents a small improvement.Finally, in the pruning-based versions DB space savings range from 21% (*N*=80) to 41% (*N*=25), with the respective time savings ranging from 18% to 44%.

TABLE V. AMAZON BOOKS DATASET RESULTS

Method	%of cases	avg NNs	MAE CF (out of 5)	MAE CF of non-rec	% DB size reduction	% speedup
full	87.7	88.4	0.34	-	0	0
aging-N	87.7	88.4	0.35	-	0	0
aging-S	87.7	88.4	0.34	-	0	0
keep-80	85.1	72.0	0.34	0.50	21	18
keep-50	84.1	65.4	0.33	0.48	28	25
keep-40	83.4	61.6	0.33	0.48	32	32
keep-30	82.3	55.8	0.33	0.47	37	41
keep-25	81.4	51.3	0.32	0.46	41	44

Fig. 4 compares the quality of the predictions computed by the different algorithm variants, against the quality of the respective predictions computed by the *full* version. We can observe that *aging-N* and *aging-S* have very similar performance, with *aging-S* having a very small performance edge. The performance of both algorithms is very similar to that of the *full* version, with 95.8% of the predictions computed by *aging-N* and *aging-S* being "very close" (i.e. ranging from 10% worse to 10% better) to the respective full verion's predictions computed by the *full* version.

Regarding the pruning-based variants, we can observe that 77% of the predictions computed by the *keep-80* variant coincide with those produced by the *full* version, while the percentage of *keep-80's* predictions with quality "very close" (i.e. ranging from 10% worse to 10% better) to those produced by the *full* version rises up to 91.6%. When *N* drops, the predictions computed by the pruning-based variants increasingly deviate from those generated by the *full* version, at a rate comparable to that of the Videogames dataset. In all cases, the positive deviations slightly outnumber the negative ones, as reflected in the value of the MAE metric depicted in Table V.

Interestingly, in this dataset we can observe an increase in the number of predictions with a high deviation, both negative and positive (i.e. in the ranges [-50%, -30%) and (30%, 50%] respectively). This is attributed to the fact that item ratings have been found to have a considerably higher variation along the time axis, as compared to other datasets. However, positive deviations outnumber negative ones by 20% (*keep-25*) to 40% (*keep-80*), hence the overall algorithm performance is not adversely affected.

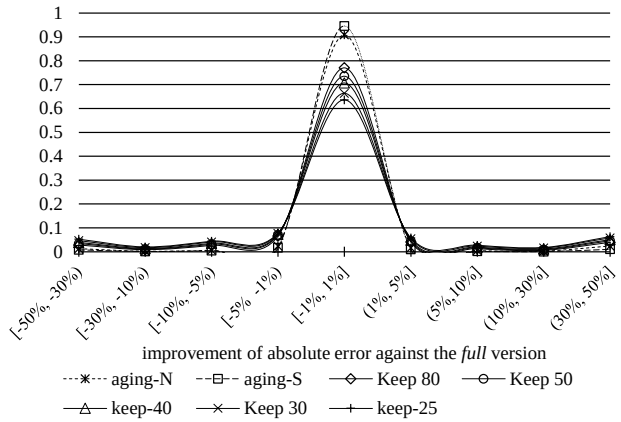

Fig. 4. Prediction quality for the Amazon Books dataset

Summarizing this case, it appears that *keep-80*, *keep-50*and *keep-40* are prominent candidates, offering speedups ranging from 18% to 32% with a small (ranging from 2.6% to 4.4%) reduction in the amount of cases for which a prediction can be computed. The predictions missed are those exhibiting high absolute error values (50% higher than the MAE), and thus are of lower quality.

E. The MovieLens simple 100K dataset

Table VI illustrates the results obtained from the MovieLens Simple 100Kdataset. The full version could compute predictions for the 99.9% of the cases; this high percentage is owing to the high density of the dataset, in which users were pre-selected to have at least 20 ratings each. This percentage dropped in the pruning-based versions from 0.1% (*N*=80) up to 3.6% (*N*=10). In the two aging variants, the MAE drops by 6.3% (*aging-S*) to 9.5% (*aging-N*); notably this is the only case where the aging-based algorithms achieved a considerable improvement over the *full* version. The pruning-based versions achieved a MAE improvement from 11% (*N*=80) to 28.6% (*N*=10), with the respective DB space savings ranging from 46% (*N*=80) to 90% (*N*=10), and the respective time savings varying from 87% (*N*=80) to 99.5% (*N*=10).

TABLE VI. MOVIELENS 100K DATASET RESULTS

Method	%of cases	avg NNs	MAE CF (out of 5)	MAE CF of non-rec	% DB size reduction	% speedup
full	99.9	121.0	0.63	-	0	0
aging-N	99.9	121.0	0.57	-	0	0
aging-S	99.9	121.0	0.59	-	0	0
keep-80	99.8	59.8	0.56	0.8	46	87
keep-50	99.7	44.4	0.51	0.85	60	95
keep-30	98.7	30.7	0.47	1.01	73	98
keep-20	98.2	21.0	0.47	0.99	81	99
keep-10	96.3	11.2	0.45	0.97	90	99.5

Fig. 5 compares the quality of the predictions computed by the different algorithm variants, against the quality of the respective predictions computed by the *full* version. The aging-based algorithms achieve the highest percentage of predictions identical to those of the *full* version but in this dataset, this percentage is limited to 42%. In these variants, most deviations (33%) fall in the range (1, 10%], and thus a total of 92.2% of the

aging-based algorithms' predictions being "very close" (ranging i.e. from 10% worse to 10% better) to the respective predictions produced by the *full* version.

Regarding the pruning-based variants, *keep-80* follows very closely the performance of the aging-based algorithms; for lower values of *N*, however, the percentage of predictions identical to those of the *full* version drops significantly, with a respective increase in the percentages being primarily in the range (10%, 30%] and secondarily in the ranges (5%, 10%] and (1%, 5%], increasing thus the overall quality of predictions (reflected in the decrease of the MAE), while the percentage of cases for which a prediction cannot be computed is not significantly affected.

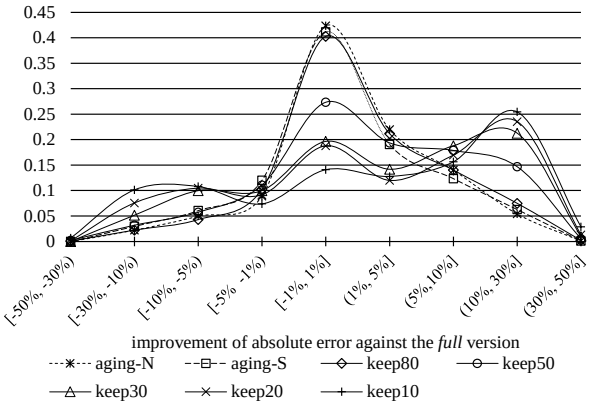

Fig. 5. Prediction quality for the MovieLens 100K dataset

Summarizing this case, it appears that *keep-50*, *keep-30* and *keep-20* are prominent candidates, offering speedups ranging from 87% to 99% with a small reduction (ranging from 0.1% to 1.7%) in the amount of cases for which a prediction can be computed. The predictions missed are those exhibiting high absolute error values (ranging from 35% to 60% higher than the MAE), and thus are of lower quality.

F. The MovieLens-latest 100K dataset

Table VII depicts the results obtained from the MovieLens latest100K dataset. The full version could compute predictions for 96.7% of cases, 3% lower than the MovieLens100K dataset. This is owing to the different properties of the dataset and especially to the number of items and users in these datasets: the MovieLens latest100K dataset contains ratings about 10,330 items and 668 users, in contrast to 1,682 items and 943 users in the MovieLens100K dataset. According to these, the average density $d = \frac{\#ratings}{\#items * \#users}$ is 0.015 for the MovieLens latest100K dataset and 0.063 (i.e. 4 times larger) in the MovieLens100K dataset. This in turn leads to a significantly lower number of nearest neighbors for the MovieLens100K latest dataset (39.4 compared to 121 in the MovieLens100K dataset) and this leads in the increased presence of grey sheep, both in the full dataset and the pruned ones. In more detail, the percentage for which a prediction could be computed drops in the pruned-based version drops by 2.4% (N=300) up to 12.1% (N=50).

Regarding the MAE, the aging-based algorithms exhibit a slight improvement, while the pruning-based algorithms achieve a reduction ranging from 10.7% (N=300) to 20% (N=50), at the expense of reducing the percentage of cases for which a prediction can be computed. The respective DB space savings range from 31% to 73%, while the respective time savings vary from 69% to 89%.

TABLE VII. MOVIELENS LATEST 100K DATASET RESULTS

Method	%of cases	avg NNs	MAE CF (out of 5)	MAE CF of non-rec	% DB size reduction	% speedup
full	96.7	39.4	0.84	-	0	0
aging-N	96.7	39.4	0.81	-	0	0
aging-S	96.7	39.4	0.82	-	0	0
keep-300	94.3	30.7	0.75	0.946	31	69
keep-150	91.6	24.3	0.72	0.985	47	76
keep-100	89.1	20.2	0.70	0.952	58	82
keep-80	88.2	17.5	0.68	0.973	63	88
keep-50	84.6	12.9	0.67	1.033	73	89

Fig. 6 compares the quality of the predictions computed by the different algorithm variants, against the quality of the respective predictions computed by the *full* version. The aging-based algorithms achieve the highest percentage of predictionsidentical to those of the *full* version which, in this dataset is approximately 65%. In these variants, most deviations fall in the range [-10, 10%], and thus a total of 96% of the aging-based algorithms' predictionsare"very close" (i.e. from 10% worse to 10% better) to the respective predictions produced by the *full* version.

Regarding the pruning-based variants, this dataset exhibits the lowest percentage of predictions exactly matching those of the *full* version; this percentage ranges from 15.7% (N=150) to 12.9% (N=50). The highest peak in the improvement distribution is located in the range (10%, 30%], which accounts for approximately 21.5% of the total predictions, followed by ranges (5, 10%] and [-30%, -10%), each one accounting for 13.5% of the total predictions. Overall, positive deviations outnumber the negative ones, and this is reflected in the reduction of the MAE which can be observed in Table VII.

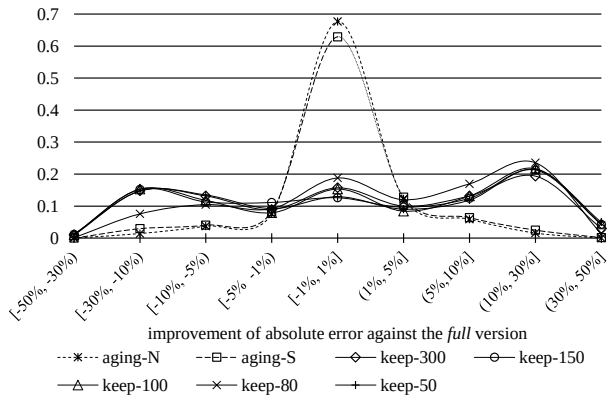

Fig. 6. Prediction quality for the MovieLens Latest 100K dataset

Overall, for this case, it appears that *keep-300* and *keep-150* are prominent candidates, offering speedups ranging from 69% to 76% with a small to tolerable reduction (ranging from 2.4% to 5.1%) in the amount of cases for which a prediction can be

computed. The predictions missed are those exhibiting high absolute error values (ranging from 26% to 34.9%), and thus are of lower quality.

V. CONCLUSION AND FUTURE WORK

In this paper we have presented two algorithms for aging and pruning user histories in collaborative filtering systems. The proposed algorithms exploit rating timestamps in the recommendation process, with the aging-based algorithm assigning weights to ratings based on their age (older ratings are given smaller weights), and the pruning algorithm retaining only the N newest ratings of each user. The proposed algorithms are evaluated against six datasets in terms of prediction accuracy, storage size gains and execution speedup. The results show that the pruning algorithm increases prediction quality and decreases prediction computation time, while achieving considerable database size gains. On the other hand, the aging algorithm exhibits performance similar to that of a typical CF algorithm. A merit of the suggested pruning technique is that it can be applied as a preprocessing step to any CF-based algorithm.

Our future work will focus on creating an unsupervised version of the pruning algorithm; we envision this algorithm to be able to automatically determine the values of N that should be tested (the value of N_D used in this paper for which 70% of the dataset D users had N_D or less ratings is a prominent starting point, however a stopping criterion and a method to determine how much N should be reduced in the next trial need to be established). Efficient and accurate sampling methods to avoid the computation of predictions for all users to determine the MAE of each algorithm will be also examined. Finally, as new ratings are constantly added in the ratings database, the formulation of triggering conditions for executing the pruning algorithm will be studied.

REFERENCES

[1] Bakshy, E, Eckles, D., Yan, R., Rosenn I.,"Social Influence in Social Advertising: Evidence from Field Experiments", Proceedings of the 13th ACM Conference on Electronic Commerce, 2012, pp. 146-161

[2] Schafer, JB., Frankowski, D., Herlocker, J., Sen, S.,"Collaborative Filtering Recommender Systems", in "The Adaptive Web", Lecture Notes in Computer Science Volume 4321, 2007, pp. 291-324

[3] Yang Song , Ali Elkahky, Xiaodong He, "Multi-Rate Deep Learning for Temporal Recommendation", Proceedings of the 39th International ACM SIGIR Conference, SIGIR 2016

[4] L. Li, L. Zheng, F. Yang, T. Li, "Modeling and broadening temporal user interest in personalized news recommendation", Expert Systems with Applications 41 (7), 2014, pp. 3168-3177

[5] J.J. McAuley, R. Pandey, J. Leskovec, "Inferring Networks of Substitutable and Complementary Products", Proceedings of KDD 2015: pp. 785-794

[6] J. McAuley, C. Targett, J. Shi, A. van den Hengel,"Image-based recommendations on styles and substitutes", Proceedings of the 38th international ACM SIGIR conference, SIGIR, 2015

[7] MovieLens datasets, http://grouplens.org/datasets/movielens/

[8] Y.Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering", ACM Transactions on Knowledge Discovery from Data (TKDD), Volume 4 Issue 1, January 2010

[9] A.Whitby, A.Jøsang, J. Indulska, "Filtering Out Unfair Ratings in Bayesian Reputation Systems", Proceedings of the Workshop on Trust in Agent Societies, at the Autonomous Agents and Multi Agent Systems Conference (AAMAS2004), New York, July 2004

[10] X.Liu, C.Aggarwal, Y-F.Li, X.Kong, X.Sun and S.Sathe,"Kernelized Matrix Factorization for Collaborative Filtering", SIAM International Conference on Data Mining, 2016

[11] V.Anthony,A.Ayala, A.Alzoghbi, M.Przyjaciel-Zablocki, A.Schätzle, G.Lausen, "Speeding up Collaborative Filtering with Parametrized Preprocessing", Proc. of the 6th International Workshop on Social Recommender Systems (SRS 2015), in conjunction with the 2015 ACM SIGKDD Conference (KDD 2015). Sydney, Australia, August 2015.

[12] M.Gates, H.Anzt, J.Kurzak, J.Dongarra, "Accelerating collaborative filtering using concepts from high performance computing", Proceedings of the 2015 IEEE International Conference on Big Data, 2015.

[13] F.Liu, H. Joo Lee, "Use of social network information to enhance collaborative filtering performance", Expert Systems with Applications: An International Journal archive 37(7), July 2010 pp.4772-4778

[14] M. Balabanovic, Y. Shoham, "Fab: content-based, collaborative recommendation",Communications of the ACM, 40(3), 1997, pp. 66-72.

[15] R. Burke, "Hybrid recommender systems: Survey and experiments", User Modeling and User-Adapted Interaction 12 (2002), no. 4, 331– 370

[16] S. Gong, "A Collaborative Filtering Recommendation Algorithm Based on User Clustering and Item Clustering", Journal of Software, Vol 5, No 7, 2010, pp. 745-752.

[17] A. Das, M. Datar, A. Garg, S. Rajaram, "Google News Personalization: Scalable Online Collaborative Filtering", Proceedings of the 16th international conference on World Wide Web, 2007, pp. 271-280.

[18] D. Margaris, P. Georgiadis, C. Vassilakis, "A Collaborative Filtering Algorithm with Clustering for Personalized Web Service Selection in Business Processes", Proceedings of RCIS 2015, pp. 169-180.

[19] Bakshy, E., Rosenn, I., Marlow, C., Adamic L.: The role of social networks in information diffusion. Procs. of the 21st international conference on World Wide Web, 519-528 (2012)

[20] M. Gao, Z. Wu,"Personalized Context-Aware Collaborative Filtering Based on Neural Network and Slope One", Proceedings of the 6th International Conference on Cooperative Design, Visualization, and Engineering, Volume 5738 of the series, 2009, pp 109-116.

[21] J. Han, C. Morag, "The influence of the sigmoid function parameters on the speed of backpropagation learning". In Mira, José; Sandoval, Francisco. From Natural to Artificial Neural Computation, 1995, pp. 195–201

[22] M. D. Ekstrand, J. T. Riedl, J. A. Konstan, "Collaborative Filtering Recommender Systems". Foundations and Trends in Human-Computer Interaction 4(2), Feb. 2011, pp. 81-173.

[23] F. Maxwell Harper, J. A. Konstan, "The MovieLens Datasets: History and Context", ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages.

[24] R. Jaschke, L. Marinho, A. Hotho, L. Schmidt-Thieme, G. Stumme, "Tag Recommendations in Folksonomies", Proceedings of the 11th European conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 07), 2007, pp. 506 – 514

[25] D. Parra-Santander, P. Brusilovsky,"Improving Collaborative Filtering in Social Tagging Systems for the Recommendation of Scientific Articles". Proceedings of Web Intelligence 2010, pp. 136-142.

[26] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, H.P. Kriegel,"Probabilistic Memory-Based Collaborative Filtering", IEEE Trans. on Knowl. and Data Eng. vol.16, no. 1, January 2004, pp. 56-69.

[27] D. He, D. Wu, "Toward a robust data fusion for document retrieval", Proceedings of the IEEE 4th International Conference on Natural Language Processing and Knowledge Engineering – NLP-KE, 2008.

[28] J. Herlocker, J. A. Konstan, and J. Riedl,"An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms", Information Retrieval, vol. 5, no. 4, 2002, pp. 287–310

[29] U. Shardanand and P. Maes, "Social Information Filtering: Algorithms for Automating "Word of Mouth"". Proceedings of the 1995 SIGCHI Conference on Human Factors in Computing Systems, 1995, pp. 210-217