future internet

MDPI

Article

Optimizing Collaborative Filtering for Accurate Rating Predictions in Very Sparse Datasets

Sofia-Anna Lapadaki [1], John Nanos [2], Dionisis Margaris [2], Costas Vassilakis [1] and Dimitris Spiliotopoulos [3,*]

[1] Department of Informatics and Telecommunications, University of the Peloponnese, Akadimaikou G. K. Vlachou, 22131 Tripoli, Greece; dit2521cst@go.uop.gr (S.-A.L.); costas@uop.gr (C.V.)
[2] Department of Digital Systems, University of the Peloponnese, 23100 Sparta, Greece; ioananos@go.uop.gr (J.N.); margaris@uop.gr (D.M.)
[3] Department of Management Science and Technology, University of the Peloponnese, Sehi Location (Former 4th Shooting Range), 22131 Tripoli, Greece
* Correspondence: dspiliot@uop.gr

Abstract

Collaborative filtering is one of the most widely used methods for user rating prediction in recommender systems. To evaluate a collaborative filtering system, rating datasets are typically used, which comprise thousands to millions of records consisting of user–item–rating tuples. Initially, a similarity metric is used to quantify the closeness between each user and every other user in the dataset, typically based on the ratings that each pair of users has given to the same items. Subsequently, the K users having the largest similarity to the target user are used to produce rating predictions, which lead to recommendations. A particularly challenging case arises when the rating dataset is very sparse. In this scenario, it is difficult not only to find users with commonly rated items but also to determine the optimal similarity metric and suitable values for variable K. Setting a small value for K results in extremely low prediction coverage, leading to unsuccessful recommendations, while setting a very large K value increases memory requirements and prediction/recommendation generation time. Through a multiparameter experiment, this work aims to determine the optimal settings for rating predictions when very sparse datasets are used in collaborative filtering recommender systems.

Keywords: collaborative filtering; rating prediction; sparse datasets; recommender systems; K-nearest neighbors (KNN); prediction accuracy; similarity metrics; sparse collaborative filtering; optimal prediction settings

Academic Editor: Ivan Serina

Received: 26 December 2025
Revised: 17 February 2026
Accepted: 20 February 2026
Published: 23 February 2026

1. Introduction

Collaborative filtering (CF) is widely regarded as one of the most commonly used memory-based methods for predicting user ratings, which ultimately lead to the generation of recommendations. The closer these predictions are to actual user preferences, the more successful the final recommendations will be. To evaluate a CF system, rating datasets are typically employed. These datasets consist of thousands to millions of records containing user–item–rating tuples (three attributes in total). Many of these datasets also include a timestamp for each rating, meaning that each record may contain a fourth attribute [1–3].

The first step of a typical CF rating prediction algorithm calculates the similarity between each pair of users in the dataset using a similarity metric. The higher the calculated

similarity between two users, the closer they are considered to be, and therefore the probability that these users will like the same items is relatively high. The most commonly used CF similarity metrics, such as the Cosine similarity and Pearson Correlation Coefficient, are typically based on the ratings given by users to common items [4,5].

Then, the second step of a typical CF algorithm determines the set of nearest neighbors (NNs) for each user. Typically, the NN for each user includes the K users with the highest similarity to that user, which will be denoted as KNN. The value of K is set by the recommender system's administrator and is usually in the range of hundreds to a few thousand [6–8].

The third step of the CF algorithm generates rating predictions using a rating prediction formula, such as the weighted average and the mean-centered approach [4,5,9]. These formulas synthesize the rating values that only the user's NNs have given to the item for which the prediction is being produced.

The overall success of a CF recommender system is typically assessed using evaluation metrics. The most commonly used metrics include (a) prediction coverage (i.e., the percentage of cases for which a prediction can be calculated), (b) mean absolute error (MAE) and root mean square error (RMSE), which indicate how close rating predictions are to actual ratings, and (c) the F1 measure, which combines precision and recall and indicates the quality of recommendations. For the MAE and RMSE, lower values are considered better, while for all of the other metrics, higher values are preferred [10,11].

A special and extremely challenging case for a CF arises when the rating dataset is very sparse (i.e., when the number of ratings is much smaller than the product between the number of users and the number of items), with its density typically being less than 1%. In this extreme case, it is difficult not only to find users with commonly rated items (a procedure that is part of the first step in a typical CF process) but also to determine suitable values for the variable K [12,13].

Considering the latter issue (the K variable), it is particularly difficult for the CF system administrator to decide on a specific value, as choosing a small value will result in extremely low prediction coverage, leading to mediocre or even unsuccessful recommendations. On the other hand, opting for a very large K value to avoid the previous issue will require significant amounts of memory and increase the time needed to generate predictions and, consequently, to provide recommendations. Based on the above, it is crucial to determine the appropriate range of values for the K variable in (very) sparse datasets.

Furthermore, the most well-known model-based method in recommender systems, Matrix Factorization (MF), also faces significant challenges when applied to sparse datasets. Specifically, MF techniques struggle with sparse datasets because the lack of sufficient ratings can lead to poor generalization, as the model has insufficient information to accurately learn the latent factors that explain user–item interactions [14–16]. In practice, when applying traditional MF techniques to very sparse datasets, predictions involving users or items with very few ratings tend to degenerate into a dataset-dependent constant value [17,18]. From the observations presented above, it follows that extreme data sparsity in CF recommender systems negatively affects both memory-based and model-based techniques.

If a sparse dataset contains additional information about users, items, or the evaluations themselves, in which case it is called an *enriched dataset*, we can exploit this additional information to alleviate the sparsity issue [19]. Examples of methods in this recommender system category include those that utilize (a) demographic and contextual data [20–22], (b) item attributes [23,24], (c) textual semantics [25–27], and (d) cross-domain data [28,29].

Based on the above, the use of a sparse dataset, which has no additional information, causes very serious problems in recommender systems that jeopardize the reliability of recommendations and, therefore, the success of the system. While these issues exist in

recommender systems operating on sparse and very sparse datasets, no systematic analysis of the behavior of the algorithms under different near neighborhood sizes, similarity metrics, and rating prediction formulas has been published. Indeed, many recent studies (e.g., [5,18,30–34]) use divergent settings, hindering comparability between studies and necessitating additional work for practitioners and researchers. Additionally, sparse and very sparse datasets suffer from low coverage, an issue that is not adequately tackled in most of the literature, which focuses solely on rating prediction accuracy and recommendation precision/recall. Notably, the use of specific similarity metrics may adversely affect coverage, as is the case with the Pearson similarity metric, which is widely employed in CF research. Finally, as noted above, Matrix-Factorization-based and ML-based techniques currently have limited interpretability and explainability when operating on sparse or very sparse datasets [14–16] and/or may tend to produce predictions that degenerate to constant dataset-specific values [17,18].

Taking the above into account, [35] asserts that in the context of sparse and very sparse datasets, neighborhood-based models constitute a more appropriate option for the implementation of recommender systems, providing increased coverage, accuracy, robustness, and effectiveness. Following these findings and considering that in this paper we focus on datasets with high and very high sparsity, we limit our study and experiments to neighborhood-based models.

Considering these observations, which constitute the motivation of our study, the initial objective of this work is to find the region where the optimal values of variable K are located for very sparse datasets. To achieve this, in this study, we run a multiparameter experiment using

- Ten sparse datasets, with densities ranging from 0.33% (the densest) to 0.005% (the sparsest);
- Three widely used similarity metrics, as well as one additional composite metric that synthesizes two individual ones;
- Two widely used rating prediction formulas.

All are commonly used in CF recommender systems research.

As a result, this work extends its scope beyond finding the optimal value(s) for the K parameter in CF recommender systems to include the identification of the optimal parameters (number of NNs (K), similarity metric, and rating prediction formula) a CF recommender system needs in total. To ensure the reliability of the results, we use datasets from different sources, categorize them into three density levels, and examine each density level separately. The findings of this work will provide researchers and professionals in CF recommender systems with the ability to directly determine and use optimal settings when working with very sparse datasets.

The remaining sections of this paper are organized as follows: in Section 2, a literature review of sparse datasets in recommender systems is outlined, while in Section 3 the foundations of our work are summarized. In Section 4, the settings of our experiments are initially introduced, and, afterwards, the results are presented and analyzed. Section 5 discusses the results, and, finally, Section 6 concludes the paper and outlines future work.

2. Literature Review

The accuracy of rating predictions in CF recommender systems is a research area that has attracted numerous studies over the last years. Many of these studies are applied to dense datasets, like the MovieLens ones, where the issue of low prediction coverage does not exist, and therefore these studies solely target prediction accuracy [11,36–38]. The evaluation metrics of the aforementioned studies mainly include the MAE, RMSE, and F1 score (which combines precision and recall metrics).

Over the last few years, many studies have focused on sparse datasets, which are considered a special and extremely challenging case for CF recommender systems, as mentioned in the Introduction. These studies are divided into two major categories. The first one consists of studies that include datasets having additional information, like item categories, social networking, and user reviews (termed as enriched datasets). Although this additional information helps them overcome the low coverage issue, these studies do not have universal applicability, as they cannot be applied to plain datasets that do not contain this extra information. The second category includes studies that are based solely on the data stored in the rating matrix and therefore retain universal applicability; however, the challenge presented in the Introduction remains. In the following subsections, we review studies from both of the aforementioned categories.

2.1. Studies with Enriched Datasets

Regarding the first category, a group of work in the literature exploits social network information to complement the user–item rating matrix, aiming to enhance rating prediction and recommendation quality, while a second group utilizes and processes review texts to elicit ratings (and, provisionally, other traits), which are then processed to generate rating predictions and recommendations.

Within the group of work that utilizes and processes review texts, [39] enhances CF by incorporating user reviews to address sparse or unavailable explicit ratings. It proposes four methods for calculating implicit ratings using sentiment analysis; it uses the Euclidean distance to measure user similarity and deploys the KNN selection method on the Amazon Movies_and_TV_5core dataset (with a density of 0.027%) and the Yelp dataset (with a density of 0.0017%) using MAE, MSE, and RMSE prediction error metrics. This approach outperforms other implicit rating calculation methods by leveraging information in user reviews, exploiting both individual sentiment words and aspect–sentiment word pairs.

User reviews are also leveraged in [30], where a rating-style mining method is introduced to enhance the CF algorithm in RecSys. By addressing the differences in users' rating styles before calculating similarity measures, the proposed method improves prediction accuracy. This is implemented by processing user reviews using a pre-trained BERT model, whose output is fed into a softmax layer, and, finally, the output is fine-tuned. Then, the output of this process is used to calculate the Pearson similarity metric between users. It deploys the KNN selection method with different K values (ranging from 10 to 190 with a step of 20) on the Amazon Movies_and_TV_5core dataset (with its density calculated at 0.027%) and evaluates the prediction error using the MAE and RMSE metrics.

The study in [40] also utilizes user review texts to address the cold start problem in recommender systems by combining NLP and machine learning with CF. By leveraging user reviews and NLP techniques that exploit features such as review usefulness and remove any fake or inconsistent ratings, the proposed approach ensures cleaner training data and improves accuracy and scalability in CF-based recommender systems. Furthermore, user feedback is collected and used to further refine the recommendation model. The datasets used include the Epinions dataset (with a density of 0.014%) and the Book-Crossing dataset (with a density of 0.014%).

User reviews are processed in [41] for the creation of an enhanced recommender system that integrates sentiment analysis with CF techniques. By analyzing user reviews using a sentiment model, the system improves recommendation accuracy. The approach comprises three stages: firstly, unsupervised "GloVe" vectorization is employed to improve classification performance and build a sentiment model using Bidirectional Long Short-Term Memory (Bi-LSTM). The sentiment model is then used to compute ratings from review texts, and these are finally used to generate predictions. The study uses the Amazon Digital

Music dataset (older version), with a density of 0.074%. The study in [42] proposes a Social Promoter Score-based CF recommender system that addresses the limitations of traditional rating-based models by incorporating explicit and implicit user–item interaction data and deep neural networks. The proposed algorithm constructs two user–item interaction matrices, the first being populated with explicit ratings and the second reflecting users' viewing activities, which are interpreted as implicit feedback. These matrices are then fed into an attention-layer-based deep neural model, which learns a common low-dimensional space that corresponds to the features of users and items and provides insight into how users rate items. This insight is then utilized to improve rating prediction accuracy. The work uses six sparse Amazon datasets, including Instant Video (with a density of 0.1%), and Home & Kitchen (with a density of 0.024%).

Considering studies that complement the user–item rating matrix with social network data, [12] introduces a hybrid CF method for personalized friend recommendations in social networks. By combining social and semantic information, the proposed approach aims to overcome data sparsity and cold start problems. This is achieved by firstly applying the incremental K-means algorithm to all users and subsequently using the KNN algorithm on new users. It uses a preprocessed Yelp dataset with a density of 0.45%. The work in [43] presents an adaptive CF algorithm that enhances rating prediction accuracy by incorporating social network relationships. The algorithm addresses challenges posed by limited social network data or CF information by computing two distinct rating predictions based on the target user's CF neighborhood and social network neighborhood, respectively, and then synthesizing the two predictions using personalized weights, thus minimizing the prediction error. The work uses the Pearson and Cosine similarity metrics on four sparse datasets, including CiaoDVD (with a density of 0.12%) and Epinions (with a density of 0.014%).

Following the same rationale, [44] proposes a recommender system that combines social relationships and user interaction data using graph convolutional CF techniques. By integrating both user similarities and social influences, the model significantly improves recommendation accuracy and recall. The main challenge addressed in that work is the presence of noise in the data; this is tackled through the identification and removal of single reviews or users with very limited interaction with the items, allowing subsequent steps of the algorithm to operate on a cleaner dataset. The datasets used in the study include the CiaoDVD dataset (with a density of 0.12%) and the Epinions dataset (with a density of 0.014%). The work in [45] presents a product recommender system using CF techniques to offer personalized suggestions on e-commerce platforms. By analyzing user preferences, purchase history, and ratings from similar customers, the system generates more accurate recommendations. To this end, product popularity is evaluated and CF is used to generate predictions; products are then combined into clusters using K-Means clustering, and clusters are used for generating recommendations. This study utilizes the Amazon Electronics dataset, with its density measured at 0.006%.

As mentioned above, while all of the above-mentioned studies are found to alleviate the dataset sparsity issue, they can only be applied to enriched datasets, as they rely on additional data to do so and therefore are not considered to have universal applicability.

2.2. Studies That Are Based on the Data Stored in the Rating Matrix

Regarding the second category, two main lines of work can be identified in the literature: the first line of work focuses on evaluating existing algorithms and metrics under different datasets and/or settings, while the second line of work introduces new similarity metrics and/or combines or modifies existing ones to increase the quality of rating predictions and recommendations.

Within the first line of work, [5] evaluates user similarity metrics in CF datasets. This work uses the Pearson, Cosine, and Jaccard similarity metrics, among others, and deploys the KNN selection method, setting the K value to 250 and 500, on all 10 datasets tested. These datasets are considered very sparse, with densities ranging from 0.24% (the Yahoo Movies dataset) to 0.014% (the Book-Crossing dataset). The prediction error metrics used are MAE, RMSE, and F1. While the study examines sparse datasets and multiple similarity metrics, coverage aspects are not considered, while only limited values of K are examined, which, as shown in the study presented in this paper, are lower than the optimal values for very sparse and extremely sparse datasets.

The evaluation survey presented in [31] compares memory-based CF models and deep learning models to address cold starts and data sparsity in e-commerce recommender systems. It uses the Cosine similarity metric and deploys the KNN method, giving K values of 10, 30, and 50. The work evaluates both memory-based CF, including Singular Value Decomposition (SVD), SVD++, KNN Baseline, and K-Nearest Neighbors (KNN) Basic, for which the hyperparmeters are optimized using grid search. The work also examines deep learning models, including LSTM, BiLSTM, Att+BiLSTM, User-Product Fusion (UPF), and Deep CF. The dataset tested is the Amazon Musical Instrument dataset, with density measured at 0.001%, while the error metrics used are MAE and RMSE. The study asserts that memory-based models (such as SVD++) are more suitable for the recommendation task in the context of sparse datasets than relying solely on DL models; nevertheless, it is noted that, again, the values of K examined in the work are (significantly) lower than the optimal values for very sparse and extremely sparse datasets.

Considering work introducing new similarity metrics and/or combining or modifying existing ones, [32] presents a trust CF approach for explainable recommendations. It aims to enhance recommendation quality while providing transparency on why and how items are recommended based on trustworthiness modeling. Trustworthiness between the active user U_{act} and a user U is computed on the basis of the degree to which U_{act} chooses the items based on user U. To enhance explainability and trust towards the recommender system, recommendations are complemented with the number of users sharing the same preferences with the target user and average trustworthiness. The method is evaluated by applying the Cosine similarity metric with the NN method (with K ranging from 10 to 80) on the Amazon Instant Video dataset, which has a density of 0.43%, while the error metric used is the RMSE.

In [46], the authors introduce an algorithm that augments the importance of NN pairs with opinion agreement on products that deviate from the dominant community opinion on the same products. To this end, the work computes a "black sheep factor" between a user and each of his/her NNs, expressing the degree to which the users rate similarly some items while at the same time disagreeing with the majority of other users on the same items; then, this factor is used to compute an enhanced similarity value, which is considered in rating prediction computation. The proposed algorithm is evaluated using the Pearson and Cosine similarity metrics, applied to four sparse datasets (including an older version of the Amazon Videogames dataset with a density of 0.09%). The NNs are selected using the threshold approach, while the error metrics utilized are MAE and RMSE.

The augmentation of the Pearson correlation similarity metric is also considered in [33], which introduces a multi-level CF method aiming to improve decision making by providing higher-quality recommendations across various online domains. The proposed approach is evaluated using the plain Pearson correlation similarity metric, as well as two Pearson correlation variations. Effectively, this work enhances standard Pearson correlation similarity by considering the number of items that users have rated in common and the value of the Pearson similarity metric between users; for users that are "sufficiently similar"

and have co-rated a high number of items, their final similarity metric is boosted to augment the weight of the respective ratings in rating prediction calculation. The set of datasets tested includes the Epinions dataset, having a density of 0.014%. The KNN method is used with a K value set to 5 and 10, and the prediction error metrics used are MAE, precision, and recall.

Finally, another approach to an augmented similarity metric is presented in [34]. More specifically, this work proposes an improved product recommendation method for CF, utilizing triangle similarity and considering both common and uncommon ratings between users, along with user rating preferences, to enhance recommendation accuracy. Essentially, for computing the final user-to-user similarity metric that will be used for rating prediction calculation, the work considers user proximity (likeness of ratings), significance (the deviation of user ratings from the median), and singularity (a metric expressing the difference of both users' average rating from the average rating of the target item). In this context, the Jaccard, Pearson, and Cosine similarity metrics are used and the KNN selection method is deployed with different K values (ranging from 5 to 50, with a step of 5) on six datasets, including the Epinions and CiaoDVD datasets (with their densities calculated at 0.014% and 0.12%, respectively). The evaluation metrics used are MAE, RMSE, and F-score. Similarly to previous cases, the values of K examined in this work are (significantly) lower than the optimal values for very sparse and extremely sparse datasets, and coverage aspects are not considered.

Table 1 provides a summary of the aforementioned studies, which are based on the data stored in the rating matrix.

Table 1. Recent CF studies that are based on the data stored in the rating matrix.

Study	Datasets Used	Sparsity of Datasets	Prediction Error Metrics	CF Settings (Similarity Metrics and K)
Yang et al. [30]	Amazon Movies and TV (5 core)	0.027%	MAE, RMSE	Pearson, K = 10(20)190
Sgardelis et al. [5]	10 datasets (Yahoo, Amazon, Book-Crossing)	From 0.24% to 0.014%	MAE, RMSE, F1	Pearson, Cosine, and Jaccard (and others), K = 250, 500
Shetty et al. [31]	Amazon Musical Instrument dataset	0.001%	MAE, RMSE	Cosine, K = 10, 30, 50
Zarzour et al. [32]	Amazon Instant Video	0.43%	RMSE	Cosine, K = 10 to 80
Margaris et al. [46]	Amazon Videogames, CDs and Vinyl, Movies and TV, Books (older versions)	0.09%, 0.02%, 0.03%, 0.004%	MAE, RSME	Cosine, Pearson, threshold > 0.0
Polatidis and Georgiadis [33]	Epinions	0.014%	MAE, precision, recall	Pearson (+2 variations), K = 5, 10
Iftikhar et al. [34]	Epinions, CiaoDVD	0.014%, 0.12%	MAE, RMSE, F1	Jaccard, Pearson, Cosine, K = 5(5)50

While the above-mentioned studies apply CF algorithms to really sparse datasets, they use predetermined settings (similarity metrics, K, etc.) in their experiments. In some cases, we even observe different settings being applied to the same datasets.

In this paper, we aim to identify the optimal settings a CF recommender system needs in total (including the parameter K, similarity metric(s), and rating prediction formula) when employing very sparse datasets, like those mentioned in the previous paragraphs. In this context, we consider the coverage aspect, which is neglected in most studies, and explore higher values for the K parameter, which plays a significant role in offering

personalized predictions to users in sparse dataset settings. To ensure the reliability of the results, we use datasets from different sources, categorize them into three density levels, examine each density level separately, and, finally, consider prediction coverage. The findings of this work will provide researchers and practitioners in CF recommender systems with the ability to directly determine and use optimal settings when working with very sparse datasets. This will facilitate the use of memory-based recommender systems, which provide higher explainability and allow for seamless operation, independently of the content that the algorithm is applied on [47].

3. Foundations

In this section, the foundations of our work are briefly presented. More specifically, we summarize the overall procedure for CF prediction generation, including the user similarity metrics, the KNN selection method, and, finally, the rating prediction formulas.

When generating predictions for CF systems, first, the similarity of each user in a dataset to every other user is computed by using a particular similarity metric. In the context of the CF literature reviewed in the prior section, the Jaccard, Cosine, and Pearson similarities are some of the most common metrics.

Jaccard similarity (JS) between users U1 and U2 is defined as the ratio of the number of items rated by both users (intersection) to the total number of unique items rated by both users (union), as shown in Equation (1). The metric is only based on the number of items rated in common, regardless of the actual rating values. The range of JS is [0, 1], where larger values signify greater similarity between the two users [36,44].

$$\mathrm{JS}(\mathrm{U1},\mathrm{U2}) = \frac{|\mathrm{I}(\mathrm{U1}) \cap \mathrm{I}(\mathrm{U2})|}{|\mathrm{I}(\mathrm{U1}) \cup \mathrm{I}(\mathrm{U2})|} \tag{1}$$

where I(Ux) represents the set of items user Ux has rated.

The Cosine similarity (CS) between users U1 and U2 in CF is derived from the Euclidean dot product of the two user rating vectors, as illustrated in Equation (2).

$$\mathrm{CS}(\mathrm{U1},\mathrm{U2}) = \frac{\sum_k \mathrm{r}_{\mathrm{U1,k}} \cdot \mathrm{r}_{\mathrm{U2,k}}}{\sqrt{\sum_k (\mathrm{r}_{\mathrm{U1,k}})^2} \cdot \sqrt{\sum_k (\mathrm{r}_{\mathrm{U2,k}})^2}} \tag{2}$$

where $k \in I(U1) \cap I(U2)$ and $r_{X,k}$ is the rating that user X has set for item k [36,48]. Generally, the range of the CS metric is $[-1, 1]$, with higher values denoting higher similarity, while if rating values are only positive (which is the case for the datasets used in this paper) the range is limited to [0, 1].

The Pearson Correlation similarity (PS) measures the linear correlation of the rating sets of users U1 and U2. PS can be derived from the CS, with each user's rating values normalized by the average rating value of the user's own ratings, as shown in Equation (3). The range of PS is $[-1, 1]$, where, again, higher values signify higher similarity between the two users [49–51].

$$\mathrm{PS}(\mathrm{U1},\mathrm{U2}) = \frac{\sum_k (\mathrm{r}_{\mathrm{U1,k}} - \overline{\mathrm{r_{U1}}}) \cdot (\mathrm{r}_{\mathrm{U2,k}} - \overline{\mathrm{r_{U2}}})}{\sqrt{\sum_k (\mathrm{r}_{\mathrm{U1,k}} - \overline{\mathrm{r_{U1}}})^2} \cdot \sqrt{\sum_k (\mathrm{r}_{\mathrm{U2,k}} - \overline{\mathrm{r_{U2}}})^2}} \tag{3}$$

where $k \in I(U1) \cap I(U2)$ and $\overline{r_X}$ represents the average value user X has given across all their ratings.

Considering these three similarity metrics, we can observe that while the JS takes into account (only) the percentage of common items between two users, the other two similarity metrics consider (only) the rating values the users have given. Because both of these aspects

are indicators of user profile resemblance, researchers have suggested combining these aspects into a single similarity metric (e.g., [5,52,53]). Following the rationale of these studies, we include in our study the Sigmoid Cosine similarity (SCS), which augments the similarity between users who share a lot of common rated items, as shown in Equation (4). The range of the SCS metric is $[-1, 1]$, with values close to 1 indicating high similarity and values close to -1 designating high dissimilarity. Similarly to the CS metric, when ratings are non-negative only, the range of the metric is modified to $[0, 1]$.

$$\mathrm{SCS}(\mathrm{U1}, \mathrm{U2}) = \mathrm{CS}(\mathrm{U1}, \mathrm{U2}) \cdot \frac{1}{1 + e^{\frac{-|\mathrm{I(U1)} \cap \mathrm{I(U2)}|}{2}}} \tag{4}$$

Effectively, the SCS metric computes a high similarity value for users U1 and U2 when these users (a) rate the same items similarly (represented by the first term in Equation (4)) and (b) have rated many items in common (represented by the second term in Equation (4)). The second term of the equation has been used as listed in the corresponding "Sigmoid PCC" metric in [5,52,53].

The reason the corresponding Sigmoid PS will not be tested is that, as will be shown in the experiments in Section 4, the PS (and therefore the Sigmoid PS) presents a major prediction coverage issue and, therefore, is excluded from the detailed analysis. This issue will be analyzed in the experimental section.

When all of the similarities between each pair of users have been computed, the set of NNs is determined for each user, which typically includes the K users with the highest similarity to them (KNN). The value of K is set by the recommender system's administrator and is usually in the range of hundreds to a few thousand.

Lastly, the rating predictions are produced using a rating prediction formula. This formula synthesizes the rating values that only the user's NNs have given to the item for which the prediction is being produced, with each NN's similarity with the user (computed in the first phase) playing the role of the weight importance of each NN's rating. The most common rating prediction formulas in CF are the weighted average (wa) and mean-centered (mc), given in Equations (5) and (6), respectively [5,54,55]. In both of these equations, the similarity between the target user U1 and a near neighbor U2 of U1 is denoted as 'sim(U1, U2)' and corresponds to the application of any similarity formula discussed above.

$$\mathrm{Pwa}_{\mathrm{U1,i}} = \frac{\sum_{\mathrm{U2} \in \mathrm{NN}_{\mathrm{U1}}} \mathrm{sim}(\mathrm{U1}, \mathrm{U2}) \cdot (\mathrm{r}_{\mathrm{U2,i}})}{\sum_{\mathrm{U2} \in \mathrm{NN}_{\mathrm{U1}}} \mathrm{sim}(\mathrm{U1}, \mathrm{U2})} \tag{5}$$

$$\mathrm{Pmc}_{\mathrm{U1,i}} = \overline{\mathrm{r_{U1}}} + \frac{\sum_{\mathrm{U2} \in \mathrm{NN}_{\mathrm{U1}}} \mathrm{sim}(\mathrm{U1}, \mathrm{U2}) \cdot (\mathrm{r}_{\mathrm{U2,i}} - \overline{\mathrm{r_{U2}}})}{\sum_{\mathrm{U2} \in \mathrm{NN}_{\mathrm{U1}}} \mathrm{sim}(\mathrm{U1}, \mathrm{U2})} \tag{6}$$

We observe that the mc approach (Pmc) can be easily derived from the wa approach (Pwa), with each user's rating values normalized by their average rating value, following the same rationale as the PS similarity metric and producing a quantification of how much more or less the target user U1 would like the particular item compared to their average rating. Then, this delta value is added to the target user's mean rating to calculate the final rating prediction value. In the experiments of this work, all 4 similarity metrics and 2 prediction formulas will be used to ensure generalizability.

4. Experimental Settings and Evaluation Results

In this section, we present the experimental evaluation of the settings of the CF rating prediction presented above with respect to recommendation success. In particular, we evaluate (a) the four user similarity metrics (JS, PS, CS, and SCS), (b) the number of NNs stored and utilized for each user, and (c) the two rating prediction formulas (wa and mc)

in terms of prediction coverage (i.e., the percentage of cases for which a prediction can be generated) and prediction accuracy (using as evaluation metrics the MAE, the RMSE, and the F1 measure, which includes precision and recall metrics). Regarding the F1 measure, we follow the approach where the recommendation for a user includes all items for which the prediction value is placed within the upper 30% of the rating scale [56–58].

4.1. Experimental Settings

Our experimental evaluation uses 10 sparse CF datasets. As described in the Introduction, these are the datasets with a density typically less than 1%. The datasets used in our work, along with their attributes, are summarized in Table 2. Regarding the Epinions and LibraryThing datasets, only the rating information was considered, excluding the social relations data.

Table 2. The datasets involved in our experiments, along with their basic attributes.

Dataset Name	Density Level	Density	Attributes
CiaoDVD [1]	sparse	0.12%	2 K users, 14.7 K items, 36 K ratings
Amazon Digital Music [2]	sparse	0.33%	5.5 K users, 3.6 K items, 65 K ratings
Yahoo Movies [3]	sparse	0.24%	7.6 K users, 12 K items, 221 K ratings
Amazon Videogames [2]	very sparse	0.033%	95 K users, 26 K items, 815 K ratings
Amazon Industrial and Scientific [2]	very sparse	0.031%	51 K users, 26 K items, 413 K ratings
Epinions [2]	very sparse	0.014%	22 K users, 296 K items, 922 K ratings
BookCrossing [4]	very sparse	0.014%	22.8 K users, 319 K items, 1 M ratings
Amazon Office Products [2]	extremely sparse	0.009%	224 K users, 87 K items, 1.8 M ratings
Amazon Toys and Games [2]	extremely sparse	0.006%	432 K users, 162 K items, 3.9 M ratings
LibraryThing [2]	extremely sparse	0.005%	70 K users, 385 K items, 1.39 M ratings

[1] https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.htm (accessed on 29 January 2026). [2] https://cseweb.ucsd.edu/~jmcauley/datasets.html (accessed on 29 January 2026). [3] https://gist.github.com/tolleiv/3784342 (accessed on 29 January 2026). [4] https://www.kaggle.com/datasets/somnambwl/bookcrossing-dataset (accessed on 29 January 2026).

As we observe in Table 2, we use datasets from different sources and with different sizes (from a few thousand to a few million ratings). The selected datasets span several product fields (movies, books, video games, toys, etc.) to ensure that the evaluation is not biased by the item domain. Furthermore, in order to increase the accuracy of our experiments, we divide the datasets tested into three categories based on dataset density:

- Sparse: this category includes datasets with densities in the range of [0.1%, 1%);
- Very sparse: this category includes datasets with densities in the range of [0.01%, 0.1%);
- Extremely sparse: this category includes datasets with densities < 0.01%.

In order to quantify the outcome of our experiments, we apply the 5-fold cross-validation technique, where each dataset is split into five equally sized subsets; each time, four of them (80% of the total dataset) are used for training ratings and the remaining one (20% of the total dataset) is used for testing ratings. Then, the CF system computes user similarities using the train portion of the dataset and attempts to predict the rating value of each of the ratings in the test portion of the dataset [5,59]. Subsequently, performance metrics (coverage, accuracy) are computed for each of the five folds and averaged to produce the final metrics for the specific dataset.

The code for the experiments was implemented in the C programming language, using specialized indexes to increase computation speed. These indexes included (a) user rating lists sorted on the item id, (b) hash tables for pre-computed user similarities, and (c) per-item lists referencing users that have evaluated them. Measurements were obtained by running the code on a Dell PowerEdge M910 (Dell Inc., Round Rock, TX, USA, 2013) with four Intel(R) Xeon(R) CPU E7-4830 processors with sixteen execution cores each, totaling

64 execution cores. The machine was equipped with 256 GB of memory. All algorithms implemented were memory-based.

The following subsection presents the experimental results per density level.

4.2. Evaluation Results of Determining the Optimal Settings for Rating Prediction Accuracy in Very Sparse CF Datasets

In this subsection, we present the evaluation results to determine the optimal settings for rating prediction accuracy in sparse CF datasets. More specifically, for each of the three dataset density categories (sparse, very sparse, and extremely sparse), we initially present the rating prediction coverage in relation to the similarity metric used and the value of parameter K considered.

4.2.1. Evaluation Results in Sparse Datasets

Figure 1 depicts the rating prediction coverage percentage in relation to the similarity metric used and the value of parameter K considered for the first sparse dataset, i.e., the CiaoDVD dataset. We observe that the PS metric has very low coverage for every value of K tested. This is because there is a significant number of users in the CiaoDVD dataset (measured at 13%) who have the same values for all of the ratings they have submitted (e.g., all items are evaluated with 5/5). As we can see, in the definition of the PS metric (Equation (3)), when a user's ratings are all equal, the value of the metric cannot be computed because the denominator of the formula is zero. Practically, this means a CF system is neither able to produce a prediction for this user (because their NNs cannot be determined) nor can it use this user as an NN for another user (because the similarity between the users cannot be computed).

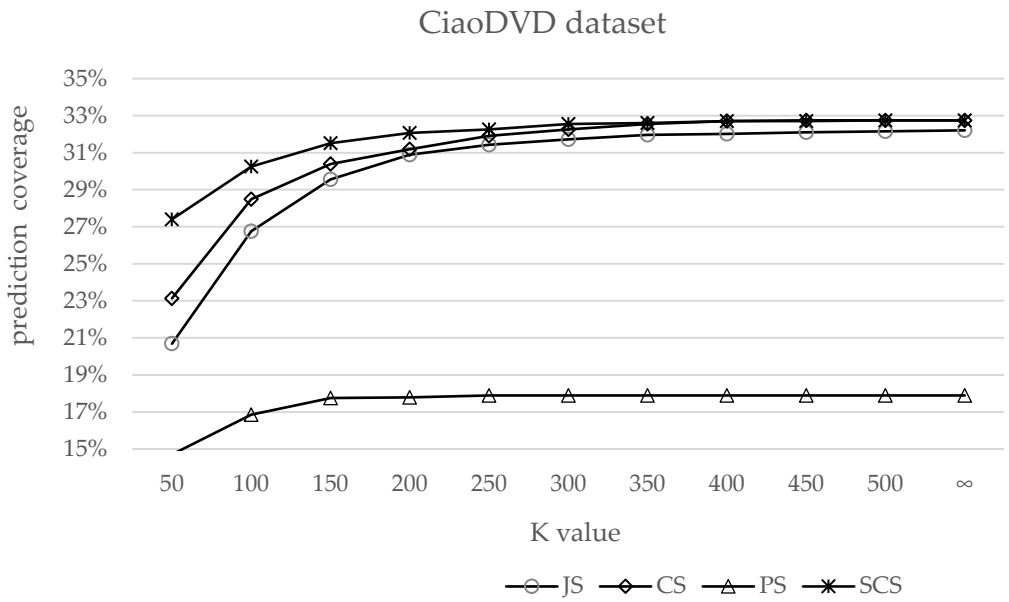

Figure 1. Rating prediction coverage for various values of K for all four metrics tested for the CiaoDVD dataset.

In Figure 1, we can also observe that for each similarity metric, as the value of K increases, prediction coverage increases until a similarity metric-specific maximum is reached. Each similarity metric converges to this maximum at a different rate and, additionally, the maximum is reached for a different value of K. Considering that prediction coverage is of topmost importance when the dataset is very sparse, in this paper, we will explore the range of parameter K values where the coverage is at least 95% of the maximum coverage.

Based on this rationale, the accepted settings are JS with K $\geq$ 250, CS with K $\geq$ 200, and SCS with K $\geq$ 150.

Figure 2 depicts the prediction MAE and F1 for all settings considered acceptable in terms of prediction coverage. We observe that the setting achieves the highest prediction accuracy when using the SCS similarity metric, setting K = 250, and selecting the MC prediction formula (from now on, for brevity, it will be referred to as "SCS-250-MC"). More specifically, when this setting is selected, the MAE equals 0.728, the RMSE is 1.020, and the F1 metric is 0.878, while the prediction coverage decrease is found to be 1.5%. Near-optimal settings are considered when setting K = 200 and K = 300 under the same similarity metric and prediction formula. More generally, the SCS similarity metric in combination with the mean-centered rating prediction computation formula achieves the highest F1 values and the lowest MAE values. On the other hand, the Jaccard similarity metric in combination with the weighted sum rating prediction computation formula yields high MAE and low F1 values. In Figure 2, we can also notice that the mean-centered-based prediction computation exhibits, on average, higher F1 measures and lower MAE metrics.

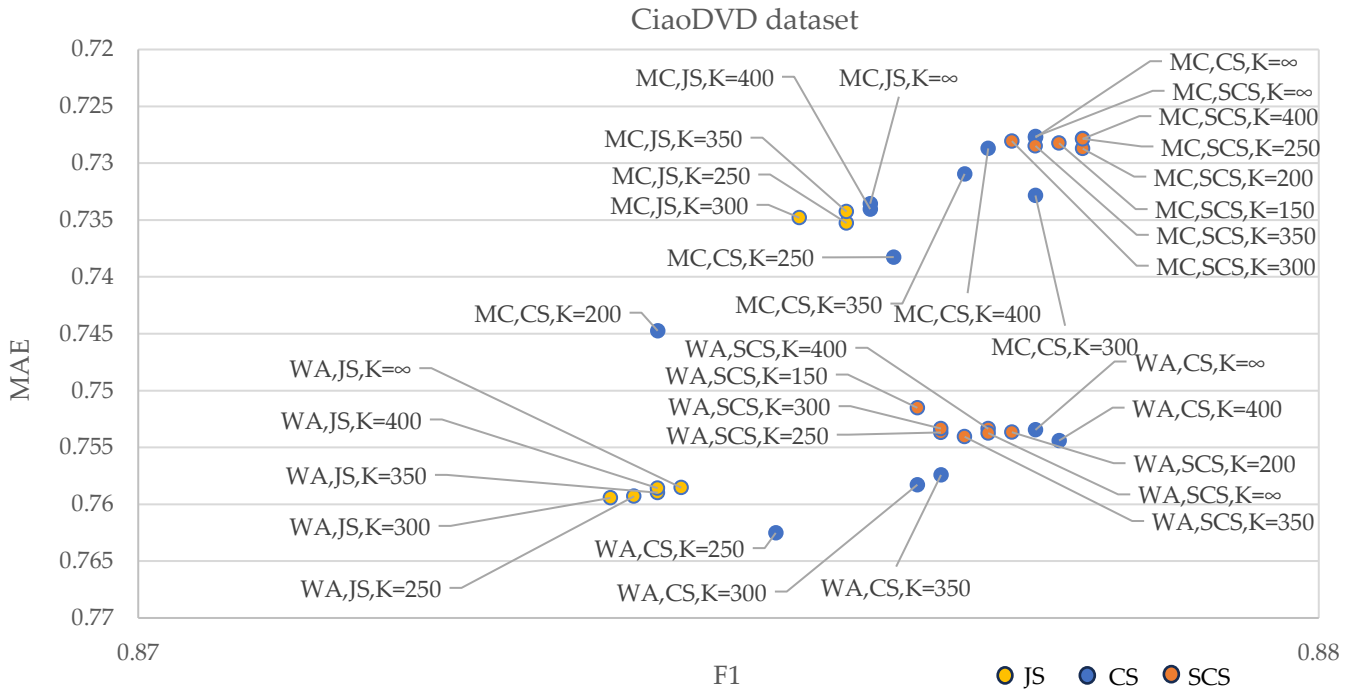

Figure 2. MAE and F1 for various values of K, both prediction formulation methods, and three similarity metrics for the CiaoDVD dataset.

When the Amazon Digital Music dataset is used, the optimal setting has been found to be the SCS-350-MC. When this setting is selected, the MAE equals 0.719, the RMSE is 1.032, and the F1 metric is 0.873, while the prediction coverage decrease is only 0.8%. Near-optimal settings are considered when setting K = 200, K = 250, and K = 300 under the same similarity metric and prediction formula, the JS-400-MC, as well as the CS-500-MC.

When the Yahoo Movies dataset is used, the optimal setting has been found to be the SCS-300-MC. When this setting is selected, the MAE equals 0.728, the RMSE is 1.029, and the F1 metric is 0.879, while the prediction coverage decrease is 2.5%. Near-optimal settings are considered when setting K = 200, K = 250, and K = 350 under the same similarity metric and prediction formula.

Overall, based on our experiments, when using sparse datasets (with density in the range of [0.1%, 1%)), the optimal settings that achieve the best prediction results are the SCS similarity metric with the MC prediction formula. Considering the optimal number

of parameter K, it was found to be in the range of [200–350]. From the results presented above, we can conclude that in sparse CF datasets, if we use the SCS similarity metric with the MC prediction formula, we need around 250 NNs to yield very good prediction results (coverage and accuracy), regardless of the other attributes (number of users and number of items) of the datasets.

4.2.2. Evaluation Results in Very Sparse Datasets

Figure 3 depicts the rating prediction coverage percentage in relation to the similarity metric used and the value of parameter K considered for the first very sparse dataset, i.e., the Amazon Videogames dataset. We observe that the PS metric has, again, very low coverage for every value of K tested.

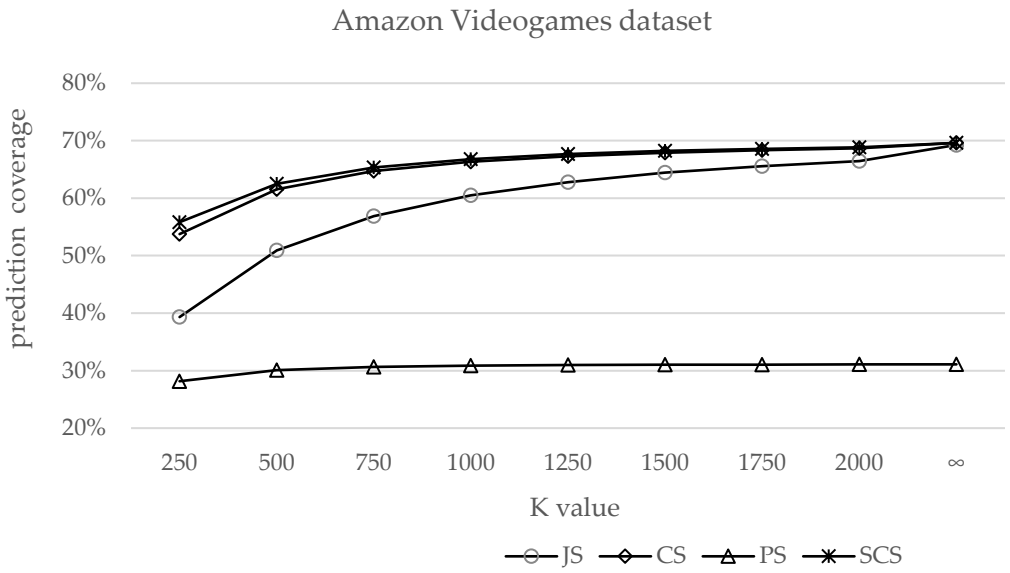

Figure 3. Rating prediction coverage for various values of K for all four metrics tested for the Amazon Videogames dataset.

We also observe that the maximum value that the prediction coverage can take is 69.6%. Assuming, again, that an acceptable coverage loss is 5%, in relative terms, any setting with prediction coverage above 66.1% is considered to be acceptable. Based on this threshold, the accepted settings are JS with $K \geq 2000$, CS with $K \geq 1000$, and SCS with $K \geq 1000$. Note that in the Amazon Videogames dataset the coverage achieved is higher than the coverage observed for the CiaoDVD dataset. While this appears to be counter-intuitive, as the Amazon Videogames dataset is more sparse than CiaoDVD, this is owing to the fact that the user-to-item ratio for the CiaoDVD dataset is approximately 2:15, while for the Amazon Videogames dataset the corresponding ratio is approximately 4:1. This means that when considering the Amazon Videogames dataset, it is more probable to find users that have commonly ranked items and, hence, personalized recommendations can be formulated for a higher percentage of the user base.

Figure 4 depicts the prediction MAE and F1 for the settings that achieved the highest prediction accuracy for each combination of similarity metric, KNN number, and prediction formula. We observe that the setting achieving the highest prediction accuracy is the SCS-1000-MC. More specifically, when this setting is selected, the MAE equals 0.733, the RMSE is 1.03, the F1 metric is 0.876, and the prediction coverage decrease is found to be 4.1%. Near-optimal settings are considered when setting K = 1250, under the same similarity metric and prediction formula, when the CS is used with K = 2000, and when the CS and JS are used with K = 1500, both cases with the MC prediction formula. Figure 4

also concurs with the findings of Figure 2, according to which the mean-centered-based prediction computation yields better results (higher F1, lower MAE) compared to the weighted averaging approach.

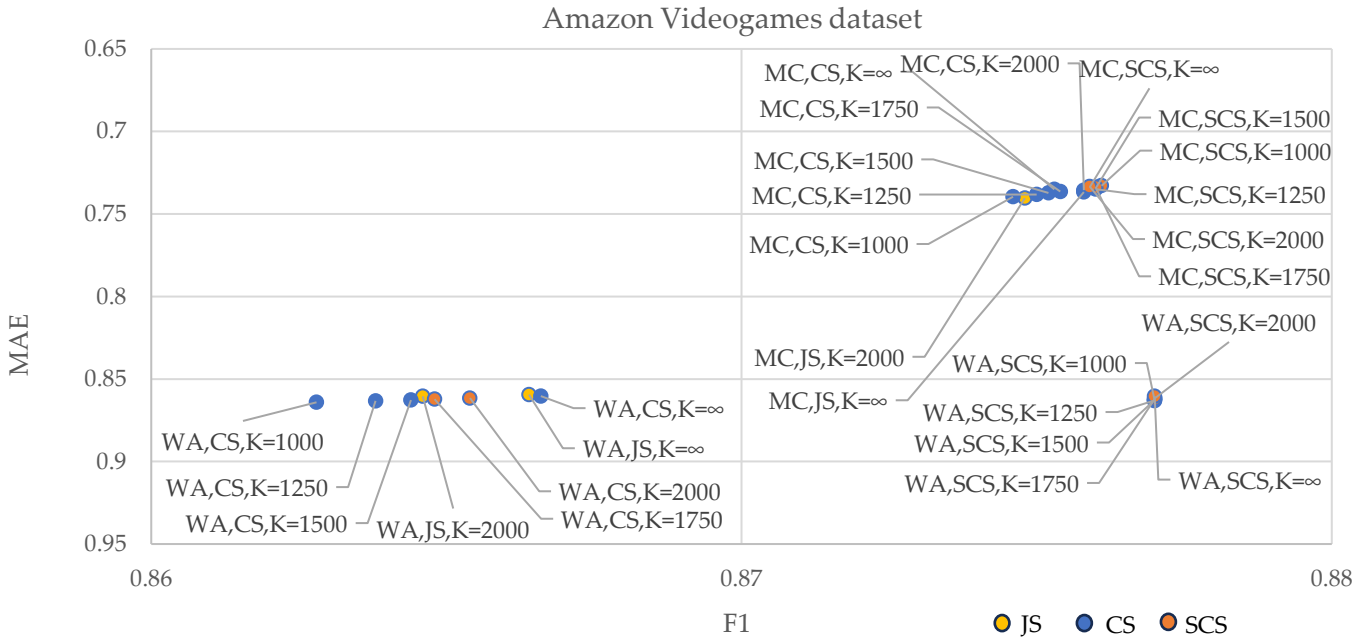

Figure 4. MAE and F1 for various values of K, both prediction formulation methods, and three similarity metrics for the Amazon Videogames dataset.

When the Amazon Industrial and Scientific dataset is used, the optimal setting has been found to be the SCS-750-MC. When this setting is selected, the MAE equals 0.760, the RMSE is 1.23, and the F1 metric is 0.885, while the prediction coverage decrease is 1.8%. Near-optimal settings are considered when setting K = 1000, under the same similarity metric and prediction formula, and when the CS and JS are also used with K = 1000, both cases with the MC prediction formula.

When the Epinions dataset is used, the optimal settings have been found to be when the SCS similarity is used with K = 1250 and selecting the MC prediction formula. When selecting this setting, the MAE equals 0.853, the RMSE is 1.12, and the F1 metric is 0.827, while the prediction coverage decrease is 3.4%. Near-optimal settings are considered when setting K = 1000 and K = 1500 under the same similarity metric and prediction formula.

When the BookCrossing dataset is used, the optimal setting has been found to be the SCS-1000-MC. When this setting is selected, the MAE equals 1.291, the RMSE is 1.593, and the F1 metric is 0.424, while the prediction coverage decrease is 1.5%. Near-optimal settings are considered when setting K = 1250, under the same similarity metric and prediction formula, and when the CS is used with K = 1500 and K = 2000 with the same prediction formula.

Overall, based on our experiments, when using very sparse datasets (with density in the range of [0.01%, 0.1%)), the optimal settings that achieve the best prediction results are the SCS similarity metric with the MC prediction formula. Considering the optimal value of parameter K, it was found to be in the range of [1000–1250]. In other words, based on our experiments, in very sparse CF datasets, if we use the SCS similarity metric with the MC prediction formula, we need around 1000 NNs to yield very good prediction results (coverage and accuracy), regardless of the other attributes (number of users and number of items) of the datasets.

4.2.3. Evaluation Results in Extremely Sparse Datasets

Figure 5 depicts the rating prediction coverage percentage in relation to the similarity metric used and the value of parameter K considered for the first very sparse dataset, i.e., the Amazon Office Products dataset. We observe that the PS metric has, again, very low coverage for every value of K tested.

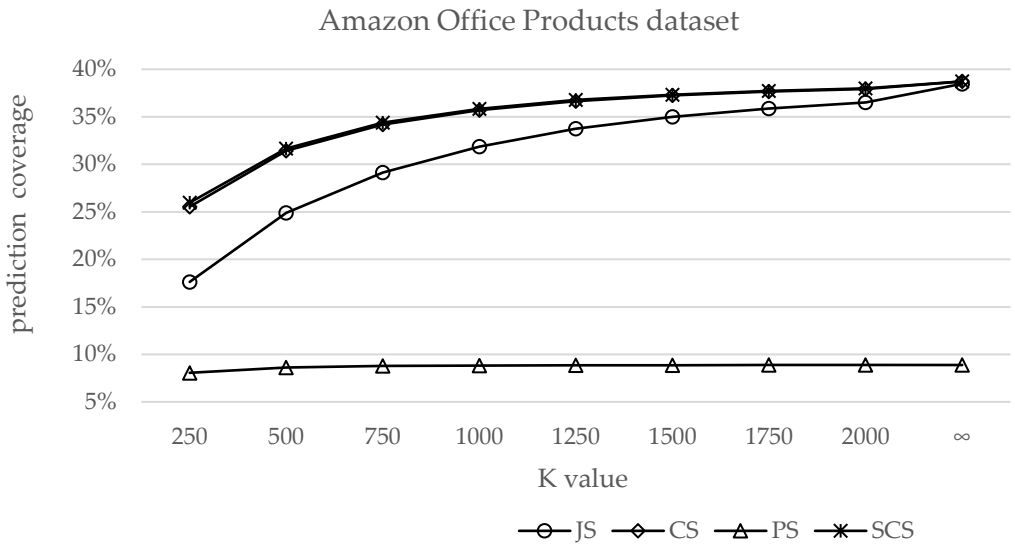

Figure 5. Rating prediction coverage for various values of K for all four metrics tested for the Amazon Office Products dataset.

We also observe that the maximum value that the prediction coverage can take is 38.7%. Assuming, again, that an acceptable coverage loss is 5%, in relative terms, any setting with prediction coverage above 36.8% is considered acceptable. Based on this threshold, the accepted settings are CS with $K \geq 1250$ and SCS also with $K \geq 1250$.

Figure 6 depicts the prediction MAE and F1 for the settings that achieved the highest prediction accuracy for each combination of similarity metric, KNN number, and prediction formula. In this dataset, we observe two settings that share optimal prediction accuracy. These are the SCS-1250-MC and the CS-1750-MC. More specifically, when these settings are selected, the MAE equals 0.628, the RMSE is 0.891, the F1 metric is 0.912, and the prediction coverage decrease is found to be 3.6% and 2.8%, respectively. Near-optimal settings are found to be the SCS-1500-MC and the CS-1500-MC. In Figure 6, we can also observe that all qualifying weighted-average-based configurations gather around a specific area (F1: 0.907, MAE: 0.68) with small variations regarding mainly the F1 measure, while all qualifying mean-centered-based configurations gather around the point (F1: 0.912, MAE: 0.63), again with small variations regarding mainly the F1 measure. Considering this finding, selecting the setting K = 1250 with a mean-centered prediction formulation and, preferably, the SCS similarity metric can offer high precision while confining the space and time overhead due to the need to store and maintain a high number of near neighbors.

When the Toys and Games dataset is used, the optimal settings found to achieve the highest prediction accuracy are the SCS-1500-MC and the CS-1500-MC. When these settings are selected, the MAE equals 0.652, the RMSE is 1.08, and the F1 metric is 0.905, while the prediction coverage decrease is 3.7%. Near-optimal settings are considered when setting K = 1750 and K = 1250 under the same similarity metrics and prediction formula. In this dataset, it is also observed that all qualifying weighted-average-based configurations gather around a single point (F1: 0.899, MAE: 0.706), while all qualifying mean-centered-based configurations gather around the point (F1: 0.905, MAE: 0.653), with marginal differences

between them. However, the K = 1250 configuration does not fulfill the inclusion rule (less than 5% drop in coverage compared to the top-performing configuration). Therefore, the value of K must be set to 1500 or above.

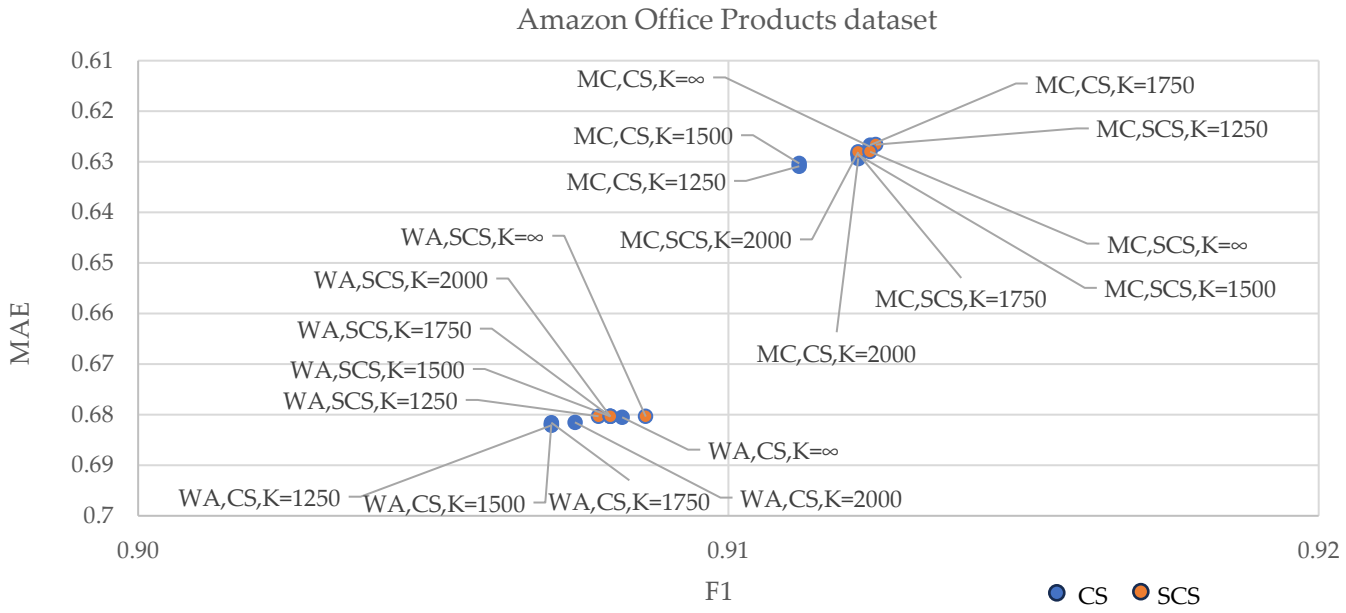

Figure 6. MAE and F1 for various values of K, both prediction formulation methods and two similarity metrics for the Amazon Office Products dataset.

When the LibraryThing dataset is used, the optimal settings have been found to be the SCS-1500-MC and the SCS-1750-MC. When selecting these settings, the MAE equals 0.776, the RMSE is 1.04, and the F1 metric is 0.814, while the prediction coverage decrease is found to be 0.6% and 0.3%, respectively. Near-optimal settings are achieved when setting K = 2000 under the same similarity metric and prediction formula. Similarly to the previous two cases, all qualifying weighted-average-based configurations gather around a single point (F1: 0.804, MAE: 0.82), and all qualifying mean-centered configurations gather around a different single point (F1: 0.813, MAE: 0.776). The minimum value of K that ensures an acceptable coverage drop (less than 5%) is 1000. However, using a higher value (K = 1250 or K = 1500) further enhances coverage.

Overall, based on our experiments, when using extremely sparse datasets (with density in the range of [0.001%, 0.01%)), the optimal settings that achieve the best prediction results are the SCS similarity metric with the MC prediction formula. Considering the optimal number of the value of parameter K, it was found to be in the range of [1250–1750]. In other words, based on our experiments, in extremely sparse CF datasets, if we use the SCS similarity metric with the MC prediction formula, we need around 1500 NNs to yield very good prediction results (coverage and accuracy), regardless of the other attributes (number of users and number of items) of the datasets.

4.2.4. Experimental Procedure and Complexity Analysis

In this subsection, we elaborate on the experimental procedure, conducting also a complexity analysis. Complexity analysis aims to provide insight into the feasibility and the computational cost of the method for large datasets, while implementation details may be used by researchers and practitioners for implementing optimized versions of CF systems.

The first phase of each experiment concerns the computation of similarities between users. The procedure followed in the first phase is as follows:

1. Each user's ratings are stored in a separate (per user) hash map. In this process, the sum of the user's rating values is computed if needed, to be finally divided by the count of the user's ratings to produce the average rating for the specific user (for the Pearson similarity metric only). Additionally, for each rating (userId, itemId, value), the user id is inserted in a hash map for item itemId. This hash map is used in the rating prediction phase. The complexity of this step is $O(|D|)$, where D is the dataset.
2. Subsequently, the process considers all user pairs, which are in the magnitude of u^2, where u is the number of users. For each pair of users, U1 and U2, the hash map containing the ratings of U1 is iterated upon, and for each of U1's ratings, the hash map containing the ratings of U2 is queried using the item of U1's rating as a key. If a matching rating is found, then the relevant computations dictated by the similarity metric are performed. Denoting the average number of ratings per user as r_M and considering that a hash map lookup has a complexity of O(1), the overall complexity of this step is $O(u^2 \times r_M)$.

Note that because all similarity metrics considered are symmetric, the process can be optimized by computing only the similarity for the ordered pair (U1, U2), skipping the computation of the similarity for the ordered pair (U2, U1). Additionally, the computations for different user pairs can be performed in parallel; this potential is exploited in the implementation of the proposed method, splitting the computations among 60 of the available cores (4 cores are set aside to serve operating system and housekeeping purposes). Parallel processing is conducted using a multithreaded model to allow for data sharing among threads (only a single copy of the rating matrix is stored, and this copy is accessed by all threads). While these two optimizations do not reduce the overall complexity, they contribute to decreasing the time needed to conduct similarity computation.

3. Finally, the NN set for each user is created. For each user U, his/her potential NNs (i.e., other users that have common ratings with U) are first gathered, and the list is then sorted by decreasing similarity value. The first-K elements of the sorted list are inserted in a hash map for user U. Assuming that the average number of potential NNs per user is k_M, the overall complexity of this step is $O(u^2 + u \times k_M \times \log(k_M) + u)$, with the first term accounting for the creation of the potential NNs list, the second term accounting for the sorting of the potential NNs lists, and the third term accounting for the creation of the per-user hash maps. Because term u^2 dominates term u, the complexity formula can be reduced to $O(u^2 + u \times k_M \times \log(k_M))$.

Steps 1–3 constitute the initialization phase of the experimental procedure, which is executed once to bootstrap data structures and perform user similarity and NN set computations. According to the analysis presented above, the complexity of this phase is

$$O(|D| + u^2 \times r_M + u^2 + u \times k_M \times \log(k_M))$$

Because the term $u^2 \times r_M$ dominates u^2, the latter term can be eliminated, and therefore the complexity formula can be reduced to

$$O(|D| + u^2 \times r_M + u \times k_M \times \log(k_M))$$

The second phase of each experiment concerns the creation of the predictions. To create a prediction for the rating that user U1 would give for item I1, the users that have rated item I1 are extracted from the relevant hash map created for item I1 (c.f. step 1 of phase 1), and for each of these users the hash map containing user U1's NNs is examined to determine if the rating of this user should be accounted for; if the user is found in U1's NNs, the relevant calculations are performed. Because hash lookups and arithmetic calculations

under both rating prediction formulas have a complexity of O(1), the complexity of this phase is $O(r_I)$, where r_I is the mean number of ratings per item.

5. Discussion

As demonstrated in the results presented in the previous section, conclusions can be drawn in regard to settings that achieve optimal results in CF rating prediction when very sparse datasets are used.

Firstly, regarding the similarity metric, there seems to be a clear advantage of the Sigmoid Cosine similarity metric in all cases/datasets tested. When using this similarity metric, the coverage decrease is acceptable (with the appropriate choice of the value of variable K, it is less than 5%, in relative terms, in every case tested), while the rating prediction accuracy achieves its optimal scores in every dataset tested. The success of this metric is justified because it considers both the number of common items that two users have rated (like the Jaccard Similarity) and the rating values these users have given to these items (like the Cosine similarity), ensuring that cases where rating similarity among users may be coincidental (i.e., cases where users have very few common ratings) are assigned a lower weight than cases where rating similarity among users has a high probability of being systematic. Furthermore, this metric can be effectively applied for users that have set the same rating values for all items they have rated, while the Pearson Similarity metric, commonly used in CF, cannot compute similarity values in these cases. This is an inherent issue of the Pearson coefficient [60], and it is worth noting that this phenomenon (i.e., cases where users have entered the same value for all of their ratings) occurs frequently in (very) sparse datasets (as mentioned in Section 4.2.1).

Secondly, regarding the prediction formula, again there seems to be a clear advantage of the mean-centered formula in all cases/datasets tested over the weighted average formula. More specifically, the performance of the weighted average formula lags behind in all performance metrics compared to the mean-centered formula in each of the settings and for all datasets tested. The success of this prediction formula is justified because it compensates for the divergent rating practices employed by different users, i.e., the fact that some users assign ratings more strictly while others are more lenient.

An important finding of this work is the number of NNs that should be stored for each user (i.e., the value of parameter K) and considered for their rating predictions. On one hand, this number should be small so that it would not require significant memory and cause long delays in generating predictions. On the other hand, it should be large enough to ensure acceptable prediction coverage. Based on the experiments of this work, presented in the previous section, the following was found:

- When using sparse datasets (with density in the range of [0.1%, 1%)): the optimal value of parameter K was found to be in the range of [200–350];
- When using very sparse datasets (with density in the range of [0.01%, 0.1%)): the optimal value of parameter K was found to be in the range of [1000–1250];
- When using extremely sparse datasets (with density in the range of [0.001%, 0.01%)): the optimal value of parameter K was found to be in the range of [1250–1750].

Figure 7 depicts a heatmap-inspired visualization of the effect of parameter K on the performance of the algorithm for datasets of different densities.

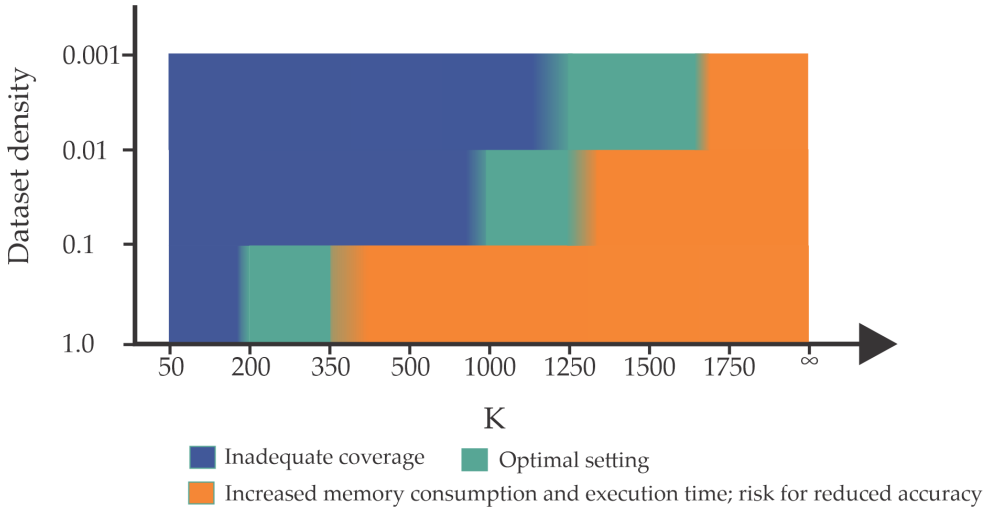

Figure 7. Heatmap visualization of the effect of the K parameter on the performance of the algorithm for datasets of different densities. Color gradients are used to denote that the transition between areas is gradual rather than abrupt.

Recapitulating, based on the findings of our experiments, if we use the SCS similarity metric with the MC prediction formula, we need around 250 NNs, 1000 NNs, and 1500 NNs, when using sparse, very sparse, and extremely sparse datasets, respectively, to achieve satisfactory prediction results (coverage and accuracy), regardless of the other attributes (number of users and number of items) of the datasets. This increase in the number of NNs that a CF recommender system needs in each density category level to produce satisfactory rating predictions is justified because the sparser a dataset, the more NNs each user U needs in order for the CF system to be able to find users who are similar to U (i.e., the similarity between them can be calculated) but who have also evaluated the item for which the prediction is being generated.

As mentioned in the Introduction, the target of this study was to determine the settings that ensure satisfactory rating prediction results not only in terms of accuracy, as shown in the previous paragraph, but also in terms of (i) prediction coverage, (ii) memory size required to store the NNs, and (iii) prediction generation time.

Figure 8 depicts the prediction coverage reduction for all 10 datasets tested in this work when using the proposed settings (determined in the previous section) compared to the (extreme) case where the CF system stores all of the NNs for every user (i.e., 40% more NNs on average, as discussed below). In Figure 7, we can observe that the average prediction coverage reduction is 2.35%, while this reduction does not exceed 4.1% in any case.

Figure 9 depicts the reduction of the space/memory size required to store the NNs, comparing the proposed settings with the case where the CF system stores all possible NNs for a user. In this figure, we observe that the average prediction memory reduction for storing the NNs is 41%, spanning 12% for the CiaoDVD dataset to 94% for the Yahoo Movies dataset. Details regarding memory savings for the optimal selection of the K parameter under the SCS similarity metric can be found in Table A1 in Appendix A.

Lastly, Figure 10 depicts the prediction computation time reduction in all 10 datasets tested in this work when using the proposed settings (determined in the previous section), compared to the case where the CF system stores all of the NNs for a user (the code used was optimized, and hash indexes were utilized to improve performance). In this figure, we observe that the average prediction time reduction is 41%, ranging from 5.1% for the CiaoDVD dataset to 92% for the Yahoo Movies dataset. Combining Figures 8 and 9, we can observe that time savings are approximately proportional to savings in the number of

NNs. For the case of CiaoDVD in particular, time savings lag behind the savings in the number of NNs; this is attributed to the low number of users in the dataset, which leads to near neighborhoods of small cardinality whose processing time is low, and therefore per-prediction standard overheads (such as locating the target user record and extracting metadata from it) play a more visible role in the overall prediction computation time.

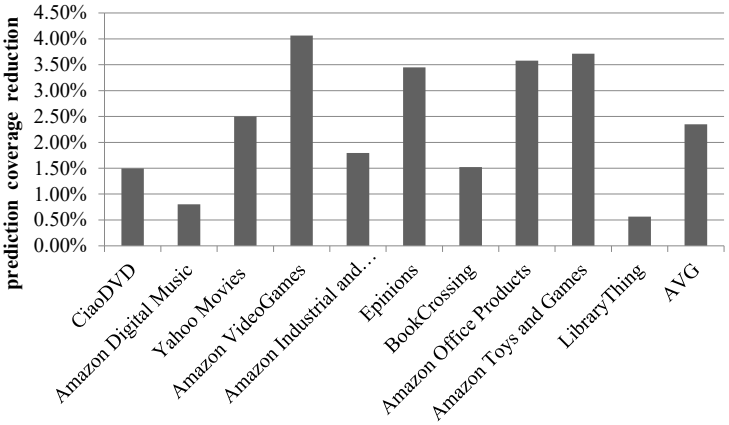

Figure 8. Rating prediction coverage reduction when selecting the proposed optimal settings.

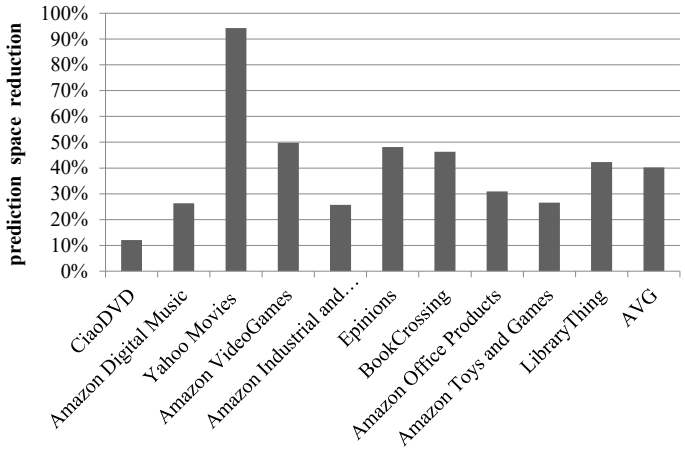

Figure 9. Rating prediction space reduction when selecting the proposed optimal settings.

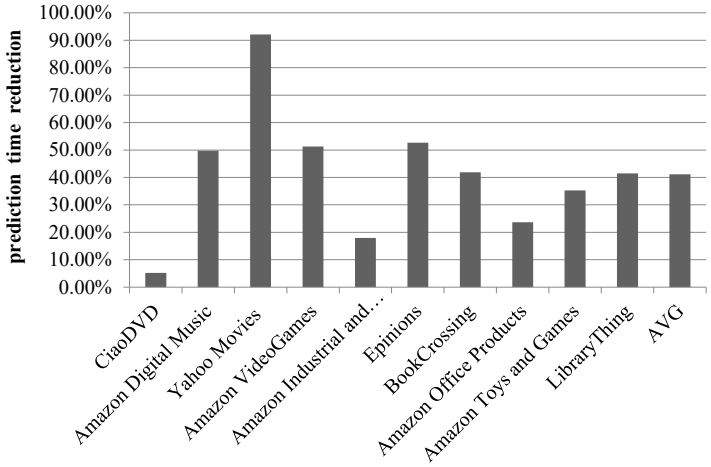

Figure 10. Rating prediction time reduction when selecting the proposed optimal settings.

Details regarding the absolute time measurements (in msec) obtained for each dataset when (a) using the unpruned near neighborhood of each user, (b) using the optimal value for K, and (c) using the minimum acceptable value for K, as well as the relative speedups, can be found in Table A2 in Appendix A.

Overall, the proposed settings are found to achieve considerable prediction space and time reductions, while the corresponding prediction coverage reduction remains very low and is considered acceptable. For example, when the proposed settings are applied to the Yahoo Movies dataset, the memory reduction is 95% (from 940 MB to 45 MB for all users' NNs), the prediction time reduction is 92% (from 35 s to 2.8 s), and the prediction coverage reduction is 2.5% (from 97.8% to 95.4%).

While the findings in this survey can provide guidance for researchers and practitioners regarding the choice of NN set size, similarity metrics, and rating prediction computation formulas, the following aspects have to be taken into account. Firstly, while the study results indicate an optimal range of NN set size per dataset class, the behavior of specific datasets may deviate from these findings; taking this into account, a tuning experiment using a sample of the target dataset may be useful to establish the concrete value of the parameter used for production sites. Moreover, the study considers three widely used similarity metrics, while metrics such as the Spearman rank correlation, the Adjusted Rand Index, and the Chebyshev distance, which have been found to have good performance in the context of sparse datasets [5], are not considered. The examination of the performance of these metrics is considered part of our future work. Notably, the methodology followed in the paper can readily accommodate both individual similarity metrics and similarity metric combinations. Indeed, the modular design of the methodology encapsulates similarity metric computation as a distinct module, which can be substituted by any other module that offers the same functionality, i.e., the initialization of the module (which may encompass actions such as the computation of per-user rating averages) and the computation of similarities between a pair of users. Similarity metric combinations can be accommodated in the same way that the SCS metric was implemented in the context of the present study, again adhering to the modular architecture followed by the methodology.

6. Conclusions and Future Work

The study presented in this paper aimed to find the optimal parameters for rating prediction generation in CF recommender systems when using very sparse datasets. More specifically, this paper evaluates four user similarity metrics, two rating prediction formulas, and varying values of parameter K in the top-K NN selection method to produce predictions for 10 (very) sparse datasets. To ensure the reliability of the results, these datasets were derived from diverse sources, and they are commonly used in CF research. No additional information was considered in any phase of the rating prediction procedure (either about their users or items), and the datasets were categorized into three levels of sparsity.

Based on the experimental results, regarding the similarity metric, there seems to be a clear advantage of the Sigmoid Cosine similarity metric in all datasets tested. Moreover, regarding the rating prediction formula, there seems to be a clear advantage of the mean-centered formula in all datasets tested, again.

The most interesting finding of this work is the number of NNs that should be stored for each user and considered for rating predictions (i.e., the value of parameter K). Based on the experiments of this work, when using sparse datasets (with density in the range of [0.1%, 1%)), the optimal value for parameter K was found to be around 250. When using very sparse datasets (with density in the range of [0.01%, 0.1%)), the optimal value for parameter K was found to be around 1000. Lastly, when using extremely sparse datasets

(with density in the range of [0.001%, 0.01%)), the optimal value for parameter K was found to be around 1500.

These values of K, in combination with the Sigmoid Cosine similarity and the mean-centered prediction formula, ensure satisfactory prediction results (coverage and accuracy), independently of any other attributes (e.g., number of users, number of items, and item domain) of the datasets. More specifically, when comparing the proposed (optimal) settings with the extreme case where the CF system stores all of the NNs for each user, the average prediction coverage reduction across all 10 datasets is only 2.35%, while this reduction does not exceed 4.1% in any dataset. Furthermore, both the average prediction memory reduction for storing the NNs and the average prediction formulation time reduction were measured at 41%.

Regarding our future work, we plan to include more similarity metrics (basic and hybrid), especially newer ones introduced in recent work. Furthermore, we are planning to modify the Pearson Similarity metric so that it does not exclude users who have given the same value for all their ratings, which leads to a drastic reduction in its coverage in sparse datasets, making it practically inapplicable.

Author Contributions: Conceptualization, S.-A.L., J.N., D.M., C.V. and D.S.; methodology, S.-A.L., J.N., D.M., C.V. and D.S.; software, S.-A.L., J.N., D.M., C.V. and D.S.; validation, S.-A.L., J.N., D.M., C.V. and D.S.; formal analysis, S.-A.L., J.N., D.M., C.V. and D.S.; investigation, S.-A.L., J.N., D.M., C.V. and D.S.; resources, S.-A.L., J.N., D.M., C.V. and D.S.; data curation, S.-A.L., J.N., D.M., C.V. and D.S.; writing—original draft preparation, S.-A.L., J.N. and D.M.; writing—review and editing, D.M., C.V. and D.S.; visualization, S.-A.L., J.N., D.M., C.V. and D.S.; supervision, D.M., C.V. and D.S.; project administration, D.M., C.V. and D.S.; funding acquisition, D.M., C.V. and D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Datasets available to the public were used in this work. These data can be found here: https://cseweb.ucsd.edu/~jmcauley/datasets.html (accessed on 29 January 2026), https://gist.github.com/tolleiv/3784342 (accessed on 29 January 2026), https://www.kaggle.com/datasets/somnambwl/bookcrossing-dataset (accessed on 29 January 2026), and https://guoguibing.github.io/librec/datasets.html (accessed on 29 January 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Table A1 provides details on the memory savings for the optimal selection of the K parameter under the SCS similarity metric; only the SCS similarity metric is reported, as it has been found to provide high coverage and increased accuracy using the minimum number of NNs. For each dataset, column "Total NNs" corresponds to the sum of the cardinalities of all users' unpruned neighborhoods, i.e.,

$$Total\ NNs = \sum_{i=1}^{N_u} |NN(u_i)| \tag{A1}$$

where N_u is the number of users in the dataset and $NN(u_i)$ is the unpruned near neighborhood of the ith user. Column "Optimal K" lists the optimal value for parameter K for the specific dataset under the SCS metric, as presented in Section 4, while column "NNs@Optimal K" lists the sum of the cardinalities of all users' near neighborhoods, where

each near neighborhood is pruned to contain the top "optimal K" nodes (or less, if the unpruned near neighborhood contained less than K neighbors). Formally,

$$NNs@Optimal\ K = \sum_{i=1}^{N_u} \min(|NN(u_i)|,\ Optimal\ K) \tag{A2}$$

Finally, column "Min K" lists the dataset-specific value of K for which coverage is considered acceptable (at least 95% of the maximum coverage), and column "NNs@Min K" lists the sum of the cardinalities of all users' near neighborhoods, where each near neighborhood is pruned to contain the top "Min K" nodes, or, formally,

$$NNs@Min\ K = \sum_{i=1}^{N_u} \min(|NN(u_i)|,\ Min\ K) \tag{A3}$$

Table A1 also includes column "Mem req@Optimal K", which lists the amount of memory utilized when running the experiment with parameter K set to the optimal value of the specific dataset. This column provides insight into the hardware resources needed to accommodate a memory-based recommender system operating a similarly sized dataset. For conciseness, the respective measurements for near neighborhoods that are (a) unpruned or (b) pruned to Min K are not listed; they scale approximately analogously to the savings or increments of the population of the respective near neighborhoods (the magnitudes do not form a perfect analogy because the rating matrix maintains the same size, regardless of the pruning level).

Table A1. Memory requirements per dataset for different settings of K.

Dataset	Total NNs	Optimal K	NNs@Optimal K	Mem Req@ Optimal K	NN Count Reduction vs. Total NNs	Min K	NNs@Min K	Optimal NN Count Increase vs. Min
CiaoDVD	188 K	250	166 K	3 MB	12%	150	133 K	24%
Amazon Digital Music	1.6 M	350	1.2 M	27 MB	26%	100	0.5 M	133%
Yahoo Movies	40.2 M	300	2.3 M	52 MB	94%	250	1.9 M	19%
Amazon Videogames	134.1 M	1000	67.3 M	1540 MB	49%	1000	67.3 M	0%
Amazon Industr. and Scient.	22.6 M	750	16.8 M	383 MB	25%	750	16.8 M	0%
Epinions	31.2 M	1250	16.2 M	371 MB	48%	750	11.1 M	46%
BookCrossing	19.7 M	1000	10.6 M	242 MB	46%	500	6.9 M	52%
Amazon Office Products	227.1 M	1250	157.0 M	3593 MB	30%	1250	157.0 M	0%
Amazon Toys and Games	434.1 M	1500	318.8 M	7297 MB	26%	1500	318.8 M	0%
LibraryThing	44.1 M	1500	25.5 M	583 MB	42%	1000	20.3 M	26%

In Table A1, we can observe that memory savings by pruning the near neighborhoods to maintain at most K NNs, where K is set to the dataset-specific optimal value, range from 12% for the CiaoDVD dataset to 94% for the Yahoo Movies dataset. In all cases, the amount of memory required to host all of the structures of the algorithm (rating matrix, near neighborhoods, hash tables, etc.) is less than 8 GB, which can be readily accommodated using contemporary hardware. Please note that while the memory requirements when setting K to the optimal value appear in most cases increased compared to those when setting K to the minimum acceptable value, setting K to the optimal value yields improvements in both coverage and prediction accuracy.

Table A2 provides details on the absolute time measurements (in msec) obtained for each dataset when (a) using the unpruned near neighborhood of each user, (b) using the optimal value for K, and (c) using the minimum acceptable value for K, as well as the relative speedups. We can observe that each rating prediction is performed in all cases in

less than 162 msec. Moreover, it is worth noting that the rating prediction process is fully parallelizable by assigning each rating prediction to a different thread/execution core.

Table A2. Execution time per dataset for different settings of K.

Dataset	Time (Unpruned)	Optimal K	Time@Optimal K	Speedup vs. Unpruned	Min K	Time@Min K	Optimal K Slowdown vs. Min K
CiaoDVD	0.071	250	0.067	5.1%	150	0.05	34.0%
Amazon Digital Music	2.6	350	1.31	48.7%	100	0.66	49.7%
Yahoo Movies	12.9	300	1.10	91.5%	250	0.83	32.5%
Amazon Videogames	69.0	1000	33.4	51.6%	1000	33.4	0.0%
Amazon Industr. and Scient.	45.7	750	37.5	17.9%	750	37.5	0.0%
Epinions	67.1	1250	32	52.6%	750	24.4	30.2%
BookCrossing	64.1	1000	37.3	41.9%	500	25.3	47.4%
Amazon Office Products	190.2	1250	145.2	23.7%	1250	145.2	0.0%
Amazon Toys and Games	251.0	1500	162.5	35.3%	1500	162	0.0%
LibraryThing	146.8	1500	86.0	41.5%	1000	59.9	43.5%

References

1. Saifudin, I.; Widiyaningtyas, T. Systematic Literature Review on Recommender System: Approach, Problem, Evaluation Techniques, Datasets. *IEEE Access* **2024**, *12*, 19827–19847. [CrossRef]
2. Vuong Nguyen, L. Classifications, evaluation metrics, datasets, and domains in recommendation services: A survey. *Int. J. Hybrid Intell. Syst.* **2024**, *20*, 85–100. [CrossRef]
3. Singh, R.; Dwivedi, P.; Kant, V. Comparative analysis of collaborative filtering techniques for the multi-criteria recommender systems. *Multimed. Tools Appl.* **2024**, *83*, 64551–64571. [CrossRef]
4. Chowdhury, P.; Sinha, B.B. Evaluating the Effectiveness of Collaborative Filtering Similarity Measures: A Comprehensive Review. *Procedia Comput. Sci.* **2024**, *235*, 2641–2650. [CrossRef]
5. Sgardelis, K.; Margaris, D.; Spiliotopoulos, D.; Vassilakis, C. An evaluation review of user similarity metrics in sparse collaborative filtering datasets. *Int. J. Data Sci. Anal.* **2025**, *20*, 6665–6693. [CrossRef]
6. Abdalla, H.I.; Amer, Y.A.; Nguyen, L.; Amer, A.A.; Al-Maqaleh, B.M. Numerical Similarity Measures Versus Jaccard for Collaborative Filtering. In *Proceedings of the 9th International Conference on Advanced Intelligent Systems and Informatics 2023*; Hassanien, A., Rizk, R.Y., Pamucar, D., Darwish, A., Chang, K.-C., Eds.; Lecture Notes on Data Engineering and Communications Technologies; Springer Nature: Cham, Switzerland, 2023; Volume 184, pp. 221–229. ISBN 978-3-031-43246-0.
7. Sun, Y.; Liu, Q. Collaborative filtering recommendation based on K-nearest neighbor and non-negative matrix factorization algorithm. *J. Supercomput.* **2025**, *81*, 79. [CrossRef]
8. Saleh, A.M.; Taqa, A.Y. A Proposed Item-Based Collaborative Filtering Model for e-Book Recommendation with a Weighted KNN. In *Proceedings of 4th International Conference on Mathematical Modeling and Computational Science*; Pal, S., Rocha, Á., Eds.; Lecture Notes in Networks and Systems; Springer Nature: Cham, Switzerland, 2025; Volume 1398, pp. 124–136. ISBN 978-3-031-90997-9.
9. Wang, Y.; Zhou, Y.; Chen, T.; Zhang, J.; Yang, W.; Huang, Z. Hybrid Collaborative Filtering-Based API Recommendation. In Proceedings of the 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS), Hainan, China, 6–10 December 2021; pp. 906–914.
10. Nudrat, S.; Khan, H.U.; Iqbal, S.; Talha, M.M.; Alarfaj, F.K.; Almusallam, N. Users' Rating Predictions Using Collaborating Filtering Based on Users and Items Similarity Measures. *Comput. Intell. Neurosci.* **2022**, *2022*, 2347641. [CrossRef] [PubMed]
11. Rajesh, D.B.; Kumar, A. Collaborative filtering models an experimental and detailed comparative study. *Sci. Rep.* **2025**, *15*, 31667. [CrossRef]
12. Ramakrishna, M.T.; Venkatesan, V.K.; Bhardwaj, R.; Bhatia, S.; Rahmani, M.K.I.; Lashari, S.A.; Alabdali, A.M. HCoF: Hybrid Collaborative Filtering Using Social and Semantic Suggestions for Friend Recommendation. *Electronics* **2023**, *12*, 1365. [CrossRef]
13. Anwar, T.; Uma, V.; Hussain, M.I.; Pantula, M. Collaborative filtering and kNN based recommendation to overcome cold start and sparsity issues: A comparative analysis. *Multimed. Tools Appl.* **2022**, *81*, 35693–35711. [CrossRef]
14. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
15. Rendle, S. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol.* **2012**, *3*, 1–22. [CrossRef]
16. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.-S. Neural Collaborative Filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3 April 2017; International World Wide Web Conferences Steering Committee: Perth, Australia; pp. 173–182.

17. Wen, H.; Ding, G.; Liu, C.; Wang, J. Matrix Factorization Meets Cosine Similarity: Addressing Sparsity Problem in Collaborative Filtering Recommender System. In *Web Technologies and Applications*; Chen, L., Jia, Y., Sellis, T., Liu, G., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2014; Volume 8709, pp. 306–317. ISBN 978-3-319-11115-5.
18. Margaris, D.; Spiliotopoulos, D.; Karagiorgos, G.; Vassilakis, C. An Algorithm for Density Enrichment of Sparse Collaborative Filtering Datasets Using Robust Predictions as Derived Ratings. *Algorithms* **2020**, *13*, 174. [CrossRef]
19. Gunathilaka, T.M.A.U.; Manage, P.D.; Zhang, J.; Li, Y.; Kelly, W. Addressing sparse data challenges in recommendation systems: A systematic review of rating estimation using sparse rating data and profile enrichment techniques. *Intell. Syst. Appl.* **2025**, *25*, 200474. [CrossRef]
20. Kolahkaj, M.; Harounabadi, A.; Nikravanshalmani, A.; Chinipardaz, R. Incorporating multidimensional information into dynamic recommendation process to cope with cold start and data sparsity problems. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 9535–9554. [CrossRef]
21. Rodpysh, K.V.; Mirabedini, S.J.; Banirostam, T. Employing singular value decomposition and similarity criteria for alleviating cold start and sparse data in context-aware recommender systems. *Electron. Commer. Res.* **2023**, *23*, 681–707. [CrossRef]
22. Lyu, Z.; Yang, M.; Li, H. Multi-view group representation learning for location-aware group recommendation. *Inf. Sci.* **2021**, *580*, 495–509. [CrossRef]
23. Choi, S.-M.; Lee, D.; Jang, K.; Park, C.; Lee, S. Improving Data Sparsity in Recommender Systems Using Matrix Regeneration with Item Features. *Mathematics* **2023**, *11*, 292. [CrossRef]
24. Natarajan, S.; Vairavasundaram, S.; Natarajan, S.; Gandomi, A.H. Resolving data sparsity and cold start problem in collaborative filtering recommender system using Linked Open Data. *Expert Syst. Appl.* **2020**, *149*, 113248. [CrossRef]
25. Elahi, M.; Khosh Kholgh, D.; Kiarostami, M.S.; Oussalah, M.; Saghari, S. Hybrid recommendation by incorporating the sentiment of product reviews. *Inf. Sci.* **2023**, *625*, 738–756. [CrossRef]
26. Duan, R.; Jiang, C.; Jain, H.K. Combining review-based collaborative filtering and matrix factorization: A solution to rating's sparsity problem. *Decis. Support Syst.* **2022**, *156*, 113748. [CrossRef]
27. Koohi, H.; Kiani, K. Two new collaborative filtering approaches to solve the sparsity problem. *Clust. Comput.* **2021**, *24*, 753–765. [CrossRef]
28. Sun, K.; Wang, Y.; He, M.; Zhou, H.; Zhang, S. Neighbor interaction-based personalised transfer for cross-domain recommendation. *Connect. Sci.* **2023**, *35*, 2263664. [CrossRef]
29. Dai, F.; Gu, X.; Wang, Z.; Li, B.; Qian, M.; Wang, W. Attention-Based Multi-view Feature Fusion for Cross-Domain Recommendation. In *Artificial Neural Networks and Machine Learning, Proceedings of the International Conference on Artificial Neural Networks 2021*; Farkaš, I., Masulli, P., Otte, S., Wermter, S., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2021; Volume 12891, pp. 204–216. ISBN 978-3-030-86361-6.
30. Yang, W.; Guo, S.H.; Zhang, C.J. A novel rating style mining method to improve collaborative filtering algorithm. *J. Phys. Conf. Ser.* **2019**, *1187*, 052101. [CrossRef]
31. Shetty, A.M.; Manjaiah, D.H.; Aljunid, M.F.; Yogesh, K.M. Comparative Analysis of Memory-Based Collaborative Filtering and Deep Learning Models for Resolving Cold Start and Data Sparsity Issues in E-commerce Recommender Systems. In Proceedings of the 2024 IEEE 3rd World Conference on Applied Intelligence and Computing (AIC), Gwalior, India, 27 July 2024; pp. 81–87.
32. Zarzour, H.; Jararweh, Y.; Al-Sharif, Z.A. An Effective Model-Based Trust Collaborative Filtering for Explainable Recommendations. In Proceedings of the 2020 11th International Conference on Information and Communication Systems (ICICS), Irbid, Jordan, 7–9 April 2020; pp. 238–242.
33. Polatidis, N.; Georgiadis, C.K. A multi-level collaborative filtering method that improves recommendations. *Expert Syst. Appl.* **2016**, *48*, 100–110. [CrossRef]
34. Iftikhar, A.; Ghazanfar, M.A.; Ayub, M.; Mehmood, Z.; Maqsood, M. An Improved Product Recommendation Method for Collaborative Filtering. *IEEE Access* **2020**, *8*, 123841–123857. [CrossRef]
35. Wu, W.; Qi, Z.; Tian, J.; Wang, B.; Tang, M.; Liu, X. An Enhanced Latent Factor Recommendation Approach for Sparse Datasets of E-Commerce Platforms. *Systems* **2025**, *13*, 372. [CrossRef]
36. Fkih, F. Similarity measures for Collaborative Filtering-based Recommender Systems: Review and experimental comparison. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 7645–7669. [CrossRef]
37. Gonzalez, A.; Ortega, F.; Perez-Lopez, D.; Alonso, S. Bias and Unfairness of Collaborative Filtering Based Recommender Systems in MovieLens Dataset. *IEEE Access* **2022**, *10*, 68429–68439. [CrossRef]
38. Zhang, S.; Chen, S.; Yu, X.; Mei, S. Research on collaborative filtering algorithm based on improved K-means algorithm for user attribute rating and co-rating. *Sci. Rep.* **2025**, *15*, 19600. [CrossRef]
39. AL-Ghuribi, S.; Mohd Noah, S.A.; Mohammed, M. An experimental study on the performance of collaborative filtering based on user reviews for large-scale datasets. *PeerJ Comput. Sci.* **2023**, *9*, e1525. [CrossRef] [PubMed]
40. Margaris, D.; Kobusinska, A.; Spiliotopoulos, D.; Vassilakis, C. An Adaptive Social Network-Aware Collaborative Filtering Algorithm for Improved Rating Prediction Accuracy. *IEEE Access* **2020**, *8*, 68301–68310. [CrossRef]

41. Mishra, K.N.; Mishra, A.; Barwal, P.N.; Lal, R.K. Natural Language Processing and Machine Learning-Based Solution of Cold Start Problem Using Collaborative Filtering Approach. *Electronics* **2024**, *13*, 4331. [CrossRef]
42. Karabila, I.; Darraz, N.; El-Ansari, A.; Alami, N.; El Mallahi, M. Enhancing Collaborative Filtering-Based Recommender System Using Sentiment Analysis. *Future Internet* **2023**, *15*, 235. [CrossRef]
43. Mandal, S.; Maiti, A. Deep collaborative filtering with social promoter score-based user-item interaction: A new perspective in recommendation. *Appl. Intell.* **2021**, *51*, 7855–7880. [CrossRef]
44. Tran, T.T.; Snasel, V.; Nguyen, L.T. Combining Social Relations and Interaction Data in Recommender System With Graph Convolution Collaborative Filtering. *IEEE Access* **2023**, *11*, 139759–139770. [CrossRef]
45. Santhosh, N.M.; Cheriyan, J.; Sindhu, M. An Intelligent Exploratory Approach for Product Recommendation Using Collaborative Filtering. In Proceedings of the 2021 2nd International Conference on Advances in Computing, Communication, Embedded and Secure Systems (ACCESS), Ernakulam, India, 2 September 2021; pp. 232–237.
46. Margaris, D.; Spiliotopoulos, D.; Vassilakis, C. Augmenting Black Sheep Neighbour Importance for Enhancing Rating Prediction Accuracy in Collaborative Filtering. *Appl. Sci.* **2021**, *11*, 8369. [CrossRef]
47. Margaris, D.; Vassilakis, C.; Spiliotopoulos, D.; Ougiaroglou, S. Rating Prediction Quality Enhancement in Low-Density Collaborative Filtering Datasets. *Big Data Cogn. Comput.* **2023**, *7*, 59. [CrossRef]
48. He, P.; Shi, J.; Ma, W.; Zheng, X. Broad collaborative filtering with adjusted cosine similarity by fusing matrix completion. *Appl. Soft Comput.* **2024**, *165*, 112075. [CrossRef]
49. Dharshan, V.; Hariharan, I.; Vimal Kumar, K. Enhanced Hybrid UBCF-IBCF Recommender Systems Using Pearson and Cosine Similarities for Improved Accuracy. In Proceedings of the 2024 International Conference on Sustainable Communication Networks and Application (ICSCNA), Theni, India, 11 December 2024; pp. 1037–1044.
50. Al-Hassan, M.; Abu-Salih, B.; Alshdaifat, E.; Aloqaily, A.; Rodan, A. An Improved Fusion-Based Semantic Similarity Measure for Effective Collaborative Filtering Recommendations. *Int. J. Comput. Intell. Syst.* **2024**, *17*, 45. [CrossRef]
51. Mana, S.C.; Sasipraba, T. Research on Cosine Similarity and Pearson Correlation Based Recommendation Models. *J. Phys. Conf. Ser.* **2021**, *1770*, 012014. [CrossRef]
52. Jain, G.; Mahara, T.; Tripathi, K.N. A Survey of Similarity Measures for Collaborative Filtering-Based Recommender System. In *Soft Computing: Theories and Applications*; Pant, M., Sharma, T.K., Verma, O.P., Singla, R., Sikander, A., Eds.; Advances in Intelligent Systems and Computing; Springer: Singapore, 2020; Volume 1053, pp. 343–352. ISBN 978-981-15-0750-2.
53. Wang, Y.; Deng, J.; Gao, J.; Zhang, P. A hybrid user similarity model for collaborative filtering. *Inf. Sci.* **2017**, *418–419*, 102–118. [CrossRef]
54. Hasan, M.; Nalagandla, R. An Ensemble Weighted User-Based Collaborative Filtering Recommender System. In Proceedings of the 2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings), Mt Pleasant, MI, USA, 7 September 2024; pp. 1–6.
55. Fkih, F. Enhancing item-based collaborative filtering by users' similarities injection and low-quality data handling. *Data Knowl. Eng.* **2023**, *144*, 102126. [CrossRef]
56. Felfernig, A.; Boratto, L.; Stettinger, M.; Tkalčič, M. Evaluating Group Recommender Systems. In *Group Recommender Systems*; SpringerBriefs in Electrical and Computer Engineering; Springer International Publishing: Cham, Switzerland, 2018; pp. 59–71. ISBN 978-3-319-75066-8.
57. Margaris, D.; Vassilakis, C.; Spiliotopoulos, D. What makes a review a reliable rating in recommender systems? *Inf. Process. Manag.* **2020**, *57*, 102304. [CrossRef]
58. AlEroud, A.; Karabatis, G. Using Contextual Information to Identify Cyber-Attacks. In *Information Fusion for Cyber-Security Analytics*; Alsmadi, I.M., Karabatis, G., Aleroud, A., Eds.; Studies in Computational Intelligence; Springer International Publishing: Cham, Switzerland, 2017; Volume 691, pp. 1–16. ISBN 978-3-319-44256-3.
59. Le, D.D.; Lauw, H.W. Collaborative Curating for Discovery and Expansion of Visual Clusters. In Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining, Virtual Event, 11 February 2022; pp. 544–552.
60. Sheugh, L.; Alizadeh, S.H. A note on pearson correlation coefficient as a metric of similarity in recommender system. In Proceedings of the 2015 AI & Robotics (IRANOPEN), Tehran, Iran, 12 April 2015; pp. 1–6.