applied sciences

MDPI

Article

Deep Matrix Factorization Approach for Collaborative Filtering Recommender Systems

Raúl Lara-Cabrera *,†, Ángel González-Prieto †and Fernando Ortega †

Departamento de Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, 28031 Madrid, Spain; angel.gonzalez.prieto@upm.es (Á.G.-P.); fernando.ortega@upm.es (F.O.)

* Correspondence: raul.lara@upm.es; Tel.: +34-91-0673650

† These authors contributed equally to this work.

Received: 18 June 2020; Accepted: 15 July 2020; Published: 17 July 2020

Abstract: Providing useful information to the users by recommending highly demanded products and services is a fundamental part of the business of many top tier companies. Recommender Systems make use of many sources of information to provide users with accurate predictions and novel recommendations of items. Here we propose, DeepMF, a novel collaborative filtering method that combines the Deep Learning paradigm with Matrix Factorization (MF) to improve the quality of both predictions and recommendations made to the user. Specifically, DeepMF performs successive refinements of a MF model with a layered architecture that uses the acquired knowledge in a layer as input for subsequent layers. Experimental results showed that the quality of both the predictions and recommendations of DeepMF overcome the baselines.

Keywords: deep learning; recommender systems; collaborative filtering; matrix factorization

1. Introduction

Presently, data processing has become a priority for the society. The amount of information we generate every day is on the rise, mainly due to the increase in the number of devices we use routinely as well as new patterns of interaction with technology that have been developed over the last few years. With such a high volume of real-time data, Machine Learning (ML) techniques became a key tool for being able to extract knowledge from data. At present, ML is one of the most active research field not only within computer science, but also in healthcare [1], sociology [2] and industry [3].

One of the most important applications of ML are Recommender System (RS) [4], which are techniques that make use of different sources of information for providing users with predictions and recommendations of items, adjusting factors such as accuracy, novelty, sparsity and stability in the recommendations [5,6]. Top tier companies such as Netflix, Spotify and Amazon, use RS provide useful information to the users by recommending highly demanded products and services. Although it is possible to find many kinds of RS in the state of the art, they all share a key component: The filtering algorithm at the core. According to the filtering algorithm used, RS might be categorized [7] into (a) Collaborative Filtering (CF), (b) demographic filtering, (c) content-based filtering and (d) hybrid filtering. Regarding CF, Matrix Factorization (MF) is one of its most widely used techniques [4,8]. It operates by decomposing the user-item interaction matrix into the product of two lower dimensional rectangular matrices as well as assigning different regularization weights to the latent factors in order to improve the prediction results [9].

A major recent breakthrough in the field of ML is the so-called Deep Learning (DL) technology, that has attracted much attention not only from the scientific community, but also from society in general. Oppositely to other ML techniques, which can only extract knowledge from the shallow of the data, typically based on statistical evidence, DL is able to detect and exploit the underlying multi-layer

structure of the data. Although DL is generally linked to artificial neural networks, the paradigm is more complex than that. As stated by LeCun et al. [10], "DL allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction". Since its inception, DL has positioned as a very top notch technology for uncovering tangled structures in high-dimensional data. In this way, it has been applied to a wide range of domains such as speech recognition [11], image recognition [12] or natural language processing [13], among others.

As might be expected, RS have not been immune to the boom in DL and it is possible to find several works in this direction. The most obvious interplay is to apply deep neural networks [14] for improving the predictions of the RSs. Indeed, they provide appropriate a posteriori biases to the input data type, exploiting any inherent structure within the data. Furthermore, deep neural networks are capable of modeling the non-linearity in data with nonlinear activations, which is one of the principal aims of modern RS, making it possible to capture complex interaction patterns between users and items.

Due to the aforementioned advantages, neural networks with deep architectures, i.e., many layers, have been used to create new RS. For instance, DeepFM [15] is an approach that models the interactions of high and low order features through deep neural networks and factorization machines, respectively. On the other hand, Autoencoder-Based Collaborative Filtering (ACF) [16] uses another well-known DL architecture, namely variational autoencoders, to build a RS model that decompose the partial observed ratings vectors in a similar way as one-hot encoding. In the same way, Bobadilla et al. [17] provide an innovative DL architecture to improve CF results by exploiting the potential of the reliability concept to raise predictions and recommendations quality by incorporating prediction errors in the DL layers. Regarding convolutional neural networks, He et al. [18] propose to use them in order to enhance CF by using the outer product instead of the customary dot product to model the interaction patterns of both users and items, as well as capturing the high-order correlations among embedding dimensions. In addition, Abavisani and Patel [19] proposed an artificial neural network that consists of a convolutional autoencoder along with a fully connected layer to learn robust deep features for classification. Along the same lines, Recurrent Neural networks have been used with RS [20–22], although under some restrictions (i.e., session-based recommendations), due to the features of this kind of deep architecture, which are suitable for sequential data processing. Even Generative Adversarial Networks (GANs) have been used to generate negative samples for a memory network-based streaming RS [23] and enhancing the Bayesian personalized ranking with adversarial training [24].

Another promising line of application of DL is to pull apart the neural networks and to apply the underlying philosophy to extend the usual MF algorithm to get deep factorizations. This approach is a common trend in several machine learning contexts, but it has never been applied to RS. For instance, Le Magoarou and Gribonval proposed FAuST [25], a deep factorization method of a matrix into a product of many sparse factors that can be achieved through hierarchical refinements, in the spirit of deep networks. FAuST has proven to be very successful in image processing, data compression and inverse linear problems but, as we will see, it fails to provide a competitive alternative when applied to RS. In a similar vein, Trigeorgis et al. [26] devised a semi-supervised learning method based on Non-negative Matrix Factorization, which was applied to the problem of face clustering, while Guo and Zhang [27] used a similar approach to perform dimension reduction and pattern recognition. A multilevel decomposition method used to derive a feature representation for speech recognition is presented in [28], which outperforms existing features for various speech recognition tasks.

In this paper, we propose a novel method for incorporating DL to CF-based RSs. For that purpose, we present a new CF method that combines the DL paradigm with MF in order to improve the quality of predictions and recommendations. The approach presented here is based on the principles of DL, but unlike state-of-the-art works, the proposed method does not use deep neural network architectures.

Instead, it transforms the MF process into a layered process, in which each layer factors out the errors made in the previous layer.

Roughly speaking, the proposed model works as follows. Consider a matrix R that collects the known ratings of the users of the RS to the supervised items. Typically, the MF looks for a factorization of the form $R \approx P \cdot Q$ for some matrices P, Q of low rank. In this way, the matrix $E^1 = R - P \cdot Q$ measures the error attained with the MF. The key point is that, if we were able to exactly predict the error, then we could correct the output of our model by tuning it to take into account the expected error. To get this, our proposal is to deepen this factorization and to look for a factorization of the error $E^1 \approx P^1 \cdot Q^1$, which gives rise to a 'second order error' $E^2 = E^1 - P^1 \cdot Q^1$. This refinement of the expected error can be continued as many times as desired in order to achieve a better control of the error. In this way, as byproduct of the application of the DL process, we end up with a more accurate model that auto-corrects its predictions with the estimation of the expected error in the guess.

Summarizing, the proposed RS model follows the DL paradigm by performing successive refinements of a MF model with a layered architecture and letting it to use the acquired knowledge in one layer to be used as input for subsequent layers. The novelty of our work lies in using the concepts of DL, not for use with artificial neural networks, but to improve the MF itself.

The rest of the paper is structured as follows. Section 2 contains the material and methods used to formalize the proposed method, as well as the associated algorithm. Section 3 presents the results of the experimental evaluation conducted in order to measure the performance of the presented method. As we will see, the proposed algorithm consistently outperforms the previous MF methods that can be found in the literature. Finally, Section 4 discusses about these results and its impact on the RS research field.

2. Materials and Methods

A current trend in the field of CF is to improve the quality of the predictions by means of different techniques of MF, such as PMF [29], NMF [30], and SVD++ [31]. However, all these techniques rely on the same approach: they pose an optimization problem through a loss function that measures the divergence of the model of the expected behavior of users and items in a collaborative context from the actual behavior. Trying to break with this paradigm, in this paper we present a novel recommendation model that, using the MF paradigm, introduces the principles of DL to refine the output of the model through successive trainings. We named this new model Deep Matrix Factorization (DeepMF).

Figure 1 summarizes the operation of DeepMF. As we can observe, the model is initialized with the usual input of a CF-based RS: a matrix R that contains the ratings of the users to the items. Note that this matrix is sparse, since a user generally only votes a very small subset of the existing items. As in classical MF approaches, this matrix R, that will be also called $R = E^0$ in our context, is approximated by a dense matrix $\hat{E}^0$ of the form $\hat{E}^0 = P^0 \cdot Q^0$, where P^0 and Q^0 are matrices of small rank. This matrix $\hat{E}^0$ provides the predicted ratings at the first step.

At this point the DL begins. A new matrix $E^1 = R - \hat{E}^0$ is built by computing the attained errors between the original ratings R and the predicted ratings stored in $\hat{E}^0$. This new matrix is, again, approximated by a factorization into two new small rank matrices $\hat{E}^1 = P^1 \cdot Q^1$, which produces the errors at the second step $E^2 = E^1 - \hat{E}^1$. This process is repeated as many times as desired, by generating and factorizing successive error matrices $E^1, \ldots, E^N$. Presumably, this sequence of error matrices converges to zero, so we get preciser predictions as we add new layers to the model.

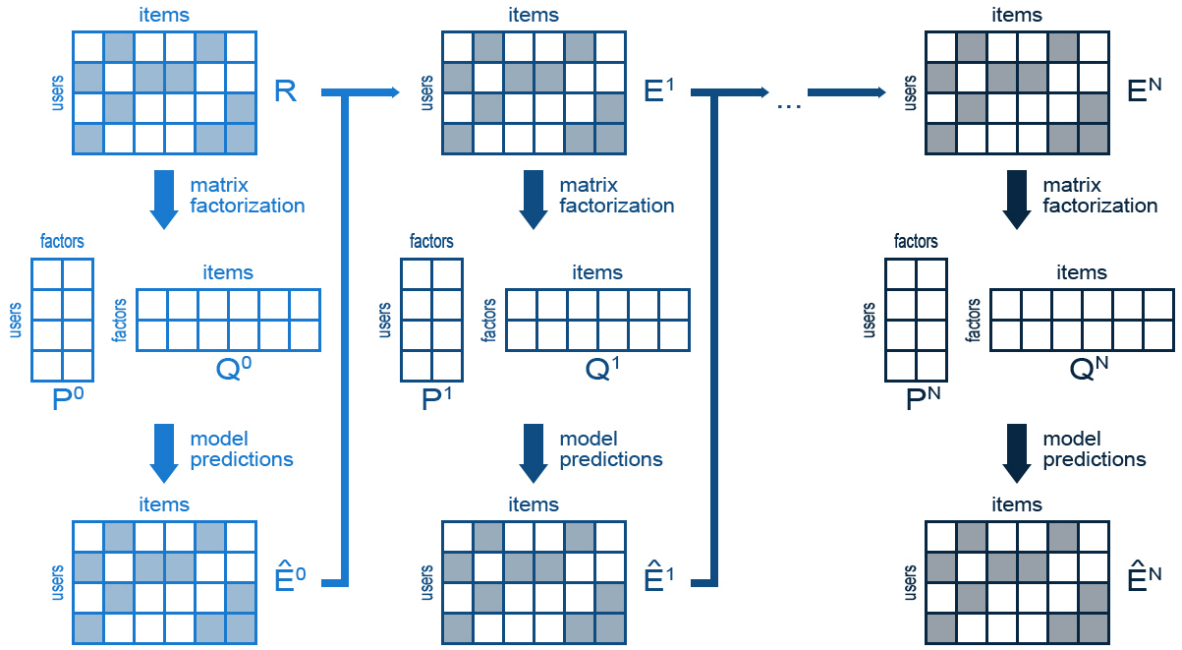


Figure 1. Illustrative explanation of proposed DeepMF method.

This method can be formalized as follows. Suppose that our CF-based RS is dealing with n users and m items, whose ratings are collected in a spare matrix $R = (R_{u,i}) \in \mathbb{R}^{n \times m}$, where $R_{u,i}$ is the rating that the user u gave to item i (typically, integer values between 1 and 5), or $R_{u,i} = \bullet$ if u has not rated the item i. We look for a factorization of $E^0 = R$ of the form $E^0 \approx P^0 \cdot Q^0$, where P^0 is a $n \times k^0$ matrix and Q^0 is a $k^0 \times m$ matrix. Here, k^0 is interpreted as the number of hidden factors we try to detect in the first step (typically k^0 is around 10) and Q^0, P^0 are seen as the projection/co-projection of the n users and m items into a k^0-dimensional latent space. These small rank matrices are learned such that the product $P^0 \cdot Q^0$ is a good approximation of the ratings matrix $E^0 = R$, that is

$$E^0 \approx \hat{E}^0 = P^0 \cdot Q^0 \tag{1}$$

in the usual euclidean distance.

By subtracting the approximation performed by $\hat{E}^0$ to the original matrix R we can obtain a new sparse matrix E^1 that contains the prediction error

$$E^1 = R - \hat{E}^0 = E^0 - P^0 \cdot Q^0. \tag{2}$$

Note that positive values in E^1 denotes that the prediction is too low and must be increased and negative values in E_1 denotes that the prediction is too high and must be decreased. As we stated before, the main contribution of the proposed method is its deep learning approach. This is achieved by performing a new factorization to the E^1 error matrix in such a way that

$$E^1 \approx \hat{E}^1 = P^1 \cdot Q^1. \tag{3}$$

The matrices P^1, Q^1 have orders $n \times k^1$ and $k^1 \times m$ for a certain number of latent factors k^1. Observe that we may take $k^1 \neq k^0$ in order to get a different level of resolution in the factorization.

In the general case, if we computed $s-1$ steps of the deep learning procedure, we compute the s-th matrix of errors as

$$E^s = E^{s-1} - \hat{E}^{s-1} = E^{s-1} - P^{s-1} \cdot Q^{s-1}, \tag{4}$$

and we look for a factorization into matrices P^s of order $n \times k^s$ and Q^s of order $k^s \times m$ such that, as much as possible in the euclidean norm,

$$E^s \approx \hat{E}^s = P^s \cdot Q^s. \tag{5}$$

Once the deep factorization process ends after N steps, the original rating matrix R can be reconstructed by adding the estimates of the errors as

$$\begin{aligned} R \approx \hat{R} &= \hat{E}^0 + E^1 = \hat{E}^0 + \hat{E}^1 + E^2 = \ldots \approx \hat{E}^0 + \hat{E}^1 + \hat{E}^2 + \ldots + \hat{E}^N \\ &= P^0 \cdot Q^0 + P^1 \cdot Q^1 + \cdots + P^N \cdot Q^N = \sum_{s=0}^{N} P^s \cdot Q^s. \end{aligned} \tag{6}$$

For any step $s = 0, \ldots, N$, the factorization $E^s \approx P^s \cdot Q^s$ is sought by the standard method of minimizing the euclidean distance between E^s and $P^s Q^s$ by gradient descent with regularization, as in PMF [29]. In this way, if we write $E^s = (e^s_{u,i})$, the rows of P^s are denoted by $p^s_1, p^s_2, \ldots, p^s_n$ and the columns of Q^s are denoted by $q^s_1, q^s_2, \ldots, q^s_m$, the loss function is given by

$$\begin{aligned} \mathcal{F}^s(p^s_u, q^s_i) &= ||E^s - P^s \cdot Q^s||^2 + \lambda^s \left(\sum_{u=1}^{n} ||p^s_u||^2 + \sum_{i=1}^{m} ||q^s_i||^2 \right) \\ &= \sum_{e^s_{u,i} \neq \bullet} \left(e^s_{u,i} - p^s_u \cdot q^s_i\right)^2 + \lambda^s \left(\sum_{u=1}^{n} ||p^s_u||^2 + \sum_{i=1}^{m} ||q^s_i||^2 \right), \end{aligned} \tag{7}$$

where λ^s is the regularization hyper-parameter of the s-th step to avoid overfitting.

The previous loss function $\mathcal{F}$ can be derived with respect to p^s_u and q^s_i resulting the following update rules for learning the model parameters

$$\begin{aligned} p^s_u &\leftarrow p^s_u + \gamma^s \left((e^s_{u,i} - p^s_u \cdot q^s_i) \cdot q^s_i - \lambda^s p^s_u \right), \\ q^s_i &\leftarrow q^s_i + \gamma^s \left((e^s_{u,i} - p^s_u \cdot q^s_i) \cdot p^s_u - \lambda^s q^s_i \right), \end{aligned} \tag{8}$$

where γ^s is the learning rate hyper-parameter of the s-th step to control the learning speed.

In this way, after finishing the nested factorization, the predicted ratings are collected in the matrix $\hat{R} = (\hat{R}_{u,i})$, where the predicted rating of the user u to the item i is given by

$$\hat{R}_{u,i} = \sum_{s=0}^{N} p^s_u \cdot q^s_i. \tag{9}$$

Note that the proposed method consists of successive repetitions of a MF process using the results of the previous MF as input. Therefore, it is possible to make an algorithmic implementation of the proposed method using a recursive approach. Algorithm 1 contains the pseudo-code of the recursive implementation of DeepMF. The algorithm receives as input the E matrix, which contains either the users' ratings to the items ($R = E^0$) for the first call or the errors of the previous factorization E^s for the successive ones. It also receives the model hyper-parameters: the number of latent factors on each step (K), the number of iterations of the gradient descent optimization on each step (T), the learning rates (Γ) and the regularizations (Λ).

Note that these hyper-parameters were stacked so that each of the factorizations performed uses different hyper-parameters. The hyper-parameters of the first factorization will be placed at the top of the stack, those of the second factorization in the next one and so on until the parameters of the deeper factorization, which will be placed at the bottom of the stack. This allows us to define the stopping criteria of the algorithm: as soon as a stack is empty the learning process will be finished. Similarly, the output of the algorithm will be a stack containing the pairs $\langle P, Q \rangle$ with the hidden factors of each of the factorizations carried out.

Algorithm 1: Pseudo-code of a recursive implementation of DeepMF.

Function `deep_mf(`E, K, T, Γ, Λ`)`**:**
  Initialize $P \leftarrow U(0,1)$ and $Q \leftarrow U(0,1)$
  $k, t, \gamma, \lambda \leftarrow pop(K, T, \Gamma, \Lambda)$
  repeat
    for *each user* $\boldsymbol{u}$ *and item* $\boldsymbol{i}$ *in* $\boldsymbol{E}$ *such as* $e_{u,i} \neq \bullet$ **do**
      for *each factor* $f \in \{1, \ldots, k\}$ **do**
        $error \leftarrow e_{u,i} - P_{u,f} \cdot Q_{i,f}$
        $P_{u,f} \leftarrow P_{u,f} + \gamma \left(error \cdot Q_{i,f} - \lambda P_{u,f} \right)$
        $Q_{i,f} \leftarrow Q_{i,f} + \gamma \left(error \cdot P_{u,f} - \lambda Q_{i,f} \right)$
      end
    end
  until *t iterations*
  if *is_empty(K)* **then**
    return new stack($\langle P, Q \rangle$)
  else
    $E' = E - P \cdot Q$
    params_stack = `deep_mf(`E', K, T, Γ, Λ`)`;
    return push($\langle P, Q \rangle$, params_stack)
  end
End Function

Algorithm 1 needs to be run in a single thread since the update of the P matrix requires the Q matrix and vice versa. However, this algorithm can easily be sped-up by using an Alternating Least Squares (ALS) approach. Algorithm 2 contains a parrallel implementation of DeepMF using an ALS approach. As can be observed, to perform a factorization, the matrices P and Q are updated t times according to the loss function gradient. This update is performed in two steps: first, the values of the Q matrix are fixed as constants allowing updating of the latent factors of each user (P_u) independently from the latent factors of the rest of the users, thus, they can be updated in parallel for each user. Second, the values of the P matrix are fixed as constants and the latent factors of each item Q_i are updated independently from the latent factors of the rest of the items, thus, they can be updated in parallel for each item. Although this parallel implementation needs to go through the rating set twice to update both the P and the Q matrices, its computation time is significantly less than that of Algorithm 1 when more than 2 execution threads are available. For example, in a dataset with 1 million ratings, the total computation time of each iteration will be the time required to update the Algorithm 2, the total computation time in a processor with 8 execution threads of each iteration will be the time required to update the matrices P and Q *2 * 1 million / 8 threads = 250,000 times*, a quarter of the total computation time required by Algorithm 1.

Algorithm 2: Pseudo-code of a parrallel recursive implementation of DeepMF using an ALS approach.

Function `deep_mf(`E, K, T, Γ, Λ`)`**:**
  Initialize $P \leftarrow U(0,1)$ and $Q \leftarrow U(0,1)$
  $k, t, \gamma, \lambda \leftarrow pop(K, T, \Gamma, \Lambda)$
  repeat
    `/* This loop can be executed in parallel for each user */`
1 **for** *each user* $\boldsymbol{u}$ *in* $\boldsymbol{E}$ **do**
2 **for** *each item* $\boldsymbol{i}$ *rated by the user* $\boldsymbol{u}$ ($e_{u,i} \neq \bullet$) **do**
3 **for** *each factor* $f \in \{1, \ldots, k\}$ **do**
4 $error \leftarrow e_{u,i} - P_{u,f} \cdot Q_{i,f}$
5 $P_{u,f} \leftarrow P_{u,f} + \gamma \left(error \cdot Q_{i,f} - \lambda P_{u,f} \right)$
6 **end**
7 **end**
8 **end**
9 `/* This loop can be executed in parallel for each item */`
10 **for** *each item* $\boldsymbol{i}$ *in* $\boldsymbol{E}$ **do**
11 **for** *each user* $\boldsymbol{u}$ *who has rated the item* $\boldsymbol{i}$ ($e_{u,i} \neq \bullet$) **do**
12 **for** *each factor* $f \in \{1, \ldots, k\}$ **do**
13 $error \leftarrow e_{u,i} - P_{u,f} \cdot Q_{i,f}$
14 $Q_{i,f} \leftarrow Q_{i,f} + \gamma \left(error \cdot P_{u,f} - \lambda Q_{i,f} \right)$
15 **end**
16 **end**
17 **end**
  until *t iterations*
  if *is_empty(K)* **then**
    return new stack($\langle P, Q \rangle$)
  else
    $E' = E - P \cdot Q$
    params_stack = `deep_mf(`E', K, T, Γ, Λ`);`
    return push($\langle P, Q \rangle$, params_stack)
  end
End Function

3. Results

In this section we describe the empirical experiments designed to evaluate the performance of the proposed method. According to the standard CF's evaluation framework [32], the evaluation of the quality of the predictions is carried out by measuring the Mean Absolute Error (MAE) of the model output. The MAE is defined as average absolute difference between the actual ratings R and the predicted ratings $\hat{R}$ of the test split R^{test}:

$$\text{MAE} = \frac{1}{\#R^{\text{test}}} \sum_{\langle u,i \rangle \in R^{\text{test}}} |R_{u,i} - \hat{R}_{u,i}|. \tag{10}$$

Analogously, the quality of the top l recommendations is evaluated by the precision and recall quality measures on the recommendation lists. Remark that, given an user u, his recommendation list of l items, L_u^l, is the collection of the l items $i_1, \ldots, i_l$ with $\hat{R}_{u,i_1}, \ldots, \hat{R}_{u,i_l}$ the highest l predicted ratings.

Now, we fix a parameter $\theta \geq 0$, which plays the role of a minimum threshold to discern whether or not an item is of interest to the users. The precision is defined as the average proportion of successful recommendations of the recommendation list:

$$\text{precision} = \frac{1}{\#U^{\text{test}}} \sum_{u \in U^{\text{test}}} \frac{\left\{ i \in L_u^l | R_{u,i} \geq \theta \right\}}{l}, \tag{11}$$

where U^{test} is the collection of users in the test split.

In the same vein, the recall is defined as the averaged proportion of successful recommendations included in the recommendation list with respect to the total number of test items the user u likes:

$$\text{recall} = \frac{1}{\#U^{\text{test}}} \sum_{u \in U^{\text{test}}} \frac{\left\{ i \in L_u^l | R_{u,i} \geq \theta \right\}}{\left\{ i \in R_u^{\text{test}} | R_{u,i} \geq \theta \right\}}, \tag{12}$$

where R_u^{test} is the collection of items in the test split at the u-th row of R.

The experimental evaluation has been carried out using MovieLens [33], FilmTrust [34] and MyAnimeList [35] datasets. Table 1 contains their main parameters. To ensure the reproducibility of these experiments, all of them have been run using the benchmark version of these datasets included in Collaborative Filtering for Java (CF4J) [36].

Table 1. Main parameters of the datasets used in the experiments.

Dataset	Number of Users	Number of Items	Number of Ratings	Number of Test Ratings	Possible Scores
MovieLens 100K	943	1682	92,026	7974	1 to 5 stars
MovieLens 1M	6040	3706	911,031	89,178	1 to 5 stars
FilmTrust	1508	2071	32,675	2819	0.5 to 4.0 with half increments
MyAnimeList	69,600	9927	5,788,207	549,027	1 to 10

The selection of the baselines has been done trying to pick a representative sample of the existing MF models. A first approach, based on the deep nature of the proposed method, is to focus on existing deep factorization models such as FAuST [25] and DNMF [27]. However recall that, as stated in Section 1, these methods were designed for image processing, not for solving CF problems, so a performance similar to other standard MF-based CF algorithms is not guaranteed. Trying to discern whether these methods would be suitable for CF, we proposed a preliminary experiment for evaluating the performance of FAuST against the MovieLens 100K dataset. To tune the hyper-parameters, a grid search was conducted whose limits are as follows. It was searched for factorizations of the rating matrix R into up to 4 smaller matrices, with 6 or 9 hidden factors for each level and a gradient descent step size (learning rate) of $\gamma = 0.0001$ or 0.01. Notice that FAuST also allows you to impose a sparsity constraint on each factor matrix. For this experiment, it was allowed a 0% of sparsity (i.e., fully dense matrix with no restrictions) or a 20% of sparsity (that is, up to 80% of non-vanishing entries are allowed). The best found model attained a MAE of 3.49262 with a factorization into 4 matrices, all with 9 hidden factors, no sparsity constraints and a learning rate of $\gamma = 0.01$. This result is very far from the current baselines reported in the literature, which are typically around 0.75 as we will see below, showing that the existing deep methods are unsuitable as RSs. In is worthy to mention that, although FAuST with no sparsity constraints is very similar to the usual PMF [29] algorithm, it looks for a factorization into normalized matrices with Frobenius norm 1. This normalization seems to be crucial for the bad performance of FAuST as RS: this restriction might be very useful for image processing and memory saving problems which are the usual applications of FAuST, but it seems to be very ill-posed for regression problems.

Due to the failure of this method as RS, it was decided to pull apart the existing deep factorization methods, designed specifically for image processing, and to choose the competing baselines from the most popular MF-based CF methods, namely PMF [29], NMF [30] and SVD++ [31].

These baselines contain several hyper-parameters that must be tuned in order to improve their performance on the selected datasets. The optimal configuration of the hyper-parameters of each baseline has been found by a grid search optimization. All combinations resulting from evaluating a wide range of values for each hyper-parameter have been evaluated, and the combination returning

the least MAE on the predictions of the test split has been selected as the most suitable for each baseline and data set. Table 2 contains the resulting hyper-parameters of this optimization process. Recall that k denotes the number of hidden factors in the factorization, γ is the learning rate of the associated gradient descent optimization and λ is the regularization hyper-parameter.

Table 2. Best hyper-parameters for each baseline found by a grid search optimization.

Dataset	PMF	NMF	SVD++
MovieLens 100K	$k = 2$, $\gamma = 0.01$, $\lambda = 0.025$	$k = 2$	$k = 2$, $\gamma = 0.0014$, $\lambda = 0.08$
MovieLens 1M	$k = 8$, $\gamma = 0.01$, $\lambda = 0.045$	$k = 2$	$k = 4$, $\gamma = 0.0014$, $\lambda = 0.05$
FilmTrust	$k = 4$, $\gamma = 0.015$, $\lambda = 0.1$	$k = 2$	$k = 2$, $\gamma = 0.0014$, $\lambda = 0.1$
MyAnimeList	$k = 10$, $\gamma = 0.005$, $\lambda = 0.085$	$k = 2$	$k = 4$, $\gamma = 0.0014$, $\lambda = 0.02$

Analogously, DeepMF also contains several hyper-parameters needed to fit the model to an specific dataset. As with the baselines, we performed a grid search optimization in order to obtain the combination of parameters that minimizes the prediction error (MAE) in the test split. In this case, the recursive nature of the proposed method causes that the number of hyper-parameter combinations to be evaluated grows exponentially.

To deal with this combinatorial explosion, we evaluated with depths from 1 to 4, i.e., performing 1 to 4 deep factorizations. Each of this factorizations has been evaluated varying the number of latent factors $k \in \{3, 6, 9\}$, the learning rate $\gamma \in \{0.01, 0.1\}$ and the regularization $\lambda \in \{0.01, 0.1\}$. The number of iterations has been fixed to $T = [50, 50, \ldots, 50]$ for all factorizations. In total, 22,620 combinations of hyper-parameters has been evaluated for each dataset. Table 3 contains the top five results returned by the grid search optimization. Note that hyper-parameters has been named as in Algorithm 1 to facilitate their interpretation.

Table 3. Top 5 results resulting from grid search optimization of DeepMF.

Dataset	Rank	Hyper-Parameters	MAE
MovieLens 100K	1	K = [3, 3, 3, 3]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.75017
	2	K = [3, 3, 3, 9]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.75077
	3	K = [3, 6, 6, 3]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.75079
	4	K = [3, 3, 3]; Γ= [0.01, 0.1, 0.01]; Λ = [0.1, 0.01, 0.1]	0.75092
	5	K = [3, 3, 9, 9]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.75105
MovieLens 1M	1	K = [9, 3, 3]; Γ= [0.01, 0.01, 0.01]; Λ = [0.1, 0.01, 0.1]	0.70943
	2	K = [9, 3, 9, 3]; Γ= [0.01, 0.01, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.70948
	3	K = [9, 3, 9]; Γ= [0.01, 0.01, 0.01]; Λ = [0.1, 0.01, 0.1]	0.70949
	4	K = [9, 3, 3, 9]; Γ= [0.01, 0.01, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.70956
	5	K = [9, 3, 3, 3]; Γ= [0.01, 0.01, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.70959
FilmTrust	1	K = [6, 6, 3]; Γ= [0.01, 0.1, 0.01]; Λ = [0.1, 0.01, 0.01]	0.64936
	2	K = [6, 6, 3]; Γ= [0.01, 0.1, 0.01]; Λ = [0.1, 0.01, 0.1]	0.64987
	3	K = [6, 3, 6]; Γ= [0.01, 0.1, 0.01]; Λ = [0.1, 0.01, 0.1]	0.65072
	4	K = [9, 3, 3]; Γ= [0.01, 0.1, 0.01]; Λ = [0.1, 0.01, 0.1]	0.65088
	5	K = [6, 6]; Γ= [0.01, 0.01]; Λ = [0.1, 0.01]	0.65102
MyAnimeList	1	K = [9, 6, 9, 6]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.97447
	2	K = [9, 9, 6, 9]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.97452
	3	K = [9, 6, 6, 6]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.97454
	4	K = [9, 6, 6, 9]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.97454
	5	K = [9, 9, 9, 6]; Γ= [0.01, 0.1, 0.01, 0.1]; Λ = [0.1, 0.01, 0.1, 0.01]	0.97458

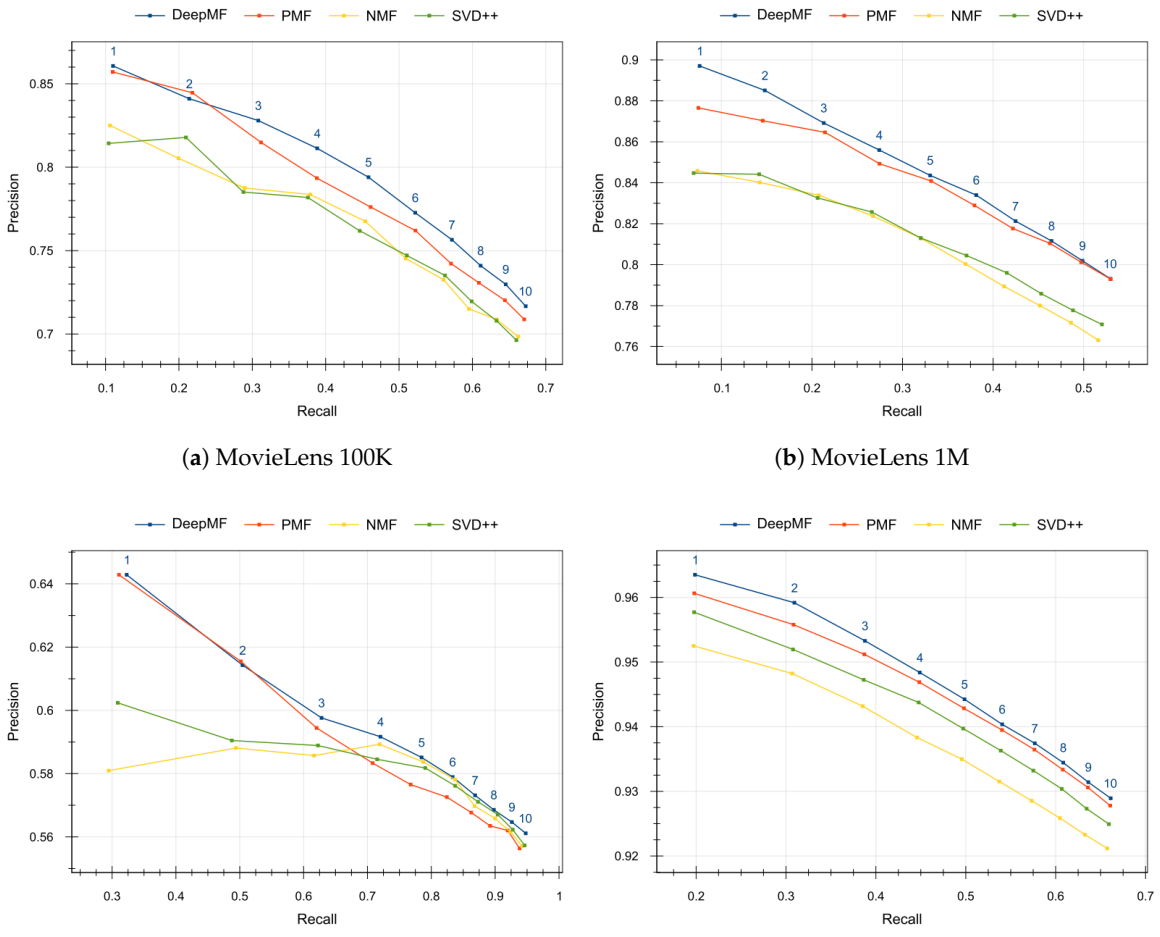
Once all the evaluated models have been set up, the best tuned models can be compared to measure the prediction and recommendation improvement of DeepMF with respect to the selected baselines. Table 4 contains the MAE of the predictions performed by all the evaluated models. We can observe that the proposed model DeepMF significantly improves the accuracy of predictions with respect to other models in MovieLens 100K, MovieLens 1M and FilmTrust datasets. Similarly, in the MyAnimeList dataset, DeepMF substantially improves PMF and NMF baselines and achieves a slightly

worse MAE than SVD++. Observe that this later underperformance might be due to the fact that MyAnimeList is significantly larger than any other dataset, so deeper models of DeepMF would be needed in order the exploit the recursive nature of the proposed method that fit the dataset better than SVD++.

Table 4. Quality of the predictions measured by the MAE. The lower the better. In bold the best recommendation model for each dataset.

Dataset	DeepMF	PMF	NMF	SVD++
MovieLens 100K	**0.75017**	0.76720	0.79138	0.78170
MovieLens 1M	**0.70943**	0.71868	0.75166	0.74285
FilmTrust	**0.64936**	0.84659	0.82911	0.65748
MyAnimeList	0.97447	1.10006	1.12025	**0.95179**

Figure 2 contains the precision and recall of the recommendations when varying the number of desired top recommendations from $l = 1$ to $l = 10$ items. The threshold θ for precision and recall has been set to $\theta = 4$ for MovieLens datasets, $\theta = 3.5$ for FilmTrust dataset and $\theta = 7$ for MyAnimeList dataset. We can observe that in all evaluated datasets the proposed method DeepMF provides the best balance between the precision and recall quality measures for any number of recommendations.



(**a**) MovieLens 100K

(**b**) MovieLens 1M

(**c**) FilmTrust

(**d**) MyAnimeList

Figure 2. Quality of the recommendations measured by precision and recall. The higher the better. Blue number over the lines represents the size of the recommendation list for each value.

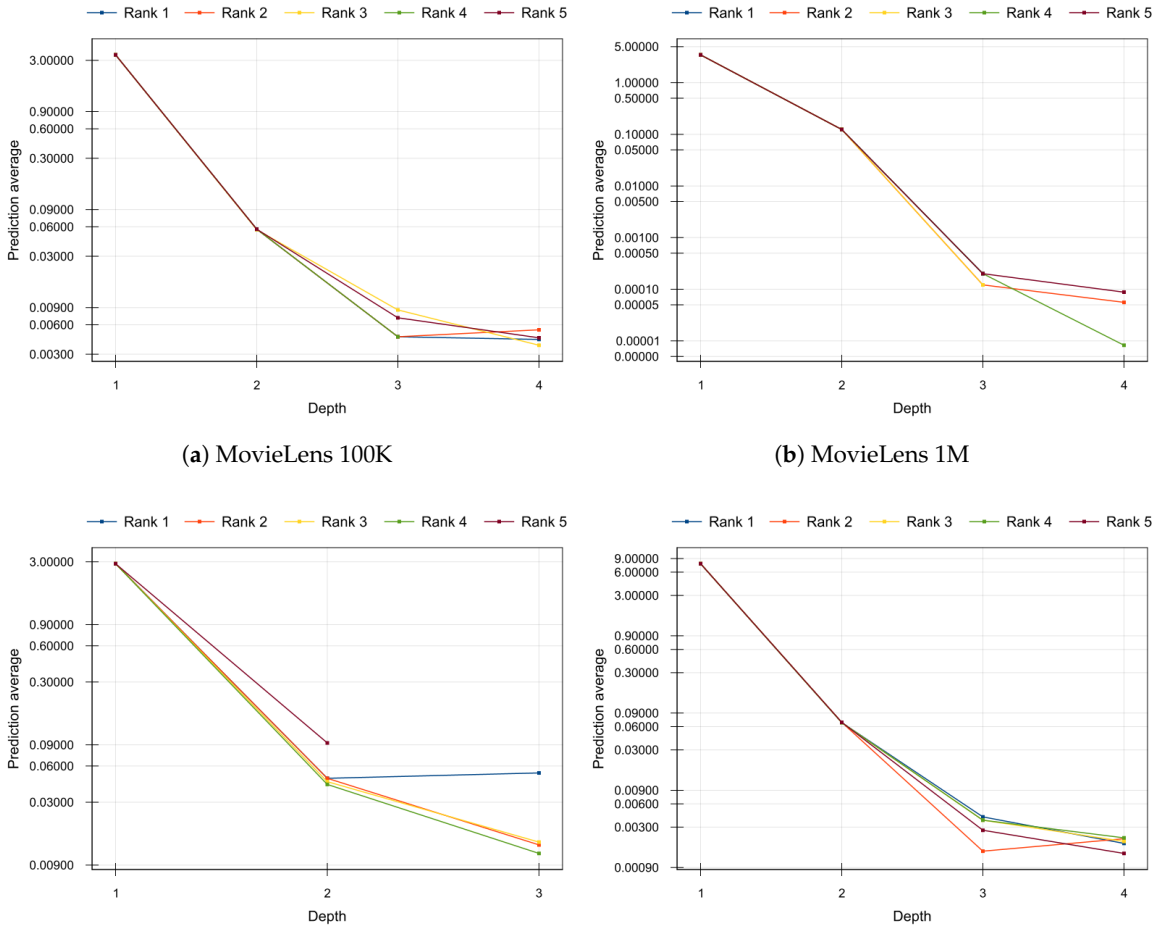
All experiments conducted in this article are committed to reproducible science. The full code of these experiments is available at https://github.com/ferortega/deep-matrix-factorization.

4. Discussion

In this paper, we presented DeepMF, a novel MF-based CF algorithm using a DL approach. As stated in Section 1 the DL is an incipient approach in the ML field that consists of a hierarchical self-training method that uses the information acquired from data. Although DL has been used mainly in artificial neural networks, the proposed model breaks this tendency by applying the foundations of DL to the field of MF.

The essence of the proposed method lies in the DL approach. Our model performs successive matrix factorizations in order to successively refine the model output. Thus, the first factorization, the least deep one, represents the classic approach of MF-based CF models and seeks to predict the score that a user will give to an item. The second factorization seeks to refine the previous prediction by trying to increase the predictions that tend to be lower than the real rating and decrease those that tend to be higher. Subsequently, the learning is deepened by correcting the errors of the errors to build a DL model that converges towards a prediction as close as possible to the real value to be learned.

This expected behavior is corroborated by the results of the experiment shown in Figure 3. On it, we plot the average value of the predictions provided by each factorization layer according to its depth. The same decreasing trend in the average prediction is observed in all the analyzed datasets: as factorization is deeper, errors to be refined tend to zero and the learning process converges. The Figure 3 includes the top 5 combinations of hyper-parameters obtained in the grid search shown in Table 3. Note that a logarithmic scale has been used on the y-axis to emphasize the differences in the deeper factorizations.



(**a**) MovieLens 100K

(**b**) MovieLens 1M

(**c**) FilmTrust

(**d**) MyAnimeList

Figure 3. Average value of the predictions provided by each factorization according to its depth. Top 5 combinations of hyper-parameters for each dataset included in Table 3 are shown.

The hypothesis of this contribution was that a DL approach applied to a MF-based CF can improve the quality of both predictions and recommendations. This hypothesis has been confirmed with the experimental results showed in Section 3. The quality of the predictions (see Table 4) and recommendations (see Figure 2) of the proposed method, DeepMF, exceeds the baselines used in the 3 datasets analyzed: MovieLens 100K, MovieLens 1M and FilmTrust.

Summarizing, the proposed model in this paper expands the landscape of matrix factorization models by importing a Deep Learning approach from the field of neural computing. From this point, several future research lines open. Maybe, the most obvious one would be to analyze the DL approach performed in this paper with other matrix factorization models and to define different loss functions depending on the depth of factorization. For example, experimental results show that SVD++ factorization works properly on MyAnimeList dataset, so it would be interesting to evaluate the performance of DeepMF using SVD++ as the initial factorization and other factorizations models for the deeper factorizations.

A more ambitious prospective work would be transferring the ideas of this paper to a purely bioinspired framework. For instance, it can be studied the incorporation of DeepMF as a model to be implemented by the neurons of a fully connected or convolutional neural network. In this way, the knowledge of the scientific community about network architectures can be applied to give rise to deeper an more involved nested patterns of matrix factorizations.

Author Contributions: Conceptualization, R.L.-C. and Á.G.-P. and F.O.; formal analysis, Á.G.-P.; methodology, R.L.-C. and F.O.; software, A.G.-P. and F.O.; writing—original draft preparation, R.L.-C. and Á.G.-P. and F.O.; writing—review and editing, R.L.-C. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by Spanish Ministry of Science and Education and Competitivity (MINECO) and European Regional Development Fund (FEDER) under grants TIN2017-85727-C4-3-P (DeepBio).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Manogaran, G.; Lopez, D. A survey of big data architectures and machine learning algorithms in healthcare. *Int. J. Biomed. Eng. Technol.* **2017**, *25*, 182–211. [CrossRef]
2. Molina, M.; Garip, F. Machine Learning for Sociology. *Ann. Rev. Sociol.* **2019**, *45*, 27–45. [CrossRef]
3. Lu, Y. Industry 4.0: A survey on technologies, applications and open research issues. *J. Ind. Inform. Integr.* **2017**, *6*, 1–10. [CrossRef]
4. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl. Based Syst.* **2013**, *46*, 109–132. [CrossRef]
5. Son, Y.; Choi, Y. Improving Matrix Factorization Based Expert Recommendation for Manuscript Editing Services by Refining User Opinions with Binary Ratings. *Appl. Sci.* **2020**, *10*, 3395. [CrossRef]
6. Zhang, D.; Liu, L.; Wei, Q.; Yang, Y.; Yang, P.; Liu, Q. Neighborhood Aggregation Collaborative Filtering Based on Knowledge Graph. *Appl. Sci.* **2020**, *10*, 3818. [CrossRef]
7. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749. [CrossRef]
8. Lara-Cabrera, R.; González-Prieto, Á.; Ortega, F.; Bobadilla, J. Evolving matrix-factorization-based collaborative filtering using genetic programming. *Appl. Sci.* **2020**, *10*, 675. [CrossRef]
9. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
10. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
11. Deng, L.; Hinton, G.; Kingsbury, B. New types of deep neural network learning for speech recognition and related applications: An overview. In Proceedings of the 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, Vancouver, BC, Canada, 26–31 May 2013; pp. 8599–8603.
12. Wu, M.; Li, C. Image recognition based on deep learning. In Proceedings of the 2015 Chinese Automation Congress (CAC), Wuhan, China, 27–29 November 2015; pp. 542–546.

13. Deng, L.; Liu, Y. *Deep Learning in Natural Language Processing*; Springer: Berlin/Heidelberg, Germany, 2018.
14. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Comput. Surv.* **2019**, *52*, 1–38. [CrossRef]
15. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, Melbourne, VIC, Australia, 19–25 August 2017; pp. 1725–1731.
16. Ouyang, Y.; Liu, W.; Rong, W.; Xiong, Z. *Autoencoder-Based Collaborative Filtering*; SpringerLink: Berlin/Heidelberg, Germany, 2014; pp. 284–291.
17. Bobadilla, J.; Alonso, S.; Hernando, A. Deep Learning Architecture for Collaborative Filtering Recommender Systems. *Appl. Sci.* **2020**, *10*, 2441. [CrossRef]
18. He, X.; Du, X.; Wang, X.; Tian, F.; Tang, J.; Chua, T.S. Outer product-based neural collaborative filtering. In Proceedings of the 27th International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 2227–2233.
19. Abavisani, M.; Patel, V.M. Deep Sparse Representation-Based Classification. *IEEE Signal Proces. Lett.* **2019**, *26*, 948–952. [CrossRef]
20. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based Recommendations with Recurrent Neural Networks. *arXiv* **2015**, arXiv:cs.LG/1511.06939.
21. Tan, Y.K.; Xu, X.; Liu, Y. Improved Recurrent Neural Networks for Session-Based Recommendations. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; Association for Computing Machinery: New York, NY, USA, 2016; pp. 17–22.
22. Wu, S.; Ren, W.; Yu, C.; Chen, G.; Zhang, D.; Zhu, J. Personal recommendation using deep recurrent neural networks in NetEase. In Proceedings of the 2016 IEEE 32nd International Conference on Data Engineering (ICDE), Helsinki, Finland, 16–20 May 2016; pp. 1218–1229.
23. Wang, Q.; Yin, H.; Hu, Z.; Lian, D.; Wang, H.; Huang, Z. Neural Memory Streaming Recommender Networks with Adversarial Training. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 2467–2475.
24. He, X.; He, Z.; Du, X.; Chua, T.S. Adversarial Personalized Ranking for Recommendation. In Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; Association for Computing Machinery: New York, NY, USA, 2018; pp. 355–364.
25. Le Magoarou, L.; Gribonval, R. Flexible multilayer sparse approximations of matrices and applications. *IEEE J. Sel. Top. Signal Process.* **2016**, *10*, 688–700. [CrossRef]
26. Trigeorgis, G.; Bousmalis, K.; Zafeiriou, S.; Schuller, B.W. A Deep Matrix Factorization Method for Learning Attribute Representations. *IEEE Trans. Pattern Anal. Mach. Intel.* **2017**, *39*, 417–429. [CrossRef]
27. Guo, Z.; Zhang, S. Sparse deep nonnegative matrix factorization. *Big Data Min. Anal.* **2020**, *3*, 13–28. [CrossRef]
28. Sharma, P.; Abrol, V.; Sao, A.K. Deep-Sparse-Representation-Based Features for Speech Recognition. *IEEE/ACM Trans. Audio Speech Lang. Process.* **2017**, *25*, 2162–2175. [CrossRef]
29. Mnih, A.; Salakhutdinov, R.R. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*; University of Toronto: Toronto, ON, Canada, 2008; pp. 1257–1264.
30. Lee, D.D.; Seung, H.S. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*; University of Toronto: Toronto, ON, Canada, 2001; pp. 556–562.
31. Koren, Y. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; Association for Computing Machinery: New York, NY, USA, 2008; pp. 426–434.
32. Bobadilla, J.; Hernando, A.; Ortega, F.; Bernal, J. A framework for collaborative filtering recommender systems. *Expert Syst. Appl.* **2011**, *38*, 14609–14623. [CrossRef]
33. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intel. Syst.* **2015**, *5*, 1–19. [CrossRef]

34. Guo, G.; Zhang, J.; Yorke-Smith, N. A Novel Bayesian Similarity Measure for Recommender Systems. In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI), Beijing, China, 3–9 August 2013; pp. 2619–2625.
35. MyAnimeList.net. MyAnimeList Dataset. 2020. Available online: https://www.kaggle.com/azathoth42/myanimelist (accessed on 18 May 2020).
36. Ortega, F.; Zhu, B.; Bobadilla, J.; Hernando, A. CF4J: Collaborative filtering for Java. *Knowl. Based Syst.* **2018**, *152*, 94–99. [CrossRef]