applied sciences

MDPI

Article

Deep Learning Architecture for Collaborative Filtering Recommender Systems

Jesus Bobadilla *, Santiago Alonso and Antonio Hernando

Dpto. Sistemas Informáticos, Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid, 28031 Madrid, Spain; santiago.alonso@upm.es (S.A.); antonio.hernando@upm.es (A.H.)

* Correspondence: jesus.bobadilla@upm.es

Received: 17 March 2020; Accepted: 30 March 2020; Published: 3 April 2020

Abstract: This paper provides an innovative deep learning architecture to improve collaborative filtering results in recommender systems. It exploits the potential of the reliability concept to raise predictions and recommendations quality by incorporating prediction errors (reliabilities) in the deep learning layers. The underlying idea is to recommend highly predicted items that also have been found as reliable ones. We use the deep learning architecture to extract the existing non-linear relations between predictions, reliabilities, and accurate recommendations. The proposed architecture consists of three related stages, providing three stacked abstraction levels: (a) real prediction errors, (b) predicted errors (reliabilities), and (c) predicted ratings (predictions). In turn, each abstraction level requires a learning process: (a) Matrix Factorization from ratings, (b) Multilayer Neural Network fed with real prediction errors and hidden factors, and (c) Multilayer Neural Network fed with reliabilities and hidden factors. A complete set of experiments has been run involving three representative and open datasets and a state-of-the-art baseline. The results show strong prediction improvements and also important recommendation improvements, particularly for the recall quality measure.

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

1. Introduction

Recommender Systems (RS) are powerful tools to address the information overload problem in the Internet. They make use of diverse sources of information. Explicit votes from users to items, and implicit interactions are the basis of the Collaborative Filtering (CF) RS. Implicit interactions examples are clicks, listened songs, watched movies, likes, dislikes, etc. Often, hybrid RS [1] are designed to ensemble CF with some other types of information filtering: demographic [2], context-aware [3], content-based [4], social information [5], etc. RS cover a wide range of recommendation targets, such as travels [6], movies [7], restaurants [8], fashion [9], news [10], etc.

Traditionally, CF RS have been implemented using machine learning methods and algorithms, mainly the memory-based K nearest neighbor algorithm, and more recently the Matrix Factorization (MF) method. MF is an efficient model-based approach that obtains accurate recommendations; it is easy to understand and to implement, and it is based on the dimension reduction basis, where a reduced set of hidden factors contains the ratings information. Predictions are obtained by making the dot product of the users and the items hidden factors. There are several implementations of the MF such as PMF [11] and BNMF. An important drawback of the MF predictions is that the linear dot product cannot catch the complex non-linear relations existing among the set of hidden factors. Neural models do not have this restriction and they are called to lead the CF research in RS.

Currently, RS research is directed toward deep learning approaches [12,13]. It is usual to find neural content-based approaches: images [8,9], text [10,14], music [15,16], videos [17,18], etc. The above targets

are processed by using different neural networks architectures: Convolutional Neural Networks [19], Recurrent Neural Networks [20], Multilayer Neural Networks (MNN) [3,21], and autoencoders [7,22] are usual approaches. The preceding publications are focused on content-based or hybrid filtering. CF neural approaches can be classified as: (1) Deep Factorization Machines (deepFM) [23] and (2) Neural Collaborative Filtering (NCF) [24]. NCF simultaneously processes item and user information using a dual neural network. The two main NCF implementations are (a) NCF [24] and (b) Neural Network Matrix Factorization (NNMF) [25]. Since NCF reports better accuracy than NNMF, we use it as our baseline. Figure 1 shows the NCF architecture: the input sparse data (ratings of the users and ratings of the items) is converted in dense information by means of an embedding layer. Then, a Multilayer Perceptron combines both features paths by concatenating them. Training is performed using the known ratings as labels. Once the architecture has learned, it can predict unknown ratings in a non-linear process. Our proposed architecture makes use of the NCF one, expanding it to hold training using known errors and then predicting reliabilities. We will refer to the proposed architecture as Reliability-based Neural Collaborative Filtering (RNCF).

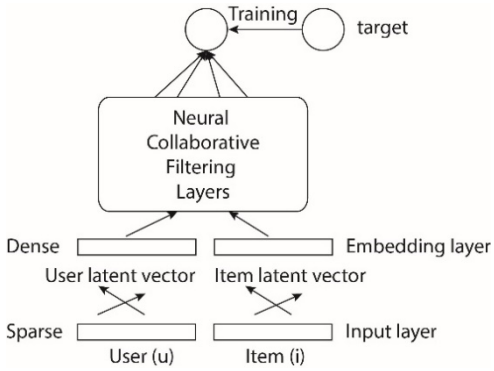

Figure 1. Neural Collaborative Filtering (NCF) architecture.

There are some publications that combine wide and deep learning [26,27] to support CF RS. The wide side is implemented using a perceptron, whereas the deep side is implemented using an MNN. The wide component learns simple relations (memorization); the deep component learns complex relations (generalization). The proposed architecture (RNCF) focuses on the deep learning approach to catch complex relations both to predict reliabilities and to predict rating values. The wide learning cannot make these functions, but it could slightly improve the overall results. A potential future work is to expand RNCF to hold a wide learning component. The deepFM architecture [28] joins a deep and a wide component to process the raw input vectors. It simultaneously learns low and high-order feature interactions. Compared to NCF, deepFM achieves a slight accuracy improvement (0.33% Criteo dataset, 0.48% Company dataset). The proposed approach (RNCF) fits better with the NCF architecture than with the deepFM one. Additionally, the wide architectures have the size of the input layer as a drawback that jeopardizes their scalability. Given the limited room for improvement using the deepFm model, we have chosen the more regular and scalable NCF architecture as a base to create our model. There are some other approaches that provide neural networks based on improved information coming from data, such as [29], where authors extract interaction behavior from bipartite graphs. Deep-learning architectures can be designed by extracting multi-criteria information from data [30]. Additionally, a classification-based deep learning collaborative filtering approach is stated in [31], where the learning process takes as information two different binary sources: (a) relevant/non-relevant votes, and (b) voted/non-voted items.

Once we have reviewed the most important neural networks solutions that the state of the art provides us, we are going to explain the main concepts of the proposed approach by means of Figure 2. Some state-of-the-art deep learning approaches to CF RS use hidden factors of users and items as embedding [24]. Then, from hidden factors, an MNN learns the existing non-linear relations

between hidden factors and ratings (Figure 2a). Once the MNN has learned, predictions can be made by feedforward running the MNN, feeding it with the hidden factors corresponding to the pairs <user, item> that have not an associated rating. These types of architectures provide improved results with regard to the classical MF approach, where predictions are obtained by means of a linear hidden factors-based dot product.

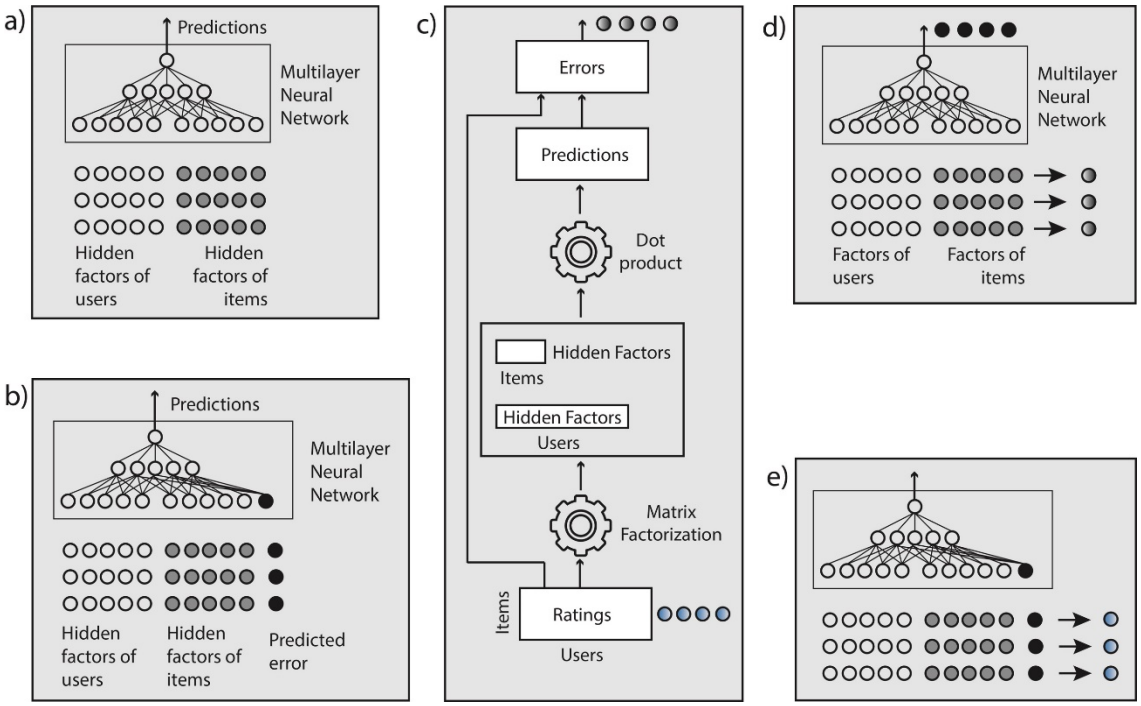

Figure 2. Proposed method (RNCF) concepts: (**a**) NCF baseline, (**b**) improved NCF by adding reliabilities, (**c**) extraction of the hidden factors and the real prediction errors, (**d**) prediction of the errors (reliabilities), (**e**) training stage in the RNCF architecture. NCF: Neural Collaborative Filtering, RNCF: Reliability-based Neural Collaborative Filtering.

Related to the reliability or confidence of the RS recommendations, some published papers [32] provide additional information to each prediction or recommendation: its reliability. As an example, we can recommend an item with 4 stars and a reliability of 2 stars, or we can recommend another item with 3.5 stars and a reliability of 4 stars. It is probable that the recommended user will choose the second item, since even knowing that our recommendation is lower (3.5 compared to 4), it is more reliable (4 compared to 2). Using the same concept, we can select recommendations from the set of predictions taking into account not only the prediction values, but also their reliabilities. By carefully studying relations between predictions and reliabilities, probably we could extract some rules to recommend. Instead, we use a deep learning stage to learn the existing non-linear relation between predictions, reliabilities, and accurate recommendations.

A recommendation reliability is defined as its predicted error [32]. That is, in addition to predicting ratings, we also predict the error of those predictions. The hypothesis of this paper claims that the CFN state-of-the-art architectures can be improved by adding the predicted errors (reliabilities) to the MNN learning. Thus, the state-of-the-art architecture shown in Figure 2a can be improved by using the architecture shown in Figure 2b.

The question that now arises is the way to get the predicted errors: reliabilities (black circles in Figure 2b). This process will need two separated stages, as shown in Figure 2c,d. Figure 2c shows the way we can get a set of real prediction errors: first, an MF is processed to obtain the users and items hidden factors; then, we predict each existing rating in the dataset. Since we predict known ratings, we can obtain real (exact) prediction errors (top of Figure 2c). The next process (Figure 2d) is to make

use of the real prediction errors to train an MNN; the MNN is fed with hidden factors in its input layer, and it learns using the real prediction errors as labels. Once the MNN has learned, it can feedforward reliabilities: error predictions (black circles) from not-voted <user, item> pairs, where each user and each item consist of their hidden factors.

Finally, to accomplish our objective: to accurately predict and recommend using hidden factors and reliabilities (Figure 2b), we must train the MNN; Figure 2e shows the way we use the known ratings (left to right faded circles) to train the MNN: known ratings are used as labels in the neural network training process, so that the MNN learns to accurately predict ratings (make predictions) from hidden factors and reliabilities. The trained MNN in Figure 2e is feedforward used to make non-linear predictions (Figure 2b).

Please note that the proposed method acts as a collaborative filtering black box that is fed just with the votes of each Recommender System. In this way, it can be used as a base to implement hybrid recommender systems containing demographic, context-based, content-based, and social information.

2. Materials and Methods

The proposed RNCF architecture gets improved recommendation results by performing three different learning processes that produce three different abstraction levels in the whole system. Borrowing diagrams from Figure 2, Figure 3 shows the overall architecture: each one of the performed processes as well as their stacked connections. The first RNCF process is drawn in the bottom gray rectangle of Figure 3; it is labeled as "Abstraction level 1". Using the CF dataset containing the ratings casted by the users to the items, we run a machine learning MF method. In this way, we obtain two dense matrices containing the hidden factors: the users and the items matrices. From the two hidden factor matrices, predictions can be obtained by processing the dot product of each selected user and item. Usually, predictions are computed just for the not-voted items of each user, since it is not useful to recommend consumed items. In our proposed RNCF architecture, in its first learning process, we do the opposite: predictions are obtained for the voted (consumed) items (Equation (11)). This is because, in the first abstraction level, we look for the prediction errors made in the MF stage. The first stage of the proposed method can be mathematically interpreted as:

$$\text{Let } U \text{ be the set of users in the CF dataset.} \tag{1}$$

$$\text{Let } I \text{ be the set of items in the CF dataset.} \tag{2}$$

Let V be the set of categorical values of the ratings, where '•' means 'not consumed'

$$V = \{1,\ 2,\ 3,\ 4,\ 5,\ \bullet\} \tag{3}$$

$$\text{Let } r_{u,i} \text{ be the rating of the user } u \in U \text{ to the item } i \in I,\ r_{u,i} \in V \tag{4}$$

$$\text{Let } F \text{ be the number of chosen factors for the MF method} \tag{5}$$

$$\text{Let } s_{u,f} \text{ be the hidden factor } f \in F \text{ of the user } u \in U \tag{6}$$

$$\text{Let } q_{f,i} \text{ be the hidden factor } f \in F \text{ of the item } i \in I \tag{7}$$

$$\text{Let } p_{u,i} \text{ be the prediction of the rating to the user } u \in U \text{ on the item } i \in I \tag{8}$$

$$p_{u,i} = \sum_{f \in F} s_{u,f}.q_{f,i} \tag{9}$$

$$\text{Let } e_{u,i} \text{ be the error of the prediction } p_{u,i} \tag{10}$$

$$e_{u,i} = (r_{u,i} - p_{u,i})^2\ \forall r_{u,i} \neq \bullet \tag{11}$$

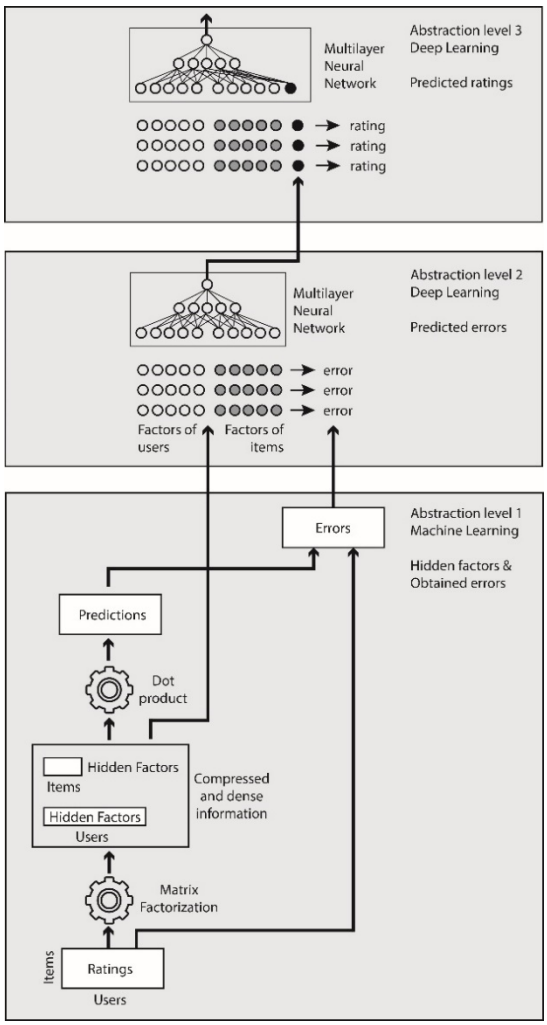

Figure 3. Proposed RNCF architecture.

The second stage of the proposed method is drawn in the middle of Figure 3, which is labeled as 'Abstraction level 2'. As it can be seen, we make use of an MNN to predict the MF error of each prediction. In this case, as usual, once the neural network has learned, predictions are made for the not-voted ratings (Equation (18)). To accomplish the learning process, the neural network input is fed with the hidden factors of users and items (Equation (13)), whereas each label output is fed with each obtained error in Stage 1 of the proposed method (Equation (15)). That is, Stage 2 MNN learns the obtained errors from Stage 1 to provide the expected error of each prediction (reliability). In its learning operation, the second-stage mathematical interpretation is:

$$\text{Let } X \text{ be the learning set of samples used to feed the MNN} \tag{12}$$

$$X = \left\{ s_{u,f},\ q_{f,i} \right\} \forall u \in U,\ \forall i \in I,\ \forall f \in F \ \Big|\ r_{u,i} \neq \bullet. \tag{13}$$

$$\text{Let } Y \text{ be the target set of labels used to feed the MNN} \tag{14}$$

$$Y = \left\{ e_{u,i} \right\} \forall u \in U,\ \forall i \in I \ \Big|\ r_{u,i} \neq \bullet. \tag{15}$$

Please note that in the first stage, real errors are obtained by predicting known (voted) ratings (Equation (11)), whereas in the second stage, we predict errors (reliabilities) for the not-voted ratings

(Equation (18)) based on the real errors on the voted ratings. Once the MNN has learned, in its feedforward operation (prediction of errors), the second-stage mathematical interpretation is:

$$\text{Let } g \text{ be the stage 2 MNN feedforward process.} \tag{16}$$

$$\text{Let } e'_{u,i} \text{ be the prediction of the error (reliability) obtained in the Stage 2 MNN} \tag{17}$$

$$e'_{u,i} = g(X)\ \forall u \in U,\ \forall i \in I \,\Big|\, r_{u,i} = \bullet. \tag{18}$$

$$\text{The MNN learns by minimizing the error } (e_{u,i} - e'_{u,i})^2\ \forall r_{u,i} \neq \bullet. \tag{19}$$

Finally, the RNCF Stage 3 is in charge of predicting the not-voted rating values. It takes, as input, the users and items hidden factors, as well as the previously predicted reliabilities. It is expected that the predicted reliabilities will provide the necessary additional information to improve predictions and recommendations. In its learning operation, the third-stage mathematical interpretation is:

$$\text{Let } X' \text{ be the learning set of samples used to feed the MNN:} \tag{20}$$

$$X' = \left\{s_{u,f},\ q_{f,i}, e'_{u,i}\right\} \forall u \in U,\ \forall i \in I, \forall f \in F \,\Big|\, r_{u,i} \neq \bullet. \tag{21}$$

$$\text{Let } Y' \text{ be the target set of labels used to feed the MNN:} \tag{22}$$

$$Y' = \left\{r_{u,i}\right\} \forall u \in U,\ \forall i \in I \,\Big|\, r_{u,i} \neq \bullet. \tag{23}$$

Once the third-stage MNN has learned, its feedforward prediction operation can be fed by selecting the not-voted ratings:

$$\text{Let } h \text{ be the stage 3 MNN feedforward process.} \tag{24}$$

$$\text{Let } t_{u,i} \text{ be the prediction of the rating :} \tag{25}$$

$$t_{u,i} = h(X')\ \forall u \in U,\ \forall i \in I \,\Big|\, r_{u,i} = \bullet. \tag{26}$$

$$\text{The MNN learns by minimizing the error } (t_{u,i} - r_{u,i})^2\ \forall r_{u,i} \neq \bullet. \tag{27}$$

Recommendation Z is chosen by selecting the N highest predictions:

$$Z = \left\{t_{u,i}\right\} \,\Big|\, t_{u,i} \geq t'_{u,i}\ ,\ t'_{u,i} \in Z^c,\ |Z| = N. \tag{28}$$

Now, the experiment's design is explained, including the used baseline, chosen datasets, tested quality measures, selected parameter values in the machine and the deep learning algorithms, used framework, etc. Experiments are performed using cross-validation and they have been tested on open CF datasets widely used by RS research papers. We have selected *Netflix**, *MovieLens* [33] and FilmTrust [34] as datasets. Table 1 contains the most relevant facts about these datasets. The cross-validation values used in the experiments are abstracted in Table 2. The chosen quality measure for prediction has been the Mean Absolute Error (MAE), whereas the chosen quality measures for recommendation have been Precision, Recall, and F1.

Table 1. Used datasets and their main facts.

Dataset	#Ratings	#Items	#Users	Ratings Values
*Netflix** (subset)	4,553,208	7000	10,000	1 to 5
MovieLens	1,000,209	3706	6040	1 to 5
FilmTrust	33,470	2509	1227	0.5 to 4 step 0.5

Table 2. Cross-validation values.

Parameter	Value
% training	80
% testing	20
N factors	$\{1, \ldots, 96\}$ step 5
Recommendation threshold (θ)	4

The *Precision* measure returns the proportion of relevant recommendations regarding the total number of N recommendations, where {relevant recommendations} are the ratings greater than or equal to a threshold. *Recall* is the proportion of relevant recommendations regarding the total number of relevant items (from the total number of items voted for the user). Please note that whereas N is a constant, the number of relevant items is not. *Recall* is a "relative" measure, since it is more difficult to extract relevant recommendations from a set of a few relevant items than from a large set of relevant items. The *F1* score combines precision and recall, giving the same relevance to each one of them: $F1 = 2 \cdot Precision \cdot Recall / (precision + recall)$.

Finally, to implement the deep learning processes, we have made use of the *Keras* API, working on top of *TensorFlow*. In this case, the programming language has been Python. *Keras* is a user friendliness API: it minimizes the required number of user actions for common use cases. *Keras* allows making designs by using fully configurable modules and it provides easy extensibility. The machine learning matrix factorization has been made using the Java-based CF4J framework [35].

The baseline for our experiments is the NCF neural architecture [24]. This is a state-of-the-art approach that outperforms the existing MF methods and gets better accuracy than the Neural Network Matrix Factorization design (NNMF). The deepFM architecture [28] has achieved similar results than the NCF one. Overall, the NCF architecture provides a representative baseline that outperforms the machine learning KNN and MF; it also is a representative baseline for neural state-of-the-art approaches to the CF RS field.

3. Results

Results in this section have been split in two subsections: the first one shows and explains the prediction quality results, both for the proposed RNCF and for the NCF baseline. The second subsection is devoted to showing the recommendation quality results.

3.1. Prediction Results

As explained in the previous section, the proposed RNCF architecture contains a Multilayer Neural Network (Figure 3, abstraction level 2) that learns from the errors made in the MF predictions. Once the network has learned, it can predict "errors in predictions" (reliabilities). The smaller the predicted error, the greater its reliability. Figure 4 shows the quality of the obtained reliabilities on the *MovieLens* dataset. As it can be seen, the neural network is not overfitted and its testing Mean Absolute Error (MAE) is close to the 0.01 value when 15 epochs have been run. The reliability values can be considered accurate enough to feed the neural network on top of the proposed architecture (Figure 3, abstraction level 3). The *Netflix* and *FilmTrust* results are very similar to the *MovieLens* one; they have been omitted to keep this subsection as short as possible.

Once reliabilities can be predicted on the abstraction layer 2 of the RNCF architecture, they can feed the abstraction layer 3 (on Figure 3). Figures 5 and 6 show the obtained MAE by using both the NCF baseline and the proposed RNCF. Figure 5 shows the *MovieLens* results, whereas Figure 6 shows the *Netflix* ones. *FilmTrust* prediction results have been omitted to keep the size of this subsection short: they show the same behavior as the *MovieLens* and the *Netflix* results.

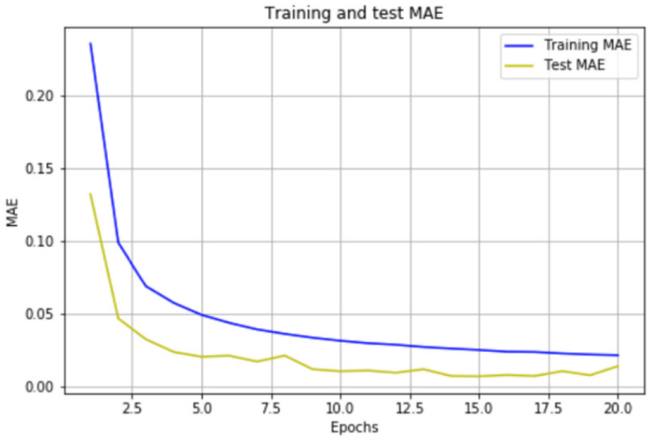

Figure 4. *MovieLens*. Mean Absolute Error (MAE) quality in the prediction of the reliabilities.

Figure 5 shows that the neural network is not overfitted both for the NCF baseline (top graph on Figure 5) and for the proposed RNCF (bottom graph on Figure 5). Comparing the testing MAE values at the end of the training (40 epochs in the baseline and 60 epochs in RNCF), we can claim the superiority of the proposed RNCF when applied to the *MovieLens* dataset: an error of 0.47 versus 0.62. RNCF achieves an improvement of 24% in the quality of the predictions. From the analysis of Figure 6 (results obtained using the *Netflix* dataset), the same conclusions can be extracted: there is not overfitting, and RNCF achieves an improvement of approximately 22% in the quality of the predictions. Overall, the proposed method achieves a strong improvement in the quality of the RS predictions.

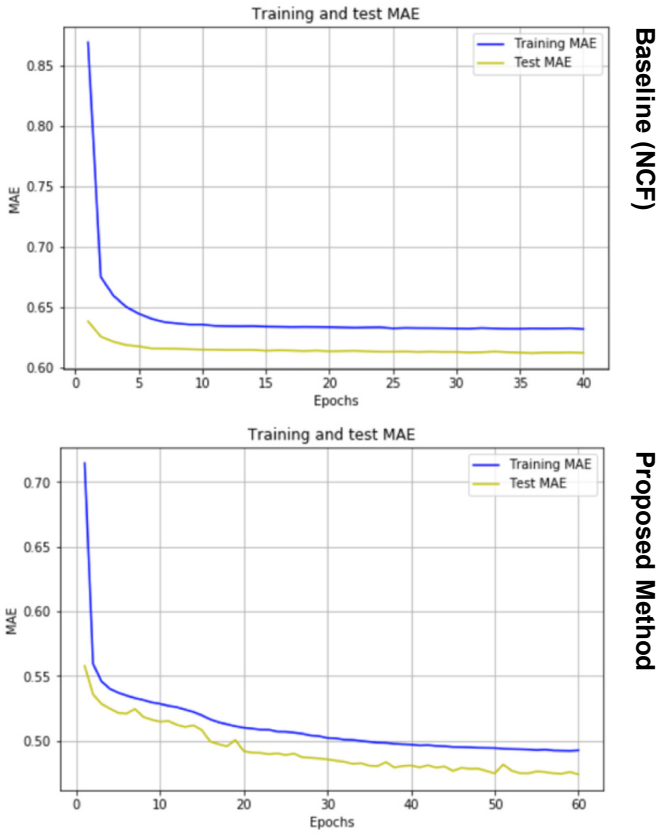

Figure 5. *MovieLens*. Mean Absolute Error (MAE) obtained by training both the NCF baseline (top chart) and the proposed RNCF neural architecture (chart below). Training and testing evolutions for the 60 first epochs (*x*-axis).

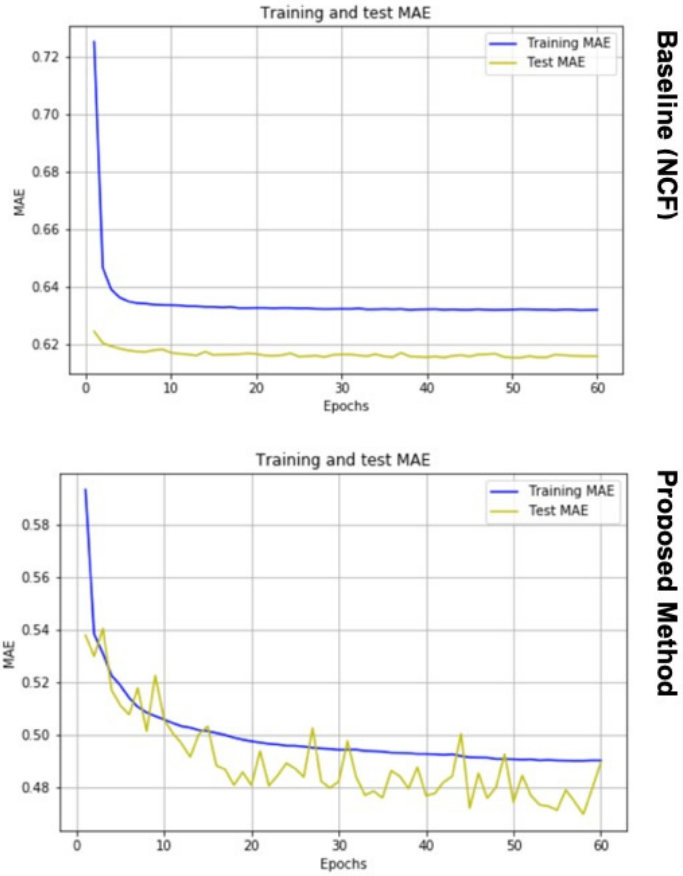

Figure 6. *Netflix*. Mean Absolute Error (MAE) obtained by training both the NCF baseline (top chart) and the proposed RNCF neural architecture (chart below). Training and testing evolutions for the 60 first epochs (*x*-axis).

Since the processed version of *Netflix* is much bigger than the *Movielens* one, we have also used a bigger value for the neural network *batch* size hyperparameter in order to avoid a significant increase in runtime. High *batch* sizes tend to produce fluctuations in trends results, as shown in Figure 6 (*Test MAE*). These fluctuations can be smoothed by processing the average of a series of separate runs of the same neural network. Fluctuations appear on the *TestMAE* curve, and not on the *TrainingMAE* curve, due to the smaller size of the test set.

3.2. Recommendation Results

In this section, the first set of experiments tests the proposed RNCF architecture on the *Netflix** dataset. *F1* results are shown in the top graph of Figure 7. As it can be seen, recommendation quality is improved for all the tested values of *N* (number of recommendations). It is remarkable that the most improved values are the most relevant ones: around 11 recommendations for a single user. The bottom graph in Figure 7 shows the details of the *F1* quality measure: we can separately see their *Precision* and *Recall* components. In this dataset, *Precision* does not significantly change from the baseline to the proposed RNCF, and the complete *F1* improvement is due to the *Recall* behavior. Additionally, low numbers of recommendations (*N*) produce better recommendation quality improvements. As we will see, the same type of results has been found on the performed experiments using the *MovieLens* and the *FilmTrust* datasets. Regarding numerical improvement values, if we take $N = 11$ as a reasonable number of recommendations, we approximately obtain the following improvement percentages: *Netflix* 34%, *MovieLens* 20%, *FilmTrust* 12%. These improvements decrease when a higher number of recommendations is selected.

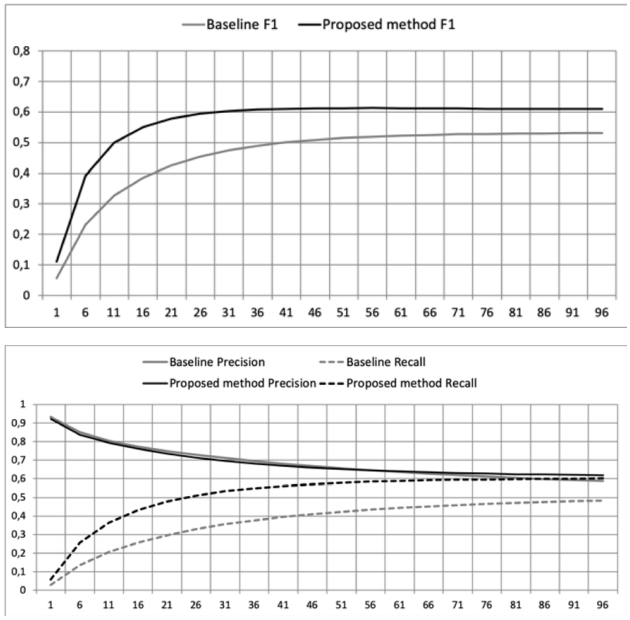

Figure 7. *Netflix* recommendation accuracy. Top chart: *F1* comparative between the NCF baseline and the proposed RNCF. Chart below: *Precision* and *Recall* comparative between the NCF baseline and the proposed RNCF; *x*-axis: number of recommendations (*N*).

We have conducted some additional research to understand why the *Recall* quality measure explains most of the *F1* improvement, as well as the decreasing quality improvement when the number of recommendations raises. Finally, we have found the justification: there is an inverse correlation between prediction values and reliability values. There is a tendency for higher predictions to present lower reliabilities. This finding has a deep influence on the evolution of both the *Precision* and the *Recall* quality measures: usually, recommendations are made by selecting the highest predicted values; instead, the proposed architecture 'discards' the non-reliable recommendations (the ones with a high predicted error). Since the proposed architecture can discard non-reliable predictions from the set of high predicted items, it will do a better job if there is a sufficient number of predictions to select and to discard, particularly due to the narrow margin that provides the low reliabilities related to high predictions. Here is where the *Recall* quality measure can show its behavior. *Recall* is a relative measure, relative to the number of relevant items: highly valued items that usually are the highly predicted ones. On the other hand, *Precision* is an absolute measure that depends on the constant *N* (number of recommendations). The *Recall* difference between the baseline method and the proposed one will increase as the number of relevant items increases, since the proposed method will be able to better choose the most reliable and highly predicted ones. Instead, *Precision* does not incorporate the number of relevant items in its equation: it just incorporates the constant *N* value. Finally, the higher the number of recommendations, the lower the RNCF recommendation improvement: simply, the higher the value of *N*, the faster the reliable items are finished, and the *Recall* improvement decreases.

Figures 8 and 9, corresponding to the *MovieLens* and the *FilmTrust* experiments, show the same main behavior of the discussed Figure 7 (*Netflix**). It is remarkable that the bigger the dataset, the higher the proposed neural architecture improvement. This is the expected result, since big datasets offer better opportunities to find items that are, at the same time, highly predicted and highly reliable. By selecting these items, our architecture can improve recommendations. This way, we can claim that the proposed RNCF deep learning architecture is scalable in the sense that it works better when the dataset size increases.

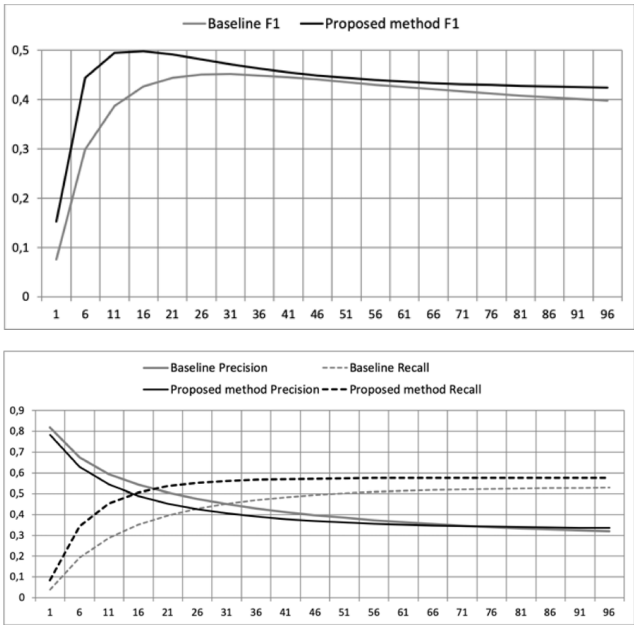

Figure 8. *MovieLens* recommendation accuracy. Top chart: *F1* comparative between the NCF baseline and the proposed RNCF. Chart below: *Precision* and *Recall* comparative between the NCF baseline and the proposed RNCF; *x*-axis: number of recommendations (*N*).

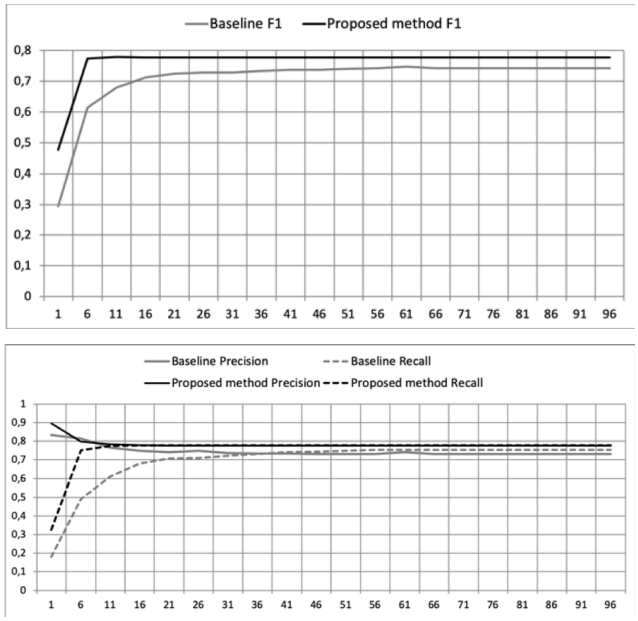

Figure 9. *FilmTrust* recommendation accuracy. Top chart: *F1* comparative between the NCF baseline and the proposed RNCF. Chart below: *Precision* and *Recall* comparative between the NCF baseline and the proposed RNCF; *x*-axis: number of recommendations (*N*).

4. Discussion

In the same way that more accurate or less accurate predictions exist, there are more reliable or less reliable predictions. The proposed deep learning architecture uses its first levels of abstraction to learn and to predict the reliability of each prediction. Subsequently, the architecture learns the non-linear relationships between accuracy and reliability; in this way, we implement a filter that selects the predictions that are most likely to satisfy the user, taking into account each prediction's reliability. This set of predictions determines the most accurate recommendations.

The results show a strong improvement in the quality of predictions. They also show an overall improvement in the recommendation quality: results from the three tested datasets show the improvement of the *F1* measure. Most of the *F1* improvement is due to its *Recall* component, since the higher the number of relevant ratings a user has issued, the greater the possibilities of the proposed method to select the most reliable predictions. On the other hand, as expected, when a large number of items is recommended, the *Recall* improvement decreases, since the most reliable and accurate items run out. For the above reasons, the experiments also show a recommendation quality improvement as the dataset size increases.

Since the proposed deep learning architecture only uses ratings as input data, it can be seen as a collaborative filtering black box that can be enhanced applying hybrid approaches: demographic, social, context-aware, content-based, etc. Finally, as future work, we propose (1) improving the proposed architecture results by enhancing its reliability extraction stage and (2) adding a wide learning component to the neural architecture, providing memorization by catching simple relations from multiple features.

Author Contributions: Authors J.B., S.A. and A.H. contributed equally to this work. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Dong, X.; Yu, L.; Wu, Z.; Sun, Y.; Yuan, L.; Zhang, F. A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 1309–1315. Available online: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14676 (accessed on 12 February 2017).
2. Pazzani, M. A Framework for Collaborative, Content-Based and Demographic Filtering. *Artif. Intell. Rev.* **1999**, *13*, 393–408. [CrossRef]
3. Ebesu, T.; Fang, Y. Neural Citation Network for Context-Aware Citation Recommendation. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 1093–1096. [CrossRef]
4. Musto, C.; Greco, C.; Suglia, A.; Semeraro, G. Askme Any Rating: A Content-Based Recommender System Based on Recurrent Neural Networks. In Proceedings of the IIR, Venezia, Italy, 7–10 May 2016.
5. Wang, X.; He, X.; Nie, L.; Chua, T.-S. Item Silk Road. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 185–194. [CrossRef]
6. Yang, C.; Bai, L.; Zhang, C.; Yuan, Q.; Han, J. Bridging Collaborative Filtering and Semi-Supervised Learning. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 20–24 August 2017; pp. 1245–1254. [CrossRef]
7. Yi, B.; Shen, X.; Zhang, Z.; Shu, J.; Liu, H. Expanded Autoencoder Recommendation Framework and Its Application in Movie Recommendation. In Proceedings of the 2016 10th International Conference on Software, Knowledge, Information Management & Applications (SKIMA), Chengdu, China, 15–17 December 2016; pp. 298–303. [CrossRef]
8. Chu, W.-T.; Tsai, Y.-L. A hybrid recommendation system considering visual information for predicting favorite restaurants. *World Wide Web* **2017**, *20*, 1313–1331. [CrossRef]
9. He, R.; McAuley, J. Ups and Downs. In Proceedings of the 25th International Conference on World Wide Web, Montreal, QC, Canada, 11–15 April 2016; pp. 507–517. [CrossRef]
10. Hsieh, C.-K.; Yang, L.; Wei, H.; Naaman, M.; Estrin, D. Immersive Recommendation. In Proceedings of the 25th International Conference on World Wide Web, Montreal, QC, Canada, 11–15 April 2016; pp. 51–62. [CrossRef]
11. Li, K.; Zhou, X.; Lin, F.; Zeng, W.; Alterovitz, G. Deep Probabilistic Matrix Factorization Framework for Online Collaborative Filtering. *IEEE Access* **2019**, *7*, 56117–56128. [CrossRef]

12. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep Learning Based Recommender System. *ACM Comput. Surv.* **2019**, *52*, 1–38. [CrossRef]
13. Mu, R.; Zeng, X.; Han, L. A Survey of Recommender Systems Based on Deep Learning. *IEEE Access* **2018**, *6*, 69009–69022. [CrossRef]
14. Wang, X.; Yu, L.; Ren, K.; Tao, G.; Zhang, W.; Yu, Y.; Wang, J. Dynamic Attention Deep Model for Article Recommendation by Learning Human Editors' Demonstration. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 20–24 August 2017; pp. 2051–2059. [CrossRef]
15. Liang, D.; Zhan, M.; Ellis, D.P. Content Aware Collaborative Music Recommendation Using Pre-Trained Neural Networks. In Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR, Málaga, Spain, 26–30 October 2015; pp. 295–301. [CrossRef]
16. Wang, X.; Wang, Y. Improving Content-Based and Hybrid Music Recommendation Using Deep Learning. In Proceedings of the 22nd ACM International Conference on Multimedia, Orlando, FL, USA, 3–7 November 2014; pp. 627–636. [CrossRef]
17. Chen, J.; Zhang, H.; He, X.; Nie, L.; Liu, W.; Chua, T.-S. Attentive Collaborative Filtering. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 335–344. [CrossRef]
18. Chen, X.; Zhang, Y.; Ai, Q.; Xu, H.; Yan, J.; Qin, Z. Personalized Key Frame Recommendation. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 315–324. [CrossRef]
19. Tang, J.; Wang, K. Personalized Top-N Sequential Recommendation via Convolutional Sequence Embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Los Angeles, CA, USA, 5–9 February 2018; pp. 565–573. [CrossRef]
20. Suglia, A.; Greco, C.; Musto, C.; De Gemmis, M.; Lops, P.; Semeraro, G. A Deep Architecture for Content-Based Recommendations Exploiting Recurrent Neural Networks. In Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, Bratislava, Slovakia, 9–12 July 2017; pp. 202–211. [CrossRef]
21. Alashkar, T.; Jiang, S.; Wang, S.; Fu, Y. Examples-Rules Guided Deep Neural Network for Makeup Recommendation. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 941–947.
22. Zhang, S.; Yao, L.; Xu, X. AutoSVD++. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 957–960. [CrossRef]
23. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Toronto, ON, Canada, 19–25 August 2017; pp. 1725–1731.
24. 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 Companion, Perth, Australia, 3–7 April 2017; pp. 173–182. [CrossRef]
25. Dziugaite, G.K.; Roy, D.M. Neural Network Matrix Factorization. *arXiv* **2015**, arXiv:1511.06443.
26. Cheng, H.-T.; Ispir, M.; Anil, R.; Haque, Z.; Hong, L.; Jain, V.; Liu, X.; Shah, H.; Koc, L.; Harmsen, J.; et al. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10. [CrossRef]
27. Yuan, W.; Wang, H.; Baofang, H.; Wang, L.; Wang, Q. Wide and Deep Model of Multi-Source Information-Aware Recommender System. *IEEE Access* **2018**, *6*, 49385–49398. [CrossRef]
28. Rendle, S. Factorization Machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, NSW, Australia, 13–17 December 2010; pp. 995–1000. [CrossRef]
29. Yin, R.; Li, K.; Zhang, G.; Lu, J. A deeper graph neural network for recommender systems. *Knowl.-Based Syst.* **2019**, *185*, 105020. [CrossRef]
30. Nassar, N.; Jafar, A.; Rahhal, Y. A novel deep multi-criteria collaborative filtering model for recommendation system. *Knowl.-Based Syst.* **2020**, *187*, 104811. [CrossRef]
31. Bobadilla, J.; Ortega, F.; Gutiérrez, A.; Alonso, S. Classification-based Deep Neural Network Architecture for Collaborative Filtering Recommender Systems. *Int. J. Interact. Multimed. Artif. Intell.* **2020**, *6*, 68. [CrossRef]

32. Zhu, B.; Ortega, F.; Bobadilla, J.; Gutierrez, A. Assigning reliability values to recommendations using matrix factorization. *J. Comput. Sci.* **2018**, *26*, 165–177. [CrossRef]
33. Harper, F.; Konstan, J. The MovieLens Datasets. *ACM Trans. Interact. Intell. Syst.* **2015**, *5*, 1–19. [CrossRef]
34. Golbeck, J.; Hendler, J. FilmTrust: Movie Recommendations Using Trust in Web-Based Social Networks. In Proceedings of the 3rd IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, 8–10 Januaty 2006; Volume 1, pp. 282–286. [CrossRef]
35. Ortega, F.; Zhu, B.; Bobadilla, J.; Hernando, A. CF4J: Collaborative filtering for Java. *Knowl.-Based Syst.* **2018**, *152*, 94–99. [CrossRef]