# Classification-based Deep Neural Network Architecture for Collaborative Filtering Recommender Systems

Jesús Bobadilla*, Fernando Ortega, Abraham Gutiérrez, Santiago Alonso

Universidad Politécnica de Madrid, Carretera de Valencia Km 7, 28031 Madrid (Spain)

Received 17 December 2019 | Accepted 31 January 2020 | Published 18 February 2020

## Abstract

This paper proposes a scalable and original classification-based deep neural architecture. Its collaborative filtering approach can be generalized to most of the existing recommender systems, since it just operates on the ratings dataset. The learning process is based on the binary relevant/non-relevant vote and the binary voted/non-voted item information. This data reduction provides a new level of abstraction and it makes possible to design the classification-based architecture. In addition to the original architecture, its prediction process has a novel approach: it does not need to make a large number of predictions to get recommendations. Instead to run forward the neural network for each prediction, our approach runs forward the neural network just once to get a set of probabilities in its categorical output layer. The proposed neural architecture has been tested by using the MovieLens and FilmTrust datasets. A state-of-the-art baseline that outperforms current competitive approaches has been used. Results show a competitive recommendation quality and an interesting quality improvement on large number of recommendations, consistent with the architecture design. The architecture originality makes it possible to address a broad range of future works.

Keywords

Deep Learning, Neural Classification, Neural Collaborative Filtering, Recommender Systems, Scalable Neural Architecture.

DOI: 10.9781/ijimai.2020.02.006

## I. Introduction

Recommender Systems (RS) [1] play an important role to address the information overload in Internet. RS can use diverse sources of information: votes from users to items, purchased products or consumed services, social information, geographical coordinates, demographic information, etc. The most accurate RS are the Collaborative Filtering-based (CF) ones; they learn from all the existing implicit or explicit information about how users vote or consume items. It is usual to merge RS types (hybrid RS) [2] to improve accuracy results. CF RS can be reinforced by means of demographic [3], context-aware [4], content-based [5] and social [6] information. The scope of the RS has extended and currently covers an endless number of targets: tourism [7], films [8], networks [6], restaurants [9], e-learning, fashion [10], news [11], healthcare [12], etc.

CF RS kernels have been implemented by using machine learning methods: memory-based and model-based ones. KNN was the main memory-based method, but currently model-based algorithms are used due to their accuracy superiority. Matrix Factorization (MF) is the most implemented approach, since it provides accurate recommendations, it is easy to understand, and it obtains a good performance. There are several MF variations such as PMF [13] [14], BNMF [15], BPR [16] and eALS [17]. MF extracts the complex relations between items and users and codes them into a reduced number of hidden factors. Predictions are obtained by making the dot product of users and items. A MF drawback is the linearity of the dot product: it does not allow to accurately combine hidden factors to provide the most suitable predictions.

* Corresponding author.

E-mail address: jesus.bobadilla@upm.es

RS research is heading towards solutions based on deep learning [18] [19]. Some of the approaches make use of non-collaborative data: images information [9] [10], text [11] [20], music [21] [22], videos [23] [24], session ID [25] [26], etc. Depending on the type of data, different deep learning architectures are used: CNN [27] [28], RNN [29] [30], MLP [31] [4], Autoencoder [8] [32], etc. The previous examples implement specific content-based approaches or RS hybrid solutions. The CF RS core is based on the ratings to items casted by users. The deep learning solutions to this challenge can be classified as: a) Neural Collaborative Filtering (NCF) [33], and b) Deep Factorization Machines (deepFM) [34]. NCF are usually based on dual neural networks to simultaneously process users and items information. The two NCF references are: 1) Neural Network Matrix Factorization (NNMF) [35] and Neural Collaborative Filtering (NCF) [33].

NCF reports better performance than NNMF, so we put the focus on it. Fig. 1 shows the NCF architecture; as explained, it takes the sparse user and item raw data vectors. Above the sparse layer is the fully connected embedding layer that projects the sparse representation to a dense vector. Then a Multilayer Perceptron (MLP) combines the two pathways features by concatenating them.

A deep learning type of architecture used in CF combines wide & deep learning [36] [37]; the wide component is a single-layer perceptron, whereas the deep component is a MLP. Combining the two learning techniques enables the RS to capture both generalization and memorization. The wide component captures simple data relations (memorization), whereas the deep component can make more valuable abstractions (generalization). Our proposed method can be extended to the wide & deep learning architecture and it will be recommended as future work. The DeepFM architecture joins a wide and a deep component to share the same input raw feature vector. It allows to simultaneously learn low and high order feature interactions from the input raw features. DeepFM consists of two blocks: FM component and deep component. The FM component is a factorization machine [38] to learn feature interactions. DeepFM achieves a 0.48% (Company dataset) and a 0.33% (Criteo dataset) of accuracy gain compared to NCF. This is a small performance improvement, so we will use the best known NCF as baseline.

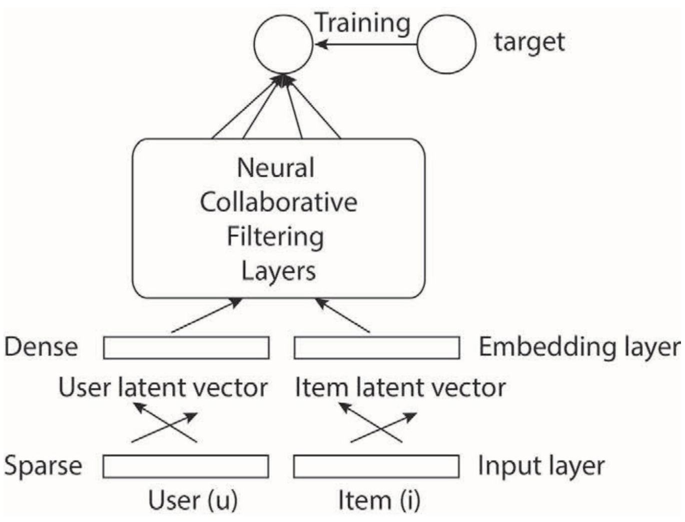

Fig. 1. NCF baseline architecture.

Classical methods make use of MF to make predictions. Some neural collaborative filtering methods replace the MF linear dot product used to predict for a non-linear MLP stage. Neural network methods that do not use MF often are based on deep hybrid models: CNN, RNN, etc. that exploit additional data to the rating matrix, usually not available for most of the existing RS. Wide and multi-view deep learning architectures have been used as pure neural network RS approaches; their main drawback is the size of the wide layers that jeopardize scalability. The proposed method architecture is scalable, because it is just based on the RS item dimensionality, relying the huge RS user dimensionality to the number of samples used to train the model.

As we have seen, existing RS deep neural approaches make use of a MF level or an embedding layer to catch the item and user latent vectors. They are regression-based architectures: for each user, recommendations are chosen from its N best predictions. Then, to recommend a user it is necessary to run I predictions (I is the number of items in the RS). Our proposed deep neural architecture does not need an external machine learning level, as the MF one. It does not use, either, explicit embedding layers. Its original architecture comes from the fact that it makes recommendations by means of a classification process instead a regression one. The classification approach requires a different design that the existing ones: it will learn by using categorical labels instead numerical values. Additionally, we are committed to create a simple and scalable architecture. To meet the stated aims we have decided to renounce to the numerical rating values and predictions: we will only make use of the binary relevant/non-relevant rating and the binary voted/non-voted information. As the reader can imagine, this decision opens a new range of architectural choices. Of course, a key question must be answered: will the accuracy of recommendations decrease significantly? State of art papers indicate that accuracy is not significantly affected, although it could vary depending on the dataset. From [39], the graph in Fig. 2 is provided.

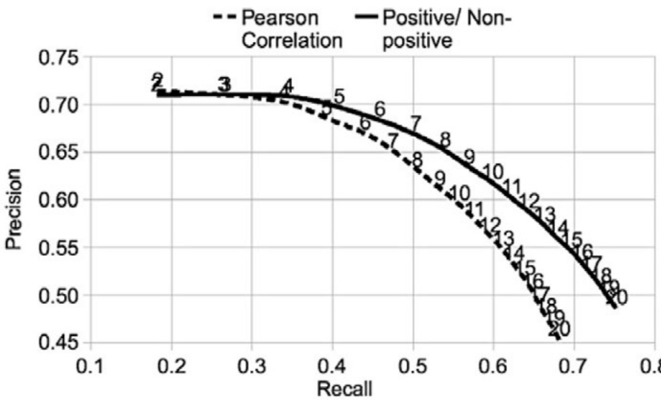

Fig. 2. MovieLens 1M Precision/Recall obtained by transforming all 4 and 5 votes into relevant votes, and 1, 2 and 3 votes into non-relevant votes, compared to the results obtained using the numerical values [39].

We will call our proposed method as Neural Classification-based Collaborative Filtering (NCCF), and we are going to motivate this approach by explaining, with the help of representative figures: a) Its innovative design, b) Its scalability basis, and c) Its straightforward method to get predictions of each user. Fig. 3 shows the evolution from MF methods to the proposed NCCF. First of all, both the MF and the NCF models (Fig. 3a & 3b) are regression-based: they provide prediction values, and the RS recommends the highest N predictions. Instead, NCCF (Fig. 3c) is based on a classification neural network. Furthermore, NCCF is not based on a previous machine learning MF process or an embedding layer; instead, raw rating data is used avoiding the necessity of feature engineering stages [18]. It is important to highlight that NCCF raw data is made up by user vectors, where each vector contains items information. NCCF does not combine user vectors with item vectors, such as the NCF baseline does. It does not combine, either, user/item vectors with dense embedding coming from a factorization machine. This make the proposed approach simpler and more scalable than the current baselines.

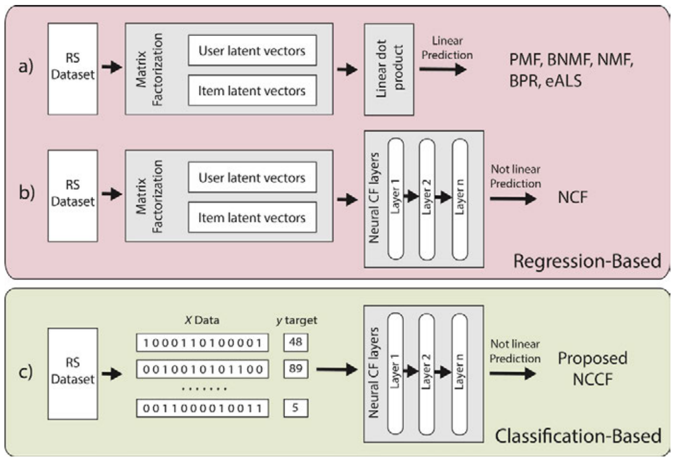

Fig. 3. Comparative between the proposed method c) and the baseline ones: a) & b).

To improve understanding of the proposed method advantages, Fig. 4 shows an outline involving the learning parameters of both the proposed and the baseline models. The MF machine learning algorithm needs to learn a set of F factors for each user (U) and for each item (I) in the dataset. F usually ranges from 12 to 40 and, as shown in Fig. 4a, in commercial RS the number of users is much larger than the number of items. The necessary MF parameters is: (U+I)F. NCF approaches are built on MF, so Fig. 4b reproduces the MF parameters shown in Fig. 4a. Additionally, the MLP needs learning parameters; if the first inner layer contains n neurons (Ln), we need (U+I+Ln)F to run the MF model and to process the first MLP layer. The following layers will need more learning parameters (Lm), but usually Lm (number of neurons in the following layer) is not a high number. The output layer just needs Lm parameters, since we just need a neuron to obtain the regression value (the prediction). Overall, the NCF baseline, when the MLP has two inner layers, needs (U+I+Ln)F+(Ln+1)Lm parameters. The NCCF proposed method is designed to be scalable: it makes use of a tiny portion of the necessary parameters of the NCF baseline. Even although it uses an output layer of dimension I, it just needs I(Lm+Ln)+LmLn learning parameters.

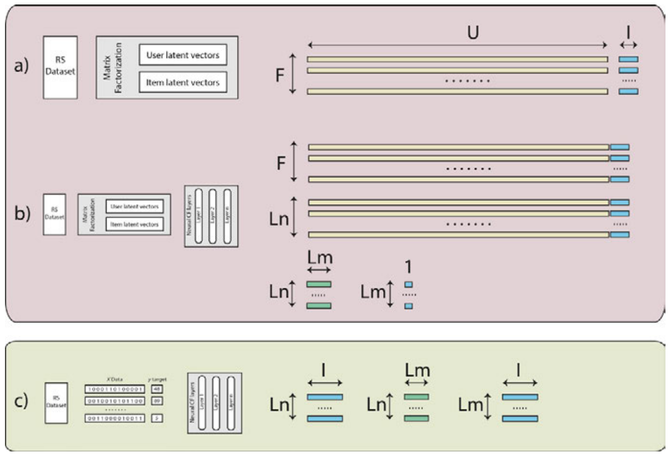

Fig. 4. Involved learning parameters in both the proposed c) and the baseline models: a) & b).

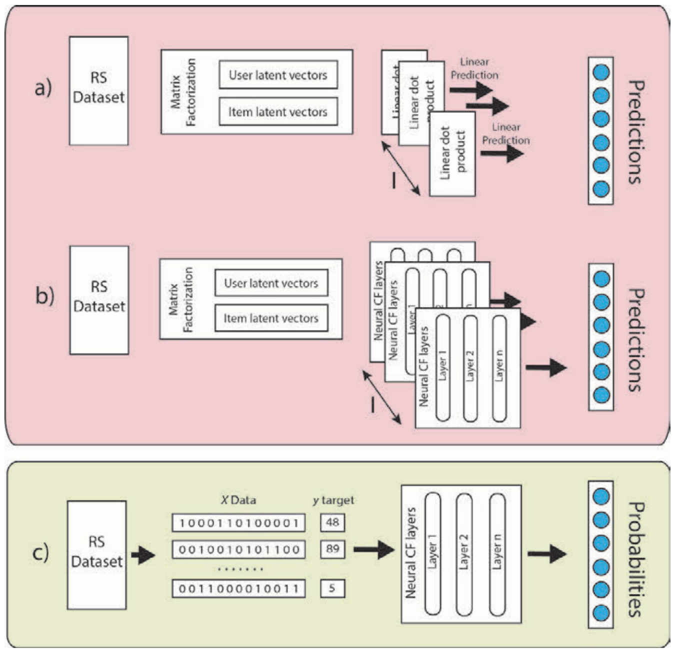

Fig. 5. Process to predict items values for each user in both the proposed c) and the baseline models: a) & b).

Another relevant improvement of the proposed method is its fully neural network processing: it allows to model the non-linear relations from the raw data to the result (predictions). It does not start from previous machine learning results that could have lost some of the rich data dependencies. Furthermore, NCCF processes RS predictions in a different way that the baselines do, as it is shown in Fig. 5: in order to provide N recommendations to a user, the MF approach (Fig. 5a) needs to run I dot products (if we dismiss the items voted for the user). The NCF baseline (Fig. 5b) must run forward I times its MLP: each one providing the pair <user,item> to the neural network. Instead, to provide recommendations to a user, the proposed NCCF (Fig. 5c) just have to run forward once its MLP. Since we are working with a categorical target, the output layer will provide I classification probabilities. To make N recommendations we just have to select the N highest probabilities; this is a simple and straightforward operation.

## II. Proposed Method

This section explains the proposed methods both in a conceptual way and in its formalized version. Each explanation type is made in a separated subsection. First subsection is centered in the concepts and the motivation of each proposed approach; a data-toy running example is provided as well as the proposed architecture explanation through a representative figure. Second subsection provides the proposed methods formalization to facilitate their implementation; formalization is explained by means of a data-toy running example.

### A. Concepts

We propose two related methods: a) the short one, and b) the long one. The short method is based on the relevant ratings of the dataset: ratings that exceed the value of an established threshold (e.g.: 4 stars in a RS where users can vote from 1 to 5 stars). We will lose the numerical fine grain of the votes and we renounce to make numerical predictions (e.g.: we recommend you this item with the 4.2 stars value). We will not work with the usual rating data; instead we make an abstraction by selecting the binary relevant/non-relevant information. Table I shows a data-toy example of CF RS dataset (“Original dataset”) and the abstracted relevant vs. non-relevant information (“Proposed short and long methods preliminary information”). Our aim is to be able to accurately recommend RS items without making use of the usual numerical prediction that in fact is a regression process. We will make RS recommendation by means of a classification process, and the proposed classification process will be mainly based on the rating relevancy abstraction.

TABLE I. Preliminary Information

| Original dataset | | | | | | |
|---|---|---|---|---|---|---|
| *rating* | i0 | i1 | i2 | i3 | i4 | i5 |
| u0 | • | 4 | • | 5 | 2 | • |
| u1 | 3 | 1 | 5 | 4 | • | 5 |
| u2 | 1 | • | 4 | • | • | 4 |
| Proposed short and long methods preliminary information | | | | | | |
| *relev.* | i0 | i1 | i2 | i3 | i4 | i5 |
| u0 | 0 | 1 | 0 | 1 | 0 | 0 |
| u1 | 0 | 0 | 1 | 1 | 0 | 1 |
| u2 | 0 | 0 | 1 | 0 | 0 | 1 |
| Proposed long method preliminary information | | | | | | |
| *relev.* | i0 | i1 | i2 | i3 | i4 | i5 |
| u0 | 0 | 1 | 0 | 1 | 0 | 0 |
| u1 | 0 | 0 | 1 | 1 | 0 | 1 |
| u2 | 0 | 0 | 1 | 0 | 0 | 1 |
| *voted* | i0 | i1 | i2 | i3 | i4 | i5 |
| u0 | 0 | 1 | 0 | 1 | 1 | 0 |
| u1 | 1 | 1 | 1 | 1 | 0 | 1 |
| u2 | 1 | 0 | 1 | 0 | 0 | 1 |

Preliminary information for both the short and long proposed methods. The value “•” means non-voted item. Threshold = 4.

By converting ratings to relevant/non-relevant data we lose a type of information: voted or non-voted item. This is due to the inherent sparse nature of CF RS: users can only vote a reduced number of the available items because we ca not see all the films, buy all the products or consume all the services that a modern RS provides. Our proposed long method incorporates the voted/non-voted information and adds it to the short method information; Table I ("Proposed long method preliminary information") shows the concept. Neural networks are able to relate both types of information to make their job (in this case to accurately classify); this means that using the long model it is possible to discern from: a) relevant vote, b) non-relevant vote, and c) not voted. The short method approach works without the 'not voted' information.

The long model approach has the advantage of working with more information than the short model, and it is expected to provide better results. Its main drawback is the size of its necessary input and output neural network layers: it will need twice input and output neurons than the short method approach (since it provides two types of information: voted/non-voted and relevant/non-relevant). The key issue is the obtained balance between performance and accuracy. Accuracy mainly will come from the relevant/non-relevant information, whereas voted/non-voted data can improve results. While the "voted" items information meaning is clear, the "non-voted" items information is a controversial subject: non-voted items meanings can be: a) the user does not know the item, b) the user knows the item but he/she does not know if he/she will like it, c) the user knows that he/she does not like the item, but he/she did not vote it, and d) the user knows that he/she likes the item, but he/she did not vote it. RS aim is centered in the a) and b) options. Our short method loses the semantic difference between not to vote an item and to vote this item as non-relevant. The lost information will be more or less important depending on the relative significance of cases c) and d). The proposed short method assigns the meaning "voted as not relevant" to the non-voted items, so case c) does not affect to the proposed short model accuracy. The proposed long method does not assign a "negative" semantic to the non-voted items, as the short model does; this means that the long method correctly manages the case d), whereas the short method does not. The accuracy difference between the proposed methods rely on the significance of the case d) in each RS: if the users are more active, this case importance decreases and vice versa. Published studies show that using binary relevancy information instead numerical votes does not significantly change accuracy results [39].

On the basis of the preceding information (Table I) both proposed methods create a set of samples to train and test the classification-based RS. For each relevant vote (numbers 1 in the relevant sections of Table I) we create a sample: data and target, thus the model number of samples will be the existing number of relevant ratings in the dataset. Each dataset user will generate a number of samples equal to her number of relevant ratings. *The key concept is to train our model with the relevant ratings of each user except the first one (this is the X data sample), and to use this rating as a target (this is the y label of the sample). This process is repeated until the last relevant rating of the user is reached*. As an example, the obtained samples from the sort method information in Table I are shown in Table II.

TABLE II. X Data and Y Label Samples from Table I; Short Method

| X | i0 | i1 | i2 | i3 | i4 | i5 | | y | |
|---|---|---|---|---|---|---|---|---|---|
| $X_{0,1}$ | 0 | 0 | 0 | 1 | 0 | 0 | | $Y_{0,1}$ | 1 |
| $X_{0,3}$ | 0 | 1 | 0 | 0 | 0 | 0 | | $Y_{0,3}$ | 3 |
| $X_{1,2}$ | 0 | 0 | 0 | 1 | 0 | 1 | | $Y_{1,2}$ | 2 |
| $X_{1,3}$ | 0 | 0 | 1 | 0 | 0 | 1 | | $Y_{1,3}$ | 3 |
| $X_{1,5}$ | 0 | 0 | 1 | 1 | 0 | 0 | | $Y_{1,5}$ | 5 |
| $X_{2,2}$ | 0 | 0 | 0 | 0 | 0 | 1 | | $Y_{2,2}$ | 2 |
| $X_{2,5}$ | 0 | 0 | 1 | 0 | 0 | 0 | | $Y_{2,5}$ | 5 |

Our proposed methods provide valuable information to make a RS classification: each user relevant rating can be predicted from the set of the remaining relevant ratings from this user. Additionally, each user relevant rating can be predicted from the rest of users' data. In Table II we can observe how the model can learn that target 3 (y_0,3 and y_1,3) can be predicted when i1 is relevant or when i2 and i5 are relevant. In the same way, target 5 prediction will be particularly probable when i2 is relevant, and this is an information coming from users 1 and 2. Current datasets contain hundreds of thousands or millions relevant ratings, thus the proposed methods will provide rich models to be used in artificial intelligence classification algorithms.

A fundamental decision design in the proposed methods is to maintain the dimensionality of the samples in reasonable sizes; we want: a) A large number of samples to train and test the model, and b) A reasonable size of the samples in the model in order to maintain the performance (time consuming) of the classification algorithms. Since commercial RS have much more users than items, to make these requirements compatible, we must use the RS user dimension to generate samples, and the RS item dimension to stablish the samples size. Some RS have a large number of items; in such cases, the proposed short model can be particularly appropriated since it reduces by half the samples size.

Fig. 6 shows a generic neural network model implementing these paper's proposed methods. Each input data column (on the left) represents a sample. Our proposed methods provide the samples; e.g.: each row in Table II feeds the corresponding column sample in the proposed short method neural network architecture. The proposed long method adds the voted/non-voted information to the neural network. We also provide the categorical labels, necessary for the classification task: there are as many classes as items in the RS (I number of items in Fig. 6). Once the neural network has been trained, for each new input sample (new user) the neural network predicts a set of probabilities: the probability of each class (item) to be classified from the sample data (user). We just have to take the N highest probabilities to obtain the N recommendations; this process is drawn on the right of both methods in Fig. 6.

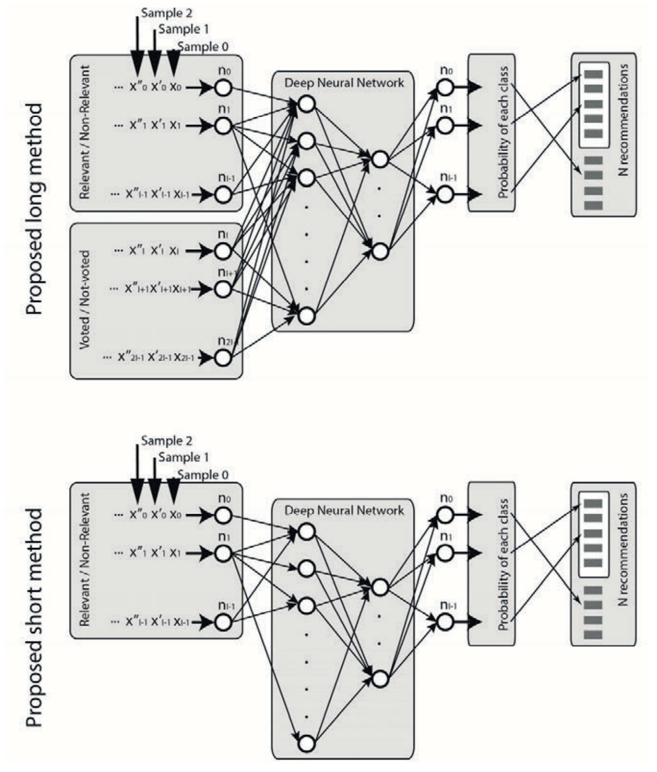

Fig. 6. Proposed neural network architecture for both the short and long methods.

## B. Formalization and Data-Toy Example

Let *I* be the set of the RS items.

Let *U* be the set of the RS users.

Let *V = {1, 2, 3, 4, 5, •}* be the set of available ratings. • means empty rating.

Let *R* be the set of the dataset ratings:

$$R = \{< u, i, v > \ \forall i \in I, \forall u \in U, v \in V\} \quad (1)$$

Let $R_u$ be the set of the dataset ratings casted by user u:

$$R_u = \{< i, v > \ \forall i \in I, v \in V\} \quad (2)$$

Let $R_{u,i}$ be the rating casted by user u to the item i:

$$R_{u,i} = v \Leftrightarrow < u, i, v > \in R_u \quad (3)$$

Let *I'* be an ordered list of the *I* elements.

Let $\theta \in V - \{\bullet\}$ be the rating relevance threshold (4)

Let $D_u$ be the intermediate data for the proposed method short version. $D_u$ differentiates the relevant and not relevant ratings casted by user u.

$$D_u = \{< u, i, 1 > \ \forall i \in I' \mid R_{u,i} \neq \bullet \ and \ R_{u,i} \geq \theta\} \cup \{< u, i, 0 > \mid R_{u,i} = \bullet \ or \ (R_{u,i} \neq \bullet \ and \ R_{u,i} < \theta)\} \quad (5)$$

where $D_{u,i}$ is the relevance of the $R_{u,i}$ vote:

$$D_{u,i} = c \Leftrightarrow < u, i, c > \in D_u \quad (6)$$

Let $X_{u,s}$ be the sample data *s* from $D_u$, and $Y_{u,s}$ the target data s from $D_u$:

$$X_{u,s} = \{D_u - < u, s, 1 >\} \cup \{< u, s, 0 >\}, \forall s \in I', \mid D_{u,s} = 1 \quad (7)$$

$$y_{u,s} = s \ \Leftrightarrow < u, s, 1 > \in \ X_{u,s} \quad (8)$$

Finally, the X and Y data to train the proposed short method is:

$$X = \{X_{u,s}: \forall u \in U, \forall s \in I'\} \quad (9)$$

$$Y = \{y_{u,s}: \forall u \in U, \forall s \in I'\} \quad (10)$$

Let $D_u || D'_u$ be the intermediate data for the proposed method long version. $D'_u$ differentiates the user u voted and not voted items.

$$D'_u = \{< u, i, 1 > \ \forall i \in I' | R_{u,i} \neq \bullet\} \cup \{< u, i, 0 > \ \forall i \in I' | R_{u,i} = \bullet\} \quad (11)$$

$$D'_{u,i} = i \Leftrightarrow < u, i, 1 > \in D'_u \quad (12)$$

Let $X'_{u,s}$ be the sample data s from $D'_u$ , and $y'_{u,s}$ the target data s from $D'_u$:

$$X'_{u,s} = \{D'_u - < u, s, 1 >\} \cup \{< u, s, 0 >\}, \ \forall s \in I' \mid D'_{u,s} = 1 \quad (13)$$

$$y'_{u,s} = \ s \Leftrightarrow < u, s, 1 > \in X'_{u,s} \quad (14)$$

Finally, the *X'* and *Y'* data to train the proposed short method is:

$$X' = \{X_{u,s}: \forall u \in U, \forall s \in I\} \quad (15)$$

$$Y' = \{y_{u,s}: \forall u \in U, \forall s \in I\} \quad (16)$$

Once the model has been trained by using the *X* and *Y* sets or the *X'* and *Y'* sets, we obtain a result set *W* with cardinality *I*. *W* contains the probability of each item to be recommended. These results are predicted by using a neural network containing I "softmax" output neurons. The N expected recommendations Z will be the N highest results from W.

$$Z = \{w_i \in W \big| w_i \geq w_j, w_j \in Z^c\}, \ \#Z \ = \mathrm{N} \quad (17)$$

We will explain the previous equations by means of the data-toy example on Table I. It shows the ratings casted by users 0, 1 and 2 on items 0 to 5. The • symbol means not voted. This data-toy uses the V set definition from the above equations

From the data-toy, the sets and lists from previous formalization are:

$$R = \{< 0{,}0{,}\bullet >, < 0{,}1{,}4 >, < 0{,}2{,}\bullet >, < 0{,}3{,}5 > \ \ldots < 2{,}4{,}\bullet >, < 2{,}5{,}4 >\} \quad (1)$$

The number that accompanies each expression refers to the applied equation in previous formalization.

The set of the dataset ratings casted by users 0, 1 and 2:

$$R_0 = \{< 0{,}0{,}\bullet >, < 0{,}1{,}4 >, < 0{,}2{,}\bullet >, < 0{,}3{,}5 >, < 0{,}4{,}2 >, < 0{,}5{,}\bullet >\} \quad (2)$$

$$R_1 = \{< 1{,}0{,}3 >, < 1{,}1{,}1 >, < 1{,}2{,}5 >, < 1{,}3{,}4 >, < 1{,}4{,}\bullet >, < 1{,}5{,}5 >\} \quad (2)$$

$$R_2 = \{< 2{,}0{,}1 >, < 2{,}1{,}\bullet >, < 2{,}2{,}4 >, < 2{,}3{,}\bullet >, < 2{,}4{,}\bullet >, < 2{,}5{,}4 >\} \quad (2)$$

As an example, we can see that:

$$R_{0,0} = \bullet, R_{0,1} = \ 4, R_{1,2} = \ 5, R_{1,5} = \ 5, R_{2,2} = \ 4, \text{etc.} \quad (3)$$

We will stablish the $\theta = 4$ threshold (4)

The set of information for the training and testing short proposed method is:

$$D_0 = \{< 0{,}0{,}0 >, < 0{,}1{,}1 >, < 0{,}2{,}0 >, < 0{,}3{,}1 >, < 0{,}4{,}0 >, < 0{,}5{,}0 >\} \quad (5)$$

$$D_1 = \{< 1{,}0{,}0 >, < 1{,}1{,}0 >, < 1{,}2{,}1 >, < 1{,}3{,}1 >, < 1{,}4{,}0 >, < 1{,}5{,}1 >\} \quad (5)$$

$$D_2 = \{< 2{,}0{,}0 >, < 2{,}1{,}0 >, < 2{,}2{,}1 >, < 2{,}3{,}0 >, < 2{,}4{,}0 >, < 2{,}5{,}1 >\} \quad (5)$$

The *X* data and the y targets to the classification neural network is extracted from $D_u$. Each $n_i$ column feeds a neuron of the neural network input layer. Each $u_i$ row is a sample for the training or the testing process.

From equations (7) to (10) we obtain the input *X* and the output *Y* values to feed and train the neural network. This is the proposed short method to arrange data in order to make RS classification and then RS recommendation.   shows the obtained results. As it can be seen, the method provides the categorical values for the labels.

The additional set of information for the training and testing long proposed method tell us about the voted and not voted items:

$$D'_0 = \{< 0{,}0{,}0 >, < 0{,}1{,}1 >, < 0{,}2{,}0 >, < 0{,}3{,}1 >, < 0{,}4{,}1 >, < 0{,}5{,}0 >\} \quad (11)$$

$$D'_1 = \{< 1{,}0{,}1 >, < 1{,}1{,}1 >, < 1{,}2{,}1 >, < 1{,}3{,}1 >, < 1{,}4{,}0 >, < 1{,}5{,}1 >\} \quad (11)$$

$$D'_2 = \{< 2{,}0{,}1 >, < 2{,}1{,}0 >, < 2{,}2{,}1 >, < 2{,}3{,}0 >, < 2{,}4{,}0 >, < 2{,}5{,}1 >\} \quad (11)$$

The proposed long method concatenates both $D_u$ and $D'_u$: $D_u || D'_u$ (Table IV).

From equations (13) to (16) we obtain the input X' and output Y' values to feed the neural network training. This is the proposed long model to arrange data in order to make RS classification and then RS recommendation. We can see it in Table V. The categorical output values are the ones shown in Table III.

TABLE III. Data-Toy Example: Classification Data and Categorical Labels for the Proposed Short Method

| $X$ | Input neurons | | | | | |
|---|---|---|---|---|---|---|
| $X_{u,s}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
| $X_{0,1}$ | 0 | 0 | 0 | 1 | 0 | 0 |
| $X_{0,3}$ | 0 | 1 | 0 | 0 | 0 | 0 |
| $X_{1,2}$ | 0 | 0 | 0 | 1 | 0 | 1 |
| $X_{1,3}$ | 0 | 0 | 1 | 0 | 0 | 1 |
| $X_{1,5}$ | 0 | 0 | 1 | 1 | 0 | 0 |
| $X_{2,2}$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $X_{2,5}$ | 0 | 0 | 1 | 0 | 0 | 0 |

| $y$ | Output neurons | | | | | |
|---|---|---|---|---|---|---|
| $y_{u,s}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
| $y_{0,1}$ | 0 | 1 | 0 | 0 | 0 | 0 |
| $y_{0,3}$ | 0 | 0 | 0 | 1 | 0 | 0 |
| $y_{1,2}$ | 0 | 0 | 1 | 0 | 0 | 0 |
| $y_{1,3}$ | 0 | 0 | 0 | 1 | 0 | 0 |
| $y_{1,5}$ | 0 | 0 | 0 | 0 | 0 | 1 |
| $y_{2,2}$ | 0 | 0 | 1 | 0 | 0 | 0 |
| $y_{2,5}$ | 0 | 0 | 0 | 0 | 0 | 1 |

TABLE IV. Data-Toy Example: VOTED/NON-VOTED Information

| | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $u_0$ | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| $u_1$ | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| $u_2$ | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |

Table V. Data-Toy Example: Classification Data and Categorical Labels for the Proposed Long Method

| $X'$ | Input neurons | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Relevant vs. non-relev. | | | | | | Voted vs. non-voted | | | | | |
| $X_{0,s}$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ | $n_0$ | $n_1$ | $n_2$ | $n_3$ | $n_4$ | $n_5$ |
| $X_{0,1}$ | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| $X_{0,3}$ | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $X_{1,2}$ | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| $X_{1,3}$ | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| $X_{1,5}$ | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| $X_{2,2}$ | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| $X_{2,5}$ | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |

## III. Experiments and Results

The designed experiments to test our proposed approaches make use of the MovieLens 1M[1] [45] dataset and the FilmTrust [44] dataset. The tested classification quality measures will mainly be the Precision and Recall ones [1]. The chosen relevancy threshold has been 4 for MovieLens and 3 for FilmTrust. The tested number of recommendations N has ranged from 1 to 96, in incrementes 5. For both datasets, the training set has been randomly obtained taking the 80% of the samples, whereas the test set has used the remaining 20%. Several neural network architectures have been tested and we have chosen the ones that have achieved better results. The Keras[2] deep learning library has been used to run experiments. For experiments reproducibility purposes we openly provide the necessary files to feed the neural networks[3]; these files have been obtained by implementing the proposed methods.

Experiment explanations have been organized by using two subsections. First subsection is devoted to compare the proposed long approach versus the proposed short approach. These experiments have been run using the MovieLens dataset, and they are explained by means of illustrative architectural figures. Results are provided both for classification accuracy and for recommendation accuracy. Second subsection compares the proposed short method with a representative state of art deep learning baseline: NCF. Recommendation and classification results are discussed both for MovieLens and FilmTrust datasets.

### A. Long versus Short Approaches Comparative

Our first set of experiments compares the results of the proposed short method versus the proposed long method, making use of the MovieLens dataset. The number I of items is 1682 in this dataset. We have trained our proposed long method by using a dense deep neural network involving a 2 x 1682 = 3364 size for the input layer, 1682 size in the output layer, 400 neurons in the first internal layer, followed by a 0.2 dropout layer, and finally 200 neurons in the second internal layer, followed by another 0.2 dropout layer. All the layers use 'relu' activation, except the output one that uses 'softmax' in order to provide probabilistic categorical results. The chosen loss function has been 'categorical cross entropy' since we are using categorical labels. The selected optimizer: 'adam'. Fig. 7 shows the explained neural network architecture. The resulting number of trainable parameters is 1,308,632.

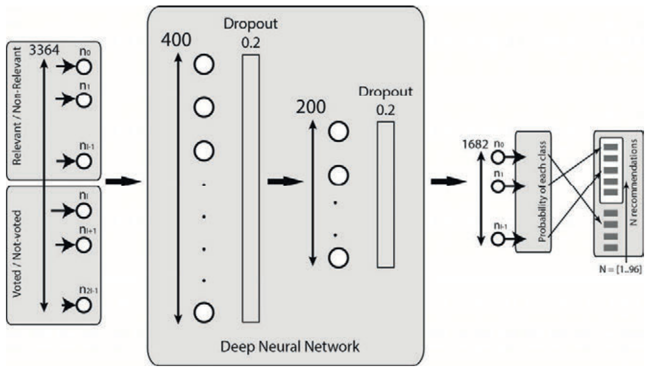

Fig. 7. Neural network architecture used to train the proposed long method on MovieLens 1M dataset.

Fig. 8 shows the results of the Fig. 7 training and testing processes. As it can be seen there is not overfitting, the loss function decreases to very low values and the accuracy raises to reach a 0.84 value. The

[1] https://movielens.org/

[2] https://keras.io/

[3] http://rs.etsisi.upm.es/

neural network learns in just 80 epochs. These results show good expectations to get accurate precision and recall values.

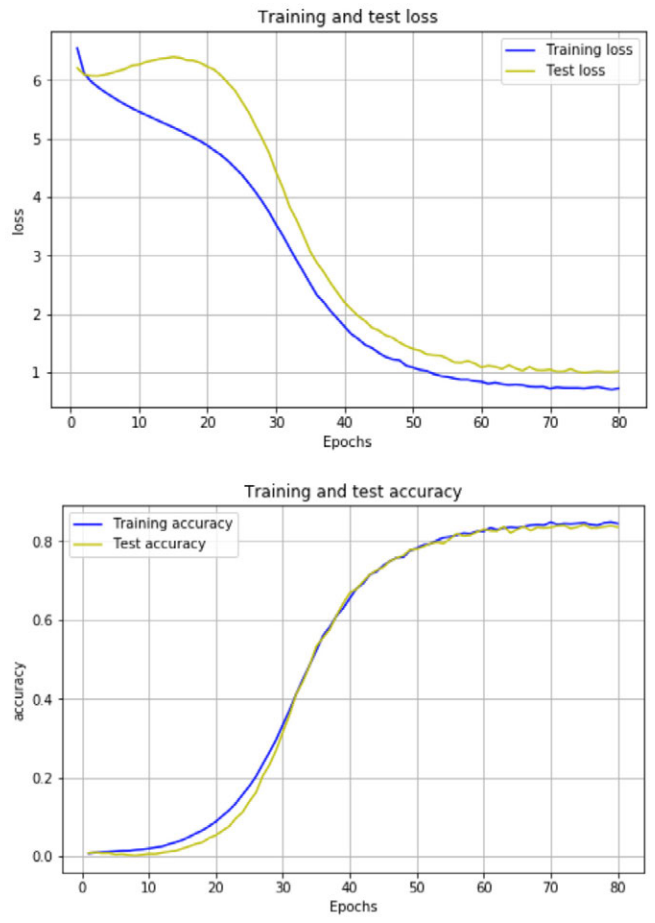

Fig. 8. The proposed long method results obtained by training the Fig. 7 deep neural network and using the MovieLens 1M dataset. Loss function (top graph) and accuracy (bottom graph) results both for training and test sets.

Analogously to the performed experiment using the proposed long model, an additional experiment has been made using the proposed short model. In this case, the size of the neural network input layer is the number of items of the dataset: 1682. Since the input layer size is half of the previous experiment one, we have reduced the neural network internal layer sizes to 200 and 100 and we have maintained the rest of parameters: two 0.2 dropouts, 'relu' and 'softmax' activations, etc. The resulting number of trainable parameters is 526,582. Fig. 9 shows the designed neural network architecture, and in Fig. 10 we can see that the obtained loss and accuracy values and evolutions are similar to the proposed long method ones. This case also feeds the expectation of accurate precision and recall results from recommendations.

It is important to distinguish between the loss and accuracy results, on one side, and the precision and recall, on the other side. Loss and accuracy are related to the training and testing results of the deep neural network (biggest square in Figs. 7 and 9). Precision and recall are related to the recommended items (most-right square in Figs. 7 and 9). Low loss values usually provide high accuracy ones. High accuracy values, in our context, means that we are correctly classifying items. Even if we classify correctly an item it does not directly mean that we have got a correct recommendation: probably this item has been voted by the user and it cannot be recommended, or it is possible that we have reached our limit of N recommendations and we cannot recommend the item.

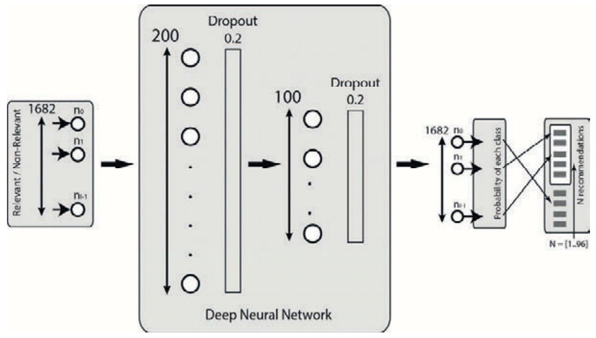

Fig. 9. Neural network architecture used to train the proposed short method. MovieLens 1M dataset.

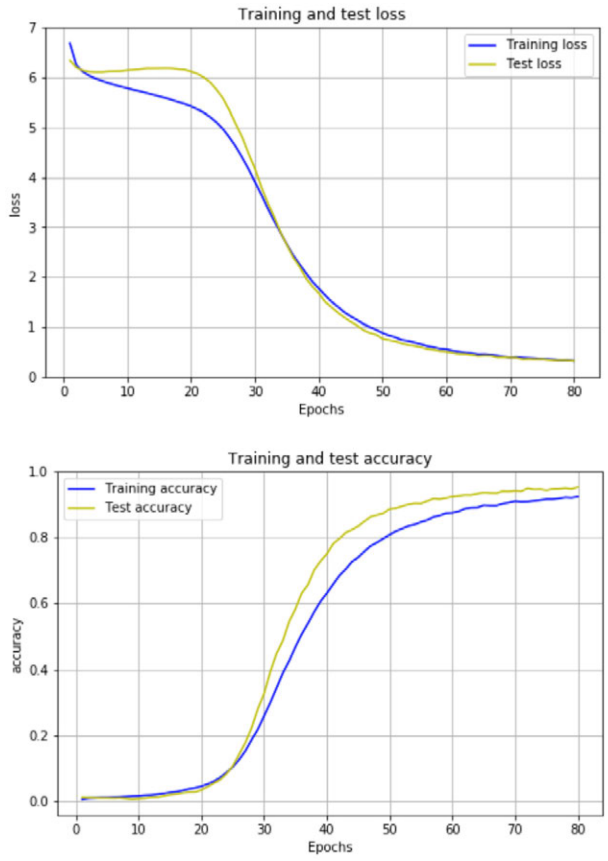

Fig. 10. The proposed short method results obtained by training the Fig. 9 deep neural network and using the MovieLens 1M dataset. Loss function (top graph) and accuracy (bottom graph) results both for training and test sets.

From each sample to predict (set of votes from some user) we will get up to N recommended items. These are the ones with the highest output probabilities (right boxes in Figs. 7 and 9; equation 17 in the formalization section). From these recommended items we get the combined precision, recall: F1, quality values using the classical information retrieval approach. Comparative results of both proposed methods are shown in Fig. 11; we can see a better performance of the short method when N is low, whereas for high number of recommendations the long method quality is something better. It makes sense, since the higher the N the more information is needed. The key issue here is that the quality difference between both approaches is not proportionally significant, whereas the deep neural network size needed to implement the long method is substantially bigger than the one needed to implement the short method (Figs. 7 and 9). Accordingly, we select the proposed short method as the preferred one, and we will use it in the rest of this paper's experiments.

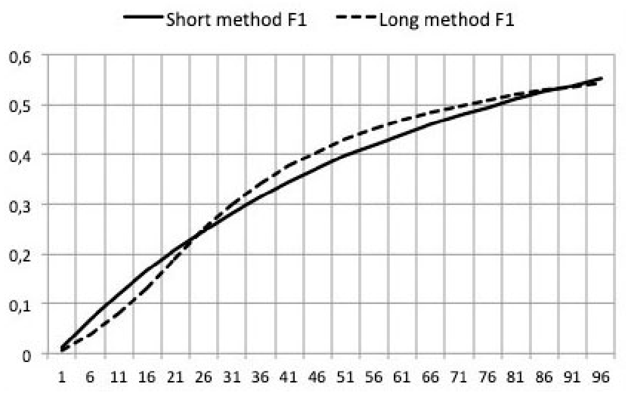

Fig. 11. Comparative recommendation F1 results obtained from RS classification using both the proposed short and long methods. Dataset: MovieLens 1M. y-axis: Combined precision & recall (F1); x-axis: Number of recommendations N.

## B. The Proposed Approach Performance

In this section, experiments are designed to compare our chosen approach (the proposed short method) with a representative state of art deep learning baseline: NCF. We have called our proposal as: NCCF (Neural Classification-based Collaborative Filtering). Experiments are run on the MovieLens and FilmTrust datasets. Precision and Recall quality measures have been tested. The chosen relevancy thresholds are 4 for MovieLens and 3 for FilmTrust. The tested number of recommendations N has ranged from 1 to 96, step 5. For both datasets, the training and test samples have been randomly split in 80% and 20% set sizes.

Experiments make use of the state of art baseline NCF [33]. NCF (Neural-based Collaborative Filtering) is a neural architecture that can express and generalize Matrix Factorization. Its design is shown in Fig. 12. NCF has been tested using several current baselines: BPR [16] and eALS [17] that are optimizations of the Matrix Factorization model, and ItemKNN [40]: the standard item-based KNN. The chosen NCF baseline can be classified as Neural Collaborative Filtering [18] and it provides similar accuracy to deepFM [34]; NCF integrates matrix factorization and Multilayer Perceptrons (MLP). Their main advantages are: a) Its performance, b) It avoid tedious feature engineering, and c) It does not need additional information to the CF dataset ratings. NCF is considered as a robust and accurate deep learning design for CF RS. The main current alternatives to the NCF are: a) The autoencoder based ones, such as CDL [41], and b) The deep hybrid models for recommendation [42] [43] [11]. Autoencoder based approaches avoid the non-neural network stage (MF) to obtain features from data. They increase the neural network complexity without reaching a remarkable accuracy rise. Deep hybrid models are based on additional data to the dataset ratings: demographic, social, context-aware, content-based information, etc. They do not provide a universal method to CF RS.

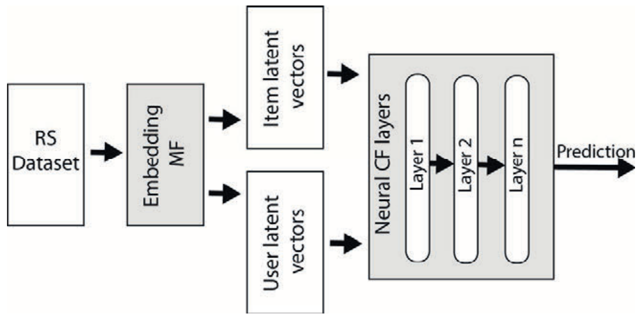

Fig. 12. NCF architecture used as baseline.

Once the baseline choice has been motivated, we are going to test our NCCF short method in both MovieLens and FilmTrust datasets. Fig. 13 shows the MovieLens comparative results for classification quality measures in both the baseline (NCF) and the proposed deep neural model (NCCF). Overall, Fig. 13 shows better precision results and worst recall results than the baseline. It is remarkable the proposed NCCF capability to maintain the precision accuracy level when a high number of recommendations is chosen. This trend is consolidated on the recall results. FilmTrust (Fig. 14) results point in the same direction: NCCF improves the baseline recall when the number of recommendations exceeds a threshold, although in this case precision is similar in both the baseline and the proposed method. The proposed method works particularly fine when the number of recommendations is not low. This is due to its original prediction schema: instead of making a prediction for each RS item, the proposed NCCF model generates a set of probabilities in the categorical output layer of the deep architecture. The result is a more equilibrated distribution of predictions along the RS set of items.

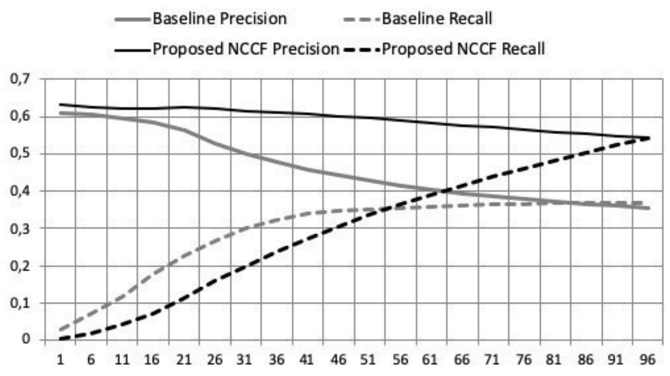

Fig. 13. MovieLens. Recommendation quality results; proposed NCCF versus the NCF baseline. NCF [33] outperforms to the state of art baselines: BPR [16], eALS [17], ItemKNN [40] and it provides similar accuracy to deepFM [34]. y-axis: Precision & recall; x-axis: Number of recommendations N.

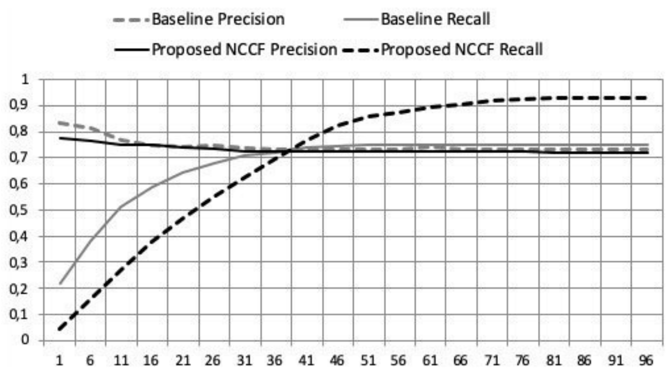

Fig. 14. FilmTrust. Recommendation quality results; proposed NCCF versus the NCF baseline. NCF [33] outperforms to the state of art baselines: BPR [16], eALS [17], ItemKNN [40] and it provides similar accuracy to deepFM [34]. y-axis: Precision & recall; x-axis: Number of recommendations N.

We can conclude by discussing that the proposed NCCF method can reach the current state of art neural collaborative filtering approaches, and it also improves recommendation results, particularly when the number of recommendations is not low. The NCCF method strengths are: 1) This is a scalable fully deep neural network approach, 2) The obtained recommendation accuracy is comparable to state of art approaches, and it improves results when the number of recommendations is not low, and 3) The size and complexity of the architecture is smaller than the autoencoder based ones.

## IV. Conclusions

A scalable and original classification-based deep neural architecture has been proposed in this paper. It is based on the concept that by reducing the ratings dataset information, accuracy will not significantly change. We renounce to the numerical rating values and transform them to the binary: relevant/non-relevant vote and voted/non-voted item. This information reduction leads to a new abstraction level and it opens a new range of classification-based architectures. Two related architectures are proposed in the paper: the short one and the long one. The large architecture makes use of both relevant/non-relevant and voted/non-voted binary information, whereas the short architecture only uses the relevant/non-relevant subset of information. Experiments show that the short architecture achieves a similar performance to the large one. Even although we lose a portion of accuracy, the short architecture is chosen due to its better scalability.

The proposed NCCF model has been tested on the MovieLens and the FilmTrust datasets and their results have been compared with the state of art baseline NCF. NCF is a powerful baseline, since it outperforms KNN, MF, BPR and eALS and it provides similar accuracy to deepFM. Results show that our scalable approach provides a competitive recommendation quality. It particularly outperforms the baseline when the number of recommendations is not low. The result is consistent with the way predictions are made: to recommend a user, instead of making a prediction for each RS item, the proposed NCCF model generates a set of probabilities in the categorical output layer of the deep architecture. This is an original approach, and it obtains a more equilibrated distribution of predictions along the RS set of items.

The proposed method novelty opens a wide range of future works, among which are: a) To design an NCCF ensemble, inspired on the random forest method, where the input data that feeds each NCCF is a random subset of the RS items, b) To make use of data augmentation from the binary relevant/non-relevant information in order to enrich the set of input samples, c) To experiment the quality impact of stablishing a multilabel target on the samples, instead the proposed one label approach, and d) To incorporate a wide (memorization) layer to create a wide & deep neural network architecture.

## References

[1] J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez, “Recommender systems survey,” Knowledge-Based Syst., vol. 46, pp. 109–132, 2013.

[2] Xin Dong, Lei Yu, ZhonghuoWu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang, “A hybrid collaborative filtering model with deep structure for recommender systems”. In Proceedings of the AAAI.1309–1315,2017. Available: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14676

[3] M.J. Pazzani, “A framework for collaborative, content-based and demographic filtering”, Artif. Intell. Rev. 13 (5-6) pp. 393–408, 1999.

[4] T. Ebesu and Y. Fang. “Neural citation network for context-aware citation recommendation”. In Proceedings of the SIGIR. ACM, Shinjuku, Tokyo, pp. 1093–1096. 2017.

[5] C. Musto, C. Greco, A. Suglia, and G. Semeraro, “Askme any rating: A content-based recommender system based on recurrent neural networks”, Proceedings of the IIR, 2016.

[6] X.Wang, X. He, L. Nie, and T.-S. Chua, “Item silk road: Recommending items from information domains to social users”, Proceedings of the SIGIR. ACM, pp. 185–194, 2017.

[7] C. Yang, L. Bai, C. Zhang, Q. Yuan, and J. Han, “Bridging collaborative filtering and semisupervised learning: A neural approach for POI recommendation”, Proceedings of the SIGKDD, pp. 1245–1254. Available: http://dl.acm.org/citation.cfm?id=3172077.3172336.

[8] B. Yi, X. Shen, Z. Zhang, J. Shu, and H. Liu, “Expanded autoencoder recommendation framework and its application in movie recommendation”. Proceedings of the SKIMA, pp. 298–303, 2016.

[9] W. Chu and Y. Tsai. “A hybrid recommendation system considering visual information for predicting favorite restaurants”. WWWJ (2017), pp. 1–19, 2017.

[10] R. He and J. McAuley. “Ups and downs: Modeling the visual evolution of fashion trends with oneclass collaborative filtering”, Proceedings of the WWW, pp. 507–517. 2016. Available: https://doi.org/10.1145/2736277.2741667.

[11] C. Hsieh, L. Yang, H. Wei, M. Naaman, and D. Estrin, “Immersive recommendation: News and event recommendations using personal digital traces”, Proceedings of the WWW, pp. 51–62. 2016. Available: https://doi.org/10.1145/2872427.2883006

[12] X. Deng, F. Huangfu, “Collaborative Variational Deep Learning for Healthcare Recommendation”, IEEE Access, Vol. 7, 2019.

[13] Z. Chen, L. Li, H. Peng, Y. Liu, H. Zhu, Y. Yang, “Sparse General Non-Negative Matrix Factorization Based on Left Semi-Tensor Product”, IEEE Access, Vol. 7, pp. 81599 – 81611, 2019.

[14] K. Li, X. Zhou, F. Lin, W. Zeng, G. Alterovitz, “Deep Probabilistic Matrix Factorization Framework for Online Collaborative Filtering”, IEEE Access, Volume: 7, 2019.

[15] A. Hernando, F. Bobadilla, F. Ortega, “A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model”, Knowledge-Based Syst. 197C (2016) pp. 188–202.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. “Bpr: Bayesian personalized ranking from implicit feedback”. In UAI, pp. 452-461, 2009.

[17] X. He, H. Zhang, M.-Y. Kan, and T.-S. Chua. “Fast matrix factorization for online recommendation with implicit feedback”. SIGIR, pp. 549-558, 2016.

[18] S. Zhang, L. Yao, A. Sun, and Y. Tay. “Deep Learning based Recommender System: A Survey and New Perspectives”. ACM Comput. Surv.1, 1, Article 1. July 2018, pp. 35.

[19] R. Mu, “A Survey of Recommender Systems Based on Deep Learning”, IEEE Access, Year: 2018 | Volume: 6, pp. 69009–69022, DOI: 10.1109/ACCESS.2018.2880197

[20] X. Wang, L. Yu, K. Ren, G. Tao, W. Zhang, Y. Yu, and J. Wang, “Dynamic attention deep model for article recommendation by learning human editors’ demonstration”, Proceedings of the SIGKDD, 2017, Available:

[21] http://doi.acm.org/10.1145/3097983.3098096

[22] D. Liang, M. Zhan, and D. P.W. Ellis, “Content-aware collaborative music recommendation using pre-trained neural networks”, Proceedings of the ISMIR, 2015, pp. 295–301.

[23] X. Wang and Y. Wang, “Improving content-based and hybrid music recommendation using deep learning”. Proceedings of the MM, 2014, pp. 627–636. Available: http://doi.acm.org/10.1145/2647868.2654940

[24] J. Chen, H. Zhang, X. He, L. Nie, W. Liu, and T.-S. Chua. “Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention”. Proceedings of the SIGIR. 2017. ACM. Available: http://doi.acm.org/10.1145/3077136.3080797

[25] X. Chen, Y. Zhang, Q. Ai, H. Xu, J. Yan, and Z. Qin. “Personalized key frame recommendation”, Proceedings of the SIGIR, 2017, pp. 315–324. Available: http://doi.acm.org/10.1145/3077136.3080776

[26] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, “Parallel recurrent neural network architectures for feature-rich session-based recommendations”, Proceedings of the Recsys, 2016, pp. 241–248. Available: http://doi.acm.org/10.1145/2959100.2959167

[27] D. Jannach and M. Ludewig, “When recurrent neural networks meet the neighborhood for sesión based recommendation”, Proceedings of the Recsys (RecSys’17), 2017, ACM, Como, pp. 306–310.

[28] I. Tang and K. Wang, “Personalized top-n sequential recommendation via convolutional sequence embedding”, Proceedings of the WSDM, 2018, pp. 565–573. Available: http://doi.acm.org/10.1145/3159652.3159656

[29] T. Xuan Tuan and T. Minh Phuong, “3D convolutional networks for session-based recommendation with content features”, Proceedings of the Recsys, 2017, pp. 138–146.

[30] A. Suglia, C. Greco, C. Musto, M. de Gemmis, P. Lops, and G. Semeraro, “A deep architecture for content-based recommendations exploiting recurrent neural networks”, Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization, 2017, ACM, Bratislava, pp. 202–211.

[31] Y. Kiam Tan, X. Xu, and Y. Liu. “Improved recurrent neural networks for session-based recommendations”, Proceedings of the Recsys, 2016, pp.

17–22. Available: http://doi.acm.org/10.1145/2988450.2988452

[32] T. Alashkar, S. Jiang, S. Wang, and Y. Fu, "Examples-rules guided deep neural network for makeup recommendation", Proceedings of the AAAI, pp. 941–947. Available: https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14773

[33] S. Zhang, L. Yao, and X. Xu, "AutoSVD++: An efficient hybrid collaborative filtering model via contractive auto-encoders", ACM, 2017, pp. 957–960.

[34] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-Seng Chua, "Neural collaborative filtering", Proceedings of the WWW. 2017, pp.173–182. Available: https://doi.org/10.1145/3038912.3052569

[35] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He, "DeepFM: A factorization-machine based neural network for CTR prediction", Proceedings of the IJCAI, 2017, pp. 2782–2788. Available: http://dl.acm.org/citation.cfm?id=3172077.3172127

[36] G. K. Dziugaite and D. M. Roy. 2015. "Neural network matrix factorization". arXiv preprint arXiv:1511.06443 (2015).

[37] H.-T. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir et al., "Wide & deep learning for recommender systems". Proceedings of the Recsys, 2016, pp. 7–10. Available: http://doi.acm.org/10.1145/2988450.2988454

[38] W. Yuan, H. Wang, B. Hu, L. Wang, Q. Wang, "Wide and Deep Model of Multi-Source Information-Aware Recommender System", IEEE Access, Year: 2018 | Volume: 6, pp. 49385-49398.

[39] S. Rendle, "Factorization machines", 2010 IEEE International Conference on Data Mining, December pp. 13-17, 2010, Sydney, Australia.

[40] J. Bobadilla, F. Serradilla, J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems", Knowledge-Based Systems 23 (2010), pp. 520–528.

[41] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, "Item-based collaborative _ltering recommendation algorithms", WWW10, May 1-5, 2001, Hong Kong. ACM 1-58113-348-0/01/0005, pp. 285-295.

[42] H. Wang, N. Wang, D.-Y. Yeung, "Collaborative deep learning for recommender systems", Proceedings of the SIGKDD, 2015, pp.1235-1244.

[43] F. Zhang, N. Jing Yuan, D. Lian, X. Xie, W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems". Proceedings of the SIGKDD, 2015, pp. 353–362. Available: http://doi.acm.org/10.1145/2939672.2939673

[44] H. Lee, Y. Ahn, H. Lee, S. Ha, S. Lee, "Quote recommendation in dialogue using deep neural network", Proceedings of the SIGIR, 2016, pp. 957–960. Available: http://dx.doi.org/10.1145/2911451.2914734

[45] J. Golbeck, J. Hendler, "FilmTrust: movie recommendations using trust in web-based social networks", CCNC 2006, 3rd IEEE Consumer Communications and Networking Conference, 2006, DOI: 10.1109/CCNC.2006.1593032

[46] F. M. Harper and J. A. Konstan. 2015. "The MovieLens Datasets: History and Context". ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), pp. 19. Available: http://dx.doi.org/10.1145/2827872

### Jesús Bobadilla

Jesús Bobadilla received the B.S. and the Ph.D. degrees in computer science from the Universidad Politécnica de Madrid and the Universidad Carlos III. Currently, he is a lecturer with the Department of Applied Intelligent Systems, Universidad Politécnica de Madrid. He is a habitual author of programming languages books working with McGraw-Hill, Ra-Ma and Alfa Omega publishers. His research interests include information retrieval, recommender systems and speech processing. He is in charge of the FilmAffinity.com research team working on the collaborative filtering kernel of the web site. He has been a researcher into the International Computer Science Institute at Berkeley University and into the Sheffield University. Head of the research group.

### Fernando Ortega

Fernando Ortega was born in Madrid, Spain in 1988. He received the B.S. degree in software engineering, the M.S. degree in artificial intelligence, and the Ph.D. degree in computer science from Universidad Politécnica de Madrid, in 2010, 2011 and 2015, respectively. From 2008 to 2015, he was a Research Assistant with the Intelligent Systems for Social Learning and Virtual Environments research group. From 2015 to 2017, he was working at BigTrueData, leading machine learning projects. From 2017 to 2018, he was an Assistant Professor at U-tad: Centro Universitario de Tecnología y Arte Digital. Since 2018, he is Assistant Professor at Universidad Politécnica de Madrid. He is the author of 30 research papers in most prestigious international journals. His research interests include machine learning, data analysis and artificial intelligence. He leads several national projects to include machine learning algorithms into the society.

### Abraham Gutiérrez

Abraham Gutiérrez was born in Madrid, Spain, in 1969. He received the B.S. and the Ph.D. degrees in computer science from the Universidad Politécnica de Madrid. Currently, he is a lecturer with the Department of Applied Intelligent Systems, Universidad Politécnica de Madrid. He is a habitual author of programming languages books working with McGraw-Hill, Ra-Ma and Alfa Omega publishers. His research interests include P-Systems, social networks and recommender systems. He is in charge of this group innovation issues, including the commercial projects.

### Santiago Alonso

Santiago Alonso received his B.S. degree in software engineering from Universidad Autónoma de Madrid and his Ph.D. degree in computer science and artificial intelligence from Universidad Politécnica de Madrid, in 2015, where he is currently an Associate Professor, participating in master and degree subjects and doing work related with advanced databases. His main research interests include natural computing (P-Systems), and did some work on genetic algorithms. His current interests include machine learning, data analysis and artificial intelligence.