IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 26, 2018, accepted August 28, 2018, date of publication September 3, 2018, date of current version September 28, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2868464

An Efficient Recommender System Method Based on the Numerical Relevances and the Non-Numerical Structures of the Ratings

BO ZHU[1,2], REMIGIO HURTADO[2,3], JESÚS BOBADILLA[2], AND FERNANDO ORTEGA[4]

[1]Beijing Institute of Technology, Beijing 100081, China
[2]Department of Computer Science, Universidad Politécnica de Madrid, 28031 Madrid, Spain
[3]Department of Computer Science, Universidad Politécnica Salesiana, Cuenca 010102, Ecuador
[4]U-tad: Centro Universitario de Tecnología y Arte Digital, Calle Playa de Liencres, 28290 Las Rozas de Madrid, Spain

Corresponding author: Remigio Hurtado (rhurtadoo@ups.edu.ec)

ABSTRACT In this paper, we propose a collaborative filtering method designed to improve the current memory-based prediction times without worsening and even improving the existing accuracy results. The accuracy improvement is achieved by combining the numerical relevance of the ratings with non-numerical information based on the votes structure. The improvement of the prediction time is achieved by setting four actions: 1) simplification of the similarity measure design, in order to minimize the necessary calculations; 2) construction and maintenance of a model that simplifies the predictions processing; 3) optimization of the computation, using a set-based model and a bit-based processing implementation; and 4) switching between the bit processing and the numerical processing, depending on the density of the users' ratings. Experimental results show the improvements both in the prediction time and the accuracy. Experiments have used a significant amount of state-of-the-art baselines and collaborative filtering public data sets.

INDEX TERMS Recommender systems, collaborative filtering, performance, prediction time, similarity measures, model-based methods.

I. INTRODUCTION

In this paper, we propose a Collaborative Filtering (CF) method for Recommender Systems (RS). This method is based on a novel similarity measure and on the *BitSet* optimization process. The novelty of the proposed solution lies in the design approach: we are facing a hybrid method in between memory-based approaches and model-based approaches. In this way, we obtain positive results in several individual aspects that usually do not fit together, such as: accuracy and prediction time, using memory-based approaches; recommendation explanations and updated results using model-based approaches. The relevance of the proposed method is based on the balance and the quality of its results: it is a method that achieves good values in accuracy, prediction time, and time to set and update its model. In addition, by making use of memory-based algorithms, our method facilitates the explanation of recommendations and makes it possible to get reliability values. The relevance of the proposed method is not based on maximizing, individually, the achievement of any of the previous objectives; its relevance is based on obtaining a good and balanced behavior in all of them. As an example: their accuracy results are close to the matrix factorization model-based methods, but they are not better. However, the proposed method, on average, obtains better accuracy results than current memory-based solutions, and it predicts faster than most of them.

The proposed method makes use of various concepts that have already been shown in different CF solutions. In this aspect, the novelty of this method is in identifying the most promising concepts and in selecting those that reinforce each other, without being redundant or canceling them. The relevance of our solution comes from: a) Unifying the selected concepts into a similarity measure, b) Having achieved a very efficient bit-based design, and c) Having tested their results using a wide set of baselines, quality measures and public datasets. The main concepts that have been identified are: a) Numerical relevance of the ratings, and b) Non-Numerical structures of the ratings. These concepts are explained, in detail, in section III-A. The novelty of our approach lies in several aspects: 1) The reduction of numerical information that we make with respect to the published solutions, extracting the most relevant information to

TABLE 1. Main advantages and disadvantages of the memory-based and model-based approaches.

Memory-based approaches		Model-based approaches	
Advantages	**Disadvantages**	**Advantages**	**Disadvantages**
It does not require an initial time to create a model	Worse accuracy than model-based methods	Better accuracy than memory-based methods	Time needed to create the model
Simplicity of algorithms	Worst recommendation times	Better recommendation times	Time needed to periodically update the model
Recommendations are always updated		Additional possibilities from the model (clustering, analytics, etc.)	Recommendations are based on an outdated model
It does not require resources to store the model			It requires resources to store the model
Ease to provide explanation of recommendations			Complex algorithms.

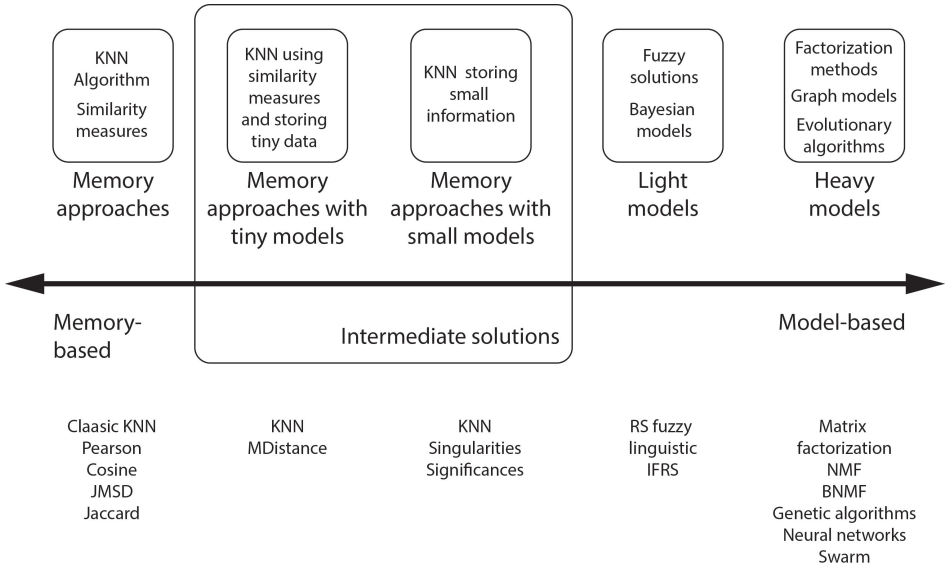

FIGURE 1. Memory-based to model-based progressive approaches schema.

obtain accuracy, and drastically reducing prediction times, b) The fusion process we perform between numerical information and structural information, and c) The optimization phase, which minimizes the computations according to the density of each dataset.

It is necessary to highlight the importance of performance improvement. We might think that, if the user's requirement is already satisfied, the improvement of the recommendation times is of little importance. The above approach is not usually valid for RS in operation, due to several reasons: 1) Accesses of users to the system have a very uneven distribution; in particular there are saturations during specific time periods. As an example, in *filmaffinity.com* we have one of these demand peaks every Saturday afternoon. These situations can lead to system crashes (''Dying of success'') or to contract an oversized and expensive hosting. The lower the processing times of the CF methods, the lower the system costs, and 2) Lower recommendation times help to achieve user satisfaction, which will always be greater the more immediate they receive their responses, even in peak periods.

In the rest of this section we are going to classify and explain our proposed method. First, we establish the differences, advantages and disadvantages between the memory-based and the model-based approaches; later we explain the possible scenarios that can be found in between each of the previous approaches. Finally, we place the proposed method in the indicated schema.

A general classification of Collaborative Filtering Recommender Systems divides the solutions into: *Memory-based* or *model-based*. As explained in the previous section, *Memory-based* approaches perform predictions and recommendations based on the ratings matrix; that is: from the original data. *Model-based* approaches, first build a model from the ratings matrix; Subsequently, predictions and recommendations are made from the model. Each of these approaches has their advantages and disadvantages, which must be assessed to choose the design of the RS. Table 1 develops this concept.

Traditionally, RS classifications divide CF approaches in model-based versus memory-based. It does not mean that the separation between both solutions is absolute: As usual in the scientific and technological approaches, there is a variety of cases that are found in both border sides of the dividing line, presenting characteristics of both models. Fig. 1 schematizes this idea: the horizontal arrow shows the location of the solutions; more to the right means a larger

model-based approach, while more to the left refers to a larger memory-based approach.

In Fig. 1 (right) we can see how there are solutions based on ''heavy'' models as well as solutions based on ''light'' models. The former requires more resources to generate and store the model, while the latter are processed faster and stored in a smaller space. The biggest advantages of the ''heavy'' models are: better accuracy and shorter recommendation time. Most used ''heavy'' models are matrix factorization and its variants (NMF, PMF, BNMF, etc.). Evolutionary algorithms, such as: genetics, neural networks, ants, swarms, etc. also require high training times. Finally, there are CF solutions based on some graph algorithms that could be classified as ''hard'' model-based.

Fig. 1 (left) shows the classic memory-based solution: KNN algorithm implemented with some of the best-known similarity measures. In this case there is no need to create a model: Recommendations are obtained directly from the ratings matrix. The outermost box in Fig. 1 contains the solutions that, being mostly memory-based, present some characteristics of the model-based approach. Our proposed method is located in this space of ''intermediate solutions'', taking ''tiny modelsv features and also ''small models'' characteristics. In section III, the proposed method design is explained in detail and its hybrid nature is justified: model-based & memory-based. The lower part of Fig. 1 highlights some of the main methods representing each of the exposed models and approaches. Next subsection deepens in the current related work.

The hypothesis of this paper claims that it is possible to obtain a similarity measure that meets the following conditions: 1) It will require the creation of a simple model, involving efficient storage space and fast building times, 2) Predictions times will be shorter than current CF similarity measures ones, and 3) The quality of the accuracy obtained will not be much worse than those reported by the current CF similarity measures and methods; in particular its accuracy will be just a little worse than the matrix factorization current approaches. In return, it presents the advantages of the memory-based algorithms, shown in Table 1.

In short, the proposed similarity measure is located in the position ''Intermediate solutions'' shown in Fig. 1. It is based on the construction of an efficient model, making use of fewer resources than the traditional model-based approaches. Its main advantage over existing KNN similarity measures is the speed with which predictions are obtained. Its main advantage over the existing model-based approaches is its model simplicity: it requires few resources to create it.

At this point, it is necessary to show the way in which CF solutions can be located in between the two traditionally separated approaches: memory-based and model-based CF approaches. Fig. 2 contains a graphic description of some progressive memory to model based solutions. In the bottom of Fig. 2, the traditional MF model-based approach is shown. Above it, a lighter solution changes hidden factors for visible features. An example of this approach is to locate numerical demographic values for each user and for each item of the dataset. Fig. 2 next simple approach ''Small model'' arranges some measure to each user and each item (e.g. mean of ratings). Finally, the tiny model showed on the top of Fig. 2 can store some global rating matrix values.

From Fig. 2, heavy models convert the source rating matrix to reduced factor matrices, whereas light models reduce the source rating matrix to reduced features matrices. In this way, we must define the differences between factors and features: Factors are obtained using some machine learning factorization method; they have a hidden nature, since we do not know the features each factor is coding. Features are obtained using some statistical method; they usually combine user and item demographic information. Heavy models get better reduction results and accuracy, whereas light models get better setting times and make it easy to explain recommendations.

It is expected than the smaller the model is, the more efficient will be to create and update it. It is also expected that more complex models will obtain better accuracy results. Our proposed similarity measure is located in the Fig. 2 section ''Memory approach small model'', where we store a compressed representation of the ratings casted for each user.

Using KNN it is possible to pre-compute all the rating matrix, looking for all the existing similarities, and to store the set of similarities results (or the neighborhood of each user). This is a large storage space, and it requires a heavy time-consuming updating process. Model-based methods can use the folding-in updating strategy. Furthermore: model-based methods can retrain the model each relatively large intervals of time, since the inclusion of new ratings effect is ''absorbed'' for an efficient model that compresses the whole rating matrix into hidden factors. KNN methods, conversely, are very sensitive to the new input ratings, since similarities search acts on users (or items) vectors pairs, that holds a tiny fraction of the whole dataset: a single rating cast by a user can change her neighborhood. The proposed method offers an intermediate solution that, in this particular issue, is not as good as the model-based ones, but it allows to use a memory-based approach and to maintain a small rating matrix model, fast to set and to update, and that allows to speed up memory-based prediction times.

II. RELATED WORK

Recommender Systems [1], [2] allow to mitigate part of the Internet information overload problem. From the point of view of an RS user, based on his past preferences, the System automatically recommends a series of items (movies, music, electronics, clothing, etc.) that are available and that the user has not consumed yet. The RS can make recommendations based on various types of information sources; the most common ones are: content-based, demographic, collaborative, social, and context-aware.

RS based on *content information* [3] carry out recommendations in the following way: if the user to whom you wish to recommend (active user) has liked a product or service, the RS recommends similar products or services: e.g. if the

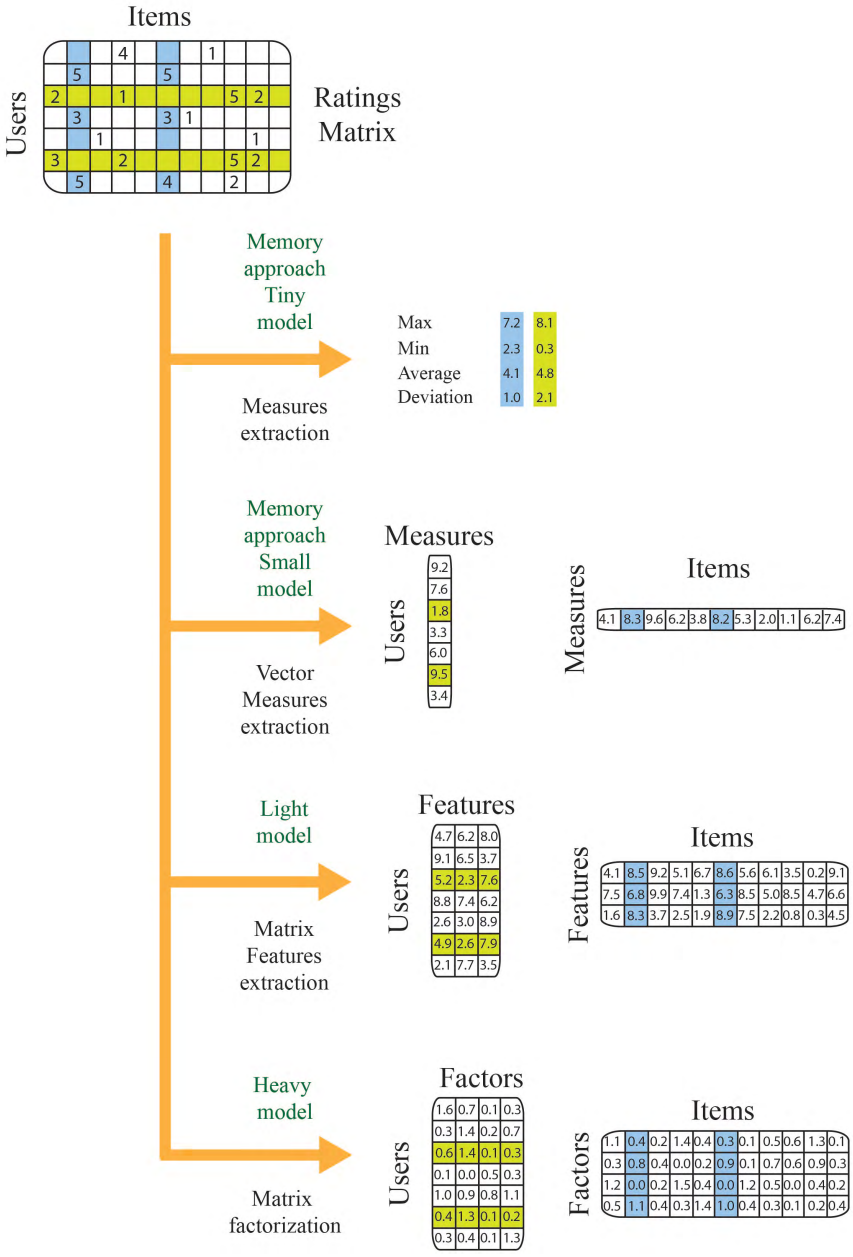

FIGURE 2. Memory-based to model-based progressive approaches solutions.

active user bought a historical novel, it is likely to be recommended a history book or a historical novel book that he has not bought or read. One of the biggest drawbacks of the content-based approach is the lack of diversity of its recommendations.

The RS based on *demographic information* [4] make recommendations based on the products consumed by demographically similar users to the active user (age, genre, location, etc.). The main drawbacks of the demographic-based RS are: 1) Complete demographic data is not usually available, and 2) There is too much variability in the preferences of each demographic group.

The *social-based* RS [5]–[7] make use of relationships between users: likes, dislikes, follows, etc. Social-based RS recommend to active users based on the preferences of their closest social network. A main problem of this approach is that most of the existing datasets do not contain enough social information. Finally, context-aware RS [8] are usually associated with the Internet of Things (IoT), where context information is collected: GPS coordinates, RFID information, credit card data, etc.

Collaborative Filtering RS [1], [9] usually offer the best recommendation results. Their operation is as follows: the active user is recommended items that have not been

consumed and that have been positively rated by users who have preferences similar to those of the active user. That is, information is extracted from all existing users (hundreds of thousands or millions) and based on that information, the active user is recommended. Normally, information is structured as a matrix that stores the preferences (explicit or implicit) of each of the users about the set of items. These matrices (efficiently saved in datasets) are enormously sparse, because a typical user has only been able to consume or rate a very small subset of the set of available items (thousands or tens of thousands).

There is a wide variety of approaches to extract the most relevant information from the sparse collaborative filtering matrices. The traditional approach was the KNN algorithm (*K Nearest Neighbors*) [10], where the most similar K users (neighborhood) are searched for each active user; subsequently, items not consumed yet by the active user that have been highly valued or consumed by its neighborhood are recommended. The previous approach is classified as *memory-based* [1], [9]: information to recommend is obtained directly from the data. The explained process is called *user-based*; it is also possible to carry out an *item-based* recommendation, obtaining neighborhood sets of each item.

Currently, CF RS are usually designed by using the *model-based* [1], [9] [11]–[16] approach: A model is created from the data, and subsequently recommendations are obtained from the model. The RS most used model-based method is the *Matrix Factorization* (MF) [7], [17]: The sparse ratings matrix is compressed into two dense factor matrices (one matrix containing the users information and another matrix containing the items information). One of the matrices has (*users* · *factors*) size, and the other matrix has (*items* · *factors*) size. The number of factors is usually small (10 to 50), and then the size of each of the two matrices is much smaller than the size of the original information (*users* · *items*). This compressed information contains the essence of the original information, coded in factors that are called hidden, because its meaning (the concept they encode) is not known. The prediction and recommendation process from this model usually improves the quality [18], [19] obtained through memory-based approaches.

Latent Factor Models (LFM) have been currently used to perform community identification and feature summarization: [20] proposes a new LFM (LFCIS) based on an objective function that evaluates the overall clustering quality taking into the consideration both edge topology and node features in the network. Non-Negative latent factor models have been published in order to fulfil this constraint on the factors [21]. The scalability issue is important in the context of big data: in [12] authors propose a new LFM method to obtain high convergence rate as well as low complexity. Finally, in [22], an LFM is provided to integrate ratings and reviews in the RS context, using Amazon datasets.

Shilling attacks and profile injection attacks are a circumstance that RS should be able to manage since malicious ratings can enter into the system. These types of attacks may lead to a degradation of user trust in the RS objectivity and accuracy. Mobasher *et al.* [23] outline some of the major issues in building secure recommender systems, concentrating in particular on the modeling of attacks and their impact on various recommendation algorithms. Gunes *et al.* [24] explain robust recommendation algorithms, introduce a novel attack classification, make a review of shilling attacks in collaborative filtering algorithms; they describe various attack types, and briefly explain some evaluation strategies. Reference [25] reviews the shilling attacks detection in privacy-preserving CF systems.

Finally, it is important to highlight two important concepts: 1) The more information a RS gathers, the better results it can provide; that's why hybrid RS [26] are usually designed in commercial RS (typically: collaborative + content + demographic), and 2) There are model-based RS different from the MF approach, although their use is not extended: fuzzy approaches [27], [28], evolutionary algorithms (ants, swarm, etc.) [29], Bayesian methods [30], clustering [31], etc.

A. MEMORY-BASED APPROACHES

The KNN algorithm is the traditional way to implement memory-based approaches. This algorithm is feed on the user to user (or item to item) similarity measures. Traditional similarity measures are [1]: Pearson correlation, Spearman Rank, sine and cosine, Jaccard, etc. Due to the high degree of sparsity of the ratings vectors, a series of similarity measures have emerged that take into account this circumstance. JMSD [32] combines the numerical information of the votes with independent information from those values, based on the proportions of the common and uncommon votes between each pair of users. PIP measure [33] focuses on improving recommendation performance under cold-start conditions where only a small number of ratings are available for similarity calculation for each user. An approach based on mean measure of divergence [34] takes rating habits of users into account. The similarity measure designed in [10] provides extremely high-quality and balanced results; these results are complemented with a low processing time, similar to the one required to execute traditional similarity metrics. A new user similarity measure [35] to improve the recommendation performance when only few ratings are available to calculate the similarities for each user. This metric not only considers the local context information of user ratings, but also the global preference of user behavior.

B. MEMORY-BASED APPROACHES USING TINY AND SMALL MODELS

Results obtained by applying traditional similarities measures can be improved by taking contextual information, drawn from the entire body of users, and using it to calculate the singularity [36] which exists, for each item, in the votes cast by each pair of users. A similar idea (significances) [37] is applied to improve the information used in CF processes by weighting the ratings of the items according to

their importance. In this way, the k-neighbors are calculated taking into account the ratings of the items, the significance of the items and the significance of each user for making recommendations to other users. RES is a similarity measure inspired by a physical resonance phenomenon [38]; authors fully consider different personalized situations in RES by mathematically modeling the consistency of users' rating behaviors, the distances between users' opinions, and the Jaccard factor with both correlated and non-related ratings. An efficient CF algorithm based on a new measure called the M-distance [39], defines similarity as the difference between the average ratings of two items; this metric stores items vectors in order to speed up prediction times.

C. LIGHT MODELS

A fuzzy linguistic recommender system [40] based on the Google Wave capabilities is proposed as tool for communicating researchers interested in common research lines. A medical diagnosis fuzzy RS (IFRS) is presented in [41] (a novel intuitionistic fuzzy recommender systems). Fuzzy approaches are common to provide hybrid RS: A fuzzy hybrid multi-agent recommender system [42] is designed and developed. They make use of interval type-2 fuzzy sets to create user models capable of capturing the inherent ambiguity of human behavior related to diverse users' tastes. Liu *et al.* [43] propose a novel inference algorithm, called the Online Bayesian Inference algorithm for CTR model, which is efficient and scalable for learning from data streams. Based on a hybrid recommendation framework that uses: Random Forests, Naïve Bayes and Logistic Regression, [44] investigates the impact of the features gathered from the Linked Open Data cloud.

D. HEAVY MODELS

A Bayesian MF Technique is presented in [30]. It is based on factorizing the rating matrix into two non-negative matrices whose components lie within the range [0, 1] with an understandable probabilistic meaning. To automatically interpret MF features as users, referred to as representative users [45] provides MF with an extra advantage. This interpretation relies on the study of the matrices that result from the factorization and on their link with the original rating matrix. The framework from [46] involves two efficient MF: dynamic single element-based CF-integrating manifold regularization and dynamic single-element-based Tikhonov graph regularization non-negative MF. A Bayesian Wishart matrix factorization method [47] models the temporal dynamics of variations among user preferences and item attractiveness in a novel algorithmic perspective. The proposed method is able to well model and properly control diverse rating behaviors across time frames and related temporal effects within time frames in the tendency of user preferences and item attractiveness. A knowledge graph [48] supplies a hybrid recommendation engine with information that builds on top of a collections of documents describing musical and sound items. Tags and textual descriptions are exploited to extract and link entities to external graphs. A privacy aware recommender system that exploits relations present between entities is proposed in [26]. They use content from user's history and entities appearing in candidate content. In order to identify such relations, we use the knowledge graph, which encodes entities and their relations. To provide predictions, [49] proposes a non-linear neural network-based approach: authors combine user and item factors to feed a neural network and to obtain recommendations. In [50], users' profiles are initially represented by tags and then a deep neural network model is used to extract the in-depth features from tag space layer by layer. Representations of the raw data will become more abstract and advanced, and therefore the unique structure of tag space will be revealed automatically. A collaborative filtering algorithm based on attributes of items [51], weights vectors for each user, considering them as a chromosome in genetic algorithm. This algorithm optimizes the weights according to historical rating. A Trust-aware recommender system [52] uses genetic algorithms, choosing the most suitable nodes for the skeleton of recommender searching. It can achieve the maximum prediction coverage with the minimum skeleton maintenance cost. Using ant colony optimization, [53] performs a depth first search for the optimal trust paths in the trust network and selects the best neighbors of an active user to provide better recommendations. An RS model based on the Support Vector Machine is proposed in [54]. The proposed model not only considers the items' content information, but also the users' demographic and behavior information to fully capture the users' interests and preferences. An improved Particle Swarm Optimization algorithm is used to improve its performance. An evolutionary approach, called Invenire, is proposed in [55] to automate the choice of techniques used by combining results of different recommendation approaches. It uses a search algorithm to optimize the techniques combination.

III. PROPOSED METHOD

This section details the proposed method. It is divided in the following subsections: III-A Design, motivation and concepts that define the method, III-B Formalization, equations, implementation, and III-C Optimization stage to improve prediction times, and III-D Algorithm description and complexity analysis.

A. METHOD DESIGN

The proposed method has been designed based on the knowledge of the existing CF similarity measures. The main design objective has been directed towards efficiency: to decrease the time needed to predict and, therefore, to recommend. The design process has been based on simplifying prediction calculations; for this purpose, data models are created. The existing similarity measures have been analyzed and we have chosen those supporting some model that accelerates their processing. Likewise, a huge amount of experiments has been done combining similarity measures in search of efficient combinations providing accurateaccurated results. Finally, several simplifications have been tested in the

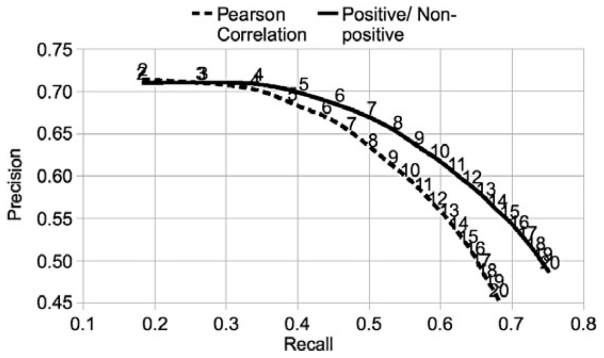

FIGURE 3. Precision/recall obtained by transforming all 4 and 5 votes into P votes (positive) and all 1, 2 and 3 votes into N votes (non-positive), compared to the results obtained using the numerical values. 20% of test users, 20% of test items, K = 150, Pearson correlation, recommendation threshold = 4.

RS ratings matrices in search of great efficiencies and novel approaches.

There are two references [32], [56] in the CF RS field from which we have taken several concepts that we use in the proposed method: Paper [32] brings an experiment where ratings 1,2,3 are transformed into *N* (negative) votes, and ratings 4,5 are transformed into *P* (positive) votes. Results show (Fig. 3) that there is no worsening in the quality of the recommendation, and that there is an improvement when the number of recommendations is high. These results are extraordinarily useful to design simplified models that allow us to accelerate prediction times. For example, assigning values 1 to positive votes and values 0 to negative votes, we could simplify the *Mean Squared Differences (MSD)* metric, replacing it to the *Mean Absolute Differences (MAD)*: we would lower prediction and recommendation execution times without changing results quality.

$$MSD(u, v) = \frac{1}{I_c} \sum_{i \in I_c} (r_{i,u} - r_{i,v})^2$$

where I_c is the set of common items voted for both u and v users.

Paper [56] makes use of a simplified mechanism to compare the numerical values of the active user with respect to the neighbor candidate. In this paper, the number of items in which the ratings difference between active user and neighbor is zero is counted, the number of items in which the ratings difference is one, and so on up to the number of cases with difference four. Top of Fig. 4 shows a datatoy example of the [56] similarity measure. If there are many items involving 0 or 1 differences, the active user and the neighbor candidate will be similar. On the contrary: many cases with 4 or 3 differences will indicate that the active user and the neighbor candidate will not be similar. If we choose properly a series of weights for each case, the similarity measure will give us the best results. In [56] weights values are obtained using genetic algorithms optimization. The genetic fitness function is the RS MAE.

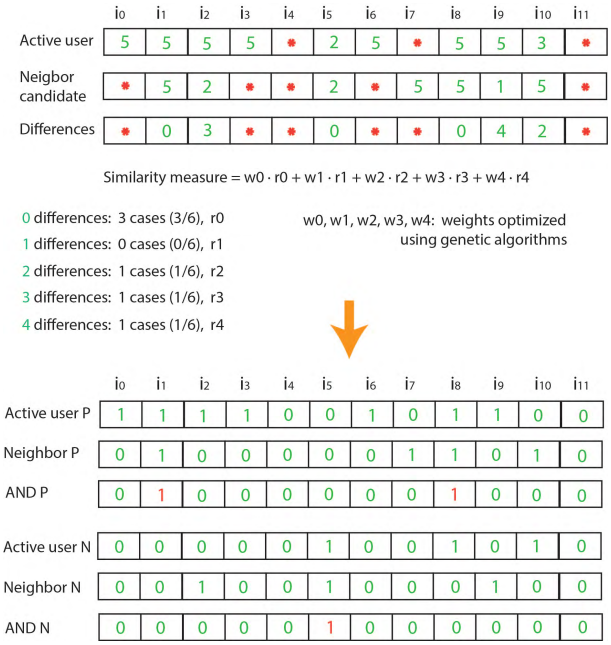

FIGURE 4. Top of the figure: similarity measure from [56]; bottom of the figure: proposed similarity measure approach where ratings {1,2,3} are transformed into *N* (negative) votes, and ratings {4,5} are transformed into *P* (positive) votes [32].

The bottom graph in Fig. 4 shows part of the proposed method design, merging [32] and [56] concepts. For each user we create two sets: the positive set *P* and the negative set *N*. Set *P* contains items voted as relevant 4,5, whereas set *N* contains items voted as non-relevant 1,2,3. Each item belonging to a set is represented by the number one; each item not belonging to a set is represented by the number zero. A simple and very efficient similarity approach is to make the AND function of each active user and neighbors candidate sets; i.e.: Active user *P* AND neighbor *P*, Active user *N* AND neighbor *N*. The greater the cardinality of the result sets (red ones in Fig. 4), the greater the similarity of the active user and the neighbor candidate.

Currently, the most used programming languages have access to libraries containing *BitSet* implementations. *BitSet* objects implement a vector of bits that grows as needed. Each component of the *BitSet* has a Boolean value. Individual indexed bits can be examined, set, or cleared. One *BitSet* may be used to modify the contents of another *BitSet* through logical AND, logical inclusive OR, and logical exclusive OR operations. *BitSet* objects are very efficient: they use light data structures and they perform the logical functions in short times. To design a similarity measure based on *BitSet* logical operations is a promising approach to reach the objectives of this paper.

Paper [32] shows the importance of the *Jaccard* similarity measure in the CF RS field. *Jaccard* uses the structural information of each user votes; it does not use the ratings numerical values. Fig. 5 shows the inverse relationship that exists between *Jaccard* values and *MAE* results: the lowest

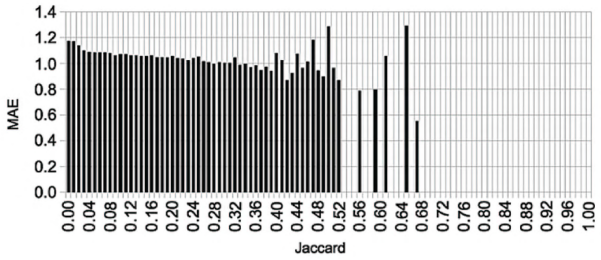

FIGURE 5. (From [32]) *MAE* improvement using *Jaccard* similarity measure. X-axis shows *Jaccard* values. Dataset: Movielens 1Mbyte.

value of *Jaccard* generates an absolute error of 1.18, while the highest value of *Jaccard* produces an absolute error of 0.87; this is a 26% improvement. Paper [32] combines this behavior to a similarity measure based on numerical values; authors obtain the *JMSD* metric, that reports very good results.

In summary, our proposed metric will be based on the following:

- To transform ratings to relevant (positive) and non-relevant (negative) values [32], [36].
- To make use of the *Jaccard* similarity measure to exploit non-numerical information [32].
- To make use of the numerical comparisons explained in [56].
- To convert ratings values into zero and one values.
- To use a design that efficiently exploit the existing implementations of binary operators.

B. METHOD FORMALIZATION AND IMPLEMENTATION

We store the following model of each user on the RS dataset (ratings: 1 to 5):

Set of relevant ratings of the user u:

$$P_u = \{i \in I | r_{u,i} \in \{4, 5\}\} \tag{1}$$

- Where I is the set of items, and $r_{u,i}$ is the rating of user u to item i.
- Examples of the P_u set are the Fig. 4 "Active user P" and "Neighbor P" sets.

Set of non-relevant ratings of the user u:

$$N_u = \{i \in I | r_{u,i} \in \{1, 2, 3\}\} \tag{2}$$

- Examples of the N_u set are the Fig. 4 "Active user N" and 1"Neighbor N" sets.

Set of all the votes that user u casted (total ratings):

$$T_u = P_u \cup N_u \tag{3}$$

Now we expose the proposed similarity measure equations: We use the sets:

$$I_u = \{i \in I | r_{u,i} \neq \bullet\} \tag{4}$$

- where $\bullet$ means absence of this rating. $r_(u, i) \neq \bullet$ means that user u has casted item i.

Jaccard to compare user u and user v is defined as:

$$Jaccard(u, v) = \frac{\#(I_u \cap I_v)}{\#(I_u \cup I_v)} \tag{5}$$

The *Jaccard* equation for our proposed method is formalized as:

$$Jaccard(u, v) = \frac{\#(T_u \cap T_v)}{\#(T_u \cup T_v)}, \quad Jaccard(u, v) \in [0..1] \tag{6}$$

The *Jaccard* similarity measure is the ratio between the number of common ratings voted by both users and the number of ratings voted by at least one user.

To compute numerical differences, we use the differences from both, the relevant ratings of user u and user v, and the non-relevant ratings of user u and user v. We call *Numerical Relevance NR* to this section of the proposed similarity measure.

Meaning of each term in NR (equation (7)), as shown at the bottom of the next page:

- $\#(P_u \cap P_v)$: Number of common relevant, {4, 5}, ratings voted by both users. These are similarity hits.
- $\#(N_u \cap N_v)$: Number of common non-relevant, {1, 2, 3}, ratings voted by both users. These are similarity hits
- $\#(P_u \cap N_v)$ and $\#(N_u \cap P_v)$: Number of common ratings voted differently by both users. These are similarity failures.
- $\#(T_u \cap T_v)$: Number of ratings voted by both users. Used to normalize results in the range $[-1..1]$.

The proposed similarity measure *Jaccard and Numerical Relevance JNR* (equations (8) and (9)), as shown at the bottom of the next page, in a first approximation, combines equations (6) and (7).

At this point we consider the incidence, in the results, of the factors: $(P_u \cap N_v)$ and $(N_u \cap P_v)$. These factors do not help to find neighbors that are similar, although they can be used to discard neighbors that behave as similar in some items, while in other items they show opposite preferences. The main questions are: Can these terms be unnecessary in practically the majority of situations? Will we get a significative prediction time improvement? Our hypothesis is that, usually, we will find neighbors with no opposing votes to the active user ones, so the terms $(P_u \cap N_v)$ and $(N_u \cap P_v)$ will have a residual influence on the accuracy results. We call *Efficient Jaccard and Numerical Relevance EJNR* to the simplified similarity measure:

$$EJNR(u, v) = \frac{\#(P_u \cap P_v) + \#(N_u \cap N_v)}{\#(T_u \cup T_v)} \tag{10}$$

$d(u, v) = 1 - EJNR(u, v)$ is a metric, or distance function:

1. $d(x, y) \geq 0$
   $C \geq 0$, where C is a set
2. $d(x, y) = 0 \Leftrightarrow x = y$

$d(u, v) = 0 \Rightarrow \#(P_u \cap P_v) + \#(N_u \cap N_v) = \#(T_u \cup T_v) \Rightarrow T_u = \#(P_u \cup N_u), T_v = \#(P_v \cup N_v)$

TABLE 2. ***MAE*** **and** ***Prediction Time*** **of the proposed** ***JNR*** **and** ***EJNR*** **similarity measures. The percentage of** ***EJNR*** **gain over** ***JNR*** **is also shown. Dataset: Movielens 1Mbyte.**

		Prediction Time (seconds)	
K	**EJNR**	**JNR**	**% Improvement**
50	0,854	1,234	30,739
100	0,881	1,286	31,509
150	0,931	1,341	30,559
200	0,992	1,394	28,834
250	1,033	1,449	28,747
300	1,089	1,494	27,112
350	1,144	1,543	25,878
400	1,187	1,598	25,726
450	1,233	1,652	25,348
500	1,280	1,702	24,774
			MAE
K	**EJNR**	**JNR**	**% Improvement**
50	0,729	0,732	0,391
100	0,729	0,731	0,256
150	0,731	0,733	0,319
200	0,735	0,735	0,000
250	0,738	0,737	-0,175
300	0,740	0,738	-0,223
350	0,742	0,739	-0,421
400	0,743	0,741	-0,372
450	0,745	0,742	-0,399
500	0,747	0,743	-0,488

3. $d(x, y) = d(y, x)$

$A \cup B = B \cup A, A \cap B = B \cap A$, where A and B are sets

4. $d(x, z) \leq d(x, y) + d(y, z)$

$\frac{(P_u \cap P_v)+(N_u \cap N_v)}{(T_u \cup T_v)} = \frac{(P_u \cap P_v)}{(T_u \cup T_v)} + \frac{(N_u \cap N_v)}{(T_u \cup T_v)}$, Jaccard is a metric and the composition of two Jaccard metrics is a metric on the collection of all finite sets.

Table 2 shows the *Mean Absolute Error* and the *Prediction Time* of both *JNR* and *EJNR* similarity measures when applied to Movielens 1M dataset. From Table 2, we observe the following:

- A significant improvement in the necessary time to make predictions (around 27%).
- No improvements in the prediction accuracy (around -0.1%).
- A very small tendency towards drop in accuracy when increasing the number of neighborhoods. This result is logical, because as the appropriate neighbors are finished, the terms not eliminated in *JNR* perform an increasingly important work with the rest of the candidates.
- A small tendency towards drop in prediction time improvement when increasing the number of neighbors.

Since the *JNR* accuracy is practically the same as that of the *EJNR*, and the prediction time of *EJNR* is much better than the *JNR* one, we choose *Efficient Jaccard and Numerical Relevance EJNR* as our proposed similarity measure. Equation (11) shows *EJNR* customized for its implementation through *BitSet* structures: set intersections are implemented using *AND* operators, while joint sets are implemented using the *OR* operator.

$$EJNR(u, v) = \frac{\#(P_u \wedge P_v) + \#(N_u \wedge N_v)}{\#(T_u \vee T_v)} \tag{11}$$

C. METHOD OPTIMIZATION

The proposed method is executed with great efficiency thanks to the combination of two elements: 1) The small model in which it is sustained, and 2) The efficiency of the binary processing approach, implemented through regular *BitSet* data structures. However, there is a circumstance in which the proposed method can become less efficient than the usual memory-based one: when there exists an extraordinarily high sparsity level. Fig. 6 shows this concept: in non-high sparse situations, traditional memory-based methods must perform costly calculations for each of the items in which both active user and neighbor candidate have issued ratings. In these cases, the proposed method easily outperforms the traditional performances. However, there are situations in which the levels of sparsity are particularly high, as outlined in the lower part of Fig. 6. In these situations, traditional memory-based algorithms and similarity measures can obtain better performances than the proposed method. This is due to the following reasons: 1) The existing k-nearest neighbors implementations are very efficient, such as the one we use [57]: we do not even need to iterate through all the dataset items, and 2) Our binary approach needs a fixed computation time, which is independent of the sparsity level of the data, and it is dependent of the number of items in the dataset.

KNN implementations do not work, in memory, on huge ratings matrixes; they act on datasets files that contain only the existing ratings, which are a very small portion of all possible ones. To find the similarity between an active user and its potential neighbors, the KNN implementations iterate between the list of ratings of the active user and the list of ratings of each neighbor. The smaller the size of these lists, the greater the speed with which the neighborhood of an active user is obtained. In this way, the sparser the dataset is the faster the similarity calculations will be made (on average). As an example, for a dataset containing 2500 items: in case a) we have an active user that contains 20 ratings and

$$NR(u, v) = \frac{\#(P_u \cap P_v) + \#(N_u \cap N_v) - \#(P_u \cap N_v) - \#(N_u \cap P_v)}{\#(T_u \cap T_v)}, \quad NR(u, v) \in [-1..1] \tag{7}$$

$$JNR(u, v) = Jaccard(u, v) \cdot NR(u, v) \tag{8}$$

$$JNR(u, v) = \frac{\#(P_u \cap P_v) + \#(N_u \cap N_v) - \#(P_u \cap N_v) - \#(N_u \cap P_v)}{\#(T_u \cup T_v)} \tag{9}$$

TABLE 3. Proposed method complexity analysis. #user = M, #item = N, #Rated items = Nsparse $\ll N$, #neighbors = K, #factors = F, #iterations = I, $\alpha \in [0, 1]$, c = BitSetOperation.

	Time Complexity	Space Complexity
User based KNN	$O(M^2 \times N) + O(KlogK \times M)$	$O(M \times (N + K))$
Sparse User based KNN	$O(M^2 \times Nsparse) + O(KlogK \times M)$	$O(M \times (Nsparse + K))$
Proposed method	$O(M^2 \times (\alpha \times Nsparse + (1 - \alpha) \times c)) + O(KlogK \times M)$	$O(M \times (Nsparse + K)) + O(M \times Bitset(N))$
Matrix Factorization based methods	$O(M \times N \times F^2 \times I)$	$O(M \times N + F \times (M + N))$

FIGURE 6. Dense and sparse situations: Traditional KNN methods only process the green marked "Processing" items. Our proposed method efficiently processes all the dataset items, as a set. When there are extremely sparse scenarios, the proposed method can worsen performance.

a neighbor candidate that contains 50 ratings (10 common ratings). In case b) the active user contains 900 ratings and the neighbor candidate contains 600 ratings (400 common ratings). Case b) needs a longer execution time than case a), not only because it requires a greater number of computations (400 versus 10), but also because it needs to process longer lists (900+600 versus 20+50). On the contrary, the optimization method proposed in this paper performs a very efficient operation (bit level) on the complete set of items: in our example, 2500 items. It is very probable that the proposed optimization method will run faster than case b), but slower than case a).

In summary, our method simultaneously processes all items, while traditional methods process individually each necessary item. When there are very few items to process, the k-nearest neighbors is more efficient. The optimization approach that we adopt is to apply the appropriate method for each case: i.e.: to apply the traditional KNN in high sparsity cases, and to apply the proposed method in more dense cases. Therefore, it is important to properly choose the density threshold from which we apply the proposed method. As an example, in Fig. 7 we show the prediction time effect that produces to establish the threshold in diverse values of density. As can be seen, in this case the optimum threshold value is located at a density of 0.6% (99.4% sparsity). This means that we apply our method when density is higher than 0.6%, and we apply the traditional KNN method when density is lower than 0.6%. In this particular dataset (Movielens 1M), when density is higher than 0.6%, the proposed method simultaneously processes the set of items in a more efficient way than the traditional KNN approach. Table 5 shows each density threshold obtained for the tested datasets.

Finally, the formalization of the explained optimization is:

ProposedEJNR

$(\mu < ActiveUser_\theta) \wedge (\mu < NeighborCandidate_\theta)$

TraditionalKNN

$(\mu \geq ActiveUser_\theta) \vee (\mu \geq NeighborCandidate_\theta)$

Where θ means density; i.e.: active user density and neighbor candidate density. μ is the threshold density (0.6 for the Fig. 7, Movielens 1M dataset).

D. ALGORITHM DESCRIPTION

To illustrate the proposed method functioning we provide its algorithmic description (algorithm 1) and its complexity analysis (Table 3). Complexity analysis compares: a) Matrix Factorization methods, b) KNN memory-based algorithms, and c) Proposed method.

From Table 3 we can compare the proposed method complexity with the traditional KNN on sparse datasets. Basically, we claim the superiority of the proposed method, based on the hypothesis that:

$Nsparse > (\alpha \times Nsparse + (1 - \alpha) \times c)$

The above equation is fulfilled due to two reasons:

1. There are situations in which the BitSet operation (c) is faster than the traditional processing. This is particularly true when specific hardware is used to do this task.

2. The proportion of fast BitSet operations $(1 - \alpha)$ is significant. As we will see in the experiments results section, the reduction of computational times is verified, and therefore the adequate proportion of the term $(1 - \alpha)$.

IV. EXPERIMENTS AND RESULTS

This section first explains the experiments methodology and design: baselines, datasets, parameters values and quality measures. Subsequently, it shows the results of the experiments, and finally it makes a brief discussion. Results of the experiments, in turn, are mainly divided into: a) Performance results (training and prediction times), and b) Accuracy results (prediction and recommendation qualities).

A. EXPERIMENTS DESIGN

The designed experiments compare the proposed method with several representative baselines, using adequate testing

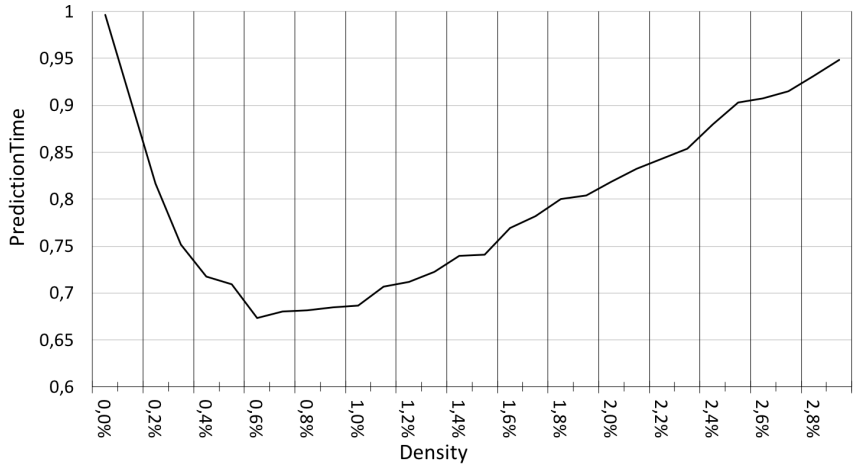

FIGURE 7. Optimization impact on prediction times when different density thresholds are set. Dataset: Movielens 1M. Low prediction times are better.

TABLE 4. Main properties of the datasets used in the experiments. We use reduced *Netflix* and *Amazon Music* versions.

Datasets	Number of ratings	Number of items	Number of users	Rating values	Results offered in
MovieLens 100K [60]	96,517	1,682	943	5-star scale, with one-star increments	Additional material
MovieLens 1M [60]	1,000,209	3,952	6,040	5-star scale, with one-star increments	Additional material
Amazon Music*	1,014,484	23,185	332,083	5-star scale, with one-star increments	Paper
MovieLens 10M [60]	9,621,849	10,677	69,878	5-star scale, with one-star increments	Paper
MovieLens 20M [60]	19,188,554	26,744	138,493	5-star scale, with one-star increments	Paper
FilmTrust [61]	33,470	2,071	1,508	0.5 to 4 with half increments	Paper
Netflix* [62]	4,774,504	17,730	24,222	5-star scale, with one-star increments	Paper
Jester [63]	4,100,000	100	73,421	[-10,...,10] scale	Paper

TABLE 5. Collaborative filtering baselines used to compare results with the proposed similarity measure.

Baseline type	Baseline	Acronym	Results offered in
Model based	Probabilistic Matrix Factorization [64]	PMF	Paper
	Bayesian Matrix Factorization [31]	BMF	
Statistical	Pearson correlation	COR	Additional material
	Constrained correlation	CCOR	
	Cosine	COS	
	Adjusted cosine	ACOS	
	Jaccard	JAC	
	Mean Squared Differences	MSD	
	Spearman rank	SPE	Paper
Current CF similarity measures	JMSD [33]	JMSD	Paper
	Singularites [37]	SING	
	NHSM [36]	NHSM	
	Proximity-Impact-Popularity [34]	PIP	
	Genetic Algorithms-based [58]	GA	Additional material

quality measures and processing diverse public CF datasets. Table 4 shows the tested datasets, and Table 5 shows the chosen CF baselines. Experiments have made use of the Original Software Publication (OSP) framework [57]. To perform the experiments, we have split users and items into test and training sets. To avoid fluctuations due to the random selection of training users and test items, we perform each experiment using 5-folds cross-validation.

Table 5 shows the baselines we use to compare the proposed method with traditional and with state of the art CF similarity measures and methods. We have classified the baselines in three groups: a) Model-based matrix

Algorithm 1 EJNR

Require: $U = \{user\}$, $I = \{item\}$, $R = \{r_{u,i}\}$, $B = \{BitSetRatings\}$,
$BP = \{BitSetPositiveRatings\}$, $BN = \{BitSetNegativeRatings\}$
Initialization:
  for $u \in U$ do
    for $i \in I$ do
      $B_u(i) \leftarrow 1$
      if $r_{u,i} > RatingThreshold$ then
        $BP_u(i) \leftarrow 1$
      else
        $BN_u(i) \leftarrow 1$
      end if
    end for
  end for
Recommendation based on neighborhood:
  for $u_a \in U$ do
    for $u_t \in U$ do
      if $|R_{u_a}| > SparsityThreshold$ and $|R_{u_t}| > SparsityThreshold$ then
        $common \leftarrow |B_{u_a} \wedge B_{u_t}|$
        $positive \leftarrow |BP_{u_a} \wedge BP_{u_t}|$
        $negative \leftarrow |BN_{u_a} \wedge BN_{u_t}|$
      else
        Do the same using sparse indexing
      end if
      union $\leftarrow |R_{u_a}| + |R_{u_t}| - common$
      $Sim(u_a, u_t) \leftarrow \frac{positive+negative}{union}$
    end for
  end for
for $u \in U$ do
  $Neighbor_u \leftarrow Top\ K\ Similarity(u)$
  $Prediction_u \leftarrow WeightedAverage(Neighbor_u)$
  $Recommendation_u \leftarrow Top\ N\ Prediction(Prediction_u)$
end for

factorization methods, b) Classical similarity measures coming from the statistical field, and c) State of the art CF similarity measures and methods. The column "Acronym" shows the acronym we use in the results figures and tables contained in the next section. The M-Distance [39] similarity measure has not been included as baseline, due to its particular neighbor selection, based on similarity threshold.

Table 6 shows the different values of the parameters that we used to perform the experiments: number of neighbors in the KNN processing, number of recommendations, number of factors in the model-based baselines, cross-validation testing and training sets percentages, precision and recall thresholds, etc.

We have chosen the CF recommendation quality measures: *Precision* and *Recall*, and the ranked recommendation quality measure *Normalized Discount Cumulative Gain (NDCG)*. Due to the particular importance given to the *prediction times* in this paper, we have completed the set of quality measures testing these values. Additionally, the *training times* are computed in the model-based methods. Table 7 summarizes the tested quality measures.

B. EXPERIMENTS RESULTS

In this section we test the proposed *EJNR* similarity measure. We compare its results with the Table 5 baselines ones. The CF public datasets used are those defined in Table 4. In order to show just the most relevant information, the less representative results are shown only as "Additional material". Tables 4 & 5 include a column indicating this circumstance. All results obtained using the MovieLens 100K dataset are shown only as "Additional material". Table 7 summarizes the tested quality measures.

We are looking for a method that balances execution times (prediction times and training times) with recommendation accuracy. So, we show comparative results from each of these objectives.

TABLE 6. Parameters values involved in the cross-validation experiments. Datasets: ML means MovieLens, NF means NetFlix, FT means FilmTrust, and JT means Jester. MovieLens 100K results are only shown as "Additional material."

	ML100K	ML1M	ML10M	ML20M	NF5M	FT	JT
General parameters							
Testing-Items%	20%						
Test-Users%	20%						
Training-Items%	80%						
Training-Users%	80%						
#Neighbors	300						30
#Recommendations	{1,...,15}						
Precision & Recall thresholds	4					3	3,5
Precision Recall #Neighbors	200						
Factorization parameters							
#Factors	50						
#Iterations	80						
Lambda (PMF)	0,055						
Alpha (BMF)	0,8						
Beta (BMF)	5						
Proposed Similarity Measure parameters							
Rating threshold	3						
Density threshold %	0,8	0,6	0,1	0,1	0,3	100	0

TABLE 7. Tested quality measures.

Performance	Recommendation quality
Model-Training time	Precision
Prediction time	Recall
	NDCG

1) TIME TO TRAIN THE MODEL

As shown in Fig. 1, CF RS can be tackled without the use of models, or using tiny, small or large models. Rows in Table 7 are ordered according to this concept: The firsts four statistical similarity measures are memory-based, and then they do not use models. The baselines *M-Distance* and *Singularities* require tiny models. The proposed *JNR* and *Efficient JNR* use a small model. Finally, *Probabilistic* and *Bayesian MF* models can be classified as large or heavy ones.

In short, the proposed *EJNR* uses more resources than the classical similarity measures, it also uses more resources than *M-Distance*, approximately the same as *Singularities*, and much less resources than the MF approaches. Therefore, it is expected that *EJNR* accuracy results will be better than the baselines ones, except for *PMF* and *BMF*. There is an inverse relationship between training times and prediction times; to study the balance between both times, we also show the obtained prediction times.

2) PREDICTION TIMES

Our proposed *EJNR* method aims to obtain an adequate balance between accuracy and performance (prediction time and model maintenance time). The heavy model-based methods such as *PMF* and *BMF* present a model maintenance time much larger than the *EJNR* small model (Table 8); however, the prediction time of the MF-based methods is unbeatable: it is based on a simple dot operation of K hidden factors. Fig. 8 shows MF-based methods on the right of the graphs, using gray colors. As it can be seen, their prediction times are very low.

Fig. 8 shows that the design made for our *EJNR* method manages to significantly improve the prediction time with respect to all the baselines considered (except the heavy-model based ones). As expected, the denser the dataset (*Jester*, in our case), the greater the improvement obtained. It is also important to note that the reputable PIP, NHSM and SINGularities baselines have, comparatively, high prediction times. From Fig. 8 it is also seen how *EJNR* improves the *JNR* prediction times. It must be remembered that *EJNR* (11) is the "time-Efficient" version of the JNR method (9).

3) RECOMMENDATION ACCURACY

From the *Precision* results, shown in Fig. 9 the proposed *EJNR* method achieves a recommendation accuracy equal to or greater than the considered baselines (with the exception of the model-based methods). It stands out the superiority achieved with respect to the *SINGularity* method and with respect to the classical similarity measures, which in the graphs are represented by *Spearman Rank* (the remaining classical similarity measures can be consulted in the "Additional material"). It is interesting to highlight the particularly good behavior of the model-based methods when applied to the *Amazon Music* dataset; this is due to the extremely high sparsity of this dataset, that only contains one million ratings casted for 332,000 users to 23,185 items. It can be compared (additional material) with *Movielens 1M*, that contains one million ratings casted for 6,040 users to 3,952 items. *Precision* results on *Amazon Music* show the good performance of the proposed *JNR* and *EJNR* methods when applied to sparse datasets, compared to the state of the art memory-based methods.

Recall results, shown in Fig. 10, show a similar comparative behavior to the *Precision* ones (Figure 9). In this case, the *Amazon Music* dataset shows, even more clearly, the superiority of model-based methods when applied to sparse datasets. It is important to highlight the concept that

TABLE 8. Times (seconds) to train or set each model. Datasets: ML: MovieLens, NF: NetFlix, FT: FilmTrust, and JT: Jester. Rows: Table 5 similarity measures and methods. Columns: Table 4 datasets. For experiments efficiency reasons, BMF has not been processed on the heavy ML10M and ML20M datasets.

	FT	ML100K	JT	ML1M	NF	ML10M	ML20M
SPE	0	0	0	0	0	0	0
JMSD	0	0	0	0	0	0	0
PIP	0	0	0	0	0	0	0
NHSM	0	0	0	0	0	0	0
SING	0,0064	0,0082	0,0192	0,0306	0,1956	0,2418	0,7
JNR	0,0062	0,0106	0,0618	0,1016	0,7024	1,5762	3,471
EJNR	0,0056	0,0098	0,0608	0,098	0,697	1,4824	3,5114
PMF	0,6064	1,247	10,5682	12,5668	94,9752	257,11	974,93
BMF	49,8658	91,7712	355,8018	657,402	2331,7668	–	–

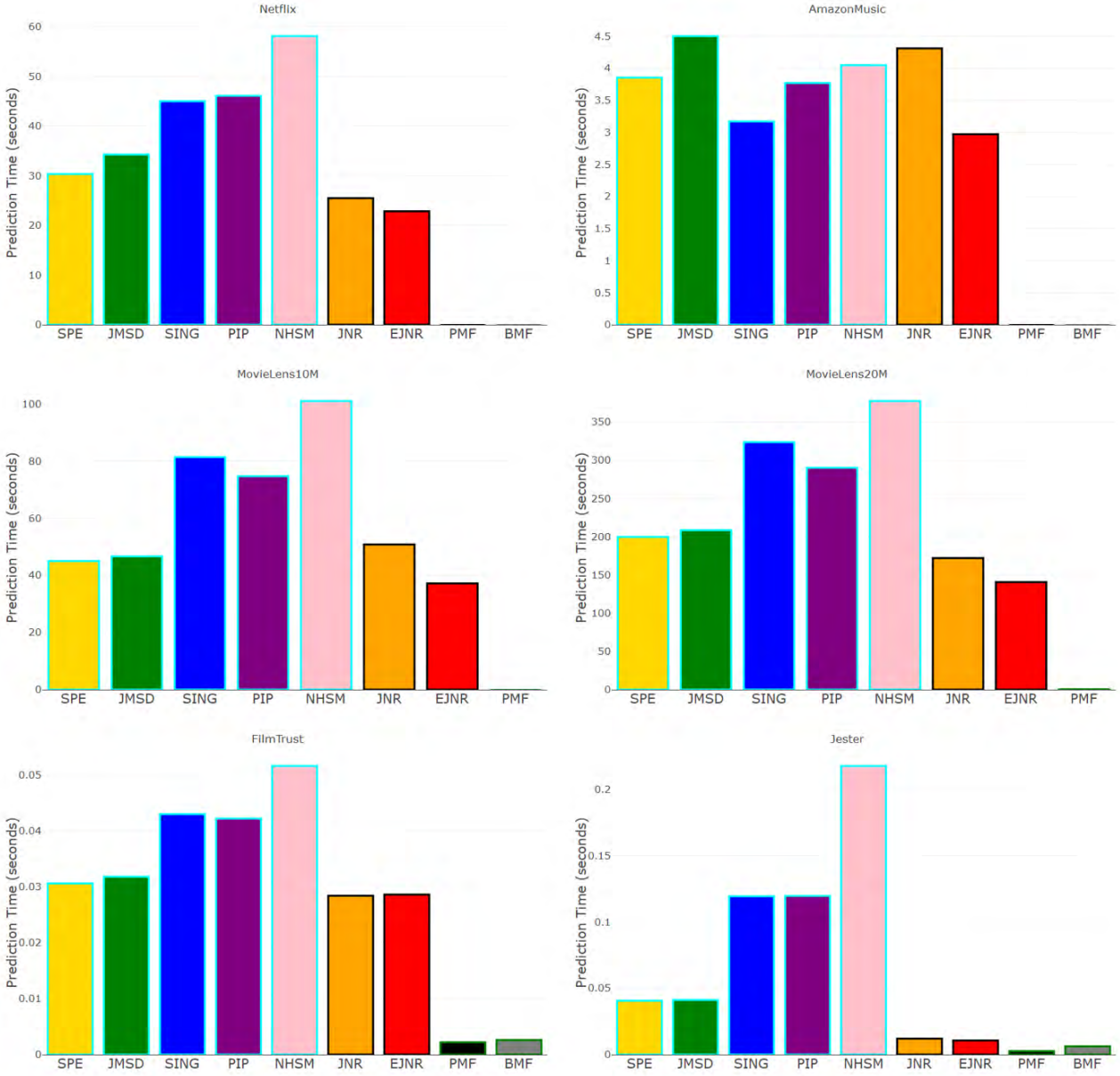

FIGURE 8. Prediction time (seconds). The smaller the values, the better the results. Datasets: Table 4. Baselines: Table 5 similarity measures and methods.

the proposed similarity measures are not designed to improve accuracy on all the possible scenarios: they are designed to balance performance and accuracy, providing fast model setting and fast prediction times, combined with similar or better accuracy results compared to the current memory-based methods.

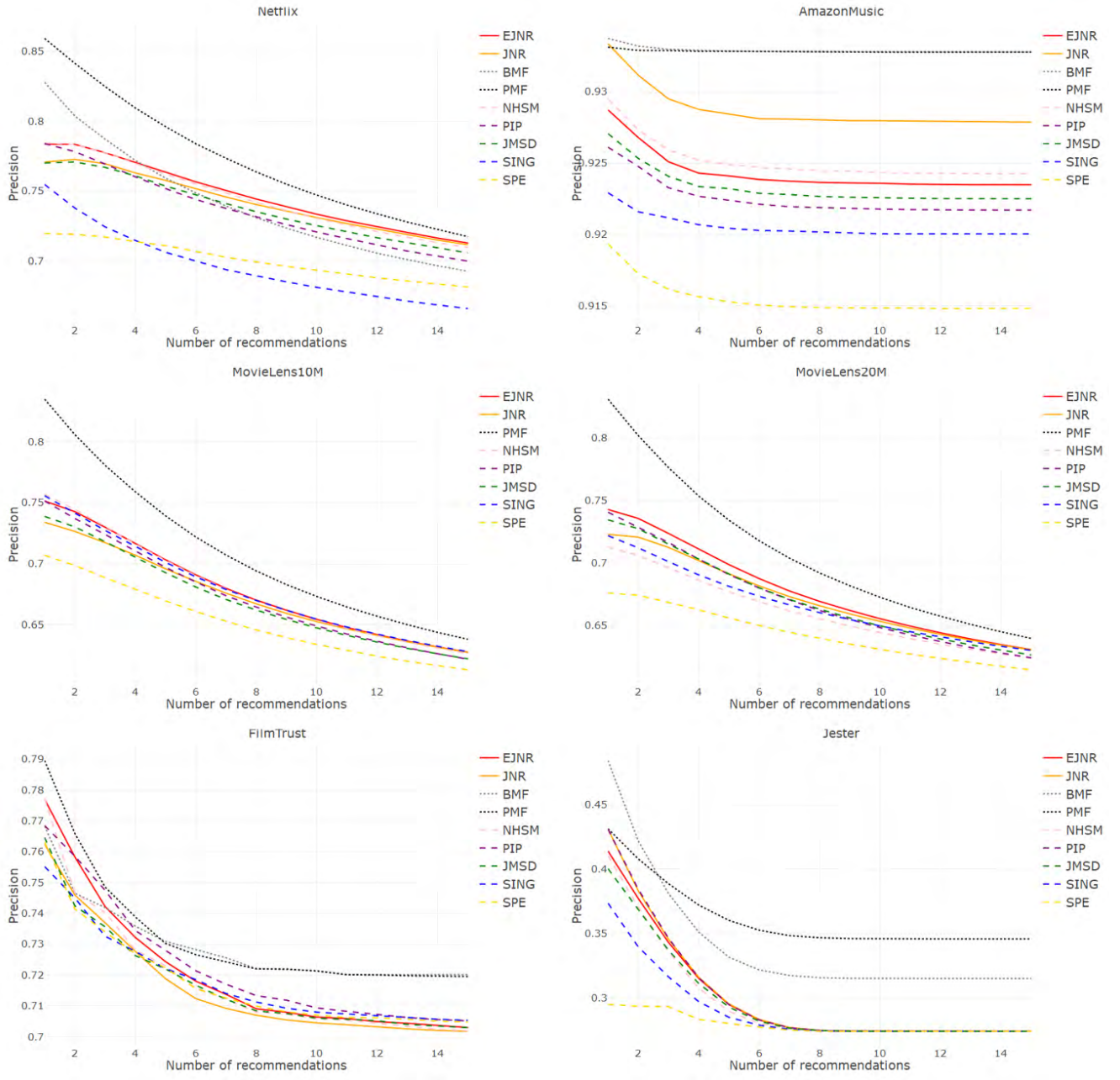

FIGURE 9. Recommendation accuracy (*Precision*). The larger the values, the better the results. Datasets: Table 4. Baselines: Table 5 similarity measures and methods.

In Fig. 11, the quality of the recommendations is tested according to the order in which the recommendations are presented. The basic idea is that we are more permissive with the failures of the last recommendations of a list than with the failures of the first recommendations of that list. To test the quality of the list (the ranking), the classic *Normalized Discount Cumulative Gain (NDCG)* measures de usefulness (gain) of recommendations based on their position in the list. The *NDCG* results from Fig. 11 show, once again, the pattern exhibited by the *Precision* and *Recall* quality measures: a) Superiority of the model-based methods, especially in very sparse datasets, b) Better averaged behavior of the proposed *JNR/EJNR* similarity measures compared to the current memory-based methods (*Singularities*, *PIP*, *JMSD*, etc.), and c) Worst results of the traditional statistics metrics (*Pearson*, *Cosine*).

C. DISCUSSION

Given the obtained performance and accuracy results, we can determine the following considerations:

- When the implantation, training and updating of a heavy-model is not a problem, the MF model-based

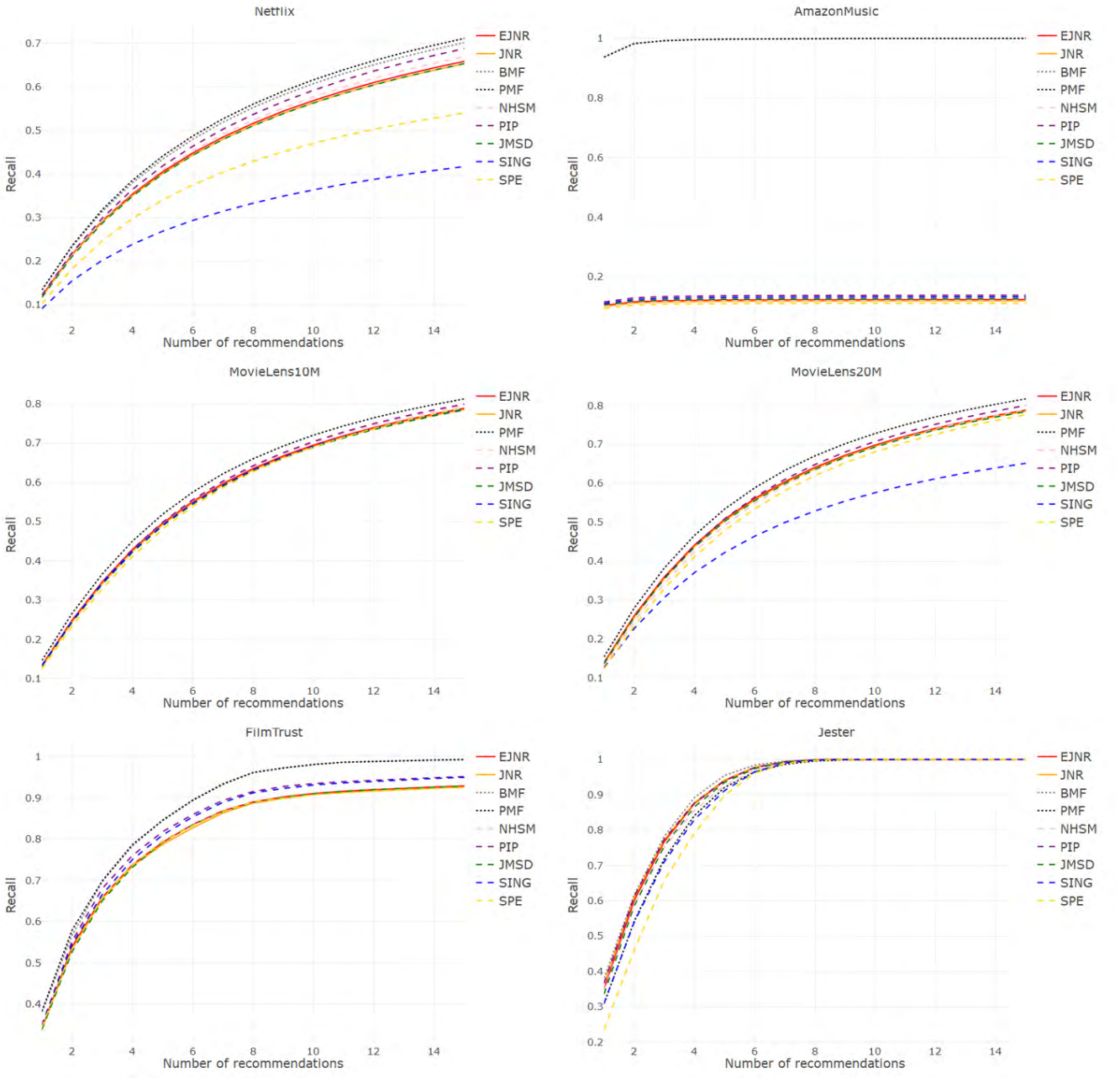

FIGURE 10. Recommendation accuracy (*Recall*). The larger the values, the better the results. Datasets: Table 4. Baselines: Table 5 similarity measures and methods.

methods are the best CF RS approaches. However, there are cases in which the use of an MF model-based method is not considered adequate: a) When the explanation of recommendations is important, b) When you want to establish reliability values associated with each prediction or recommendation, c) When it is necessary to always work with the model fully updated, d) In cases where the maintenance of a model is considered complex or expensive, etc.

- When we choose to use a CF RS method not based on MF, our *EJNR* offers an adequate accuracy/performance balance. The results obtained show values of prediction time, prediction accuracy and recommendation accuracy better than those provided by modern baselines.
- The design approach based on the *numerical relevances* and the *non-numerical structure* of the ratings has been shown to improve the quality of accuracy. The establishment of a small model has improved prediction times. The additional use of set operations, implemented by means of *BitSet* structures, provides additional optimization that further improves the prediction times.

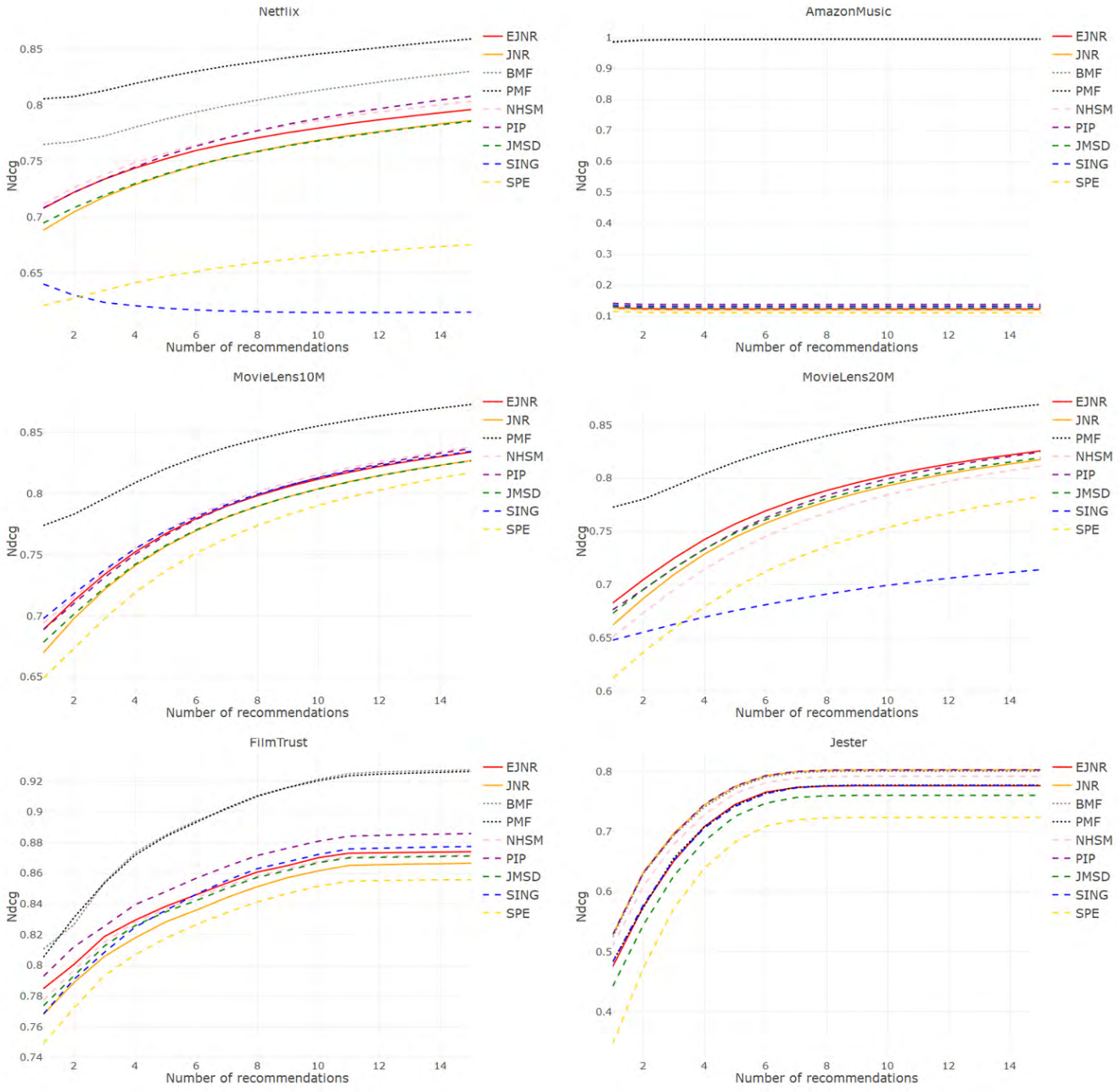

FIGURE 11. Recommendation accuracy (*NDCG*). The larger the values, the better the results. Datasets: Table 4. Baselines: Table 5 similarity measures and methods.

- Although the proposed method is *EJNR*, in cases where accuracy improvement is more important than performance improvement, the *JNR* method can be used.
- Choosing model-based methods we need big training times (Table 8, ML-20M, PMF: 974 s.), but we obtain very short prediction times (Figure 8, ML-20M). Conversely, choosing memory-based methods we do not need to make a training task (Table 8, ML-20M, JMSD: 0 s.), nor re-training processes, but we require large prediction times (Figure 8, ML-20M). The proposed similarity measure, from the time process outlook, provides an intermediate solution (Table 8, ML-20M, EJNR: 3,5 s., Figure 8, ML-20M). It needs little training/setting time compared with model-based methods (3,5 s. vs. 974 s.) and it provides the advantages of memory-based methods (explanation of recommendations, etc.). Additionally, the proposed similarity measure gets better prediction times than current memory-based approaches (Figure 8). Using huge Big Data datasets, we should to: 1) Evaluate training-times, re-training times, and prediction times, and 2) Set restrictions: maximum prediction time, maximum

degree of model outdating, maximum training time, maximum retraining time, need of reliability measures, need of recommendation explanations, etc. Is up to the Big Data RS designers to choose model-based approaches or intermediate solutions such as the proposed one.

V. CONCLUSIONS

The classic differentiation between model-based and memory-based methods becomes diffuse when we design what in this paper has been called: ''tiny or small model-based methods'', as opposed to ''heavy'' model-based methods, such as the matrix factorization one. Tiny or small model-based methods build a simple model that allows: 1) To simplify the design of the similarity measures, and 2) To accelerate the prediction and recommendation processing.

The proposed method shares some properties with the model-based ones: a) It needs a (small) model, b) It needs to update the model, c) Prediction times are faster than using the memory-based similarity measures, and d) Accuracy is high. Unlike model-based approaches, the proposed method is processed using the k-nearest neighbors algorithm; for this reason, its prediction time, while being better than in memory-based approaches, is not as good as the model-based one. In summary, the proposed method shares memory and model-based properties. Its use is adequate when maintaining a ''small-model'' is acceptable, and when we want to improve the memory-based results, especially its performance.

The proposed method bases its good results on a series of concepts that coexist adequately with each other: 1) Combination of numerical and structural information, 2) Processing simplification: classifying ratings as relevant and non-relevant (numerical relevance), 3) Design based on sets, and implementation based on BitSets, and 4) Storage of a small model to speed up prediction times.

As future works, we propose the use of the proposed method in those fields where model-based approaches show their greatest difficulties: explanation of recommendations, recommendation to groups of users, reliabilities associated to predictions, real-time solutions, etc. In this context, it will be particularly interesting to make a re-design of the proposed similarity measure in order to defending from profile attacks and shilling attacks directed to collaborative filtering recommender systems.

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez, ''Recommender systems survey,'' *Knowl.-Based Syst.*, vol. 46, pp. 109–132, Jul. 2013.

[2] J. Lu, D. Wu, M. Mao, W. Wang, and G. Zhang, ''Recommender system application developments: A survey,'' *Decision Support Syst.*, vol. 74, pp. 12–32, Jun. 2015.

[3] M.-L. Wu, C.-H. Chang, and R.-Z. Liu, ''Integrating content-based filtering with collaborative filtering using co-clustering with augmented matrices,'' *Expert Syst. Appl.*, vol. 41, no. 6, pp. 2754–2761, 2014.

[4] M. Y. H. Al-Shamri, ''User profiling approaches for demographic recommender systems,'' *Knowl.-Based Syst.*, vol. 100, pp. 175–187, 2016.

[5] Y.-M. Li, C.-T. Wu, and C.-Y. Lai, ''A social recommender mechanism for e-commerce: Combining similarity, trust, and relationship,'' *Decision Support Syst.*, vol. 55, no. 3, pp. 740–752, 2013.

[6] J. Tang, X. Hu, and H. Liu, ''Social recommendation: A review,'' *Social Netw. Anal. Mining*, vol. 3, no. 4, pp. 1113–1133, 2013.

[7] J. Yu, M. Gao, W. Rong, Y. Song, and Q. Xiong, ''A social recommender based on factorization and distance metric learning,'' *IEEE Access*, vol. 5, pp. 21557–21566, 2017.

[8] Z. Yang, B. Wu, K. Zheng, X. Wang, and L. Lei, ''A survey of collaborative filtering-based recommender systems for mobile Internet applications,'' *IEEE Access*, vol. 4, pp. 3273–3287, 2016.

[9] Y. Shi, M. Larson, and A. Hanjalic, ''Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges,'' *ACM Comput. Surv.*, vol. 47, pp. 3:1–3:45, May 2014.

[10] J. Bobadilla, F. Ortega, A. Hernando, and A. Arroyo, ''A balanced memory-based collaborative filtering similarity measure,'' *Int. J. Intell. Syst.*, vol. 27, no. 10, pp. 939–946, 2012.

[11] M. Shang, X. Luo, Z. Liu, J. Chen, Y. Yuan, and M. Zhou, ''Randomized latent factor model for high-dimensional and sparse matrices from industrial applications,'' *IEEE/CAA J. Autom. Sinica*, pp. 1–11, Jul. 2018, doi: doi: 10.1109/JAS.2018.7511189.

[12] X. Luo, M. Zhou, S. Li, Z. You, Y. Xia, and Q.-S. Zhu, ''A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 579–592, Mar. 2016.

[13] X. Luo, M. Zhou, Y. Xia, Q. Zhu, A. C. Ammari, and A. Alabdulwahab, ''Generating highly accurate predictions for missing QoS data via aggregating nonnegative latent factor models,'' *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 3, pp. 524–537, Mar. 2016.

[14] X. Luo *et al.*, ''An incremental-and-static-combined scheme for matrix-factorization-based collaborative filtering,'' *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 1, pp. 333–343, Jan. 2016.

[15] X. Luo *et al.*, ''An efficient second-order approach to factorize sparse matrices in recommender systems,'' *IEEE Trans. Ind. Informat.*, vol. 11, no. 4, pp. 946–956, Aug. 2015.

[16] X. Luo, M. Zhou, Y. Xia, and Q. Zhu, ''An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems,'' *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1273–1284, May 2014.

[17] X. Guan, C. T. Li, and Y. Guan, ''Matrix factorization with rating completion: An enhanced SVD model for collaborative filtering recommender systems,'' *IEEE Access*, vol. 5, pp. 27668–27678, 2017.

[18] S. M. McNee, J. Riedl, and J. A. Konstan, ''Being accurate is not enough: How accuracy metrics have hurt recommender systems,'' in *Proc. Abstr. Hum. Factors Comput. Syst. (CHI)*, 2006, pp. 1097–1101.

[19] A. Said and A. Bellogín, ''Comparative recommender system evaluation: Benchmarking recommendation frameworks,'' in *Proc. 8th ACM Conf. Recommender Syst. (RecSys)*, New York, NY, USA, 2014, pp. 129–136.

[20] T. He, L. Hu, K. C. C. Chan, and P. Hu, ''Learning latent factors for community identification and summarization,'' *IEEE Access*, vol. 6, pp. 30137–30148, 2018.

[21] X. Luo, M. Zhou, M. Shang, S. Li, and Y. Xia, ''A novel approach to extracting non-negative latent factors from non-negative big sparse matrices,'' *IEEE Access*, vol. 4, pp. 2649–2655, 2016.

[22] L. Qiu, S. Gao, W. Cheng, and J. Guo, ''Aspect-based latent factor model by integrating ratings and reviews for recommender system,'' *Knowl.-Based Syst.*, vol. 110, pp. 233–243, Oct. 2016.

[23] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams, ''Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness,'' *ACM Trans. Internet Technol.*, vol. 7, no. 3, p. 23, Oct. 2007.

[24] I. Gunes, C. Kaleli, A. Bilge, and H. Polat, ''Shilling attacks against recommender systems: A comprehensive survey,'' *Artif. Intell. Rev.*, vol. 42, pp. 767–799, Dec. 2014.

[25] I. Gunes and H. Polat, ''Detecting shilling attacks in private environments,'' *Inf. Retr. J.*, vol. 19, pp. 547–572, Dec. 2016.

[26] T. K. Paradarami, N. D. Bastian, and J. L. Wightman, ''A hybrid recommender system using artificial neural networks,'' *Expert Syst. Appl.*, vol. 83, pp. 300–313, Oct. 2017.

[27] R. Yera and L. Martinez, ''Fuzzy tools in recommender systems: A survey,'' *Int. J. Comput. Intell. Syst*, vol. 10, no. 1, pp. 776–803, 2017.

[28] S. P. Perumal, K. Arputharaj, and G. Sannasi, ''Fuzzy family tree similarity based effective e-learning recommender system,'' in *Proc. 8th Int. Conf. Adv. Comput. (ICoAC)*, Jan. 2017, pp. 146–150.

[29] T. Horváth and A. C. P. L. F. de Carvalho, "Evolutionary computing in recommender systems: A review of recent research," *Natural Comput.*, vol. 16, pp. 441–462, Sep. 2017.
[30] A. Hernando, J. Bobadilla, and F. Ortega, "A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model," *Knowl.-Based Syst.*, vol. 97, pp. 188–202, Apr. 2016.
[31] J. Bobadilla, R. Bojorque, A. H. Esteban, and R. Hurtado, "Recommender systems clustering using Bayesian non negative matrix factorization," *IEEE Access*, vol. 6, pp. 3549–3564, 2018.
[32] J. Bobadilla, F. Serradilla, and J. Bernal, "A new collaborative filtering metric that improves the behavior of recommender systems," *Knowl.-Based Syst.*, vol. 23, no. 6, pp. 520–528, 2010.
[33] H. J. Ahn, "A new similarity measure for collaborative filtering to alleviate the new user cold-starting problem," *Inf. Sci.*, vol. 178, no. 1, pp. 37–51, Jan. 2008.
[34] Suryakant and T. Mahara, "A new similarity measure based on mean measure of divergence for collaborative filtering in sparse environment," *Procedia Comput. Sci.*, vol. 89, pp. 450–456, Jan. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050916311644
[35] H. Liu, Z. Hu, A. Mian, H. Tian, and X. Zhu, "A new user similarity model to improve the accuracy of collaborative filtering," *Knowl.-Based Syst.*, vol. 56, pp. 156–166, Jan. 2014.
[36] J. Bobadilla, F. Ortega, and A. Hernando, "A collaborative filtering similarity measure based on singularities," *Inf. Process. Manage.*, vol. 48, no. 2, pp. 204–217, 2012.
[37] J. Bobadilla, A. Hernando, F. Ortega, and A. Gutiérrez, "Collaborative filtering based on significances," *Inf. Sci.*, vol. 185, no. 1, pp. 1–17, 2012.
[38] Z. Tan and L. He, "An efficient similarity measure for user-based collaborative filtering recommender systems inspired by the physical resonance principle," *IEEE Access*, vol. 5, pp. 27211–27228, 2017.
[39] M. Zheng, F. Min, H. R. Zhang, and W. B. Chen, "Fast recommendations with the m-distance," *IEEE Access*, vol. 4, pp. 1464–1468, 2016.
[40] J. Serrano-Guerrero, E. Herrera-Viedma, J. A. Olivas, A. Cerezo, and F. P. Romero, "A Google wave-based fuzzy recommender system to disseminate information in University digital libraries 2.0," *Inf. Sci.*, vol. 181, no. 9, pp. 1503–1516, May 2011.
[41] L. H. Son and N. T. Thong, "Intuitionistic fuzzy recommender systems: An effective tool for medical diagnosis," *Knowl.-Based Syst.*, vol. 74, pp. 133–150, Jan. 2015.
[42] P. Vashisth, P. Khurana, and P. Bedi, "A fuzzy hybrid recommender system," *J. Intell. Fuzzy Syst.*, vol. 32, no. 6, pp. 3945–3960, 2017.
[43] C. Liu, T. Jin, S. C. H. Hoi, P. Zhao, and J. Sun, "Collaborative topic regression for online recommender systems: An online and bayesian approach," *Mach. Learn.*, vol. 106, pp. 651–670, May 2017.
[44] C. Musto, P. Lops, M. de Gemmis, and G. Semeraro, "Semantics-aware recommender systems exploiting linked open data and graph-based features," *Knowl.-Based Syst.*, vol. 136, pp. 1–14, 2017.
[45] M. Aleksandrova, A. Brun, A. Boyer, and O. Chertov, "Identifying representative users in matrix factorization-based recommender systems: Application to solving the content-less new item cold-start problem," *J. Intell. Inf. Syst.*, vol. 48, pp. 365–397, Apr. 2017.
[46] Y. Li, D. Wang, H. He, L. Jiao, and Y. Xue, "Mining intrinsic information by matrix factorization-based approaches for collaborative filtering in recommender systems," *Neurocomputing*, vol. 249, pp. 48–63, Aug. 2017.
[47] C. Luo and X. Cai, "Bayesian wishart matrix factorization," *Data Mining Knowl. Discovery*, vol. 30, pp. 1166–1191, Sep. 2016.
[48] S. Oramas, V. C. Ostuni, T. D. Noia, X. Serra, and E. D. Sciascio, "Sound and music recommendation with knowledge graphs," *ACM Trans. Intell. Syst. Technol.*, vol. 8, pp. 21:1–21:21, Oct. 2016.
[49] T. K. Paradarami, N. D. Bastian, and J. L. Wightman, "A hybrid recommender system using artificial neural networks," *Expert Syst. Appl.*, vol. 83, pp. 300–313, Oct. 2017.
[50] Y. Zuo, J. Zeng, M. Gong, and L. Jiao, "Tag-aware recommender systems based on deep neural networks," *Neurocomputing*, vol. 204, pp. 51–60, Sep. 2016.
[51] M. Salehi, I. N. Kamalabadi, and M. B. Ghaznavi-Ghoushchi, "Attribute-based collaborative filtering using genetic algorithm and weighted c-means algorithm," *Int. J. Bus. Inf. Syst.*, vol. 13, no. 3, pp. 265–283, 2013.
[52] W. Yuan and D. Guan, "Optimized trust-aware recommender system using genetic algorithm," *Neural Netw. World*, vol. 27, no. 1, pp. 77–94, 2017.
[53] F. S. Gohari, H. Haghighi, and F. S. Aliee, "A semantic-enhanced trust based recommender system using ant colony optimization," *Appl. Intell.*, vol. 46, pp. 328–364, Mar. 2017.
[54] X. Wang, F. Luo, C. Sang, J. Zeng, and S. Hirokawa, "Personalized movie recommendation system based on support vector machine and improved particle swarm optimization," *IEICE Trans. Inf. Syst.*, vol. 100, no. 2, pp. 285–293, 2017.
[55] E. Q. da Silva, C. G. Camilo, Jr., L. M. L. Pascoal, and T. C. Rosa, "An evolutionary approach for combining results of recommender systems techniques based on collaborative filtering," *Expert Syst. Appl.*, vol. 53, pp. 204–218, Jul. 2016.
[56] J. Bobadilla, F. Ortega, A. Hernando, and J. Alcalá, "Improving collaborative filtering recommender system results and performance using genetic algorithms," *Knowl.-Based Syst.*, vol. 24, no. 8, pp. 1310–1316, 2011.
[57] F. Ortega, B. Zhu, J. Bobadilla, and A. Hernando, "CF4J: Collaborative filtering for Java," *Knowl.-Based Syst.*, vol. 152, pp. 94–99, Jul. 2018.
[58] F. M. Harper and J. A. Konstan, "The movielens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, pp. 19:1–19:19, Dec. 2015.
[59] J. Golbeck and J. Hendler, "FilmTrust: Movie recommendations using trust in Web-based social networks," in *Proc. IEEE Consum. Commun. Netw. Conf.*, vol. 96, Jan. 2006, pp. 282–286.
[60] J. Bennett and S. Lanning, "The netflix prize," in *Proc. KDD Cup Workshop*, New York, NY, USA, 2007, p. 35.
[61] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: A constant time collaborative filtering algorithm," *Inf. Retr.*, vol. 4, no. 2, pp. 133–151, 2001.
[62] R. Salakhutdinov and A. Mnih, "Bayesian probabilistic matrix factorization using Markov chain monte carlo," in *Proc. 25th Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2008, pp. 880–887.

BO ZHU received the bachelor's degree from the Beijing Institute of Technology, China, in 2011, and the master's degree from the Universidad Politécnica de Madrid in 2012. He is currently pursuing the Ph.D. degree in machine learning with E.T.S.I.S.I., Universidad Politécnica de Madrid, Spain. He visited the Beijing Institute of Technology, China, in 2017. His current research interests include unsupervised learning, recommender systems, and information retrieval.

REMIGIO HURTADO was born in 1989. He received the B.S. degree in systems engineering from Universidad Politécnica Salesiana, Ecuador, in 2012, the master's degree in information and software technology with the Instituto Tecnológico y de Estudios Superiores de Monterrey, Mexico, in 2014, and master's degree in computer science and technology from the Universidad Politécnica de Madrid, Spain, 2017.

He is currently a Lecturer with the Universidad Politécnica Salesiana, Ecuador. His research interests include the recommender systems and natural language processing. He is also a member of the Research Team in Artificial Intelligence and Assistive Technology, and his team is collaborating with the Universidad Politécnica de Madrid.

JESÚS BOBADILLA received the B.S. degree in computer science from the Universidad Politécnica de Madrid and the Ph.D. degree in computer science from Universidad Carlos III. He is currently a Lecturer with the Department of Applied Intelligent Systems, Universidad Politécnica de Madrid. He is also in charge of the FilmAffinity.com Research Team, where he is involved in the collaborative filtering kernel of the web site. He has been a Researcher with the International Computer Science Institute, Berkeley University and Sheffield University. Head of the research group. He is a Habitual Author of programming languages books with McGraw-Hill, Ra-Ma, and Alfa Omega publishers. His research interests include information retrieval, recommender systems, and speech processing.

FERNANDO ORTEGA was born in 1988. He received the B.S. degree in software engineering and the Ph.D. degree in computer sciences from the Universidad Politécnica de Madrid, Madrid, in 2010 and 2015, respectively. He is currently an Assistant Professor with the Department of Engineering, U-tad: Centro Universitario de TecnologÍa Y Arte Digital. He has published several papers in the most relevant international journals. He also actively collaborates in various projects with technology companies. His main research fields are machine learning, data analysis, and artificial intelligence.