



A graph-based approach for minimising the knowledge requirement of explainable recommender systems

Marta Caro-Martínez¹ · Guillermo Jiménez-Díaz¹ · Juan A. Recio-García¹

Received: 17 May 2022 / Revised: 21 March 2023 / Accepted: 9 May 2023 /
Published online: 25 May 2023
© The Author(s) 2023

Abstract

Traditionally, recommender systems use collaborative filtering or content-based approaches based on ratings and item descriptions. However, this information is unavailable in many domains and applications, and recommender systems can only tackle the problem using information about interactions or implicit knowledge. Within this scenario, this work proposes a novel approach based on link prediction techniques over graph structures that exclusively considers interactions between users and items to provide recommendations. We present and evaluate two alternative recommendation methods: one item-based and one user-based that apply the *edge weight*, *common neighbours*, *Jaccard neighbours*, *Adar/Adamic*, and *Preferential Attachment* link prediction techniques. This approach has two significant advantages, which are the novelty of our proposal. First, it is suitable for minimal knowledge scenarios where explicit data such as ratings or preferences are not available. However, as our evaluation demonstrates, this approach outperforms state-of-the-art techniques using a similar level of interaction knowledge. Second, our approach has another relevant feature regarding one of the most significant concerns in current artificial intelligence research: the recommendation methods presented in this paper are easily interpretable for the users, improving their trust in the recommendations.

Keywords Explainable recommender systems · Interaction graphs · Link prediction techniques · Interpretability

✉ Marta Caro-Martínez
martcaro@ucm.es

Guillermo Jiménez-Díaz
gjimenez@ucm.es

Juan A. Recio-García
jareciog@fdi.ucm.es

¹ Department of Software Engineering and Artificial Intelligence, Universidad Complutense de Madrid, Madrid, Spain

1 Introduction

It is becoming common to employ recommendation technologies to aid users in finding interesting items on the Web. Due to the overwhelming amount of information, there is a wide range of products such as books, music, games, and trips that are difficult to discover on the Web. Recommender systems [1] enable users to find items according to their preferences and needs.

Classic techniques to implement recommender systems use information about explicit data, such as ratings (collaborative filtering [14]) and preferences (content-based [31]). However, in some cases, we cannot use that information because it is unavailable: we are in a *minimal knowledge scenario*. Then, we need to implement recommender systems based on minimal knowledge, such as the implicit information obtained from the user's interaction with the system. We propose using the implicit information in the interactions between users and items, representing them in a knowledge graph, which is analysed to find recommendations. This recommendation approach presents advantages compared to traditional implementations since it only needs interaction knowledge without other explicit data, such as preferences and ratings. In a recent survey by Ji et al [22], the authors focussed on the wide use of knowledge representations and reasoning based on graphs to solve complex tasks. Furthermore, the authors classified recommender systems as a knowledge-aware application where integrating knowledge graphs can enhance the reasoning behind the recommendation and, therefore, its interpretability. In the context of a recommender system, interpretability can help users comprehend the output of the system and encourage goals such as trust, confidence in the decision-making, or utility of the system [51]. A recent survey by Guo et al [19] claimed that knowledge graphs are a valuable technique to make recommendations with the advantage of being more understandable than classic techniques. This way, knowledge graphs are interpretable by the target users, increasing their trust in the recommendation.

In this work, we propose a novel approach that can be included in this group of interpretable recommender systems based on knowledge graphs modelling users' behaviour. To tackle the problem of providing and explaining recommendations, we infer and model the knowledge within the interactions between users and items as graphs that are later analysed using link prediction techniques, which are Social Network Analysis technologies [36, 47]. Concretely, we propose two different approaches—a user-based and item-based interaction graphs—where we apply link prediction techniques to compute similarities and generate recommendations.

However, common recommendation approaches, such as collaborative filtering (matrix factorisation, decision trees, naive Bayes, neural networks), can also be adapted to provide recommendations only using interaction knowledge. Therefore, a comparison regarding performance and interpretability is required to clarify the potential of our proposal. Primarily, our approach can be considered a local method that focuses on the neighbourhood of the target user in the interaction graph. In contrast, standard recommender systems create global models with all the available knowledge in the system. It impacts the interpretability of the models, as local approaches are more understandable than global ones [5, 44, 50]. Explainability and interpretability are concepts that do not mean the same. Interpretability is the property of an artificial intelligence model that allows understanding its functionality [3, 15]. Explainability refers to the feature of the artificial intelligence system to show the reasons behind the decision of the system, which allows users to understand that system and trust in it [3, 15]. Therefore, when an intelligent system is interpretable, users can understand the system itself. However, if the system is explainable, users understand the decisions of

the system and accept its prediction, which affects the users' decision-making [18]. The graph-based method we propose in this paper is interpretable: users can understand why a recommendation has been made.

The main goal of this work is to demonstrate that the performance of our proposal is comparable to these recommendation techniques when there is only minimal knowledge available but has the advantage of interpretability. We provide an experimental evaluation to compare our proposal with state-of-the-art recommendation techniques in a large and well-known data set such as MovieLens, which contains user ratings about movies.

In Sect. 2, we comment on some related work to our proposal, and afterwards, in Sect. 3, we describe our methods: both recommender approaches are based on interaction graphs. Then, we explain the experiments carried out with our approaches and classic techniques and how we compare them using MovieLens in Sect. 4. Moreover, in Sect. 4.6, we discuss the results to conclude which are the best methods. Finally, in Sect. 5, we make some conclusions about this work.

2 Related work

These days, recommender systems are necessary due to the incredible offer of products available on some platforms, such as Amazon, eBay, or Netflix [1]. Some systems show non-personalised information; for example, the best sellers in a category are shown to the user as a recommendation. However, the best recommendations are the personalised ones that are adapted to each user [45]. That is why the research in this field has a remarkable presence in artificial intelligence investigation [7, 24]. Traditionally, several techniques are used to make recommendations. According to the literature, we can include these techniques into three main groups: collaborative filtering (based on ratings), content-based (based on descriptions of users and items) and knowledge-based approaches (based on descriptions, domain knowledge and restrictions) [1, 7, 32]. There are also hybrid recommender systems that combine different methods to get better recommendations, mitigating the weaknesses of individual techniques. Moreover, there are two types of collaborative filtering methods. The first one is memory-based and can be divided into two main groups: based on users or based on items. Memory-based collaborative filtering is based on similar users to the target user [1, 14, 20, 37, 48, 49]. The other type of collaborative filtering is the model-based one. These methods are more complex than previous ones but, at the same time, achieve excellent results. These methods encompass machine learning and data mining techniques as predictive models, like decision trees, naive Bayes, neural networks or support vector machines. One of the most popular implementations within this group is matrix factorisation, which is widely used due to its proven effectiveness [8, 25, 26].

In the literature, we can also determine that some of these techniques are adapted for recommending new items when there is a minimal knowledge scenario. Mainly, these techniques are model-based collaborative filtering methods, although memory-based collaborative filtering can also be used to solve this problem. We can find reviews and some examples of this application in the literature [2, 39, 52, 53].

These collaborative filtering techniques adapted to minimal knowledge scenarios use the users' interactions with the items of the system. They do not consider additional knowledge about the interaction itself (explicit preferences), such as the value of the ratings, for example, or about the content, such as user preferences and item descriptions, because this knowledge does not exist or is unavailable. Therefore, they use explicit knowledge instead of implicit

information. They are helpful when users can only express their like or dislike for an item. According to this, we can consider two types of interaction definitions: binary interactions or unary interactions [1, 45]. In binary interactions, users can express a liking for one item as a positive interaction or dislike for an item as a negative interaction. If the user does not express the like, the interaction does not exist. For example, we can consider a like for an item if the user rates this item with 4 or 5 stars. A dislike would be a value of 1, 2 or 3. However, the users only can like one item in the case of unary ratings. It exists only for positive interactions. One example of unary ratings is the purchase of one item. So, the user has interacted with this item. If she does not buy an item, then the interaction does not exist; it is not a negative interaction. Minimal knowledge scenarios are the ones where we have unary ratings. The book by Aggarwal [1] and the work by Lü et al [35] are where we can find the most comprehensive compilation of collaborative filtering solutions applied on binary or unary ratings.

However, we must clarify the applicability of different approaches when we have unary ratings. Collaborative filtering and other alternatives found in the literature to get recommendations are not applicable in minimal knowledge scenarios (when we have only unary ratings) in a straightforward way. To apply those techniques in this type of knowledge, as Aggarwal [1] explains, we need to adapt the unary ratings and transform them into binary or ternary ratings, as we have done in our evaluation. We can see that this transformation is always done somehow in every technique found in the state-of-the-art when applied to minimal knowledge scenarios. Next, we discussed three examples.

In references [40] and [21], interactions carried out are represented with 1, while no interactions are represented with 0. In reference [40], Sect. 3.2, we can see that authors mention that their approach is based on weighted low-rank approximation, which “is applied to a collaborative filtering problem with a naive weighting scheme assigning 1 to observed examples and 0 to missing (unobserved)”. A few lines below show that the approach starts working from a matrix R , whose values are 0 and 1 (binary ratings obtained from the unary rating transformation). In reference [21], Sect. 4, the authors say that the knowledge representation they used is as follows: “if a user u consumed item i , then we have an indication that u likes i (representation = 1). On the other hand, if u never consumed i , we believe no preference (representation = 0)”.

Therefore, both references use binary ratings: “like” and not observed. They assume that the items users have interacted with are the ones they like. However, they represent the items that users have not interacted with a unique value that could represent two types of items: items not interesting for users (so that is why users did not interact with them) or items interesting for users (but not found by users yet). So, this type of representation could introduce bias considering the inaccurate knowledge.

In reference [43], they transform the representation in ternary ratings. They explain how they do that in Sect. 3.2. They have three values: ? for non-observed items, + for observed items that the user likes, and – for observed items that users like less than + ones. In this paper, we propose a method that has the advantage of using unary ratings straightforwardly over the rest of the techniques. We do not make any transformation; we only use factual information, so we avoid introducing bias in our knowledge.

Moreover, some recommender approaches are based on graphs with or without the link prediction techniques application in the literature. Liben-Nowell and Kleinberg [29] described and compared different similarity metrics from link prediction techniques. Another interesting work is the publication [11], which depicts an approach that considers patterns in a graph to establish the proper order of recommendations. The paper by [4] proposes a recommendation approach using a knowledge representation similar to ours. To make recommendations,

they combine the knowledge about: (1) the most popular items (the ones that participate in more amount of interactions), and (2) the interactions carried out by users whose interactions are similar to the ones carried out by the target user. The main goal of this approach is to solve the problem of cold start and sparsity in traditional recommender systems. In the work [12], we find an example of link prediction techniques used in bipartite graphs to solve a recommendation problem. In the paper [57], the authors represented interactions and similar relations in a graph with complex numbers. They used this structure with link prediction techniques to make recommendations. In the case of the work [58], the authors propose using link prediction techniques on graphs to alleviate typical recommender systems problems like sparsity and scalability. The publication by Li and Chen [28] introduces a methodology based on graphs and link prediction techniques which uses random walks for getting recommendations. Using random walks over graphs is a recommendation algorithm we can find in other papers in the literature. For example, [6] introduces this methodology to get context-aware location recommendations. In the work [59], the authors propose using graphs to represent different relationships in the graph and link prediction techniques to predict new relationships, which could be applied to recommender systems. They use a Bayesian personalised ranking-based optimisation technique to get these relationships. We can observe in the work [61] that link prediction can be applied on bipartite graphs to get movie recommendations representing the genre in the links. The graph representation in the work by [60] is more complex because it uses graph features to represent latent features in a multi-step relation path. This representation aims to discover new knowledge that can be used for recommender systems. This graph representation is evaluated using link prediction techniques to predict this new knowledge. As the last example, in the work [17], authors propose to use graphs and link prediction techniques to recommend new research collaborations. The graph represents a wide variety of knowledge about a scientist and her work.

The explainability of recommender systems is also a significant problem to be addressed, as many of these systems act as a black box for the users. When users do not understand the reasons behind a recommendation, they do not trust the system, so it is not useful for them [16]. Some works explore graphs to make explainable recommendations, for example, the work by Wang et al [55]. The authors describe a new model Knowledge-aware Path Recurrent Network, which uses a knowledge graph to make recommendations. The graph represents not only interactions between users and items but also their features. This proposal improves the performance of other models like collaborative knowledge base embedding or neural factorisation machines. Moreover, Knowledge-aware Path Recurrent Network stands out because it is an interpretable model. However, it uses the whole model to get the paths that get the recommendations and explanations, which means it is a global model. According to the literature, this could make understanding the recommendations more difficult. We also can encounter an example of an explainable recommender system that uses graphs in the work by Xian et al [56]. This proposal, called Policy-Guided Path Reasoning, considers the paths in the graph to generate interpretable recommendations. In our previous work [9], we proposed a graph-based recommender method that generates recommendations using an interaction graph as the knowledge source. Furthermore, we also used this knowledge source based on graphs to explain black-box recommender systems [10]. As we mentioned, there are surveys [19, 22] that study the importance of using graphs to tackle the recommendation problem. As they state, graphs are useful for making recommendations, increasing the interpretability of the recommender system and the users' trust.

In this work, we propose a methodology to make recommendations when there is not enough knowledge to apply straightforward classic recommender techniques or other proposals discussed in this section or found in the literature. Therefore, this new proposal can

be advantageous when the lack of information is so high. Furthermore, the proposal is more interpretable than classic techniques or other proposals from the literature. It can be more effective than them because it can offer interpretability and explainability by itself, providing satisfaction and trust to target users. These two key advantages are the main differences from the works discussed in this section. In the next section, we detail how our proposed approach works.

3 Recommendation with interaction graphs

Our approach is based on the basic premise that, at least, users interact with items in any recommender system. We can represent these interactions as a tuple $R = (t, u, i, x)$, where:

- t is the timestamp when the interaction was carried out,
- $u \in U$ is the user that carried out the interaction,
- $i \in I$ is the item with which u has interacted on t ,
- x is the value associated with the interaction. In many cases, x is the rating provided by u to i in a specific interaction. For example, when a user u rates a book i in Goodreads, x is the rating provided. Nevertheless, we can have other information in x besides ratings. For instance, in online judges, x is the verdict provided by the platform to u for a solution of the problem i .

Most current techniques exploit the value of x associated with the interaction to provide a recommendation. However, in several scenarios, acquiring these values may be difficult or even infeasible. Therefore, we propose a recommendation approach based on interaction graphs and link prediction methods suitable for recommending items when the values associated with the interaction are not available or helpful. It is important to note that these interaction graphs do not require further information about users, item features, or ratings. As we will prove in the evaluation section, this approach is much less information-demanding than standard recommendation methods but can achieve similar performance. Furthermore, it is more interpretable than these standard techniques because its underlying rationale is more straightforward for users.

Then, when x values are unavailable, we can only access the information about whether u has interacted with i . Through this abstraction, we can build a graph representing the interactions as a non-weighted bipartite graph: $G(N, L)$. The nodes in N belong either to the user set U or to the item set I . The links L are created when a node representing the user u has interacted with a node representing an item i . We can also represent G as an adjacency matrix \mathbb{A} , where $\mathbb{A}[i, u]$ is equal to 1 if user u has interacted with the item i , \emptyset otherwise. Therefore, our graph-based model describes the unary interactions in the system. We do not minimise the graph: it is inherently minimal because it represents minimal knowledge. The main idea of our recommendation methods is to use this graph and link prediction techniques to get similarities between nodes. With these similarities, we can find the most interesting items for the target user.

To enable the application of link prediction techniques, we can perform a bipartite network projection and obtain two different weighted graphs: an *item–item* graph G_I and a *user–user* graph G_U . In Fig. 1, we can see an example of a *bipartite network projection*. This figure is also a diagram to illustrate the minimal scenario where our approach is suitable to be applied. With the adjacency matrix, we could see an example of the interactions carried out for a particular situation. That matrix can be transformed on an interaction graph as we can see in the figure. For example, on YouTube, those interactions could represent the videos

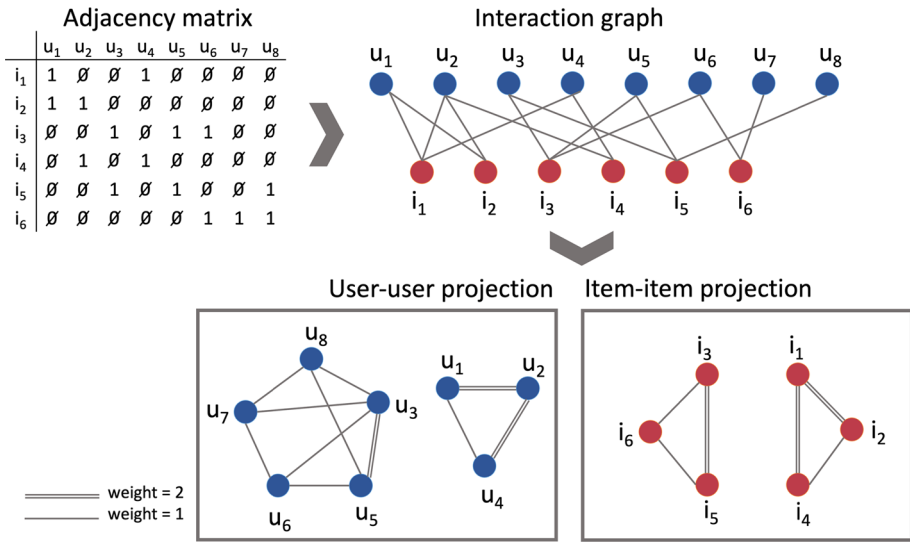


Fig. 1 Transformation of a bipartite graph into a non-bipartite graph through *bipartite network projection* [62]. We obtained two different graphs—user-based (left) and item-based (right)—from the original graph

watched by the users. This is an example of a minimal scenario because users do not rate the videos that they watch. Moreover, the recommender system does not know what is the video content or explicit user preferences. Another example could be online judges. Online judges are systems where users can solve programming problems getting a verdict from the system according to their solution. In this scenario, users do not rate the problems that they solve and we do not have useful information about the problems to make recommendations. These are motivating examples that illustrate the recommendation task that we want to figure out in this work.

Because users may generate many interactions with items, the resulting graph may be very dense. For this reason, we must also apply filtering techniques to reduce the density and reduce the computing cost without losing sensible data. This filtering is performed by removing the links whose weight is lower than a threshold value denoted as θ .

From the two graphs resulting from the *bipartite network projection*, we can make recommendations in two different ways as described next.

3.1 Item-based recommendation process

The item-based recommendation approach uses the graph where nodes represent items. This process recommends a target user u the most similar items to those with which u has interacted previously. We define graph $G_I = \langle N_I, L_I \rangle$, where $N_I = \{n_i, n_j, \dots\}$ is the set of nodes representing the items I of the recommender system and $L_I = \{(n_i, n_j)\}$ are the links between items according to the bipartite projection. In this approach, two nodes are linked if they have at least one user in common that has interacted with both items. The weight of the link, w_{ij} , represents the number of users who have interacted with them. In addition, as we mentioned, we remove the links whose weight is lower than a threshold value of θ to reduce the density of the graph.

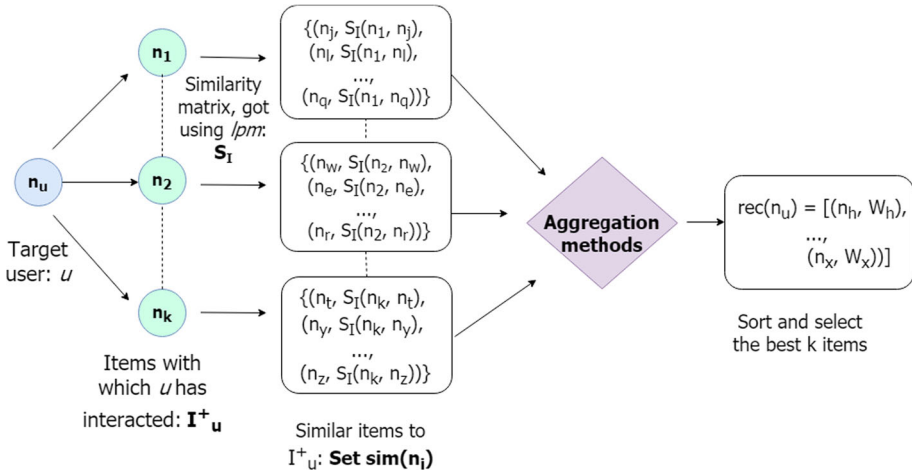


Fig. 2 Overview of the recommendation process using the item-based graph

The process to make recommendations for a target user u using the item-based graph is (also illustrated in Fig. 2):

1. *Step 1.* Build a similarity matrix S_I that stores all the similarity values between all of the pairs of nodes in the graph G_I using one of the link prediction methods, $LinkPred()$, described later in Sect. 3.3. This matrix represents the similarities between the items within the recommender system.
2. *Step 2.* Obtain the set $I_u^+ \subset I$ with the items that user u has interacted with. Complementary, $I_u^- = I \setminus I_u^+$ is the set of items that user u has not yet interacted with.
3. *Step 3.* Considering function $item(n_i)$ returns the item represented by node n_i , for every node n_i that $item(n_i) \in I_u^+$, obtain a list $sim_u(n_i)$ containing the most similar items to the item represented by n_i , and u has not interacted yet, ordered according to their similarity in S_I . This list is defined as:

$$\begin{aligned}
 sim_u(n_i) &= \{(i_j, s_{ij}), (i_k, s_{ik}), \dots | s_{ij} \geq s_{ik}\} \\
 \text{where, } \forall (i_x, s_{i_x}) \in sim_u(n_i) : i_x &= item(n_x) \\
 i_x &\in I_u^- \\
 s_{i_x} &= S_I[n_i, n_x]
 \end{aligned}
 \tag{1}$$

4. *Step 4.* Join the $sim_u(n_i)$ lists for every n_i that $item(n_i) \in I_u^+$ and obtain a global list $gsim_u$ containing all the similar to I_u^+ items (and associated similarity), which u has not interacted with. This list may contain repeated entries with different similarity values for any item:

$$gsim_u = \bigcup_{item(n_i) \in I_u^+} sim_u(n_i)
 \tag{2}$$

5. *Step 5.* Aggregate the similarity weights s_{i_x} for those items that are repeated in list $gsim_u$. We have experimented with different aggregation methods, $Aggr()$, described in Sect. 3.4. As a result, we obtain the following final ordered list rec_u :
6. *Step 6.* Finally, from $rec(u)$ recommend the first k elements.

Algorithm 1 Optimised algorithm for the computation of the recommendation list using the item-based approach.

```

Input:  $G_I = \{I, L_I\}, I_u^+, LinkPred(), Aggr()$ 
Output:  $rec$ 
1 foreach  $n_i \in I$  do
2   foreach  $n_j \neq n_i \in I$  do
3      $S_I[n_i, n_j] = LinkPred(G_I, n_i, n_j)$ 
4   end
5 end
6  $I_u^- = I \setminus I_u^+$ 
    $rec = \{\}, gsim = \{\}$ 
   foreach  $item(n_x) \in I_u^+$  do
7     foreach  $item(n_i) \in I_u^-$  do
8        $s_x = S_I[n_i, n_x]$ 
        $insert(gsim, (item(n_x), s_x))$ 
9     end
10  end
11 foreach  $(i_x, ?) \in gsim$  do
12    $sims = \{\}$ 
   foreach  $(i_x, s) \in gsim$  do
13      $insert(sims, s)$ 
14   end
15    $S_x = Aggr(sims)$ 
    $orderedInsert(rec, (i_x, S_x))$ 
16 end
17 return  $rec$ 

```

Algorithm 1 shows the optimised algorithm used to compute the item-based recommendation list according to the previous steps. Next, we describe the alternative user-based approach.

3.2 User-based recommendation process

In this case, the user-based recommendation approach is based on the graph with users as nodes. This approach recommends to u those items which the most similar users have interacted with, but u has not interacted with yet. We define our graph as $G_u = \langle N_U, L_U \rangle$, where $N_U = \{n_u, n_v, \dots\}$ are the nodes representing the users U of the system and $L_U = \{(n_u, n_v)\}$ are the links between users according to the bipartite projection. In this case, two nodes are linked if they have at least one item with which both users have interacted. We compute the weight of a link, w_{uv} , as the number of common items with which both users have interacted. The link does not exist if the pair of users has not interacted with at least one common item. Furthermore, as we did in the previous approach, we filter the links whose weight is lower than a threshold value of θ .

We follow the steps to make recommendations to a target user u with a user-based process analogously to the item-based approach. However, in this case, user u is directly represented in the graph through its corresponding node n_u , and function $user(n_u)$ returns the user represented by the graph node. In Fig. 3, we show the recommendation process.

1. *Step 1.* Build a similarity matrix S_U that stores all the similarity values between every pair of nodes in the graph G_U using a link prediction method $LinkPred$. This matrix contains the similarity between users of the recommender system.

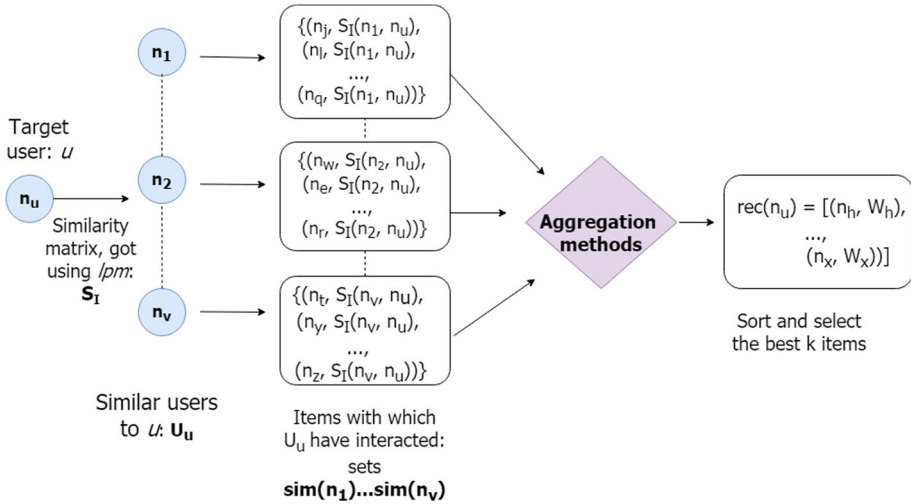


Fig. 3 Overview of the recommendation process using the user-based graph

Algorithm 2 Optimised algorithm for the computation of the recommendation list using the user-based approach

Input: $G_U = (U, L_U), n_u, I_u^-, I_v^+ \forall v \neq u, LinkPred(), Aggr()$

Output: rec

```

18  $gsim = \{\}$ 
19 foreach  $n_v \neq n_u \in U$  do
     $s_v = S_U[n_u, n_v] = LinkPred(G_U, n_u, n_v)$ 
    foreach  $i_x \in I_u^- \cap I_v^+$  do
20      $insert(gsim, (i_x, s_v))$ 
21   end
22 end
23 foreach  $(i_x, ?) \in gsim$  do
24    $sims = \{\}$ 
    foreach  $(i_x, s) \in gsim$  do
25      $insert(sims, s)$ 
26   end
27    $S_x = Aggr(sims)$ 
     $orderedInsert(rec, (i_x, S_x))$ 
28 end
29 return  $rec$ 

```

- Step 2. Obtain the set $U_u \subset U$ containing the users whose representing nodes n_v are similar to n_u according to S_U . Also, compute the I_u^- set and all the I_v^+ sets for every $v \in U_u$.
- Step 3. For every node n_v that $v \in U_u$ obtain the list $sim_u(n_v)$ with the items that user n_v has interacted but user n_u has not interacted yet. In this case, all of the items have the same similarity value, s_v , corresponding to the similarity between both users $S_U[n_u, n_v]$.

This list is defined as:

$$\begin{aligned}
 sim_u(n_v) &= \{(i_j, s_v), (i_k, s_v), \dots\} \\
 \text{where, } \forall (i_x, s_v) \in sim_u(n_v) : i_x &\in I_v^+ \cap I_u^- \\
 s_v &= \mathbb{S}_U[n_u, n_v]
 \end{aligned}
 \tag{3}$$

4. *Step 4.* Join the $sim_u(n_v)$ lists for every $v \in U_u$ and obtain a global list $gsim_u$ containing all the items (and associated similarity) with which u has not interacted. This list may contain repeated entries with different similarity values for any item i_j :

$$gsim_u = \bigcup_{user(n_v) \in U_u} sim_u(n_v)
 \tag{4}$$

5. *Step 5.* Next, aggregate the similarity values for those items that are repeated in the $gsim_u$ list using an aggregation function $Aggr()$. As a result, we obtain the following final ordered list rec_u :
6. *Step 6.* Finally, recommend the first k items in $rec(n_u)$.

Algorithm 2 presents the algorithm corresponding to the user-based recommendation process.

Both graph-based recommendation approaches define two hook functions which enrich our proposal with multiple alternatives. These functions represent the link prediction methods, $LinkPred()$, employed in the first step of the algorithms, and the aggregation strategies, $Aggr()$, to join the item lists. The following subsections will detail the alternatives that can be applied for both functions.

3.3 Link prediction methods

Link prediction techniques are a set of methods from the Social Network Analysis field that allow finding new links that will appear or have disappeared in a graph [34, 36, 47, 54]. Being G_t a graph at time t , link prediction techniques focus on predicting the evolution of links at G_{t+1} .

As described before, our method requires a similarity matrix S that stores the similarity values between every pair of nodes in the graph. However, we can predict undefined scores corresponding to non-connected pairs of nodes using link prediction techniques. This way, $S[n_a, n_b] = LinkPred(G, n_a, n_b)$ is the similarity value between the node n_a and the node n_b computed by the link prediction method $LinkPred()$ applied to graph G . The higher $LinkPred(G, n_a, n_b)$ is, the more willing to recommend n_b to a target user who has not interacted with n_b yet if the user has also interacted with n_a , and vice versa. Consequently, the pairs of nodes with the highest $LinkPred(G, n_a, n_b)$ scoring will be linked (and recommended). The link prediction methods to compute matrix S can be classified as:

- *Node-based methods.* The similarity between a pair of nodes is computed using the features or attributes of the nodes.
- *Neighbour-based methods.* The similarity between a pair of nodes n_a and n_b is computed using the data about the neighbourhoods of n_a and n_b .
- *Path-based methods.* They exploit the available information about the neighbours of each pair of nodes and the possible paths between both nodes.
- *Random walk-based methods.* They use transition probabilities from a node to its neighbours and non-connected nodes to simulate social interactions.

In previous research about graph-based recommendation techniques [9, 23], we proposed a variation of the classic link prediction methods from the literature [29, 30, 33, 36, 47, 54]. We defined these methods by taking into account their interpretability. Concretely, we require link prediction methods that turn our methods into a *local model* to provide interpretable recommendations. Local models in Explainable Artificial Intelligence are based on only using a part of the model knowledge or a subset of the data set to provide explanations instead of the more complex global model, which uses knowledge from the whole data set or the whole model. This subset is obtained from the elements of the neighbourhood to be explained [5, 44, 50]. Local models fit perfectly with our graph-based recommendation approach because we are not interested in recommendations far from the target nodes we are considering. In the item-based graph, we cannot recommend items that are too far from the items with which the target user has interacted. Similarly, we must discard users who are not in the local neighbourhood of the target user for the user-based approach. This way, although path- and random walks-based methods can also achieve good performances, they are not considered in this work because they must be applied to the whole graph, belonging, therefore, to the global model category regarding their explainability. Hence, the proper methods to provide interpretable recommendations are the node- and neighbourhood-based ones because they only require a local subgraph around the target node.

To define these methods, we need some preliminary notation: $\mathcal{N}(n)$ represents the neighbours of node n . $|\mathcal{N}(n)|$ represents the number of neighbours (or *node degree*) of node n . w_{ab} represents the weight of the link between nodes n_a and n_b (0 if no link between both nodes). $\mathcal{W}(n_a) = \sum w_{ax} : n_x \neq n_a \in N$ represents the *weighted node degree* of node n_a , which is the sum of the weights in the links directly connected to n_a .

Next, we detail the link prediction metrics that we consider to estimate the possibility of a connection between two nodes. We have selected these metrics due to their performance in our previous findings [9, 23] and because they belong to the local model category regarding their explicability. Most of them have two versions—a weighted and an unweighted version—and do not need to be normalised.

Edge weight (EW). This metric estimates the similarity between two nodes as the weight of the link.

$$EW(G, n_a, n_b) = w_{ab} \tag{5}$$

An unweighted version of this metric is also defined as $EW(G, n_a, n_b) = 1$ if $w_{ab} \neq 0$, but we discarded it because of its evident simplicity.

Common Neighbours (CN). The similarity between the two nodes is the number of neighbours they have in common. The rationale behind this metric is that the greater the intersection of the neighbour sets of any two nodes, the greater the chance of future association between them. *Weighted Common Neighbours (WCN)* is the weighted version of this metric.

$$CN(G, n_a, n_b) = |\mathcal{N}(n_a) \cap \mathcal{N}(n_b)| \tag{6}$$

$$WCN(G, n_a, n_b) = \sum_{n_z \in \{\mathcal{N}(n_a) \cap \mathcal{N}(n_b)\}} w_{az} + w_{zb} \tag{7}$$

Jaccard Neighbours (JN). This metric is an improvement of $CN(n_a, n_b)$ as it measures the number of common neighbours of n_a and n_b compared with the number of total neighbours of both nodes. It does not have a weighted metric version.

$$JN(G, n_a, n_b) = \frac{|\mathcal{N}(n_a) \cap \mathcal{N}(n_b)|}{|\mathcal{N}(n_a) \cup \mathcal{N}(n_b)|} \tag{8}$$

Adar/Adamic (AA). This metric also measures the intersection of neighbour sets of two nodes in the graph but emphasises the smaller overlap. The weighted version of this metric is *Weighted Adar/Adamic (WAA)*.

$$AA(G, n_a, n_b) = \sum_{n_z \in \{\mathcal{N}(n_a) \cap \mathcal{N}(n_b)\}} \frac{1}{\log |\mathcal{N}(n_z)|} \tag{9}$$

$$WAA(G, n_a, n_b) = \sum_{n_z \in \{\mathcal{N}(n_a) \cap \mathcal{N}(n_b)\}} \frac{w_{az} + w_{zb}}{\log(1 + \mathcal{W}(n_z))} \tag{10}$$

Preferential Attachment (PA). It is based on the consideration that there is a higher probability of creating links between nodes that already have many links. The probability of creating a link between nodes n_a and n_b is computed as the product of their link output degree. Thus, the higher the degree of both nodes, the higher the likelihood of linking. This metric has the drawback of leading to high probability values for highly connected nodes to the detriment of the less connected ones in the network. *Weighted Preferential Attachment (WPA)* is its weighted version, where the link weights are considered when computing the degree of nodes n_a and n_b .

$$PA(G, n_a, n_b) = |\mathcal{N}(n_a)| \cdot |\mathcal{N}(n_b)| \tag{11}$$

$$WPA(G, n_a, n_b) = \mathcal{W}(n_a) \cdot \mathcal{W}(n_b) \tag{12}$$

Once we have presented the link prediction methods, the following section introduces the aggregation strategies used by the algorithms described in Sects. 3.1 and 3.2.

3.4 Aggregation methods

The last requirement of our graph-based recommendation approaches is the aggregation strategy to join the list of retrieved items. The goal is to aggregate the similarity of those nodes that are repeated in list $gsim_u = \{(n_j, s_j), (n_k, s_k), \dots\}$. As we described previously, we compute this list either from the item-based approach or the user-based approach (Eqs. 2 and 4). Here, $\{s_j, \dots, s_k\}$ are the similarity values from the S matrix. The $gsim_u$ list results from joining sets $sim_u(n_*)$; therefore, it may contain repeated items. We propose the following aggregation strategies:

Highest Similarity. The simplest aggregation method that we propose is based on the highest similarity. The aggregation value is the highest similarity value s_x that n_x has in all of its appearances in the $gsim_u$ list.

$$\bar{s}_x = \max(s_x) \tag{13}$$

Simple voting. The aggregated similarity value \bar{s}_x is the number of times that the node n_x appears in the list of items $gsim_u$.

$$\bar{s}_x = count(n_x) \tag{14}$$

Weighted voting. The aggregated similarity value \bar{s}_x is the weighted addition of the similarity values of the item n_x in the list, where w_i is every weight of $n_x \in gsim_u$.

$$\bar{s}_x = \sum \frac{s_x}{w_i} \tag{15}$$

Table 1 Example of applying different aggregation methods to a user-based graph

U_u	$sim_u(n_{v*})$	$S_U[n_u, n_v]$
n_{v3}	$\{i_1, i_2, i_5\}$	5
n_{v1}	$\{i_1, i_2, i_4\}$	4
n_{v2}	$\{i_2, i_3, i_5\}$	2
n_{v4}	$\{i_1, i_2, i_3\}$	1

	Simple voting	Weighted voting	Positional voting
\bar{s}_1	$1 + 1 + 1 = 3$	$5/12 + 4/12 + 1/12 = 0.83$	$1/1 + 1/2 + 1/4 = 1.75$
\bar{s}_2	$1 + 1 + 1 + 1 = 4$	$5/12 + 4/12 + 2/12 + 1/12 = 1$	$1/1 + 1/2 + 1/3 + 1/4 = 2.08$
\bar{s}_3	$1 + 1 = 2$	$2/12 + 1/12 = 0.25$	$1/3 + 1/4 = 0.58$
\bar{s}_4	1	$4/12 = 0.33$	$1/2 = 0.5$
\bar{s}_5	$1 + 1 = 2$	$5/12 + 2/12 = 0.58$	$1/1 + 1/3 = 1.33$
$rec(n_u)$	$[i_2, i_1, i_3, i_5, i_4]$	$[i_2, i_1, i_3, i_5, i_4]$	$[i_2, i_1, i_5, i_3, i_4]$

(Top) describes the information obtained from the graph and the similarity methods where U_u : Similar users to n_u ; $sim_u(n_v)$: items that each n_{v*} user has interacted; $S_U[n_u, n_v]$: similarity between n_u and every n_v computed using a link prediction metric. (Bottom) the result of applying the aggregation methods

Positional voting. In this case, the aggregated similarity value \bar{s}_x depends on the position $pos()$ of n_x in the list $gsim_u$ sorted by similarity value in decreasing order:

$$\bar{s}_x = \frac{1}{\sum pos(n_x)} \tag{16}$$

To illustrate how the voting systems work, we provide an example (see Table 1) based on a simple user graph. We have four users n_v similar to our target user n_u and $gsim_u = \{i_1, i_2, i_3, i_4, i_5\}$ contains the item candidates that those users have interacted with, but n_u has not interacted yet with repeated weights. As we can observe, each voting system generates different ranked lists of items. For example, if we choose $k = 2$, the list of items recommended to u is $[i_2, i_1]$ for all voting systems. However, if we choose $k = 3$, we obtain two different lists: $[i_2, i_1, i_3]$ for the simple voting system and $[i_2, i_1, i_5]$ for the weighted and positional voting systems. This way, we illustrate that the choice of the voting system to use in our recommender system has a relevant impact on the result.

Taking into account the similarity metrics and aggregation methods proposed here, we could consider memory-based collaborative filtering adapted to the minimal knowledge scenario as one of the specific configurations of our interaction graph-based methods. Memory-based collaborative filtering behaviour is based on (1) finding the most similar items to those liked by the user or (2) finding the items liked by the most similar users to the target user. However, in the minimal knowledge scenario, we only have unary ratings to generate recommendations, i.e. interactions with no value associated with the rating. Therefore, we could redefine memory-based methods to work in the minimal knowledge scenario if we only consider: (1) the number of users who have interacted with two items (in the item-based model) or; (2) the number of items with which two users have interacted (in the user-based model). Actually, this approach is equivalent to the EW measure defined for our graph-based models. The memory-based collaborative filtering adapted to the minimum knowledge scenario will subsequently recommend the items with the highest similarity. In that case, it is equivalent to the interaction graph-based models configured with the EW similarity metric and the aggregation method based on the maximum similarity.

4 Evaluation

This paper aims to demonstrate that our graph-based recommendation approach using link prediction techniques achieves better performance than state-of-the-art recommenders in a scenario of minimal knowledge availability. Therefore, we compare with those techniques that can be adapted to use only the user-item interaction information.

According to the literature, several recommendation methods are suitable to this scenario when ratings or other associated values to the interaction are not available [1, 35, 39, 52]. Most likely, the most popular recommendation techniques are collaborative filtering approaches based on neighbours (item-based or user-based). We do not consider the matrix factorisation method because it is highly focused on the rating values. We have also selected other representative techniques of model-based collaborative filtering that can be adapted to only work with information about interactions [2, 45]: decision trees, naive Bayes, neural networks, random forest, and support vector machines. To compare these techniques in a minimal knowledge scenario, we reduce the knowledge they use to create the associated prediction model.

This evaluation also considers and discusses the inherent interpretability of these collaborative filtering methods. We demonstrate that the local models used by our graph-based proposals are generally more interpretable than the global models built by these recommendation techniques [19, 22]. The implementation of the graph-based models, their evaluation, and the code generated to analyse the data set are available at GitHub.¹

After presenting the data set (Sect. 4.1) and the experimental set-up (Sect. 4.2), we determine the knowledge requirement of each technique involved in the evaluation (Sect. 4.3) and the evaluation metrics we used (Sect. 4.4). Later, we analyse the results obtained with our graph-based approaches (Sect. 4.5.1) and compare them with the results obtained with classic techniques (Sect. 4.5.2). We also compare their interpretability (Sect. 4.5.3). Finally, we discuss the results obtained in both experiments and the comparison between their performances (Sect. 4.6).

4.1 Data set analysis

The data set chosen for the evaluation is the MovieLens 100K data set² as it is widely used in the literature. It includes 100K ratings in tuple form $R = (t, u, i, x)$, where u is the user, i is the movie that u has watched, x is the rating that u has provided to i and t is the timestamp when u rated i .

However, the original data set is too sparse to get significant results with any recommender method. Consequently, we performed a preliminary analysis to define a sub-sampling approach that increases its density without losing relevant knowledge. We follow the model proposed in the work [13] for data set benchmarking as we did in our previous works [9, 10, 23].

The result of this descriptive analysis is reported in Table 2 and graphically represented in Fig. 4. This figure shows the users–item ratio according to the following stratification: users that have interacted at least with 100% (blue), 75% (red), 50% (green), 25% (purple) and 12.5% (orange) of the most popular items. Taking into account the work [13] as well, the

¹ <https://github.com/martcaro/Graph-Recommendations>

² <https://grouplens.org/datasets/movielens/100k/>

Table 2 Descriptive analysis of the data set used for the evaluation

Metric	D_Original	D_50	D_25	D_12
# Ratings	100,000	15,954	34,030	42,482
# Items	1,682	200	200	200
# Users	943	129	376	611
Density	0.063	0.618	0.453	0.348
<i>Items</i>				
Maximum # ratings per item	583	126	348	499
Median # ratings per item	27	80	162.5	197.5
Average # ratings per item	59.453	79.77	170.15	212.41
Minimum # ratings per item	1	22	54	84
<i>Users</i>				
Maximum # ratings per user	737	173	173	173
Median # ratings per user	65	121	86	63
Average # ratings per user	106.045	123.674	90.505	69.529
Minimum # ratings per user	20	100	50	25
<i>Ratings</i>				
% Ratings ≥ 4	55.375	62.774	64.331	64.672
% Ratings ≤ 4	44.625	37.226	35.669	35.328

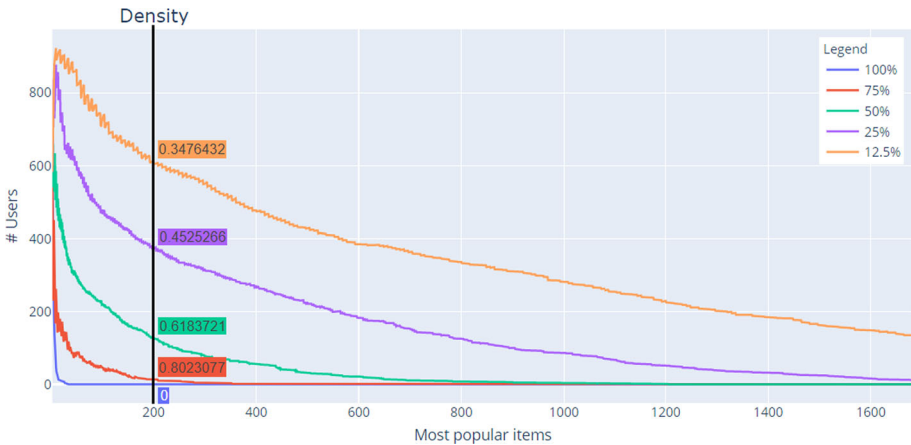


Fig. 4 Study of MovieLens data set to define a sub-sampling approach that increases its density without losing relevant knowledge. Lines show the number of users that interacted with at least the 100% (blue), 75% (red), 50% (green) and 25% (purple) of the most popular movies (Color figure online)

density was calculated as follows:

$$density = \frac{\#interactions}{\#users \cdot \#items} \tag{17}$$

From this study, we can consider 75% a high density but lacking enough users to perform a significant evaluation. However, other percentages let us explore the performance of the

recommendation approaches with different data set densities. To obtain relatively small sub-samples that let us perform the large number of simulations required to compare all the approaches presented in this paper, we decided to limit the data set to the 200 most popular items. This way, we can explore different significant data set densities by selecting the users that interacted at least with the 50%, 25% and 12.5% of the 200 most popular items. Concretely, their densities are 0.61, 0.45, and 0.34, respectively (see Table 2).

4.2 Experimental set-up

To find out whether the recommendations provided by our methods were interesting to the target users, we needed to build a model for each user u_t . First, we needed to split the set to obtain the appropriate training and test sets to measure the accuracy of the recommendation methods. We decided to obtain them in the same way for each of the three data sets obtained from the analysis: through cross-validation, randomly choosing 10% of the items (a total of 20 movies out of the 200 most popular ones) to create the test set and the remaining 90% to build the training set. Therefore, the test set considers all those interactions carried out by the target user u_t with those 20 randomly chosen movies. In turn, the interactions carried out by u_t with these movies are removed from the training set. With the interactions from the training set, we built our graph-based models (considering the process described in Sects. 3.1 and 3.2), or the collaborative filtering models used in the evaluation. In the experimentation process, the recommendation methods will only be able to recommend items from this test set. Target user u_t has not interacted with those items. However, those items appear in interactions carried out by other users in the training set. Therefore, we have information about these items to be able to check if we are going to recommend them to u_t .

In the next series of experiments, we compare the accuracy of graph-based methods with other recommendation techniques, namely with memory-based collaborative filtering algorithms (based on users or items) and model-based collaborative filtering (machine learning techniques). We implement memory-based methods with the Pearson correlation coefficient as a similarity metric since it is one of the most commonly used metrics. However, it exploits rating values to obtain similarities between items, so we wanted to check how it behaves with implicit rating values in minimal knowledge scenarios.

4.3 Minimal knowledge scenarios

As explained above, we must adapt the data set so that these models provide recommendations in a minimal knowledge scenario by transforming the original ratings to represent only the interactions between the item and the user. This process is illustrated in Table 3.

We should note that memory-based recommenders (denoted as CF from now on) cannot work with unknown values, having to use, at least, the ternary representation. They need to represent positive interactions (with ratings of either 4 or 5), negative interactions (ratings with a value lower than 4) and non-interactions. However, collaborative filtering based on models or machine learning models (denoted as ML from now on) can at least work with the binary representation in which positive ratings are encoded as 1, negative ratings are 0, and unknown values are not included in the model. As we discussed in Sect. 2, according to Aggarwal [1], there is an alternative representation that encodes negative and unknown ratings with the same value (usually 0). However, it introduces a critical bias. Finally, the *unary* representations only represent positive ratings with 1. If the user has not interacted with or did not like a product, the corresponding value is not specified.

Table 3 Incremental minimisation of the knowledge required to create the prediction model

Original rating	Ternary	Binary	Unary
5-4	1	1	1
3-1	0	0	–
No rating	–1	–	–
Minimal knowledge representation	CF	ML	Graphs

CF and ML require *ternary* or *binary* representations, respectively, whereas graph-based approaches can be applied to the *unary* scenario

The graph-based methods are the only ones suitable for the unary scenario, which considerably reduces the knowledge required by the recommender. This means that, of the models studied in this evaluation, the graph-based ones are the only ones that can be implemented in scenarios where only positive interactions exist, as in the case of online judges. Thus, the comparison must be made considering that it is impossible to evaluate these techniques with the same level of knowledge. Graph-based methods are evaluated with the most minimal knowledge scenario (unary representation). However, the explicit representation of negative and unknown knowledge enhances the accuracy of machine learning and memory-based models.

In the following, we present the evaluation metrics used to compare accuracy.

4.4 Evaluation metrics

For every target user, recommendation methods output an ordered list with k items. Therefore, to evaluate their performance, we apply the following metrics commonly found in the literature [20, 38, 41, 42, 46], where r are the relevant items contained by the recommendation list with size k :

One-hit@k. It is the percentage of recommendations that contain at least a relevant item for the target user. It is defined as:

$$1H@k = \begin{cases} 1, & \text{if } |r| \geq 1 \\ 0, & \text{otherwise} \end{cases}$$

Precision@k. This metric shows the proportion of items relevant to the target user in the recommended list. The precision at k is computed as:

$$P@k = \frac{|r|}{k}$$

Precision@R (R-precision). Precision at k has the disadvantage that the total number of relevant items in the collection strongly influences this metric. This number is important for our evaluation as we are working with local models to provide interpretability. This way, the relevant items are those nodes accessible by the local neighbourhood in the graph. *R-precision* solves that issue: it points out the proportion of relevant items to the target user among the total relevant items of the test set, denoted as R . This metric is equivalent to $P@R$:

$$P@R = \frac{|r|}{|R|}$$

R represents the accessible items in the evaluation set. In the item-based graph, R contains the nodes linked to the item with which the target user has interacted. However, in the user-based graph, R contains the items with which the users whose nodes are linked to the target user node have interacted.

We have not considered *recall* to evaluate the performance of our recommenders as we are proposing interpretable local models that only retrieve items from the neighbourhood of the target item/user. Therefore, evaluating their exhaustiveness when recommending any relevant item makes no sense, as they are only accessible by a global approach.

When we designed the evaluation set-up, we also considered ranking metrics, for example, Mean Reciprocal Rank. However, we believe that using them does not make sense. Some of the techniques being evaluated (our graph-based methods and collaborative filtering based on neighbours) do rank the recommendation list. But, the machine learning techniques do not rank the recommended items. These techniques implement the recommendation task as a classification task, so they only predict if the items will be interesting for the target user, not determining which is the grade of the user's interest in the item. Therefore, we decided to use only precision-based metrics to evaluate the results obtained in the evaluation.

The following section describes the results obtained from evaluating our graph-based methods.

4.5 Evaluation results

Once we have seen what methodology we have carried out to perform the evaluation, we analyse the results in this section. In Sect. 4.5.1, we detail the precision achieved with the graph-based methods, while in Sect. 4.5.2, we compare the accuracies of the graph-based methods with those obtained with the classical techniques. Finally, in Sect. 4.5.3 we discuss the interpretability of the evaluated models. All the results achieved in this evaluation can be accessed on the GitHub repository indicated above.³

4.5.1 Performance of the graph-based approaches

The first step in our evaluation is to assess the performance of the graph-based recommendation approaches. To do so, we evaluate all possible link prediction methods and aggregation strategy set-ups. For the sake of readability, this section only presents the results obtained with the sub-sample corresponding to users that interacted with at least the 50% of the most popular items and a recommendations list size of $k = 10$. However, an exhaustive comparison with the remaining configurations ($k = [3, 5, 10]$) and sub-samples (25% and 12.5%) will be discussed in Sects. 4.5.2 and 4.6.

As a representative configuration, Table 4 presents the results of the evaluation carried out with $k = 10$ and filtering threshold $\theta = 5$. The results shown in that table are the mean of the values obtained for all the users involved in the test set, considering each evaluation metric. That is why the $1H@10$ value is not 1 or 0 because for some users the $1H@10$ value is 1, and for other users, the $1H@10$ is 0. Therefore, the mean value has to be in $[0, 1]$. In general, we can conclude that results with graph-based approaches are quite heterogeneous regarding the model (item graph or user graph), the aggregation method, the similarity metric, and the recommendation list size. However, we can find several patterns and highlight some particularities. Graph-based methods generally obtain excellent results when we use the *one hit*

³ <https://github.com/martcaro/Graph-Recommendations/tree/main/results>

Table 4 Mean results got from the item-based and user-based approaches when $k = 10$, $\theta = 5$ and users have interacted with at least 50% of the movies

Sim	Agg	Item			User		
		1H@10	P@10	P@R	1H@10	P@k	P@R
AA	Similarity	1.00*+	0.66*	0.66*+	1.00*+	0.64	0.43
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.66*+	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00	0.65	0.40
CN	Similarity	1.00*+	0.66*	0.66*+	1.00*+	0.64	0.43
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.66*+	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00*+	0.65	0.44*
EW	Similarity	1.00*+	0.66*	0.65	1.00*+	0.66	0.22
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.66*+	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00*+	0.68*+	0.31
JN	Similarity	1.00*+	0.66*	0.66*+	1.00*+	0.63	0.41
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.66*+	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00*+	0.65	0.43
PA	Similarity	1.00*+	0.66*	0.66*+	1.00*+	0.64	0.44*
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.66*+	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00*+	0.66	0.39
WAA	Similarity	1.00*+	0.66*	0.65	1.00*+	0.62	0.40
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.65	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00*+	0.66	0.42
WCN	Similarity	1.00*+	0.66*	0.65	1.00*+	0.62	0.40
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.65	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00*+	0.66	0.42
WPA	Similarity	1.00*+	0.66*	0.65	1.00*+	0.62	0.40
	Simple	1.00*+	0.56	0.57	1.00*+	0.66	0.36
	Weighted	1.00*+	0.66*	0.65	1.00*+	0.66	0.36
	Positional	0.99	0.55	0.53	1.00*+	0.66	0.42

The best results among aggregation methods are in bold. The best evaluation metric values are marked with *. The best results of metrics between approaches are denoted with +. *Sim* means the similarity metric used and *Agg* is the aggregation method used

metric. This good result is independent of the similarity measure or the graph approach (item or user-based). Therefore, we can conclude that it is inherent to the graph-based approach. Later, in Sect. 4.5.2 we present that other recommendation methods do not achieve such a good performance with this evaluation metric.

Regarding *precision@k* and *r-precision*, graph-based approaches are quite heterogeneous, although they are significant enough. While the *precision@k* values are very alike for all the link prediction methods in the item-based approach, we can highlight the performance of the *weighted voting* and the *highest similarity* aggregation strategies. The reason behind these results could be associated with using the similarity values to aggregate the estimations of the link prediction methods, which seems to be a helpful tool for getting successful recommendations. We can observe a similar pattern from the obtained scores with *r-precision*. The highest similarity and weighted voting work better than simple and positional voting (0.66 over 0.57 and 0.53, respectively). We can also note that WAA, WCN and WPA perform slightly worse than the other link prediction measures, although the difference is not remarkable.

The most significant difference between *precision@k* and *r-precision* emerges in the user-based method. The obtained values are not as good as the ones achieved with the item-based model. Moreover, it is not clear which aggregation method, and similarity metric performs the best. PA with the *highest similarity* and CN with the *positional voting* achieve the best results (0.44). However, JN also gets similar scores (0.43 with positional voting and 0.41 with the highest similarity). This behaviour could be related to the fact that PA considers the nodes at the ends of the edge. CN and JN take into account the relationships in neighbourhoods. In contrast, the rest of the similarity metrics use edge weights. WAA, WCN and WPA, achieve 0.42, all using the *positional voting*. They use weights but also use the knowledge from their simpler versions. From this, we could also conclude that the *positional voting* is the aggregation method that works better in most cases with the highest scores, although it is not the best aggregation method if we use AA, EW and PA. Therefore, *positional voting* sorts the list of recommendations better, which is understandable because of the nature of their work.

In conclusion, we can confirm that the item and user-based approaches perform equally well when finding relevant recommendations from the list of available items. However, if we only consider the local items represented by the neighbourhood of the graph, the item-based approach achieves better results. The item-based model seems to perform better when using knowledge about the weights of the graph. Nevertheless, the user-based approach is more uniform regarding this kind of knowledge, and it seems to work better with the *positional voting*.

4.5.2 Comparative evaluation with other recommendation techniques

In the next series of experiments, we compare the performance of the graph-based approaches to other recommendation techniques, specifically collaborative filtering algorithms (user and item-based approaches using Pearson coefficient as similarity measure), decision trees (DT), naive Bayes (NB), neural networks (NN), random forest (RF) and support vector machines (SVM). As explained previously, we must adapt them to provide recommendations in a minimal knowledge scenario by transforming the original ratings to represent only the item-user interactions. This process is illustrated by Table 3.

In Fig. 5, we compare the performance of all the considered approaches regarding the size of the recommended list. These mean results were computed using the data set whose users have interacted with at least 50% of the movies, being similar to the results obtained with other sub-samplings. The graph-based recommenders are those that achieved the best performance according to Sect. 4.5.1. We can observe that graph-based approaches perform

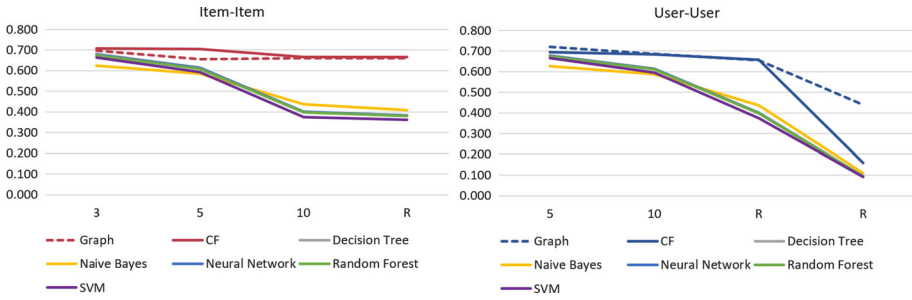


Fig. 5 Mean results obtained by P@k with $k \in \{3, 5, 10, R\}$ in both Item–Item and User–User configurations using the 50% sub-sample

better than other ML techniques. The item graph-based approach improves the performance of the ML techniques at 22% (using *precision@k* when $k = 10$) or 28% (using *r-precision*). In the case of the user graph-based approach, our proposal improves the performance of the ML techniques at 24% (using *precision@k* when $k = 10$) or 33% (using *r-precision*). CF is the only exception, achieving the same performance for the item–item configuration (0% of improvement when $k = 10$) but a higher precision for user–user (2% of improvement with *precision@k* and 28% with *r-precision*). However, it is essential to note that this CF algorithm is boosted by the inclusion of negative and unknown knowledge in the *ternary representation*.

Tables 5 and 6 summarise the results comparing different sub-samples. These values shown in those tables are the mean values for each evaluation metric calculated with the scores obtained from the test set. We also show the percentage improvements between our graph-based approaches and ML or CF (always comparing the best results achieved for each approach).

Again, we can conclude that the graph-based approaches are the methods that achieve higher performance with less knowledge (unary representation). Their results are particularly similar to CF with a ternary representation, especially in the item-based model. It is clear that machine learning techniques work worse than collaborative filtering and graph-based models. We cannot observe a significant difference if we observe the *one hit* results as it is the simplest metric, but this difference is remarkable if we pay attention to *precision@k* and *r-precision*.

There are no significant changes between sub-samples 50% and 25%. However, we can observe a different pattern in the results with the data set where users only interacted with 12.5% of the items. Here, scores decrease significantly, and we can conclude that the performance with this sub-sample is lower than the performance with more dense data sets, independently of the recommendation approach. It corroborates our initial supposition about the problem of *sparsity* when applying graph-based recommendation techniques. However, it also confirms that this problem also impacts other recommenders in a similar way.

4.5.3 Discussion of interpretability

In terms of interpretability, we can draw some conclusions. According to the literature, global methods are less interpretable than local methods. Thus, we can assume that our graph-based methods are more interpretable than the other evaluated recommendation methods but require less knowledge. In Fig. 6, we can see the graphical explanation of both models with a real

Table 5 Comparison between mean results obtained with item-based models

Met.	Sub.	k	HS	SV	WV	PV	DT	NB	NN	RF	SVM	CF	Δ (G-ML)	Δ (G-CF)	
1H@k	12.5	3	0.79	0.59	0.79	0.58	0.60	0.52	0.60	0.58	0.52	0.89	19%	-10%	
		5	0.88	0.76	0.87	0.73	0.64	0.54	0.63	0.61	0.54	0.94	24%	-6%	
		10	0.90	0.89	0.90	0.91	0.64	0.57	0.64	0.62	0.55	0.97	27%	-6%	
P@k	25	3	0.97	0.89	0.97	0.77	0.88	0.76	0.88	0.88	0.88	0.98	9%	-1%	
		5	0.99	0.94	0.98	0.90	0.94	0.80	0.80	0.94	0.93	0.99	5%	0%	
		10	0.99	0.99	1.00	0.99	0.95	0.82	0.82	0.95	0.95	0.93	1.00	5%	0%
	50	3	0.99	0.97	0.99	0.95	0.96	0.96	0.95	0.96	0.95	0.96	0.98	3%	1%
		5	1.00	1.00	1.00	0.98	0.98	0.98	0.98	0.98	0.97	0.96	0.99	2%	1%
		10	1.00	1.00	1.00	0.99	0.98	0.98	0.98	0.98	0.97	0.96	1.00	2%	0%
12.5	3	0.57	0.31	0.57	0.32	0.36	0.36	0.31	0.35	0.34	0.30	0.59	21%	-2%	
	5	0.48	0.32	0.52	0.32	0.29	0.29	0.25	0.28	0.27	0.24	0.54	23%	-2%	
	10	0.42	0.31	0.43	0.32	0.18	0.18	0.20	0.17	0.16	0.14	0.46	23%	-3%	
P@R	25	3	0.76	0.60	0.77	0.42	0.58	0.48	0.58	0.58	0.56	0.76	19%	1%	
		5	0.70	0.51	0.69	0.42	0.51	0.42	0.42	0.51	0.51	0.47	0.68	19%	2%
		10	0.59	0.44	0.59	0.41	0.33	0.33	0.33	0.33	0.33	0.30	0.59	26%	0%
	50	3	0.70	0.58	0.71	0.60	0.68	0.68	0.63	0.68	0.67	0.66	0.73	3%	-2%
		5	0.70	0.56	0.69	0.58	0.61	0.59	0.59	0.61	0.61	0.59	0.70	9%	0%
		10	0.66	0.56	0.66	0.55	0.40	0.44	0.44	0.40	0.40	0.38	0.66	22%	0%
12.5	All	0.46	0.31	0.49	0.30	0.22	0.23	0.21	0.20	0.20	0.17	0.54	26%	-s-5%	
25	All	0.62	0.48	0.62	0.42	0.34	0.34	0.33	0.34	0.34	0.31	0.62	28%	0%	
50	All	0.66	0.57	0.66	0.53	0.38	0.38	0.41	0.38	0.38	0.36	0.66	28%	0%	

The best results for each row are in bold with an error margin ± 0.01 . The columns HS, SV, WV, and PV denote graphs using highest similarity, simple voting, weighted voting, and positional voting, respectively. The two last columns show the improvement in our approach regarding the best ML technique (Δ (G-ML)) and CF (Δ (G-CF))

Table 6 Comparison between mean results got with user-based models

Met.	Sub.	k	HS	SV	WV	PV	DT	NB	NN	RF	SVM	CF	Δ (G-ML)	Δ (G-CF)	
1H@k	12.5	3	0.87	0.80	0.80	0.89	0.60	0.52	0.60	0.58	0.52	0.90	29%	-1%	
		5	0.95	0.88	0.88	0.94	0.63	0.54	0.63	0.61	0.54	0.95	32%	0%	
		10	0.99	0.91	0.91	0.98	0.64	0.57	0.64	0.62	0.55	0.99	35%	0%	
P@k	25	3	0.92	0.73	0.97	0.98	0.88	0.76	0.88	0.88	0.88	0.95	10%	3%	
		5	0.99	0.73	0.98	0.94	0.80	0.80	0.94	0.94	0.93	0.99	5%	0%	
		10	1.00	0.73	1.00	0.95	0.82	0.82	0.95	0.95	0.93	1.00	5%	0%	
P@k	50	3	0.97	0.99	1.00	0.98	0.96	0.95	0.96	0.95	0.96	0.95	4%	5%	
		5	1.00	1.00	1.00	0.98	0.98	0.98	0.98	0.97	0.96	1.00	2%	0%	
		10	1.00	1.00	1.00	0.98	0.98	0.98	0.98	0.97	0.96	1.00	2%	0%	
P@k	12.5	3	0.54	0.58	0.58	0.63	0.35	0.31	0.35	0.34	0.30	0.57	28%	6%	
		5	0.52	0.49	0.52	0.56	0.29	0.25	0.28	0.27	0.24	0.55	27%	1%	
		10	0.46	0.42	0.43	0.46	0.18	0.20	0.17	0.16	0.14	0.47	26%	-1%	
P@k	25	3	0.61	0.24	0.77	0.78	0.58	0.48	0.58	0.58	0.56	0.65	20%	13%	
		5	0.61	0.15	0.69	0.71	0.51	0.42	0.51	0.51	0.51	0.47	0.64	20%	7%
		10	0.56	0.07	0.59	0.60	0.33	0.33	0.33	0.33	0.33	0.60	27%	0%	
P@R	50	3	0.68	0.68	0.74	0.77	0.68	0.63	0.68	0.67	0.66	0.70	9%	7%	
		5	0.67	0.68	0.69	0.72	0.61	0.59	0.61	0.61	0.61	0.59	0.68	11%	4%
		10	0.66	0.66	0.66	0.68	0.40	0.44	0.40	0.40	0.40	0.38	0.66	24%	2%
P@R	12.5	R	0.35	0.26	0.28	0.32	0.05	0.05	0.05	0.05	0.04	0.11	30%	24%	
		25	R	0.33	0.22	0.49	0.07	0.07	0.07	0.07	0.07	0.06	0.13	42%	36%
		50	R	0.44	0.36	0.36	0.44	0.09	0.11	0.09	0.09	0.09	0.16	33%	28%

The best results for each row are in bold with an error margin ± 0.01 . The columns HS, SV, WV, and PV denote graphs using highest similarity, simple voting, weighted voting, and positional voting, respectively. The two last columns show the improvement in our approach regarding the best ML technique (Δ (G-ML)) and CF (Δ (G-CF))

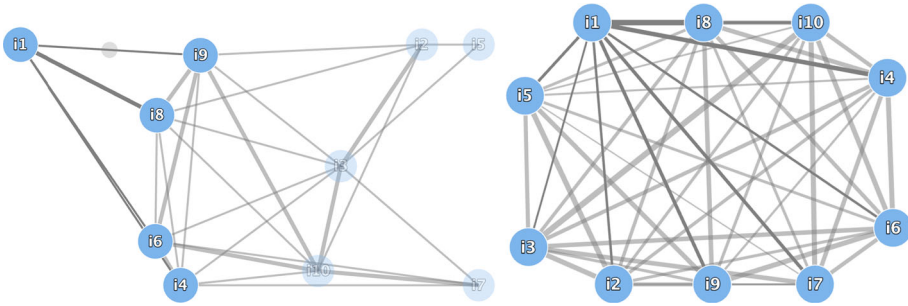


Fig. 6 Visual explanation of a real item–item recommendation. Nodes represent items (item 1 is being recommended), and link weights the number of common users. Left: Graph-based recommendation (local model). Right: Collaborative filtering (global model)

example. On the one hand, memory-based collaborative filtering (on the right in the figure) is a global method that makes predictions using the similarity of any element in the system. Therefore, all elements are connected to the others in the graphical representation. However, graph-based methods (on the left in the figure) are local. The link prediction-based similarity metrics used by our methods only use the neighbourhood information, so only the similarity between specific pairs of nodes is represented. This may be more understandable since the connections refer to the probability of recommendation or similarity. Therefore, as mentioned in the Introduction, we can consider our methods more explainable if the users understand their functionality.

4.6 Results and discussion

From the previous evaluation, we can draw several interesting conclusions that are also related to the requirement of interpretability:

- Performance of the graph-based approaches depends on the similarity measure and the aggregation method. Their appropriate configuration is significant, although the results are very similar in most cases.
- Item graph-based recommender performs better if we use Weighted Voting as the aggregation method and Edge Weight as the similarity measure. In the user-based graph case, their most effective configuration is not so explicit and depends on the information density in the data set.
- Graph-based methods and CF perform better than ML techniques in their corresponding minimal knowledge scenarios. Although machine learning techniques are used in the literature to make recommendations when explicit knowledge does not exist, they are not specific tools to tackle the problem of making recommendations with restricted knowledge. Additionally, most ML approaches are not interpretable (decision trees are the exception) as they are black boxes whose outcomes cannot be explained to the user.
- The similarity in the precision obtained between interaction graphs and memory-based collaborative filtering lies in the fact that we can consider memory-based methods as a specific configuration of interaction graphs in minimal knowledge scenarios. However, this scenario is different for both methods: graphs use a unary representation, while collaborative filtering uses a ternary representation. Therefore, graphs require less knowledge.

- According to the literature, graph-based recommender systems are considered more interpretable than traditional methods since the latter are global methods while graph-based models are local methods.
- If we compare item-based and user-based models (regardless of whether they are graph or collaborative filtering), we can conclude that item-based models are better than user-based models. It makes sense because, in item-based models, we obtain the recommendations directly from the items with which the target user has interacted. On the other hand, the user-based models first obtain a set of similar users and then the items to recommend. If we still need a user-based model, the best option is to use graph-based methods over collaborative filtering, as they provide better precision than the latter and are more interpretable.
- Our work has some limitations too. First, our methods are local, so we cannot access all the possible results available in the whole system. Therefore, we can have a lack of variability and serendipity. Second, we could conduct online experiments with users to find out their opinions about the explanations provided by our proposal and compare them to the baseline methods. This way, we could confirm the conclusions we extracted from the literature. Dispersion in the data set is the third limitation. It affects the precision of the recommender system, regardless of the recommendation method used. The denser the data set, the better the recommendations will be. This problem is not particular to our approach but is also one of the biggest problems in the recommender systems field. Finally, our approach computation complexity is quadratic ($O(n^2)$), as we can observe in Algorithm 1 and Algorithm 2. Collaborative filtering based on neighbours has the same computation complexity due to they also have to find similarities between users and items in a similar process to our approach. Although the collaborative filtering based on models (the machine learning techniques) that we have studied in this work get a low performance than our approach, their computation complexity is better, they have a linear complexity ($O(n)$), as they have to apply the model itself without using similarities.

Summarising the evaluation, we can conclude that our proposal has two major features: (1) it requires less knowledge but achieves a similar—or even higher—performance, and (2) it is interpretable for the target users, letting them understand the rationale of the recommendations.

5 Conclusions and future work

This paper presents a novel approach to provide interpretable and explainable recommendations on minimal knowledge scenarios. We develop two recommendation approaches based on interaction graphs representing either item–item or user–user relationships. Then, we use these data structures to find the similarities between graph nodes through several link prediction and aggregation techniques.

The main advantage of these methods is that they only require minimal knowledge about the user–item interaction, but the achieved performance is comparable to the standard recommendation techniques. Moreover, our approaches are easy to explain to the users, as they are interpretable local models.

This paper provides an exhaustive evaluation to find the most suitable configuration of the graph-based approach regarding link prediction and aggregation strategies. Then we compare its performance with other standard recommendation approaches in a minimal knowledge scenario: collaborative filtering based on memory, decision trees, naive Bayes,

neural networks, random forest and support vector machines. This evaluation demonstrates that our graph-based approaches require less knowledge but achieve similar or even higher performance and are easily interpretable by the users.

In future work, we can further evaluate our recommender approaches with real users to corroborate our conclusions. The evaluations with users can provide more reliable and accurate results regarding user satisfaction and experience. We would also like to explore the performance with different data set stratifications that let us explore the impact of sparsity and cold start in the recommenders. Moreover, we could use other options to apply to our approach instead of the similarity metrics and matrices obtained with them. For example, we could use other quasi-local or global similarity metrics from link prediction techniques, like Katx Index or random walks [27]. We do not know for sure how that could affect the performance of our graph-based method. We could hypothesise that it could be better than the performance obtained in this work though because using quasi-local or global metrics requires more knowledge that is useful to get better predictions. However, these options might reduce the interpretability of our approach, as we discussed in this work. We would need to do additional experimentation to confirm our hypothesis. Another option could be to use learning-based approaches based on link prediction over our graphs. This type of approach considers the link prediction problem as a binary classification task [27, 54]. So we could reconsider our recommendation task as the prediction of what links between users and items are going to appear, i.e. our future recommendations. We have also to study this approach to know how this option could influence the results presented in this work. Finally, we could also combine our two approaches, the user-based and the item-based, to try to solve the cold-start problem and the sparsity that our approach suffers as well, in a similar same way Arthur et al [4] did in their work.

Acknowledgements Supported by the Horizon 2020 Future and Emerging Technologies (FET) programme of the European Union through the iSee project (CHIST-ERA-19-XAI-008-PCI2020-120720-2), the UCM (Research Group 921330), and the PERXAI project PID2020-114596RB-C21, funded by the Ministry of Science and Innovation (MCIN/AEI/10.13039/501100011033).

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Aggarwal CC (2016) Recommender systems. Springer, Berlin
2. Aiolfi F (2013) Efficient top-n recommendation for very large scale binary rated datasets. In: Proceedings of the 7th ACM conference on Recommender systems, pp 273–280
3. Arrieta AB, Díaz-Rodríguez N, Del Ser J et al (2020) Explainable artificial intelligence (xai): concepts, taxonomies, opportunities and challenges toward responsible ai. *Inf Fus* 58:82–115
4. Arthur JK, Zhou C, Osei-Kwakye J et al (2022) A heterogeneous couplings and persuasive user/item information model for next basket recommendation. *Eng Appl Artif Intell* 114(105):132
5. Arya V, Bellamy RK, Chen PY, et al (2019) One explanation does not fit all: a toolkit and taxonomy of AI explainability techniques. arXiv preprint [arXiv:1909.03012](https://arxiv.org/abs/1909.03012)

6. Bagci H, Karagoz P (2016) Context-aware location recommendation by using a random walk-based approach. *Knowl Inf Syst* 47(2):241–260
7. Bobadilla J, Ortega F, Hernando A et al (2013) Recommender systems survey. *Knowl-Based Syst* 46:109–132
8. Bokde D, Girase S, Mukhopadhyay D (2015) Matrix factorization model in collaborative filtering algorithms: a survey. *Proc Comput Sci* 49:136–146
9. Caro-Martínez M, Jimenez-Díaz G (2017) Similar users or similar items? comparing similarity-based approaches for recommender systems in online judges. In: *International conference on case-based reasoning*, Springer, Berlin, pp 92–107
10. Caro-Martínez M, Recio-García JA, Jimenez-Díaz G (2019) An algorithm independent case-based explanation approach for recommender systems using interaction graphs. In: *International conference on case-based reasoning*, Springer, Berlin, pp 17–32
11. Chen J, Wang X, Wang C (2018) Understanding item consumption orders for right-order next-item recommendation. *Knowl Inf Syst* 57(1):55–78
12. Chiluka N, Andrade N, Pouwelse J (2011) A link prediction approach to recommendations in large-scale user-generated content systems. In: *European conference on information retrieval*, Springer, pp 189–200
13. Dooms S, Bellogín A, Pessemier TD et al (2016) A framework for dataset benchmarking and its application to a new movie rating dataset. *ACM Trans Intell Syst Technol (TIST)* 7(3):41
14. Ekstrand MD, Riedl JT, Konstan JA et al (2011) Collaborative filtering recommender systems. *Found Trends Hum Comput Interact* 4(2):81–173
15. Escalante HJ, Escalera S, Guyon I et al (2018) *Explainable and interpretable models in computer vision and machine learning*. Springer, Berlin
16. Friedrich G, Zanker M (2011) A taxonomy for generating explanations in recommender systems. *AI Mag* 32(3):90–98
17. Giarelis N, Kanakaris N, Karacapilidis N (2020) On the utilization of structural and textual information of a scientific knowledge graph to discover future research collaborations: a link prediction perspective. In: *International conference on discovery science*, Springer, Berlin, pp 437–450
18. Gilpin LH, Bau D, Yuan BZ, et al (2018) Explaining explanations: an overview of interpretability of machine learning. In: *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, IEEE, pp 80–89
19. Guo Q, Zhuang F, Qin C et al (2020) A survey on knowledge graph-based recommender systems. *IEEE Trans Knowl Data Eng* 34(8):3549–3568
20. Herlocker JL, Konstan JA, Terveen LG et al (2004) Evaluating collaborative filtering recommender systems. *ACM Trans Inf Syst (TOIS)* 22(1):5–53
21. Hu Y, Koren Y, Volinsky C (2008) Collaborative filtering for implicit feedback datasets. In: *2008 Eighth IEEE international conference on data mining*, IEEE, pp 263–272
22. Ji S, Pan S, Cambria E, et al (2020) A survey on knowledge graphs: Representation, acquisition and applications. *arXiv preprint arXiv:2002.00388*
23. Jimenez-Díaz G, Gómez-Martín PP, Gómez-Martín MA et al (2017) Similarity metrics from social network analysis for content recommender systems. *AI Commun* 30(3–4):223–234
24. Karimi M, Jannach D, Jugovac M (2018) News recommender systems-survey and roads ahead. *Inf Process Manage* 54(6):1203–1227
25. Koren Y, Bell R (2015) *Advances in collaborative filtering*. In: *Recommender systems handbook*. Springer, Berlin, pp 77–118
26. Koren Y, Bell R, Volinsky C (2009) Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37
27. Kumar A, Singh SS, Singh K et al (2020) Link prediction techniques, applications, and performance: a survey. *Phys A* 553(124):289
28. Li X, Chen H (2013) Recommendation as link prediction in bipartite graphs: a graph kernel-based machine learning approach. *Decis Support Syst* 54(2):880–890
29. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. *J Am Soc Inform Sci Technol* 58(7):1019–1031
30. Lichtenwalter RN, Lussier JT, Chawla NV (2010) New perspectives and methods in link prediction. In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp 243–252
31. Lops P, De Gemmis M, Semeraro G (2011) Content-based recommender systems: State of the art and trends. In: *Recommender systems handbook*. Springer, Berlin, pp 73–105
32. Lu J, Wu D, Mao M et al (2015) Recommender system application developments: a survey. *Decis Support Syst* 74:12–32

33. Lü L, Zhou T (2010) Link prediction in weighted networks: the role of weak ties. *EPL (Europhys Lett)* 89(1):18,001
34. Lü L, Zhou T (2011) Link prediction in complex networks: a survey. *Phys A* 390(6):1150–1170
35. Lü L, Medo M, Yeung CH et al (2012) Recommender systems. *Phys Rep* 519(1):1–49
36. Martínez V, Berzal F, Cubero JC (2016) A survey of link prediction in complex networks. *ACM Comput Surv (CSUR)* 49(4):1–33
37. Massa P, Avesani P (2004) Trust-aware collaborative filtering for recommender systems. In: *OTM confederated international conferences on the move to meaningful internet systems*, Springer, Berlin, pp 492–508
38. McFee B, Lanckriet GR (2010) Metric learning to rank. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp 775–782
39. Miranda C, Jorge AM (2008) Incremental collaborative filtering for binary ratings. In: *2008 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology*, IEEE, pp 389–392
40. Pan R, Zhou Y, Cao B, et al (2008) One-class collaborative filtering. In: *2008 Eighth IEEE international conference on data mining*, IEEE, pp 502–511
41. Pazzani MJ (1999) A framework for collaborative, content-based and demographic filtering. *Artif Intell Rev* 13(5–6):393–408
42. Powers DM (2020) Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*
43. Rendle S, Freudenthaler C, Gantner Z, et al (2012) BDPR: bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*
44. Ribeiro MT, Singh S, Guestrin C (2016) “why should i trust you?” explaining the predictions of any classifier. In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp 1135–1144
45. Ricci F, Rokach L, Shapira B (2015) Recommender systems: introduction and challenges. In: *Recommender systems handbook*. Springer, Berlin, pp 1–34
46. Said A, Bellogín A (2014) Comparative recommender system evaluation: benchmarking recommendation frameworks. In: *Proceedings of the 8th ACM conference on recommender systems*, ACM, pp 129–136
47. Sanz-Cruzado J, Pepa SM, Castells P (2018) Structural novelty and diversity in link prediction. In: *Companion Proceedings of the The Web Conference*, pp 1347–1351
48. Sarwar BM, Karypis G, Konstan JA et al (2001) Item-based collaborative filtering recommendation algorithms. *WWW* 1:285–295
49. Schafer JB, Frankowski D, Herlocker J, et al (2007) Collaborative filtering recommender systems. In: *The adaptive web*. Springer, Berlin, pp 291–324
50. Singh J, Anand A (2019) Exs: Explainable search using local model agnostic interpretability. In: *Proceedings of the twelfth ACM international conference on web search and data mining*, pp 770–773
51. Tintarev N (2007) Explanations of recommendations. In: *Proceedings of the 2007 ACM conference on Recommender systems*, pp 203–206
52. Volkovs M, Yu GW (2015) Effective latent models for binary feedback in recommender systems. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp 313–322
53. Vozalís E, Margaritis KG (2003) Analysis of recommender systems algorithms. In: *The 6th Hellenic European conference on computer mathematics and its applications*, pp 732–745
54. Wang P, Xu B, Wu Y et al (2015) Link prediction in social networks: the state-of-the-art. *Sci China Inf Sci* 58(1):1–38
55. Wang X, Wang D, Xu C, et al (2019) Explainable reasoning over knowledge graphs for recommendation. In: *Proceedings of the AAAI conference on artificial intelligence*, pp 5329–5336
56. Xian Y, Fu Z, Muthukrishnan S, et al (2019) Reinforcement knowledge graph reasoning for explainable recommendation. In: *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval*, pp 285–294
57. Xie F, Chen Z, Shang J et al (2015) A link prediction approach for item recommendation with complex number. *Knowl-Based Syst* 81:148–158
58. Yang X, Zhang Z, Wang K (2012) Scalable collaborative filtering using incremental update and local link prediction. In: *Proceedings of the 21st ACM international conference on information and knowledge management*, pp 2371–2374
59. Zhang B, Choudhury S, Al-Hasan M, et al (2016) Trust from the past: Bayesian personalized ranking based link prediction in knowledge graphs. *Tech. rep.*, Pacific Northwest National Lab.(PNNL), Richland, WA (United States)

60. Zhang M, Wang Q, Xu W, et al (2018) Discriminative path-based knowledge graph embedding for precise link prediction. In: European conference on information retrieval, Springer, Berlin, pp 276–288
61. Zhao D, Zhang L, Zhao W (2016) Genre-based link prediction in bipartite graph for music recommendation. *Proc Comput Sci* 91:959–965
62. Zhou T, Ren J, Medo M et al (2007) Bipartite network projection and personal recommendation. *Phys Rev E* 76(4):046,115

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Marta Caro-Martínez got her PhD in Computer Science in 2022 at the Complutense University of Madrid, where she also got a degree in Computer Science in 2015 and a Master's degree in the same field in 2017. Her research work focuses on recommender systems and explainable artificial intelligence. She uses graphs and social network analysis techniques to implement recommendations and explanations. She has also researched interactive visualization and augmented reality applications for museums. Her publications have appeared at several international conferences and journals.



Guillermo Jiménez-Díaz is Computer Research Scientist and Associate Professor at Universidad Complutense Madrid. He received his PhD Universidad Complutense Madrid in Computer Science. His thesis focused on using virtual environments and active-based learning to teach object-oriented programming. His research is concerned to recommender systems and its combination with techniques from social network analysis. Our research is applied in two different domains, namely tourism and e-learning. He is also interested in the application of augmented reality technologies in Museums.



Juan A. Recio-Garcia is Associate Professor at the Department of Software Engineering and Artificial Intelligence at the Computer Science Faculty at the Complutense University of Madrid, where he held the position of Head of Department from 2015 to 2019. Currently, he holds the BOSCH-UCM Honorary Chair on Artificial Intelligence applied to Internet of Things and is Board Member of the IAAA Student Chapter in Mexico for the promotion of artificial intelligence. He is lead investigator of several national-funded projects, and he has conducted several contracts with companies in the area of artificial intelligence. His research has focused on the confluence of software engineering and case-based reasoning, developing the COLIBRI platform for building CBR systems. He has also worked in the areas of context-aware and social recommender systems. Currently, his research is focused on explainable artificial intelligence (XAI).