Neural Computing and Applications (2023) 35:14081–14092
https://doi.org/10.1007/s00521-023-08410-6

ORIGINAL ARTICLE

Neural group recommendation based on a probabilistic semantic aggregation

Jorge Dueñas-Lerín[2,3] · Raúl Lara-Cabrera[1,2] · Fernando Ortega[1,2] · Jesús Bobadilla[1,2]

Received: 28 July 2022 / Accepted: 13 February 2023 / Published online: 22 March 2023

Abstract
Recommendation to groups of users is a challenging subfield of recommendation systems. Its key concept is how and where to make the aggregation of each set of user information into an individual entity, such as a ranked recommendation list, a virtual user, or a multi-hot input vector encoding. This paper proposes an innovative strategy where aggregation is made in the multi-hot vector that feeds the neural network model. The aggregation provides a probabilistic semantic, and the resulting input vectors feed a model that is able to conveniently generalize the group recommendation from the individual predictions. Furthermore, using the proposed architecture, group recommendations can be obtained by simply feedforwarding the pre-trained model with individual ratings; that is, without the need to obtain datasets containing group of user information, and without the need of running two separate trainings (individual and group). This approach also avoids maintaining two different models to support both individual and group learning. Experiments have tested the proposed architecture using three representative collaborative filtering datasets and a series of baselines; results show suitable accuracy improvements compared to the state of the art.

Keywords Group recommender system · Collaborative filtering · Aggregation models · Deep learning

1 Introduction

Personalization is one of the fields of Artificial Intelligence (AI) that has a greater impact on the lives of individuals. We can find a multitude of services that provide us with a personalized choice of news, videos, songs, restaurants, clothes, travels, etc. The most relevant tech companies make extensive use of personalization services: Amazon, Netflix, Spotify, TripAdvisor, Google, TikTok, etc. These companies generate their personalized recommendations using Recommender System (RS) [1] applications. Recommender System (RS) provides to their users personalized products or services (items) by filtering the most relevant information regarding the logs of items consumed by the users, the time and place that took place, as well as the existing information about users, their social networks, and the content of items (texts, pictures, videos, etc.). We can classify Recommender System (RS) attending to their filtering strategy as demographic [2], content-based [3], context-aware [4], social [5], Collaborative Filtering (CF) [6, 7] and filtering ensembles [8, 9]. Currently, the Matrix Factorization (MF) [10] machine learning model is used to obtain accurate and fast recommendations between input data (votes). Matrix Factorization (MF) translates the very sparse and huge matrix of discrete votes (from users to items) into two dense and relatively small matrices of real values. One of the matrices contains the set of short and dense vectors representing users, whereas the second matrix vectors represent items. Each vector element (real

Jorge Dueñas-Lerín, Raúl Lara-Cabrera, Fernando Ortega and Jesús Bobadilla have contributed equally to this work.

✉ Jorge Dueñas-Lerín
jorgedl@alumnos.upm.es

Raúl Lara-Cabrera
raul.lara@upm.es

Fernando Ortega
fernando.ortega@upm.es

Jesús Bobadilla
jesus.bobadilla@upm.es

[1] Departamento de Sistemas Informáticos, Universidad Politécnica de Madrid, Alan Turing s/n, 28031 Madrid, Spain

[2] KNODIS Research Group, Universidad Politécnica de Madrid, 28031 Madrid, Spain

[3] Universidad Politécnica de Madrid, Madrid, Spain

value) is called the '*hidden factor value*', since it represents some complex and unknown relationship between the input data (votes).

However, Matrix Factorization (MF) machine learning models are fast and accurate, and they also present a remarkable drawback: They cannot detect, in their hidden factors, the complex nonlinear relationships between the input data. Neural Network (NN) can solve this problem through their nonlinear activation functions. Neural Network (NN)-based Recommender System (RS) [2, 11] makes a compression of information by coding the patterns of the rating matrix in their embeddings and hidden layers [12]. These embeddings play the role of the Matrix Factorization (MF) hidden factors, enriching the result by incorporating non-linear relations. The most well-known Neural Network (NN) base Recommender System (RS) approaches are Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP) [13].

Group Recommendation (GR) [7, 14] is a subfield of the Recommender System (RS) area where recommendations are made to sets of users instead of to individual users (e.g., to recommend a movie to a group of three friends). As in the regular Recommender System (RS), the goal is to make accurate recommendations to the group. In this case, several policies can be followed; the most popular are (a) to minimize the mean accuracy error: to recommend the items that, on average, most like to all the group members, and (b) to minimize the maximum accuracy error: to recommend the items that does not excessively dislike to any of the group members. It is important to state that there are not open datasets containing group information to be used by group recommendation models; for this reason, generally randomly generated groups are used for training and testing research models.

Regardless of the machine learning approach used to implement Group Recommendation (GR) Recommender System (RS), the most notable design concept is to establish where to locate the aggregation stage to convert individual information to group information. The general rule is the sooner the aggregation stage, the better the performance of Group Recommendation (GR) [14]. There are three different locations where group information can be aggregated into a unified group entity: (a) before the model, (b) in the model, and (c) after the model. The most intuitive approach is to combine individual recommendations into a unified group recommendation (option c) [15]. This approach is known as Individual Preference Aggregation (IPA) and requires processing several individual recommendations followed by rank aggregation. However, the process is slow and not particularly accurate. On the other hand, to consider the entire group for the recommendation, we should work before or inside the model (options a or b). These approaches are known as Group Preference Aggregation (GPA). Aggregating group information before the model requires working with the user-item interaction matrix in a higher-dimensional space, which can lead to misinformation problems. To aggregate group information in the model, we need to work with the user's hidden vector in the low-dimensional space.

Aggregating several hidden vectors from individual users into a unified virtual user hidden vector [16] avoids compute the model predictions several times and makes the rank aggregation stage unnecessary. In addition, it takes advantage of operating with condensed information coming from the Matrix Factorization (MF) compression of information: the virtual user can be obtained simply by averaging the representative short and dense vectors of the users group; this is efficient and accurate. An interesting question is can Neural Network (NN) operate the same way that Matrix Factorization (MF) does to obtain virtual users and generate recommendations? First, note that many Neural Network (NN)-based Recommender System (RS) models compress the user embedding in a different latent space than the item embedding, and it can be a problem; then, the Neural Network (NN) nonlinear ensemble representations are more complex than the Matrix Factorization (MF) hidden factor representations; consequently, simply averaging the ensembles of the users in the group does not automatically ensure a representative virtual user embedding. Furthermore, model-based aggregations (option 'b' in the previous paragraph) are model dependent, and then, it is necessary to design and test different solutions for different Neural Network (NN)-based Recommender System (RS) models, whereas the Neural Network (NN) latent spaces are the state of the art to catch users and items relations, some other machine learning approaches have been designed, such as the use of the random walk with restart method [17] providing a framework to relate users, items, and groups, and to exploit the item content and the profiles of the users. A three-stage method [18] is proposed to increase the precision and fairness of Group Recommendation (GR), where binary Matrix Factorization (MF), graphs and the dynamic consensus model are processed sequentially. Some relevant and current GR research aims to make use of the concept of member preference (influence or expertise) concept, based on similarity and trust. The key idea is to detect the group leaders as group members that are trusted more than others and have more influence than others. In [19], fuzzy clustering and an implicit trust metric are combined to find neighborhoods. Group Recommendation (GR) based on an average strategy applied to user preference differences [20] has been combined with trusted social networks to correct recommendations. An aggregation approach for GR mimics crowdsourcing concepts to estimate the level of expertise of group members [21]; it is implemented using parameters of

sensitivity and specificity. The impact of social factors on a Group Recommendation (GR) computational model is evaluated in [22], using the expertise factor, the influences of personality, preference similarities, and interpersonal relationships.

In this paper, we present how we can generate Group Recommendation (GR) using Neural Network (NN)-based Recommender System (RS) by training the model using raw Collaborative Filtering (CF) data (i.e., the ratings of individual users to the items without any additional information). The Group Recommendation (GR) have been tested using the two most popular implementations of Neural Network (NN)-based Recommender System (RS): Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP). As stated previously, to make recommendations to groups using Neural Network (NN)-based Recommender System (RS), information of the individual users must be aggregated. The chosen information aggregation design is to merge the users of the group in the input vector that feeds the user embedding of the Neural Network (NN). This aggregation design is not novel, since it has been used by [23] applied to a Multi-Layer Perceptron (MLP) architecture. However, our approach combines several innovative aspects in comparison with the state of the art. On the one hand, the aggregation of the users in the group is a probabilistic function rather than a simple multi-hot encoding [23]; this better captures the relative importance of users in the input vector that feeds the Neural Network (NN), moreover: This aggregation approach serves as front-end for any Neural Network (NN) Group Recommendation (GR) model. On the other hand, we propose the use of a simple Recommender System (RS) Neural Network (NN) model (Generalized Matrix Factorization (GMF)) instead of the deepest Multi-Layer Perceptron (MLP) one [23]; the hypothesis is that complex models overfit Group Recommendation (GR) scenarios, since they are designed to accurately predict individual predictions, whereas Group Recommendation (GR) must satisfy an average of the tastes in the group of users, that is, Group Recommendation (GR) should be designed to generalize the set of individual tastes in the group. Furthermore, the proposed architecture just needs a single training to provide both individual recommendations and group recommendations; particularly, the model is trained by only using individual recommendations (as in regular Recommender System (RS)). Once the model is trained to return individual predictions, we can fill the input vector by aggregating all the users in the group, then feedforward the trained model and finally obtain the recommendation for the group of users. Anyway, the impact of these innovative aspects can be evaluated in Sect. 3, where we empirically compare the proposed aggregation designs with respect to the main baseline [23]

In summary, the Group Recommendation (GR) state of the art presents the following drawbacks: (a) Some research relies on additional data to the Collaborative Filtering (CF) ratings, such as trust or reputation information that is not available on the majority of datasets, (b) different proposals make the aggregation of individual users before (Individual Preference Aggregation (IPA)) or after (Ranking) the model, making it impossible to benefit from the machine learning model inner representations (Group Preference Aggregation (GPA)), and (c) The proposed GR neural model solutions tend to apply architectures designed to make individual recommendations, rather than group ones; this leads to the model overfitting and to a low scalability referred to the number of users in a group. To fill the gap, our proposed model: (a) Acts exclusively on Collaborative Filtering (CF) ratings, (b) Makes user aggregation in the model, and (c) Its model depth and design enable adequate learning generalizations. Additionally, the provided experiments test the proposed model according to different aggregation strategies to set the group labels used in the learning stage. In contrast, a notable limitation of our architecture and the experiments is the lack of testing on particularly demanding scenarios such as cold start in groups users, extremely sparse data sets, impact of popular item bias, and fear Group Recommendation (GR).

The rest of the paper is structured as follows: Sect. 2 introduces the tested models and aggregation functions; Sect. 3 describes the experiment design, the selected quality measures, the chosen datasets and shows the results obtained; Sect. 4 provides their explanations; and Sect. 5 highlights the main conclusions of the article and the suggested future work.

2 Proposed model

In Collaborative Filtering (CF), interactions (purchase, viewing, rating, etc.) between users and items are stored in a sparse matrix since it is common for users to interact only with a small proportion of the available items and, in the same way, only a small percentage of existing users interact with the items. The sparsity levels of this matrix are around 95–98% as shown in Table 2. To handle this sparsity, current Collaborative Filtering (CF) models based on Neural Network (NN) [13] work with a projection of users and items into a low-dimensional latent space using embedding layers. Embedding layers are a very popular type of layer used in Neural Network (NN) that receive as input any entity and return a vector with a low-dimensional representation of the entity in a latent space. These vectors are commonly named *latent factors*. In order to transform the entity into its low-dimensional representation, the

embedding layer first transforms the entity into a one hot encoding representation (typically using a hash function). Figure 1 summarizes this process.

In the context of Collaborative Filtering (CF), two embedding layers are required: one for the users and the other for the items. Later, both embedding layers are combined using a Neural Network (NN) architecture (see Fig. 2). For example, the models Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP) use a dot layer and a concatenate layer followed by some fully connected dense layers as architectures, respectively.

Formally, we define a Neural Network (NN) model Φ that predicts the rating that a user u will give to an item i ($\hat{r}_{u,i}$) combining the latent factors provided by the embedding layer (Emb_L) of the user u ($\vec{l_u}$) and the item i ($\vec{l_i}$):

$$\begin{aligned} &\text{Emb}_L(u) = \vec{l_u} \\ &\text{Emb}_L(i) = \vec{l_i}, \\ &\Phi(\vec{l_u}, \vec{l_i}) = \hat{r}_{u,i} \end{aligned} \tag{1}$$

As stated in Sect. 1, when working with Group Recommendation (GR), a straightforward strategy is Individual Preference Aggregation (IPA) [24]. This strategy makes a prediction for each member of the group and then performs an aggregation. This strategy does not treat the group as a whole. If we have a group of users $G = \{u_1, u_2, ..., u_n\}$, the prediction of the rating of this group G to an item i ($\hat{r}_{G,i}$) is computed as the average value of the individual predictions:

$$\hat{r}_{G,i} = \frac{1}{\|G\|} \sum_{u \in G} \hat{r}_{u,i} = \frac{1}{\|G\|} \sum_{u \in G} \Phi(\vec{l_u}, \vec{l_i}) \tag{2}$$

On the other hand, the Group Preference Aggregation (GPA) strategies take into account the group as a whole. It should be noted that the order of users within the group and the length of it should not affect the aggregation; thus, the aggregations should meet the constraints of permutation invariant and fixed result length [23]. Our goal with the Group Preference Aggregation (GPA) strategy is to be able to obtain a prediction $\hat{r}_{G,i}$ with a single forward propagation

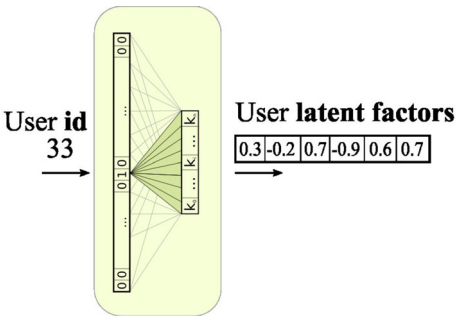

Fig. 1 Embedding layer schema

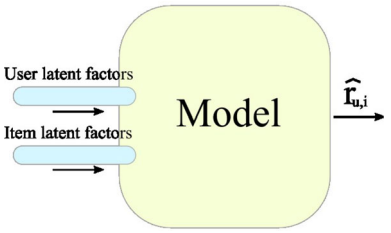

Fig. 2 Collaborative filtering-based neural network model

and to treat the group as a whole entity. We can achieve this by aggregating the latent factors of each user that belongs to the group to obtain the latent factor of the group $\vec{l}_G$. Once the latent factors of the group are aggregated, the model Φ can be used to compute the predictions:

$$\begin{aligned} &\text{Emb}_L(G) = \vec{l}_G \\ &\hat{r}_{Gi} = \Phi(\vec{l}_G, \vec{l}_i) \end{aligned} \tag{3}$$

The aggregation of group latent factors in embedding layers can be achieved by modifying the input of the Neural Network (NN). As mentioned previously, embedding layers have as input a one hot representation of the entities. This approach is adequate when performing individual predictions, however, for group recommendations, we need to apply a multi-hot representation to the users' embedding layer, i.e., we encode the group by setting multiple inputs of the user embedding layer (the inputs related with the users that belong to the group) to a value higher than 0. This encoding allows us to take into account all group users at the same time for the extraction of latent factors of the group $\vec{l}_G$.

The simplest aggregation, which is used by the DeepGroup model [23], is to use as input for embedding a constant value proportional to the size of the group. We define the input of the user's embedding layer for the user u as

$$EmbeddingInput_{\text{Average}}(u) = \begin{cases} \frac{1}{\|G\|} & if \quad u \in G \\ 0 & if \quad u \notin G \end{cases} \tag{4}$$

We call this aggregation '*Average*' since the embedding layer will generate the group latent factor equal to the average of the latent factors of all users in the group.

Recommender System (RS) can give better predictions the more information they have about users, so to take advantage of this fact, we have tested the '*Expertise*' aggregation in which we give a weight to the users proportional to the number of votes they have entered into the system. Let $\|R_u\|$ the number of ratings of the user u, the input of the users' embedding layer for the user u is defined as

Table 1 Complete aggregation example

(a) Rating count.

User	u_{13}	u_{24}	u_{30}	u_{42}
#Rating	2	5	6	3

(b) Input values to the users' embedding layer.

Strategy/User	u_{13}	u_{24}	u_{30}	u_{42}
Average	0,25	0,25	0,25	0,25
Expertise	0,13	0,31	0,38	0,19
Softmax	0,01	0,26	0,70	0,03

Users' latent factors assuming a latent space of size 3.

User/factor	l_1	l_2	l_3
u_{13}	0,1	0,6	0,3
u_{24}	0,7	0,2	0,9
u_{30}	0,8	0,4	0,1
u_{42}	0,5	0,7	0,8

Group latent factors using different aggregations

Agg factor	l_{G_1}	l_{G_2}	l_{G_3}
Average	0,525	0,475	0,525
Expertise	0,629	0,425	0,508
Softmax	0,758	0,359	0,331

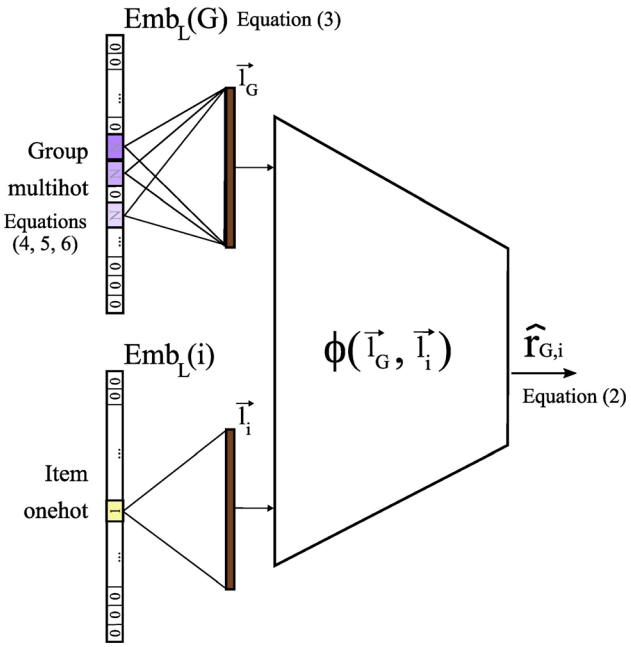

Fig. 3 Graphical representation of the proposed model

$$EmbeddingInput_{\text{Expertise}}(u) = \begin{cases} \dfrac{\|R_u\|}{\sum_{g \in G} \|R_g\|} & if \quad u \in G \\ 0 & if \quad u \notin G \end{cases} \tag{5}$$

In addition to the '*Expertise*' aggregation, we also proposed the '*Softmax* aggregation as a smooth version of the '*Expertise*' aggregation. In this case, the input of the users' embedding layer for the user u is defined as

$$EmbeddingInput_{\text{Softmax}}(u) = \begin{cases} \dfrac{e^{\|R_u\|}}{\sum_{g \in G} e^{\|R_g\|}} & if \quad u \in G \\ 0 & if \quad u \notin G \end{cases} \tag{6}$$

In Fig. 3, we can see where the equations fit in the group recommendation process. The first step is to generate the multi-hot vector with some of the described aggregation (Eqs. 4, 5, 6). This vector (multi-hot representation of the group) is fed into the embedding layer to obtain a vector of the latent factors of the groups $\vec{l}_G$ (Eq. 3). Once the latent

factors of the group and the item are obtained, they are used to feed the model Φ (Generalized Matrix Factorization (GMF) or Multi-Layer Perceptron (MLP)) and produce the rating prediction for the group G on the item i (Eq. 2).

In Table 1, we can find an example with some users (13, 24, 30 and 42) with different rating counts (Table 1a), their input values to the users' embedding layer in a multi-hot fashion (Table 1b), their individual latent factors (Table 1c), and the final group latent factors with different aggregations (Table 1d).

3 Experimental evaluation

In this section, we show the experiments carried out to validate the aggregation proposed in this manuscript. As previously stated, the experiments have been performed using the most popular Neural Network (NN)-based Recommender System (RS) architectures: Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP). We have chosen these two architectures because they are the best known and offer the best results for individual predictions. However, the aggregation strategies proposed can be applied to any Neural Network (NN) architecture based on embedding layers.

The choice of datasets has been made considering that (a) there are no open datasets containing information on group voting; and (b) Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP) models should be trained using individual voting, since the proposed aggregations allow computing predictions for groups on already trained models. For these reasons, we have chosen the following gold standard datasets in the field of Recommender System (RS): `MovieLens1M` [25], the most popular dataset in the research of Recommender System (RS); `FilmTrust` [26], a dataset smaller than `MovieLens1M` to measure the performance of the aggregation in datasets with a lower number of users, items, and ratings; and `MyAnimeList`, a dataset with a range of votes higher than the `MovieLens1M`. Other popular datasets such as `Netflix Prize` or `MovieLens10M` have not been selected due to the high computational time required to train and test the models.

Table 2 Main parameters of the datasets used in the experiments

Dataset	#users	#items	#ratings	Scores	Sparsity
Movie lens1M	6040	3706	911,031	1–5	95.94
Filmtrust	1508	2071	35,497	0–5	87.98
My anime list	19,179	2692	548,967	1–10	98.94

The main parameters of the selected datasets can be found in Table 2.

The generation of synthetic groups has been carried out in such a way that all groups have voted at least five items in test. In this way, it is possible to evaluate both the quality of the predictions and the quality of the recommendations to the groups as detailed below. Groups of different sizes (from 2 to 10 users) have been generated. For each group size, 10000 synthetic groups have been generated. The generation of a group has been carried out following the following algorithm:

1. Define the size of the group S.
2. Random select 5 items rated in test by at least S users.
3. Find all users who have rated the 5 items selected in 2.
4. If we found fewer than S users, go back to 2. Otherwise, random select S users and create a group.

To measure the quality of the predictions for the group, we have calculated the mean absolute error

$$MAE = \frac{1}{\#\text{groups}} \sum_{G} \frac{1}{\|G\| \cdot \|I_G\|} \sum_{u \in G} \sum_{i \in I_G} \mid \hat{r}_{G,i} - r_{u,i} \mid, \tag{7}$$

the mean squared error

$$MSE = \frac{1}{\#\text{groups}} \sum_{G} \frac{1}{\|G\|} \frac{1}{\|G\| \cdot \|I_G\|} \sum_{u \in G} \sum_{i \in I_G} \left(\hat{r}_{G,i} - r_{u,i}\right)^2, \tag{8}$$

and mean maximum group error

$$MAX = \frac{1}{\#\text{groups}} \sum_{G} \max_{u \in G} \max_{i \in I_G} \mid \hat{r}_{G,i} - r_{u,i} \mid, \tag{9}$$

where I_G denotes the items rated by the group G

To measure the quality of the recommendations for the group, we have calculated the Normalized Discounted Cumulative Gain (NDCG) score

$$NDCG@N = \frac{1}{\#groups} \sum_{G} \frac{DCG_G@N}{IDCG_G@N}, \tag{10}$$

$$DCG_G@N = \sum_{i \in X_G^N} \frac{\bar{r}_{G,i}}{log_2(pos_G(i) + 1)}, \tag{11}$$

$$IDCG_G@N = \sum_{i \in T_G^N} \frac{\bar{r}_{G,i}}{log_2(ipos_G(i) + 1)}, \tag{12}$$

where N is the number of items recommended to the group (in our experiments $N = 5$ according to the generation of synthetic groups), X_G^N is the set of N items recommended to the group G, $\text{pos}_G(i)$ is the position of the item i in the group's G recommendation list, T_G^N is the set of the top N items for the group G, $\text{ipos}_G(i)$ is the ideal rank of the item

Table 3 Mean absolute error

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	0.74205 (0.409)	0.76075 (0.341)	0.76893 (0.299)	0.77009 (0.271)	0.77745 (0.249)	0.77659 (0.234)	0.77681 (0.221)	0.77599 (0.212)	0.77558 (0.201)
GMF Expertise	0.74393 (0.41)	0.76207 (0.341)	0.77018 (0.299)	0.77155 (0.27)	0.779 (0.249)	0.77782 (0.234)	0.77834 (0.221)	0.77729 (0.211)	0.77685 (0.201)
GMF Softmax	0.74246 (0.409)	0.7608 (0.341)	0.76891 (0.299)	0.77012 (0.27)	0.77751 (0.249)	0.7766 (0.234)	0.77687 (0.221)	0.77602 (0.212)	0.77562 (0.201)
MLP IPA	0.74956 (0.444)	0.77342 (0.361)	0.78055 (0.313)	0.78211 (0.28)	0.78853 (0.258)	0.78633 (0.241)	0.78678 (0.228)	0.78591 (0.219)	0.78509 (0.207)
MLP Avg DeepGroup	**0.7236** **(0.486)**	**0.74289** **(0.404)**	**0.74916** **(0.355)**	**0.75018** **(0.32)**	**0.75656** **(0.295)**	0.75537 (0.275)	**0.75551** **(0.261)**	**0.75388** **(0.25)**	**0.75355** **(0.238)**
MLP Expertise	0.72596 (0.486)	0.74432 (0.405)	0.75031 (0.355)	0.75132 (0.32)	0.75787 (0.295)	0.75607 (0.276)	0.75709 (0.261)	0.75481 (0.25)	0.755 (0.238)
MLP Softmax	0.72407 (0.485)	0.74297 (0.404)	0.74925 (0.355)	0.75019 (0.32)	0.75669 (0.295)	**0.75531** **(0.275)**	0.7556 (0.261)	0.75389 (0.25)	0.75361 (0.238)

(a) MovieLens1M

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	0.61552 (0.451)	0.71033 (0.32)	0.73011 (0.28)	0.73419 (0.252)	0.73606 (0.232)	0.73832 (0.217)	0.74045 (0.203)	0.74298 (0.193)	0.74212 (0.185)
GMF Expertise	0.6149 (0.444)	0.71239 (0.319)	0.73144 (0.281)	0.73583 (0.252)	0.73742 (0.232)	0.7396 (0.217)	0.74166 (0.203)	0.74418 (0.193)	0.74336 (0.184)
GMF Softmax	0.61512 (0.448)	0.71088 (0.319)	0.73035 (0.28)	0.73445 (0.252)	0.73624 (0.232)	0.73847 (0.217)	0.74057 (0.203)	0.74309 (0.193)	0.74222 (0.185)
MLP IPA	0.58165 (0.442)	0.70368 (0.325)	0.72062 (0.281)	0.7252 (0.252)	0.72788 (0.232)	0.73073 (0.217)	0.73353 (0.203)	0.73623 (0.193)	0.73525 (0.185)
MLP Avg DeepGroup	0.58199 (0.447)	**0.70129** **(0.319)**	**0.71693** **(0.275)**	**0.7212** **(0.247)**	**0.72421** **(0.228)**	**0.727** **(0.214)**	**0.72976** **(0.2)**	**0.7328** **(0.191)**	**0.73188** **(0.183)**
MLP Expertise	**0.57903** **(0.453)**	0.70449 (0.321)	0.71891 (0.276)	0.72315 (0.247)	0.72581 (0.228)	0.72865 (0.213)	0.73127 (0.2)	0.73429 (0.191)	0.73336 (0.183)
MLP Softmax	0.58063 (0.45)	0.70226 (0.319)	0.71734 (0.275)	0.72153 (0.247)	0.72443 (0.228)	0.7272 (0.214)	0.72993 (0.2)	0.73295 (0.191)	0.73202 (0.183)

(b) FilmTrust

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	**0.93068** **(0.567)**	0.95819 (0.477)	0.97595 (0.417)	0.99149 (0.38)	1.0017 (0.346)	1.01404 (0.329)	1.01942 (0.312)	1.022 (0.297)	1.02445 (0.282)
GMF Expertise	0.93675 (0.572)	0.96424 (0.48)	0.98092 (0.419)	0.99603 (0.382)	1.00649 (0.349)	1.01805 (0.331)	1.02352 (0.314)	1.0258 (0.3)	1.0284 (0.284)
GMF Softmax	0.93219 (0.568)	0.95848 (0.477)	0.97559 (0.417)	0.99104 (0.38)	1.00133 (0.346)	1.01363 (0.329)	1.0191 (0.312)	1.02173 (0.297)	1.02422 (0.282)
MLP IPA	0.95479 (0.585)	0.98155 (0.484)	0.99709 (0.42)	1.00977 (0.381)	1.01745 (0.347)	1.02794 (0.329)	1.03188 (0.311)	1.03335 (0.298)	1.03451 (0.282)
MLP Avg DeepGroup	0.93161 (0.618)	0.95609 (0.515)	**0.96961** **(0.45)**	0.98456 (0.408)	0.99331 (0.371)	1.0052 (0.351)	1.00857 (0.332)	1.01039 (0.317)	1.01233 (0.3)
MLP Expertise	0.93803 (0.622)	0.96132 (0.517)	0.97587 (0.453)	0.98923 (0.409)	0.99851 (0.373)	1.00879 (0.353)	1.01258 (0.334)	1.01493 (0.319)	1.01599 (0.302)
MLP Softmax	0.93351 (0.619)	**0.95594** **(0.515)**	0.97007 (0.451)	**0.9843** **(0.408)**	**0.99329** **(0.371)**	**1.00486** **(0.351)**	**1.00844** **(0.332)**	**1.0101** **(0.317)**	**1.01211** **(0.3)**

(c) MyAnimeList

Table 4 Mean squared error

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	**0.87504** **(0.914)**	**0.90947** **(0.767)**	0.92437 (0.676)	0.92648 (0.609)	0.94191 (0.567)	0.93896 (0.53)	0.9376 (0.5)	0.93832 (0.48)	0.93571 (0.454)
GMF Expertise	0.88044 (0.918)	0.91329 (0.77)	0.92754 (0.678)	0.92979 (0.61)	0.94504 (0.568)	0.94135 (0.531)	0.93994 (0.501)	0.94037 (0.481)	0.93737 (0.454)
GMF Softmax	0.87614 (0.914)	0.90952 (0.767)	**0.92424** **(0.676)**	**0.92645** **(0.609)**	**0.94188** **(0.567)**	**0.93885** **(0.53)**	**0.93752** **(0.5)**	**0.93823** **(0.48)**	**0.93561** **(0.454)**
MLP IPA	0.94301 (0.982)	0.96557 (0.814)	0.96809 (0.712)	0.96499 (0.635)	0.97727 (0.591)	0.96889 (0.55)	0.96634 (0.518)	0.96648 (0.498)	0.96187 (0.47)
MLP Avg DeepGroup	0.99415 (1.037)	1.03182 (0.875)	1.04278 (0.779)	1.04242 (0.695)	1.05597 (0.651)	1.05056 (0.605)	1.04927 (0.574)	1.04836 (0.552)	1.04669 (0.524)
MLP Expertise	0.9986 (1.038)	1.03522 (0.88)	1.04511 (0.78)	1.04435 (0.696)	1.05806 (0.651)	1.05121 (0.604)	1.05084 (0.575)	1.0491 (0.553)	1.04787 (0.523)
MLP Softmax	0.99487 (1.035)	1.03145 (0.874)	1.04258 (0.779)	1.04235 (0.694)	1.05605 (0.651)	1.05034 (0.604)	1.04925 (0.574)	1.04822 (0.552)	1.04659 (0.524)

(a) MovieLens1M

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	0.67941 (1.077)	0.7889 (0.688)	0.82014 (0.608)	0.82302 (0.544)	0.82574 (0.502)	0.83038 (0.469)	0.8324 (0.44)	0.8375 (0.42)	0.83544 (0.401)
GMF Expertise	0.67314 (1.05)	0.79105 (0.686)	0.82229 (0.608)	0.82546 (0.543)	0.82758 (0.501)	0.83205 (0.468)	0.83382 (0.439)	0.83875 (0.418)	0.83673 (0.399)
GMF Softmax	0.67596 (1.063)	0.7892 (0.687)	0.82043 (0.608)	0.82333 (0.544)	0.82592 (0.502)	0.83053 (0.469)	0.83251 (0.44)	0.83758 (0.42)	0.83552 (0.401)
MLP IPA	**0.61169** **(1.002)**	0.77197 (0.683)	0.79514 (0.592)	0.7988 (0.528)	0.80315 (0.487)	0.80807 (0.454)	0.81084 (0.427)	0.81611 (0.406)	0.81408 (0.388)
MLP Avg DeepGroup	0.61859 (1.009)	**0.7636** **(0.674)**	**0.78665** **(0.585)**	**0.79085** **(0.523)**	**0.79606** **(0.484)**	**0.80126** **(0.452)**	**0.80444** **(0.425)**	**0.81023** **(0.405)**	**0.80841** **(0.387)**
MLP Expertise	0.62484 (0.979)	0.7691 (0.68)	0.78956 (0.586)	0.79332 (0.522)	0.79799 (0.483)	0.80305 (0.45)	0.80595 (0.423)	0.81166 (0.403)	0.80977 (0.385)
MLP Softmax	0.62106 (0.992)	0.7649 (0.675)	0.78709 (0.585)	0.79116 (0.523)	0.79625 (0.484)	0.80142 (0.452)	0.80456 (0.424)	0.81033 (0.404)	0.8085 (0.387)

(b) FilmTrust

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	**1.47037** **(1.922)**	**1.53981** **(1.648)**	**1.58373** **(1.43)**	**1.63441** **(1.324)**	**1.66233** **(1.213)**	**1.70615** **(1.178)**	**1.72188** **(1.111)**	**1.73177** **(1.065)**	**1.74048** **(1.015)**
GMF Expertise	1.49874 (1.963)	1.57277 (1.689)	1.61625 (1.463)	1.66444 (1.351)	1.69199 (1.239)	1.73293 (1.202)	1.74812 (1.133)	1.75649 (1.087)	1.76487 (1.035)
GMF Softmax	1.47707 (1.932)	1.54383 (1.654)	1.58617 (1.433)	1.63575 (1.326)	1.66342 (1.214)	1.70677 (1.179)	1.72239 (1.112)	1.73223 (1.066)	1.74092 (1.016)
MLP IPA	1.56623 (1.959)	1.61737 (1.655)	1.64909 (1.431)	1.69132 (1.32)	1.71029 (1.209)	1.74683 (1.169)	1.75711 (1.105)	1.76368 (1.058)	1.76877 (1.008)
MLP Avg DeepGroup	1.61669 (2.032)	1.68103 (1.718)	1.71103 (1.492)	1.75681 (1.368)	1.77828 (1.254)	1.81759 (1.217)	1.82764 (1.148)	1.83403 (1.098)	1.84177 (1.049)
MLP Expertise	1.64412 (2.059)	1.70965 (1.747)	1.74467 (1.517)	1.78446 (1.386)	1.80671 (1.276)	1.84221 (1.232)	1.85095 (1.163)	1.85803 (1.116)	1.86351 (1.063)
MLP Softmax	1.62458 (2.038)	1.68445 (1.723)	1.7153 (1.496)	1.75838 (1.368)	1.78024 (1.256)	1.81848 (1.218)	1.82855 (1.149)	1.83446 (1.099)	1.84249 (1.049)

(c) MyAnimeList

Table 5 Mean max error

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	**1.02112** **(0.6)**	**1.2213** **(0.598)**	1.35658 (0.588)	**1.45195** **(0.583)**	**1.54115** **(0.578)**	1.59908 (0.577)	1.64979 (0.573)	1.69807 (0.576)	1.73598 (0.571)
GMF Expertise	1.02474 (0.602)	1.22441 (0.599)	1.35928 (0.59)	1.4551 (0.584)	1.54414 (0.579)	1.60112 (0.577)	1.65118 (0.574)	1.69926 (0.576)	1.7366 (0.571)
GMF Softmax	1.02191 (0.6)	1.22143 (0.598)	**1.35653** **(0.588)**	1.45202 (0.583)	1.54118 (0.578)	**1.59897** **(0.577)**	**1.64965** **(0.573)**	**1.69792** **(0.576)**	**1.7358** **(0.571)**
MLP IPA	1.04098 (0.642)	1.25715 (0.614)	1.38214 (0.602)	1.47972 (0.591)	1.56516 (0.586)	1.62067 (0.581)	1.66905 (0.576)	1.71687 (0.58)	1.7528 (0.573)
MLP Avg DeepGroup	1.07144 (0.666)	1.28743 (0.643)	1.41937 (0.635)	1.51234 (0.633)	1.59812 (0.638)	1.65421 (0.642)	1.70673 (0.641)	1.75552 (0.647)	1.79703 (0.645)
MLP Expertise	1.07476 (0.667)	1.28932 (0.644)	1.42078 (0.635)	1.51325 (0.634)	1.59883 (0.638)	1.65371 (0.64)	1.70637 (0.64)	1.75428 (0.645)	1.79566 (0.643)
MLP Softmax	1.07226 (0.666)	1.28716 (0.643)	1.41906 (0.635)	1.51227 (0.633)	1.59788 (0.638)	1.65391 (0.641)	1.7066 (0.641)	1.75515 (0.646)	1.79669 (0.645)

(a) MovieLens1M

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	0.85192 (0.555)	1.13757 (0.573)	1.27696 (0.573)	1.36783 (0.57)	1.44244 (0.571)	1.51262 (0.575)	1.56818 (0.573)	1.62204 (0.571)	1.6636 (0.567)
GMF Expertise	0.85171 (0.549)	1.13845 (0.571)	1.27815 (0.571)	1.36895 (0.567)	1.4425 (0.569)	1.51199 (0.573)	1.56717 (0.57)	1.62058 (0.569)	1.66146 (0.565)
GMF Softmax	0.85147 (0.552)	1.13761 (0.572)	1.27708 (0.572)	1.3679 (0.569)	1.44234 (0.571)	1.51245 (0.575)	1.56799 (0.572)	1.62184 (0.571)	1.66335 (0.567)
MLP IPA	**0.78405** **(0.555)**	1.11519 (0.564)	1.24935 (0.56)	1.33899 (0.555)	1.41348 (0.556)	1.48001 (0.557)	1.5341 (0.554)	1.58622 (0.552)	1.62791 (0.549)
MLP Avg DeepGroup	0.79205 (0.553)	**1.11222** **(0.557)**	**1.24695** **(0.557)**	1.33662 (0.555)	1.4118 (0.557)	1.47902 (0.559)	1.53438 (0.556)	1.58697 (0.555)	1.62916 (0.55)
MLP Expertise	0.79191 (0.56)	1.11441 (0.556)	1.24814 (0.555)	1.33715 (0.551)	**1.41142** **(0.554)**	**1.47765** **(0.556)**	**1.53235** **(0.553)**	**1.58432** **(0.551)**	**1.62581** **(0.547)**
MLP Softmax	0.79235 (0.555)	1.11261 (0.556)	1.24698 (0.556)	**1.33654** **(0.554)**	1.41161 (0.556)	1.47872 (0.558)	1.53406 (0.555)	1.58662 (0.554)	1.62879 (0.55)

(b) FilmTrust

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	**1.30264** **(0.845)**	**1.58715** **(0.866)**	**1.79755** **(0.864)**	**1.9702** **(0.877)**	**2.10648** **(0.876)**	**2.22836** **(0.896)**	**2.31758** **(0.895)**	**2.39541** **(0.91)**	**2.47078** **(0.919)**
GMF Expertise	1.31784 (0.858)	1.60964 (0.883)	1.8266 (0.882)	1.99902 (0.894)	2.13631 (0.891)	2.25748 (0.91)	2.34706 (0.908)	2.42357 (0.922)	2.49875 (0.932)
GMF Softmax	1.30626 (0.847)	1.59061 (0.869)	1.80139 (0.867)	1.97309 (0.879)	2.10896 (0.878)	2.23046 (0.898)	2.31946 (0.896)	2.39701 (0.911)	2.47222 (0.92)
MLP IPA	1.34151 (0.874)	1.63805 (0.873)	1.84317 (0.862)	2.01188 (0.868)	2.1406 (0.863)	2.25649 (0.876)	2.33763 (0.875)	2.4112 (0.889)	2.4836 (0.896)
MLP Avg DeepGroup	1.36617 (0.893)	1.66385 (0.902)	1.8675 (0.896)	2.03818 (0.907)	2.17095 (0.906)	2.28936 (0.924)	2.37588 (0.921)	2.4495 (0.934)	2.52558 (0.943)
MLP Expertise	1.38038 (0.9)	1.68165 (0.913)	1.89237 (0.908)	2.0611 (0.918)	2.1951 (0.914)	2.31311 (0.929)	2.39796 (0.923)	2.47081 (0.935)	2.54585 (0.943)
MLP Softmax	1.37076 (0.894)	1.66676 (0.904)	1.87147 (0.898)	2.04076 (0.908)	2.17377 (0.907)	2.29111 (0.924)	2.37734 (0.921)	2.45073 (0.934)	2.52696 (0.943)

(c) MyAnimeList

Table 6 Discounted cumulative gain

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	**0.98021** **(0.024)**	**0.98543** **(0.018)**	0.9886 (0.015)	**0.99067** **(0.012)**	0.99178 (0.011)	0.99286 (0.01)	0.99358 (0.009)	**0.99415** **(0.008)**	**0.99457** **(0.008)**
GMF Expertise	0.97995 (0.024)	0.98528 (0.019)	0.98853 (0.015)	0.99058 (0.012)	0.99171 (0.011)	0.99282 (0.01)	0.99351 (0.009)	0.99411 (0.008)	0.99454 (0.008)
GMF Softmax	0.98006 (0.024)	0.98536 (0.019)	**0.9886** **(0.015)**	0.99064 (0.012)	**0.99178** **(0.011)**	**0.99288** **(0.01)**	**0.9936** **(0.009)**	0.99415 (0.008)	0.99456 (0.008)
MLP IPA	0.97854 (0.026)	0.98342 (0.02)	0.98689 (0.016)	0.98906 (0.014)	0.99046 (0.012)	0.99178 (0.011)	0.99251 (0.01)	0.99303 (0.009)	0.99358 (0.009)
MLP Avg DeepGroup	0.97778 (0.026)	0.98247 (0.021)	0.98556 (0.017)	0.98762 (0.015)	0.98894 (0.014)	0.98999 (0.013)	0.99064 (0.012)	0.99109 (0.011)	0.99152 (0.011)
MLP Expertise	0.97777 (0.026)	0.98251 (0.021)	0.9855 (0.017)	0.98763 (0.015)	0.98894 (0.014)	0.99004 (0.013)	0.99071 (0.012)	0.99116 (0.011)	0.9916 (0.011)
MLP Softmax	0.97784 (0.026)	0.98248 (0.021)	0.98554 (0.017)	0.98762 (0.015)	0.98895 (0.014)	0.98998 (0.013)	0.99069 (0.012)	0.9911 (0.011)	0.99153 (0.011)

(a) MovieLens1M

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	0.96788 (0.028)	0.97795 (0.022)	0.9815 (0.019)	0.98393 (0.016)	0.98596 (0.014)	0.9876 (0.013)	0.98869 (0.011)	0.98967 (0.011)	0.99038 (0.01)
GMF Expertise	0.96795 (0.028)	0.97794 (0.022)	0.9815 (0.019)	0.98392 (0.016)	0.98593 (0.014)	0.9876 (0.013)	0.98868 (0.012)	0.98965 (0.011)	0.99038 (0.01)
GMF Softmax	0.96795 (0.028)	0.97796 (0.022)	0.9815 (0.019)	0.98394 (0.016)	0.98596 (0.014)	0.9876 (0.013)	0.98868 (0.012)	0.98967 (0.011)	0.99038 (0.01)
MLP IPA	**0.97057** **(0.027)**	0.9788 (0.023)	0.98249 (0.019)	0.98523 (0.016)	0.98704 (0.015)	**0.98896** **(0.012)**	0.98994 (0.011)	**0.9911** **(0.01)**	0.99165 (0.01)
MLP Avg DeepGroup	0.96897 (0.027)	0.9789 (0.022)	0.98275 (0.018)	0.98527 (0.016)	**0.98729** **(0.014)**	0.98895 (0.012)	**0.98998** **(0.011)**	0.99104 (0.01)	0.99166 (0.009)
MLP Expertise	0.9694 (0.027)	**0.97897** **(0.022)**	0.98276 (0.018)	**0.98532** **(0.016)**	0.98724 (0.014)	0.98893 (0.012)	0.98993 (0.011)	0.99106 (0.01)	0.99166 (0.009)
MLP Softmax	0.9693 (0.027)	0.97891 (0.022)	**0.98278** **(0.018)**	0.98527 (0.016)	0.98728 (0.014)	0.98895 (0.012)	0.98998 (0.011)	0.99106 (0.01)	**0.99167** **(0.009)**

(b) FilmTrust

Model \Group Size	2	3	4	5	6	7	8	9	10
GMF IPA GMF Avg	0.98898 (0.014)	**0.99218** **(0.01)**	**0.99401** **(0.008)**	0.9949 (0.007)	0.99571 (0.006)	**0.99615** **(0.005)**	0.99662 (0.005)	0.99679 (0.005)	0.99714 (0.004)
GMF Expertise	0.98893 (0.014)	0.99209 (0.01)	0.99383 (0.008)	0.99479 (0.007)	0.99566 (0.006)	0.99609 (0.005)	0.99658 (0.005)	0.99674 (0.005)	0.99708 (0.004)
GMF Softmax	**0.98898** **(0.014)**	0.99217 (0.01)	0.99399 (0.008)	**0.99492** **(0.007)**	**0.99572** **(0.006)**	0.99615 (0.005)	**0.99664** **(0.005)**	**0.99679** **(0.005)**	**0.99715** **(0.004)**
MLP IPA	0.98723 (0.015)	0.99062 (0.011)	0.99255 (0.009)	0.9936 (0.008)	0.99458 (0.007)	0.99507 (0.006)	0.99573 (0.006)	0.99591 (0.005)	0.99632 (0.005)
MLP Avg DeepGroup	0.9868 (0.016)	0.99023 (0.012)	0.99212 (0.01)	0.99307 (0.008)	0.99415 (0.007)	0.99459 (0.007)	0.99519 (0.006)	0.99537 (0.006)	0.99565 (0.005)
MLP Expertise	0.9867 (0.016)	0.99021 (0.012)	0.9921 (0.01)	0.99312 (0.008)	0.99414 (0.007)	0.99458 (0.007)	0.99521 (0.006)	0.99542 (0.006)	0.9957 (0.005)
MLP Softmax	0.98684 (0.016)	0.99028 (0.012)	0.99211 (0.01)	0.99308 (0.008)	0.99416 (0.007)	0.9946 (0.007)	0.99521 (0.006)	0.99539 (0.006)	0.99566 (0.005)

(c) MyAnimeList

i for the group G, and $\bar{r}_{G,i}$ is the average rating of the users belonging to the group G for the item i.

We can see the results of the experiment executed with these scores in Table 3 (MAE), Table 4 (MSE), Table 5 (Max), and Table 6 (NDCG). The cells with the best results

have been highlighted (gray background and bold), while the standard deviation of each metric is in parentheses. All results are analyzed in Sect. 4.

All experiments have been run using an NVIDIA Quadro RTX 8000 GPU with 48 GB GDDR6 of memory, 4,608 NVIDIA Tensor Cores and a performance of 16.3 TFLOPS. We are committed to reproducible science, so the source code of all experiments with the values of the parameters used and their random seeds have been shared on GitHub.[1]

4 Discussion

The main goal of this research is to evaluate different aggregation techniques to make recommendations to groups. As shown in Sect. 3, we can see different trends according to (a) the models used; (b) the way group information is aggregated; (c) the datasets on which they act; and (d) the size of the groups.

Focusing on the models, we can see how Multi-Layer Perceptron (MLP), which has several hidden layers, obtains a lower MAE; however, Generalized Matrix Factorization (GMF), a simpler model, obtains a lower MSE. Although the Multi-Layer Perceptron (MLP) model has great power in these types of problem, it seems to overfit, generating very good recommendations for some users in the group but bad ones for the rest, hence achieving higher MSE values. On the other hand, the Generalized Matrix Factorization (GMF) model can obtain smaller maximum errors in each group, which means that no user in the group is badly affected by the recommendation. In the results, we can also observe how the models with higher maximum errors lead to a poorer order of items according to user preferences and obtain worse performance in the Normalizaed Discounted Cumulative Gain (NDCG) metric.

Looking at the aggregation of users, we can see that the best performing user aggregation is the average, followed by a very similar performance by the Softmax. However, the use of expert user weighting without softmax produces worse results. Based on the results, we can observe that in models that do not use a deep architecture, with several hidden layers, the Individual Preference Aggregation (IPA) and Group Preference Aggregation (GPA) strategies produce similar results when the aggregation function is a linear transformation of latent factors (Generalized Matrix Factorization (GMF)). However, we can see how the nonlinearity of Multi-Layer Perceptron (MLP) produces different results between both two strategies.

Regarding the different datasets, we can see that there is a clear trend in the models that achieve the best results in complex datasets with a large number of users, items, and votes, such as Movilens or MyAnimeList, while in the FilmTrust dataset, with a smaller number of votes, there is no clear trend.

In terms of group size, as more users have the group, the probability of finding discrepancies between user preferences increases. Therefore, we can see how a larger group size leads to higher values in all error metrics.

5 Conclusions and future work

With the irruption of Neural Network (NN) in the world of Collaborative Filtering (CF), the possibilities of their ability to find nonlinear patterns within user preferences to generate better predictions are opening up. To use these systems to generate a recommendation for a group of users, we need to aggregate their preferences. As we have seen in this research, there are several key points at which aggregation can be performed. Group Preference Aggregation (GPA) strategies do the aggregation before or inside the model, so they have the advantage of taking into account the preferences of the entire group and that a single feed-forward step generates the prediction. In contrast, the Individual Preference Aggregation (IPA) strategy, requires multiple predictions for each user and performs the aggregations after the model. In this study, we have tested how different approaches to perform Group Preference Aggregation (GPA) work in different datasets comparing different metrics.

As future work, there are two key factors to consider. First, in this research, the researchers have designed user aggregation techniques presented to the models; in future work, these functions will be explored by different machine learning models. The second key point is that in this work, models perform a knowledge transfer from the model trained for individuals to make group predictions; it is shown that although the models have high performance (MAE improvement), they tend to overfit when working in groups (larger errors in group prediction leading to worse MSE). To solve this problem, future work will try to perform a specialization training stage for groups after individual training.

Funding This work has been co-funded by the *Ministerio de Ciencia e Innovación* of Spain and the European Regional Development Fund (FEDER) under grants PID2019-106493RB-I00 (DL-CEMG) and the *Comunidad de Madrid* under *Convenio Plurianual* with the *Universidad Politécnica de Madrid* in the actuation line of *Programa de Excelencia para el Profesorado Universitario*.

Data availability The `MovieLens1M`, `FilmTrust` and `MyAnimeList` dataset along with the source code of the experiments that support the findings of this study is available in `neural-cf-for-`

[1] https://github.com/KNODIS-Research-Group/neural-cf-for-groups

groups GitHub's repository [https://github.com/KNODIS-Research-Group/neural-cf-for-groups].

Declarations

Conflict of interest The authors of this paper declare that they have no conflict of interest.

References

1. Batmaz Z, Yurekli A, Bilge A, Kaleli C (2019) A review on deep learning for recommender systems: challenges and remedies. Artif Intell Rev 52(1):1–37. https://doi.org/10.1007/s10462-018-9654-y
2. Bobadilla J, González-Prieto Á, Ortega F, Lara-Cabrera R (2021) Deep learning feature selection to unhide demographic recommender systems factors. Neural Comput Appl 33(12):7291–7308. https://doi.org/10.1007/s00521-020-05494-2
3. Deldjoo Y, Schedl M, Cremonesi P, Pasi G (2020) Recommender systems leveraging multimedia content. ACM Comput Surv 53(5):1–38. https://doi.org/10.1145/3407190
4. Kulkarni S, Rodd SF (2020) Context aware recommendation systems: a review of the state of the art techniques. Comput Sci Rev 37:100255. https://doi.org/10.1016/j.cosrev.2020.100255
5. Shokeen J, Rana C (2020) A study on features of social recommender systems. Artif Intell Rev 53(2):965–988. https://doi.org/10.1007/s10462-019-09684-w
6. Bobadilla J, Alonso S, Hernando A (2020) Deep learning architecture for collaborative filtering recommender systems. Appl Sci 10(7):2441. https://doi.org/10.3390/app10072441
7. Dara S, Chowdary CR, Kumar C (2020) A survey on group recommender systems. J Intell Inf Syst 54(2):271–295. https://doi.org/10.1007/s10844-018-0542-3
8. Forouzandeh S, Berahmand K, Rostami M (2021) Presentation of a recommender system with ensemble learning and graph embedding: a case on movielens. Multimed Tools Appl 80(5):7805–7832. https://doi.org/10.1007/s11042-020-09949-5
9. Çano E, Morisio M (2017) Hybrid recommender systems: a systematic literature review. Intell Data Anal 21(6):1487–1524. https://doi.org/10.3233/IDA-163209
10. Salakhutdinov R, Mnih A (2007) Probabilistic matrix factorization. In: Proceedings of the 20th international conference on neural information processing systems. NIPS'07, pp. 1257–1264. Curran Associates Inc., Red Hook, NY, USA. https://doi.org/10.5555/2981562.2981720
11. Bobadilla J, González-Prieto Á, Ortega F, Lara-Cabrera R (2022) Deep learning approach to obtain collaborative filtering neighborhoods. Neural Comput Appl 34(4):2939–2951. https://doi.org/10.1007/s00521-021-06493-7
12. Huang T, Zhang D-f, Bi L (2020) Neural embedding collaborative filtering for recommender systems. Neural Comput Appl 32:1–15. https://doi.org/10.1007/s00521-020-04920-9
13. He X, Liao L, Zhang H, Nie L, Hu X, Chua T-S (2017) Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web. WWW '17, pp. 173–182. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE. https://doi.org/10.1145/3038912.3052569
14. Ortega F, Bobadilla J, Hernando A, GutiéRrez A (2013) Incorporating group recommendations to recommender systems: alternatives and performance. Inf Process Manage 49(4):895–901. https://doi.org/10.1016/j.ipm.2013.02.003
15. Baltrunas L, Makcinskas T, Ricci F (2010) Group recommendations with rank aggregation and collaborative filtering. In: Proceedings of the fourth acm conference on recommender systems. RecSys '10, pp. 119–126. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1864708.1864733
16. Ortega F, Hernando A, Bobadilla J, Kang JH (2016) Recommending items to group of users using matrix factorization based collaborative filtering. Inf Sci 345:313–324. https://doi.org/10.1016/j.ins.2016.01.083
17. Feng S, Zhang H, Wang L, Liu L, Xu Y (2019) Detecting the latent associations hidden in multi-source information for better group recommendation. Know-Based Syst 171:56–68. https://doi.org/10.1016/j.knosys.2019.02.002
18. Abolghasemi R, Engelstad P, Herrera-Viedma E, Yazidi A (2022) A personality-aware group recommendation system based on pairwise preferences. Inf Sci 595:1–17. https://doi.org/10.1016/j.ins.2022.02.033
19. Barzegar Nozari R, Koohi H (2020) A novel group recommender system based on members' influence and leader impact. Know-Based Syst 205:106296. https://doi.org/10.1016/j.knosys.2020.106296
20. Wang X, Su L, Zhou Q, Wu L, Zhang Y (2020) Group recommender systems based on members' preference for trusted social networks. Sec Commun Netw 2020:1–11. https://doi.org/10.1155/2020/1924140
21. Ismailoglu F (2022) Aggregating user preferences in group recommender systems: a crowdsourcing approach. Decis Support Syst 152:113663. https://doi.org/10.1016/j.dss.2021.113663
22. Guo J, Zhu Y, Li A, Wang Q, Han W (2016) A social influence approach for group user modeling in group recommendation systems. IEEE Intell Syst 31(5):40–48. https://doi.org/10.1109/MIS.2016.28
23. Sajjadi Ghaemmaghami S, Salehi-Abari A (2021) DeepGroup: group recommendation with implicit feedback. Association for Computing Machinery, New York, pp 3408–3412
24. Hu L, Cao J, Xu G, Cao L, Gu Z, Cao W (2014) Deep modeling of group preferences for group-based recommendation. In: Proceedings of the twenty-eighth AAAI conference on artificial intelligence. AAAI'14, pp. 1861–1867. AAAI Press, Palo Alto, California. https://doi.org/10.1609/aaai.v28i1.9007
25. Harper FM, Konstan JA (2015) The movielens datasets: history and context. ACM Trans Interact Intell Syst 5(4):1–19. https://doi.org/10.1145/2827872
26. Guo G, Zhang J, Yorke-Smith N (2013) A novel bayesian similarity measure for recommender systems. In: Proceedings of the twenty-third international joint conference on artificial intelligence. IJCAI '13, pp. 2619–2625. AAAI Press, Menlo Park, California. https://doi.org/10.5555/2540128.2540506