SANS: Setwise Attentional Neural Similarity Method for Few-shot Recommendation

Zhenghao Zhang[1,2], Tun Lu[1,2(✉)], Dongsheng Li[3], Peng Zhang[1,2(✉)], Hansu Gu[4], and Ning Gu[1,2]

[1] School of Computer Science, Fudan University, Shanghai, China
[2] Shanghai Key Laboratory of Data Science, Fudan University, Shanghai, China
{zhzhang18,lutun,zhangpeng_,ninggu}@fudan.edu.cn
[3] Microsoft Research Asia, Shanghai, China
dongshengli@fudan.edu.cn
[4] Amazon.com, Seattle, USA
guhansu@gmail.com

Abstract. Recommender systems generate personalized recommendations for users based on their historical data. However, if some users have few interactions in the training data, i.e., few-shot users, recommendations for them will be inaccurate. In this paper, we propose a setwise attentional neural similarity method (SANS) for the few-shot recommendation problem. Unlike general recommendation algorithms, we eliminate direct representations of few-shot users. First, a neural similarity method is proposed to effectively estimate the correlation between items. Then, we propose a setwise attention mechanism to obtain recommendation scores by aggregating the correlations between a candidate item and items in a candidate user's historical interactions. To facilitate model training in the few-shot scenario, training samples are generated by episode sampling, and each training sample is assigned with an adaptive weight to emphasize the importance of few-shot users. We simulate the few-shot recommendation problem on three real-world datasets and extensive results show that SANS can outperform the state-of-the-art recommendation algorithms in few-shot recommendation.

Keywords: Collaborative Filtering · Few-shot Learning · Neural Networks · Top-N Recommendation

1 Introduction

Recommender systems recommend items to users based on their historical interactions with other items. However, in practice, there are many newcomers or inactive users who have few interactions in online services, i.e., few-shot users, which makes it challenging to train accurate recommendation models for them. Due to the long-tail distribution of user activities in online services, these few-shot users are non-negligible and it is desirable to deliver high quality recommendations for these few-shot users.

Existing general-purpose recommendation algorithms cannot well address the few-shot recommendation problem. In many recent recommendation algorithms, especially deep learning-based ones [2, 6], large number of training data are required in model learning, e.g., learn the embedding vectors of users and items. However, if a user has very few interactions, then the embedding vector of this user cannot be well learned, resulting in poor recommendation performance. For instance, many deep neural network-based methods adopt deep neural networks to learn the representations of users/items and capture the complex non-linear relationships among user/items, which may suffer from severe overfitting issue in the few-shot recommendation scenario. Item-based recommendation algorithms, such as FISM [8] and NAIS [5], can eliminate the overfitting issue on user modeling. However, how to effectively estimate the correlation between items (e.g., similarity) in the few-shot scenario is still an open question.

In this paper, we propose SANS, an item-based deep recommendation algorithm for few-shot recommendation. In SANS, we do not explicitly learn the representations of users like many existing works but represent each user by the set of items in his/her historical interactions. A neural similarity method is proposed to estimate the correlation between each pair of items, i.e., the probability that a user who likes one of the items will also like the other one. When recommending items for a user, we propose a setwise attention method which utilizes items in the user's historical interactions as a support set, estimates importance of each item in the support set via attention mechanism, and finally aggregates the correlations between the candidate item and the support set to generate the final prediction. To facilitate model training for few-shot users, we generate training samples by episode sampling and propose a new weighted loss function in which each training sample is assigned with an adaptive weight to emphasize the importance of few-shot users.

In summary, the main contributions of this paper are as follows:

- We propose SANS, a deep recommendation algorithm consisting of a neural similarity module and a setwise attention module to address the few-shot recommendation problem.
- We design a weighted loss function in which weights are adaptively assigned according to the number of user's interactions. Combined with episode sampling, the training of SANS is highly effective.
- We conduct extensive experiments on three real-world datasets, which demonstrate that SANS can substantially outperform state-of-the-art recommendation algorithms on few-shot users.

The rest of this paper is organized as following: Section 2 defines the few-shot recommendation problem and introduces item-based collaborative filtering. Section 3 proposes the network architecture of SANS, the weighted loss and its training procedure. Section 4 presents experimental results. Section 5 discusses the related work about collaborative filtering, cold-start recommendation and few-shot learning. Finally Section 6 concludes the paper.

2 Preliminaries

This section first defines the few-shot recommendation problem and then introduces the item-based collaborative filtering algorithm.

2.1 Few-shot Recommendation

Let U and I donate the set of users and items, respectively. The training set S consists of the user-item tuples $S = \{(u, i) : u \in U, i \in I\}$. We define the set of items interacted by user u as $I_u^+ = \{i \in I : (u, i) \in S\}$. For each user-item tuple,

$$y_{ui} = \begin{cases} 1, & (u, i) \in S \\ 0, & (u, i) \notin S \end{cases}, \tag{1}$$

where y_{ui}=0 means the interaction between user u and item i hasn't been observed. Then, we formulate the problem of few-shot recommendation as follows.

Definition 1. *(N-shot Recommendation) For a subset of users U^*, if $\forall u \in U^*$ satisfies $|I_u^+| = N$ $(N > 0)$, the problem of recommending items to all users within U^* is N-shot recommendation.*

When N is small, e.g, 3 or 5, we can call it few-shot recommendation. There is a related problem in previous recommender system research called cold-start [14]. However, the difference between few-shot and cold-start is that few-shot users have no additional personal information other than few interactions. Few-shot users are special cold-start users.

2.2 Item-based Collaborative Filtering

The item-based collaborative filtering method [13] uses similarities between candidate items and users' historical items to rank candidate items. For a user u with historical interactions I_u^+, the predicted score of user u on item i under the implicit feedback setting is:

$$\hat{y}_{ui} \propto \sum_{j \in I_u^+} a_{uj} s_{ij}, \tag{2}$$

where s_{ij} denotes the similarity between item i and item j. The similarity can be computed using different metrics such as cosine [12], Pearson [13], etc., or learned from data [5, 8]. a_{uj} donates the preference of user u on item j when predicting u's preference on i. a_{uj} is usually set to 1 in existing methods, i.e., all historical interactions are equally important on predicting u's preference on i.

The item-based collaborative filtering is well suited for few-shot recommendation due to two reasons: 1) users are not explicitly modeled. Since there are no user-related model parameters, the difficulty in training user models in few-shot scenario doesn't exist; 2) new users with few interactions can also have recommendations, which is ideal for online services with newcomers everyday.

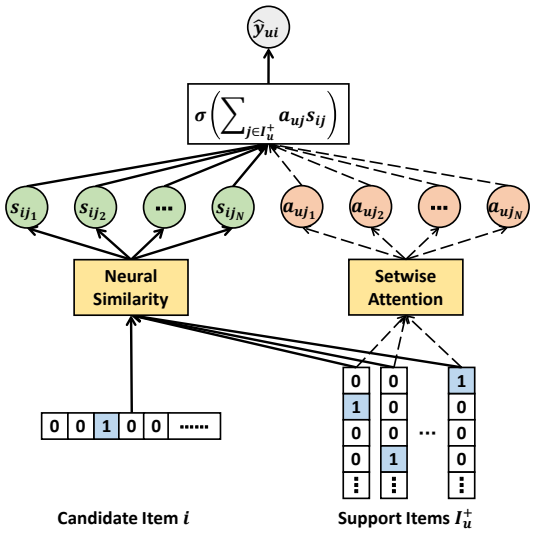


Fig. 1. The network architecture of SANS.

3 SANS: Setwise Attentional Neural Similarity Method

3.1 Algorithm Design

The Overall Architecture As illustrated in Fig. 1, where $I_u^+ = \{j_1, j_2, \ldots, j_N\}$, SANS consists of two main components: 1) a neural similarity module, which estimates the correlation of two items using neural network, i.e., the similarity term in Equation 2. More specifically, the neural similarity module outputs the possibility of users who interacted with one of the items will also interact the other item; 2) a setwise attention module, which estimates the preference of a user over each item in his/her historical interactions, i.e., the preference term in Equation 2. More specifically, we assume that items from user history are not equally important in reflecting user preference, and we obtain the relative importance of different items via the proposed setwise attention module.

Neural Similarity If each user has only one interaction, then the recommendation problem becomes estimating the probability that a user u who likes item i also likes another item j: $\Pr(y_{uj} = 1 | y_{ui} = 1)$. Let $\Pr(y_{uj} = 1, y_{ui} = 1)$ donates the joint probability that two items are favored by a user. The score of personalized item ranking for the user u who has only one interaction with item i is computed as follows:

$$\Pr(y_{uj} = 1 | y_{ui} = 1) = \frac{\Pr(y_{uj} = 1, y_{ui} = 1)}{\Pr(y_{ui} = 1)} = \Pr(y_{uj} = 1, y_{ui} = 1), \quad (3)$$

where $\Pr(y_{ui} = 1) = 1$ since the item i has already been in the historical interactions of user u. $\Pr(y_{uj} = 1, y_{ui} = 1)$ is the probability of the co-occurrence

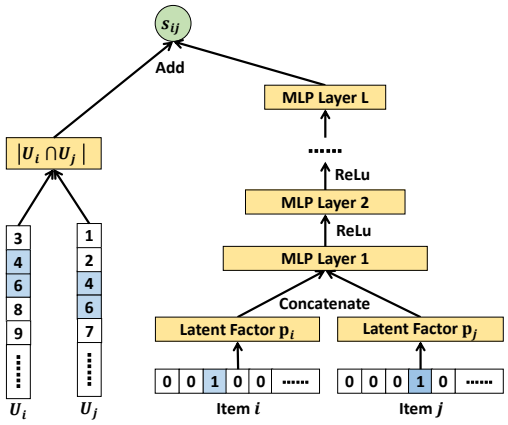


Fig. 2. The network architecture of the neural similarity module.

of item i and item j on user u, which can be approximated using the following equation:

$$\Pr(y_{uj} = 1, y_{ui} = 1) = \frac{|U_i^+ \cap U_j^+|}{|U|}, \tag{4}$$

where U_i/U_j is the set of users who interacted with item i/j. $|U|$ (the number of users) is a constant, which has no effect in ranking problems. Therefore, the probability that a user who likes one item also likes another item is proportional to the number of common users between two items:

$$\Pr(y_{uj} = 1 | y_{ui} = 1) \propto |U_i^+ \cap U_j^+|. \tag{5}$$

In addition, previous works [1, 2] have shown that multi-layer perceptron (MLP) can help to capture the high-order relationships between entities. Therefore, we combine the above two ideas and propose our neural similarity method as shown in Fig. 2.

More formally, the neural similarity between item i and item j is as follows:

$$s_{ij} = |U_i^+ \cap U_j^+| + f_\theta(\begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_j \end{bmatrix}), \tag{6}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are embedding vectors for item i and item j, respectively. They are concatenated together and then passed to an MLP f_Θ:

$$\begin{aligned} \mathbf{z}_1 &= \text{ReLU}(\mathbf{W}_1^T \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_j \end{bmatrix} + \mathbf{b}_1) \\ \mathbf{z}_2 &= \text{ReLU}(\mathbf{W}_2^T \mathbf{z}_1 + \mathbf{b}_2) \\ &\cdots \\ f_\theta(\begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_j \end{bmatrix}) &= \mathbf{W}_L^T \mathbf{z}_{L-1} + b_L \end{aligned}, \tag{7}$$

where $\mathbf{W}_l$ and $\mathbf{b}_l$ are the weight matrix and bias of the l-th layer. They are represented as $\mathbf{\Theta} = (\mathbf{W}_1, \ldots, \mathbf{W}_L, \mathbf{b}_1, \ldots, b_L)$. The output of the MLP is finally added to $|U_i^+ \cap U_j^+|$. The MLP can benefit the similarity learning due to two reasons: 1) it can capture high-order non-linear relationships between items in addition to the number of common users; 2) it can estimate the similarity between two items without common users via representation learning.

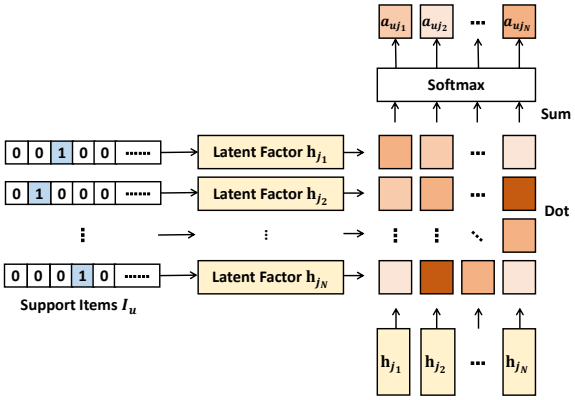


Fig. 3. The network architecture of the setwise attention module.

Setwise Attention In the previous section, we have modeled the neural similarity between items, based on which the probability that a user who has more than one interactions will like another item can be obtained based on the weighted sum overs the similarities between a set of support items and the candidate item:

$$\hat{y}_{ui} = \sigma\left(\sum_{j \in I_u^+} a_{uj} s_{ij}\right). \tag{8}$$

σ is the sigmoid function, which converts the weighted sum to a value between 0 and 1. The preference a_{uj} should be based on the set of items interacted by user u. However, it is challenging to introduce parameters related to user u in the few-shot recommendation problem. To this end, we propose the setwise attention method as illustrated in Fig. 3, which is formally described as follows:

$$a_{uj} = \frac{\exp(\sum_{k \in I_u^+} \mathbf{h}_k^T \mathbf{h}_j)}{\sum_{i' \in I_u^+} \exp(\sum_{k \in I_u^+} \mathbf{h}_k^T \mathbf{h}_{i'})}, \tag{9}$$

where $\mathbf{h}_k$, $\mathbf{h}_j$ and $\mathbf{h}_{i'}$ are embedding vectors of items. The setwise attention module estimates a user's preferences over items in his or her historical interactions. If there are a few highly similar items in a user's historical data, we can infer that this user really likes this type of items and we should emphasize more on

recommending items that are similar to these items. Otherwise, we know that the user has no preference differences among his/her historical items, so that it would be better to assign nearly equal weights to different items.

3.2 The Weighted Loss

For point-wise ranking problem, the binary cross-entropy loss function has been widely adopted. However, treating each training sample as equally important may not be optimal for model training [7]. Intuitively, users with many interactions contribute more to the total loss than users with few interactions. Thus, the model will converge when the users with many interactions are well trained but the few-shot users are usually underfitted due to fewer gradient updates. To remedy this, we propose a weighting mechanism to assign weights for each user based on the number of interacted items as follows:

$$c_u = c_0 \frac{|I_u^+|^\alpha}{\sum_{j \in U} |I_j^+|^\alpha}, \tag{10}$$

where c_0 sets the magnitude of weights and α controls the impact of the number of interactions on the weights. α is chosen in $[-1, 0]$. When $\alpha = 0$, all users have equal weights. When $\alpha = -1$, the weight is inversely proportional to the number of interactions.

The weights of users are then added into the binary cross-entropy loss forming the proposed weighted loss function for SANS as follows:

$$\begin{aligned} \mathcal{L} = & \sum_{(u,i,I_u^S,y_{ui}) \in \mathcal{D}} c_u(-y_{ui} \log \hat{y}_{uj} - (1 - y_{ui}) \log(1 - \hat{y}_{uj})) \\ & + \lambda_\Theta \|\mathbf{\Theta}\|^2 + \lambda_P \|\mathbf{P}\|^2 + \lambda_H \|\mathbf{H}\|^2, \end{aligned} \tag{11}$$

where I_u^S is a support set sampled from I_u^+, which mimics few-shot recommendation in each training sample. λ_Θ, λ_P and λ_H are L2 regularization coefficients.

3.3 Model Training

In SANS, we have three sets of model parameters: $\mathbf{\Theta}$, $\mathbf{P}$ and $\mathbf{H}$. $\mathbf{\Theta}$ and $\mathbf{P}$ are the parameters of the MLP and embedding matrix, respectively, in the neural similarity module. $\mathbf{H}$ is the embedding matrix for the setwise attention module.

Algorithm 1 presents the learning details. First, we calculate the weights for each user based on the number of interactions he/she has, then randomly initialize all parameters using Gaussian distribution. Training samples are dynamically generated in each iteration. For each user-item pair (u, i) in the dataset, a support set of items with size N is sampled from I_u^+, and a negative item is sampled from $I \setminus I_u^+$. Then, each quadruple—the user u, the positive item i or negative item j, the support set I_u^S, and the label y_{ui} is added to the collection of training samples. Finally, the SANS model is trained using training samples by the above episode sampling in each iteration, in which the model parameters can be updated by the stochastic gradient descent (SGD)-based methods.

Algorithm 1: LearnSANS

Data: Number of shots N, users U, items I, implicit feedback S, number of epoches T, weight magnitude c_0, weight strength α and regularization strength (λ_Θ, λ_P, λ_H)
Result: SANS model weights $(\mathbf{\Theta}, \mathbf{P}, \mathbf{H})$

```
foreach u ∈ U do
    c_u ← c_0 |I_u^+|^α / Σ_{j∈U} |I_j^+|^α;
Initialize (Θ, P, H) randomly;
for i ← 1 to T do
    D = ∅;
    foreach (u, i) ∈ S do
        Sample a support set I_u^S sized N from I_u^+;
        Sample an negative item j from I \ I_u^+;
        D ← D ∩ {(u, i, I_u^S, 1)};
        D ← D ∩ {(u, j, I_u^S, 0)};
    Train SANS using D with weighted loss (Equation 11);
```

4 Experiments

In this section, we compare SANS with state-of-the-art algorithms in few-shot recommendation scenario aiming to answer the following research questions:

- **RQ1**: how does SANS perform compared with state-of-the-art recommendation algorithms in few-shot recommendation?
- **RQ2**: how does each component in SANS affect the performance of the overall model?
- **RQ3**: how does the performance of SANS change with different few-shot scenarios?

4.1 Experimental Settings

Dataset We evaluate the proposed SANS method on three real-world datasets: Last.FM[1], Steam[2] and Douban[3], which are publicly available. The statistics of the three datasets are shown in Table 1.

- **Last.FM** was from HetRec 2011, which contains listening relationships between users and artists. The interactions from Last.FM are implicit.
- **Steam** was shared by Kaggle users, which contains purchase or play records of a set of Steam users. We convert all records into user-game tuples and remove duplicate ones.
- **Douban** was collect by Yin et al. [23], which contains ratings on books given by Douban users. The raw dataset is large, so we sample 5,000 users out of 383,033 users and treat all the ratings as implicit positive feedback.

[1] https://grouplens.org/datasets/hetrec-2011/
[2] https://www.kaggle.com/tamber/steam-video-games
[3] https://opendata.pku.edu.cn/dataset.xhtml?persistentId=doi:10.18170/DVN/LA9GRH

Table 1. Statists of the three datasets.

Dataset	#Interaction	#User	#Item	Sparsity
Last.FM	82,155	1,885	6,953	99.37%
Steam	103,594	2,189	3,933	98.80%
Douban	199,053	5,000	34,604	99.88%

Evaluation Protocols We use a user-based splitting method to split the dataset to simulate few-shot recommendation. First, we divide users into training users and test users by 8:2. All the interactions from training users are added to the training set. Then, for each user in the test user set, we sample N interactions from his or her interactions into the training set and put all the remaining interactions to the test set if this user has more than N interactions in his or her history. If the number of interactions from a user is less than N, we put this user back to the training set. We evaluate the recommendation performance by NDCG@10 [22] and Recall@10. For each user in the test set of Last.FM and Steam datasets, we rank all the items by predicated scores and exclude items that have already been interacted by the user in the training set to generate a top-10 list of recommended items. For the Douban dataset, we sampled 10,000 items and mix them with ground truth items as candidate items for each user. The NDCG@10 and Recall@10 values for that user can be calculated base on interactions in the test set. The performance on the entire dataset is reported by the average NDCG@10 and Recall@10 over all test users.

Compared Methods We compare SANS with various types of methods including baseline method, item-based methods, matrix factorization-based methods, deep learning-based methods and metric learning-based method as follows:

- **ItemPop** recommends top-N popular items to users. It is a non-personalized baseline method.
- **ALS** [7] uses point-wise loss and treats all unknown feedback as negative to learn the matrix factorization model. By taking advantage of its mathematical property, their matrix factorization model can be trained using all negative feedback.
- **BPR** [12] uses pair-wise loss and collects negative samples by sampling to learn the matrix factorization model.
- **KNN** [13] first computes the similarity between each pair of items and then sorts items by the sum of their similarities to the user's interactions. We use cosine similarity as the similarity metric in the experiments.
- **NAIS** [5] is an item-based collaborative filtering method which models item similarity by the dot product of two embedding vectors. Besides, attention mechanism is used to generate weights for different items.
- **NeuMF** [6] is a deep learning-based collaborative filtering method, which uses deep neural networks instead of a linear function to model the interactions between users and items.

- **CFNet** [2] combines representation learning-based collaborative filtering approach and matching function-based collaborative filtering approach using neural networks to achieve higher performance.
- **LRML** [20] is a collaborative filtering method based on metric learning which learns embeddings of users and items in a unified hyperspace. Items are ranked by the Euclidean distance to a user by assuming that users will prefer items that are close to them in the hyperspace.

The experiments are implemented using Tensorflow. We use the released code from the authors to implement the following compared methods: NAIS[4], NeuMF[5], CFNet[6] and LRML[7].

Hyperparameter Settings Hyperparameters of each method are tuned by the random search method. More specifically, all methods are tuned via a validation set and hyperparameters with the highest NDCG@10 are chosen as the optimal ones. We tuned the learning rates of all methods in $[0.001, 0.005, 0.01, 0.05]$ and the regularization coefficients in $[0.1, 0.01, 0.001]$. The optimal learning rates and regularization coefficients vary across datasets. For SANS, we fixed c_0 to 2,000 and tested the similarity embedding size of $[8, 16, 32, 64]$, the attention embedding size of $[4, 8, 12, 16]$ and the weight strength α of $[-0.5, -0.1, -0.05, -0.01]$. Finally, we set the dimension of the similarity embedding to 32, the dimension of the attention embedding to 4, and the weight strength α to -0.05. The architecture of MLP f_{Θ} is $64 \rightarrow 32 \rightarrow 16$. Deeper neural networks tend to achieve higher performance, but there is a law of diminishing marginal utility on the depth of networks. The SANS model is learned by the Adam optimizer [9].

4.2 Performance Comparison (RQ1)

We evaluate the performance of SANS and all the compared methods on three datasets with the number of shots increasing from 1 to 3 in Table 2. We have the following observations from the results:

- The performance of SANS is better than all other methods with N increasing from 1 to 3, which demonstrates the advantage of SANS on the few-shot recommendation.
- When N = 1, most of the compared methods have very low NDCG@10 and Recall@10, and some of them (e.g., BPR in Last.FM and CFNet in Steam) even perform worse than ItemPop. This indicates that existing methods indeed cannot well address the few-shot recommendation problem.
- Complex methods based on advanced techniques, e.g., matrix factorization and neural networks do not always exhibit higher performance in few-shot

[4] https://github.com/AaronHeee/Neural-Attentive-Item-Similarity-Model
[5] https://github.com/hexiangnan/neural_collaborative_filtering
[6] https://github.com/familyld/DeepCF
[7] https://github.com/cheungdaven/DeepRec

Table 2. Performance comparison between SANS and all compared methods in N-shot recommendation on three datasets. Relative improvements over the strongest baselines are also reported at the end of each table.

(a) Last.FM

Model	NDCG@10			Recall@10		
	N=1	N=2	N=3	N=1	N=2	N=3
ItemPop	0.2655	0.2600	0.2521	0.0548	0.0552	0.0548
BPR	0.2424	0.3016	0.3486	0.0510	0.0656	0.0762
ALS	0.3270	0.3659	0.3983	0.0698	0.0801	0.0890
KNN	0.2759	0.3361	0.3858	0.0631	0.0765	0.0885
NAIS	0.3392	0.3828	0.4045	0.0708	0.0815	0.0880
NeuMF	0.3940	0.4285	0.4558	0.0820	0.0922	0.1009
CFNet	0.2676	0.2610	0.2574	0.0554	0.0559	0.0565
LRML	0.3522	0.4061	0.4326	0.0738	0.0880	0.0963
SANS	**0.4205**	**0.4501**	**0.4839**	**0.0889**	**0.0978**	**0.1073**
Improvement	6.72%	5.05%	6.17%	8.36%	5.98%	6.32%

(b) Steam

Model	NDCG@10			Recall@10		
	N=1	N=2	N=3	N=1	N=2	N=3
ItemPop	0.3552	0.3432	0.3401	0.1210	0.1204	0.1226
BPR	0.3815	0.4380	0.4632	0.1567	0.1884	0.2100
ALS	0.3726	0.4402	0.4580	0.1442	0.1816	0.1976
KNN	0.3538	0.4551	0.4943	0.1411	0.1874	0.2127
NAIS	0.3891	0.4292	0.4461	0.1540	0.1769	0.1910
NeuMF	0.4361	0.4516	0.4551	0.1692	0.1874	0.2004
CFNet	0.3528	0.3390	0.3381	0.1175	0.1143	0.1184
LRML	0.4071	0.4377	0.4553	0.1460	0.1688	0.1976
SANS	**0.4806**	**0.5206**	**0.5223**	**0.1849**	**0.2102**	**0.2244**
Improvement	10.21%	14.30%	5.66%	9.29%	11.54%	5.50%

(c) Douban

Model	NDCG@10			Recall@10		
	N=1	N=2	N=3	N=1	N=2	N=3
ItemPop	0.0180	0.0169	0.0171	0.0055	0.0054	0.0057
BPR	0.0998	0.1187	0.1271	0.0276	0.0353	0.0383
ALS	0.0726	0.0838	0.0910	0.0199	0.0246	0.0272
KNN	0.0895	0.1245	0.1468	0.0245	0.0365	0.0436
NAIS	0.0462	0.0534	0.0602	0.0125	0.0150	0.0177
NeuMF	0.0803	0.0883	0.0939	0.0222	0.0262	0.0280
CFNet	0.0142	0.0138	0.0142	0.0041	0.0038	0.0043
LRML	0.0761	0.1017	0.1136	0.0213	0.0307	0.0350
SANS	**0.1161**	**0.1601**	**0.1841**	**0.0305**	**0.0450**	**0.0525**
Improvement	16.31%	28.52%	25.42%	10.28%	23.04%	20.43%

recommendation. For instance, KNN outperforms almost all compared methods in the Douban dataset (only except BPR with N=1). This indicates that complex models will easily overfit and be less desirable in the few-shot recommendation problems.
– SANS is more desirable due to: 1) overfitting will be less problematic because SANS does not learn user representations; 2) the proposed neural similarity method is more robust than conventional similarity methods due to the combination of simple and complex similarity modeling; 3) the proposed weighted loss function can emphasize few-shot users during model training, which can further alleviate inappropriate convergence on few-shot users.

4.3 Ablation Analysis (RQ2)

Here, we perform ablation analysis to investigate the impact of each component in SANS. The experimental results are shown in Table 3. SANS-SIM uses the linear part of neural similarity only, and SANS-MLP uses the MLP part of neural similarity, SANS-BCE replaces the weighted loss with binary cross-entropy loss, and SANS-AVG replaces setwise attention with the mean over neural similarities.

Table 3. Performance of SANS with each component removed in N-shot recommendation on the three datasets.

Dataset	Model	NDCG@10			Recall@10		
		N=1	N=2	N=3	N=1	N=2	N=3
Last.FM	SANS	**0.4205**	**0.4501**	**0.4839**	**0.0889**	**0.0978**	**0.1073**
	SANS-SIM	0.3703	0.4431	0.4732	0.0792	0.0967	0.1045
	SANS-MLP	0.3665	0.3809	0.3897	0.0763	0.0822	0.0862
	SANS-AVG	-	0.4443	0.4657	-	0.0964	0.1034
	SANS-BCE	0.4189	0.4477	0.4762	0.0886	0.0972	0.1053
Steam	SANS	**0.4806**	**0.5206**	**0.5223**	**0.1849**	**0.2102**	**0.2244**
	SANS-SIM	0.4592	0.4952	0.4943	0.1760	0.1957	0.2074
	SANS-MLP	0.4235	0.4283	0.4454	0.1613	0.1733	0.1895
	SANS-AVG	-	0.5085	0.5199	-	0.2080	0.2221
	SANS-BCE	0.4748	0.5114	0.5199	0.1818	0.2095	0.2209
Douban	SANS	**0.1161**	**0.1601**	**0.1841**	**0.0305**	0.0450	0.0525
	SANS-SIM	0.0826	0.1376	0.1684	0.0220	0.0390	0.0487
	SANS-MLP	0.0260	0.0346	0.0350	0.0073	0.0100	0.0099
	SANS-AVG	-	0.1592	0.1832	-	0.0449	0.0527
	SANS-BCE	0.1160	0.1596	0.1835	0.0304	**0.0451**	**0.0530**

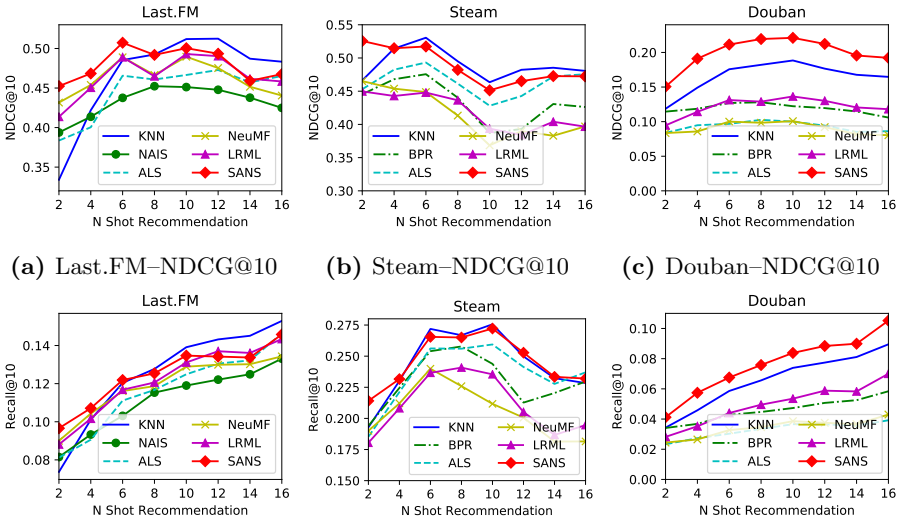
Impact of Neural Similarity SANS-SIM performs better than SANS-MLP and is closer to SANS, suggesting that it is reasonable to use the number of common users to measure the similarity between items. Adding MLP to the neural similarity yields the best-performing model because MLP can fit the nonlinear residual part of the neural similarity.

Impact of Setwise Attention The attention mechanism is designed for recommendation that is more than one shot, so we only compare SANS-AVG with SANS in $N > 1$ scenarios. The attention mechanism has significant impacts on the Last.FM and Steam datasets, while the improvement is negligible on the Douban dataset. The reason may be due to that Douban is much more sparse and has much more items than the other datasets so that the interactions of the few-shot users are too random to provide any additional information.

Impact of Weighted Loss The weighted loss can help to emphasize few-shot users when training the neural similarity module and setwise attention module in SANS. We can see from the results that the weighted loss can contribute to significant improvements on the Last.FM and Steam datasets but the improvement on the Douban dataset is negligible. Again, this should be due to the sparsity of the Douban dataset, so that most users are with very few ratings and giving higher weights to few-shot users does not make significant differences.

4.4 Analysis on Number of Shots (RQ3)

To find the best scenarios for SANS, we evaluate SANS with different numbers of shots. We split the three datasets with different numbers of shots from 2 to 16 by a step of 2. Due to space limitation, we only present the comparisons with the five best-performing methods on all datasets.



(a) Last.FM–NDCG@10 **(b)** Steam–NDCG@10 **(c)** Douban–NDCG@10

(d) Last.FM–Recall@10 **(e)** Steam–Recall@10 **(f)** Douban–Recall@10

Fig. 4. Performance comparison between SANS and five compared methods in N-shot recommendation with N varying from 2 to 16.

The experimental results are shown in Fig. 4. The relative improvements of SANS vary across different datasets. On the Last.FM dataset, SANS achieves the best performance when $N < 8$. On the Steam dataset, SANS achieves the best performance when $N \leq 4$. But on the Douban dataset, SANS consistently outperforms all the compared methods with significant margins. We have the following observations from the results: 1) SANS is desirable for few-shot recommendation on all datasets; 2) SANS is also more desirable on extremly sparse datasets, e.g., the Douban dataset, than the other methods even with lots of non-few-shot users; 3) KNN outperforms many recently proposed deep learning-based methods, which indicates that deep learning-based methods may not be appropriate for recommendations on few-shot users or on extremely sparse data due to higher chances of overfitting.

5 Related Work

5.1 Collaborative Filtering

Collaborative filtering-based recommendation algorithms achieve competitive performance in both rating prediction [11] and item ranking [2]. Classical collaborative filtering methods mainly includes matrix factorization-based methods which learn user/item feature vectors by various learning algorithms [7, 12, 17] and item-based methods which generate predictions based on item-item similarity matrix [5, 8]. However, many classical methods are linear and thus cannot capture non-linear relationships between users and items, so that many deep learning-based collaborative filtering methods have been proposed in recent years [2, 6, 24]. Because of the representation power of deep neural networks, deep learning-based methods outperform classical methods in most scenarios.

5.2 Cold Start Recommendation

Cold-start recommendation methods can also solve the few-shot recommendation problem if there is additional information to be exploited. For example, the attribute-to-feature mapping method [4] learns the mapping from features of users or items to their embedding vectors. Then, they generate embedding vectors of new users or items based on their features. Social information is also useful to alleviate the cold start problem [15, 16]. In addition, cross-domain recommendation algorithm [19] can use feedback from source domains to address cold-start recommendation in the target domain. However, additional information is not always available, so that the above methods may fail in practice. However, SANS does not require any additional information, i.e., more general than these methods.

5.3 Few-shot Learning

Few-shot learning was first proposed for object classification in computer vision [3], which derives a new classifier for objects from new category using

few training samples. The performance of few-shot learning has been significantly improved with the advances of deep learning and several neural network structures have been recently proposed [10,18,21]. Siamese neural networks [10] address one-shot learning by learning the similarity between samples using a network architecture composed of twin networks with shared weights. Matching networks [21] solve one-shot learning problems by conditional similarity. Prototypical networks [18] generate a prototypical vector for each few-shot category and predictions can be calculated based on distances to prototypes. To the best of our knowledge, there are very little prior works in few-shot recommendation.

6 Conclusion

This paper proposes SANS to address the few-shot recommendation problem. SANS consists of a neural similarity module to estimate the similarity of each pair of items and a setwise attention module to estimate user preferences. To facilitate training, we propose an adaptive weighted loss function with episodic sampling. Experimental studies on real datasets show that SANS can outperform state-of-the-art recommendation algorithms in the few-shot recommendation problem.

Acknowledgements. This work was supported by National Key Research and Development Project (No. 2018YFC0832303); National Natural Science Foundation of China (NSFC) under the Grants nos. 61932007 and 61902075.

References

1. Cheng, H.T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st workshop on deep learning for recommender systems. pp. 7–10 (2016)
2. Deng, Z.H., Huang, L., Wang, C.D., Lai, J.H., Philip, S.Y.: Deepcf: A unified framework of representation learning and matching function learning in recommender system. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 61–68 (2019)
3. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE transactions on pattern analysis and machine intelligence **28**(4), 594–611 (2006)
4. Gantner, Z., Drumond, L., Freudenthaler, C., Rendle, S., Schmidt-Thieme, L.: Learning attribute-to-feature mappings for cold-start recommendations. In: 2010 IEEE International Conference on Data Mining. pp. 176–185. IEEE (2010)
5. He, X., He, Z., Song, J., Liu, Z., Jiang, Y.G., Chua, T.S.: Nais: Neural attentive item similarity model for recommendation. IEEE Transactions on Knowledge and Data Engineering **30**(12), 2354–2366 (2018)
6. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: Proceedings of the 26th international conference on world wide web. pp. 173–182 (2017)
7. Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 263–272. Ieee (2008)

8. Kabbur, S., Ning, X., Karypis, G.: Fism: factored item similarity models for top-n recommender systems. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 659–667 (2013)
9. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
10. Koch, G., Zemel, R., Salakhutdinov, R.: Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop. vol. 2. Lille (2015)
11. Li, D., Chen, C., Lu, T., Chu, S., Gu, N.: Mixture matrix approximation for collaborative filtering. IEEE Transactions on Knowledge and Data Engineering (2019)
12. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. pp. 452–461 (2009)
13. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. pp. 285–295 (2001)
14. Schein, A.I., Popescul, A., Ungar, L.H., Pennock, D.M.: Methods and metrics for cold-start recommendations. In: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 253–260 (2002)
15. Sedhain, S., Menon, A.K., Sanner, S., Xie, L., Braziunas, D.: Low-rank linear cold-start recommendation from social data. In: Thirty-first AAAI conference on artificial intelligence (2017)
16. Sedhain, S., Sanner, S., Braziunas, D., Xie, L., Christensen, J.: Social collaborative filtering for cold-start recommendations. In: Proceedings of the 8th ACM Conference on Recommender systems. pp. 345–348 (2014)
17. Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., Hanjalic, A.: Climf: learning to maximize reciprocal rank with collaborative less-is-more filtering. In: Proceedings of the sixth ACM conference on Recommender systems. pp. 139–146 (2012)
18. Snell, J., Swersky, K., Zemel, R.: Prototypical networks for few-shot learning. In: Advances in neural information processing systems. pp. 4077–4087 (2017)
19. Tang, J., Wu, S., Sun, J., Su, H.: Cross-domain collaboration recommendation. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 1285–1293 (2012)
20. Tay, Y., Anh Tuan, L., Hui, S.C.: Latent relational metric learning via memory-based attention for collaborative ranking. In: Proceedings of the 2018 World Wide Web Conference. pp. 729–739 (2018)
21. Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al.: Matching networks for one shot learning. In: Advances in neural information processing systems. pp. 3630–3638 (2016)
22. Wang, Y., Wang, L., Li, Y., He, D., Chen, W., Liu, T.Y.: A theoretical analysis of ndcg ranking measures. In: Proceedings of the 26th annual conference on learning theory (COLT 2013). vol. 8, p. 6 (2013)
23. Yin, H., Cui, B., Li, J., Yao, J., Chen, C.: Challenging the long tail recommendation. arXiv preprint arXiv:1205.6700 (2012)
24. Zhang, S., Yao, L., Sun, A., Tay, Y.: Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR) **52** (2019)