ConsisRec: Enhancing GNN for Social Recommendation via Consistent Neighbor Aggregation

Liangwei Yang, Zhiwei Liu, Yingtong Dou
Department of Computer Science, University of Illinois at Chicago
{lyang84,zliu213,ydou5}@uic.edu

Jing Ma
Business School, Sichuan University
jingma@stu.scu.edu.cn

Philip S. Yu
Department of Computer Science, University of Illinois at Chicago
psyu@uic.edu

ABSTRACT
Social recommendation aims to fuse social links with user-item interactions to alleviate the cold-start problem for rating prediction. Recent developments of Graph Neural Networks (GNNs) motivate endeavors to design GNN-based social recommendation frameworks to aggregate both social and user-item interaction information simultaneously. However, most existing methods neglect the social inconsistency problem, which intuitively suggests that social links are not necessarily consistent with the rating prediction process. Social inconsistency can be observed from both context-level and relation-level. Therefore, we intend to empower the GNN model with the ability to tackle the social inconsistency problem. We propose to sample consistent neighbors by relating sampling work to aggregate both social and user-item interaction information. A recommender system predicts how likely a user is interested in an item \[19–21, 26, 27\]. However, due to the high cost of data collection, most existing recommender systems suffer from the cold-start problem \[18\]. To alleviate it, we can incorporate the social information among users \[15, 32, 35\], which functions as a side information of the user-item interaction. Previous endeavor \[2\] shows that users’ online behaviors are greatly influenced by their social networks, such as the friendship on Wechat \[36\], following links on Twitter \[16\] and trusting links on shopping website \[7\]. Therefore, fusing the social links with user-item interactions is advantageous to improve the recommendation performance, which is defined as the social recommendation problem.

The recent developments of Graph Neural Networks (GNNs) \[17, 19\] help handle social recommendation tasks by simultaneously aggregating the information from both social graph and user-item graph \[9, 24\]. Based on the assumption that neighbors share similar contexts, GNN learns node embeddings by aggregating neighbor information recursively on graphs \[5\]. SocialGCN \[33, 34\] proposes to enhance user embedding by simulating how users are influenced by the recursive social diffusion process. GraphRec \[6\] and GraphRec+ \[7\] jointly model three types of aggregation upon social graph, user-item graph and item-item graph, to learn user/item embeddings comprehensively. DSCF \[8\] includes high-order social links through a sequential learning on random walks. MGNN \[37\] builds mutual social embedding layers to aggregate information from user-item rating graph and social graph.

However, most existing GNN-based social recommendation models ignore the social inconsistency problem \[17\]. Specifically, the social inconsistency suggests that social links are not necessarily consistent with the rating prediction process. Aggregating the information from inconsistent social neighbors spoils the ability of a GNN to characterize beneficial information for recommendation. The social inconsistency can be categorized into two levels:

- **Context-level:** It indicates that users connected in a social graph may have discrepant item contexts. We demonstrate the context-level social inconsistency in Figure 1(a). As seen, \(u_1\) and \(u_2\)’s inconsistent neighbors because the items of \(u_3\) are all books, while \(u_2\)’s rated items all belong to sports. They have rather discrepant item contexts.
- **Relation-level:** There are multiple relations when simultaneously modeling social graph and user-item graph. For example, besides social relations, we also distinguish user-item relations by their rating values. In Figure 1(a), we observe the \(u_1\) and \(u_2\) are social neighbors and both connected with \(t_1\). However, \(u_1\) highly likes \(t_1\) (5 score) while \(u_2\) dislikes it (1 score). It leads to the relation-level inconsistency because though socially connected, they are of inconsistent item preference.

1 INTRODUCTION
A recommender system predicts how likely a user is interested in an item \[19–21, 26, 27\]. However, due to the high cost of data...
To this end, we intend to empower the GNN model to solve
the social inconsistency problem, which is non-trivial. On the
other hand, the contexts for both users and items are rather complex
and difficult to express explicitly. On the other hand, we should
model multiple relations simultaneously and distinguish the con-
sistent neighbors. Therefore, we propose a novel framework to
tackle the social inconsistency problem when conducting social
recommendation, which is named as ConsisRec. It is built upon
a GNN model [13], aggregating neighbors to learn node embed-
dings. To alleviate the social inconsistency problem, ConsisRec
first generates a query embedding for selecting consistent neighbors.
Then, it employs a neighbor sampling strategy for the selection
process, where the sampling probability is based on our proposed
consistency scores between the query embedding and neighbor
eMBEDDINGS. After sampling, it adopts relation attention to tackle
the relation-level inconsistency. As such, neighbors with consistent
relations are assigned with high importance factors for aggregation.
Therefore, the learned node embeddings for rating prediction
are aggregated from consistent contexts and relations. The code
is available online at https://github.com/YangLiangwei/ConsisRec.

The contributions of this paper are listed as follows:

- To the best of our knowledge, we are the first work empowering
  the GNN model to tackle the social inconsistency problem when
  conducting social recommendation.
- We propose a novel framework, ConsisRec, to learn consistent
  node embeddings for rating prediction.
- Experiments on two real-world datasets show the effectiveness
  of ConsisRec. Detailed analyses of ConsisRec justify its efficacy.

2 PRELIMINARIES

Social recommendation problem consists of two sets of entities, a
user set \( \mathcal{U} = \{ u_1, u_2, ..., u_m \} \) and an item set \( \mathcal{T} = \{ t_1, t_2, ..., t_n \} \),
where \( m \) and \( n \) are the total number of users and items, respectively.

It includes two types of information, an incomplete rating matrix
\( R \in \mathbb{R}^{m \times n} \) and the user-user social graph \( G_s = \{ \mathcal{U}, \mathcal{E}_s \} \). The
rating \( R_{u,t} \) denotes user \( u \)'s preference to item \( t \), where
higher score is interpreted as more preferring. An social edge \((u_i, u_j) \in \mathcal{E}_s \)
indicates that user \( i \) has a social connection with \( j \), e.g., trust.

The objective of social recommendation is to complete the rating
matrix by fusing the rating matrix and the social graph. Therefore,
we solve social recommendation problem by constructing a hetero-
geneous graph \( G = \{ V, E_r \}_r \}, \) where \( V \) denotes both user and
item nodes and \( E_r \) denotes edges upon relation \( r \). Besides user-user
and item-item links, we also distinguish the user-item links by their
rating scores. The score set varies on different datasets. E.g., Ciao
dataset [30] has 6 rating values, i.e., \{0, 1, 2, 3, 4, 5\}. Hence, the edges
on Ciao have 8 types, i.e., \( R = 8 \), one being social relation, one being
item-item relation and the others being different rating values.

3 PROPOSED MODEL

The framework of ConsisRec is shown in Figure 1(b).\(^1\) It has em-
bedding layer, query layer, neighbor sampling and relation attention.

\(^1\)Although we only illustrate the one-layer version of ConsisRec in the figure, our
model is flexible to have \( L \) layers as introduced in the following section.

3.1 Embedding Layer

Following existing works [10, 33], we maintain an embedding layer
\( E \in \mathbb{R}^{d \times (m+n)} \), each column of which represents the trainable
embedding for each node. We can index it to retrieve the embedding
of a node \( v \in \mathcal{U} \cup \mathcal{T} \) as \( e_v \in \mathbb{R}^d \). In the following sections, without
specific statements, we use an index \( v \) to denote a node, which
can either be a user or an item, while \( u \) and \( t \) specifically denote
a user and an item node. Apart from node embeddings, we also
train a relation embedding vector for each relation \( r \) to characterize
relation-level social inconsistency, denoted as \( e_r \).

3.2 Query Layer

To overcome the social inconsistency problem, we should aggregate
consistent neighbors to learn node embeddings. Since social incon-
stistency are both in context-level and relation-level, we should
distinguish consistent neighbors for each pair \((u, t)\). Therefore,
ConsisRec employs a query layer to exclusively select consistent
neighbors for the query pair \((u, t)\). It generates a query embedding
by mapping the concatenation of user and item embeddings:

\[
q_{u,t} = \sigma \left( W_?^T (e_u \oplus e_t) \right),
\]

where \( q_{u,t} \) is the query embedding, \( e_u, e_t \in \mathbb{R}^d \) are the embedding
for node \( u \) and \( t \), respectively, \( \oplus \) denotes concatenation, \( W_q \in \mathbb{R}^{2d \times d} \)
is the mapping matrix, and \( \sigma \) is a ReLU activation function.

We design a query layer to dynamically sample neighbors based on
different items. It is because when users buy different items, they
would inquire different friends. Thus, \( u \)'s rating score of \( t \) is related
to friends who are familiar with this query item \( t \).

3.3 Neighbor Sampling

Neighbor sampling has been applied to GNN to boost training [3, 4,
38] and improve ranking performance [25]. Compared with
previous work, ConsisRec aims to deal with the inconsistency problem
in social recommendation, and dynamic samples different social
neighbors based on different items. Next, we present how to sample
neighbors for learning the embedding of \( u \) and \( t \). The framework
of ConsisRec to aggregate node embeddings can be formalized as:

\[
h_v^{(l)} = \sigma \left( W_v^{(l)} \sum_{i \in N_v} s_v^{(l)} (h_i^{(l-1)} \oplus AGG^{(l)}(h_i^{(l-1)})); i \in \mathcal{N}_v) \right),
\]

where \( \sigma \) is a ReLU activation function, \( h_v^{(l)} \in \mathbb{R}^d \) is the hidden
embedding of node \( v \) at \( l \)-th layer, \( \mathcal{N}_v \) is the sampled neighbors
of node \( v \), AGG is an aggregation function, and \( W_v^{(l)} \in \mathbb{R}^{2d \times d} \)
is the mapping function. \( h_v^{(0)} \) is the initial node embedding of \( v \), i.e., \( e_v \).

Instead of equally aggregating all neighbors, we should empha-
size more on consistent neighbors while ignoring those inconsistent
neighbors. Therefore, we propose to use neighbor sampling method
to select those consistent neighbors. The sampling probability for
neighbor node \( i \) at \( l \)-th layer is defined by the consistency score
between query \( q \) and all the neighbors as:

\[
p^{(l)} (i; q) = s_v^{(l)} (i; q) / \sum_{j \in \mathcal{N}_v} s_v^{(l)} (j; q).
\]
where $s(t)(i;q)$ denotes the consistency score between the neighbor $i$ and the query $q$ in $l$-th GNN layer. It is defined as:

$$s(t)(i;q) = \exp(-\|q - h_{t}(i)^{(l)}\|^2), \quad (4)$$

where $h_{t}(i)^{(l)}$ denotes the node embedding of node $i$ at $l$-th layer. For both nodes $u$ and $t$, during the inference of rating score, we use the same query embedding. Thus, we ignore the subscript and write it as $q$ for simplicity. We present this process as the sampling blocks in Figure 1(b), where the probabilities for neighbors are denoted as $p_i$. The number of sampled neighbors is proportional to the total number of neighbors, where the ratio is $0 \leq \gamma \leq 1$. As such, we sample more neighbors if a node is connected to more nodes.

### 3.4 Relation Attention

After sampling the neighbors, we should aggregate their embeddings as illustrated in Eq. (2). However, the relation-level social inconsistency suggests that we should distinguish different relations. To this end, we apply a relation attention module in ConsisRec for those sampled neighbors. It learns the importance of those sampled nodes by considering the associated relations.

The relation attention assigns an importance factor $a_i$ for each sampled node $i$. We can rewrite the AGG function in Eq. (2) as:

$$\text{AGG}^{(l)} = \sum_{i=1}^{Q} a_i^{(l)} \cdot h_{t}^{(l-1)}, \quad (5)$$

where $a_i^{(l)}$ is the importance of the $i$-th neighbor sampled from Eq. (3) and $Q$ denotes the total number of sampled neighbors. Assuming the relation for the edge $(a, i)$ is $r_i$, we calculate $a_i$ by adopting the self-attention mechanism as:

$$a_i^{(l)} = \frac{\exp(w_a^\top (h_{t}^{(l-1)} \oplus e_{r_i}))}{\sum_{j=1}^{Q} \exp(w_a^\top (h_{t}^{(l-1)} \oplus e_{r_j}))}, \quad (6)$$

where $e_{r_i} \in \mathbb{R}^d$ represents the relation embedding of relation $r_i$ and $w_a \in \mathbb{R}^{2d}$ is trainable parameter for the self-attention layer and $a_i$ is the attention weights. We illustrate the relation attention as the green block in Figure 1(b).

### 3.5 Rating Prediction and Optimization

After $L$ layer propagation, we obtain the embedding of $u$ and $t$, which are denoted as $h_{u}^{(L)}$ and $h_{t}^{(L)}$. We calculate the rating score of the user-item pair $(u, t)$ by the inner-product of embeddings as:

$$\hat{R}_{u,t} = h_{u}^{(L)} \cdot h_{t}^{(L)}. \quad (7)$$

Then the loss function is defined as the Root Mean Squared Error (RMSE) between $\hat{R}_{u,t}$ and ground truth rating score $R_{u,t}$ among all $(u, t)$ pairs in $E_{\text{rating}}$, which is calculated as

$$L = \sqrt{\frac{\sum_{(u,t) \in E_{\text{rating}}} (R_{u,t} - \hat{R}_{u,t})^2}{|E_{\text{rating}}|}}. \quad (8)$$

where $E_{\text{rating}}$ is the set of all rating edges. We use Adam [12] as the optimizer with a weight decay rate of 0.0001 to avoid over-fitting.

## 4 EXPERIMENTS

### 4.1 Experimental Setup

#### 4.1.1 Datasets.

Ciao and Epinions2 are two representative datasets [28–31] for studying social recommendation problem. We remove users without social links because they are out of social recommendation scope. Ciao has 7,317 users, 104,975 items with 111,781 social links. Epinions has 18,069 users, 261,246 items with 355,530 social links. We also linked items that share more than 50% of their neighbors.

#### 4.1.2 Baselines.

To justify the effectiveness of ConsisRec, we compare ConsisRec with 6 baseline methods, including matrix factorization methods, non-GNN graph embedding methods, and GNN-based methods. SoRec [22], SocialMF [11] and SoReg [23] incorporate social links with matrix factorization methods. CUNE [39] adopts collaborative graph embedding methods. GCMC+SN [1] and GraphRec [6] employ GNNs for learning node embeddings.

#### 4.1.3 Evaluation Metrics.

To evaluate the quality of the social recommendation, two common metrics, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE), are adopted for the rating prediction task [6]. Note that lower values of both indicate better performance. And a small improvement in both may have a significant impact on the quality of top-N recommendation [14].

---

2https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.htm
4.1.4 Experimental Settings. Each dataset is randomly split to 60%, 20%, and 20% for the training, validation, and testing, respectively. The grid search is applied for hyper-parameters tuning. We searched neighbor percent in \(\{0.2, 0.4, 0.6, 0.8, 1.0\}\). For embedding size, we search in \(\{8, 16, 32, 64, 128, 256\}\). The learning rate is searched in \(\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1\}\). The batch size is searched in \(\{32, 64, 128, 256, 512\}\). Only one GNN layer is used for both Ciao and Epinions datasets. To cope with the over-fitting problem, early stopping was utilized in all experiments, i.e., stop training if the RMSE on the validation set is not improved for five epochs.

4.2 Performance Evaluation

Table 1: Overall comparison. The best and the second-best results are in bold and underlined, respectively.

| Method      | Ciao       | Epinions   |
|-------------|------------|------------|
|             | RMSE       | MAE        | RMSE       | MAE        |
| SoRec       | 1.2024     | 0.8693     | 1.3389     | 1.0618     |
| SoReg       | 1.0066     | 0.7595     | 1.0751     | 0.8309     |
| SocialMF    | 1.0013     | 0.7535     | 1.0706     | 0.8264     |
| GCMC+SN     | 1.0301     | 0.7970     | 1.1070     | 0.8480     |
| GraphRec    | 1.0040     | 0.7591     | 1.0799     | 0.8219     |
| CUNE        | 1.0002     | 0.7591     | 1.0681     | 0.8284     |
| ConsisRec   | \textbf{0.9722} | \textbf{0.7394} | \textbf{1.0495} | \textbf{0.8046} |
| Improvement | 2.79%      | 1.87%      | 1.74%      | 2.1%       |

The experiment results of all the methods are shown in Table 1. GCMC, GraphRec, CUNE and ConsisRec perform better than SoRec, SoReg and SocialMF, which shows GNN and graph embedding based methods have a better capability to aggregate neighbor information. ConsisRec achieves the best results on both Ciao and Epinions datasets. It has an 1.7% relative improvement on two datasets compared with the second-best one on average, which can be interpreted as a significant improvement [6]. The results show the benefits brought by tackling the social inconsistency problems.

4.3 Ablation Study

An ablation study is further made to evaluate different components in ConsisRec. We create three variants of ConsisRec, which are A, B, and C. A is built by removing the query layer, which directly uses user embedding instead of query embedding to select the corresponding neighbors. B is built by removing neighbor sampling, which aggregates all neighbors. C is built by removing relation attention, which assigns equal weights to edges with different relations. The experimental results are illustrated in Figure 2.

We can observe that ConsisRec consistently achieves the best performance against other variants, demonstrating that all components are necessary to yield the best results. Additionally, we observe that the variant B (removing neighbor sampling module) dramatically spoils the performance, which justifies the importance of selecting consistent neighbors. The worse performance of variant A and C compared with ConsisRec also proves the importance of query layer and relation attention, respectively.

4.4 Parameter Sensitivity

Influential hyper-parameters in ConsisRec includes neighbor percent, embedding size and the learning rate. Due to space limitation, we only represent results on Ciao dataset in Figure 3. For neighbor percent, we observe an obvious error increment when neighbor percent rising from 0.8 to 1.0, which results from aggregating those inconsistent neighbors. The best embedding size on Ciao is 16. Smaller embedding size is insufficient for representing node information, while large embedding size would lead to the over-fitting problem. Learning rate has a critical impact on model performance, which needs to be tuned carefully.

5 CONCLUSION AND FUTURE WORK

In this paper, we identify the social inconsistency problem related to social recommendation. The proposed ConsisRec contains three modifications on GNN to tackle the social inconsistency problem. Experiment results on two real-world datasets show the effectiveness of ConsisRec. Future work includes better ways to filter informative neighbors and identify the inconsistency problems inherited in other graph related research directions.

6 ACKNOWLEDGEMENTS

This work is supported in part by NSF under grants III-1763325, III-1909323, and SaTC-1930941.
