Leveraging Tripartite Interaction Information from Live Stream E-Commerce for Improving Product Recommendation

Sanshi Lei Yu[1], Zhuoxuan Jiang[1,2], Dong-Dong Chen, Shanshan Feng[2], Dongsheng Li, Qi Liu, Jinfeng Yi

June 23, 2021

[1]Equal contributions.

[2]Corresponding authors.

Contents

Introduction

Recently, live streaming E-commerce is advancing rapidly.

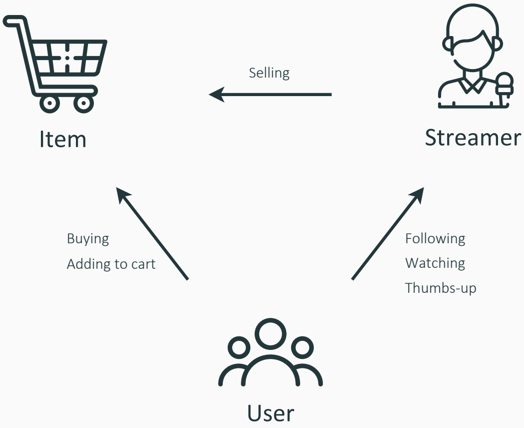


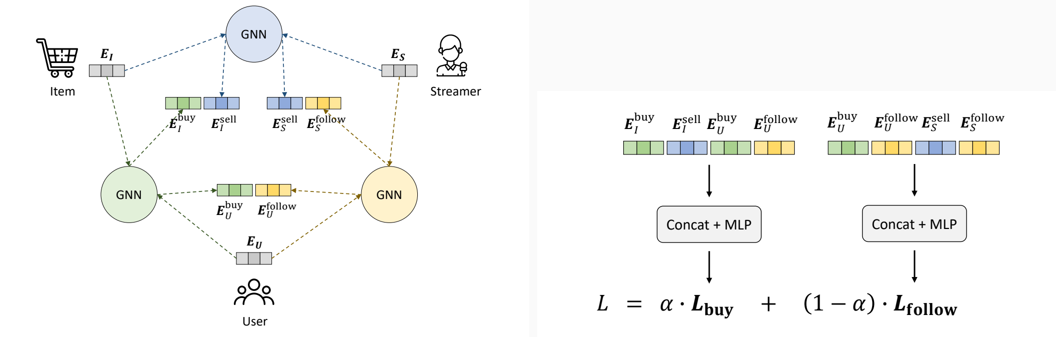
Figure 1: Example of the live stream E-commerce scenario

Introduction

We collect two new real-world datasets of tripartite interactions from an E-Commerce website, and propose the **streamers' influence**:

1. The streamers connect users with items.
2. The streamers connect similar users.
3. The streamers connect similar items.

We propose the Live Stream E-Commerce Graph Neural Network (LSEC-GNN), which leverages streamers' influence for improving product recommendation.



(a) Bipartite node embedding learning

(b) Model prediction and optimization

Figure 2: The overall architecture of LSEC-GNN framework

Problem Definition

Definition

User-Item Buying Graph, denoted as $\mathcal{G}_{\text{buy}} = (\mathcal{U} \cup \mathcal{I}, A_{\text{buy}})$, where $\mathcal{U}$ is the set for users and $\mathcal{I}$ is the set for items. $A_{\text{buy}} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{I}|}$ is the adjacency matrix for the graph. For the k-th row, j-th column element $a_{j,k}$ in A_{buy}, we have:

$$a_{j,k} = \begin{cases} 1 & \text{when } u_j \in \mathcal{U} \text{ bought } i_k \in \mathcal{I}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Problem Definition

Definition

User-Streamer Following Graph, denoted as $\mathcal{G}_{\text{follow}} = (\mathcal{U} \cup \mathcal{S}, A_{\text{follow}})$, where $\mathcal{U}$ is the set for users and $\mathcal{S}$ is the set for streamers. $A_{\text{follow}} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{S}|}$ is the adjacency matrix for the graph. For the k-th row, j-th column element $a_{j,k}$ in A_{follow}, we have:

$$a_{j,k} = \begin{cases} 1 & \text{when } u_j \in \mathcal{U} \text{ followed } s_k \in \mathcal{S}, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Problem Definition

Definition

Streamer-Item Selling Graph, denoted as $\mathcal{G}_{\text{sell}} = (\mathcal{S} \cup \mathcal{I}, A_{\text{sell}})$, where $\mathcal{S}$ is the set for streamers and $\mathcal{I}$ is the set for items. $A_{\text{sell}} \in \mathbb{R}^{|\mathcal{S}| \times |\mathcal{I}|}$ is the adjacency matrix for the graph. For the k-th row, j-th column element $a_{j,k}$ in A_{sell}, we have:

$$a_{j,k} = \begin{cases} 1 & \text{streamer } s_j \in \mathcal{S} \text{ sold } i_k \in \mathcal{I} \text{ and the sales reaches the threshold,} \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Problem Definition

Definition

Heterogeneous Tripartite Interaction Graph is denoted as $\langle \mathcal{G}_{\text{buy}}, \mathcal{G}_{\text{follow}}, \mathcal{G}_{\text{sell}} \rangle$. It captures tripartite perspectives of interaction relationship between humans and products specifically in live stream E-Commerce.

Problem Definition

Definition

Heterogeneous Tripartite Interaction Graph is denoted as $\langle \mathcal{G}_{\text{buy}}, \mathcal{G}_{\text{follow}}, \mathcal{G}_{\text{sell}} \rangle$. It captures tripartite perspectives of interaction relationship between humans and products specifically in live stream E-Commerce.

Definition

Product Recommendation with Tripartite Interaction Information

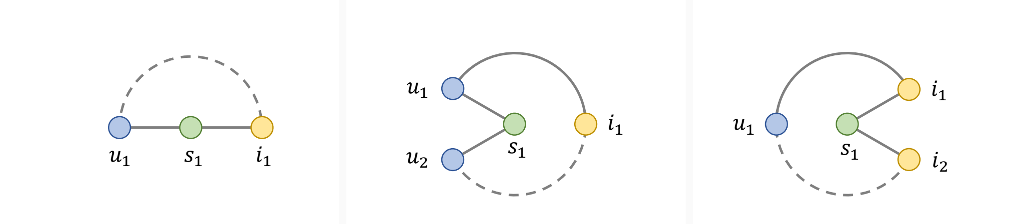
Given **Heterogeneous Tripartite Interaction Graph** $\langle \mathcal{G}_{\text{buy}}, \mathcal{G}_{\text{follow}}, \mathcal{G}_{\text{sell}} \rangle$, the problem of product recommendation with tripartite interaction information aims to learn low-dimensional representations for nodes and make recommendations between users and items based on the representations.

Data Analysis

We collect two new real-world datasets of tripartite interactions from an E-Commerce website. After three simulation experiments by the Monte Carlo method[1], we propose the **streamers' influence**:

1. The streamers connect users with items.
2. The streamers connect similar users.
3. The streamers connect similar items.

[1]For the details of the experiments, please refer to the paper.



(a) The streamers connect users with items

(b) The streamers connect similar users

(c) The streamers connect similar items

Figure 3: Three patterns of **streamers’ influence**. *u*, *i*, *s* mean user, item and steamer, respectively. The solid lines represent the observed interactions while the dash lines represent the unobserved but potential interactions.

The streamers connect users with items

Users are more likely to buy items sold by their following streamers.

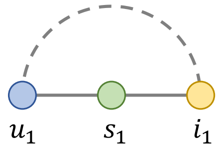


Figure 4: The first pattern of **streamers' influence**

The streamers connect similar users

The users following common streamer(s) are more similar (w.r.t. the bought items).

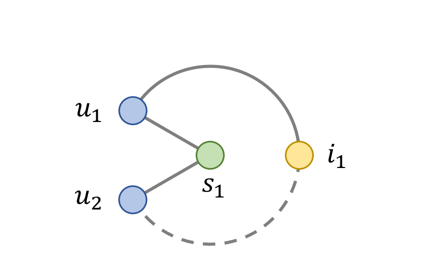


Figure 5: The second pattern of **streamers' influence**

The streamers connect similar items

The items sold by common streamer(s) are more similar (w.r.t. buyers)

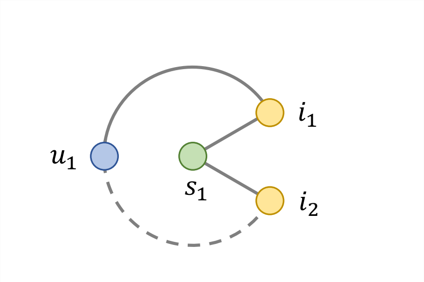
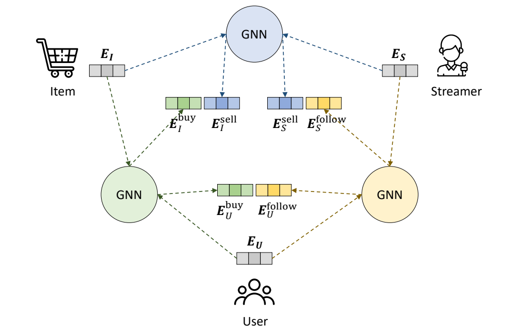


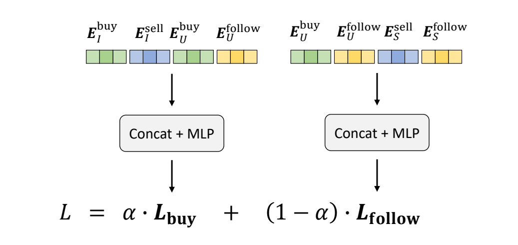
Figure 6: The third pattern of **streamers' influence**

Methodology

To make fully use of the tripartite interaction information, we propose the Live Stream E-Commerce Graph Neural Network (LSEC-GNN) framework.



(a) Bipartite node embedding learning



(b) Model prediction and optimization

Figure 7: The overall architecture of LSEC-GNN framework

Methodology

Bipartite Node Embedding Learning

Bipartite Node Embedding Learning

First we build the embedding lookup table for each type of nodes:

$$
\begin{aligned}
\mathsf{E}_U &= [e_{u_1}, e_{u_2}, \ldots, e_{u_{|\mathcal{U}|}}] \\
\mathsf{E}_I &= [e_{i_1}, e_{i_2}, \ldots, e_{i_{|\mathcal{I}|}}] \\
\mathsf{E}_S &= [e_{s_1}, e_{s_2}, \ldots, e_{s_{|\mathcal{S}|}}]
\end{aligned} \tag{4}
$$

Bipartite Node Embedding Learning

First we build the embedding lookup table for each type of nodes:

$$\begin{aligned} \mathsf{E}_U &= [e_{u_1}, e_{u_2}, \ldots, e_{u_{|\mathcal{U}|}}] \\ \mathsf{E}_I &= [e_{i_1}, e_{i_2}, \ldots, e_{i_{|\mathcal{I}|}}] \\ \mathsf{E}_S &= [e_{s_1}, e_{s_2}, \ldots, e_{s_{|\mathcal{S}|}}] \end{aligned} \tag{4}$$

Then in each bipartite graph, multiple GNN embedding propagation layers are operated, which refines a node's representation by aggregating the embeddings of the interacted nodes.

Bipartite Node Embedding Learning

First we build the embedding lookup table for each type of nodes:

$$\begin{aligned} \mathsf{E}_U &= [e_{u_1}, e_{u_2}, \ldots, e_{u_{|\mathcal{U}|}}] \\ \mathsf{E}_I &= [e_{i_1}, e_{i_2}, \ldots, e_{i_{|\mathcal{I}|}}] \\ \mathsf{E}_S &= [e_{s_1}, e_{s_2}, \ldots, e_{s_{|\mathcal{S}|}}] \end{aligned} \tag{4}$$

Then in each bipartite graph, multiple GNN embedding propagation layers are operated, which refines a node's representation by aggregating the embeddings of the interacted nodes.

The LSEC-GNN framework does not limit the choice of GNN layer. To demonstrate the embedding propagation process, we will take GCN as the example.

In GCN, the l-th is updated as follows:

$$\begin{aligned} H^{(l+1)} &= f_{\mathrm{GCN},l}(H^{(l)}) \\ &= \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \end{aligned} \tag{5}$$

where $\tilde{A} = A + I_N$ is the adjacency matrix of a bipartite graph $\mathcal{G}$ with added self-connections. I_N denotes the identity matrix, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and $W^{(l)}$ is a layer-specific trainable weight matrix. $\sigma(\cdot)$ denotes the activation function. $H^{(l)}$ is the input embeddings in l-th layer.

Note the previous equation is for homogeneous graph while the bipartite graph is a heterogeneous graph. So we need to convert it into a homogeneous graph before applying the equation. Take **User-Item Buying Graph** for example, the input embedding $H^{(0)}$ and adjacency matrix A for the converted homogeneous graph can be formulated as:

$$\begin{aligned} H^{(0)} &= [\mathbf{E}_U, \mathbf{E}_I] \\ A &= \begin{bmatrix} 0 & A_{\text{buy}} \\ A_{\text{buy}}^{\intercal} & 0 \end{bmatrix} \end{aligned} \tag{6}$$

where $\mathbf{E}_U$ and $\mathbf{E}_I$ are the outputs of user and item embedding lookup tables, respectively. A_{buy} is the adjacency matrix for **User-Item Buying Graph**.

If we use two GCN layers to refine node embeddings, then on **User-Item Buying Graph** we have:

$$H^{(1)} = f^{\text{buy}}_{\text{GCN},0}([\mathrm{E}_U, \mathrm{E}_I])$$
$$[\mathrm{E}^{\text{buy}}_U, \mathrm{E}^{\text{buy}}_I] = f^{\text{buy}}_{\text{GCN},1}(H^{(1)})$$

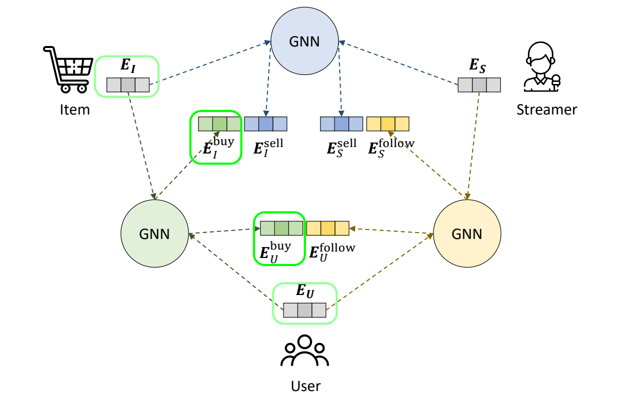


Figure 8: Bipartite node embedding learning

Bipartite Node Embedding Learning

Similarly, on **User-Streamer Following Graph** we have:

$$H^{(1)} = f_{\text{GCN},0}^{\text{follow}}([\mathrm{E}_U, \mathrm{E}_S])$$

$$[\mathrm{E}_U^{\text{follow}}, \mathrm{E}_S^{\text{follow}}] = f_{\text{GCN},1}^{\text{follow}}(H^{(1)})$$

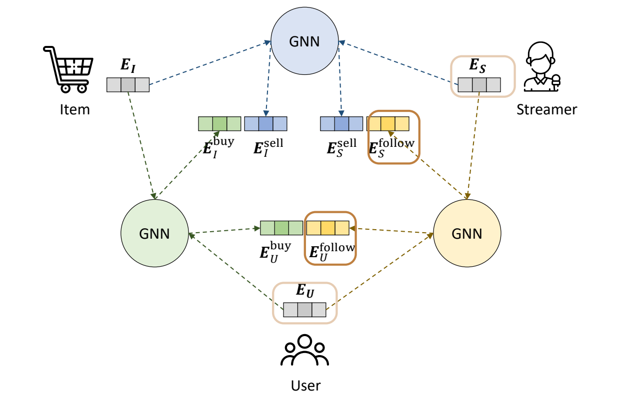


Figure 9: Bipartite node embedding learning

On **Streamer-Item Selling Graph** we have:

$$H^{(1)} = f^{\text{sell}}_{\text{GCN},0}([\mathrm{E}_S, \mathrm{E}_I])$$
$$[\mathrm{E}^{\text{sell}}_S, \mathrm{E}^{\text{sell}}_I] = f^{\text{sell}}_{\text{GCN},1}(H^{(1)})$$

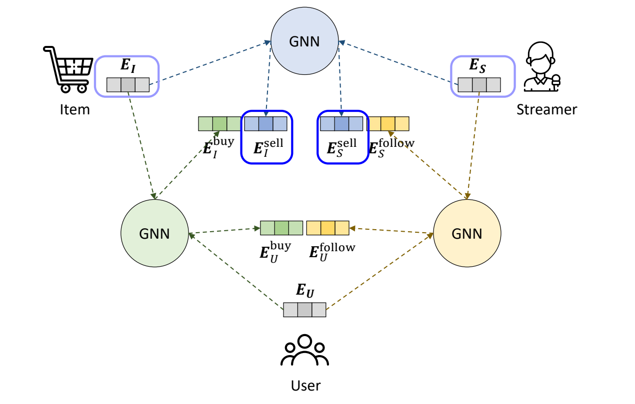


Figure 10: Bipartite node embedding learning

Bipartite Node Embedding Learning

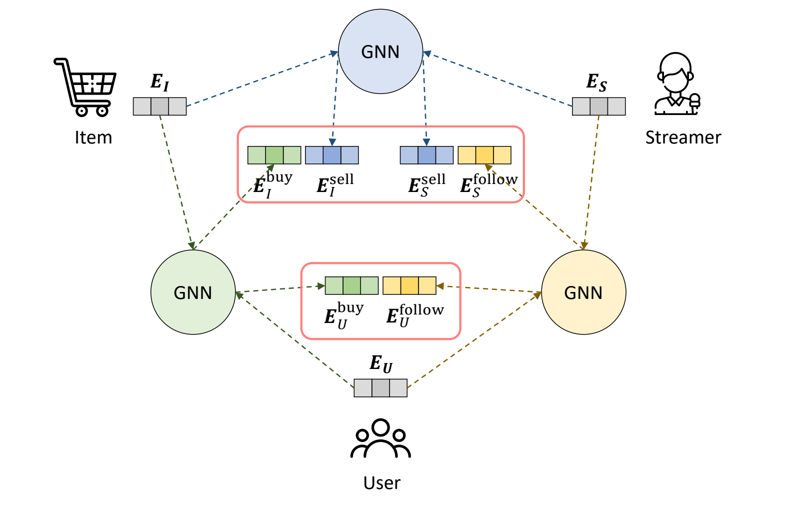


Figure 11: Bipartite node embedding learning

Methodology

Model Prediction

How to make use of the embeddings for prediction?

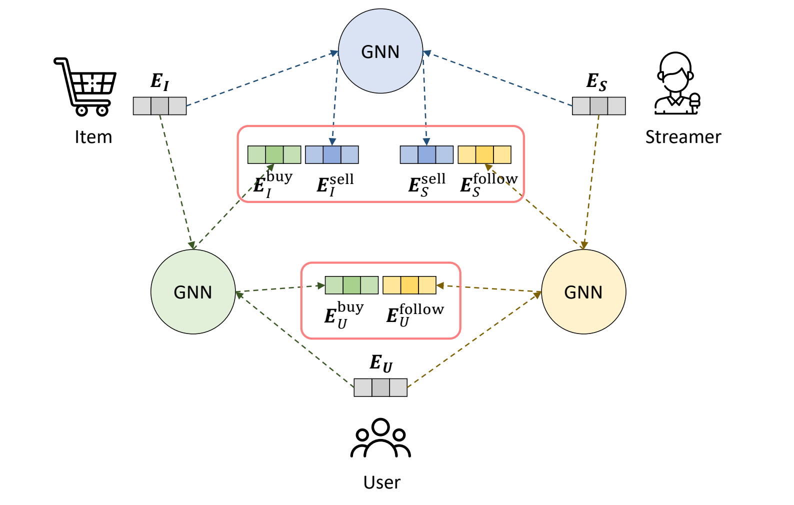


Figure 12: Bipartite node embedding learning

Model Prediction

To capture the complex relationships between vectors from different spaces:

Model Prediction

To capture the complex relationships between vectors from different spaces:

Concat + MLP !

Model Prediction

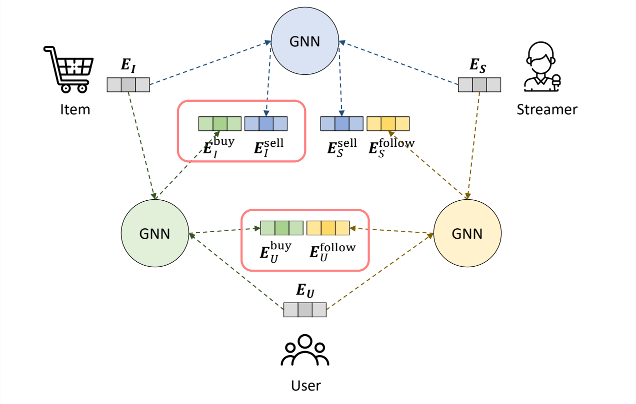


Figure 13: Bipartite node embedding learning

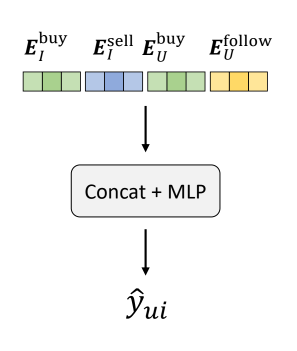


Figure 14: Model prediction with Concat + MLP

Model Prediction

If we use a two-layers MLP, the buying preference score $\hat{y}_{ui}$ between user u and item i is formulated as:

$$\begin{aligned} v_{ui} &= [\mathbf{E}_I^{\text{sell}}{}_i \| \mathbf{E}_I^{\text{buy}}{}_i \| \mathbf{E}_U^{\text{buy}}{}_u \| \mathbf{E}_U^{\text{follow}}{}_u] \\ \hat{y}_{ui} &= \sigma_2(\mathbf{W}_2^T(\sigma_1(\mathbf{W}_1^T v_{ui} + \mathbf{b}_1)) + \mathbf{b}_2) \end{aligned} \tag{7}$$

where $\mathbf{W}_x$, $\mathbf{b}_x$ and σ_x denote the weight matrix, bias vector and activation function for the x-th layer of MLP, respectively.

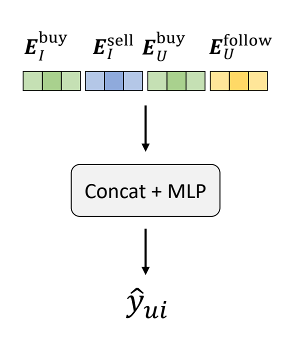


Figure 13: Model prediction with Concat + MLP

Methodology

Multi-Task Optimization

We adopt the **binary cross-entropy loss** (aka, **log loss**).

$$L_{\text{buy}} = - \sum_{(u,i) \in \mathcal{Y} \cup \mathcal{Y}^-} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}). \tag{8}$$

where $\mathcal{Y}$ is the set for positive items. If interaction between user u and item i is observed, put (u, i) into $\mathcal{Y}$. $\mathcal{Y}^-$ is the set for negative items. We construct it with negative sampling strategy. Specifically, for each (u, i) in $\mathcal{Y}$, we randomly select K uninteracted items $i_1^-, i_2^-, \ldots, i_K^-$. Then we put $(u, i_1^-), (u, i_2^-), \ldots, (u, i_K^-)$ into $\mathcal{Y}^-$. y_{ui} is the ground truth. $y_{ui} = 1$ if $(u, i) \in \mathcal{Y}$. If $(u, i) \in \mathcal{Y}^-$ then $y_{ui} = 0$. $\hat{y}_{ui}$ is the prediction value for y_{ui}.

Buying Task Optimization

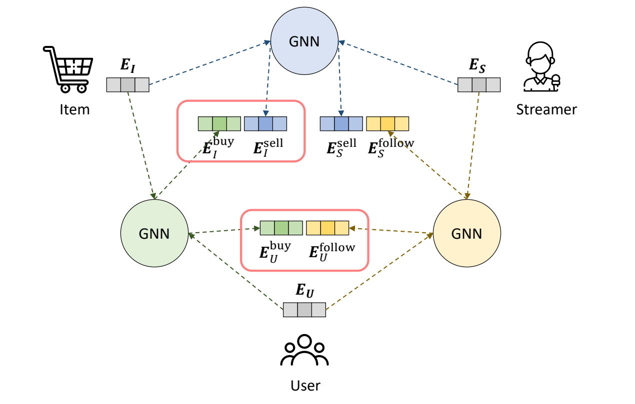


Figure 14: Bipartite node embedding learning

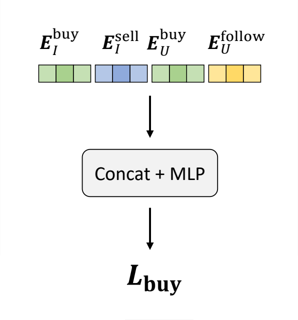


Figure 15: Buying task prediction and optimization

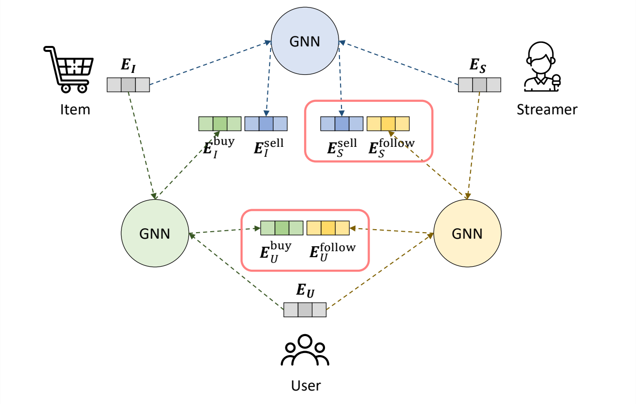


Figure 16: Bipartite node embedding learning

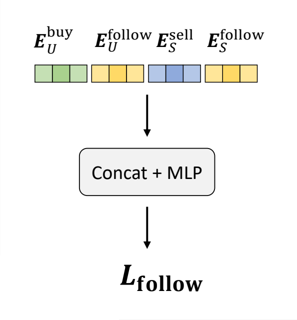


Figure 17: Following task prediction and optimization

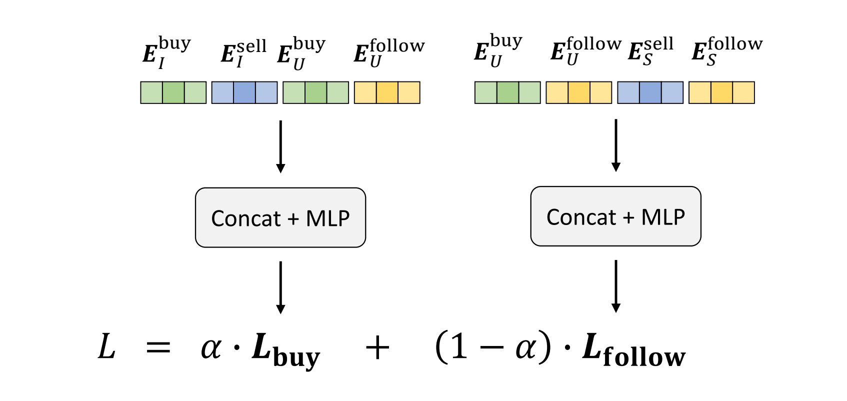


Figure 18: Multi-task optimization

Experiments

Experiments

Dataset Description

Dataset Description

We build the Live Stream E-Commerce (LSEC) datasets in the following steps:

Dataset Description

We build the Live Stream E-Commerce (LSEC) datasets in the following steps:

1. **Select the interaction relationships**
   we select three relationships from Live Stream E-Commerce scenario:
   User-Item Buying, **User-Streamer Following** and **Streamer-Item Selling**.

Dataset Description

We build the Live Stream E-Commerce (LSEC) datasets in the following steps:

1. **Select the interaction relationships**
   we select three relationships from Live Stream E-Commerce scenario:
   User-Item Buying, **User-Streamer Following** and **Streamer-Item Selling**.
2. **Divide the time range**
   From December 1, 2020, the first 30 days are for training, the following 10 days for validation and the last 10 days for test.

Dataset Description

We build the Live Stream E-Commerce (LSEC) datasets in the following steps:

1. **Select the interaction relationships**
   we select three relationships from Live Stream E-Commerce scenario: **User-Item Buying**, **User-Streamer Following** and **Streamer-Item Selling**.
2. **Divide the time range**
   From December 1, 2020, the first 30 days are for training, the following 10 days for validation and the last 10 days for test.
3. **Construct the node lists**
   We randomly sample the data and construct the lists for users, items and streamers. Based on different sampling ratios, we get two groups of the lists.

Dataset Description

We build the Live Stream E-Commerce (LSEC) datasets in the following steps:

1. **Select the interaction relationships**
   we select three relationships from Live Stream E-Commerce scenario:
   User-Item Buying, **User-Streamer Following** and **Streamer-Item Selling**.
2. **Divide the time range**
   From December 1, 2020, the first 30 days are for training, the following 10 days for validation and the last 10 days for test.
3. **Construct the node lists**
   We randomly sample the data and construct the lists for users, items and streamers. Based on different sampling ratios, we get two groups of the lists.
4. **Extract the data**
   Extract the three types of interactions between the nodes. The sales threshold for **User-Streamer Following** is set as 20. The length of ranked list for Top-N recommendation evaluation is set as 500.

In this way, we got two datasets: **LSEC-Small** and **LSEC-Large**. The statistics of them are summarized in the following tables.

Table 1: Statistics for LSEC-Small

Node	#
User	29 422
Item	31 630
Streamer	4633

Edge	#	Density
Buying	451 441	0.0485 %
Following	1 659 943	1.2178 %
Selling	1 168 165	0.7972 %

Table 2: Statistics for LSEC-Large

Node	#
User	202 850
Item	109 502
Streamer	7395

Edge	#	Density
Buying	3 062 463	0.0138 %
Following	5 439 288	0.3626 %
Selling	1 953 881	0.2413 %

Experiments

Experimental Settings

Evaluation Metrics

We refer to the following commonly-used metrics in Top-N recommendation:

- **AUC** Area Under the ROC Curve.
- **MRR** Mean Reciprocal Rank.
- **NDCG@N** Normalized Discounted Cumulative Gain.
- **Recall@N**.

N is set to 10 and 50. The average metrics across all users are taken as the final metrics.

Baseline Models

We choose one non-graph model and three graph-based models as our baselines:

Baseline Models

We choose one non-graph model and three graph-based models as our baselines:

- **NCF** includes three instantiations of neural CF: GMF, MLP, and the fusion of the former two, called NeuMF. In this work, NeuMF achieves the best performance, hence we choose it as the baseline.

Baseline Models

We choose one non-graph model and three graph-based models as our baselines:

- **NCF** includes three instantiations of neural CF: GMF, MLP, and the fusion of the former two, called NeuMF. In this work, NeuMF achieves the best performance, hence we choose it as the baseline.
- **GCN-RS** GCN is a well-known graph-based model. It is originally proposed for node classification task and we adapt it for recommendation. This method is denoted as GCN-RS.

Baseline Models

We choose one non-graph model and three graph-based models as our baselines:

- **NCF** includes three instantiations of neural CF: GMF, MLP, and the fusion of the former two, called NeuMF. In this work, NeuMF achieves the best performance, hence we choose it as the baseline.
- **GCN-RS** GCN is a well-known graph-based model. It is originally proposed for node classification task and we adapt it for recommendation. This method is denoted as GCN-RS.
- **LightGCN** is a variant of GCN by removing feature transformation and nonlinear activation. For fair comparison, we replace the inner product predictor with MLP predictor to keep the same setting as our proposed methods.

Baseline Models

We choose one non-graph model and three graph-based models as our baselines:

- **NCF** includes three instantiations of neural CF: GMF, MLP, and the fusion of the former two, called NeuMF. In this work, NeuMF achieves the best performance, hence we choose it as the baseline.
- **GCN-RS** GCN is a well-known graph-based model. It is originally proposed for node classification task and we adapt it for recommendation. This method is denoted as GCN-RS.
- **LightGCN** is a variant of GCN by removing feature transformation and nonlinear activation. For fair comparison, we replace the inner product predictor with MLP predictor to keep the same setting as our proposed methods.
- **GAT-RS** GAT is a variant of GCN by applying an attention mechanism to learn the weights for neighbors aggregation, instead of simply taking the average among them. We adapt it for recommendation task and name it GAT-RS.

Hyper-parameters Settings

The validation set is used in hyperparameter tuning and early-stop mechanism. The hyperparameters we used are as follows:

- Learning rate: 0.0005
- Batch size: 4096
- Embedding lookup table dimension: 200
- GNN layer and dimension: $-1 \rightarrow 128 \rightarrow 64$ (-1 for *inferred from other conditions*, the same below.)
- Negative sampling ratio K: 4
- MLP layer and dimension: $-1 \rightarrow 128 \rightarrow 1$
- Loss coefficient α for multi-task training: 0.5

Hyper-parameters Settings

The validation set is used in hyperparameter tuning and early-stop mechanism. The hyperparameters we used are as follows:

- Learning rate: 0.0005
- Batch size: 4096
- Embedding lookup table dimension: 200
- GNN layer and dimension: $-1 \to 128 \to 64$ (-1 for *inferred from other conditions*, the same below.)
- Negative sampling ratio K: 4
- MLP layer and dimension: $-1 \to 128 \to 1$
- Loss coefficient α for multi-task training: 0.5

After hyperparameters tuned, we train the model with the best hyperparameters and once it got early stopped we evaluate it directly on the test set and report the metrics as the final result.

Experiments

Performance Comparison

Performance Comparison

We repeat each experiment for 5 times and report the average metrics on the test set.

Table 3: Comparison of the models on LSEC-Small dataset

Model	AUC	MRR	NDCG@10	NDCG@50	Recall@10	Recall@50
NCF	0.8103	0.1588	0.2392	0.3188	0.2832	0.5405
GCN-RS	0.8440	0.1835	0.2862	0.3705	0.3363	0.6078
LightGCN	0.8483	0.1858	0.2895	0.3719	0.3374	0.6069
GAT-RS	0.8506	0.1828	0.2889	0.3742	0.3352	0.6183
LSEC-GCN	0.8581	**0.1924**	**0.3072**	**0.3869**	0.3537	0.6205
LSEC-LightGCN	**0.8641**	0.1842	0.3022	0.3854	**0.3615**	**0.6380**
LSEC-GAT	0.8611	0.1873	0.3012	0.3867	0.3525	0.6375

Performance Comparison

Table 4: Comparison of the models on LSEC-Large dataset

Model	AUC	MRR	NDCG@10	NDCG@50	Recall@10	Recall@50
NCF	0.8272	0.1767	0.2805	0.3608	0.3101	0.5633
GCN-RS	0.8545	0.1988	0.3219	0.4091	0.3626	0.6365
LightGCN	0.8532	0.2001	0.3346	**0.4184**	0.3741	0.6363
LSEC-GCN	0.8482	**0.2048**	**0.3374**	0.4153	0.3735	0.6233
LSEC-LightGCN	**0.8679**	0.1981	0.3333	0.4157	**0.3752**	**0.6427**

Experiments

Ablation Study

Ablation Study

To investigate the impacts of heterogeneous relations modeling and multi-task training, we conduct some ablation experiments. We use GCN as the example aggregator and run the experiments upon LSEC-Small dataset. The results are in the tables.

Table 5: Comparison of the LSEC-GCN model with different relations and tasks on LSEC-Small dataset (0, 1 and 2 represent the three relations, **User-Item Buying**, **User-Streamer Following** and **Streamer-Item Selling**, respectively)

Relations	Tasks	AUC	MRR	NDCG@10	NDCG@50	Recall@10	Recall@50
0	0	0.8440	0.1835	0.2862	0.3705	0.3363	0.6078
0, 1	0	0.8449	0.1864	0.2923	0.3731	0.3389	0.6036
0, 2	0	0.8453	0.1748	0.2749	0.3553	0.3270	0.5877
0, 1, 2	0	0.8547	0.1915	0.3057	0.3844	0.3509	0.6146
0, 1, 2	0, 1	**0.8581**	**0.1924**	**0.3072**	**0.3869**	**0.3537**	**0.6205**

Conclusion

Conclusion

Problem

How to leverage the tripartite interaction information in live stream E-Commerce to improve product recommendation?

Problem

How to leverage the tripartite interaction information in live stream E-Commerce to improve product recommendation?

Solving Process

Problem

How to leverage the tripartite interaction information in live stream E-Commerce to improve product recommendation?

Solving Process

1. Propose **streamers' influence** from data analysis.

Conclusion

Problem

How to leverage the tripartite interaction information in live stream E-Commerce to improve product recommendation?

Solving Process

1. Propose **streamers' influence** from data analysis.
2. Propose LSEC-GNN framework.

Conclusion

Problem

How to leverage the tripartite interaction information in live stream E-Commerce to improve product recommendation?

Solving Process

1. Propose **streamers' influence** from data analysis.
2. Propose LSEC-GNN framework.
3. Verify the effectiveness of LSEC-GNN on real-world live stream datasets.

Conclusion

Problem

How to leverage the tripartite interaction information in live stream E-Commerce to improve product recommendation?

Solving Process

1. Propose **streamers' influence** from data analysis.
2. Propose LSEC-GNN framework.
3. Verify the effectiveness of LSEC-GNN on real-world live stream datasets.
4. Investigate the impacts of heterogeneous relations modeling and multi-task training with ablation study.

Thanks!