Interactions Expressed in Sequences and Graphs

Sequential Recommendations and GNN-based RS

Klaudia Balcer

Advanced Data Mining

Contents

Sequential Recommendations

Sequential Recommendations

GRU4Rec

GRU

much to update the hidden state of the unit. The activation of the GRU is a linear interpolation between the previous activation and the candidate activation $\hat{\mathbf{h}}_\mathbf{t}$:

$$\mathbf{h_t} = (1 - \mathbf{z_t})\mathbf{h_{t-1}} + \mathbf{z_t}\hat{\mathbf{h}}_\mathbf{t} \tag{2}$$

where the update gate is given by:

$$\mathbf{z_t} = \sigma(W_z\mathbf{x_t} + U_z\mathbf{h_{t-1}}) \tag{3}$$

while the candidate activation function $\hat{\mathbf{h}}_\mathbf{t}$ is computed in a similar manner:

$$\hat{\mathbf{h}}_\mathbf{t} = \tanh\left(W\mathbf{x_t} + U(\mathbf{r_t} \odot \mathbf{h_{t-1}})\right) \tag{4}$$

and finaly the reset gate $\mathbf{r_t}$ is given by:

$$\mathbf{r_t} = \sigma(W_r\mathbf{x_t} + U_r\mathbf{h_{t-1}}) \tag{5}$$

GRU4Rec

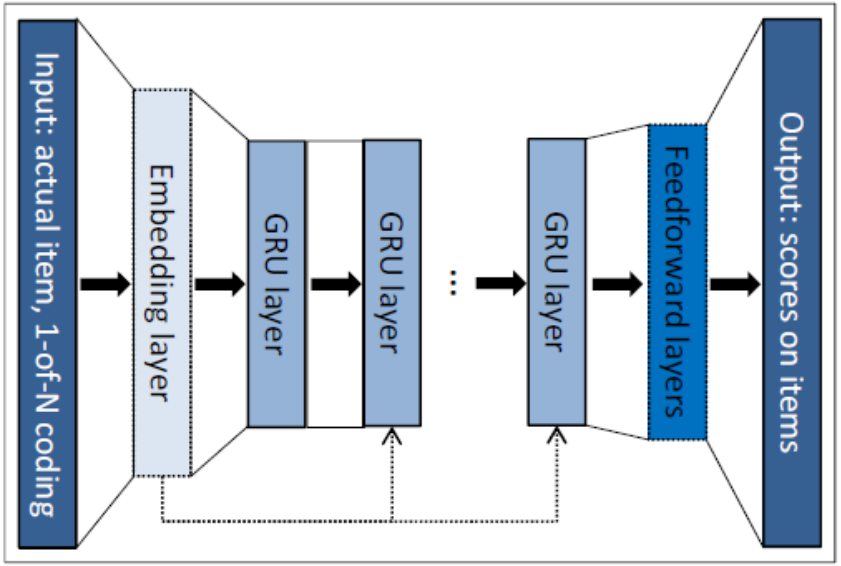
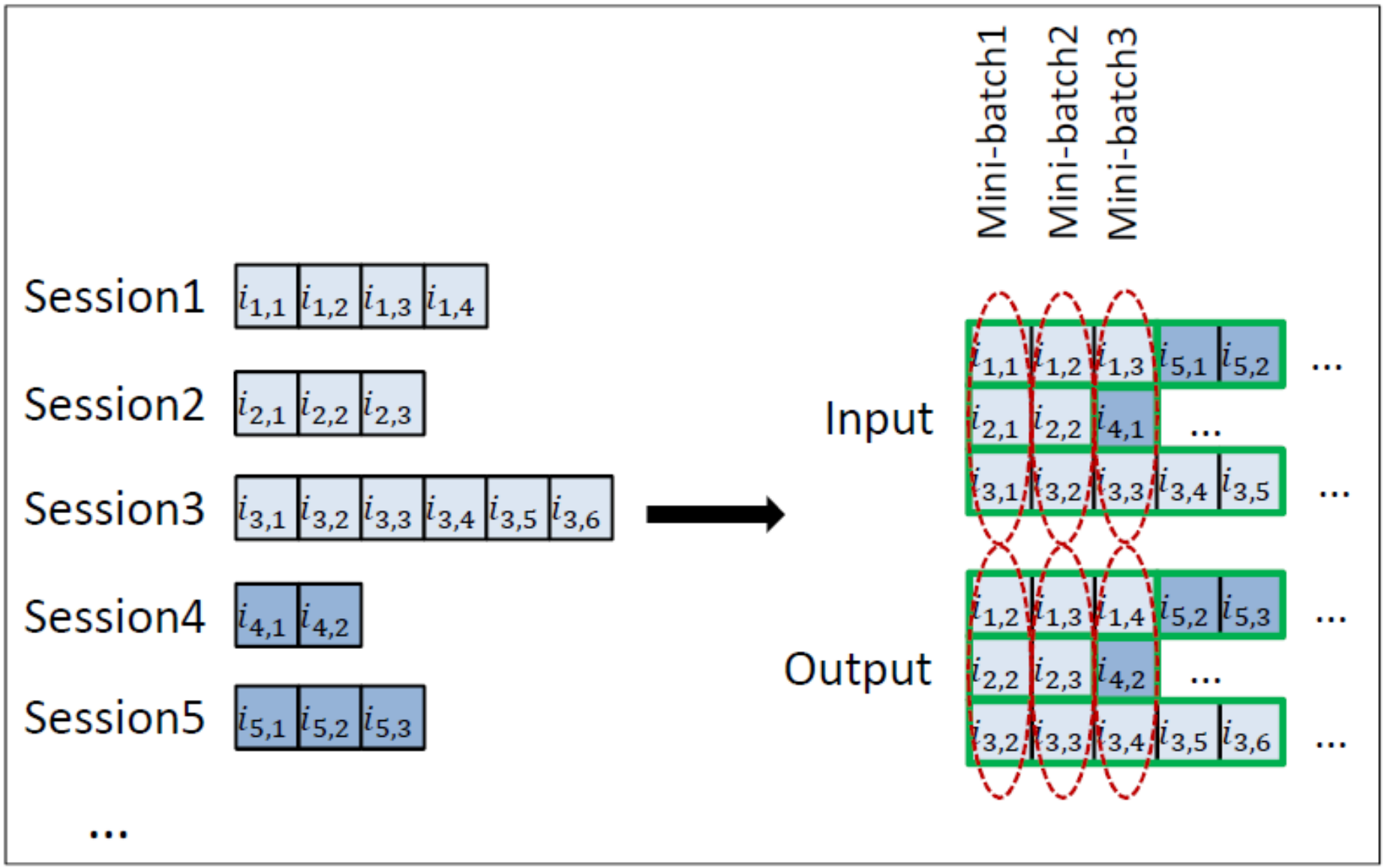


Figure 1: General architecture of the network. Processing of one event of the event stream at once.

- session-based recommendations (when matrix factorization is not suitable)

https://arxiv.org/abs/1511.06939

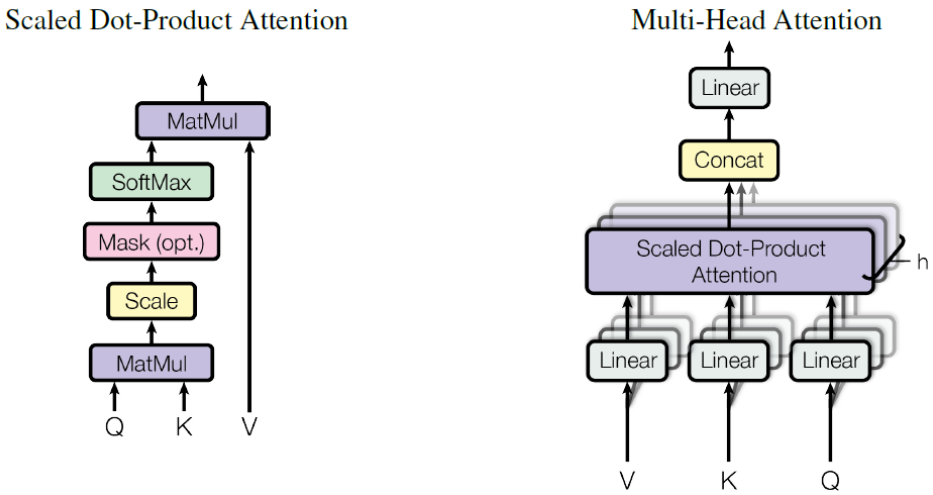
GRU4Rec


Mini-batch1
Mini-batch2
Mini-batch3
Session1
Session2
Session3
Session4
Session5
...
Input
Output

Sequential Recommendations

SASRec

Transformer



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Q, K V - Queries, Keys, and Values matrices
order in sequence –> positional embeddings

https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

Self-Attentive Sequential Recommendation

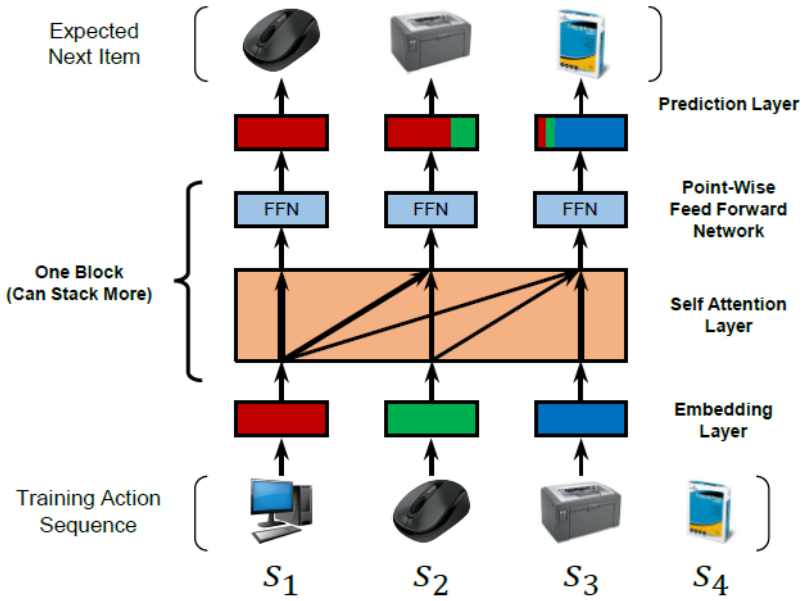


Figure 1: A simplified diagram showing the training process of SASRec. At each time step, the model considers all previous items, and uses attention to 'focus on' items relevant to the next action.

The goal of our work is to balance [. . .], by proposing a sequential model based on self-attention (SASRec) that allows us to capture ***long-term semantics*** *(like an RNN), but, using an attention mechanism, makes its* ***predictions based on relatively few actions*** *(like an MC).*

https://arxiv.org/abs/1808.09781

SASRec

- Embeddings Layer (fixed length: cutting/padding):

$$\hat{\mathrm{V}}_s = \begin{bmatrix} \mathrm{V}_{s_1} + \mathrm{P}_1 \\ \mathrm{V}_{s_2} + \mathrm{P}_2 \\ \vdots \\ \mathrm{V}_{s_n} + \mathrm{P}_n \end{bmatrix}$$

  Positional Embeddings P are trainable.
- Attention block:
  - Self-Attention Layer:

$$\mathrm{S} := \mathrm{SA}(\hat{\mathrm{V}}_s) = \text{Attention}(Q : \hat{\mathrm{V}}_s \mathrm{W}^Q, K : \hat{\mathrm{V}}_s \mathrm{W}^K, V : \hat{\mathrm{V}}_s \mathrm{W}^V)$$

    Causality: modify the attention by forbidding all links between Q_i and K_j if $j > i$.
  - Feed forward network: linear -> ReLU -> linear on S
  - To prevent overfitting authors use: layer normalization, dropout and residual connections.
- The predictions are made on the basis of the dot product.
- Binary cross-entropy loss (1 random negative sample).
- Attention weights give us some interpretability (see the paper).

Graph-based RS

Graph-based RS

LightGCN

Vanilla Gradient Neural Network (recap)

- $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X^{\mathcal{V}}, X^{\mathcal{E}})$ - a graph is a tuple of vertices, edges, vertices features and edges features.
- The target is to provide meaningful node embedding matrix V.
- In each layer we have two major phases:
  1. Aggregation:
  $$\mathrm{m}_v^{(l)} = f\left(X_v^{\mathcal{V}}, X_{\mathcal{N}_v}^{\mathcal{E}}, X_{\mathcal{N}_v}^{\mathcal{V}}, \mathrm{V}_{\mathcal{N}_v}^{(l-1)}\right)$$
  2. Message passing:
  $$\mathrm{v}_v^{(l)} = g\left(\mathrm{v}_v^{(l-1)}, \mathrm{m}_v^{(l)}\right)$$

https://cs.mcgill.ca/ wlh/comp766/files/chapter4_draft_mar29.pdf

Graph Convolutional Network (recap)

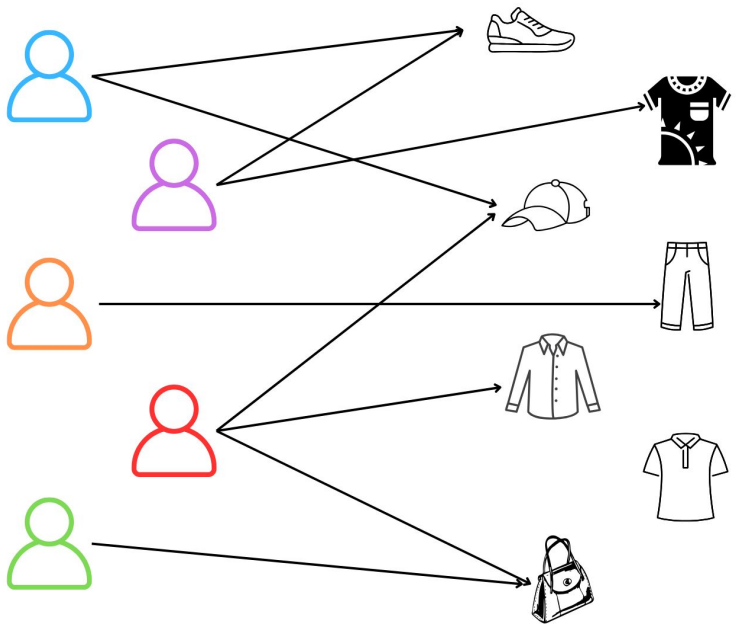
$$\mathrm{V}^{(l)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}\mathrm{V}^{(l-1)}W^{(l)}\right)$$

- $\tilde{A} = A + I_N$: the adjacency matrix of the undirected graph $\mathcal{G}$ with added self-connections (I_N – the identity matrix).
- $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$: the diagonal degree matrix of $\tilde{A}$.
- $W^{(l)}$: a layer-specific trainable weight matrix.
- $\sigma(\cdot)$: an activation function, such as the ReLU function, $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$.
- $\mathrm{V}^{(l)} \in \mathbb{R}^{N \times D}$: the matrix of activations in the l^{th} layer.
- $\mathrm{V}^{(0)} = X^{\mathcal{V}}$: the input feature matrix.

$$\mathrm{V}_v^{(l)} = \sigma\left(\frac{1}{\sqrt{|\mathcal{N}_v|+1}}\sum_{w \in \mathcal{N}_v \cup \{v\}} \frac{1}{\sqrt{|\mathcal{N}_w|+1}}\mathrm{V}_w^{(l-1)}W^{(l)}\right)$$

https://arxiv.org/pdf/1609.02907

Data Structure



- $\mathcal{V} = (\mathcal{U}, \mathcal{I})$ – we have nodes for both users and items
- $\mathcal{E} = \{(u, i) : \text{ user } u \text{ interacted with item } i\}$
- $A = \begin{bmatrix} 0 & R \\ R^\top & 0 \end{bmatrix}$ – adjacency matrix, R – binary interaction matrix
- We need both item and feature embeddings. We use the standard procedure with dot-product scores and BPR loss.

LightGCN - Key Idea

- ~~node and edge features~~
- ~~non-linear activations~~
- neighbourhood aggregation

$$\mathrm{V}^{(l)} = \left(D^{-\frac{1}{2}} A D^{-\frac{1}{2}}\right) \mathrm{V}^{(l-1)}$$

- self-connection obtained after going through all layers:

$$V = \frac{1}{L+1} \sum_{l=0}^{L} V^{(l)}$$

- the only trainable parameters are embedding vectors

https://arxiv.org/abs/2002.02126 In the paper you can also find:

-> an ablation study on previous Neural Grapf Collaborative Filtering model

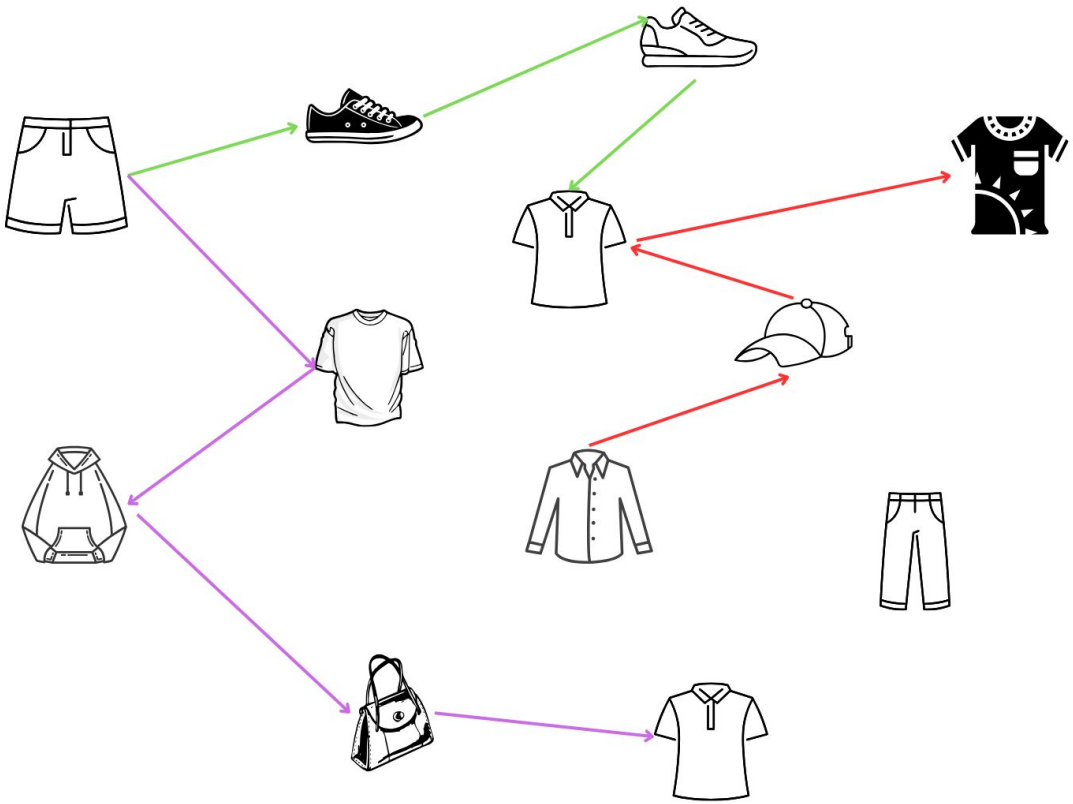
-> comparison to similar architectures

-> experiments

Graph-based RS

[uncovered] SR-GNN, TAGNN

Data Structure



- $\mathcal{V} = \mathcal{I}$ – we consider only items as nodes as the users are anonymous (and cannot *come back*)
- an edge connect two items consecutive in a sequence

SR-GNN model architecture

$$\mathrm{a}_{m,j}^{(g)} = \mathrm{A}_{m,j}[\mathrm{v}_{m,1}^{(g-1)}, \mathrm{v}_{m,2}^{(g-1)}, \ldots, \mathrm{v}_{m,L_m-1}^{(g-1)}]^\top \mathrm{H} + \mathrm{b}, \tag{1}$$

$$\mathrm{z}_{m,j}^{(g)} = \sigma\left(\mathrm{W}_z \mathrm{a}_{m,j}^{(g)} + \mathrm{U}_z \mathrm{v}_j^{(g-1)}\right), \tag{2}$$

$$\mathrm{r}_{m,j}^{(g)} = \sigma\left(\mathrm{W}_r \mathrm{a}_{m,j}^{(g)} + \mathrm{U}_r \mathrm{v}_j^{(g-1)}\right), \tag{3}$$

$$\tilde{\mathrm{v}}_j^{(g)} = \tanh\left(\mathrm{W}_o \mathrm{a}_{m,j}^{(g)} + \mathrm{U}_o\left(\mathrm{r}_{m,j}^{(g)} \odot \mathrm{v}_j^{(g-1)}\right)\right), \tag{4}$$

$$\mathrm{v}_j^{(g)} = \left(1 - \mathrm{z}_{m,j}^{(g)}\right) \odot \mathrm{v}_j^{(g-1)} + \mathrm{z}_{m,j}^{(g)} \odot \mathrm{v}_j^{(g)}, \tag{5}$$

where $\mathrm{A}_{m,j} \in \mathbb{R}^{1\times 2N}$ is the vector of two columns of the adjacency matrix corresponding to the jth row item, $\mathrm{z}_{m,j}^{(g)}$ is the reset gate, $\mathrm{r}_{m,j}^{(g)}$ is the update gate, $[\ldots]$ is concatenation, σ is the sigmoid function, and $\odot$ is the element-wise multiplication.

https://dl.acm.org/doi/abs/10.1609/aaai.v33i01.3301346

SR-GNN model architecture

Then, we need to provide a session embedding s_m. It will be a transformation of a global and a local embedding. The local embedding is just the embedding of the last item in the session. The global embedding is an aggregation of all embeddings in the session. The whole procedure can be described as follows:

$$\mathrm{s}_m^l = \mathrm{v}_{m,L_m-1}, \tag{6}$$

$$\alpha_j = \mathrm{q}^T \sigma(\mathrm{W}_1 \mathrm{v}_{m,L_m-1} + \mathrm{W}_2 \mathrm{v}_{m,j} + \mathrm{c}), \tag{7}$$

$$\mathrm{s}_m^g = \sum_{j=1}^{L_m-1} \alpha_j \mathrm{v}_{m,j}, \tag{8}$$

$$\mathrm{s}_m = \mathrm{W}_3[\mathrm{s}_m^l, \mathrm{s}_m^g], \tag{9}$$

where $\mathrm{q}, \mathrm{W}_1, \mathrm{W}_2$ control the weights, W_3 reduces the dimensionality, all of which are learnable parameters.

descriptions from my master thesis

[uncovered] Augmentations for Interactions

Augmentations in Recommender Systems

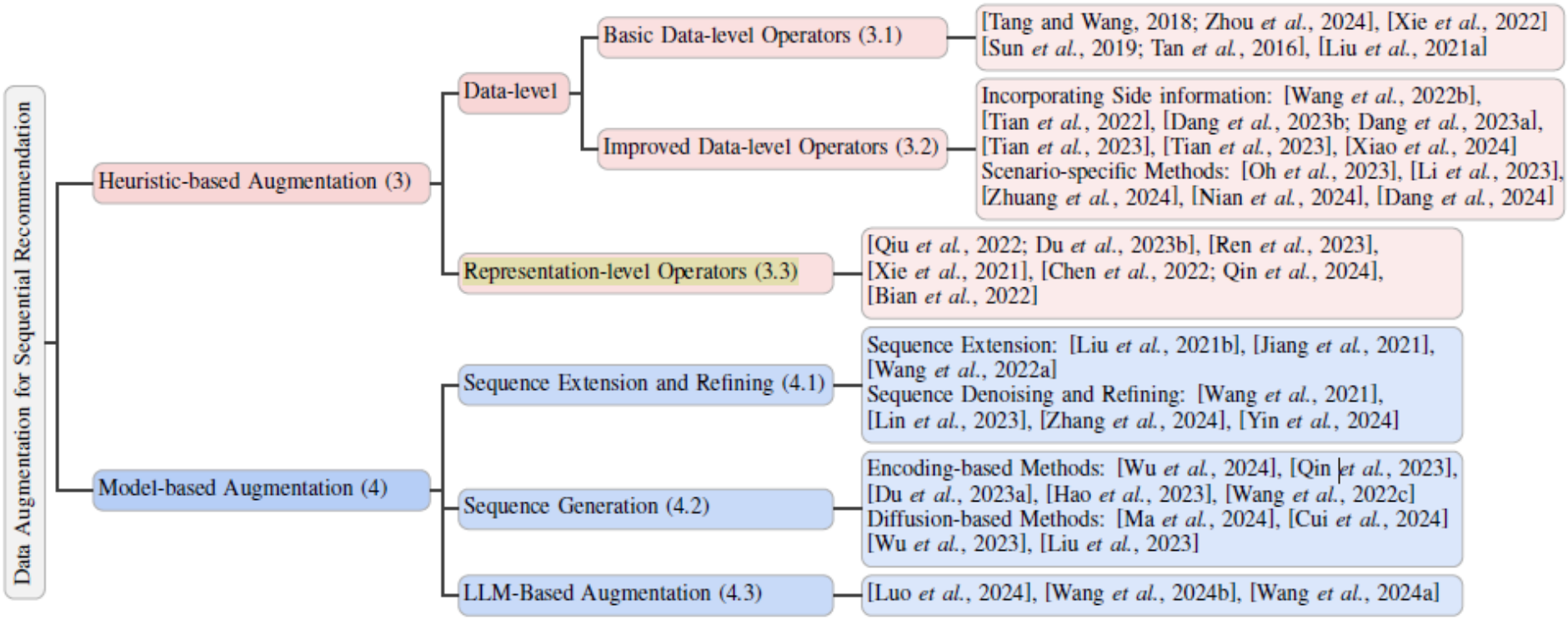


Figure 1: The taxonomy of data augmentation for sequential recommendation.

https://arxiv.org/abs/2409.13545

[uncovered] Augmentations for Interactions

[uncovered] SimGCL

SimGCL – noise

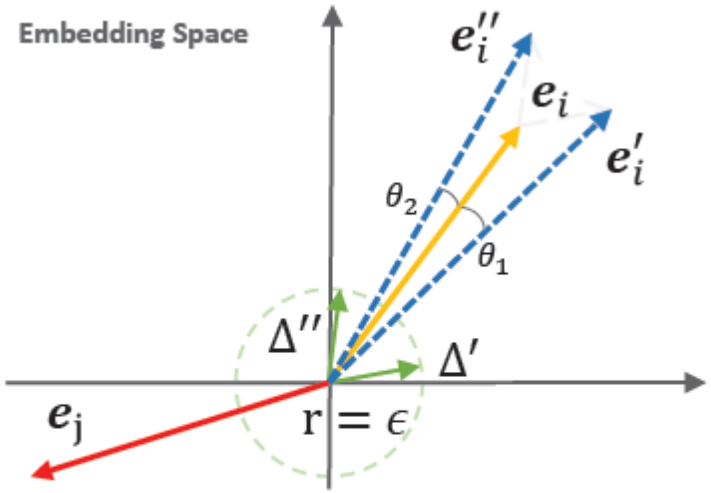


Figure 3: An illustration of the proposed random noise-based data augmentation in $\mathbb{R}^2$.

- noise augmentations result in even embedding distribution

https://dl.acm.org/doi/10.1145/3477495.3531937

InfoNCE loss function

$$\mathcal{L}_{joint} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{cl}, \tag{1}$$

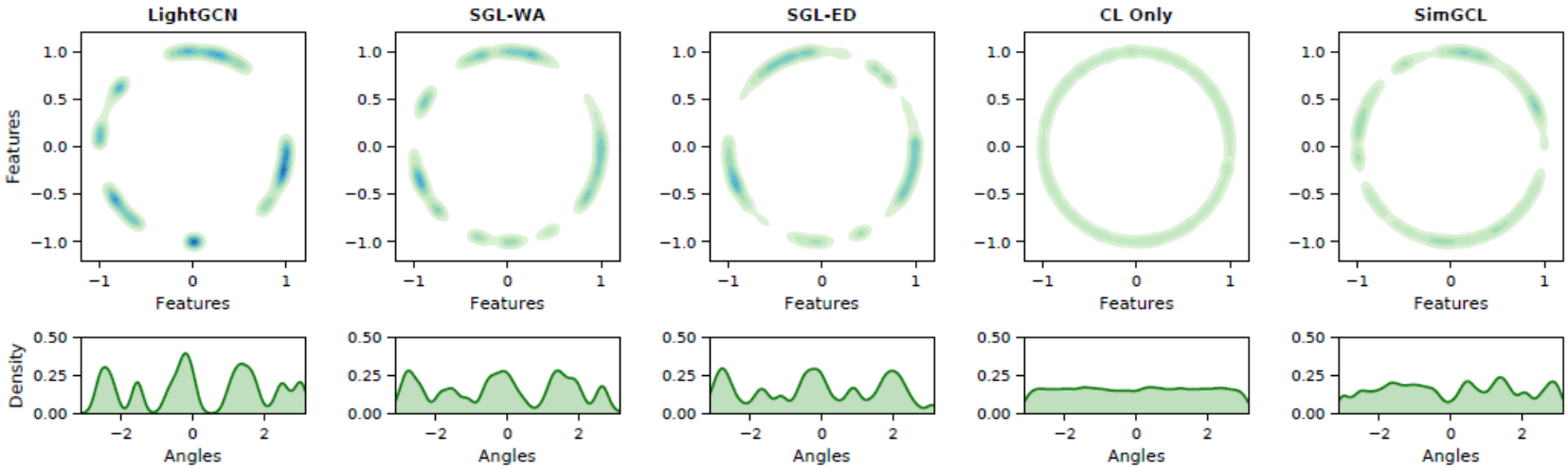
which consists of two losses: recommendation loss $\mathcal{L}_{rec}$ and CL loss $\mathcal{L}_{cl}$. The InfoNCE in SGL is formulated as:

$$\mathcal{L}_{cl} = \sum_{i \in \mathcal{B}} -\log \frac{\exp(\mathbf{z}_i'^{\top} \mathbf{z}_i'' / \tau)}{\sum_{j \in \mathcal{B}} \exp(\mathbf{z}_i'^{\top} \mathbf{z}_j'' / \tau)}, \tag{2}$$

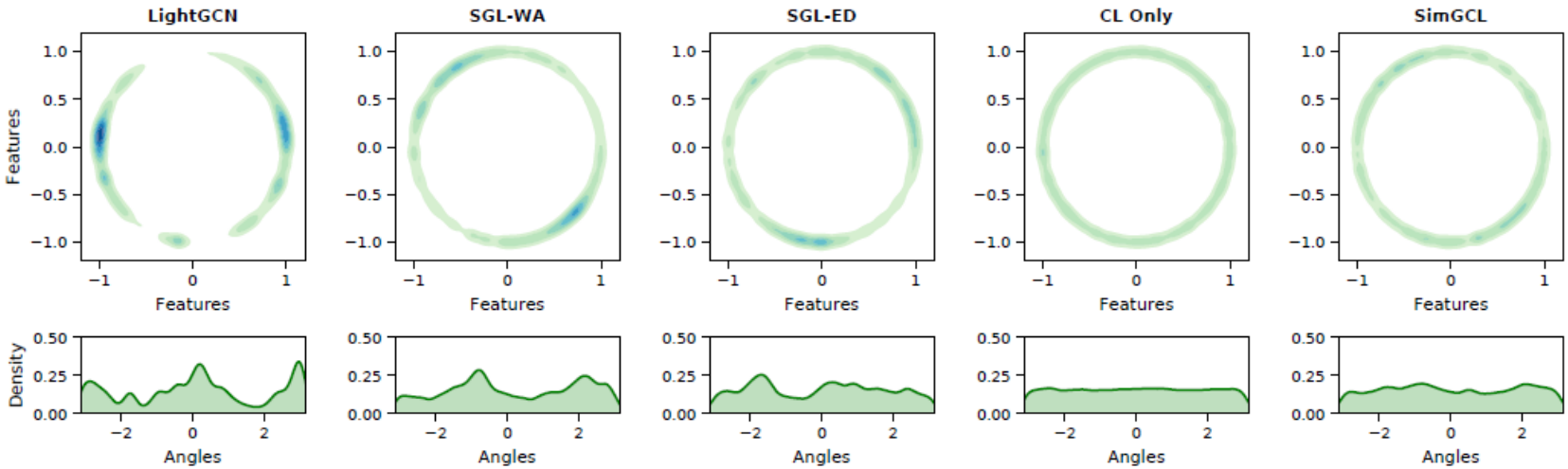
- L_2 normalized embeddings
- differences between contrasitve views
- the key factor in SGL was the loss function (not the augmentations with edge/node dropout)

InfoNCE: https://arxiv.org/abs/1807.03748

Embedding Uniformity



(a) Distribution of item representations learned from the dataset of Yelp2018.



(b) Distribution of item representations learned from the dataset of Amazon-Book.

[uncovered] Augmentations for Interactions

[uncovered] DiffuASR

Diffusion

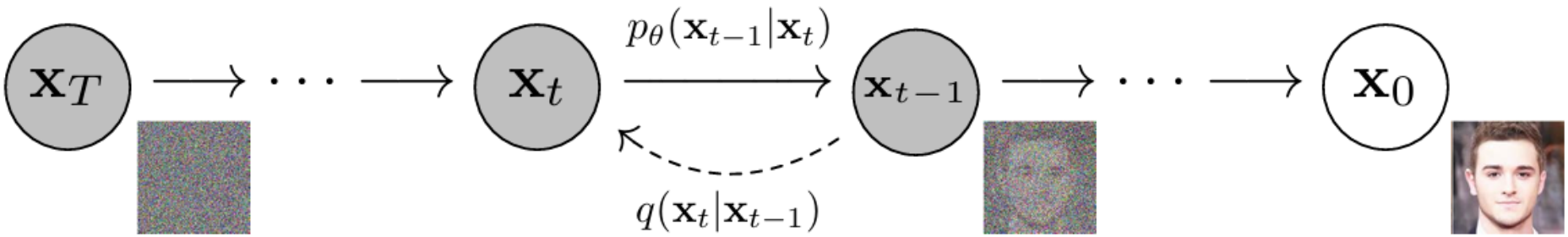


Figure 2: The directed graphical model considered in this work.

DiffuASR

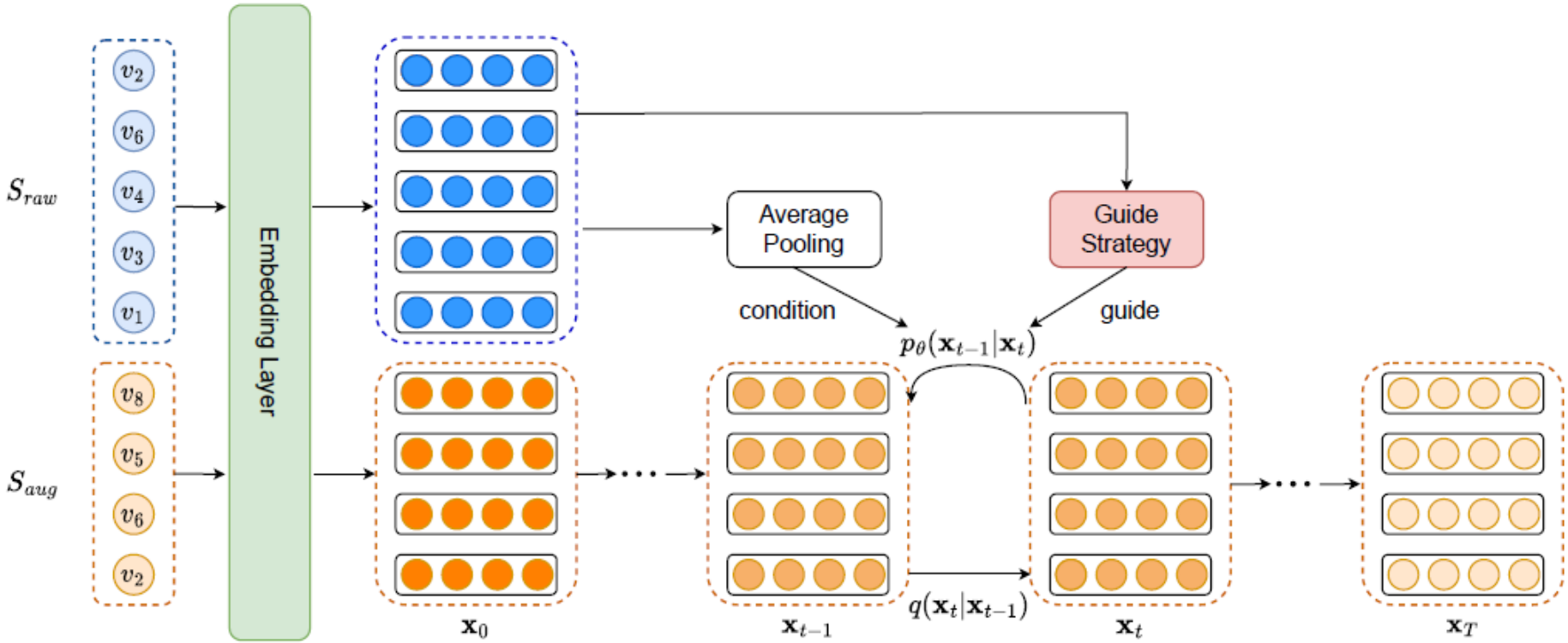


Figure 3: The overview of the proposed Diffusion Augmentation for Sequential Recommendation (DiffuASR).

https://dl.acm.org/doi/10.1145/3583780.3615134

SU-Net

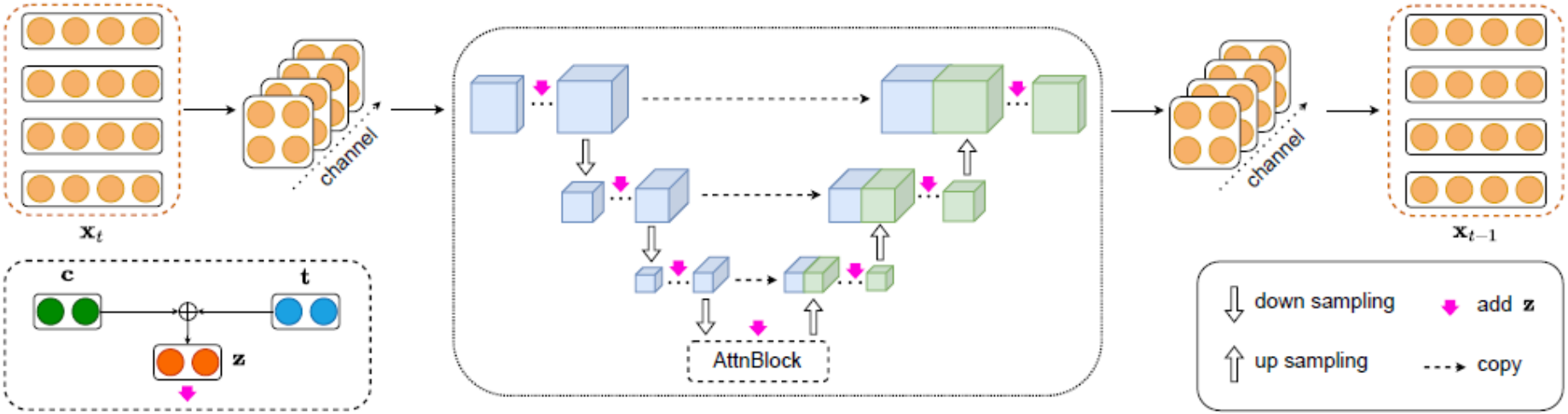


Figure 4: The architecture of the proposed SU-Net.

[uncovered] Matrix Form – revisited

[uncovered] Matrix Form – revisited

[uncovered] DiffRec

DiffRec

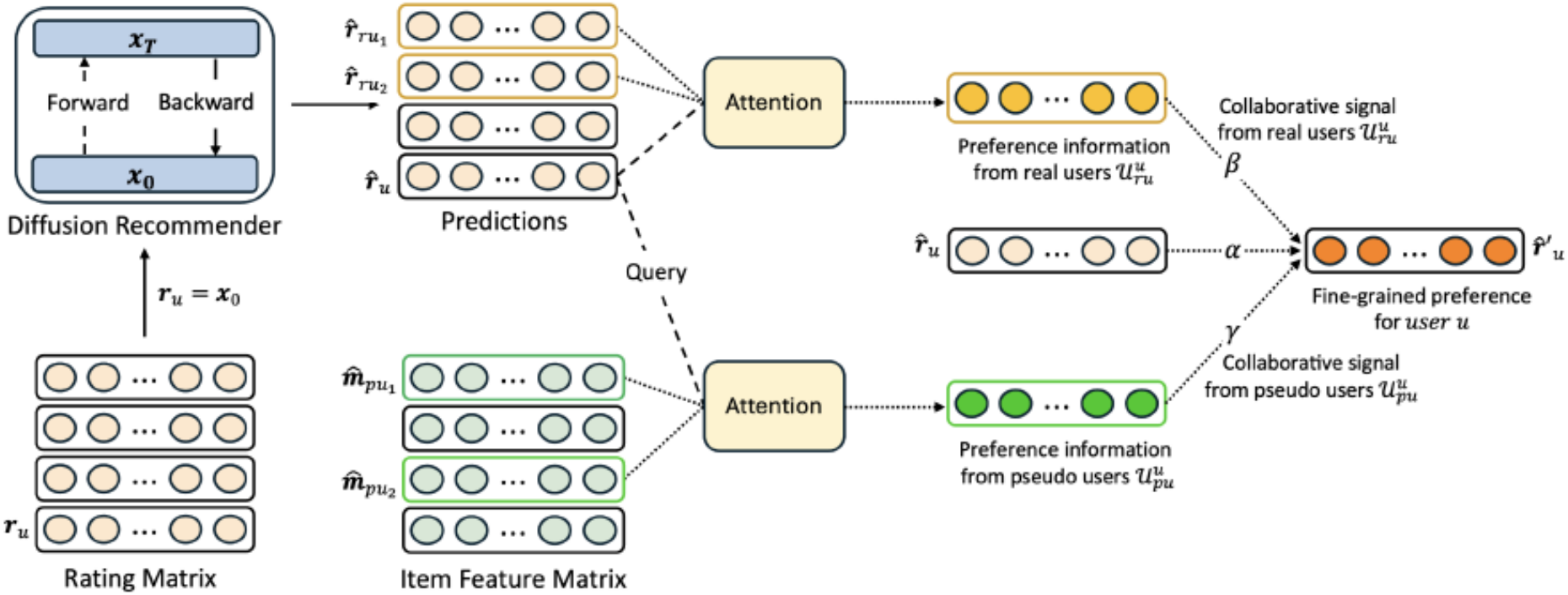


Figure 1: The overview of the proposed Collaborative Diffusion Model for Recommender System (CDiff4Rec).

https://arxiv.org/abs/2501.18997