Cross-Domain 맛보기

with PPGN : Cross-Domain Recommendation via Preference Propagation GraphNet

2022. 02. 08

박지민(kpdpkp@gmail.com)

What is Cross Domain?

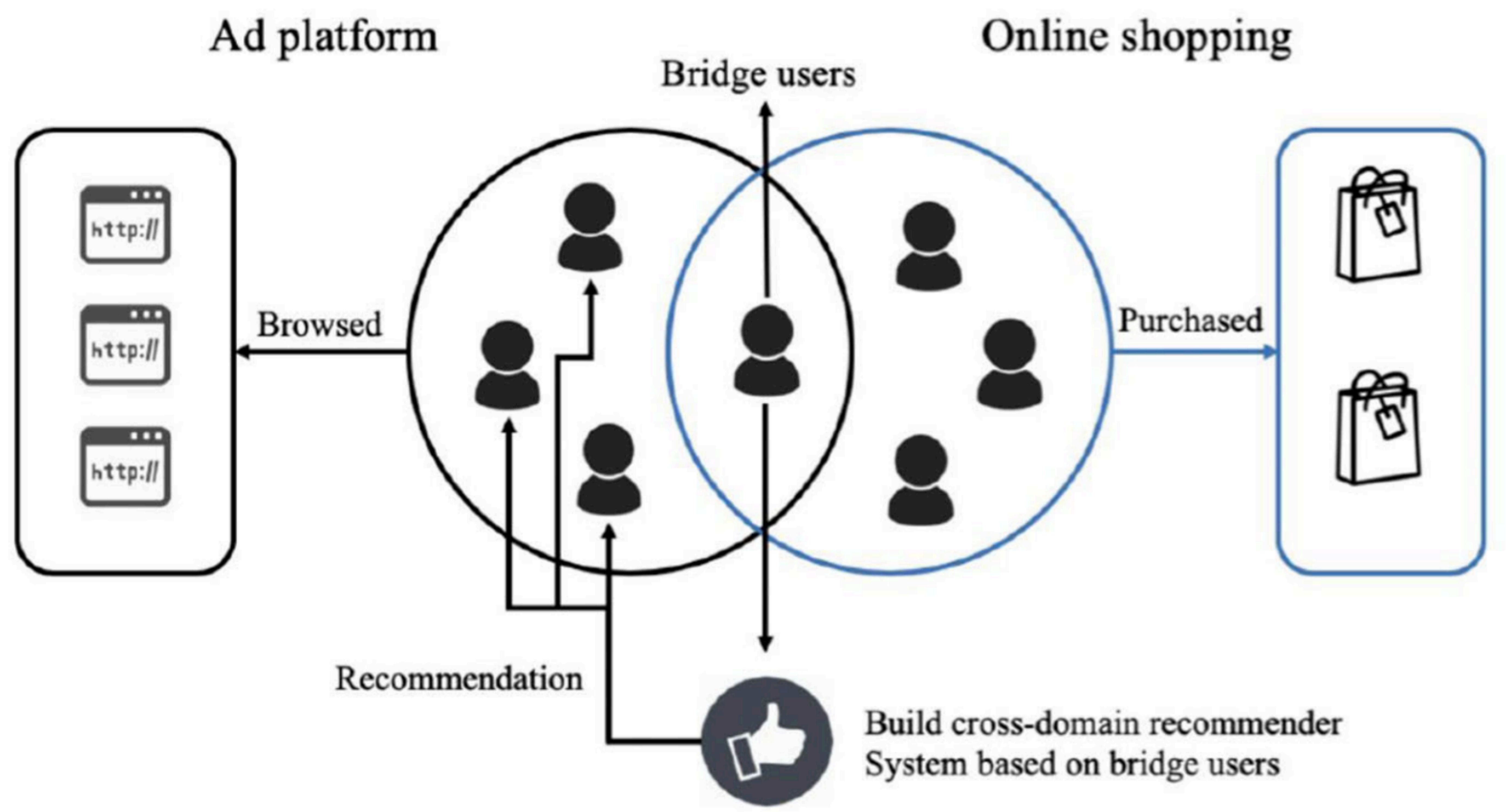


FIGURE 3. Overview of the framework.

Why Cross Domain? (1)

As a Modeler

- Alleviate the sparsity issue
  - 비디오 스트리밍 사업을 하고 있다
  - 웹툰 영역으로 확장하고 싶다
  - 웹툰 데이터가 적다 == Sparsity가 높다
  - 비디오 데이터를 활용하자
  - Kind of Augmentation?

Why Cross Domain? (2)

As a 사장님 <= 🚨주관적인 생각

- 색다른 경험
  - <아이언맨> ==> <마블 코믹스 만화>
  - <인터스텔라> ==> <SF 소설>
  - 드라마 <지금 우리 학교는> ==> 만화 <지금 우리 학교는>

When?

어디까지가 다른 도메인인가?

- 영화 vs TV 드라마
- 클래식 vs Pop vs EDM
- 액션 vs 멜로 vs 전쟁 영화
- 웹툰 vs 쇼핑
- 알 수도 있는 친구 vs 추천 그룹

Why PPGN?

Cross-Domain Recommendation via Preference Propagation GraphNet

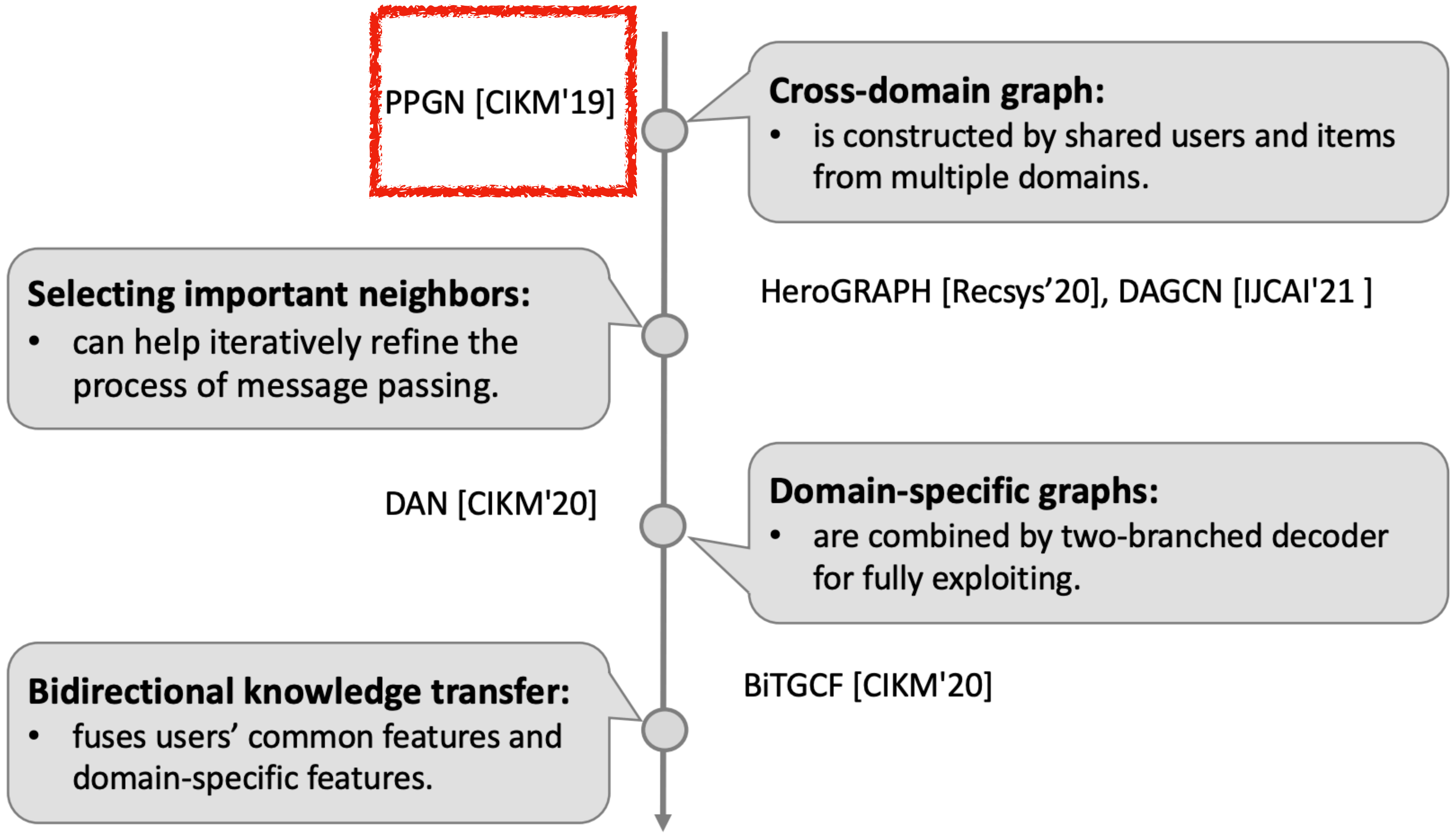


Fig. 16. Illustration of GNN models for cross-domain recommendation.

Table 11. Details of GNN models of cross-domain recommendation.

Model	Graph	Information Transferring
PPGN[204]	cross-domain graph	cross-domain propagation
BiTGCF[92]	domain-specific graphs	common user attributes
DAN[146]	domain-specific graphs	two-branched decoder
HeroGRAPH[30]	cross-domain graph	cross-domain propagation
DAGCN[49]	cross-domain graph	cross-domain propagation

Previous Work

Cross-domain Recommendation Without Sharing User-relevant Data

- Embedding Level Sharing

  => **Model Level**

- No High-Order

  => **with graph**

- Complex. Transfer Learning

  => Joint-Obejctive

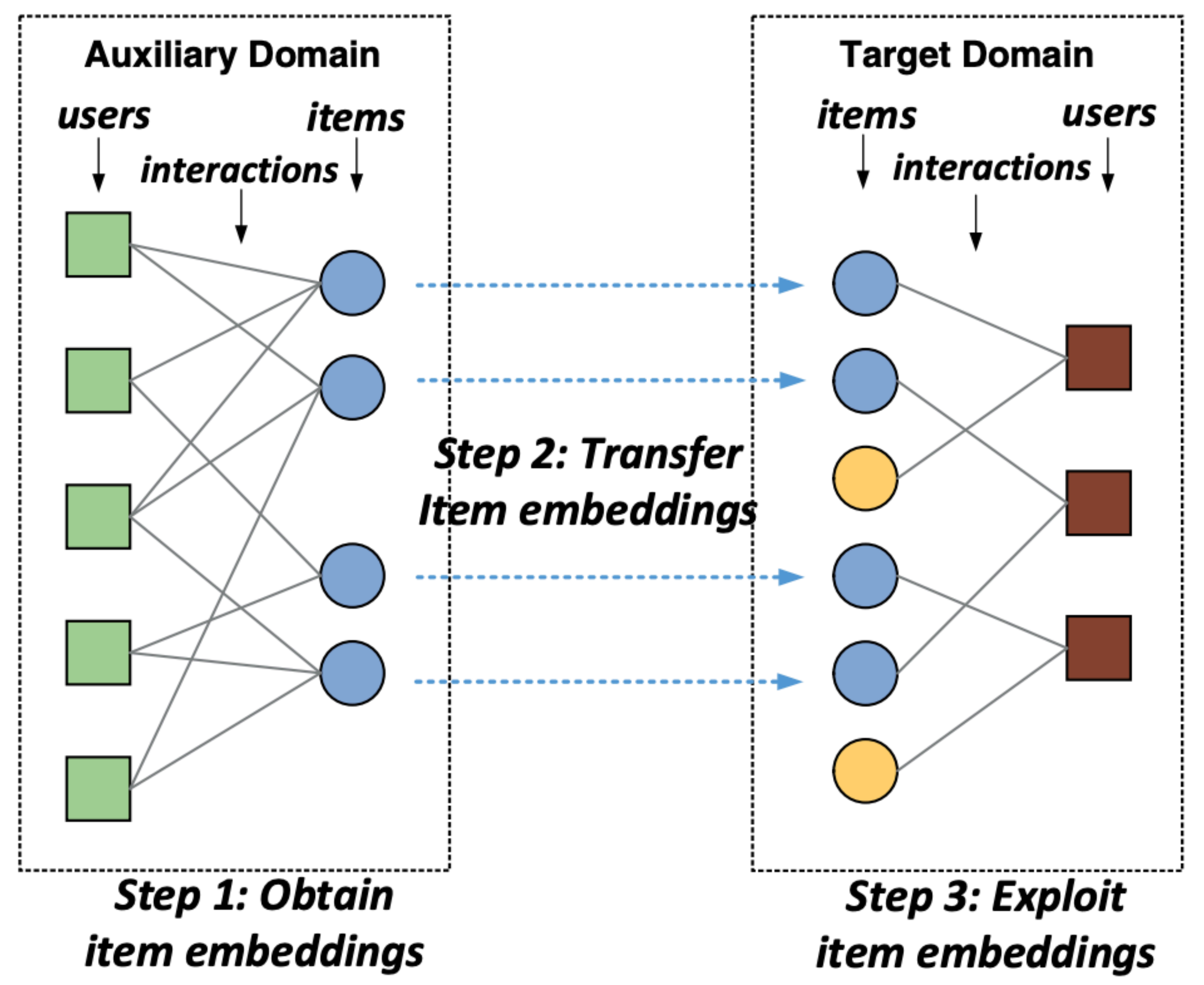


Figure 1: Illustration of our solution for cross-domain recommendation without sharing user-relevant data.

Previous Work

Cross-domain Recommendation Without Sharing User-relevant Data

- single-target CDR(Cross-Domain Reco)
  - from source domain to target domain
- Dual-target CDR
  - mutual utilization of information
- Multi-target CDR

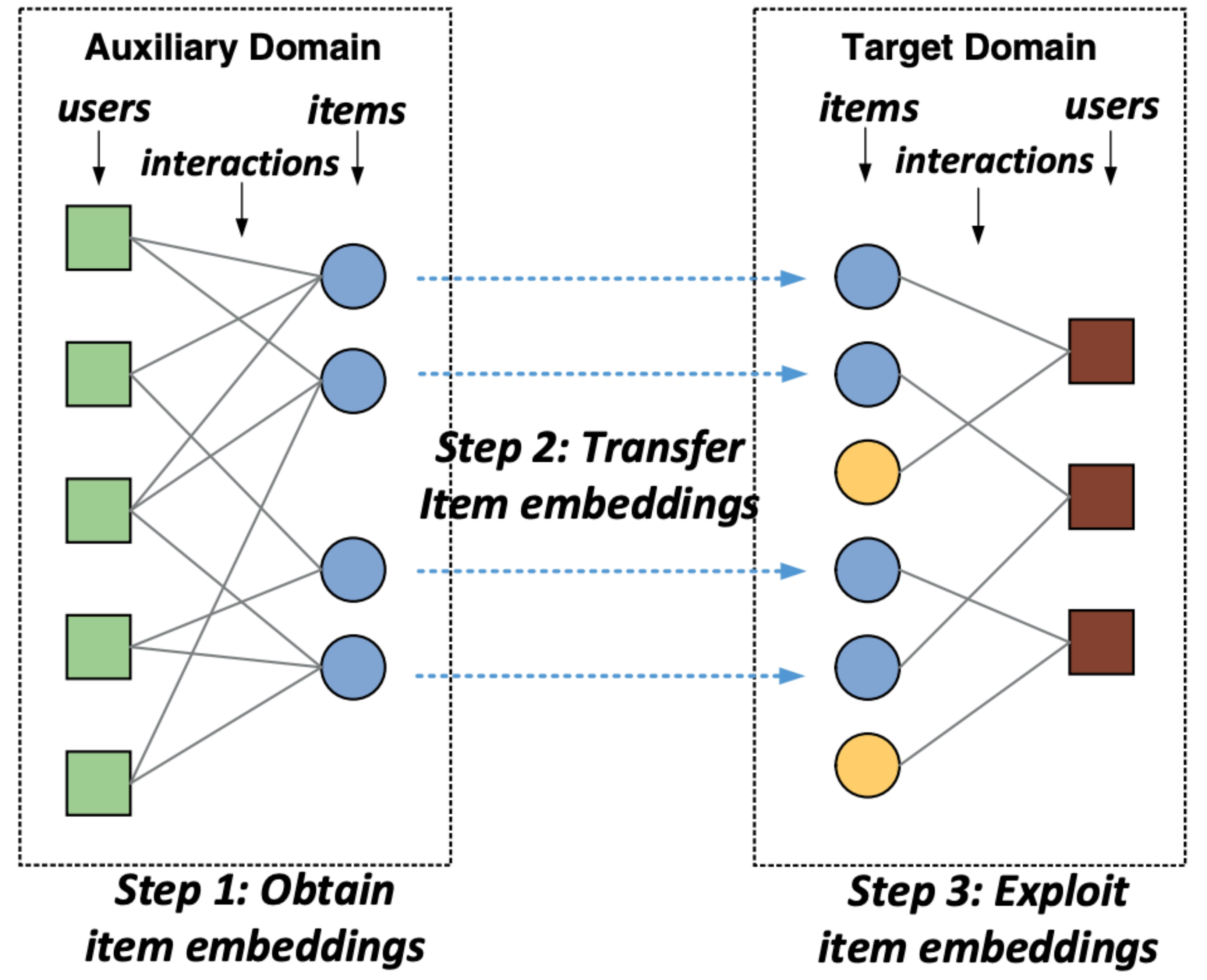


Figure 1: Illustration of our solution for cross-domain recommendation without sharing user-relevant data.

Proposed Architecture

$\mathbb{R}^{(p+n+q)\times(p+n+q)}$, so we get $\hat{\mathbf{A}}$, $\hat{\mathbf{A}} = \mathbf{D}^{-1}(\mathbf{A}+\mathbf{I})$.

$$\mathbf{E_0} = \Bigg[\underbrace{e_1^{i_a}, \cdots, e_p^{i_a}}_{\mathcal{D}_a \text{ items embeddings}}, \overbrace{e_1^{u}, \cdots, e_n^{u}}^{\text{users embeddings}}, \underbrace{e_1^{i_b}, \cdots, e_q^{i_b}}_{\mathcal{D}_b \text{ items embeddings}}\Bigg]^T, \quad (2)$$

$$\mathbf{E_l} = \sigma(\hat{\mathbf{A}}\mathbf{E_{l-1}} W_l + b_l),$$

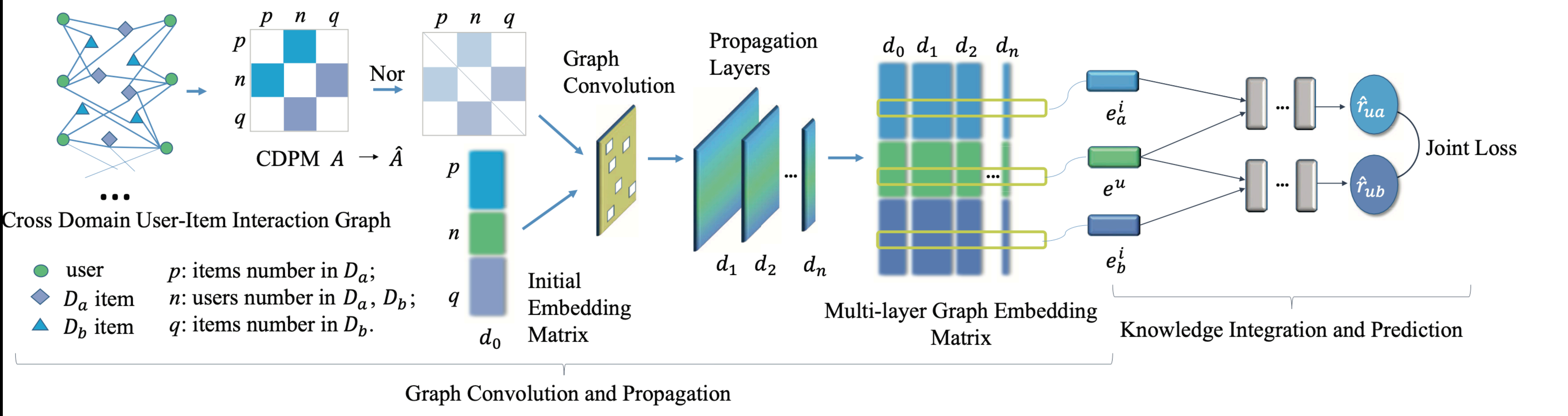
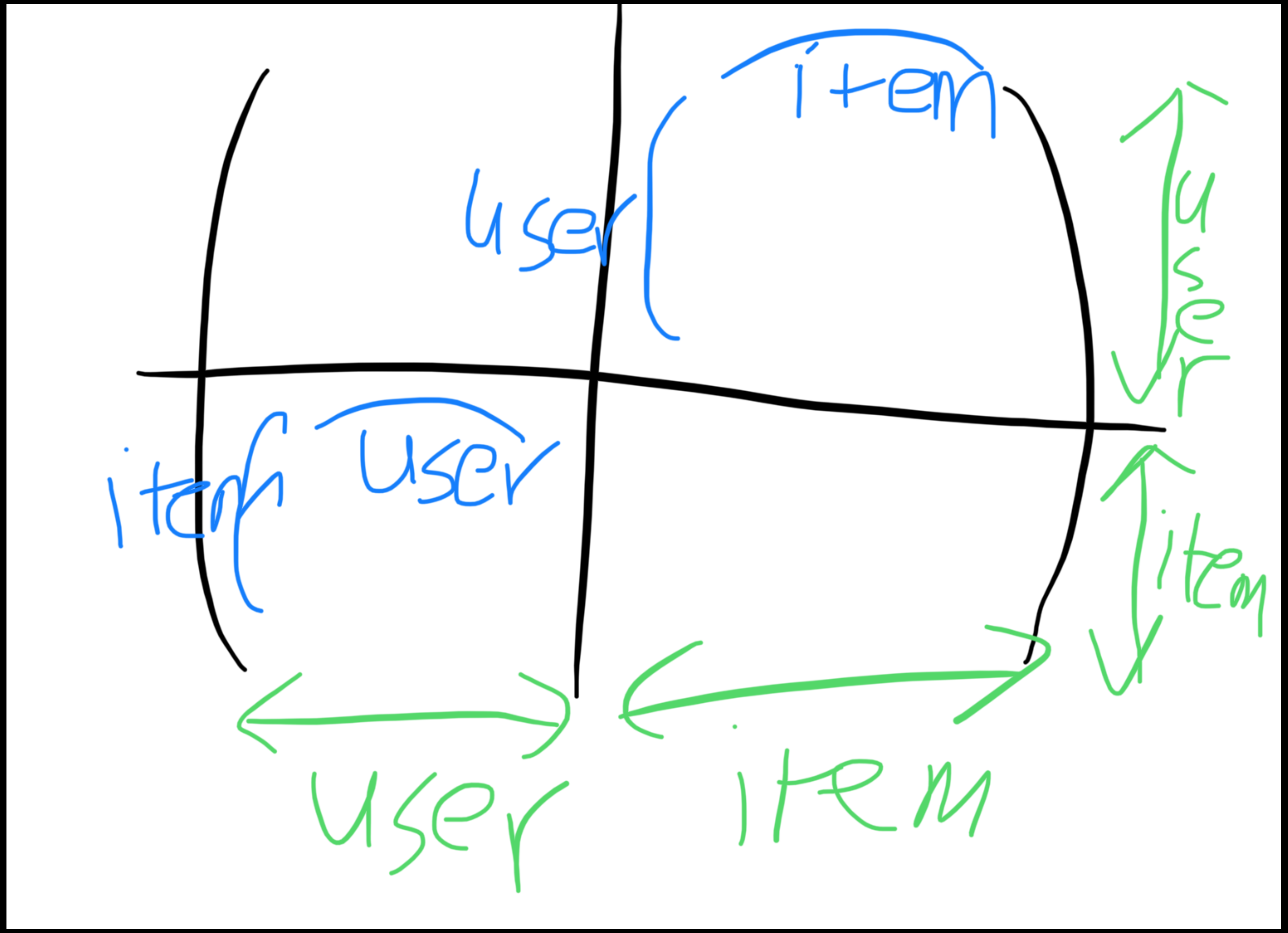


Figure 2: The network architecture of PPGN.

Recap - NGCF

$$\mathbf{E}^{(l)} = \text{LeakyReLU}\Big((\mathcal{L} + \mathbf{I})\mathbf{E}^{(l-1)}\mathbf{W}_1^{(l)} + \mathcal{L}\mathbf{E}^{(l-1)} \odot \mathbf{E}^{(l-1)}\mathbf{W}_2^{(l)}\Big), \quad (7)$$

$$\mathcal{L} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \text{ and } \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^\top & \mathbf{0} \end{bmatrix}, \quad (8)$$



자연스럽게 확장하기

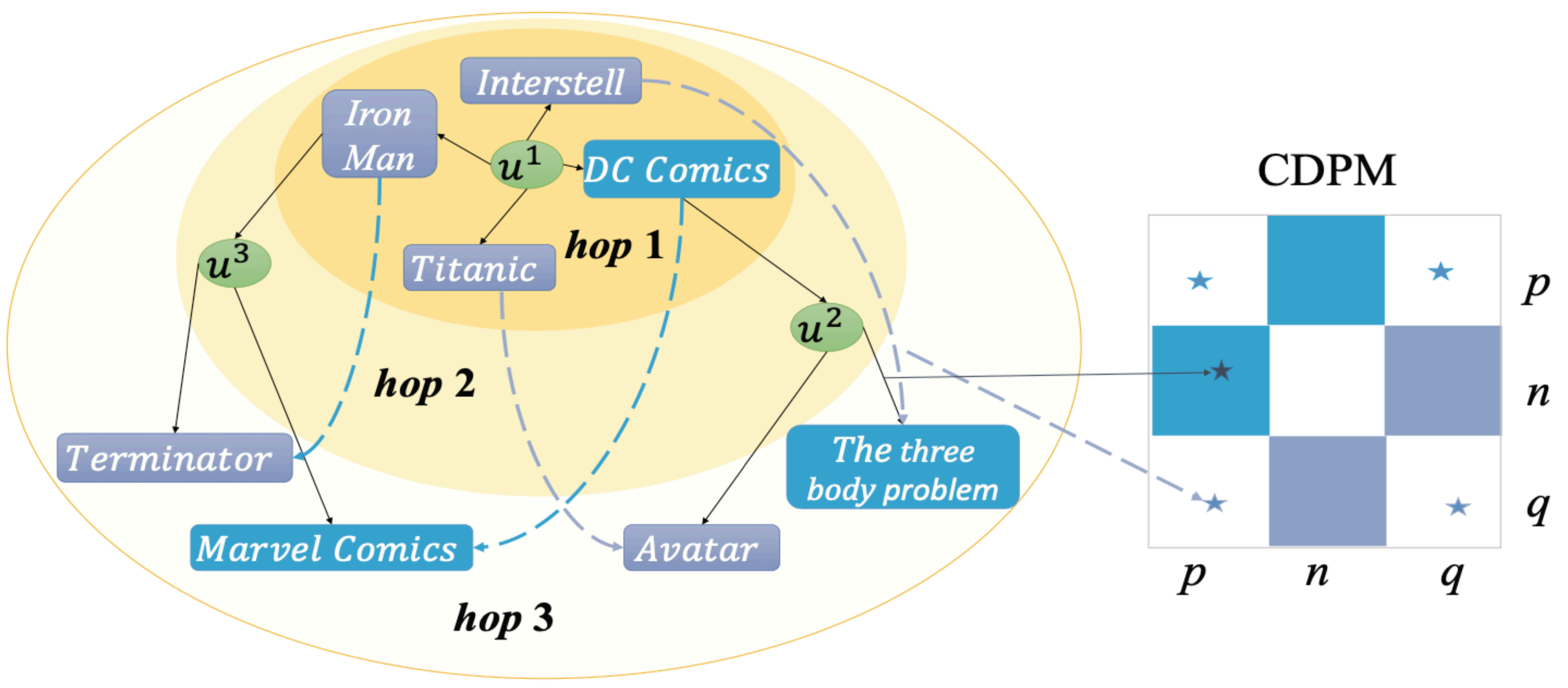


Figure 1: An illustration of the preference propagation via the joint interaction graph. Items in different colors are from different domains. The graph is further reconstructed as the sparse Cross-Domain Preference Matrix (CDPM) on the right, which can be processed by the model immediately. Solid lines denote the known user behaviors, while dotted lines denote the potential recommendation. Through multi-hop propagation, our model can capture the transitions of the user preference and make better predictions.

```
    plain_adj_mat = sp.dok_matrix(
        (
            num_items_s + num_users + num_items_t,
            num_items_s + num_users + num_items_t,
        ),
        dtype=np.float32,
    ).tolil()
    plain_adj_mat[num_items_s : num_items_s + num_users, :num_items_s] = R_s
    plain_adj_mat[:num_items_s, num_items_s : num_items_s + num_users] = R_s.T
    plain_adj_mat[
        num_items_s : num_items_s + num_users, num_items_s + num_users :
    ] = R_t
    plain_adj_mat[
        num_items_s + num_users :, num_items_s : num_items_s + num_users
    ] = R_t.T
    plain_adj_mat = plain_adj_mat.todok()

    norm_adj_mat = normalized_adj_single(
        plain_adj_mat + sp.eye(plain_adj_mat.shape[0])
    )

    sp.save_npz(norm_adj_path, norm_adj_mat)

print("Get adjacent mats successfully.")

return norm_adj_mat
```

https://github.com/WHUIR/PPGN/blob/master/runner/train.py#L208

Training - Loss

$$
\begin{aligned}
\mathcal{L} &= \mathcal{L}_{ua} + \mathcal{L}_{ub} + \mathcal{L}_{reg} \\
&= -\sum_{(i_a, u, i_b) \in T} r_{ua} \log \hat{r}_{ua} + (1 - r_{ua}) \log (1 - \hat{r}_{ua}) \\
&\quad + r_{ub} \log \hat{r}_{ub} + (1 - r_{ub}) \log (1 - \hat{r}_{ub}) + \lambda \sum |\Theta|
\end{aligned} \tag{5}
$$

Training Strategy (1) - Split

Considering the size of $\hat{A}$ is generally huge, always up to hundreds of thousands, it's hard to conduct matrix multiplication between $\hat{A}$ and E_{l-1} in the graph convolution layer in one go. For the sake of scalability, we propose to split $\hat{A}$ into rows to get multiple sub-matrices $\hat{A}_i$ and perform multiplication operation with E_{l-1} respectively, then concatenate the results back to one matrix:

$$\hat{A} = \left[\hat{A}_0, \hat{A}_1, \cdots, \hat{A}_{sn}\right]$$
$$\hat{A}E_{l-1} = \left[\hat{A}_0 E_{l-1}, \cdots, \hat{A}_{sn} E_{l-1}\right]$$

```python
def _split_A_hat(self, X):
    fold_len = math.ceil((X.shape[0]) / self.n_fold)
    A_fold_hat = [
        self._convert_sp_mat_to_sp_tensor(
            X[i_fold * fold_len : (i_fold + 1) * fold_len]
        )
        for i_fold in range(self.n_fold)
    ]

    return A_fold_hat
```

https://github.com/WHUIR/PPGN/blob/master/runner/model.py#L81

```python
def creat_gcn_embedd(self):
    A_fold_hat = self._split_A_hat(self.norm_adj_mat)
    embeddings = tf.concat(
        [
            self.all_weights["item_embeddings_s"],
            self.all_weights["user_embeddings"],
            self.all_weights["item_embeddings_t"],
        ],
        axis=0,
    )
    all_embeddings = [embeddings]

    for k in range(len(self.layers_plus) - 1):
        temp_embedd = [
            tf.sparse_tensor_dense_matmul(A_fold_hat[f], embeddings)
            for f in range(self.n_fold)
        ]

        embeddings = tf.concat(temp_embedd, axis=0)
```

https://github.com/WHUIR/PPGN/blob/master/runner/model.py#L58

Training Strategy (2) - Imbalance

Our PPGN requires data inputs in forms of (i_a, u, i_b), where i_a and i_b can be positive or negative samples, and the ratio between them is $1 : \eta(\eta > 1)$. To solve this sample imbalance problem, we apply a weighting strategy to the loss function as follow,

$$\mathcal{L}' = - \sum_{(i_a,u,i_b)\in T} \alpha \left(r_{ua} \log \hat{r}_{ua} + (1 - r_{ua}) \log (1 - \hat{r}_{ua})\right) + \beta \left(r_{ub} \log \hat{r}_{ub} + (1 - r_{ub}) \log (1 - \hat{r}_{ub})\right) + \lambda \sum |\Theta| \tag{6}$$

$$\alpha = \begin{cases} \eta, & \text{if } r_{ua} = 1; \\ 1, & \text{if } r_{ua} = 0. \end{cases} \quad \beta = \begin{cases} \eta, & \text{if } r_{ub} = 1; \\ 1, & \text{if } r_{ub} = 0. \end{cases} \tag{7}$$

where α and β are the weight values determined by the labels of input set, which speeds up the training process.

```python
        loss_list_s = tf.nn.sigmoid_cross_entropy_with_logits(…
        loss_list_t = tf.nn.sigmoid_cross_entropy_with_logits(…
        loss_w_s = tf.map_fn(
            lambda x: tf.cond(tf.equal(x, 1.0), lambda: 5.0, lambda: 1.0),
            self.label_s,
        )
        loss_w_t = tf.map_fn(
            lambda x: tf.cond(tf.equal(x, 1.0), lambda: 5.0, lambda: 1.0),
            self.label_t,
        )

        self.loss_s = tf.reduce_mean(tf.multiply(loss_list_s, loss_w_s))
        self.loss_t = tf.reduce_mean(tf.multiply(loss_list_t, loss_w_t))

        self.loss = self.loss_s + self.loss_t
```

https://github.com/WHUIR/PPGN/blob/master/runner/model.py#L139

Experiment - Dataset

- Domain A, Domain B, User
- A -> B ?
- B -> A ?
- U -> A
- U -> B

Table 1: Statistics of the two pairs of datasets

Datasets	# users	# items	# ratings	density
Books	37, 388	269, 301	1, 254, 288	0.012%
Movies and TV	37, 388	49, 273	792, 319	0.043%
CDs and Vinyl	5, 331	55, 848	376, 347	0.126%
Digital Music	5, 331	3, 563	63, 303	0.333%

Metrics	Dataset	BPRMF	NeuMF	NeuMF+	CoNet	SCoNet	PPGN-IP	PPGN
HR@10	Books	.3654	.4300	.4291	.5223	.5141	.4594	**.5770**
	Movies and TV	.4538	.5665	.5605	.6460	.6465	.5689	**.6909**
	CDs and Vinyl	.5532	.6421	.6655	.7539	.7547	<u>.7668</u>	**.7839**
	Digital Music	.4742	.5322	.5991	.7179	.7205	<u>.7492</u>	**.7874**
MRR@10	Books	.1543	.2241	.2249	.3273	.3261	.1835	**.3280**
	Movies and TV	.2034	.2775	.2742	.3651	.3829	.2498	**.3869**
	CDs and Vinyl	.2742	.3092	.3593	.4735	.4875	.4192	**.5012**
	Digital Music	.1431	.1549	.2472	.3855	.3878	<u>.4112</u>	**.4388**
NDCG@10	Books	.2365	.2725	.2724	.3396	.3370	.2470	**.3574**
	Movies and TV	.2654	.3445	.3416	.4060	.4210	.3164	**.4249**
	CDs and Vinyl	.3532	.3933	.4303	.5227	.5291	.5020	**.5697**
	Digital Music	.2045	.2432	.3297	.4436	.4603	<u>.4911</u>	**.5147**

What's Next?

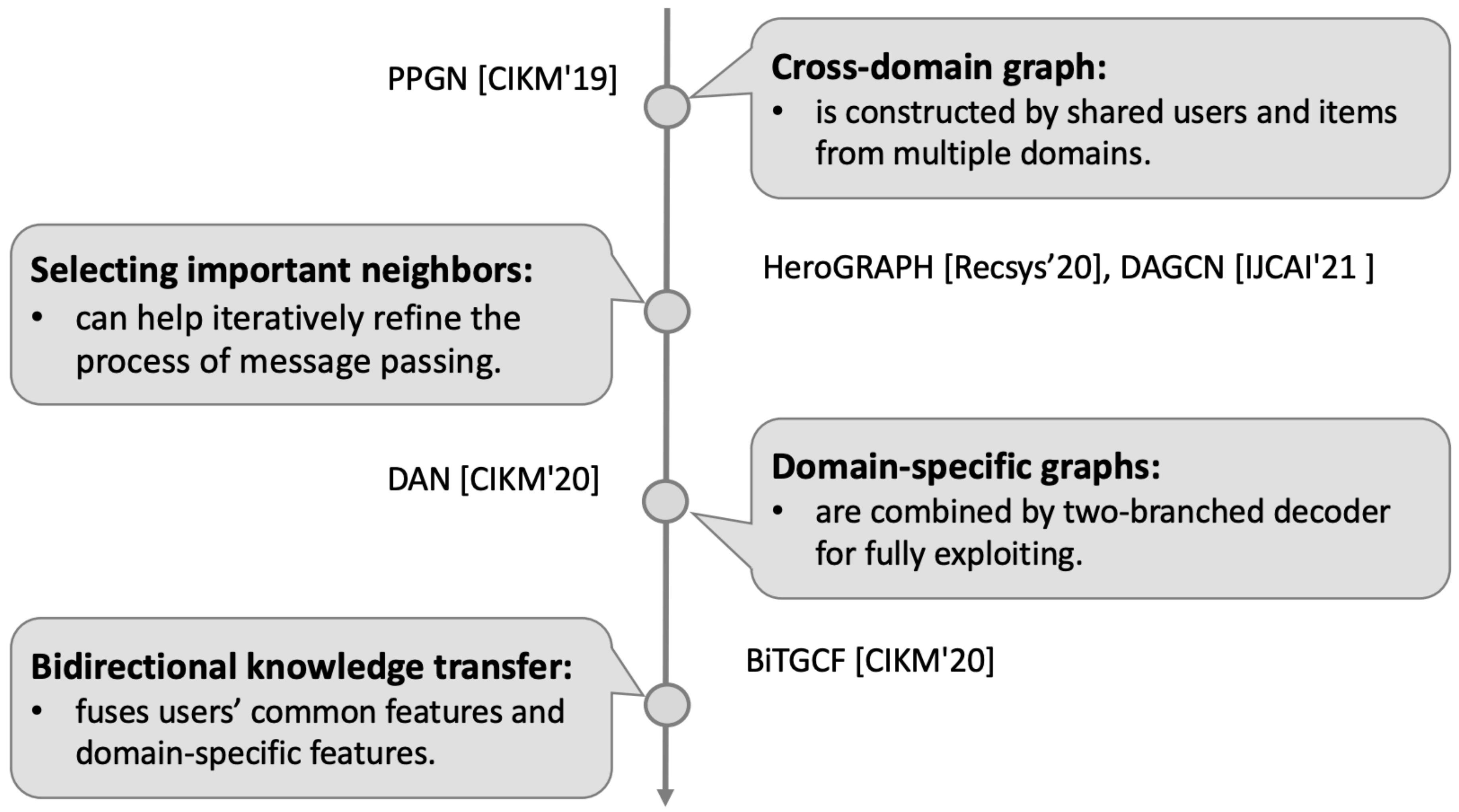


Fig. 16. Illustration of GNN models for cross-domain recommendation.

Table 11. Details of GNN models of cross-domain recommendation.

Model	Graph	Information Transferring
PPGN[204]	cross-domain graph	cross-domain propagation
BiTGCF[92]	domain-specific graphs	common user attributes
DAN[146]	domain-specific graphs	two-branched decoder
HeroGRAPH[30]	cross-domain graph	cross-domain propagation
DAGCN[49]	cross-domain graph	cross-domain propagation

Cross-Domain Recommendation: Challenges, Progress, and Prospects

Feng Zhu[1], **Yan Wang**[2], **Chaochao Chen**[1*], **Jun Zhou**[1], **Longfei Li**[1], **Guanfeng Liu**[2]
[1] Ant Group, Hangzhou 310012, China
[2] Department of Computing, Macquarie University, Sydney, NSW 2109, Australia

link

recommendation_dataset_for_pre-training & transfer learning & lifelong learning & cross-domain recommendation & cold-start recommendation

DataSets links for recommender systems research, in particular for transfer learning, user representation, pre-training,lifelong learning, cold start recommendation

https://drive.google.com/file/d/1imhHUsivh6oMEtEW-RwVc4OsDqn-xOaP/view?usp=sharin

A large-scale recommmendation datasets used in

https://github.com/fajieyuan/recommendation_dataset_pretraining