Machine Learning with Graphs (MLG)

RecSys with Graph Neural Networks

Utilize GNN to **enable realistic settings**

Part 2

Cheng-Te Li (李政德)

Institute of Data Science

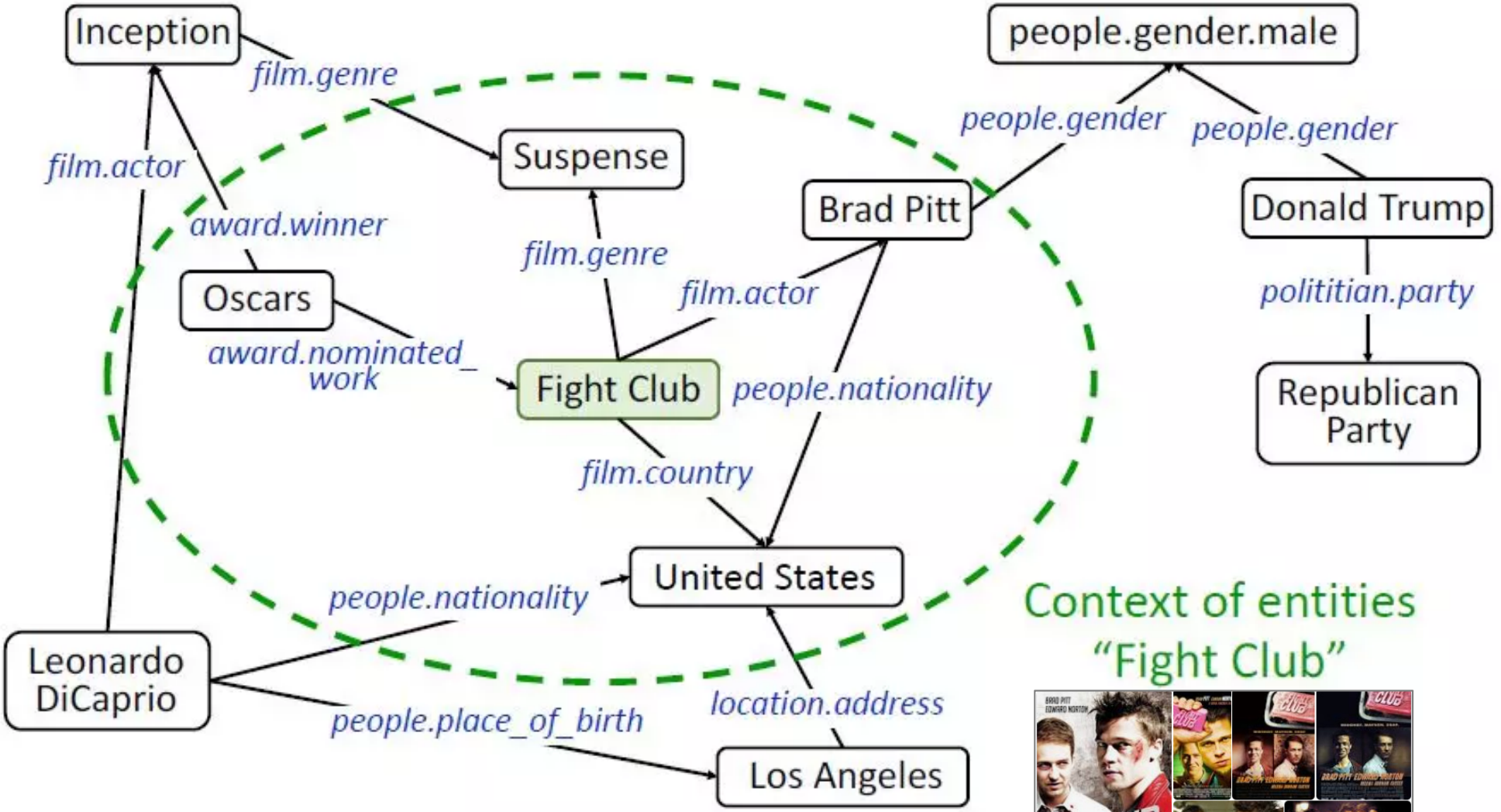
National Cheng Kung University

chengte@mail.ncku.edu.tw

References

- H. Wang et al. "**Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation**" @ WWW 2019. **220 cites**
- J. Wu et al. "**Self-supervised Graph Learning for Recommendation**" @ ACM SIGIR 2021. **82 cites**
- M. Zhang and Y. Chen. "**Inductive Matrix Completion Based on Graph Neural Networks**" @ ICLR 2020. **95 cites**

Knowledge Graph (KG)



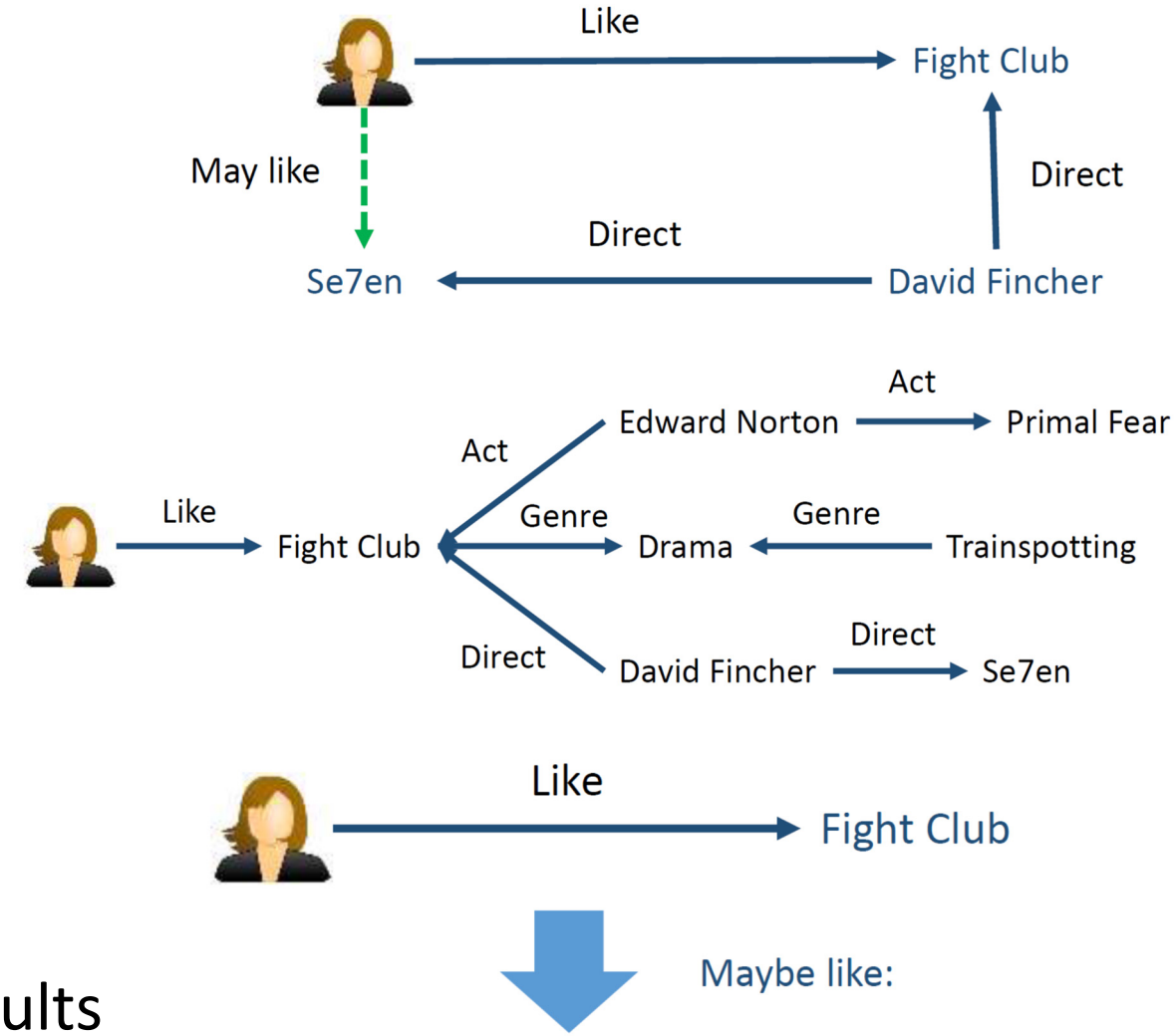
Context of entities "Fight Club"

鬥陣俱樂部 (Fight Club)

1999 年 · 劇情/黑色幽默 · 2 小時 31 分鐘

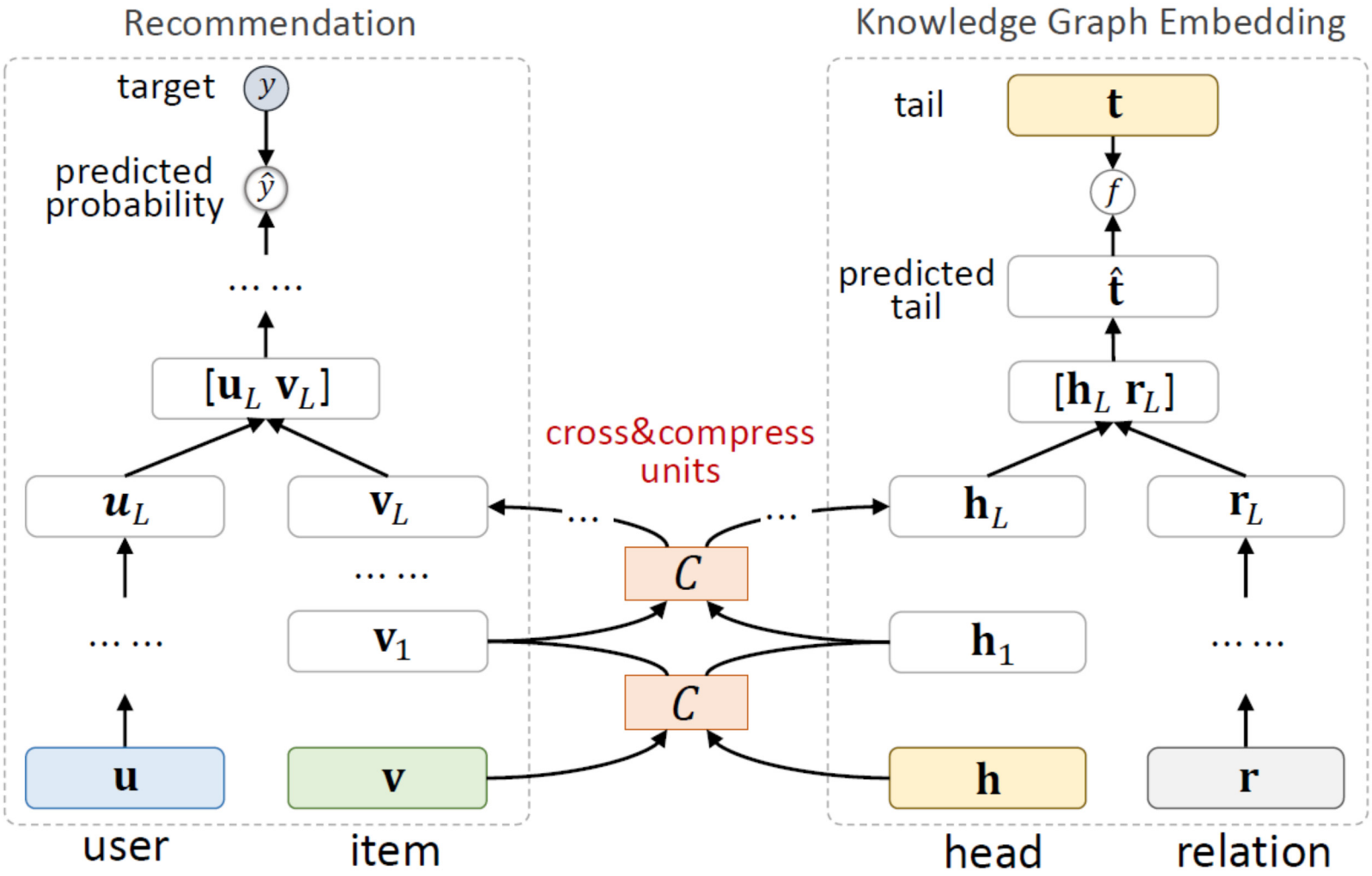
Knowledge Graph-Enhanced Recommendation

- Precision
  - More semantic content about items
  - Deep user interest
- Diversity
  - Different relations in KG
  - Extend user's interest in different paths
- Explainability
  - Connect user interest and recommendation results
  - Improve user satisfaction, boost user trust



Primal Fear, because they share the same actor
Trainspotting, because they share the same genre
Se7en, because they share the same director

RS + KGE via Multi-Task Learning



$$[\mathbf{v}_{l+1}, \mathbf{e}_{l+1}] = C(\mathbf{v}_l, \mathbf{e}_l)$$

layer for cross&compress units is small (low-level features shared between tasks)

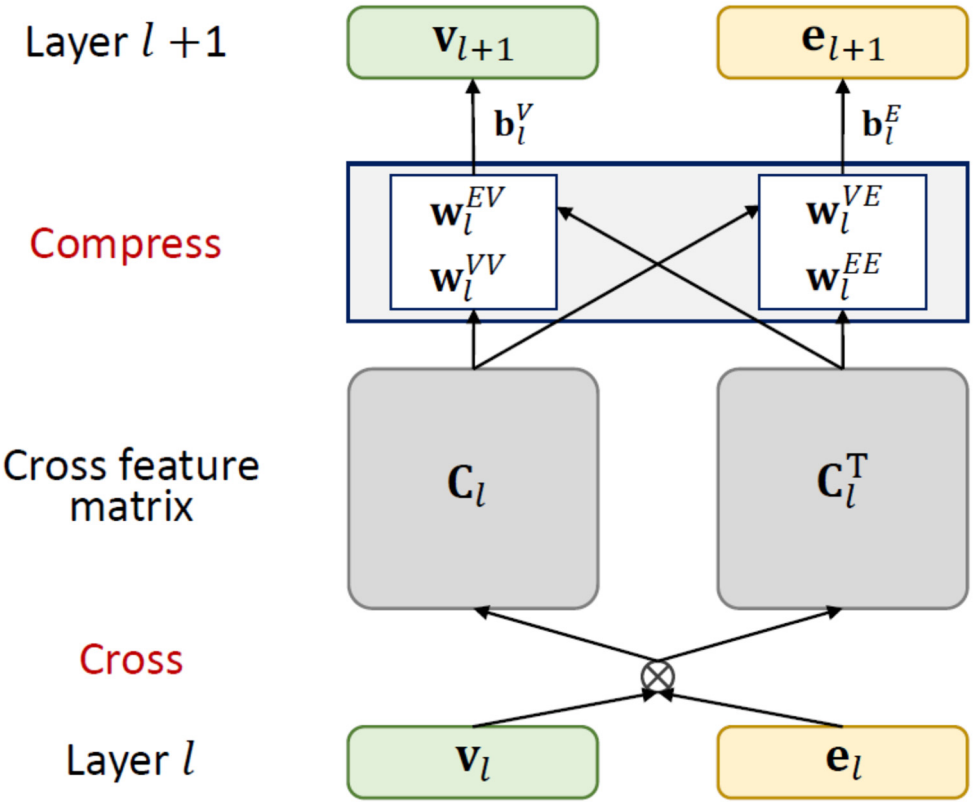
Cross & Compress Units

- Model feature interactions between items and entities
- **Cross** operation
  - Construct cross feature matrix $C_l = \mathbf{v}_l \mathbf{e}_l^{\mathrm{T}} \in \mathbb{R}^{d \times d}$

$$\mathbf{C}_l = \mathbf{v}_l \mathbf{e}_l^{\top} = \begin{bmatrix} v_l^{(1)} e_l^{(1)} & \cdots & v_l^{(1)} e_l^{(d)} \\ \cdots & & \cdots \\ v_l^{(d)} e_l^{(1)} & \cdots & v_l^{(d)} e_l^{(d)} \end{bmatrix}$$

- **Compress** operation
  - Project C_l into their latent representation spaces

$$\mathbb{R}^{d \times d} \rightarrow \mathbb{R}^{d}$$



Item v and its associated entity e

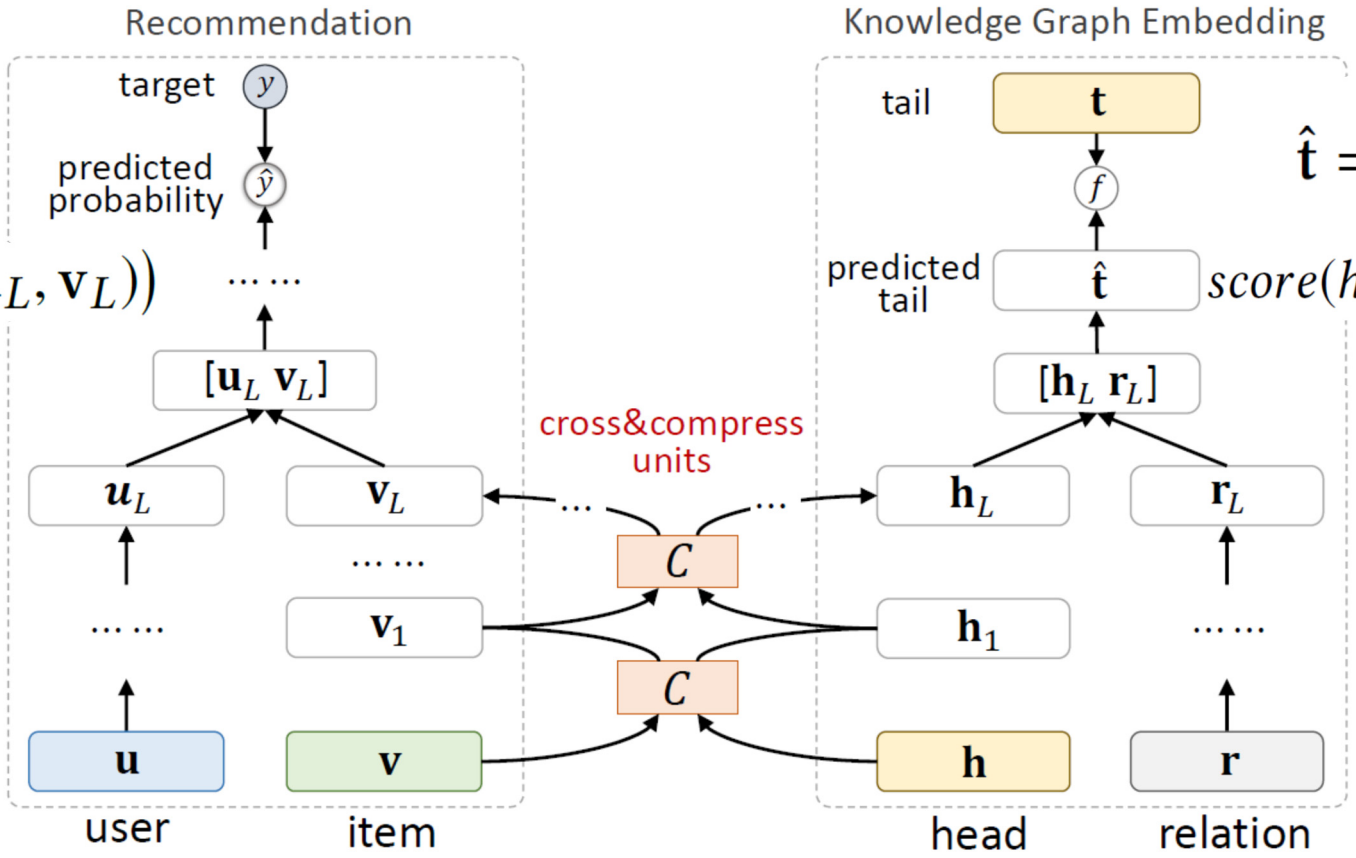
$$\mathbf{v}_{l+1} = \mathbf{C}_l \mathbf{w}_l^{VV} + \mathbf{C}_l^{\top} \mathbf{w}_l^{EV} + \mathbf{b}_l^V = \mathbf{v}_l \mathbf{e}_l^{\top} \mathbf{w}_l^{VV} + \mathbf{e}_l \mathbf{v}_l^{\top} \mathbf{w}_l^{EV} + \mathbf{b}_l^V$$

$$\mathbf{e}_{l+1} = \mathbf{C}_l \mathbf{w}_l^{VE} + \mathbf{C}_l^{\top} \mathbf{w}_l^{EE} + \mathbf{b}_l^E = \mathbf{v}_l \mathbf{e}_l^{\top} \mathbf{w}_l^{VE} + \mathbf{e}_l \mathbf{v}_l^{\top} \mathbf{w}_l^{EE} + \mathbf{b}_l^E,$$

RS and KGE

Cross Entropy Loss

BPR Loss



$$\hat{y}_{uv} = \sigma\left(f_{RS}(\mathbf{u}_L, \mathbf{v}_L)\right)$$

H-layer MLP

$$\hat{\mathbf{t}} = \mathcal{M}^K\left(\begin{bmatrix}\mathbf{h}_L \\ \mathbf{r}_L\end{bmatrix}\right)$$

$$score(h, r, t) = f_{KG}(\mathbf{t}, \hat{\mathbf{t}}) = \sigma(\mathbf{t}^\top \hat{\mathbf{t}})$$

Use L-layer MLP for $\mathbf{u}$:

$$\mathbf{u}_L = \mathcal{M}(\mathcal{M}(\cdots \mathcal{M}(\mathbf{u}))) = \mathcal{M}^L(\mathbf{u})$$

Use L-layer MLP for $\mathbf{r}$:

$$\mathbf{r}_L = \mathcal{M}^L(\mathbf{r})$$

Use C&C for $\mathbf{v}$:

$$\mathbf{v}_L = \mathbb{E}_{e \sim \mathcal{S}(v)}\left[C^L(\mathbf{v}, \mathbf{e})[\mathbf{v}]\right]$$

Use C&C for $\mathbf{h}$:

$$\mathbf{h}_L = \mathbb{E}_{v \sim \mathcal{S}(h)}\left[C^L(\mathbf{v}, \mathbf{h})[\mathbf{e}]\right]$$

References

- H. Wang et al. "**Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation**" @ WWW 2019. **220 cites**
- J. Wu et al. "**Self-supervised Graph Learning for Recommendation**" @ ACM SIGIR 2021. **82 cites**
- M. Zhang and Y. Chen. "**Inductive Matrix Completion Based on Graph Neural Networks**" @ ICLR 2020. **95 cites**

Recap: LightGCN for Collaborative Filtering

Step 1: Representation aggregation layers

$$\mathbf{Z}^{(l)} = H\left(\mathbf{Z}^{(l-1)}, \mathcal{G}\right)$$

$$\mathbf{a}_u^{(l)} = f_{\text{aggregate}}\left(\{\mathbf{z}_i^{(l-1)} | i \in \mathcal{N}_u\}\right)$$

$$\mathbf{z}_u^{(l)} = f_{\text{combine}}(\mathbf{z}_u^{(l-1)}, \mathbf{a}_u^{(l)}),$$

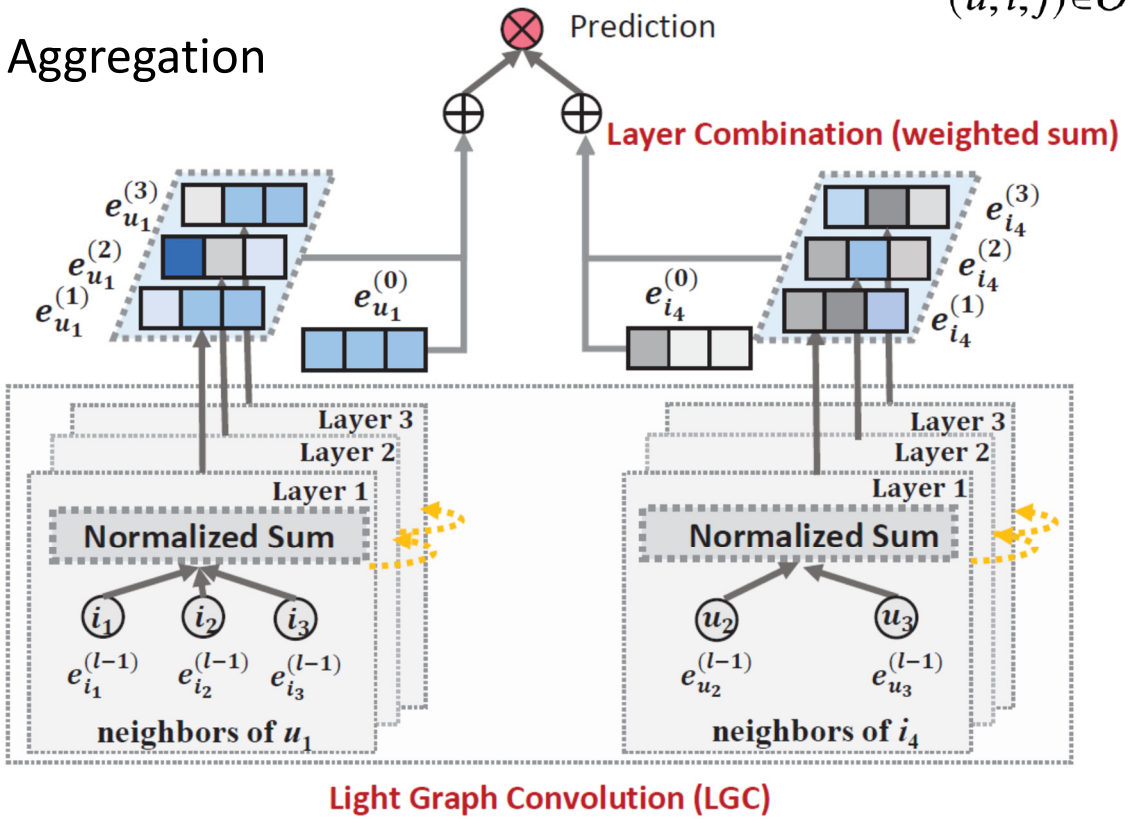
Step 2: Readout layer

$$\mathbf{z}_u = f_{\text{readout}}\left(\{\mathbf{z}_u^{(l)} | l = [0, \cdots, L]\}\right)$$

Step 3: Supervised loss

$$\mathcal{L}_{main} = \sum_{(u,i,j) \in O} -\log \sigma(\hat{y}_{ui} - \hat{y}_{uj})$$

Neighbor Aggregation



Limitations of Existing GNN RecSys

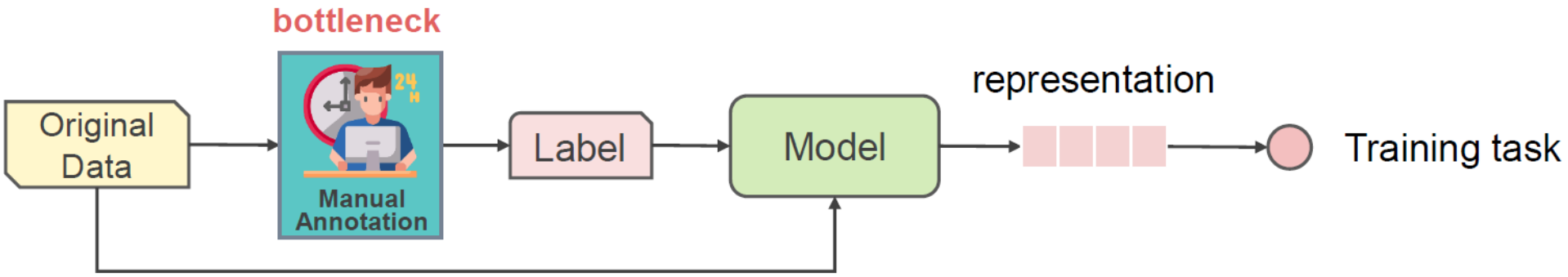
- Sparse Supervision Signal
  - The supervision signal comes from the observed interactions → extremely sparse
- Skewed Data Distribution
  - Power-law distribution
  - High-degree items exert larger impact on the representation learning
- Noises in Interactions
  - Implicit feedback makes the learning more vulnerable to interaction noises

Supervised, Unsupervised vs. Self-Supervised

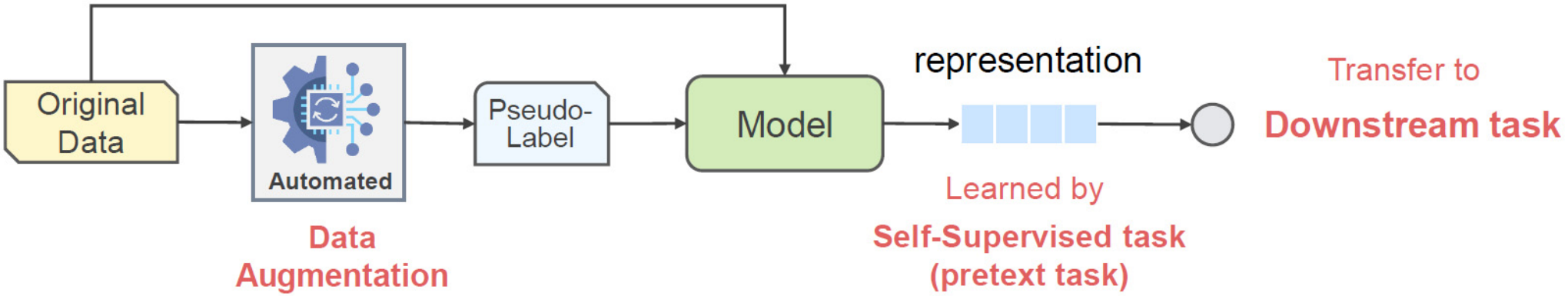
- **Supervised Learning** : learning with labeled data
  - Approach: collect a large dataset → manually label the data → train a model → evaluate and deploy
  - It is the dominant form of ML at present
  - Learned feature representations on large data can be transferred via pre-trained models to smaller domain-specific data
- **Unsupervised Learning** : learning with unlabeled data
  - Approach: discover patterns in data either via clustering similar instances, density estimation, or dimensionality reduction, etc.
- **Self-supervised Learning** : representation learning with unlabeled data
  - Learn feature representations by unlabeled data via **pretext tasks**
  - The term "self-supervised" refers to creating **its own supervision** (i.e., without supervision, without labels)

Supervised vs. Self-Supervised

Supervised learning workflow

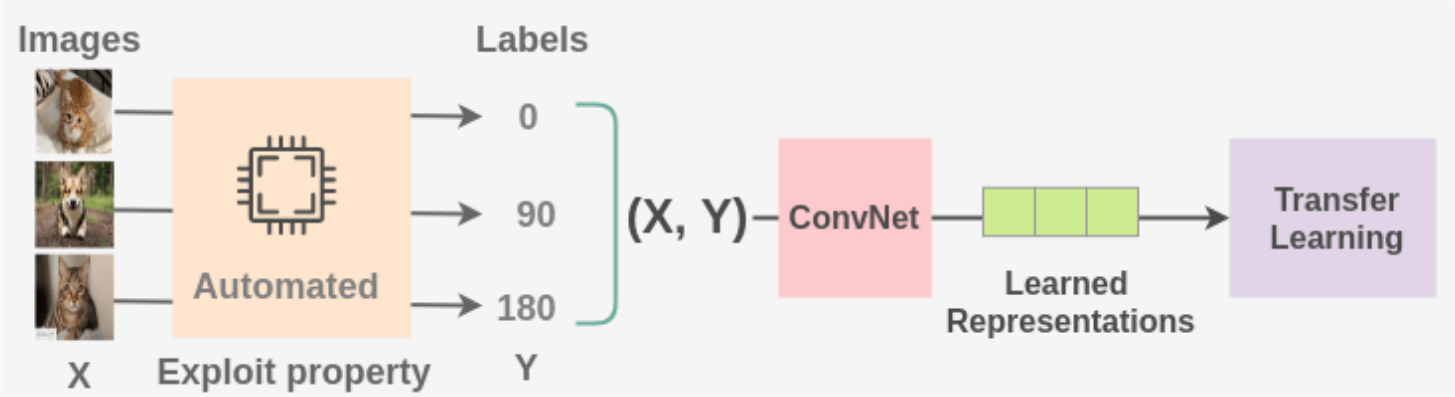


Self-Supervised learning workflow

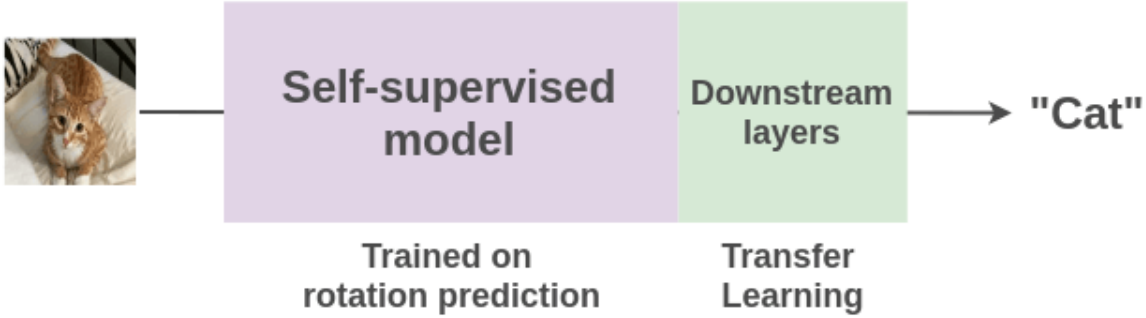


Self-Supervised Learning (SSL)

- Pretext Task: train a model to predict the rotation degree of rotated images with cats and dogs (we can collect million of images from internet, labeling is not required)



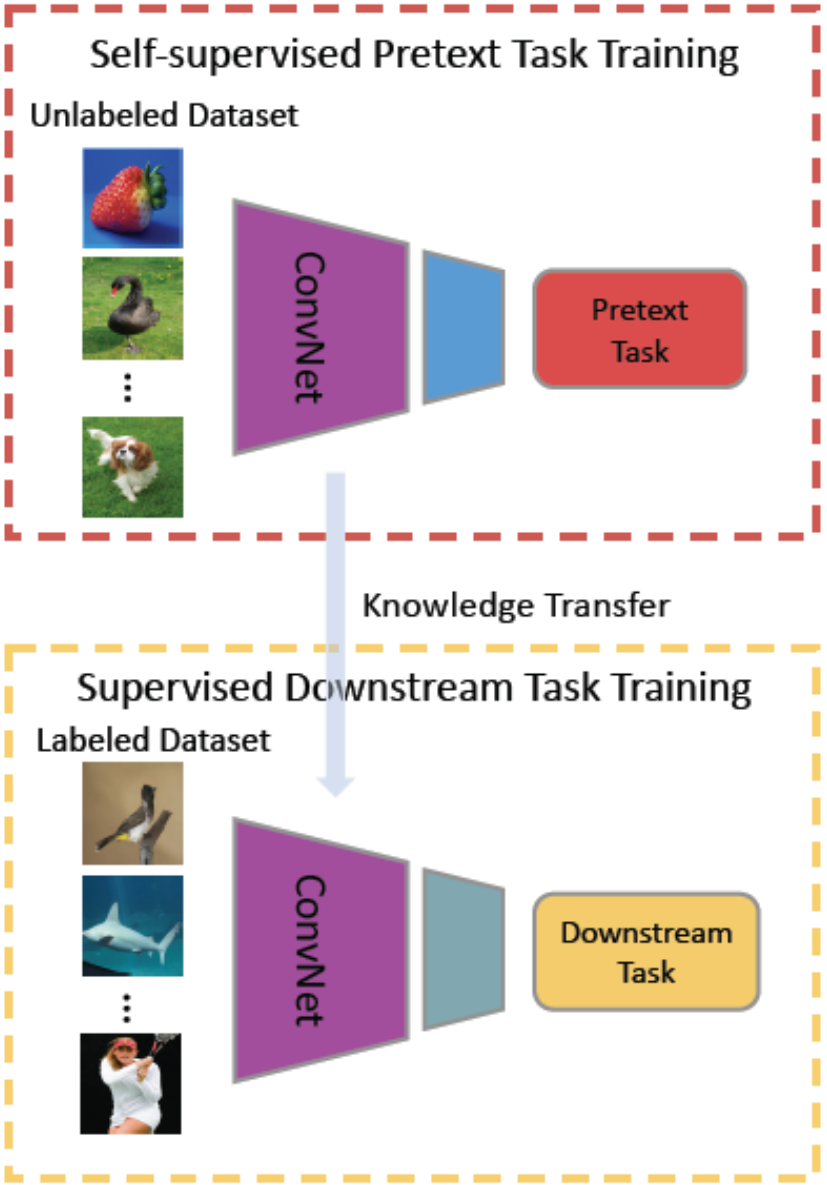
- Downstream Task: use transfer learning and fine-tune the learned model from the pretext task for classification of cats vs dogs with very few labeled examples



The General SSL Pipeline

Three Steps:

1) Identify the pretext task!
2) Pre-train the base model on a large unlabeled dataset for pretext task
3) For the downstream task, re-use the trained base model, and fine-tune the top layers (model weights) on a small labeled dataset

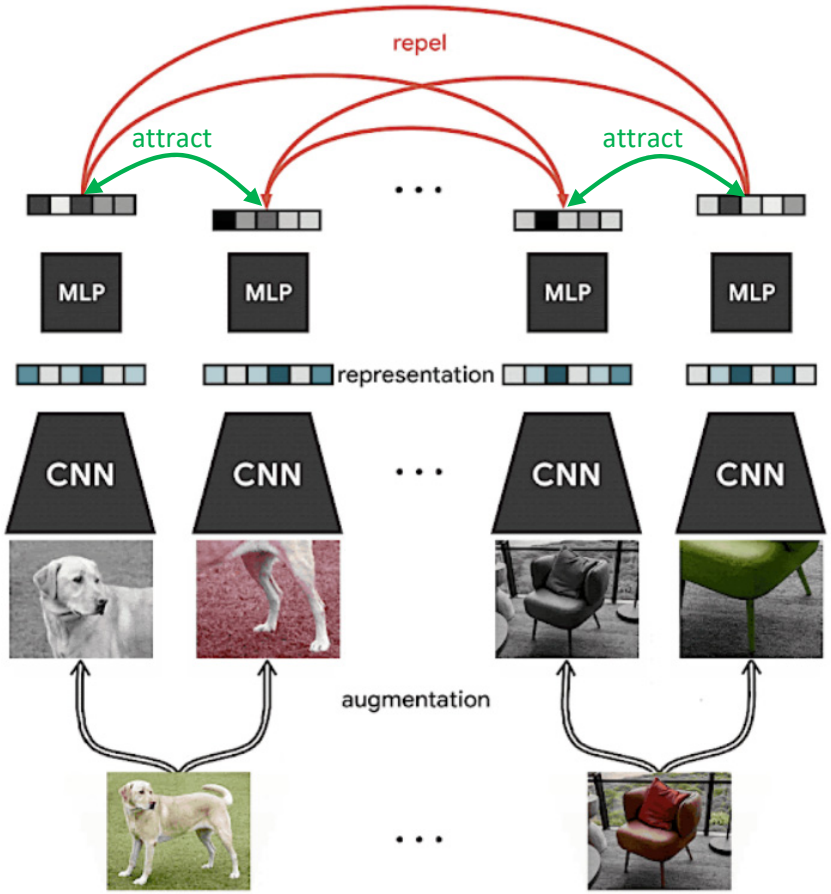


Self-Supervised Learning (SSL)

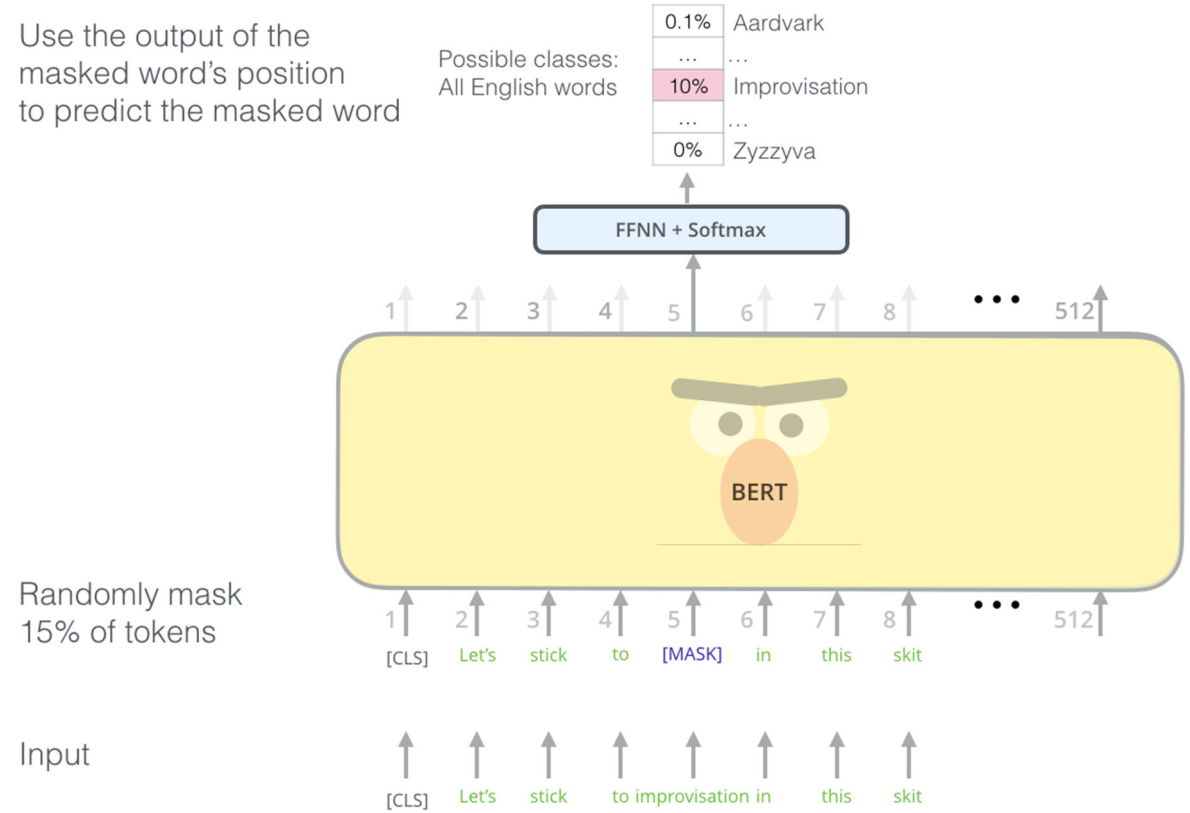
- Why SSL?
  - Create labeled datasets is expensive, time-consuming, tedious
    - Hire labelers, prepare labeling manuals, create GUIs, create storage pipelines, etc.
    - High quality annotations can be particularly expensive (e.g., medicine)
  - SSL takes advantage of the vast amount of unlabeled data on the internet (images, videos, text)
    - Rich discriminative features can be obtained by training without actual labels
  - SSL can potentially generalize better since we learn more about the world
- Challenges for SSL
  - How to identify a suitable pretext task for an application
  - There is no gold standard for comparison of learned feature representations
  - Select a suitable loss functions, since there is no single objective as the test set accuracy in supervised learning

Prominent Works in SSL

- Most prominent works are done in Computer Vision and NLP
- They take advantage of Data Augmentation



https://github.com/google-research/simclr



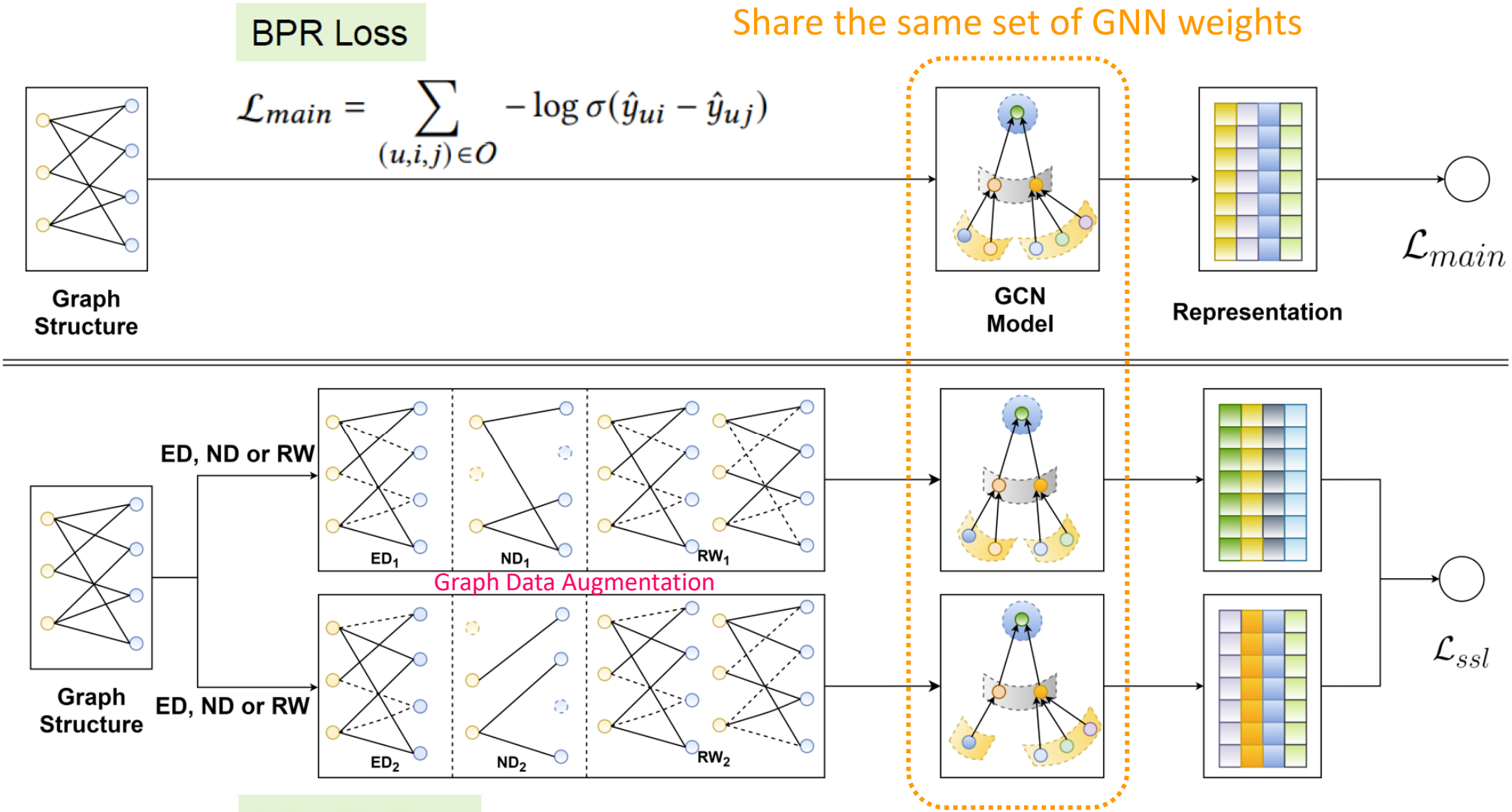
https://jalammar.github.io/illustrated-bert/

Self-Supervised Recommendation (SSR)

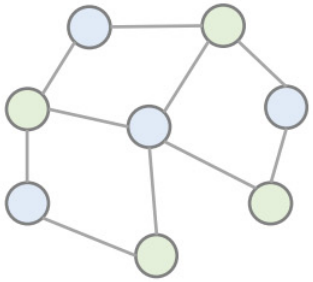
- The principle of SSL is well-aligned with RecSys' needs for more annotated data
- SSR has following essential features

1) Obtain more supervision signals by semi-automatically exploiting the raw data itself
2) Have a pretext task(s) to (pre-)train the recommendation model with the augmented data
3) Recommendation task is the unique primary task and the pretext task plays a secondary role to enhance recommendation

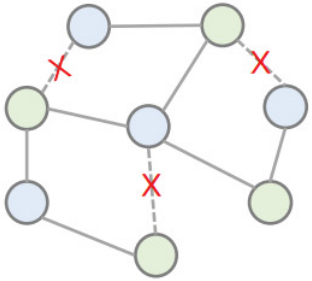
Self-supervised Graph Learning for RecSys



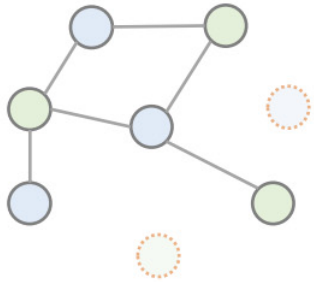
Graph Data Augmentation (1/3)



Original Graph



Edge Dropout



Node Dropout

- Edge/Node Dropout: with the probability ρ, each edge/node would be removed from the graph

Idea behind

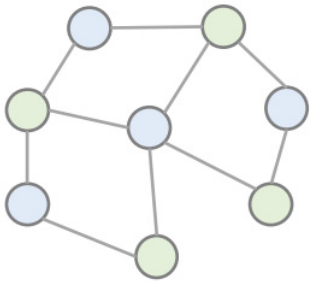
Only partial connections/nodes are informative to learn quality node representations.

Formulation

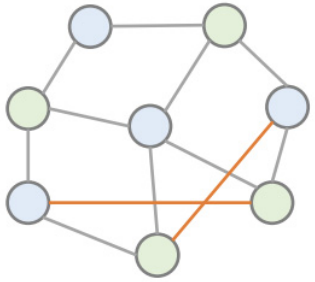
$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{E-dropout}}(\mathcal{G}) = (\mathcal{V}, \mathbf{m} \odot \mathcal{E}) \qquad \mathbf{m} \in (0,1)^{|\mathcal{E}|}$$

$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{N-dropout}}(\mathcal{G}) = (\mathcal{V} \odot \mathbf{m}, \mathcal{E} \odot \mathbf{m}') \qquad \mathbf{m} \in (0,1)^{|\mathcal{V}|}$$

Graph Data Augmentation (2/3)



Original Graph



Diffusion

- Graph Diffusion: Opposite to the dropout-based method, diffusion adds edges into the graph to create views

Idea behind

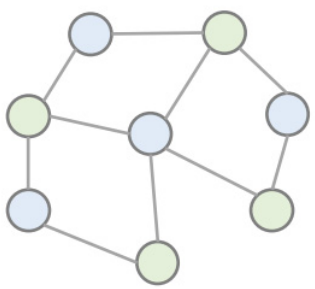
The missing user behaviors include unknown positive preferences which can be represented with weighted user-item edges.

Formulation

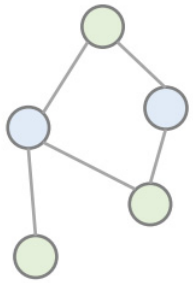
$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{diffusion}}(\mathcal{G}) = (\mathcal{V}, \mathcal{E} + \tilde{\mathcal{E}})$$

Discover the possible edges by calculating the similarities of the user and item representations and retain the edges with top-*K* similarities.

Graph Data Augmentation (3/3)



Original Graph



Subgraph Sampling

- Subgraph Sampling: samples a portion of nodes and edges by rules to form subgraphs

Idea behind

The sampled graph reflects the local connectivity and semantics.

Formulation

$$\tilde{\mathcal{G}}, \tilde{\mathbf{A}} = \mathcal{T}_{\text{sampling}}(\mathcal{G}) = (\mathcal{Z} \in \mathcal{V}, \mathbf{A}[\mathcal{Z}, \mathcal{Z}]) \qquad \mathcal{Z} \text{ the sampled node set}$$

A lot of approaches can be used to induce subgraphs like meta-path guided random walks and the ego-network sampling.

Experimental Evaluation

SGL achieves significant improvements over the state-of-the-art baselines → outstanding performance

Dataset	Yelp2018		Amazon-Book		Alibaba-iFashion	
Method	Recall	NDCG	Recall	NDCG	Recall	NDCG
NGCF	0.0579	0.0477	0.0344	0.0263	0.1043	0.0486
LightGCN	0.0639	0.0525	0.0411	0.0315	0.1078	0.0507
Mult-VAE	0.0584	0.0450	0.0407	0.0315	0.1041	0.0497
DNN+SSL	0.0483	0.0382	0.0438	0.0337	0.0712	0.0325
SGL-ED	**0.0675**	**0.0555**	**0.0478**	**0.0379**	**0.1126**	**0.0538**
%Improv.	5.63%	5.71%	9.13%	12.46%	4.45%	6.11%
p-value	5.92e-8	1.89e-8	5.07e-10	3.63e-10	3.34e-8	4.68e-10

Randomly select 80% data for training set, and 20% data for testing set

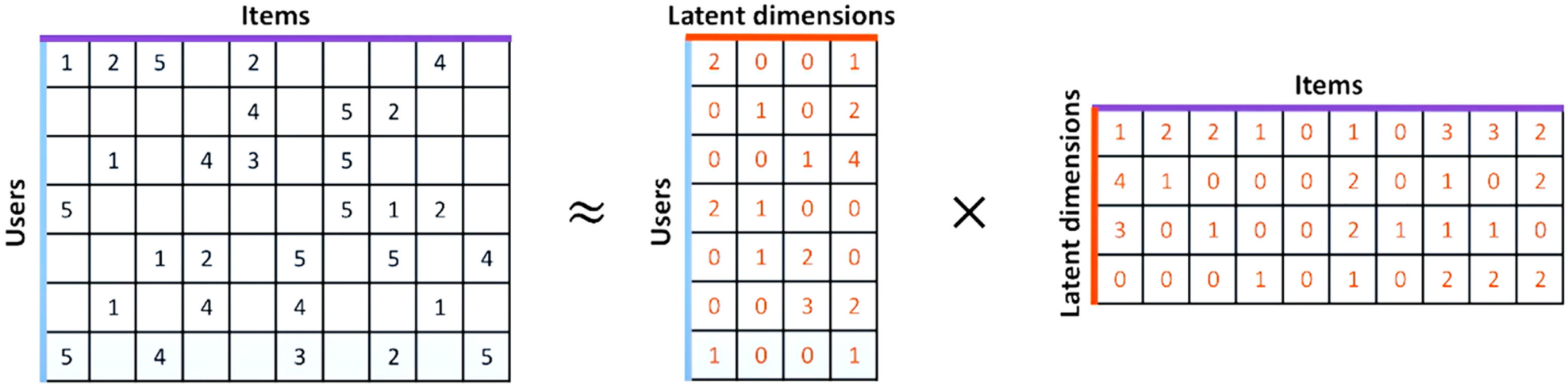
Dataset	#Users	#Items	#Interactions	Density
Yelp2018	31,668	38,048	1,561,406	0.00130
Amazon-Book	52,643	91,599	2,984,108	0.00062
Alibaba-iFashion	300,000	81,614	1,607,813	0.00007

References

- H. Wang et al. "**Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation**" @ WWW 2019. **220 cites**
- J. Wu et al. "**Self-supervised Graph Learning for Recommendation**" @ ACM SIGIR 2021. **82 cites**
- M. Zhang and Y. Chen. "**Inductive Matrix Completion Based on Graph Neural Networks**" @ ICLR 2020. **95 cites**

Typical Matrix Factorization

- MF: Factorize rating matrix into the product of user and item embeddings



Problems

- Transductive: cannot generalize to users/items unseen during training
- Re-Training: when rating matrix changes values or adds new columns/rows, requires a time-consuming re-training

Inductive Matrix Competition (IMC)

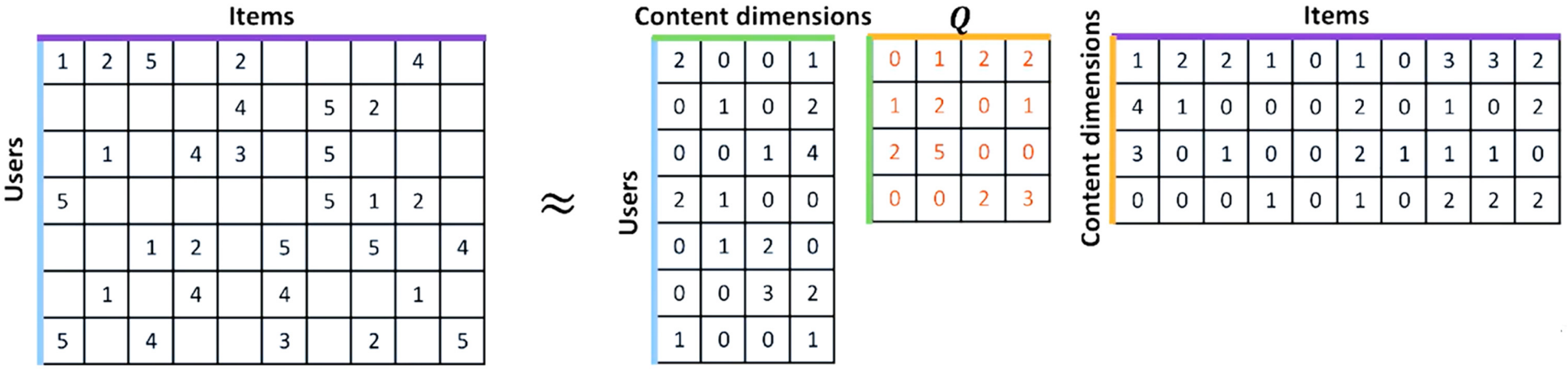
- Complete empty entries by the product of user and item content features

Content features: (action, romantic, horror, disaster)

User i: (1, 1, 0, 1) = x_i

Movie j: (0, 1, 0, 1) = y_j

$$r_{i,j} = x_i^T \boldsymbol{Q} y_j$$

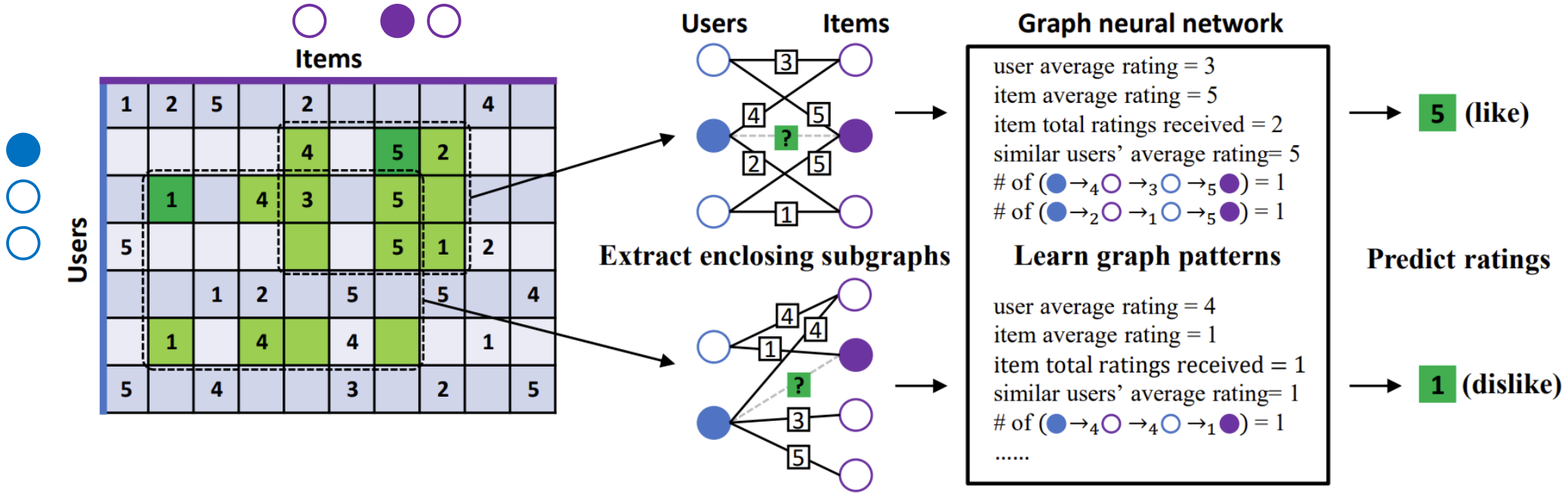


Problems

- High-quality content often not available
- Worse performance than MF

Inductive Graph Matrix Completion (IGMC)

- Infer ratings from local subgraphs, instead of content



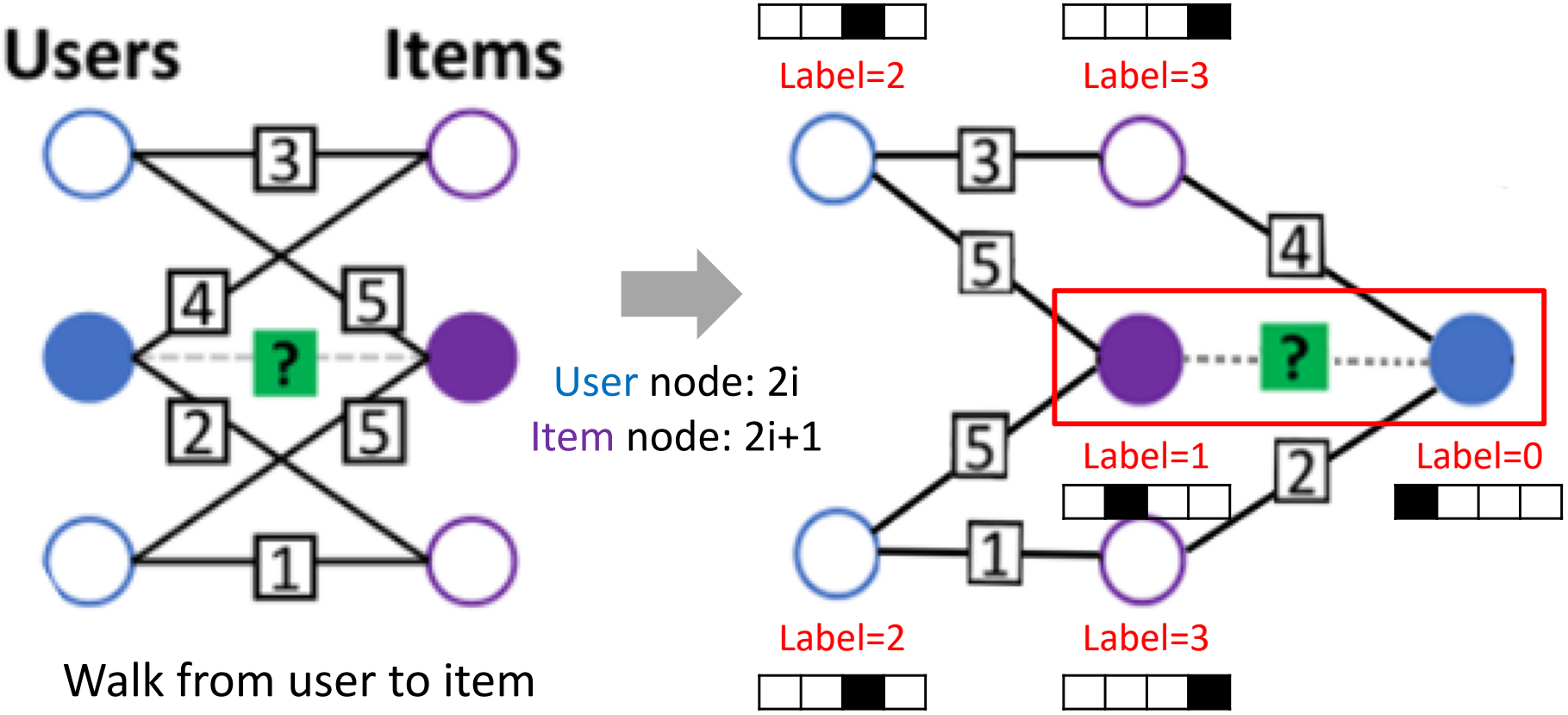
- Extract an enclosing (bipartite) subgraph for each observed user-item rating
- Feed enclosing subgraphs into a GNN to predict ratings
- Allow inductive learning: for new-coming users/items

Vector Initialization for Inductive Learning

IGMC Initialization for node vectors:

1) Distinguish the target user and target item between which the target rating is located
2) Differentiate user-type nodes from item-type nodes
3) Determined inside each encoding subgraph (independent of global bipartite graph)

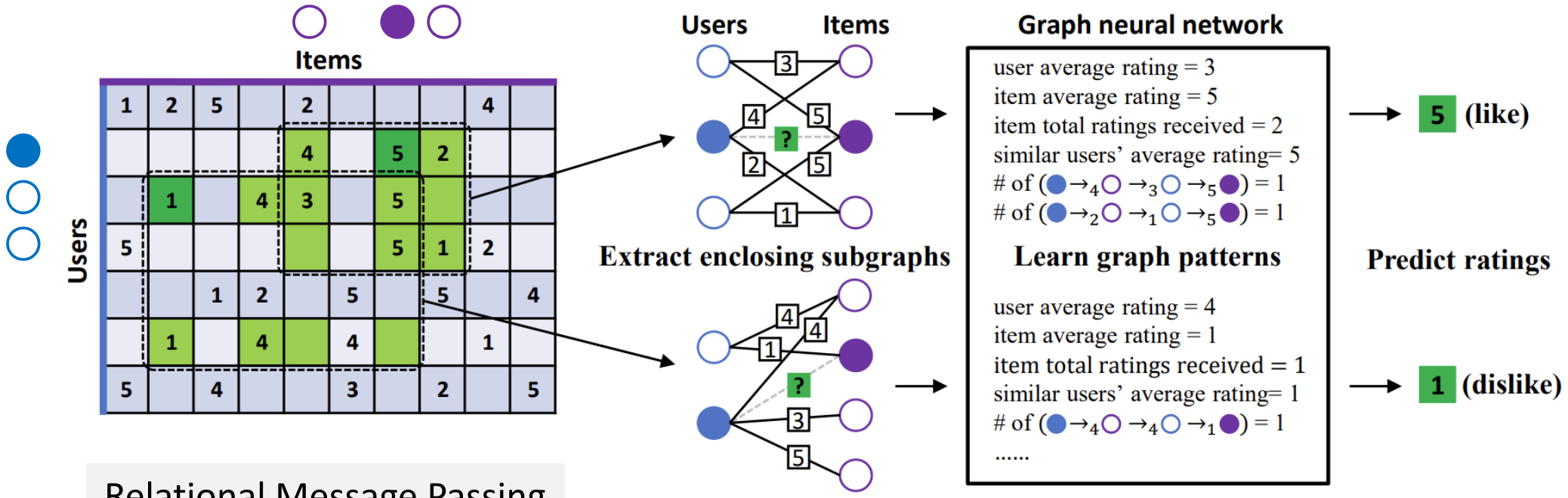
→ enable the inductive capability



Walk from user to item

i.e., U → I → U → I → …

Inductive Graph Matrix Completion



Relational Message Passing

$$\mathbf{x}_i^{l+1} = \mathbf{W}_0^l \mathbf{x}_i^l + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_r(i)} \frac{1}{|\mathcal{N}_r(i)|} \mathbf{W}_r^l \mathbf{x}_j^l \qquad \mathcal{R} = \{1, 2, 3, 4, 5\}$$

$\mathbf{h}_i = \text{concat}(\mathbf{x}_i^1, \mathbf{x}_i^2, \ldots, \mathbf{x}_i^L)$ # Embedding concat over L layers

$\mathbf{g} = \text{concat}(\mathbf{h}_u, \mathbf{h}_v)$ # Pooling from node embedings to subgraph embedding

$\hat{r} = \mathbf{w}^\top \sigma(\mathbf{W}\mathbf{g})$ # Prediction by MLP

Why Learning from Local Subgraphs?

- Compared to IMC
  - Leverage subgraph patterns only, **need no content features**
  - Also has much better performance
- Compared to MF, IGMC is inductive
  - Learn GNN parameters instead of user/item embeddings
  - Can be applied to enclosing subgraphs of **users/items unseen** during the training – inductive learning
  - If the rating **matrix changes**, IGMC does not need re-training – just apply learned GNN weights
  - If can even **transfer to new tasks** – transfer learning

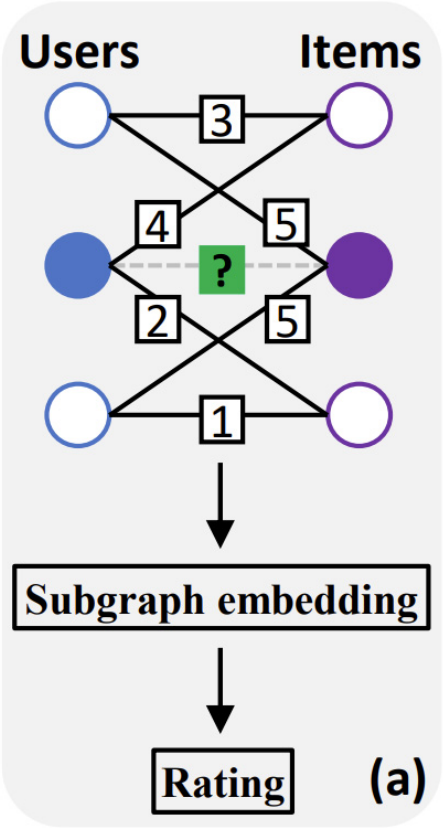
Graph-level GNN vs. Node-level GNN

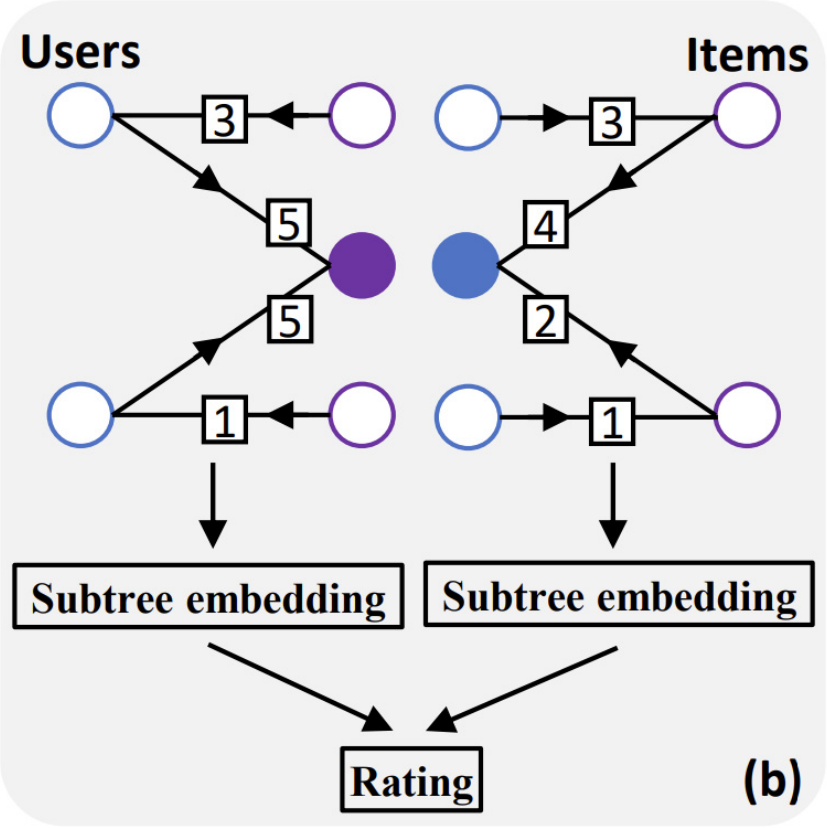
e.g., IGMC

e.g., LightGCN

Learning user-item interaction patterns from the enclosing subgraph

user average rating = 3
item average rating = 5
item total ratings received = 2
similar users' average rating= 5
of (●→4○ →3○ →5●) = 1
of (●→2○ →1○ →5●) = 1





The learned node embeddings are essentially encoding two rooted subtrees around the two nodes independently, which fails to model the **interactions** and **correspondences** between the nodes of the two trees

Graph-level GNN

Mapping the enclosing subgraph around the target user and item to their rating

Node-level GNN

Learn target user's and item's embeddings and use the node embeddings to predict the rating

these two cases (a) and (b) look **identical** to a node-based GNN approach

References

- H. Wang et al. "**Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation**" @ WWW 2019. **220 cites**
- J. Wu et al. "**Self-supervised Graph Learning for Recommendation**" @ ACM SIGIR 2021. **82 cites**
- M. Zhang and Y. Chen. "**Inductive Matrix Completion Based on Graph Neural Networks**" @ ICLR 2020. **95 cites**