Knowledge Graph Convolutional Networks for Recommender System

(Wang, Zhao, et.al. 2018)

Zihao Li, 2020.02.09

Motivation

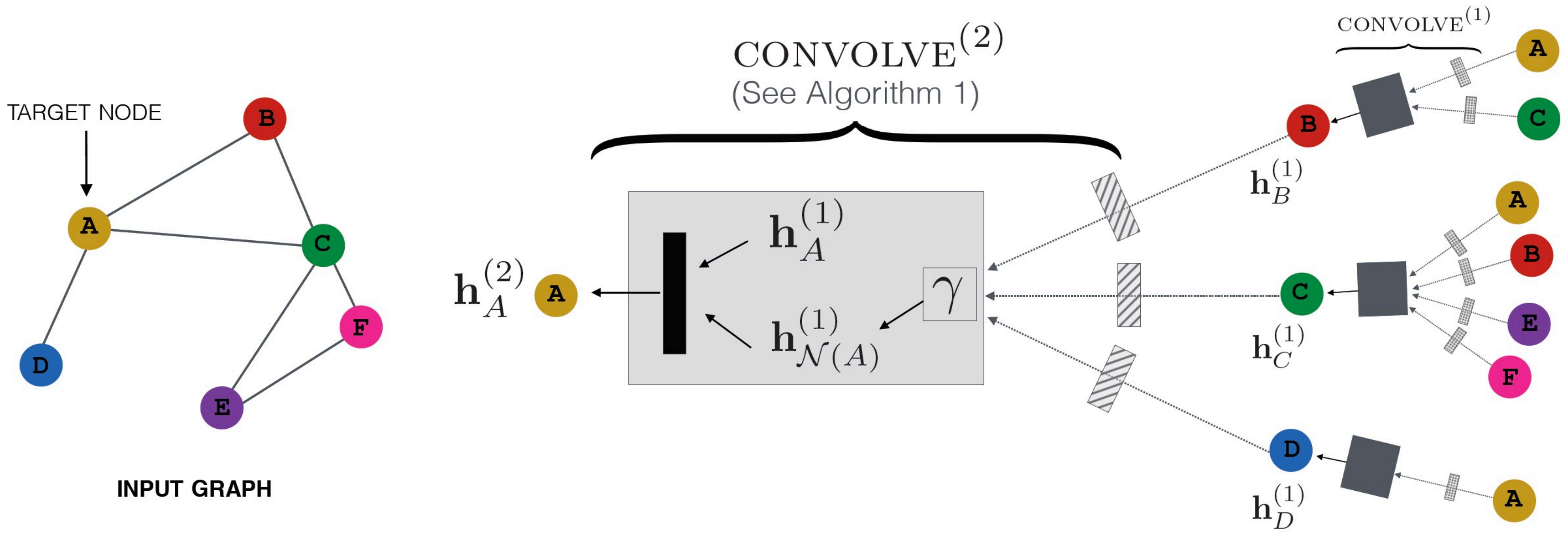
- **Why should we apply Knowledge Graph(KG) to Recommendation System(RS)?**

a) **Alleviating sparsity and cold start problem**;
b) Improving the accuracy, by **combining more external information**;
c) **Exploiting the internal point of interesting(POI)** of users;
d) KG connects a user's historically-liked and recommended items, thereby bringing **explainability** to recommender systems;

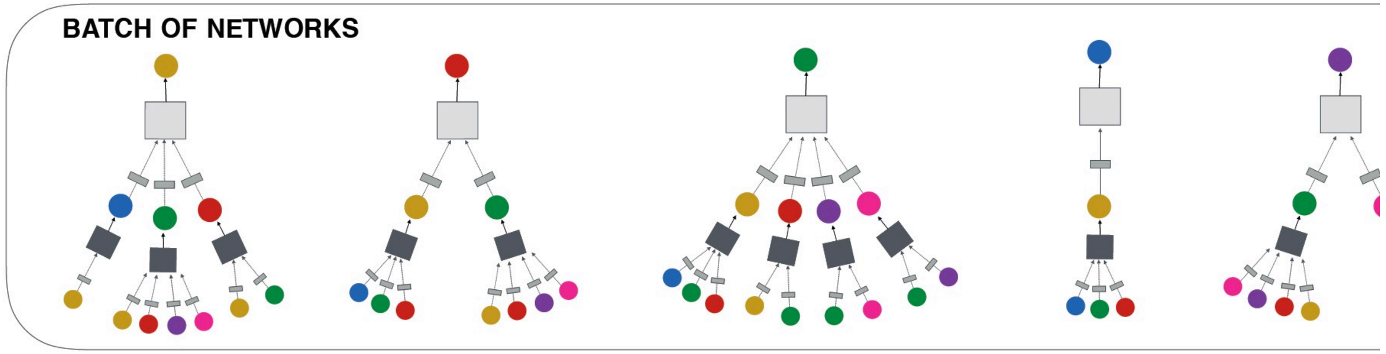
- **Why KG?**

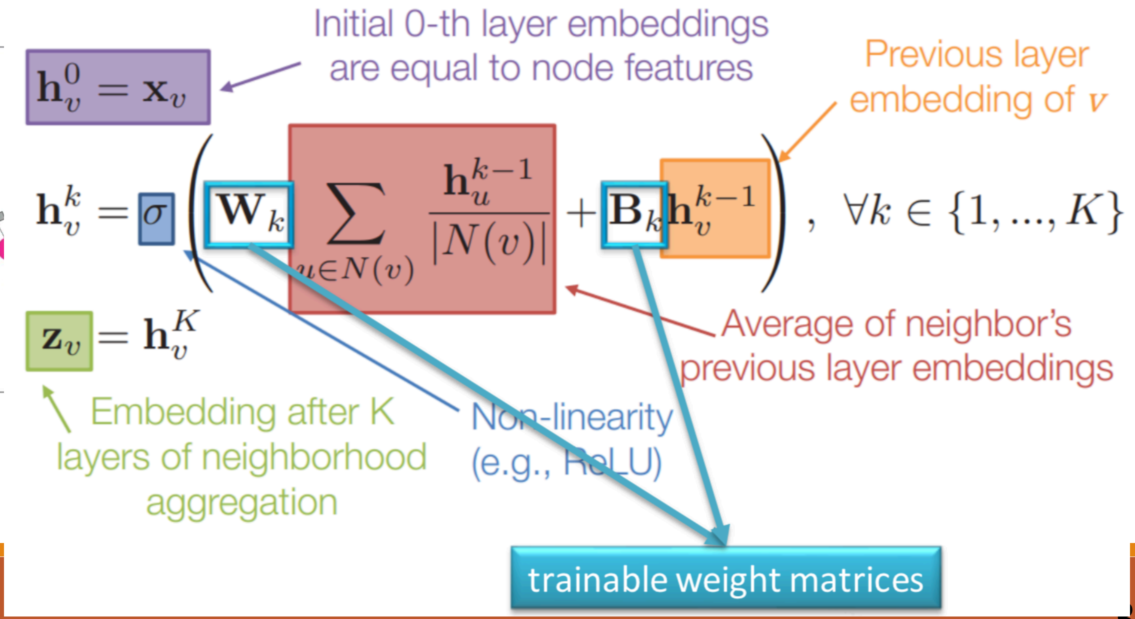
a) Benefiting from the **Resource Description Framework (RDF),** Knowledge Graph is allowed to **organize and structure the features and attributes** effectively;
b) User-item interact matrix may have some overlap with Knowledge Graph entity, which creates a precondition of combination. For instance, (J.D.Salinger, author, The catcher in the rye), Peter read The catcher in the rye;

Motivation



Graph Neural Networks





$$\boxed{\mathbf{h}_v^0 = \mathbf{x}_v}$$

$$\mathbf{h}_v^k = \boxed{\sigma}\left(\boxed{\mathbf{W}_k}\boxed{\sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}} + \boxed{\mathbf{B}_k}\boxed{\mathbf{h}_v^{k-1}}\right), \quad \forall k \in \{1, \dots, K\}$$

$$\boxed{\mathbf{z}_v} = \mathbf{h}_v^K$$

Initial 0-th layer embeddings are equal to node features
Previous layer embedding of v
Average of neighbor's previous layer embeddings
Non-linearity (e.g., ReLU)
Embedding after K layers of neighborhood aggregation
trainable weight matrices

Motivation & Contribution

- **Why utilizing graph convolutional networks(GCN)?**

a) Key idea: **aggregate and incorporate neighborhood information** with bias when calculating the representation of a given entity in the Knowledge graph;
b) Through the operation of neighborhood aggregation, the local proximity structure is successfully **captured and stored in each entity**;
c) Neighbors are **weighted by scores**, which characterizes both the semantic information of Knowledge Graph and users' personalized interests in relations;

- **Contribution or highlight work**

It was the first attempt to apply the knowledge graph convolutional networks to resolve recommendation problem.

Problem Formulation

- Note users as $\mathcal{U} = \{\mu_1, \mu_2, \dots \mu_M\}$; items as $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$; user-item interaction matrix $Y \in \mathbb{R}^{M \times N}$, where $y_{uv} = 1$, if user u engages with item v, such as clicking, browsing, or purchasing; otherwise $y_{uv} = 0$;

- Define KG $\mathcal{G}$ is comprised of entity-relation-entity triples (h, r, t);

- Hence, we aim to predict whether user u has potential interest in item v with which he has had no interaction before;

- Learning the object function: $\hat{y}_{uv} = \mathcal{F}(u, v | \Theta, Y, \mathcal{G})$, where $\hat{y}_{uv}$ denotes the probability that user u will engage with item v, and Θ denotes the model parameters of function $\mathcal{F}$;

KGCN Layer

- **User-relation:** Exploring the latten POI of users, that is to say **constructing the relations between users and item features or attributes** (represent by r)

$\pi_r^u = g(u, r)$, where $g(\cdot)$ can be inner product;

- **Layer embedding (convolution):** based on neighbor notes of upper layer to represent the entity of next layer

$$v_{N(v)}^u = \sum_{e \in N(v)} \tilde{\pi}_{r_{v,e}}^u e\,, \qquad \tilde{\pi}_{r_{v,e}}^u = \frac{\exp(\pi_{r_{v,e}}^u)}{\sum_{e \in N(v)} \exp(\pi_{r_{v,e}}^u)}$$

Where, e is the representation of entity, and $\tilde{\pi}_{r_{v,e}}^u$ can be regarded as weight of each entity, which can also be recognized as the normalized user-relation score, and reflecting the preference of users'.

- **Final layer:** aggregate the entity representation and its neighborhood representation into a single vector

KGCN Layer

(1) Sum: $agg_{sum} = \sigma\left(W \cdot \left(v + v_{S(v)}^{u}\right) + b\right);$

(2) Concat: $agg_{concat} = \sigma\left(W \cdot concat\left(v, v_{S(v)}^{u}\right) + b\right);$

(3) Neighbor: $agg_{sum} = \sigma\left(W \cdot v_{S(v)}^{u} + b\right);$

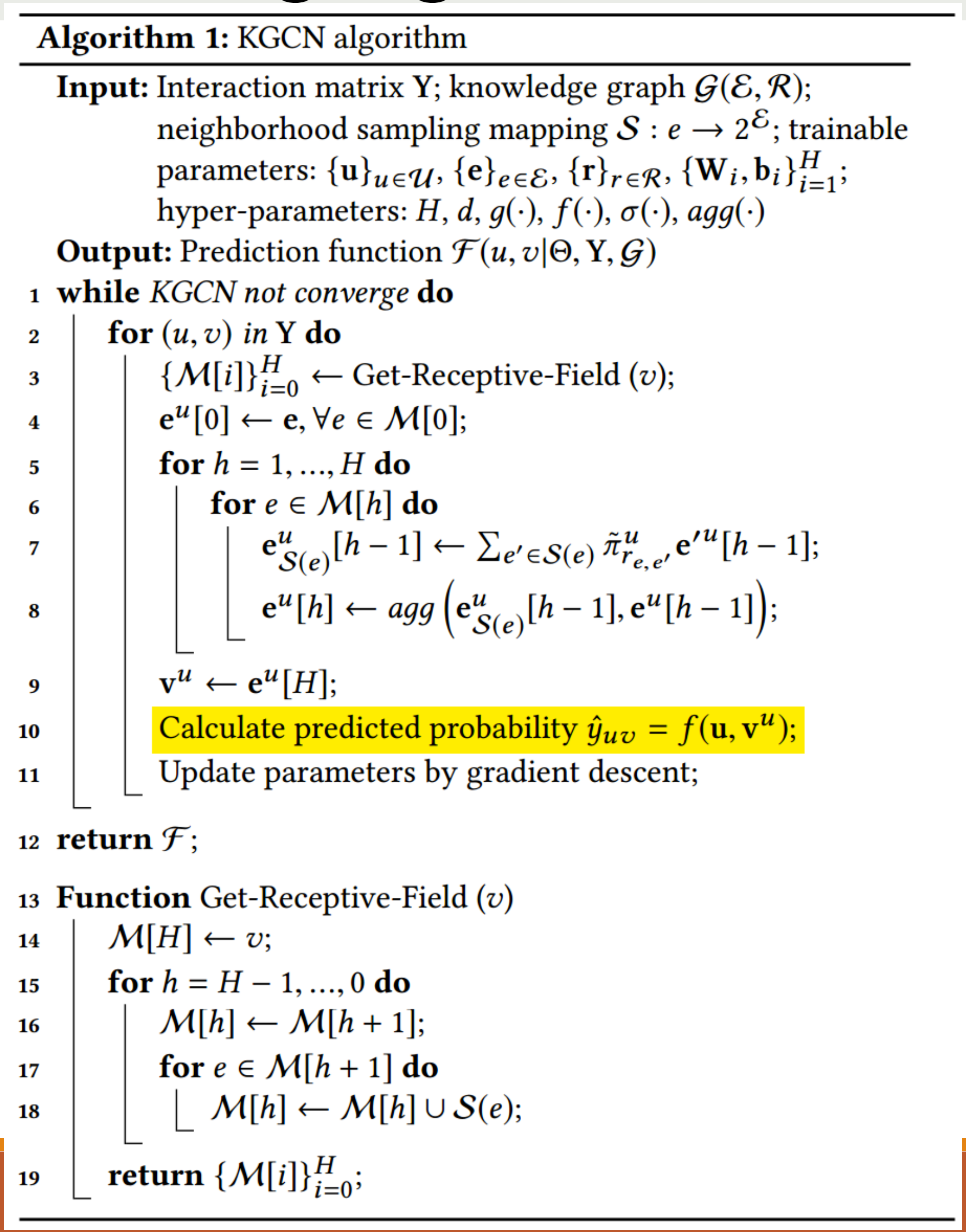
Where, $v_{S(v)}^{u}$ is the neighborhood representation, consider the numbers of neighbors of each entity may be different, here the author apply **negative sampling** method to select the fixed number neighbors.

- **Prediction:** $\hat{y}_{uv} = f(u, v^{u});$

- **Loss function:** $\mathcal{L} = \sum_{u \in \mathcal{U}} (\sum_{v: y_{uv}=1} \mathcal{J}(y_{uv}, \hat{y}_{uv}) - \sum_{i=1}^{T^u} \mathbb{E}_{v_i \sim P(v_i)} \mathcal{J}\left(y_{uv_i}, \hat{y}_{uv_i}\right)) + \lambda \|F\|_2^2$

Where, J means cross-entropy loss, P is a negative sampling distribution and $T^{u} = |\{v: y_{uv} = 1\}|$ and P follows a uniform distribution.

Learning Algorithm



Algorithm 1: KGCN algorithm

```
Input: Interaction matrix Y; knowledge graph $\mathcal{G}(\mathcal{E},\mathcal{R})$;
       neighborhood sampling mapping $\mathcal{S}: e \to 2^{\mathcal{E}}$; trainable
       parameters: $\{\mathbf{u}\}_{u\in\mathcal{U}}$, $\{\mathbf{e}\}_{e\in\mathcal{E}}$, $\{\mathbf{r}\}_{r\in\mathcal{R}}$, $\{\mathbf{W}_i,\mathbf{b}_i\}_{i=1}^{H}$;
       hyper-parameters: $H$, $d$, $g(\cdot)$, $f(\cdot)$, $\sigma(\cdot)$, $agg(\cdot)$
Output: Prediction function $\mathcal{F}(u,v|\Theta,\mathrm{Y},\mathcal{G})$
while KGCN not converge do
    for $(u,v)$ in Y do
        $\{\mathcal{M}[i]\}_{i=0}^{H} \leftarrow$ Get-Receptive-Field $(v)$;
        $\mathbf{e}^u[0] \leftarrow \mathbf{e}, \forall e \in \mathcal{M}[0]$;
        for $h = 1,\dots,H$ do
            for $e \in \mathcal{M}[h]$ do
                $\mathbf{e}^u_{\mathcal{S}(e)}[h-1] \leftarrow \sum_{e'\in\mathcal{S}(e)} \tilde{\pi}^u_{r_{e,e'}} \mathbf{e'}^u[h-1]$;
                $\mathbf{e}^u[h] \leftarrow agg\left(\mathbf{e}^u_{\mathcal{S}(e)}[h-1], \mathbf{e}^u[h-1]\right)$;
        $\mathbf{v}^u \leftarrow \mathbf{e}^u[H]$;
        Calculate predicted probability $\hat{y}_{uv} = f(\mathbf{u},\mathbf{v}^u)$;
        Update parameters by gradient descent;
return $\mathcal{F}$;
Function Get-Receptive-Field $(v)$
    $\mathcal{M}[H] \leftarrow v$;
    for $h = H-1,\dots,0$ do
        $\mathcal{M}[h] \leftarrow \mathcal{M}[h+1]$;
        for $e \in \mathcal{M}[h+1]$ do
            $\mathcal{M}[h] \leftarrow \mathcal{M}[h] \cup \mathcal{S}(e)$;
    return $\{\mathcal{M}[i]\}_{i=0}^{H}$;
```

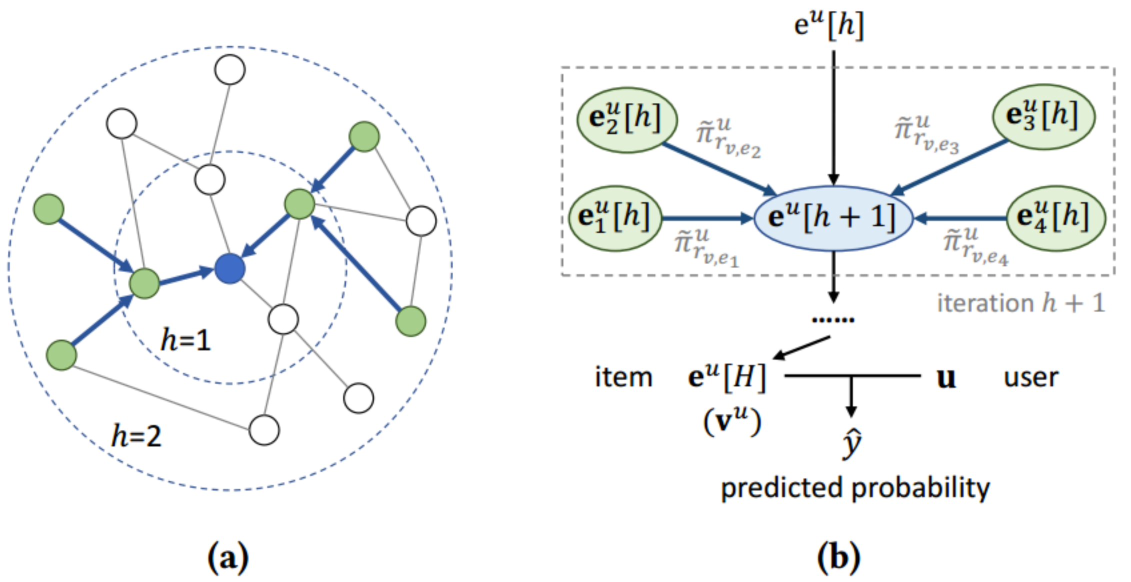


Figure 1: (a) A two-layer receptive field (green entities) of the blue entity in a KG. (b) The framework of KGCN.

Datasets

- **MovieLens-20M** is a widely used benchmark dataset in movie recommendations, which consists of approximately 20 million explicit ratings (ranging from 1 to 5) on the MovieLens website;

- **Book-Crossing** contains 1 million ratings (ranging from 0 to 10) of books in the Book Crossing community;

- **Last.FM** contains musician listening information from a set of 2 thousand users from Last.fm online music system;

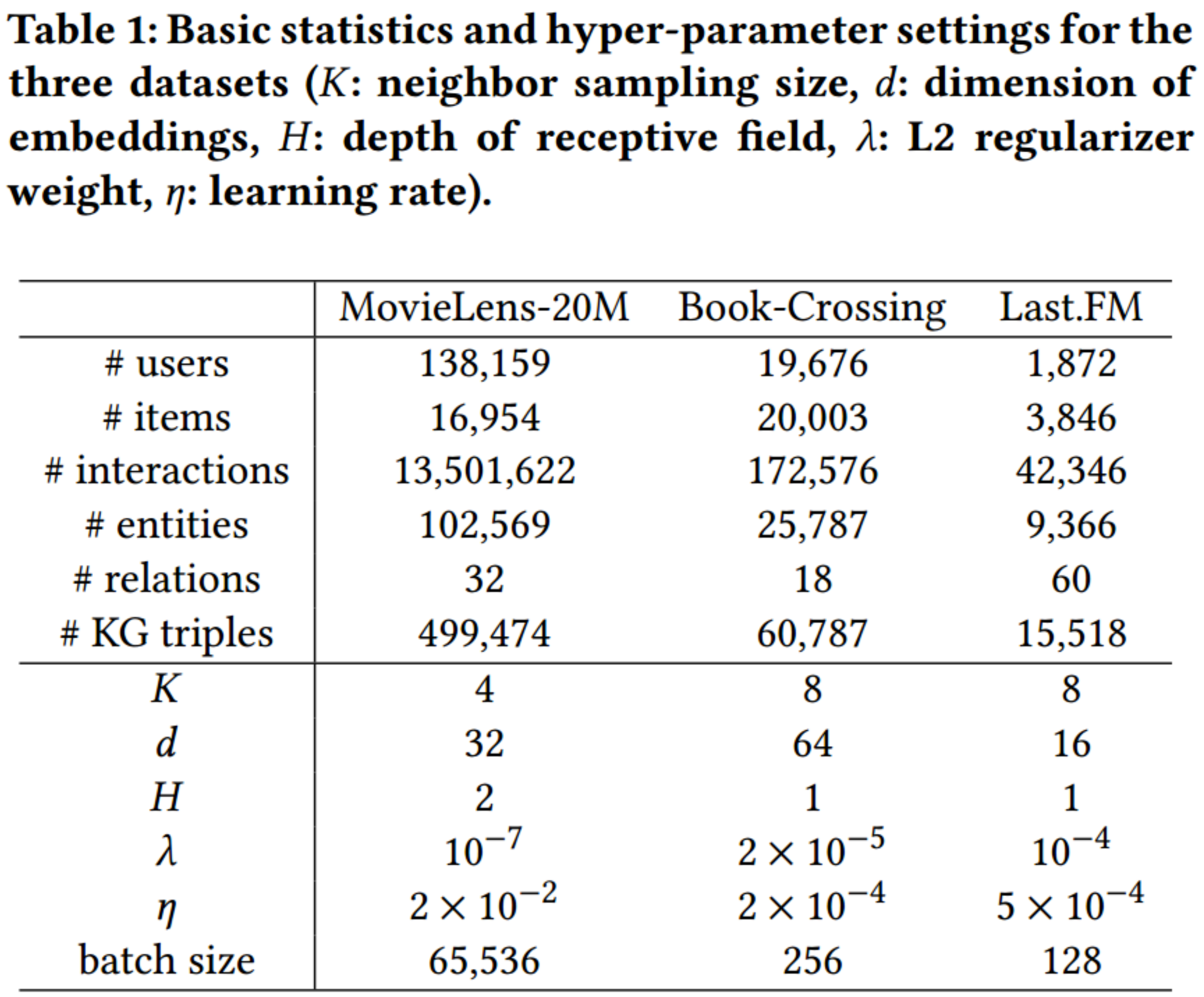


Table 1: Basic statistics and hyper-parameter settings for the three datasets (K: neighbor sampling size, d: dimension of embeddings, H: depth of receptive field, λ: L2 regularizer weight, η: learning rate).

	MovieLens-20M	Book-Crossing	Last.FM
# users	138,159	19,676	1,872
# items	16,954	20,003	3,846
# interactions	13,501,622	172,576	42,346
# entities	102,569	25,787	9,366
# relations	32	18	60
# KG triples	499,474	60,787	15,518
K	4	8	8
d	32	64	16
H	2	1	1
λ	10^{-7}	2×10^{-5}	10^{-4}
η	2×10^{-2}	2×10^{-4}	5×10^{-4}
batch size	65,536	256	128

Baselines

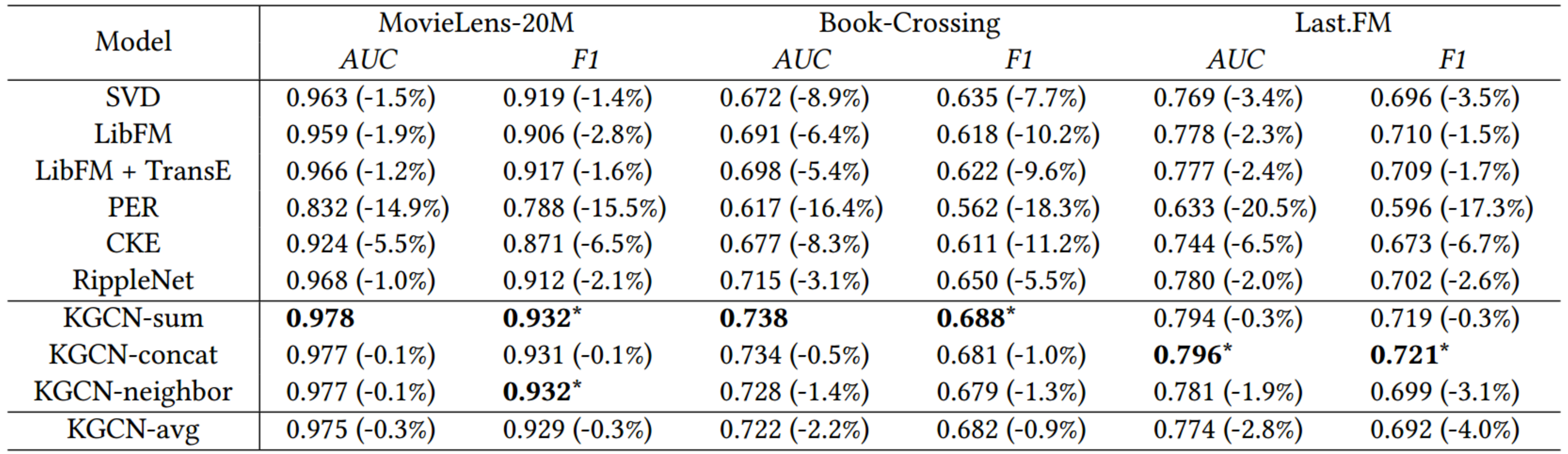
- **SVD** is a classic CF-based model using inner product to model user-item interactions;
- **LibFM** is a feature-based factorization model in CTR scenarios;
- **LibFM + TransE** extends LibFM by attaching an entity representation learned by TransE to each user-item pair;
- **PER** treats the KG as heterogeneous information networks and extracts meta-path based features to represent the connectivity between users and items;
- **CKE** combines CF with structural, textual, and visual knowledge in a unified framework for recommendation;
- **RippleNet** is a memory-network-like approach that propagates users' preferences on the KG for recommendation;

Motivation

Method	disadvantage
Knowledge Graph embedding (KGE)	focus on modeling rigorous semantic relatedness, which are more suitable for KG completion and link prediction rather than recommendation.
PER and FMG	treat KG as a heterogeneous information network, and extract different meta-graph based latent features to represent the connectivity between users and items. However, relying on manually designed meta-graphs, which are hardly to be optimal.
RippleNet	A memory-network-like model, the embedding matrix of a relation R can hardly be trained to capture the sense of importance. In addition, the size of ripple set may go unpredictably with the increase of the size of KG, which incurs heavy computation and storage overhead

Results

Table 2: The results of *AUC* and *F*1 in CTR prediction.

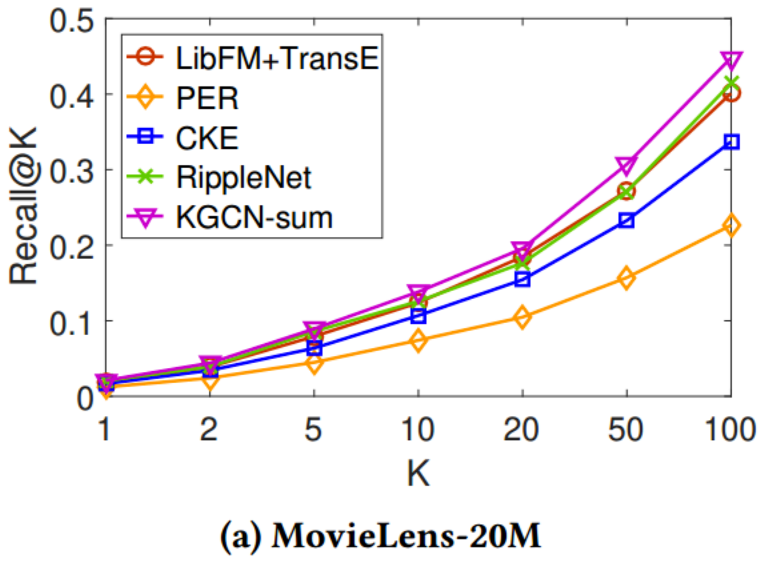


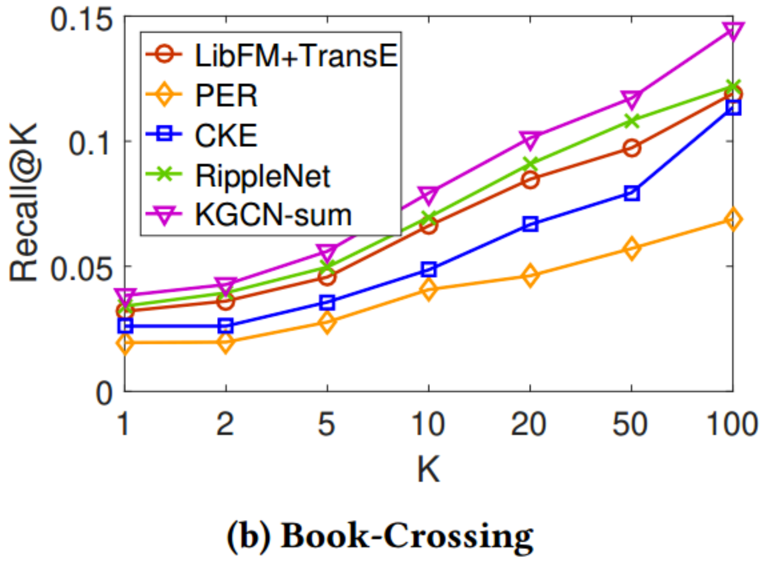
Model	MovieLens-20M		Book-Crossing		Last.FM	
	AUC	*F1*	*AUC*	*F1*	*AUC*	*F1*
SVD	0.963 (-1.5%)	0.919 (-1.4%)	0.672 (-8.9%)	0.635 (-7.7%)	0.769 (-3.4%)	0.696 (-3.5%)
LibFM	0.959 (-1.9%)	0.906 (-2.8%)	0.691 (-6.4%)	0.618 (-10.2%)	0.778 (-2.3%)	0.710 (-1.5%)
LibFM + TransE	0.966 (-1.2%)	0.917 (-1.6%)	0.698 (-5.4%)	0.622 (-9.6%)	0.777 (-2.4%)	0.709 (-1.7%)
PER	0.832 (-14.9%)	0.788 (-15.5%)	0.617 (-16.4%)	0.562 (-18.3%)	0.633 (-20.5%)	0.596 (-17.3%)
CKE	0.924 (-5.5%)	0.871 (-6.5%)	0.677 (-8.3%)	0.611 (-11.2%)	0.744 (-6.5%)	0.673 (-6.7%)
RippleNet	0.968 (-1.0%)	0.912 (-2.1%)	0.715 (-3.1%)	0.650 (-5.5%)	0.780 (-2.0%)	0.702 (-2.6%)
KGCN-sum	**0.978**	**0.932***	**0.738**	**0.688***	0.794 (-0.3%)	0.719 (-0.3%)
KGCN-concat	0.977 (-0.1%)	0.931 (-0.1%)	0.734 (-0.5%)	0.681 (-1.0%)	**0.796***	**0.721***
KGCN-neighbor	0.977 (-0.1%)	**0.932***	0.728 (-1.4%)	0.679 (-1.3%)	0.781 (-1.9%)	0.699 (-3.1%)
KGCN-avg	0.975 (-0.3%)	0.929 (-0.3%)	0.722 (-2.2%)	0.682 (-0.9%)	0.774 (-2.8%)	0.692 (-4.0%)

* Statistically significant improvement by unpaired two-sample t-test with $p = 0.1$.

- The performance of KG-free baselines, SVD and LibFM, are actually better than the two KG-aware baselines PER and CKE, which indicates that PER and CKE cannot make full use of the KG with manually designed meta-paths and TransRlike regularization.
- PER performs worst among all baselines, since it is hard to define optimal meta-paths in reality.

Results





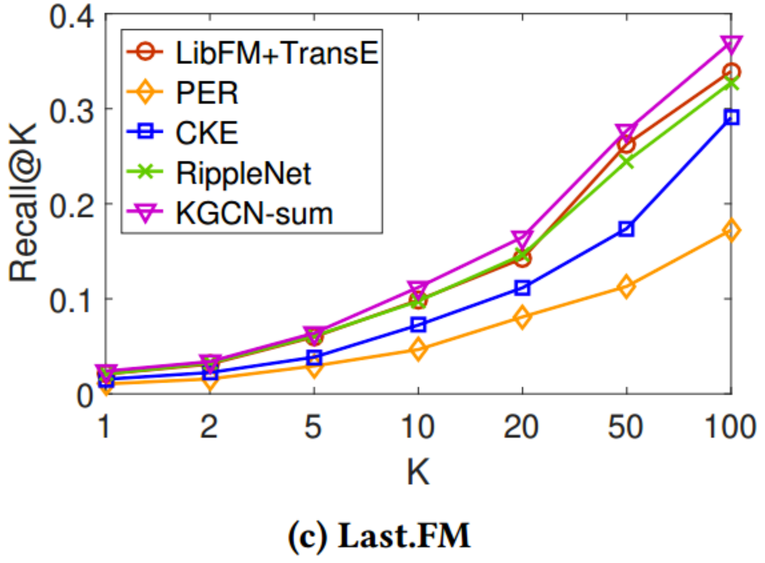


Figure 2: The results of *Recall@K* in top-*K* recommendation.

- In general, the KGCN-sum acquires the best Recall on all the datasets, while the improvement of recall is still limitation;
- Based on the recall curve, the prediction of recommendation is rather difficult;
- LibFM+TransE has a good preformance compared with other three method except KGCN, which proofs that the introduction of KG is helpful for recommendation in general;

Summary & Conclusion

- KGCN extends non-spectral GCN approaches to the knowledge graph by aggregating neighborhood information selectively and biasedly, which is able to learn both structure information and semantic information of the KG as well as users’ personalized and potential interests；

- The KG construction relying on Microsoft Satori, whose quality effect the final results directly, and we have to construct KG first then to recommend, to some fields especially for professional fields, maybe it is impractical;

- Fixed size set of neighbors can be improved in follow-on works;

Q&A

Thanks