陈冲

- 清华大学计算机系，智能控制与技术国家重点实验室，信息检索组（THUIR），博士三年级研究生，导师为张敏副教授。
- 主要研究方向包括基于深度学习的推荐系统，可解释推荐系统，以及高效快速的推荐系统。
- 在WWW，SIGIR，WSDM，TOIS，AAAI等人工智能国际会议和期刊上发表了多篇学术论文。
- 在Github上开源维护多个推荐系统相关的工具包

Non-Sampling Learning for Personalized Recommendation

Chong Chen, Min Zhang

Department of Computer Science and Technology,

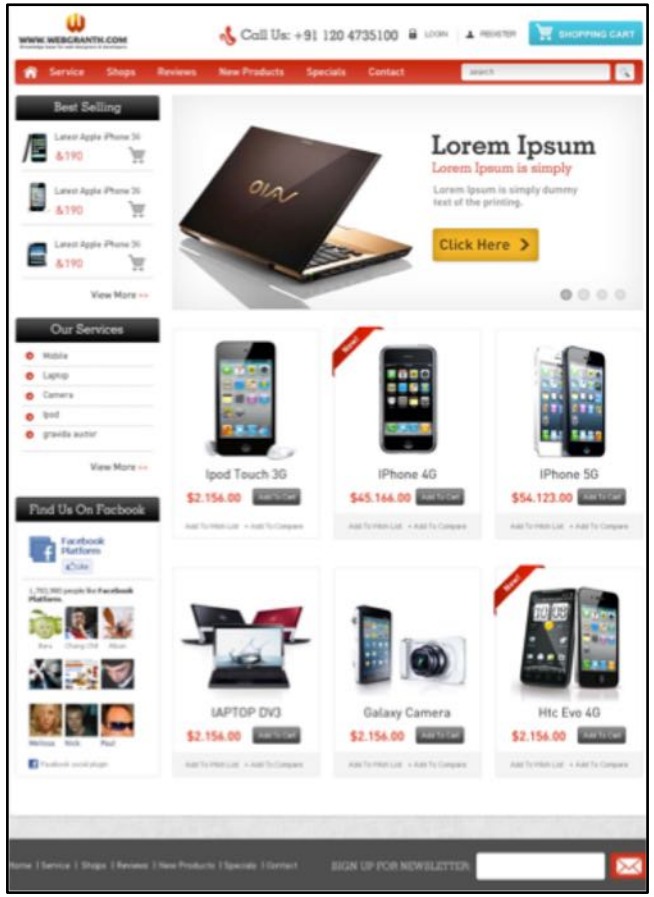
Tsinghua University

Outline

- Background
- Negative-Sampling *VS* Non-Sampling Learning
- Efficient Non-Sampling Learning Method
- Recommendation Models with Efficient Non-Sampling Learning
- Discussion

Background (1):

Value of Recommender System (RecSys)



- RecSys has become a major monetization tool for customer-oriented online services
  - E.g., E-commerce, News Portal, Social Networks, etc.
- Some statistics:
  - YouTube homepage: 60%+ clicks [Davidson et al. 2010]
  - Netflix: 80%+ movie watches, 1billion+ value/year [Gomze-Uribe et al 2016]
  - Amazon: 30%+ page views [Smith and Linden, 2017]

Background (2):

Users' Sparse Feedback Information

- More commonly, users interact with items through implicit feedback
  - E.g., users' viewing records and purchase history, etc.
- It is difficult to utilize implicit feedback data as it is binary and only has positive examples
  - "Negative Information" in implicit feedback
- Users usually rate or click a small set of items compared to hundreds of millions of items in the system

Utilizing Implicit Feedback Data

-Negative-Sampling *VS* Non-Sampling Learning

- Two strategies have been widely used in previous studies:
  - Negative sampling strategy that samples negative instances from unlabeled data
  - Non-Sampling (whole-data based) strategy that sees all the unlabeled data as negative

	Advantages	Disadvantages
Negative sampling	Less training samples and fast training speed	Weak robustness, which may decrease the model's performance
Non-Sampling	leverages the full data with a potentially better coverage	Inefficiency is an issue of traditional non-sampling method

Progresses and limitations in Neural RecSys Models

Complex Neural Network

- ✓✓ Exploring new deep learning architectures for Rec. Sys.
  - Attention, MLP, CNN, etc
  - Superior ability to complex network structures
- ✓✓ With substantial number of parameters

based

Can we find some solutions to *efficiently* learn a neural recommendation model *without sampling*?

Negative Sampling

- ✓✓ Not robust
- ✓✓ Difficult to achieve the optimal performance in practical applications

Complexity Issue of Non-sampling Learning

- In implicit data, the user-item interactions **R** is defined as:

$$R_{uv} = \begin{cases} 1, & \text{if interaction (user } u, \text{ item } v) \text{ is observed;} \\ 0, & \text{otherwise.} \end{cases}$$

- To learn model parameters, Hu et al. introduced a weighted regression loss, which associates a confidence to each prediction:

$$\begin{aligned} \mathcal{L}_{\mathcal{I}}(\Theta) &= \sum_{u \in B} \sum_{v \in V} c_{uv}^{I} (R_{uv} - \hat{R}_{uv})^2 \\ &= \sum_{u \in B} \sum_{v \in V} c_{uv}^{I} (R_{uv}^2 - 2 R_{uv} \hat{R}_{uv} + \hat{R}_{uv}^2) \end{aligned}$$

Complexity: $O(|\mathbf{B}||\mathbf{V}|d)$

Efficient Non-sampling Learning Theorem

THEOREM 1: *For a generalized matrix factorization framework whose prediction function is:*

$$\hat{y}_{uv} = \mathbf{h}^T (\mathbf{p}_u \odot \mathbf{q}_v)$$

where $\mathbf{p}_u \in \mathbb{R}^d$ *and* $\mathbf{q}_v \in \mathbb{R}^d$ *are latent vectors of user* u *and item* v, $\odot$ *denotes the element-wise product of vectors, the gradient of loss Eq.(3) is exactly equal to that of:*

$$\tilde{\mathcal{L}}(\Theta) = \sum_{u \in \mathbf{U}} \sum_{v \in \mathbf{V}^+} \left((c_v^+ - c_v^-) \hat{y}_{uv}^2 - 2 c_v^+ \hat{y}_{uv} \right) + \sum_{i=1}^{d} \sum_{j=1}^{d} \left((h_i h_j) \left(\sum_{u \in \mathbf{U}} p_{u,i} p_{u,j} \right) \left(\sum_{v \in \mathbf{V}} c_v^- q_{v,i} q_{v,j} \right) \right)$$

if the instance weight c_{uv} *is simplified to* c_v.

$$O(|\mathbf{B}||\mathbf{V}|d)$$

#Hidden

$$O((|\mathbf{B}| + |\mathbf{V}|)d^2 + |\mathcal{R}_\mathbf{B}|d)$$

#Users #Items #Positive

Loss Inference

$$\mathcal{L}_{\mathcal{I}}(\Theta) = \sum_{u \in B} \sum_{v \in V} c_{uv}^{I} (R_{uv}^2 - 2R_{uv}\hat{R}_{uv} + \hat{R}_{uv}^2)$$

User actions, R_{uv}: (0,1)

R_{uv} =0 for neg. feedbacks,

$$\begin{aligned}
\mathcal{L}_{\mathcal{I}}(\Theta) &= const - 2\sum_{u \in B} \sum_{v \in V^+} c_{uv}^{I+} \hat{R}_{uv} + \sum_{u \in B} \sum_{v \in V} c_{uv} \hat{R}_{uv}^2 \\
&= const - 2\sum_{u \in B} \sum_{v \in V^+} c_{uv}^{I+} \hat{R}_{uv} + \sum_{u \in B} \sum_{v \in V^+} c_{uv}^{I+} \hat{R}_{uv}^2 + \sum_{u \in B} \sum_{v \in V^-} c_{uv}^{I-} \hat{R}_{uv}^2 \\
&= const - 2\sum_{u \in B} \sum_{v \in V^+} c_{uv}^{I+} \hat{R}_{uv} + \sum_{u \in B} \sum_{v \in V^+} c_{uv}^{I+} \hat{R}_{uv}^2 + \sum_{u \in B} \sum_{v \in V} c_{uv}^{I-} \hat{R}_{uv}^2 - \sum_{u \in B} \sum_{v \in V^+} c_{uv}^{I-} \hat{R}_{uv}^2 \\
&= const + \underbrace{\sum_{u \in B} \sum_{v \in V} c_{uv}^{I-} \hat{R}_{uv}^2}_{\mathcal{L}_{\mathcal{I}}^{\mathcal{A}}(\Theta)} + \sum_{u \in B} \sum_{v \in V^+} \left((c_{uv}^{I+} - c_{uv}^{I-}) \hat{R}_{uv}^2 - 2c_{uv}^{I+} \hat{R}_{uv} \right)
\end{aligned}$$

Bottleneck

Loss Inference

$$\mathcal{L}_{\mathcal{I}}(\Theta) = \sum_{u \in B} \sum_{v \in V} c_{uv}^{I} (R_{uv}^2 - 2R_{uv}\hat{R}_{uv} + \hat{R}_{uv}^2)$$

$$\mathcal{L}_{\mathcal{I}}(\Theta) = const + \underbrace{\sum_{u \in B} \sum_{v \in V} c_{uv}^{I-} \hat{R}_{uv}^2}_{\mathcal{L}_{\mathcal{I}}^{\mathcal{A}}(\Theta)} + \sum_{u \in B} \sum_{v \in V^+} \left((c_{uv}^{I+} - c_{uv}^{I-}) \hat{R}_{uv}^2 - 2c_{uv}^{I+} \hat{R}_{uv} \right)$$

Bottleneck

$$\sum_{i=1}^{d} \sum_{j=1}^{d} \left(\left(\sum_{u \in B} p_{u,i}^{I} p_{u,j}^{I} \right) \left(\sum_{v \in V} c_{v}^{I-} q_{v,i} q_{v,j} \right) (h_{1,i} h_{1,j}) \right)$$

$$\hat{R}_{uv}^2 = \sum_{i=1}^{d} h_{1,i} p_{u,i}^{I} q_{v,i} \sum_{j=1}^{d} h_{1,j} p_{u,j}^{I} q_{v,j}$$

$$= \sum_{i=1}^{d} \sum_{j=1}^{d} \left(p_{u,i}^{I} p_{u,j}^{I} \right) \left(q_{v,i} q_{v,j} \right) \left(h_{1,i} h_{1,j} \right)$$

Independent, opportunity to speed-up by precomputing the two terms.

Loss Inference

$$\mathcal{L}_{\mathcal{I}}(\Theta) = \sum_{u \in B} \sum_{v \in V} c_{uv} (R_{uv} - \hat{R}_{uv})^2$$

$$O(|\mathbf{B}||\mathbf{V}|d)$$

$$\tilde{\mathcal{L}}_{\mathcal{I}}(\Theta) = \sum_{i=1}^{d} \sum_{j=1}^{d} \left(\left(\sum_{u \in B} p_{u,i}^{I} p_{u,j}^{I} \right) \left(\sum_{v \in V} c_v^{I-} q_{v,i} q_{v,j} \right) (h_{1,i} h_{1,j}) \right) + \sum_{u \in B} \sum_{v \in V^+} \left((1 - c_v^{I-}) \hat{R}_{uv}^2 - 2 \hat{R}_{uv} \right)$$

#Hidden

$$O((|\mathbf{B}| + |\mathbf{V}|)d^2 + |\mathcal{R}_{\mathbf{B}}|d)$$

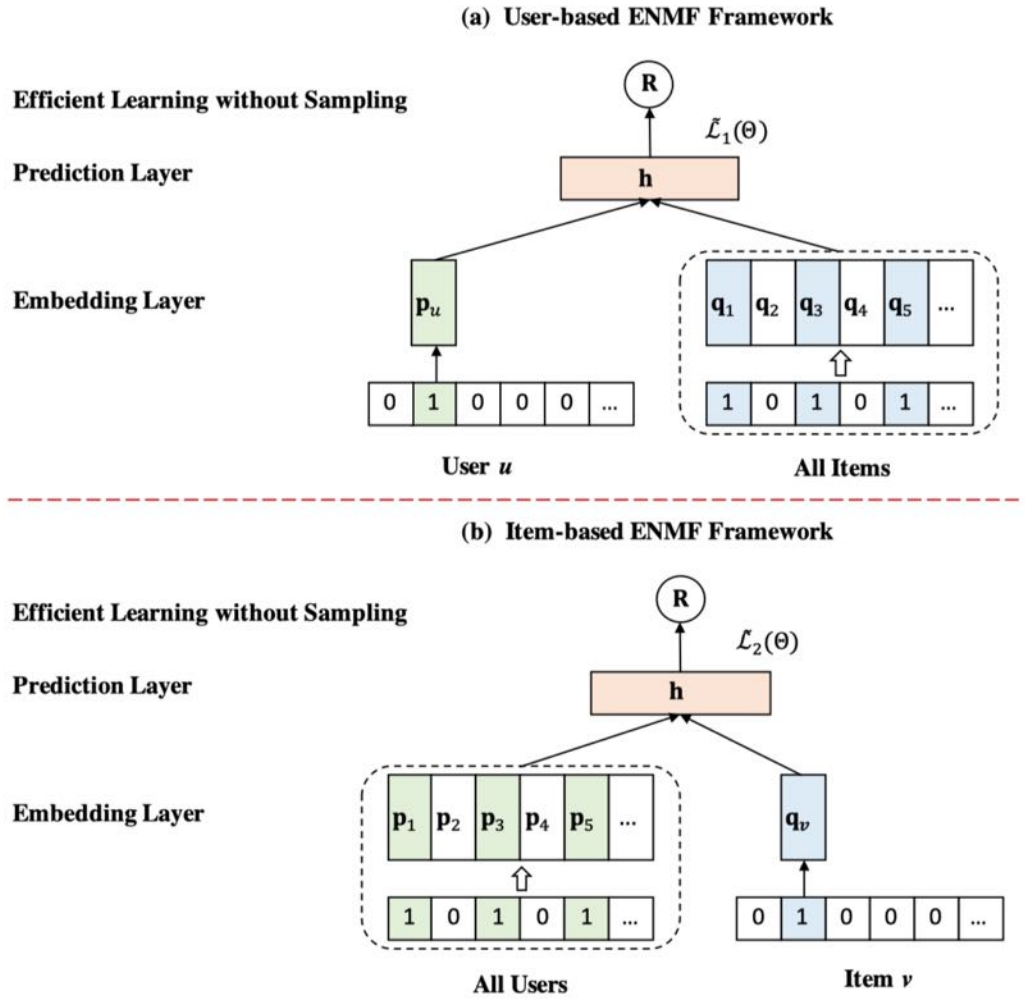
#Users #Items #Positive

No approximation is introduced, the optimization results are exactly the same with the original whole-data based regression loss

Recommendation Models with Efficient Non-Sampling Learning

- Plain recommendation scenario (only the ID information is utilized):
  - We propose ENMF, 5%+ better and 30+ times faster than SOTA method (TOIS accept)
- Social-aware recommendation scenario:
  - We propose EATNN, 4%+ better and 7+ times faster than SOTA method (SIGIR 2019 accept)
- Multi-Behavior recommendation scenario:
  - We propose EHCF, 40%+ better and 10+ times faster than SOTA method (AAAI 2020 accept)
- Context-aware recommendation scenario:
  - We propose ENSFM, 9%+ better and 5+ times faster than SOTA method (WWW 2020 accept)
- Knowledge enhanced recommendation scenario:
  - We propose JNSKR, 5%+ better and 20+ times faster than SOTA method (SIGIR 2020 accept)

Plain Recommendation Scenario



- Efficient Neural Matrix Factorization (ENMF) without sampling
- Inputs:
  - User based: a user and all his/her item interactions
  - Item-based: an item with all its user interactions

Chong Chen, Min Zhang, Yongfeng Zhang, Yiqun Liu and Shaoping Ma. **Efficient Neural Matrix Factorization without Sampling for Recommendation**. ACM Transactions on Information Systems. (TOIS Vol. 38, No. 2, Article 14)

Plain Recommendation Scenario

- User-based efficient loss:

$$\tilde{\mathcal{L}}_1(\Theta) = \sum_{u \in \mathbf{B}} \sum_{v \in \mathbf{V}^+} \left((c_v^+ - c_v^-) \hat{R}_{uv}^2 - 2 c_v^+ \hat{R}_{uv} \right) + \sum_{i=1}^{d} \sum_{j=1}^{d} \left((h_i h_j) \left(\sum_{u \in \mathbf{B}} p_{u,i} p_{u,j} \right) \left(\sum_{v \in \mathbf{V}} c_v^- q_{v,i} q_{v,j} \right) \right)$$

  - Complexity: $O((|\mathbf{U}| + \frac{|\mathbf{U}||\mathbf{V}|}{|\mathbf{B}|})d^2 + |\mathcal{R}|d)$

- Item-based efficient loss:

$$\tilde{\mathcal{L}}_2(\Theta) = \sum_{u \in \mathbf{U}^+} \sum_{v \in \mathbf{B}} \left((c_v^+ - c_v^-) \hat{R}_{uv}^2 - 2 c_v^+ \hat{R}_{uv} \right) + \sum_{i=1}^{d} \sum_{j=1}^{d} \left((h_i h_j) \left(\sum_{u \in \mathbf{U}} p_{u,i} p_{u,j} \right) \left(\sum_{v \in \mathbf{B}} c_v^- q_{v,i} q_{v,j} \right) \right)$$

  - Complexity: $O((|\mathbf{V}| + \frac{|\mathbf{U}||\mathbf{V}|}{|\mathbf{B}|})d^2 + |\mathcal{R}|d)$

Experimental settings

- Datasets:
- Baselines:
  - BPR(UAI'09)
  - WMF (ICDM'08)
  - ExpoMF (WWW'16)
  - GMF (WWW'17)
  - NCF (WWW'17)
  - ConvNCF (IJCAI'18)
- Evaluation methods: HR@K, NDCG@K, K= 50, 100, 200

Dataset	#User	#Item	#Interaction	Density
Ciao	7,267	11,211	157,995	0.19%
Epinion	20,608	23,585	454,002	0.09%
Movielens	6,940	3,706	1,000,209	4.47%

Characteristics	MP	ItemKNN	BPR	WMF	ExpoMF	GMF	NCF	ConvNCF	ENMF
Top-N Recommendation	√	√	√	√	√	√	√	√	√
Neural Model	\	\	\	\	\	√	√	√	√
Whole-data based	\	\	\	√	√	\	\	\	√

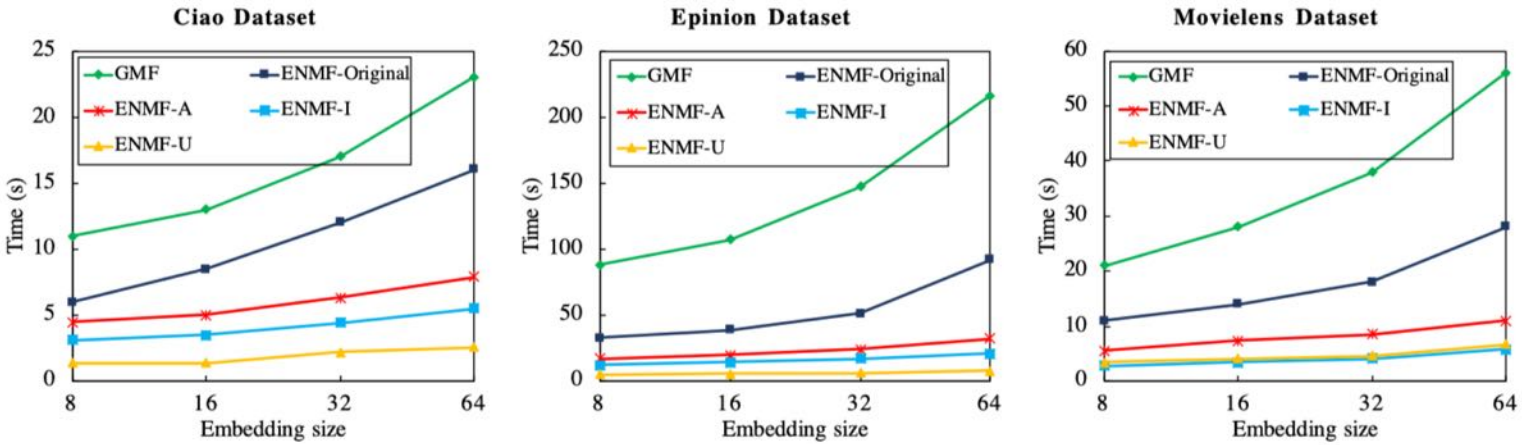
Model Comparisons

Ciao	HR@50	HR@100	HR@200	NDCG@50	NDCG@100	NDCG@200	RI
MP	0.1047	0.1384	0.1776	0.0396	0.0452	0.0506	+67.96%
ItemKNN	0.1453	0.1884	0.2468	0.0497	0.0581	0.0668	+26.14%
BPR	0.1531	0.1930	0.2558	0.0517	0.0598	0.0685	+21.91%
WMF	0.1587	0.2011	0.2608	0.0562	0.0631	0.0714	+16.40%
ExpoMF	0.1602	0.1994	0.2613	0.0569	0.0626	0.0709	+16.41%
GMF	0.1668	0.2103	0.2674	0.0633	0.0687	0.0752	+9.36%
NCF	0.1651	0.2108	0.2712	0.0629	0.0695	0.0764	+8.84%
ConvNCF	0.1682	0.2237	0.2741	0.0641	0.0714	0.0787	+5.90%
ENMF-U	**0.1750****	**0.2296****	**0.2945****	**0.0651****	**0.0741****	**0.0830****	–
ENMF-I	**0.1749****	**0.2311****	**0.2946****	**0.0643***	**0.0734****	**0.0823****	–
ENMF-A	**0.1757****	**0.2331****	**0.3015****	**0.0662****	**0.0753****	**0.0850****	–

Epinion	HR@50	HR@100	HR@200	NDCG@50	NDCG@100	NDCG@200	RI
MP	0.0661	0.1068	0.1659	0.0234	0.0299	0.0382	+153.96%
ItemKNN	0.1312	0.2082	0.2929	0.0455	0.0563	0.0682	+34.41%
BPR	0.1708	0.2338	0.3007	0.0548	0.0646	0.0747	+17.04%
WMF	0.1765	0.2384	0.3158	0.0605	0.0685	0.0789	+11.07%
ExpoMF	0.1784	0.2368	0.3064	0.0602	0.0691	0.0781	+11.70%
GMF	0.1811	0.2513	0.3388	0.0613	0.0739	0.0845	+5.52%
NCF	0.1816	0.2534	0.3442	0.0621	0.0750	0.0869	+4.08%
ConvNCF	0.1833	0.2510	0.3418	0.0617	0.0742	0.0851	+4.87%
ENMF-U	**0.1893****	**0.2647****	**0.3523****	**0.0639****	**0.0761****	**0.0883****	–
ENMF-I	**0.1888****	**0.2667****	**0.3534****	**0.0634****	**0.0759****	**0.0884****	–
ENMF-A	**0.1911****	**0.2688****	**0.3546****	**0.0648****	**0.0773****	**0.0893****	–

- Performance comparison on two datasets for all methods
- ENMF Consistently significantly outperforms the best baseline
- Improves more than 4%

Efficiency Analysis



Model	*Ciao*			*Epinion*			*Movielens*		
	S	**I**	**T**	**S**	**I**	**T**	**S**	**I**	**T**
GMF	23s	300	115m	216s	500	30h	56s	500	7h
NCF	34s	300	170m	305s	500	42h	91s	500	12h
ConvNCF	88s	300	440m	510s	500	70h	246s	500	34h
ENMF-Original	16s	300	80m	65s	200	216m	28s	300	140m
ENMF-U	2.6s	300	13m	8s	200	27m	6.7s	300	34m
ENMF-I	5.5s	300	28m	21s	200	70m	5.8s	300	29m
ENMF-A	8s	150	20m	32s	100	53m	11s	50	9m

Comparison of runtime

s:second; m: minute; h: hour; d: day

S: training time for a single iteration;

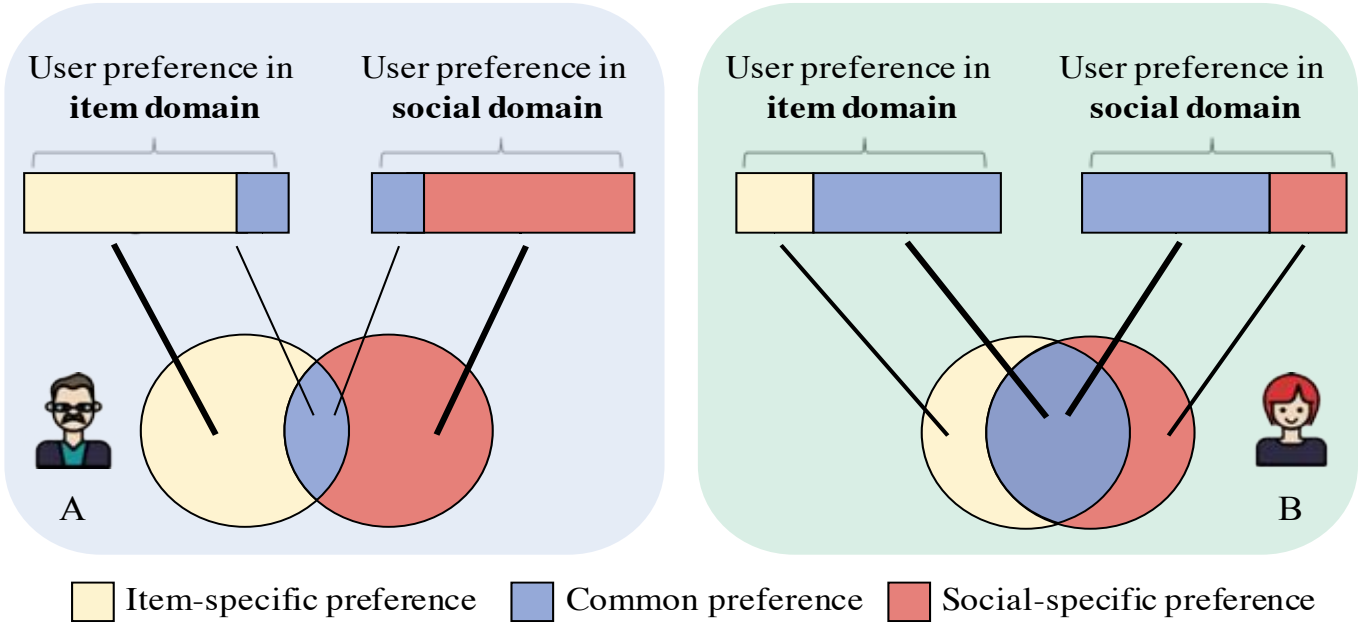
I: Overall iterations;

T: Total time

10 times faster

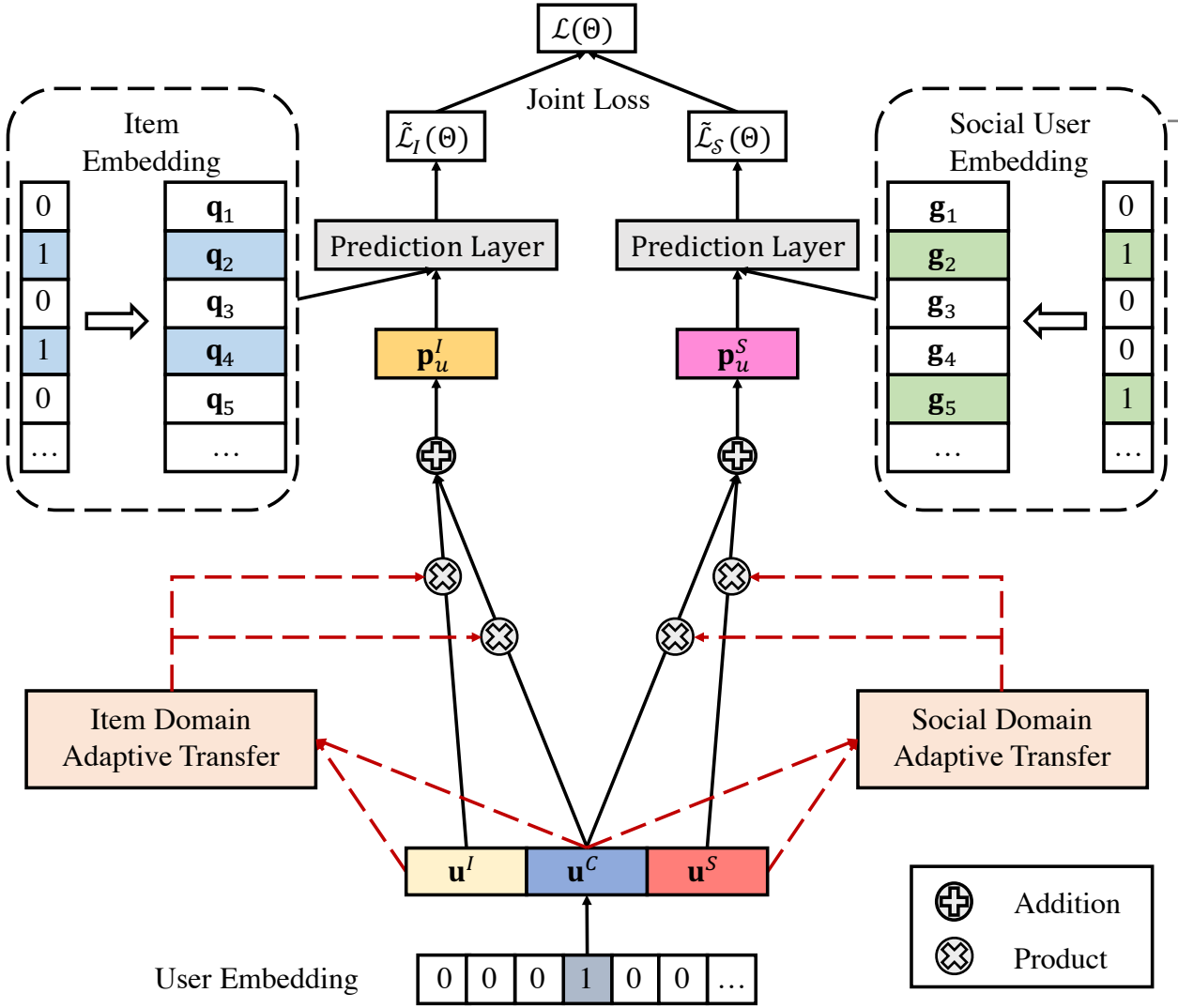
Social-aware Recommendation Scenario

- Leveraging users' social connections helps reduce sparsity
- Improve the performance of recommender systems
  → Social-aware recommendation



- But the preference sharing between item domain and social domain are varied for different users in real life.

Social-aware Recommendation Scenario



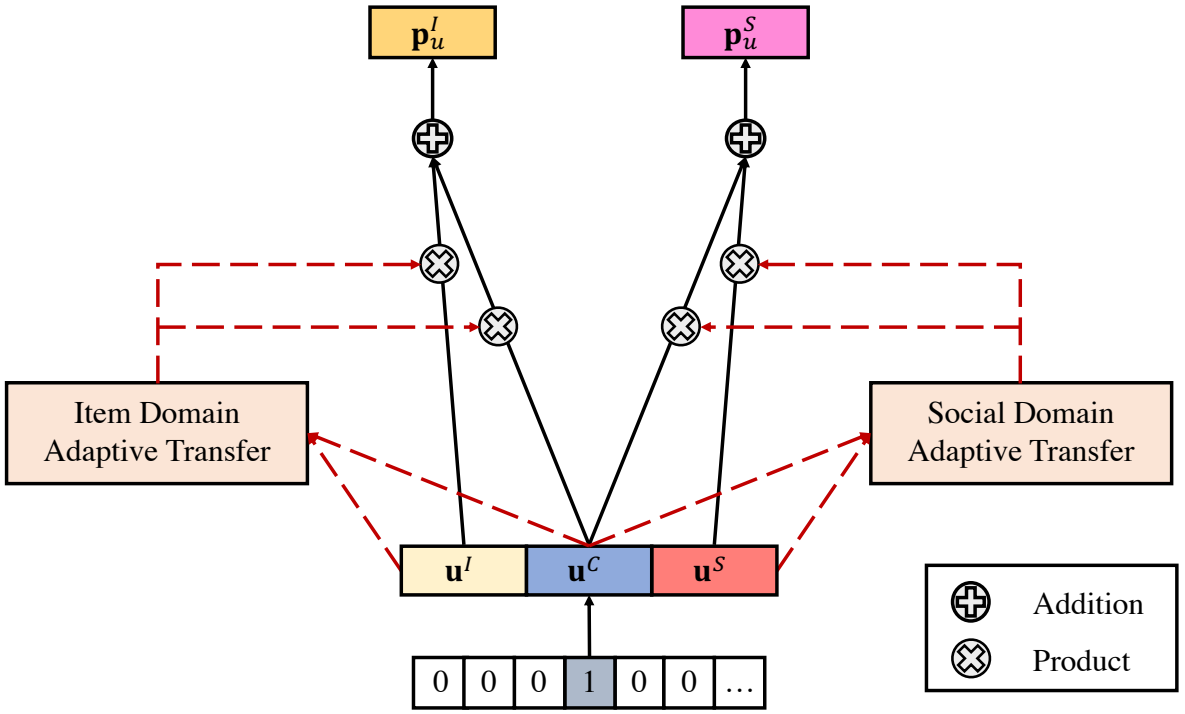
- Efficient Adaptive Transfer Neural Network (EATNN)

$$\hat{R}_{uv} = \mathbf{h}_I^T(\mathbf{p}_u^I \odot \mathbf{q}_v); \quad \hat{X}_{ut} = \mathbf{h}_S^T(\mathbf{p}_u^S \odot \mathbf{g}_t)$$

Chong Chen, Min Zhang, Chenyang Wang, Weizhi Ma, Minming Li, Yiqun Liu and Shaoping Ma. **An Efficient Adaptive Transfer Neural Network for Social-aware Recommendation**. The 42th International ACM SIGIR Conference on Research and Development in Information Retrieval. (**SIGIR 2019**)

https://github.com/chenchongthu/EATNN

Attention-based Adaptive Transfer



$\mathbf{u}^I$	item-specific latent factor vector of user u
$\mathbf{u}^S$	social-specific latent factor vector of user u
$\mathbf{u}^C$	common latent factor vector of user u

Attention score:

$$\alpha^*_{(C,u)} = \mathbf{h}_\alpha^T \sigma(\mathbf{W}_\alpha \mathbf{u}^C + \mathbf{b}_\alpha); \quad \alpha^*_{(I,u)} = \mathbf{h}_\alpha^T \sigma(\mathbf{W}_\alpha \mathbf{u}^I + \mathbf{b}_\alpha)$$

$$\beta^*_{(C,u)} = \mathbf{h}_\beta^T \sigma(\mathbf{W}_\beta \mathbf{u}^C + \mathbf{b}_\beta); \quad \beta^*_{(S,u)} = \mathbf{h}_\beta^T \sigma(\mathbf{W}_\beta \mathbf{u}^S + \mathbf{b}_\beta)$$

Normalization:

$$\alpha_{(C,u)} = \frac{\exp(\alpha^*_{(C,u)})}{\exp(\alpha^*_{(C,u)}) + \exp(\alpha^*_{(I,u)})} = 1 - \alpha_{(I,u)}$$

$$\beta_{(C,u)} = \frac{\exp(\beta^*_{(C,u)})}{\exp(\beta^*_{(C,u)}) + \exp(\beta^*_{(I,u)})} = 1 - \beta_{(S,u)}$$

User representation:

$$\mathbf{p}_u^I = \alpha_{(I,u)}\mathbf{u}^I + \alpha_{(C,u)}\mathbf{u}^C; \quad \mathbf{p}_u^S = \beta_{(S,u)}\mathbf{u}^S + \beta_{(C,u)}\mathbf{u}^C$$

Joint Learning

For item domain:

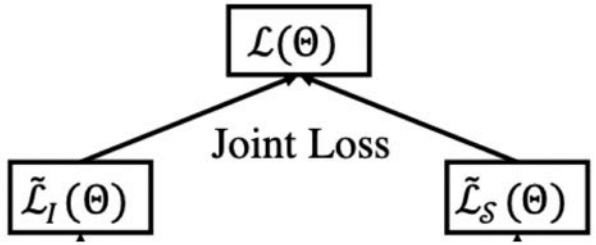
$$\tilde{\mathcal{L}}_I(\Theta) = \sum_{i=1}^{d}\sum_{j=1}^{d}\left((h_{I,i}h_{I,j})\left(\sum_{u\in\mathbf{B}} p^I_{u,i}p^I_{u,j}\right)\left(\sum_{v\in\mathbf{V}} c_v^{I-}q_{v,i}q_{v,j}\right)\right) + \sum_{u\in\mathbf{B}}\sum_{v\in\mathbf{V}^+}\left((1-c_v^{I-})\hat{R}_{uv}^2 - 2\hat{R}_{uv}\right)$$

For social domain:

$$\tilde{\mathcal{L}}_S(\Theta) = \sum_{i=1}^{d}\sum_{j=1}^{d}\left((h_{S,i}h_{S,j})\left(\sum_{u\in\mathbf{B}} p^S_{u,i}p^S_{u,j}\right)\left(\sum_{t\in\mathbf{U}} c_t^{S-}g_{t,i}g_{t,j}\right)\right) + \sum_{u\in\mathbf{B}}\sum_{t\in\mathbf{U}^+}\left((1-c_t^{S-})\hat{X}_{ut}^2 - 2\hat{X}_{ut}\right)$$

Joint Learning:

$$\mathcal{L}(\Theta) = \tilde{\mathcal{L}}_{\mathcal{I}}(\Theta) + \mu\tilde{\mathcal{L}}_{\mathcal{S}}(\Theta)$$



Experimental settings

- Datasets:
- Baselines:
  - BPR(UAI'09)
  - ExpoMF(WWW'16)
  - NCF (WWW'17)
  - SBPR (CIKM'14)
  - TranSIV (CIKM'17)
  - SAMN (WSDM'19)
- Evaluation methods: Recall@K, NDCG@K, K=10, 50, 100

	Ciao	*Epinion*	*Flixster*
#User	7,267	20,608	69,251
#Item	11,211	23,585	17,318
#Item Interaction	157,995	454,002	7,940,096
#Social Connection	111,781	351,486	967,195

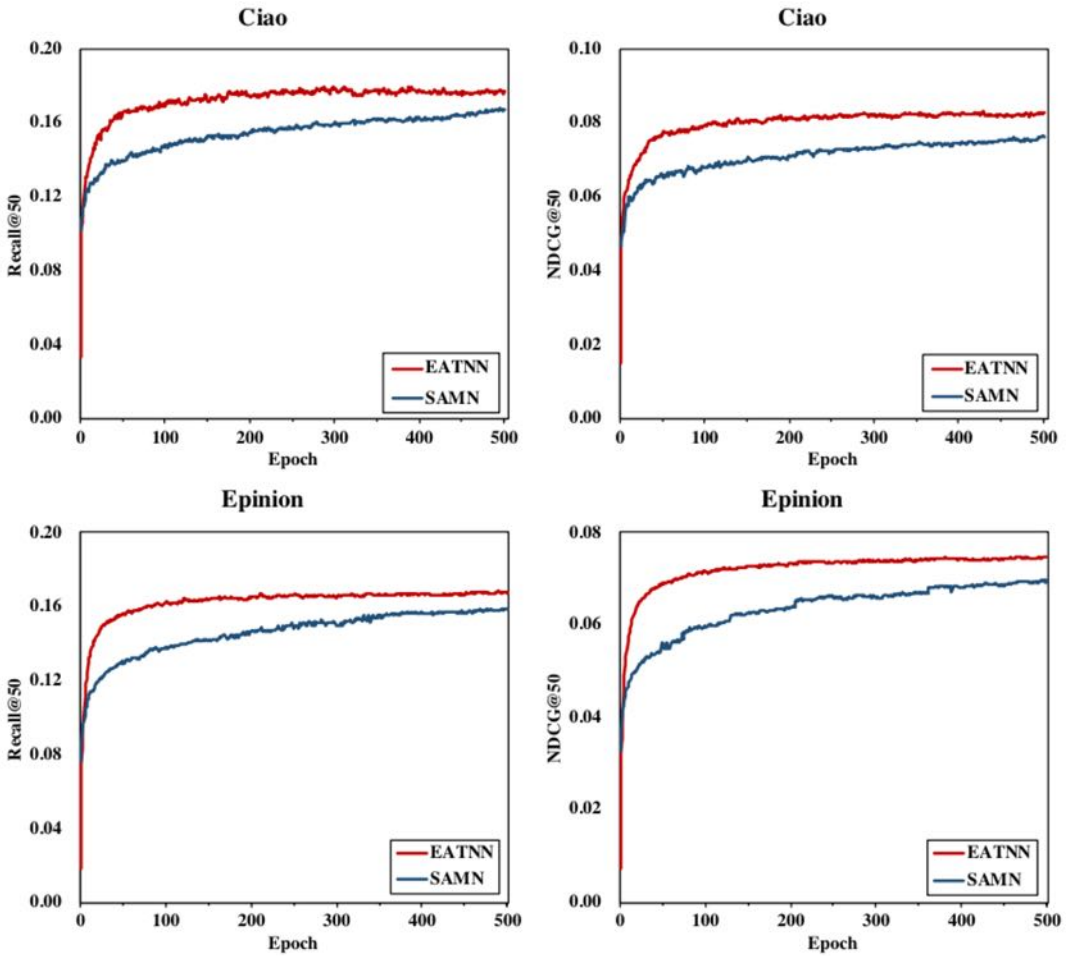
Characteristics	BPR	ExpoMF	NCF	SBPR	TranSIV	SAMN	EATNN
Item domain	√	√	√	√	√	√	√
Social domain	\	\	\	√	√	√	√
Neural model	\	\	√	\	\	√	√
Adaptive transfer	\	\	\	\	\	\	√
Whole-data	\	√	\	\	√	\	√

Model Comparisons

Ciao	Recall@10	Recall@50	Recall@100	NDCG@10	NDCG@50	NDCG@100	RI
BPR	0.0591	0.1600	0.2135	0.0409	0.0688	0.0805	+20.08%
ExpoMF	0.0642	0.1556	0.2050	0.0445	0.0706	0.0816	+17.03%
NCF	0.0667	0.1584	0.2141	0.0456	0.0718	0.0837	+13.84%
SBPR	0.0623	0.1631	0.2146	0.0436	0.0695	0.0832	+16.30%
TranSIV	0.0678	0.1651	0.2184	0.0473	0.0753	0.0865	+10.20%
SAMN	0.0719	0.1671	0.2233	0.0495	0.0768	0.0883	+6.97%
EATNN	**0.0778****	**0.1764****	**0.2305****	**0.0547****	**0.0824****	**0.0943****	–
Epinion	Recall@10	Recall@50	Recall@100	NDCG@10	NDCG@50	NDCG@100	RI
BPR	0.0528	0.1477	0.2115	0.0353	0.0613	0.0751	+21.49%
ExpoMF	0.0611	0.1508	0.2077	0.0422	0.0673	0.0798	+11.82%
NCF	0.0535	0.1489	0.2144	0.0367	0.0624	0.0772	+19.06%
SBPR	0.0547	0.1511	0.2142	0.0387	0.0665	0.0783	+15.71%
TranSIV	0.0631	0.1552	0.2227	0.0423	0.0681	0.0829	+8.49%
SAMN	0.0621	0.1583	0.2274	0.0417	0.0698	0.0842	+7.62%
EATNN	**0.0696****	**0.1675****	**0.2309****	**0.0474****	**0.0749****	**0.0887****	–
Flixster	Recall@10	Recall@50	Recall@100	NDCG@10	NDCG@50	NDCG@100	RI
BPR	0.1733	0.3945	0.5272	0.1612	0.2193	0.2568	+35.88%
ExpoMF	0.2596	0.4488	0.5659	0.2012	0.2633	0.3002	+10.94%
NCF	0.2613	0.4564	0.5632	0.2112	0.2687	0.3075	+8.81%
SBPR	0.2314	0.4517	0.5697	0.1989	0.2514	0.3016	+14.05%
TranSIV	0.2748	0.4633	0.5749	0.2277	0.2804	0.3224	+4.35%
SAMN	0.2767	0.4661	0.5746	0.2316	0.2833	0.3251	+3.51%
EATNN	**0.2948****	**0.4736****	**0.5896****	**0.2401****	**0.2962****	**0.3319****	–

- Performance comparison on three datasets for all methods
- Best Baselines:
  - TranSIV: non-Neural, whole-data
  - SAMN: Neural model, sampled data
- EATNN
  - Consistently significantly outperforms the best baseline

Efficiency Analysis



Comparison of runtime

s:second; m: minute; h: hour; d: day

S: training time for a single iteration;

I: Overall iterations;

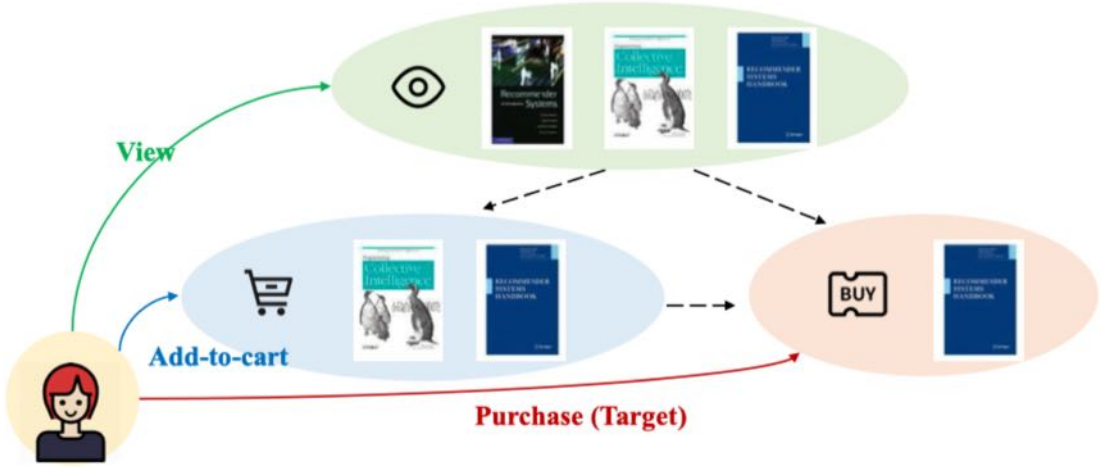
T: Total time

Model	*Ciao*			*Epinion*			*Flixster*		
	S	I	T	S	I	T	S	I	T
TranSIV	55s	50	46m	410s	50	342m	37m	50	31h
SAMN	31s	500	258m	92s	500	767m	56m	200	8d
EATNN	1.8s	200	6m	11s	200	37m	8m	200	27h

7 times faster

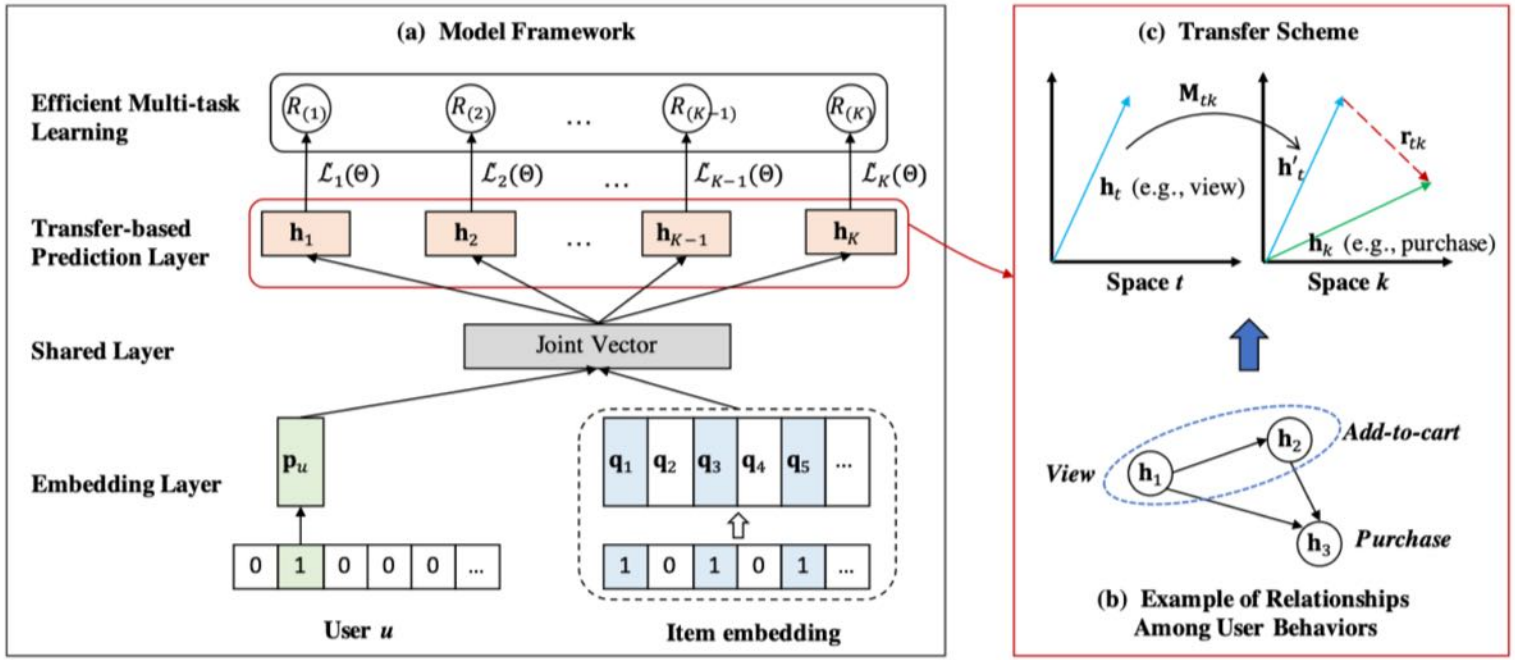
Multi-Behavior Recommendation Scenario

- Heterogeneous (multi-behavior) feedback (e.g., view, click, and purchase) is widespread in many online systems



- There exist strong transfer relations among different behaviors

Multi-Behavior Recommendation Scenario



For the k-th behavior

$$\hat{R}_{(k)uv} = \mathbf{h}_k^T(\mathbf{p}_u \odot \mathbf{q}_v) = \sum_{i=1}^{d} h_{k,i} p_{u,i} q_{v,i}$$

Chong Chen, Min Zhang, Weizhi Ma, Yongfeng Zhang, Yiqun Liu and Shaoping Ma. **Efficient Heterogeneous Collaborative Filtering without Negative Sampling for Recommendation**. The 44th AAAI Conference on Artificial Intelligence. (**AAAI 2020**)

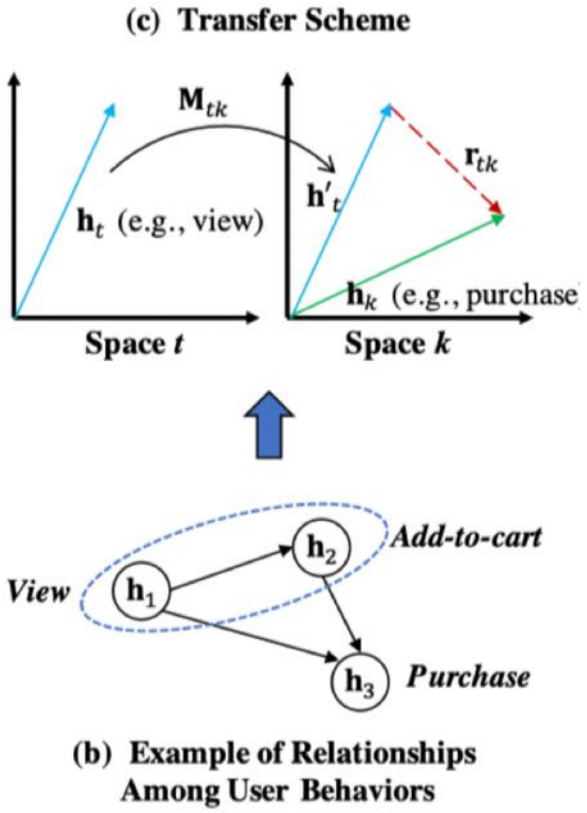
Behavior Transferring

For behavior t to k:

$$f_{\mathbf{h}_t \to \mathbf{h}_k} = \mathbf{h}_t \mathbf{M}_{tk} + \mathbf{r}_{tk}$$

The prediction vector $\mathbf{h}_k$ is:

$$\mathbf{h}_k = \sum_t (f_{\mathbf{h}_t \to \mathbf{h}_k}) = \sum_t (\mathbf{h}_t \mathbf{M}_{tk} + \mathbf{r}_{tk})$$



Multi-task Learning

Loss for the k-th behavior:

$$\tilde{\mathcal{L}}_k(\Theta) = \sum_{u \in \mathbf{B}} \sum_{v \in \mathbf{V}^{k+}} \left((c_{uv}^{k+} - c_{uv}^{k-}) \hat{R}_{(k)uv}^2 - 2 c_{uv}^{k+} R_{(k)uv} \hat{R}_{(k)uv} \right) + \sum_{i=1}^{d} \sum_{j=1}^{d} \left((h_{k,i} h_{k,j}) \left(\sum_{u \in \mathbf{B}} p_{u,i} p_{u,j} \right) \left(\sum_{v \in \mathbf{V}} c_v^{k-} q_{v,i} q_{v,j} \right) \right)$$

Multi-task learning:

$$\mathcal{L}(\Theta) = \sum_{k=1}^{K} \lambda_k \tilde{\mathcal{L}}_k(\Theta)$$

Experimental settings

- Datasets:
- Baselines:
  - BPR(UAI'09)
  - ExpoMF (WWW'16)
  - NCF (WWW'17)
  - CMF (WWW'15)
  - MC-BPR (RecSys'16)
  - NMTR (ICDE'19, TKDE'20)
- Evaluation methods: HR@K, NDCG@K, K=10, 50, 100

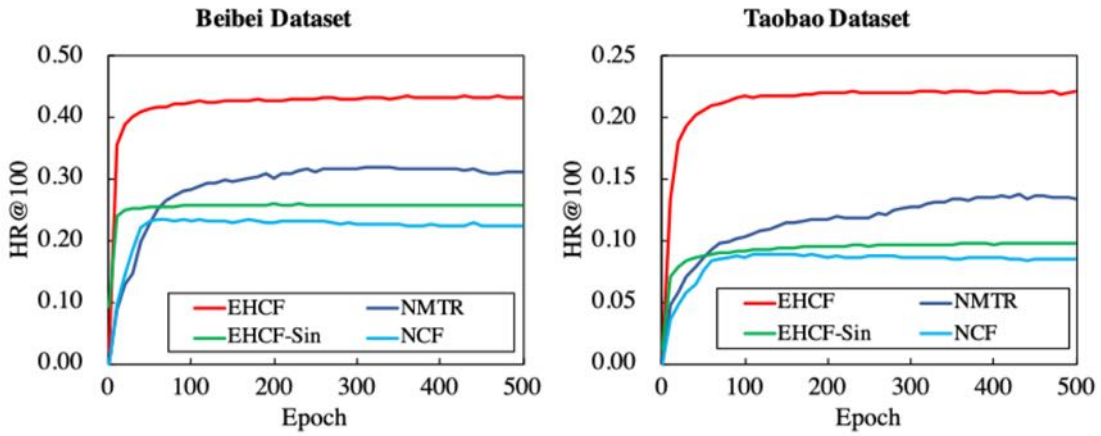
Dataset	#User	#Item	#View	#Add-to-cart	#Purchase
Movielens	6,940	3,706	–	–	1,000,209
Beibei	21,716	7,977	2,412,586	642,622	304,576
Taobao	48,749	39,493	1,548,126	193,747	259,747

Model Comparisons

	Beibei	HR@10	HR@50	HR@100	HR@200	NDCG@10	NDCG@50	NDCG@100	NDCG@200
Single	BPR	0.0437	0.1246	0.2192	0.3057	0.0213	0.0407	0.0539	0.0689
	ExpoMF	0.0452	0.1465	0.2246	0.3282	0.0227	0.0426	0.0553	0.0723
	NCF	0.0441	0.1562	0.2343	0.3583	0.0225	0.0445	0.0584	0.0757
	EHCF-Sin	**0.0464****	**0.1637****	**0.2586****	**0.3743****	**0.0247****	**0.0484****	**0.0639****	**0.0799****
Heterogeneous	CMF	0.0482	0.1582	0.2843	0.4288	0.0251	0.0462	0.0661	0.0852
	MC-BPR	0.0504	0.1743	0.2755	0.3862	0.0254	0.0503	0.0653	0.0796
	NMTR	0.0524	0.2047	0.3189	0.4735	0.0285	0.0609	0.0764	0.0968
	EHCF	**0.0608****	**0.3316****	**0.4312****	**0.5460****	**0.0325****	**0.1213****	**0.1374****	**0.1535****
	Taobao	HR@10	HR@50	HR@100	HR@200	NDCG@10	NDCG@50	NDCG@100	NDCG@200
Single	BPR	0.0376	0.0708	0.0871	0.1035	0.0227	0.0269	0.0305	0.0329
	ExpoMF	0.0386	0.0713	0.0911	0.1068	0.0238	0.0270	0.0302	0.0334
	NCF	0.0391	0.0728	0.0897	0.1072	0.0233	0.0281	0.0321	0.0345
	EHCF-Sin	**0.0398***	**0.0743****	**0.0936****	**0.1141****	**0.0244***	**0.0298****	**0.0339****	**0.0372****
Heterogeneous	CMF	0.0483	0.0774	0.1185	0.1563	0.0252	0.0293	0.0357	0.0379
	MC-BPR	0.0547	0.0791	0.1264	0.1597	0.0263	0.0297	0.0361	0.0397
	NMTR	0.0585	0.0942	0.1368	0.1868	0.0278	0.0334	0.0394	0.0537
	EHCF	**0.0717****	**0.1618****	**0.2211****	**0.2921****	**0.0403****	**0.0594****	**0.0690****	**0.0789****

- Performance comparison on two datasets for all methods
- Improves more than 40%
- Non-sampling performs much better on multi-behavior scenario

Efficiency Analysis



Model	*Movielens-1M*			*Beibei*			*Taobao*		
	S	I	T	S	I	T	S	I	T
NCF	91s	100	152m	62s	100	104m	115s	100	192m
EHCF-Sin	**4.5s**	100	**8m**	**3.2s**	100	**6m**	**6s**	100	**10m**
NMTR	–	–	–	165s	200	550m	180s	200	600m
EHCF-Original	–	–	–	62s	200	207m	192s	200	640m
EHCF	–	–	–	**7s**	200	**24m**	**16s**	200	**54m**

Comparison of runtime

s:second; m: minute; h: hour; d: day

S: training time for a single iteration;

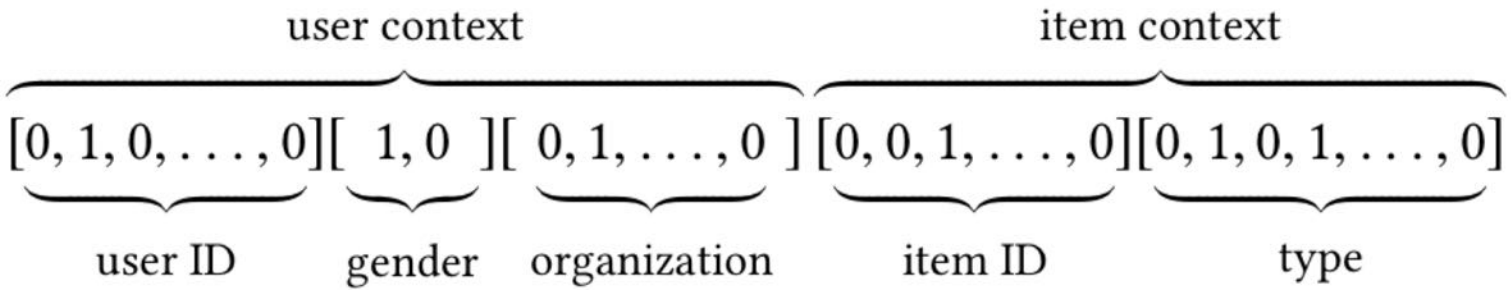
I: Overall iterations;

T: Total time

10 times faster

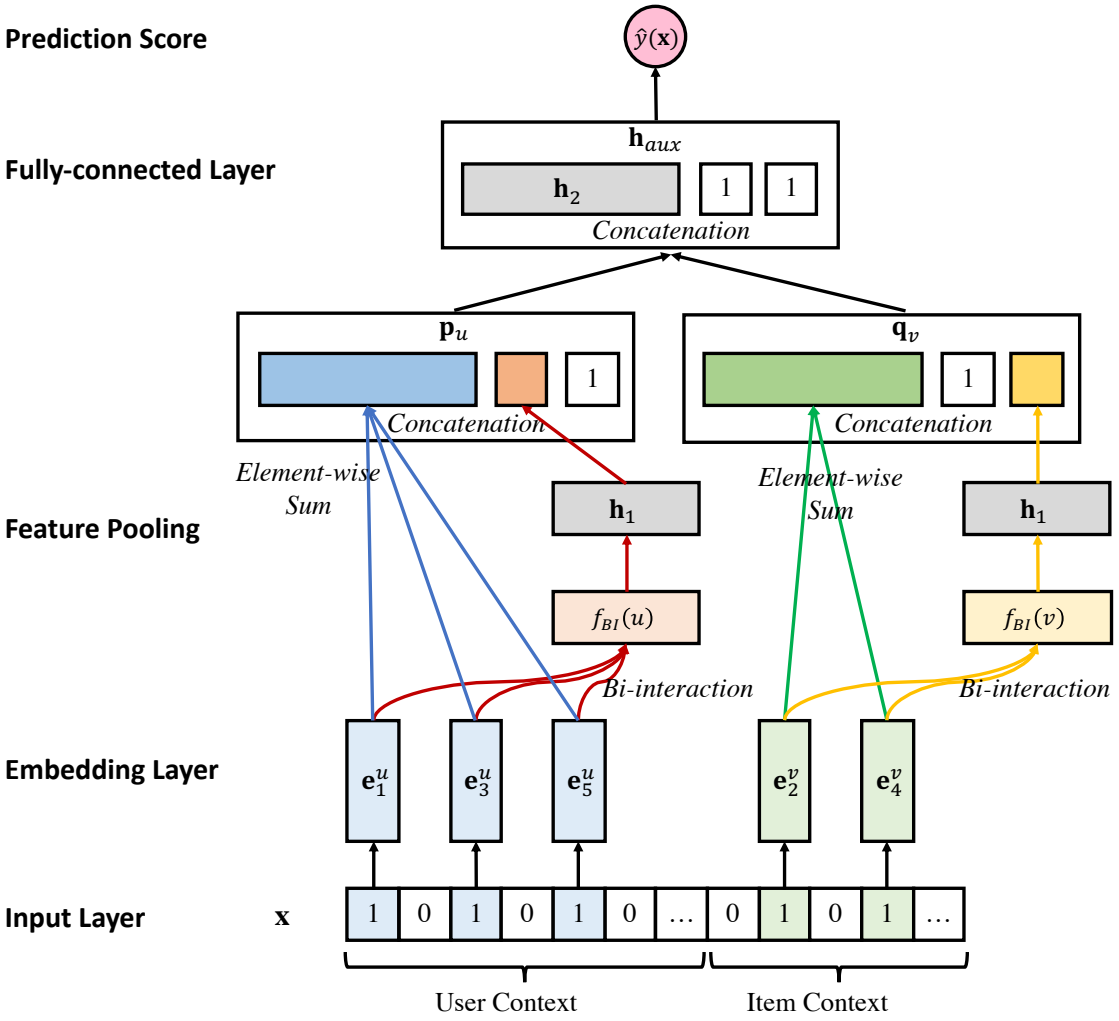
Context-aware Recommendation Scenario

- Leveraging contextual information
  - user demographics, item attributes, and time/location of the current transaction, etc.



- Factorization Machines (FM) with negative sampling is a popular solution

Context-aware Recommendation Scenario



Efficient Non-sampling Factorization Machines (ENSFM)

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{m+n} w_i x_i + \underbrace{\mathbf{h}^T \sum_{i=1}^{m+n} \sum_{j=i+1}^{m+n} (x_i \mathbf{e}_i \odot x_j \mathbf{e}_j)}_{f(\mathbf{x})} \quad (6)$$

THEOREM 4.1. *The prediction function of a generalized factorization machines (Eq.(6)) can be reformulated into a matrix factorization function:*

$$\hat{y}_{FM}(\mathbf{x}) = \mathbf{h}^T (\mathbf{p}_u \odot \mathbf{q}_v) \quad (7)$$

where $\mathbf{p}_u$ only dependents on user context u and $\mathbf{q}_v$ only dependents on item context v.

Chong Chen, Min Zhang, Weizhi Ma, Yiqun Liu and Shaoping Ma. **Efficient Non-Sampling Factorization Machines for Optimal Context-Aware Recommendation**. The Web Conference 2020 (**WWW 2020**)

Proof

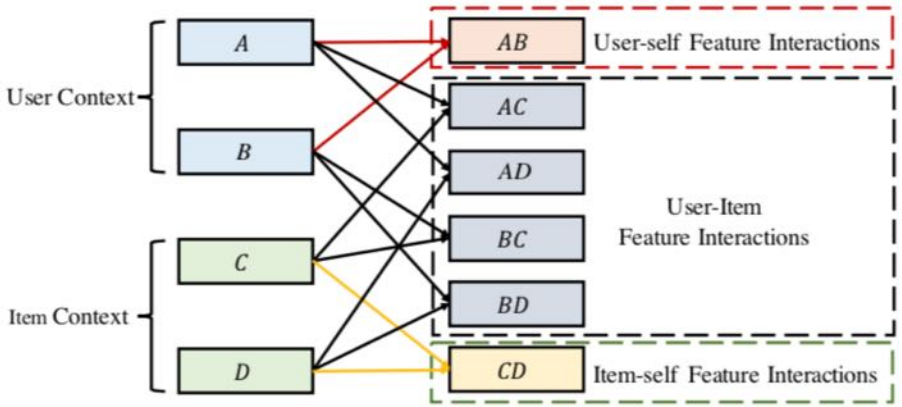
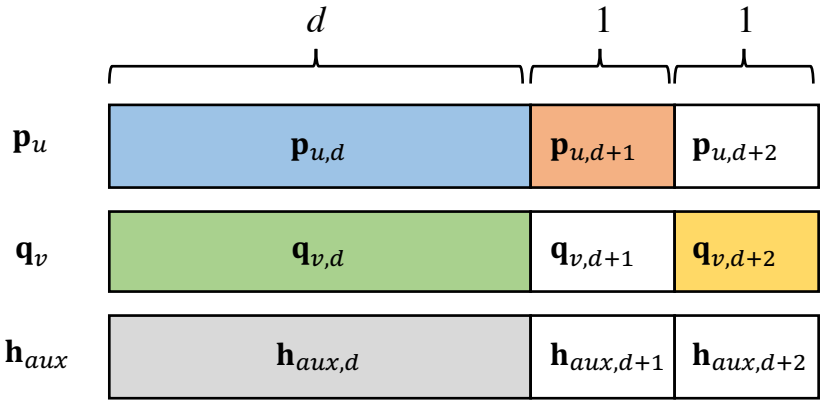


Figure 2: An example of feature interactions, which can be divided into three groups: user-self, item-self, and user-item. User-self feature interactions are independent of item features, while item-self interactions are also independent of user features.

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{m+n} w_i x_i + \underbrace{\mathbf{h}^T \sum_{i=1}^{m+n} \sum_{j=i+1}^{m+n} (x_i \mathbf{e}_i \odot x_j \mathbf{e}_j)}_{f(\mathbf{x})} \tag{6}$$

$$f(\mathbf{x}) = \mathbf{h}_1^T \left(\underbrace{\sum_{i=1}^{m} \sum_{j=i+1}^{m} (x_i^u \mathbf{e}_i^u \odot x_j^u \mathbf{e}_j^u)}_{f_{BI}(u)} + \underbrace{\sum_{i=1}^{n} \sum_{j=i+1}^{n} (x_i^v \mathbf{e}_i^v \odot x_j^v \mathbf{e}_j^v)}_{f_{BI}(v)} \right) + \mathbf{h}_2^T \left(\sum_{i=1}^{m} x_i^u \mathbf{e}_i^u \odot \sum_{i=1}^{n} x_i^v \mathbf{e}_i^v \right) \tag{8}$$

Proof



$$\mathbf{p}_u = \begin{bmatrix} \mathbf{p}_{u,d} \\ \mathbf{p}_{u,d+1} \\ \mathbf{p}_{u,d+2} \end{bmatrix}; \mathbf{q}_v = \begin{bmatrix} \mathbf{q}_{v,d} \\ \mathbf{q}_{v,d+1} \\ \mathbf{q}_{v,d+2} \end{bmatrix}; \mathbf{h}_{aux} = \begin{bmatrix} \mathbf{h}_{aux,d} \\ \mathbf{h}_{aux,d+1} \\ \mathbf{h}_{aux,d+2} \end{bmatrix} \tag{9}$$

where

$$\mathbf{p}_{u,d} = \sum_{i=1}^{m} x_i^u \mathbf{e}_i^u; \mathbf{p}_{u,d+1} = \mathbf{h}_1^T f_{BI}(u) + w_0 + \sum_{i=1}^{m} w_i^u x_i^u; \mathbf{p}_{u,d+2} = 1 \tag{10}$$

$$\mathbf{q}_{v,d} = \sum_{i=1}^{n} x_i^v \mathbf{e}_i^v; \mathbf{q}_{v,d+1} = 1; \mathbf{q}_{v,d+2} = \mathbf{h}_1^T f_{BI}(v) + \sum_{i=1}^{n} w_i^v x_i^v \tag{11}$$

$$\mathbf{h}_{aux,d} = \mathbf{h}_2; \mathbf{h}_{aux,d+1} = 1; \mathbf{h}_{aux,d+2} = 1 \tag{12}$$

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{i=1}^{m+n} w_i x_i + \underbrace{\mathbf{h}^T \sum_{i=1}^{m+n} \sum_{j=i+1}^{m+n} (x_i \mathbf{e}_i \odot x_j \mathbf{e}_j)}_{f(\mathbf{x})} \tag{6}$$

$$\Rightarrow \quad \hat{y}_{FM}(\mathbf{x}) = \mathbf{h}_{aux}^T (\mathbf{p}_u \odot \mathbf{q}_v) \tag{13}$$

https://github.com/chenchongthu/ENSFM https://github.com/THUIR/ENSFM

Efficient Mini-batch Learning Algorithm

Algorithm 1 ENSFM Learning algorithm

Require: Training data $\{\mathbf{Y}, \mathbf{U}, \mathbf{V}, \mathbf{X}\}$; weights of entries c; learning rate η; embedding size d

Ensure: Neural parameters Θ

Randomly initialize neural parameters Θ
while *Stopping criteria is not met* **do**
  Build auxiliary vectors $\mathbf{P}$ for all users (Eq.(9,10)) ▷ $O(m|\mathbf{U}|(d+2))$
  Build auxiliary vectors $\mathbf{Q}$ for all items (Eq.(9,11)) ▷ $O(n|\mathbf{V}|(d+2))$
  Build auxiliary vector $\mathbf{h}$ (Eq.(9,12)) ▷ $O(d+2)$
  while *An epoch is not end* **do**
    Randomly draw a training batch $\{\mathbf{Y_B}, \mathbf{B}, \mathbf{V}, \mathbf{X}\}$
    Compute the loss $\tilde{\mathcal{L}}(\Theta)$ (Eq.(15)) ▷ $O((|\mathbf{B}|+|\mathbf{V}|)(d+2)^2 + |\mathcal{R}_\mathbf{B}|(d+2))$
    Update model parameters
  end while
end while
return Θ

$$f_{BI}(u) = \frac{1}{2}\left((\sum_{i=1}^{m} x_i^u \mathbf{e}_i^u)^2 - \sum_{i=1}^{m}(x_i^u \mathbf{e}_i^u)^2\right)$$

$$\tilde{\mathcal{L}}(\Theta) = \sum_{u\in\mathbf{B}}\sum_{v\in\mathbf{V}^+}\left((c_v^+ - c_v^-)\hat{y}(\mathbf{x})^2 - 2c_v^+\hat{y}(\mathbf{x})\right) + \sum_{i=1}^{d}\sum_{j=1}^{d}\left((h_{aux,i}h_{aux,j})\left(\sum_{u\in\mathbf{B}} p_{u,i}p_{u,j}\right)\left(\sum_{v\in\mathbf{V}} c_v^- q_{v,i}q_{v,j}\right)\right) \quad (15)$$

Experimental settings

- Datasets:
- Baselines:
  - FM (ICDM 10)
  - DeepFM (IJCAI 17)
  - NFM (SIGIR 17)
  - ONCF (IJCAI 18)
  - CFM (IJCAI 19)
  - ENMF (SIGIR 19)
- Evaluation methods: HR@K, NDCG@K, K=5, 10, 20

Dataset	#User	#Item	#Feature	#Instance	#Field
Frappe	957	4,082	5,382	96,203	10
Last.fm	1,000	20,301	37,358	214,574	4
Movielens	6,040	3,706	10,021	1,000,209	6

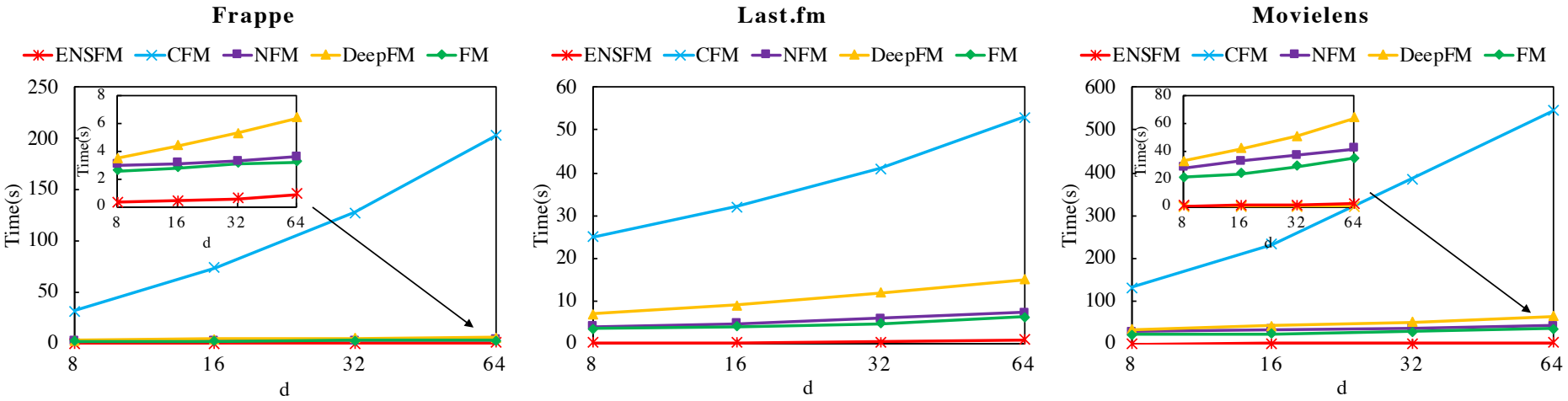
Model Comparisons

Frappe[1]	HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20	RI
PopRank	0.2539	0.3493	0.4136	0.1595	0.1898	0.2060	+143.3%
FM (*Rendle et al., 2010*)	0.4204	0.5486	0.6590	0.3054	0.3469	0.3750	+39.86%
DeepFM (*Guo et al., 2017*)	0.4632	0.6035	0.7322	0.3308	0.3765	0.4092	+27.77%
NFM (*He et al., 2017*)	0.4798	0.6197	0.7382	0.3469	0.3924	0.4225	+23.64%
ONCF (*He et al., 2018*)	0.5359	0.6531	0.7691	0.3940	0.4320	0.4614	+13.24%
CFM (*Xin et al., 2019*)	0.5462	0.6720	0.7774	0.4153	0.4560	0.4859	+9.15%
ENMF (*Chen et al., 2019*)	0.5682	0.6833	0.7749	0.4314	0.4642	0.4914	+6.95%
ENSFM	**0.6094****	**0.7118****	**0.7889****	**0.4771****	**0.5105****	**0.5301****	–
Last.fm[1]	HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20	RI
PopRank	0.0013	0.0023	0.0032	0.0007	0.0011	0.0013	+26566%
FM (*Rendle et al., 2010*)	0.1658	0.2382	0.3537	0.1142	0.1374	0.1665	+108.4%
DeepFM (*Guo et al., 2017*)	0.1773	0.2612	0.3799	0.1204	0.1473	0.1772	+94.59%
NFM (*He et al., 2017*)	0.1827	0.2678	0.3783	0.1235	0.1488	0.1765	+91.76%
ONCF (*He et al., 2018*)	0.2183	0.3208	0.4611	0.1493	0.1823	0.2176	+58.11%
CFM (*Xin et al., 2019*)	0.2375	0.3538	0.4841	0.1573	0.1948	0.2277	+48.05%
ENMF (*Chen et al., 2019*)	0.3188	0.4254	0.5279	0.2256	0.2531	0.2894	+15.94%
ENSFM	**0.3683****	**0.4729****	**0.5793****	**0.2744****	**0.3082****	**0.3352****	–
Movielens	HR@5	HR@10	HR@20	NDCG@5	NDCG@10	NDCG@20	RI
PopRank	0.0084	0.0308	0.0763	0.0041	0.0111	0.0227	+388.9%
FM (*Rendle et al., 2010*)	0.0377	0.0687	0.1164	0.0234	0.0334	0.0453	+52.32%
DeepFM (*Guo et al., 2017*)	0.0413	0.0754	0.1351	0.0247	0.0365	0.0503	+38.43%
NFM (*He et al., 2017*)	0.0421	0.0775	0.1334	0.0268	0.0381	0.0521	+33.91%
ONCF (*He et al., 2018*)	0.0491	0.0801	0.1368	0.0301	0.0402	0.0543	+24.70%
CFM (*Xin et al., 2019*)	0.0514	0.0812	0.1398	0.0318	0.0419	0.0567	+20.22%
ENMF (*Chen et al., 2019*)	0.0534	0.0867	0.1523	0.0332	0.0448	0.0606	+13.10%
ENSFM	**0.0601****	**0.1024****	**0.1690****	**0.0373****	**0.0508****	**0.0674****	–

[1] For Frappe and Last.fm datasets, the results of PopRank, FM, DeepFM, NFM, ONCF, and CFM are the same as those reported in [42] since we share exactly the same data splits.

- Performance comparison on three datasets for all methods
- Best Baselines:
  - ENMF: whole-data
  - CFM: sampled data
- ENSFM
  - Consistently significantly outperforms the best baseline

Efficiency Analysis



Comparison of runtim

s:second; m: minute; h: hour; d: day

S: training time for a single iteration;

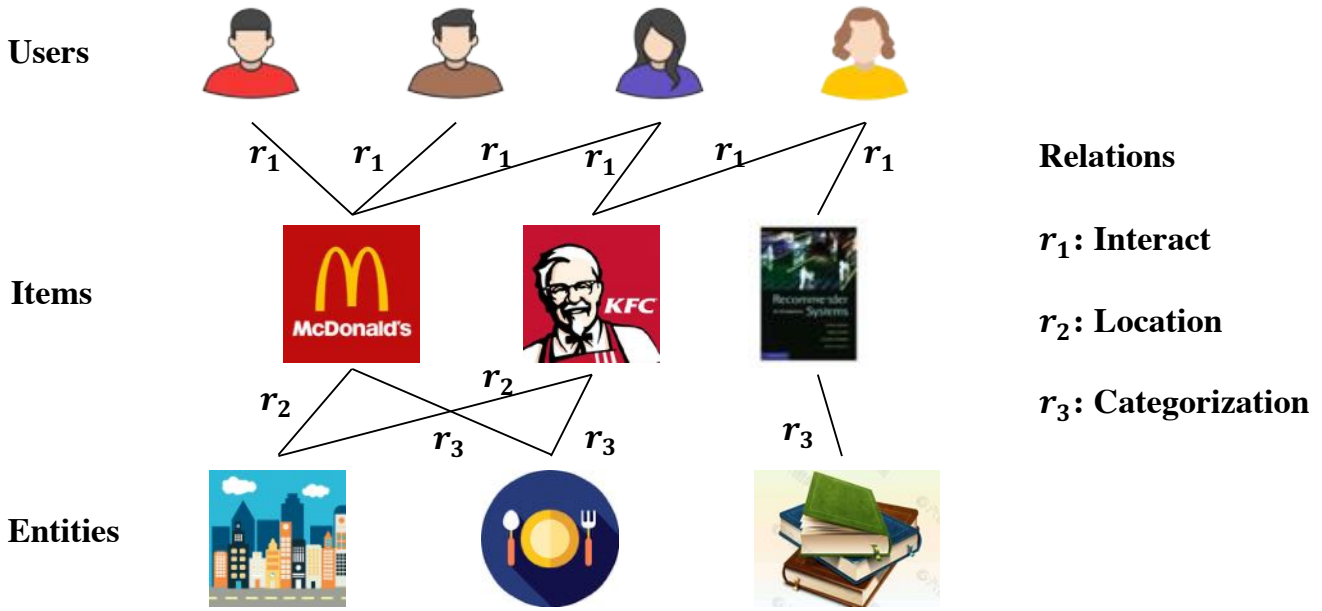
I: Overall iterations;

T: Total time

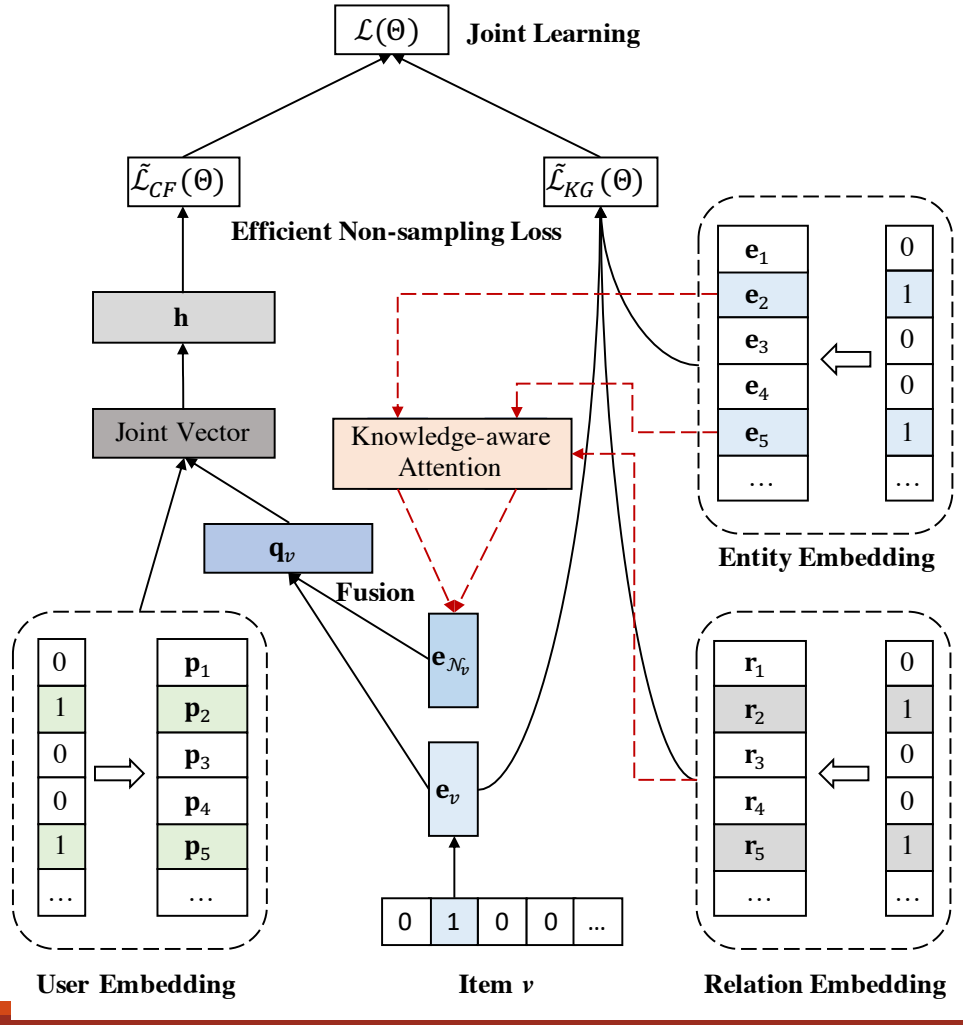
Model	*Frappe*			*Last.fm*			*Movielens*		
	S	I	T	S	I	T	S	I	T
FM	3.2s	500	27m	6.2s	500	52m	35s	500	5h
NFM	3.6s	500	30m	7.3s	500	61m	42s	500	6h
DeepFM	6.4s	500	54m	15s	500	324m	64s	500	9h
CFM	203s	500	28h	54s	500	125m	9m	500	3d
ENSFM	**0.9s**	200	**3m**	**1.1s**	500	**10m**	**2s**	200	**7m**

Knowledge Graph Enhanced Recommendation

- Knowledge graph enhanced recommendation
  - The relation type and information value are both considered to construct the recommendation model.
  - Item acts as the bridge to link user-item interactions and knowledge graph.



Jointly Non-Sampling Learning for Knowledge Graph Enhanced Recommendation （JNSKR）



- The structure of JNSKR consists of three main components:
  - KG embedding part to learn structural KG information through the proposed efficient non-sampling method
  - Attentive user-item preference modeling part, which infers the user-item preference score with an attention mechanism
  - Joint learning part that integrates the above two parts in an end-to-end fashion.

https://github.com/chenchongthu/JNSKR

Non-sampling Knowledge Graph Embedding

- Knowledge graph embedding is an effective way to convert entities and relations as vector representations
- We propose to apply non-sampling strategy for knowledge graph embedding learning.
- For a commonly-used non-sampling loss:

$$\mathcal{L}_{KG}(\Theta) = \sum_{h \in \mathbf{B}} \sum_{t \in \mathbf{E}} \sum_{r \in \mathbf{R}} w_{hrt}(g_{hrt} - \hat{g}_{hrt})^2$$

Complexity: $O(|\mathbf{B}||\mathbf{E}||\mathbf{R}|d)$

Efficient Non-sampling Knowledge Graph Embedding

$$\mathcal{L}_{KG}(\Theta) = \sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}}\sum_{r\in\mathbf{R}} w_{hrt}(g_{hrt}^2 - 2g_{hrt}\hat{g}_{hrt} + \hat{g}_{hrt}^2)$$

g_{hrt} : (0,1)

g_{hrt} =0
for neg. feedbacks,

$$\tilde{\mathcal{L}}_{KG}(\Theta) = -2\sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}^+}\sum_{r\in\mathbf{R}^+} w_{hrt}^+\hat{g}_{hrt} + \sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}}\sum_{r\in\mathbf{R}} w_{hrt}\hat{g}_{hrt}^2$$

$$= \overbrace{\sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}^+}\sum_{r\in\mathbf{R}^+}\left((w_{hrt}^+ - w_{hrt}^-)\hat{g}_{hrt}^2 - 2w_{hrt}^+\hat{g}_{hrt}\right)}^{\mathcal{L}_{KG}^P(\Theta)}$$

$$+ \overbrace{\sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}}\sum_{r\in\mathbf{R}} w_{hrt}^-\hat{g}_{hrt}^2}^{\mathcal{L}_{KG}^A(\Theta)}$$

Bottleneck

Efficient Non-sampling Knowledge Graph Embedding

$$\tilde{\mathcal{L}}_{KG}(\Theta) = -2\sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}^+}\sum_{r\in\mathbf{R}^+} w^+_{hrt}\hat{g}_{hrt} + \sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}}\sum_{r\in\mathbf{R}} w_{hrt}\hat{g}^2_{hrt}$$

$$= \overbrace{\sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}^+}\sum_{r\in\mathbf{R}^+}\left((w^+_{hrt}-w^-_{hrt})\hat{g}^2_{hrt} - 2w^+_{hrt}\hat{g}_{hrt}\right)}^{\mathcal{L}^P_{KG}(\Theta)}$$

$$+ \overbrace{\sum_{h\in\mathbf{B}}\sum_{t\in\mathbf{E}}\sum_{r\in\mathbf{R}} w^-_{hrt}\hat{g}^2_{hrt}}^{\mathcal{L}^A_{KG}(\Theta)}$$

$$\sum_{i=1}^{d}\sum_{j=1}^{d}\left(\left(\sum_{r\in\mathbf{R}} r_i r_j\right)\left(\sum_{h\in\mathbf{B}} w^-_h e_{h,i}e_{h,j}\right)\left(\sum_{t\in\mathbf{E}} e_{t,i}e_{t,j}\right)\right)$$

$O(|\mathbf{B}||\mathbf{E}||\mathbf{R}|d) \longrightarrow O((|\mathbf{B}|+|\mathbf{E}|+|\mathbf{R}|)d^2)$

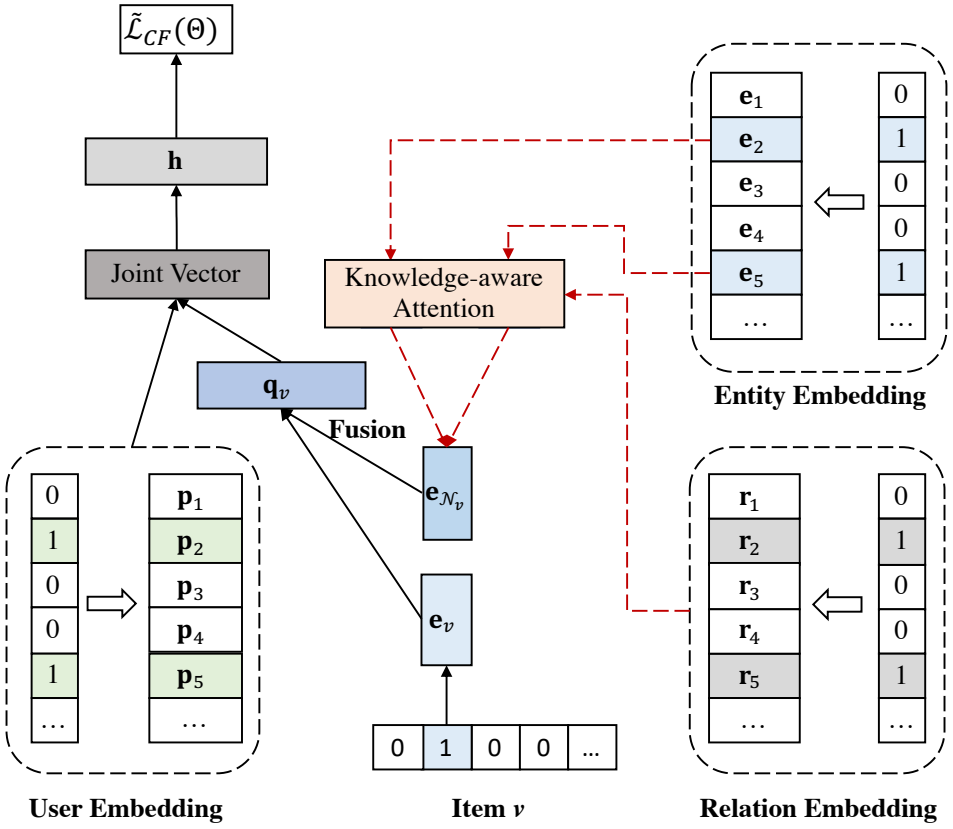
To address the inefficiency issue, g_{hrt} need to be a score function that can be properly expanded, **DistMult** is adopted in our paper:

$$\hat{g}_{hrt} = \mathbf{e}_h^T \cdot diag(\mathbf{r}) \cdot \mathbf{e}_t = \sum_i^d e_{h,i}r_i e_{t,i}$$

$$\hat{g}^2_{hrt} = \sum_i^d e_{h,i}r_i e_{t,i}\sum_j^d e_{h,j}r_j e_{t,j}$$

$$= \sum_i^d\sum_j^d (e_{h,i}e_{h,j})(r_i,r_j)(e_{t,i}e_{t,j})$$

User-Item Preference Modeling



Prediction:

$$\hat{y}_{uv} = \mathbf{h}^T \left(\mathbf{p}_u \odot \mathbf{q}_v\right)$$

Combination of $\mathbf{q}_v$:

$$\mathbf{q}_v = \mathbf{e}_v + \mathbf{e}_{\mathcal{N}_v}$$

$$= \mathbf{e}_v + \sum_{(v,r,t)\in\mathcal{N}_v} \alpha_{(r,t)}\mathbf{e}_t$$

Knowledge-aware attention:

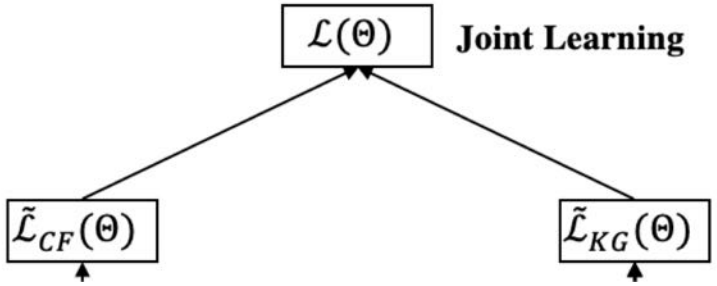
$$\alpha^*_{(r,t)} = \mathbf{h}^T_\alpha \sigma(\mathbf{W}_1\mathbf{e}_t + \mathbf{W}_2\mathbf{r} + \mathbf{b})$$

$$\alpha_{(r,t)} = \frac{\exp(\alpha^*_{(r,t)})}{\sum_{(v,r',t')\in\mathcal{N}_v} \exp(\alpha^*_{(r',t')})}$$

Joint Learning

Recommendation:

$$\tilde{\mathcal{L}}_{CF}(\Theta) = \sum_{u \in \mathbf{U}^+} \sum_{v \in \mathbf{B}} \left((c_v^+ - c_v^-)\hat{y}_{uv}^2 - 2c_v^+ \hat{y}_{uv} \right) + \sum_{i=1}^{d} \sum_{j=1}^{d} \left((h_i h_j) \left(\sum_{u \in \mathbf{U}} p_{u,i} p_{u,j} \right) \left(\sum_{v \in \mathbf{B}} c_v^- q_{v,i} q_{v,j} \right) \right)$$



Knowledge Graph Embedding:

$$\tilde{\mathcal{L}}_{KG}(\Theta) = \sum_{h \in \mathbf{B}} \sum_{t \in \mathbf{E}^+} \sum_{r \in \mathbf{R}^+} \left((w_{hrt}^+ - w_{hrt}^-)\hat{g}_{hrt}^2 - 2w_{hrt}^+ \hat{g}_{hrt} \right) + \sum_{i=1}^{d} \sum_{j=1}^{d} \left(\left(\sum_{r \in \mathbf{R}} r_i r_j \right) \left(\sum_{h \in \mathbf{B}} w_h^- e_{h,i} e_{h,j} \right) \left(\sum_{t \in \mathbf{E}} e_{t,i} e_{t,j} \right) \right)$$

Joint Learning:

$$\mathcal{L}(\Theta) = \tilde{\mathcal{L}}_{CF}(\Theta) + \mu \tilde{\mathcal{L}}_{KG}(\Theta) + \lambda \|\Theta\|_2^2$$

Experimental settings

- Datasets:
- Baselines:
  - NCF (WWW 17)
  - ENMF (SIGIR 19)
  - NFM (SIGIR 17)
  - CKE (KDD 16)
  - CFKG (Algorithms 18)
  - RippleNet (CIKM 18)
  - KGAT (KDD 19)
- Evaluation methods: Recall@K, NDCG@K, K=10, 20, 40

		Amazon-book	*Yelp2018*
User-Item Interaction	**#Users**	70, 679	45, 919
	#Items	24, 915	45, 538
	#Interactions	847, 733	1, 185, 068
Knowledge Graph	**#Entities**	88, 572	90, 961
	#Relations	39	42
	#Triplets	2, 557, 746	1, 853, 704

Exactly the same data splits for objective comparison

Model Comparisons

Models	*Amazon-book*						
	Recall@10	Recall@20	Recall@40	NDCG@10	NDCG@20	NDCG@40	RI
NCF	0.0874	0.1319	0.1924	0.0724	0.0895	0.1111	+17.03%
ENMF	0.1002	0.1472	0.2085	0.0797	0.0998	0.1215	+5.49%
NFM	0.0891	0.1366	0.1975	0.0723	0.0913	0.1152	+14.44%
CKE	0.0875	0.1343	0.1946	0.0705	0.0885	0.1114	+17.14%
CFKG	0.0769	0.1142	0.1901	0.0603	0.077	0.0985	+32.62%
RippleNet	0.0883	0.1336	0.2008	0.0747	0.0910	0.1164	+13.99%
KGAT	0.1017	0.1489	0.2094	0.0814	0.1006	0.1225	+4.31%
JNSKR	**0.1056****	**0.1558****	**0.2178****	**0.0842****	**0.1068****	**0.1271****	–
Models	***Yelp2018***						
	Recall@10	Recall@20	Recall@40	NDCG@10	NDCG@20	NDCG@40	RI
NCF	0.0389	0.0653	0.1060	0.0603	0.0802	0.1087	+14.28%
ENMF	0.0403	0.0711	0.1109	0.0611	0.0877	0.1097	+9.15%
NFM	0.0396	0.0660	0.1082	0.0603	0.0810	0.1094	+13.03%
CKE	0.0399	0.0657	0.1074	0.0608	0.0805	0.1091	+13.13%
CFKG	0.0288	0.0522	0.0904	0.0450	0.0644	0.0897	+44.27%
RippleNet	0.0402	0.0664	0.1088	0.0613	0.0822	0.1097	+11.90%
KGAT	0.0418	0.0712	0.1128	0.0630	0.0867	0.1129	+7.26%
JNSKR	**0.0456****	**0.0749****	**0.1209****	**0.0687****	**0.0917****	**0.1211****	–

The results of KGAT are the same as those reported in [38] since we share exactly the same data splits and experimental settings.

- Performance comparison on three datasets for all methods
- Best Baselines:
  - ENMF: without knowledge
  - KGAT: with knowledge
- JNSKR
  - Consistently and significantly outperforms the best baseline

Efficiency Analysis

Comparison of runtime

s:second; m: minute; h: hour;
S: training time for a single iteration;
I: Overall iterations;
T: Total time

Model	*Amazon-book*			*Yelp2018*		
	S	I	T	S	I	T
CKE	66s	200	220m	75s	200	250m
CFKG	27s	200	90m	45s	200	150m
RippleNet	15m	200	50h	11m	200	37h
KGAT	9m	300	45h	7m	300	35h
JNSKR	**14s**	200	**47m**	**16s**	200	**54m**

20+ times faster than the best baseline!

Discussion

- Recently, there is a surge of interest in applying novel neural networks for recommendation tasks.
- More complex models do not necessarily lead to better results since they are more difficult to optimize and tune
- We empirically shows that a proper learning method is even more important than advanced neural network structures
- We expect future research should focus more on designing models with better learning algorithms for specific tasks, rather than relying on complex models and expensive computational power for minor improvements

Thank You!

z-m@tsinghua.edu.cn

cc17@mails.tsinghua.edu.cn