Knowledge Graph Aware Recommender System Survey

曾缘

Mar. 19th,2021

Recommender System

Collaborative Filtering	Content-based Filtering	Hybrid Method
基于用户可能对有相似的交互记录的人所选择的item感兴趣的假设。 从全局的user-item的交互数据中，学习user和item的表示。	基于用户可能会对过去有过交互记录的item的相似item感兴趣的假设。 从辅助信息的内容中，提取item的表示；从个人交互的项目特征中提取user的表示。	通过将user和item的内容信息（user side information和item side information）合并到基于CF的框架中，以获得更好的推荐性能。

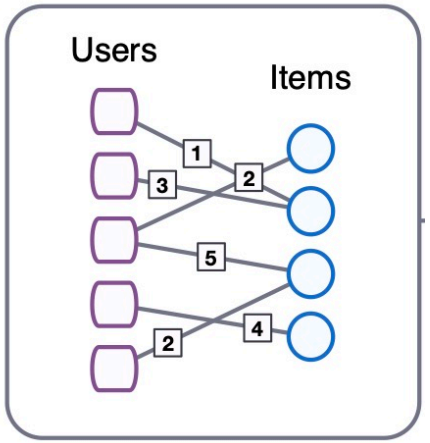
KG-Based Recommender System

- Embedding-based methods
- Path-based methods
- Unified methods

Embedding-Based Methods

- 基于嵌入的方法通常直接使用KG中的信息来丰富物品或用户的表示。
- 为了充分利用知识图的信息，需要采用知识图嵌入（KGE）算法将知识图谱编码为低秩嵌入。通常使用的KGE算法如TransE、TransR、DistMult等。
- User-item graph

  users, items, and their related attributes function as nodes.



Bipartite graph

Embedding-Based Methods

2018：Learning over Knowledge-Base Embeddings for Recommendation (CFKG)

提出一种知识表示学习方法来嵌入异质实体来进行推荐。

Problem formulation: 给定 user-item knowledge graph（包含用户和物品以及各个实体间的关系）

任务：为每个用户生成一个Top-N的推荐列表

Model structure: 首先，定义了一个专门用于推荐系统的user-item knowledge graph结构，然后在此图的基础上进行协同过滤以提供个性化的推荐。

User-item knowledge graph :

定义了5种实体类型、6种关系类型

user, item, word, brand, category

buy 用户购买了该物品

belong_to_category 物品属于该类别

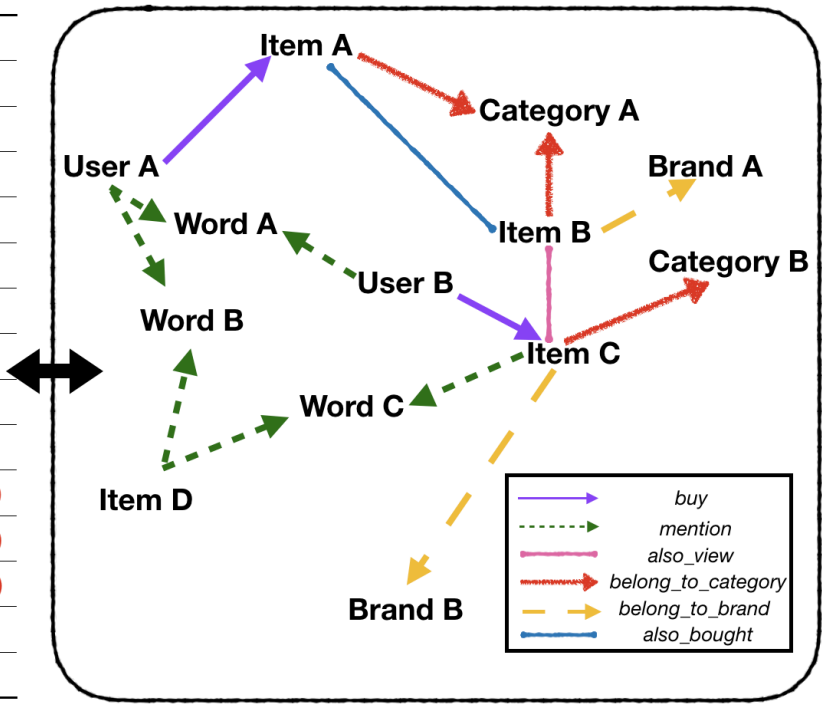
belong_to_brand 物品属于该品牌

mention_word 用户或物品提到过的单词

also_bought 用户购买了第一个 也购买了第二个物品

also_view 用户购买了第一个 浏览了第二个物品

Behavior triplets
(User A, Item A, buy)
(User B, Item C, buy)
(User A, Word A, mention_word)
(User A, Word B, mention_word)
(Item D, Word B, mention_word)
(User B, Word A, mention_word)
(Item D, Word C, mention_word)
(Item C, Word C, mention_word)
(Item A, Item B, also_bought)
(Item B, Item C, also_view)
(Item A, Category A, belong_to_category)
(Item B, Category A, belong_to_category)
(Item C, Category B, belong_to_category)
(Item B, Brand A, belong_to_brand)
(Item C, Brand B, belong_to_brand)



User behavior Graph

Collaborative Filtering based on User-Item Knowledge Graph :

$$L = \sum_{(i,j,r)\in S} \left\{ \sum_{(i,j',r)\in S^t} \left[\gamma + d\left(trans_{e_r}(e_i), e_j\right) - d\left(trans_{e_r}(e_i), e_{j'}\right)\right]_+ + \sum_{(i',j,r)\in S^h} \left[\gamma + d\left(trans_{e_r}(e_i), e_j\right) - d\left(trans_{e_r}(e_{i'}), e_j\right)\right]_+ \right\}$$

Personalized Recommendation :

获得图中所有实体和关系的嵌入，假设关系buy的嵌入是e_{buy}，而目标用户的嵌入是e_u，那么我们可以通过计算距离:

$$d(trans_{e_{buy}}(e_i), e_j).$$

按距离的升序排序为用户生成推荐。

Performance on top-10 recommendation :

Dataset	CDs				Clothing				Cell Phones				Beauty			
Measures(%)	NDCG	Recall	HT	Prec	NDCG	Recall	HT	Prec	NDCG	Recall	HT	Prec	NDCG	Recall	HT	Prec
BPR	2.009	2.679	8.554	1.085	0.601	1.046	1.767	0.185	1.998	3.258	5.273	0.595	2.753	4.241	8.241	1.143
BPR_HFT	2.661	3.570	9.926	1.268	1.067	1.819	2.872	0.297	3.151	5.307	8.125	0.860	2.934	4.459	8.268	1.132
VBPR	0.631	0.845	2.930	0.328	0.560	0.968	1.557	0.166	1.797	3.489	5.002	0.507	1.901	2.786	5.961	0.902
DeepCoNN	4.218	6.001	13.857	1.681	1.310	2.332	3.286	0.229	3.636	6.353	9.913	0.999	3.359	5.429	9.807	1.200
CKE	4.620	6.483	14.541	1.779	1.502	2.509	4.275	0.388	3.995	7.005	10.809	1.070	3.717	5.938	11.043	1.371
JRL	5.378*	7.545*	16.774*	2.085*	1.735*	2.989*	4.634*	0.442*	4.364*	7.510*	10.940*	1.096*	4.396*	6.949*	12.776*	1.546*
CFKG	**5.563**	**7.949**	**17.556**	**2.192**	**3.091**	**5.466**	**7.972**	**0.763**	**5.370**	**9.498**	**13.455**	**1.325**	**6.370**	**10.341**	**17.131**	**1.959**
Improvement	3.44	5.35	4.66	5.13	78.16	82.87	72.03	72.62	23.05	26.47	22.99	20.89	44.90	48.81	34.09	26.71

Evaluation Metrics :

Precision 准确率：提取出的正确信息条数 / 提取出的信息条数 （查准）

Recall 召回率：提取出的正确信息条数 / 样本中的信息条数 （查全）

Embedding-Based Methods

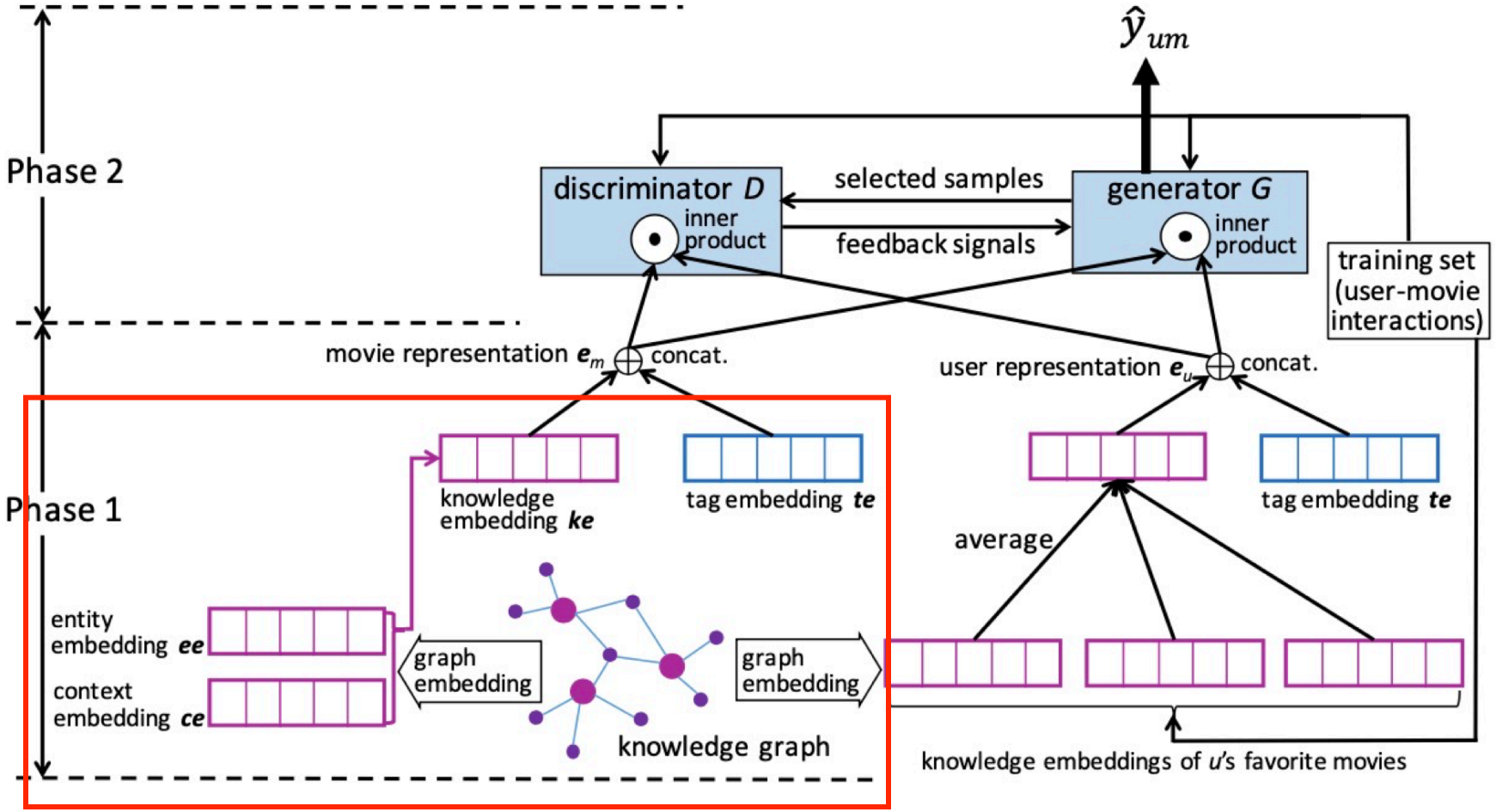
2018：A Knowledge-Enhanced Deep Recommendation Framework Incorporating GAN-Based Models

将KGE算法与GAN框架结合，来提升user和item的表示效果。

Problem formulation : 给定 knowledge graph（包含用户和电影信息）

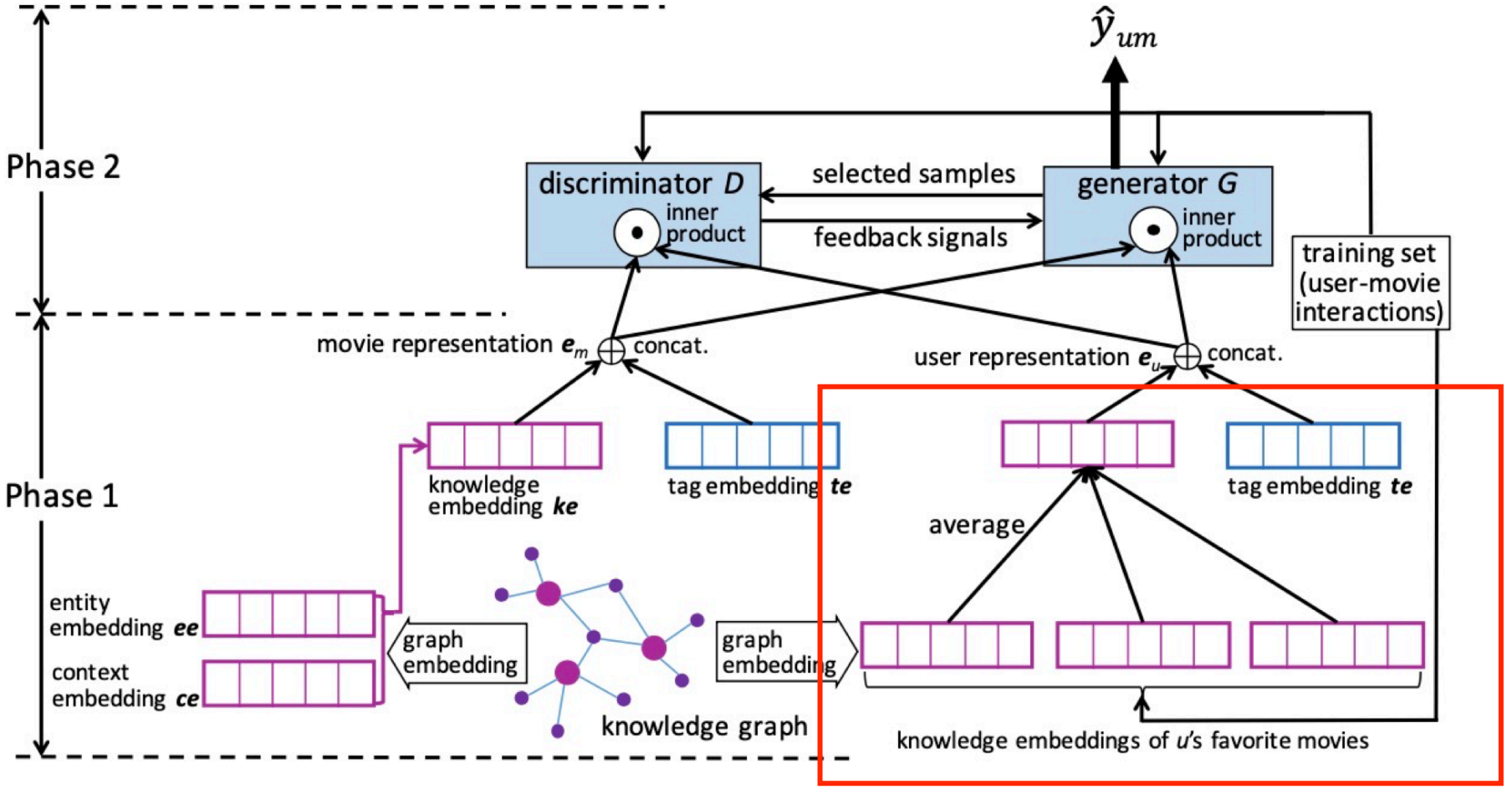
任务：为用户生成一个Top-N的电影推荐列表

The framework :



阶段1：引入KG的知识，生成movie和user的特征嵌入。

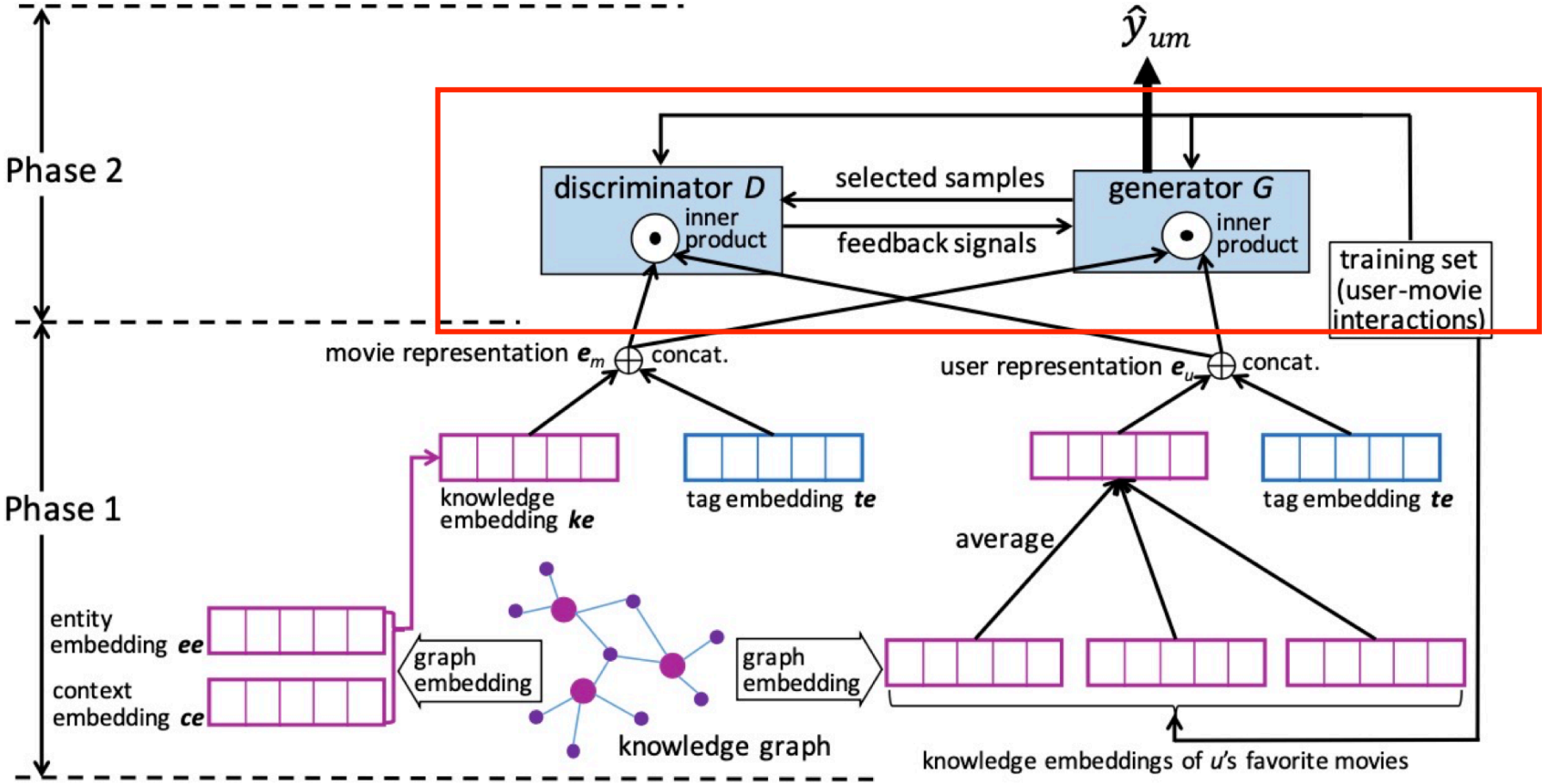
The framework :



阶段1：引入KG的知识，生成movie和user的特征嵌入。

The framework :

利用生成器或鉴别器的输出，实现top-N电影推荐



阶段2：构造生成器G和鉴别器D，来学习用户的最佳表示。

Performance :

TOP-N MOVIE RECOMMENDATION RESULTS WHEN $r = 0.1\%$.

Model	Precision			Average Precision			nDCG		
	@3	@5	@8	@3	@5	@8	@3	@5	@8
COS	0.568	0.535	0.521	0.478	0.409	0.354	0.581	0.562	0.529
NFM	0.560	0.532	0.511	0.476	0.408	0.355	0.576	0.551	0.533
NCF	0.556	0.530	0.515	0.476	0.407	0.358	0.579	0.538	0.555
IRGAN	0.556	0.525	0.510	0.476	0.404	0.354	0.577	0.550	0.533
KGAN	0.723	0.678	0.628	0.661	0.583	0.506	0.738	0.703	0.663
KTGAN	**0.759**	**0.719**	**0.657**	**0.701**	**0.631**	**0.545**	**0.771**	**0.741**	**0.693**

Embedding-Based Methods

2019：Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation

用知识图嵌入任务来辅助推荐任务，提出了一种多任务特征学习的推荐方法。

Problem formulation :

给定 一组用户U, 一组物品V, user-item交互矩阵 Y和知识图谱 G

（其中Y是一个隐反馈矩阵，当$y_{uv} = 1$时代表user u和item v有过交互，否则$y_{uv} = 0$。）

任务：预测用户对某个之前没有交互的item v是否有潜在兴趣

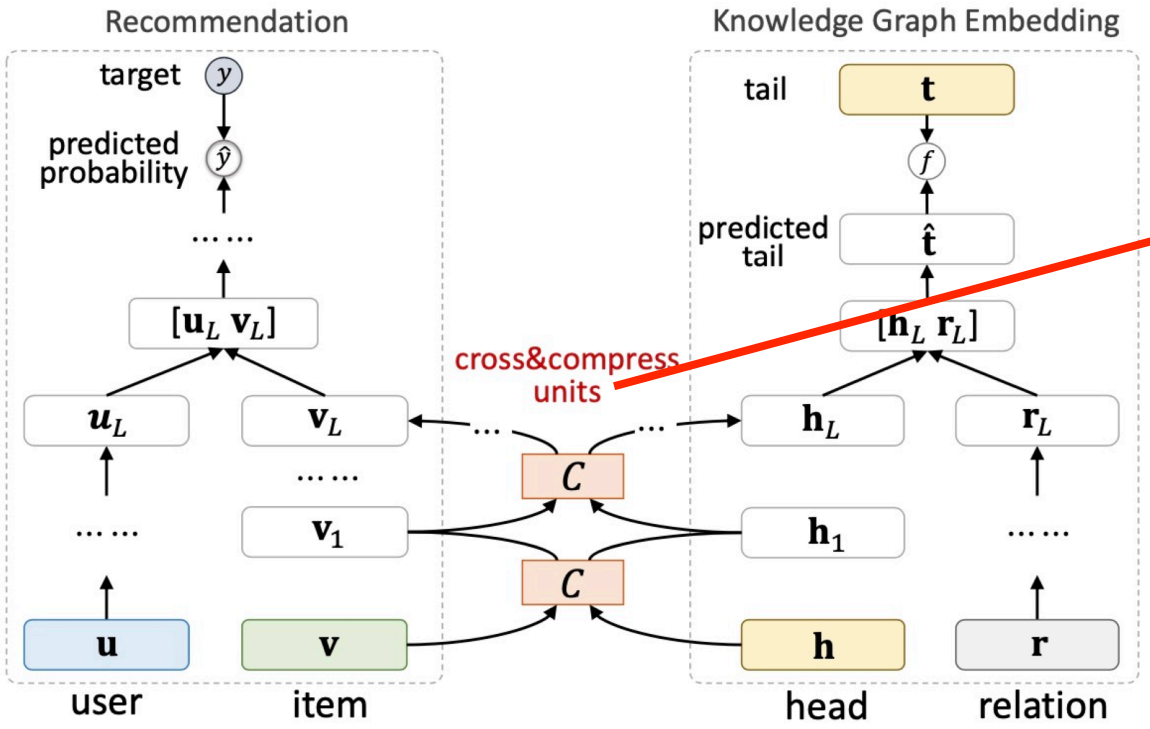
The framework :

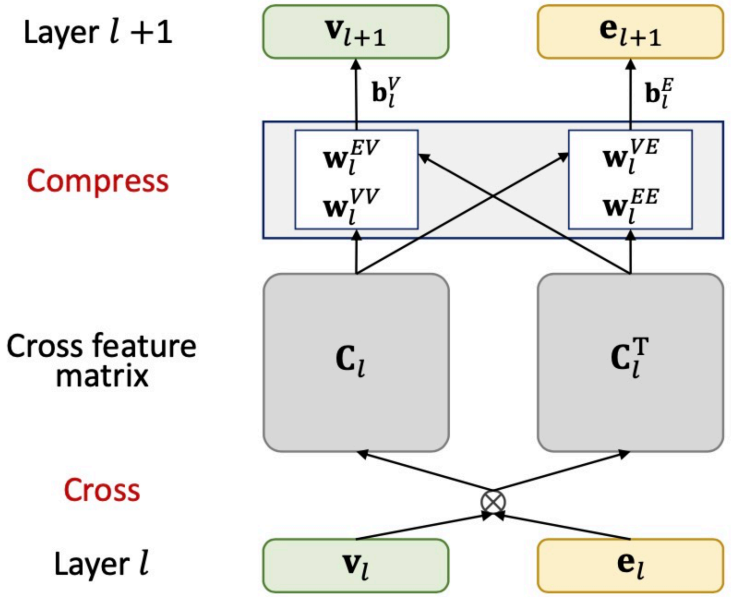
Recommendation module

KGE module

Cross&Compress units

Cross&Compress units :



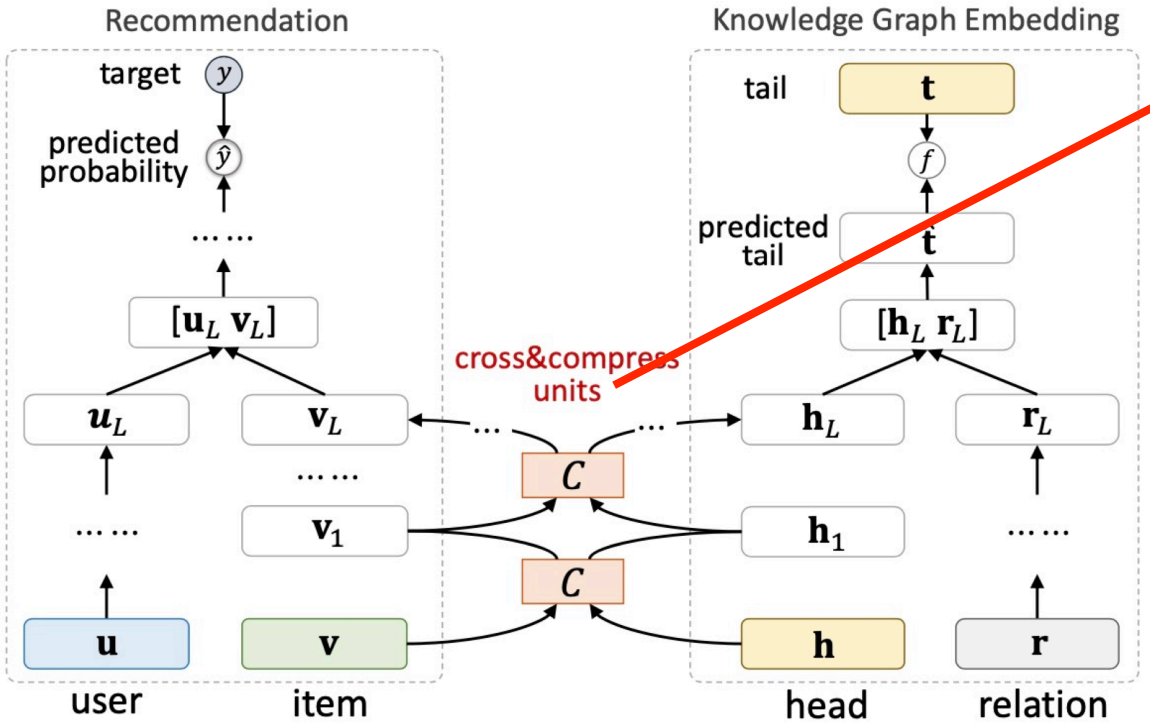


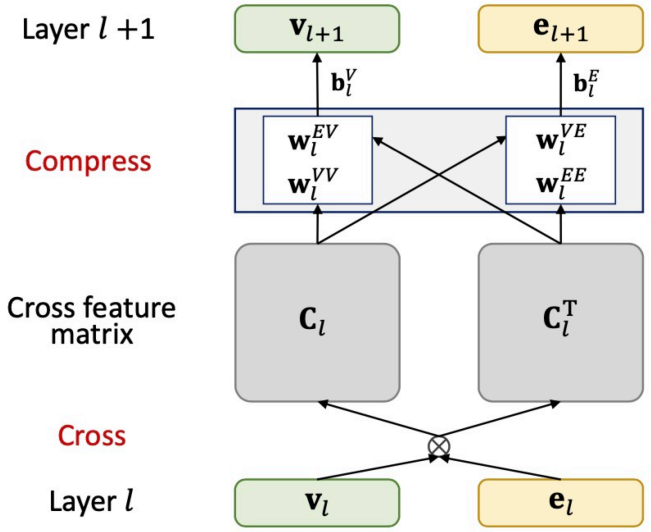
交叉操作：

在第l层，用item v和一个与它相关的实体e的特征向量$\mathbf{v}_l$和e_l，构建一个交叉特征矩阵：

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

The framework :





压缩操作：

通过将交叉特征矩阵投影到他们的特征表示空间中，输出$l+1$层的item v和entity e的特征向量：

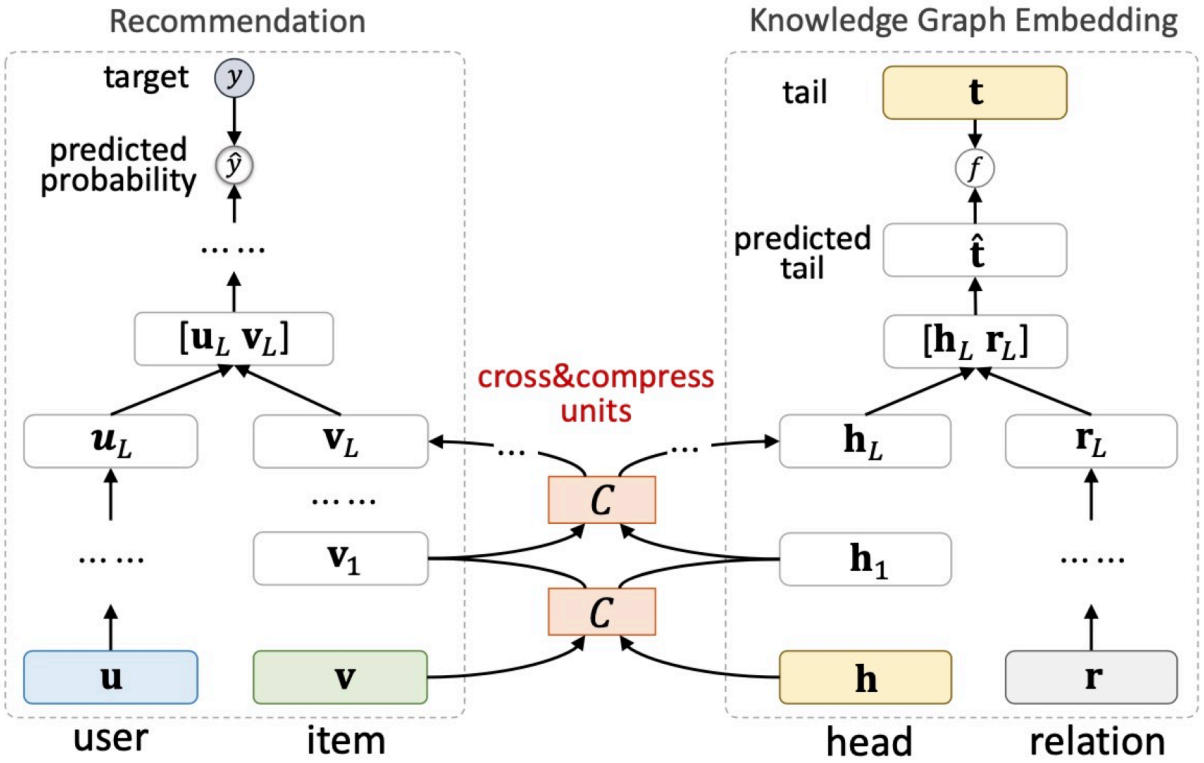
$$\mathbf{v}_{l+1} = \mathbf{C}_l \mathbf{w}_l^{VV} + \mathbf{C}_l^\top \mathbf{w}_l^{EV} + \mathbf{b}_l^V = \mathbf{v}_l \mathbf{e}_l^\top \mathbf{w}_l^{VV} + \mathbf{e}_l \mathbf{v}_l^\top \mathbf{w}_l^{EV} + \mathbf{b}_l^V,$$

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

The framework :

左侧是推荐模块。以user和item作为输入，采用MLP（多层感知机）和交叉压缩单元，为user和item分别提取特征。
最后将提取的特征一起送入另一个MLP，输出预测概率。

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



右侧是知识图谱嵌入模块。对给定的知识三元组(h,r,t)，我们首先利用多个交叉压缩单元和非线性层来处理头部h和关系r。然后将得到的特征向量拼接起来，输入到用于预测尾部t的k层MLP，得到$\hat{t}$，最后输出预测分数：

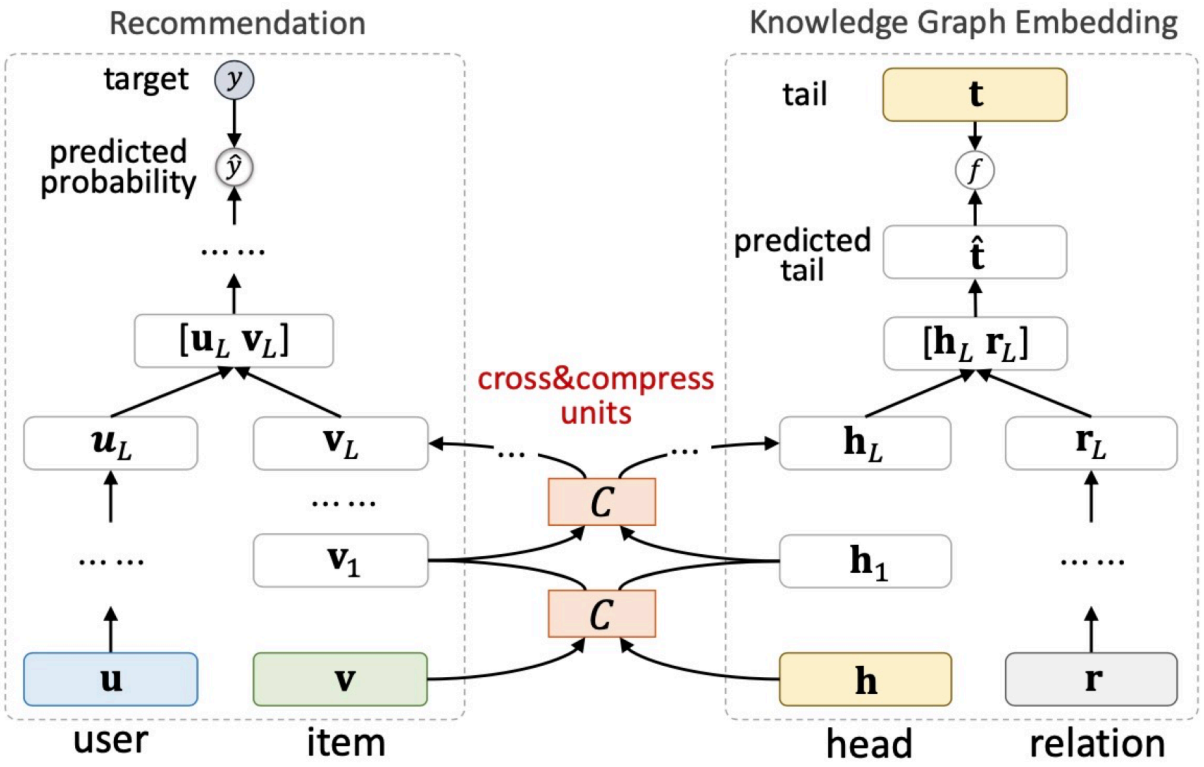
$$score(h, r, t) = f_{KG}(\mathbf{t}, \hat{\mathbf{t}}),$$

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

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

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

The framework :



The complete loss function of MKR is as follows:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_{RS} + \mathcal{L}_{KG} + \mathcal{L}_{REG} \\ &= \sum_{u \in \mathcal{U}, v \in \mathcal{V}} \mathcal{J}(\hat{y}_{uv}, y_{uv}) \\ &\quad - \lambda_1 \Big(\sum_{(h,r,t) \in \mathcal{G}} score(h,r,t) - \sum_{(h',r,t') \notin \mathcal{G}} score(h',r,t') \Big) \\ &\quad + \lambda_2 \|\mathbf{W}\|_2^2.\end{aligned}$$

Performance:

Model	MovieLens-1M		Book-Crossing		Last.FM		Bing-News	
	AUC	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*	*AUC*	*ACC*
PER	0.710 (-22.6%)	0.664 (-21.2%)	0.623 (-15.1%)	0.588 (-16.7%)	0.633 (-20.6%)	0.596 (-20.7%)	-	-
CKE	0.801 (-12.6%)	0.742 (-12.0%)	0.671 (-8.6%)	0.633 (-10.3%)	0.744 (-6.6%)	0.673 (-10.5%)	0.553 (-19.7%)	0.516 (-20.0%)
DKN	0.655 (-28.6%)	0.589 (-30.1%)	0.622 (-15.3%)	0.598 (-15.3%)	0.602 (-24.5%)	0.581 (-22.7%)	0.667 (-3.2%)	0.610 (-5.4%)
RippleNet	**0.920** (+0.3%)	0.842 (-0.1%)	0.729 (-0.7%)	0.662 (-6.2%)	0.768 (-3.6%)	0.691 (-8.1%)	0.678 (-1.6%)	0.630 (-2.3%)
LibFM	0.892 (-2.7%)	0.812 (-3.7%)	0.685 (-6.7%)	0.640 (-9.3%)	0.777 (-2.5%)	0.709 (-5.7%)	0.640 (-7.1%)	0.591 (-8.4%)
Wide&Deep	0.898 (-2.1%)	0.820 (-2.7%)	0.712 (-3.0%)	0.624 (-11.6%)	0.756 (-5.1%)	0.688 (-8.5%)	0.651 (-5.5%)	0.597 (-7.4%)
MKR	0.917	**0.843**	**0.734**	**0.704**	**0.797**	**0.752**	**0.689**	**0.645**
MKR-1L	-	-	-	-	0.795 (-0.3%)	0.749 (-0.4%)	0.680 (-1.3%)	0.631 (-2.2%)
MKR-DCN	0.883 (-3.7%)	0.802 (-4.9%)	0.705 (-4.3%)	0.676 (-4.2%)	0.778 (-2.4%)	0.730 (-2.9%)	0.671 (-2.6%)	0.614 (-4.8%)
MKR-stitch	0.905 (-1.3%)	0.830 (-1.5%)	0.721 (-2.2%)	0.682 (-3.4%)	0.772 (-3.1%)	0.725 (-3.6%)	0.674 (-2.2%)	0.621 (-3.7%)

Evaluation Metrics :

AUC 如果随机挑选一个正样本和一个负样本，分类算法将这个正样本排在负样本前面的概率。（ROC曲线下面积）

ACC 被正确分类的正例和负例，占所有参与分类样本的比例。

Path-Based Methods

- 基于路径的方法通常使用user-item graph，并利用图中实体的连接模式进行推荐。
- 元路径：

$$\mathrm{P} = A_0 \xrightarrow{R_1} A_1 \xrightarrow{R_2} \cdots \xrightarrow{R_k} A_k$$

定义了一段在A_0和A_k之间的复合关系 $\mathrm{R} = R_1 \circ R_2 \circ \cdots \circ R_k$，给定一段元路径，元路径中存在很多条实例路径。

Path-Based Methods

2018：Leveraging Meta-path based Context for Top-N Recommendation with A Neural Co-Attention Model

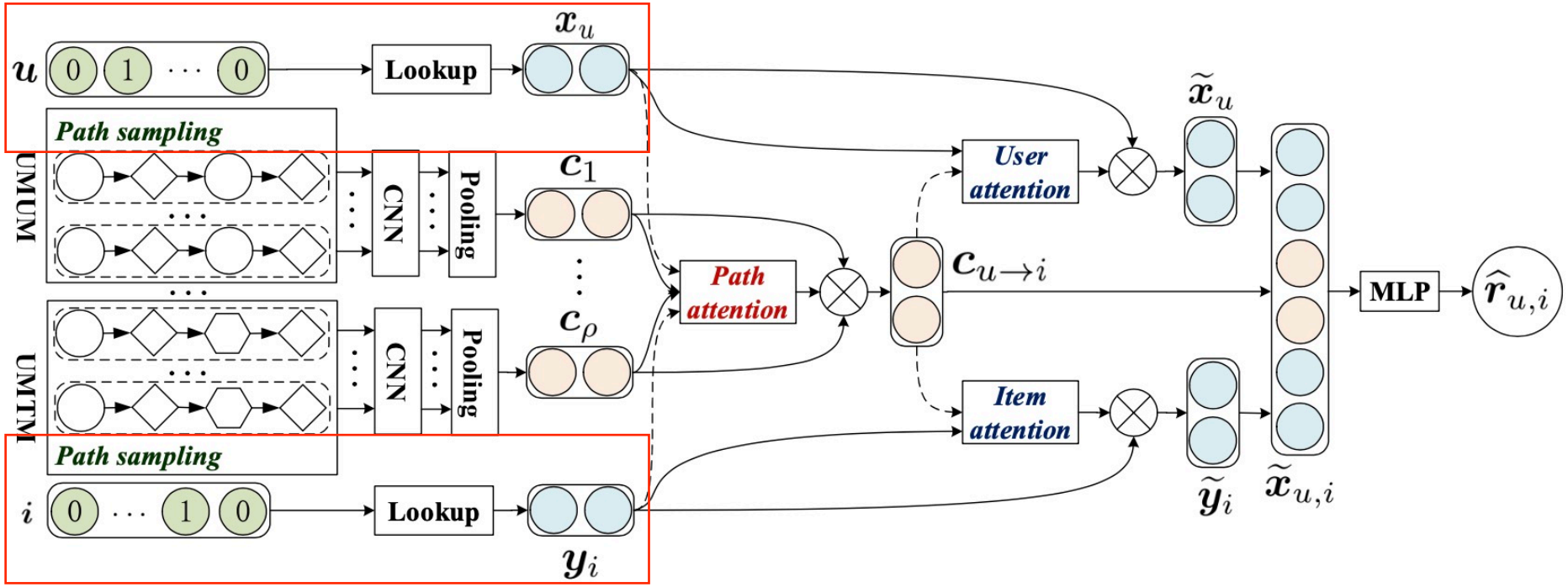
显式地引入元路径，从基于元路径的上下文中挖掘和提取有用的信息，从而提高推荐性能。

Problem formulation :

给定一个具有用户隐式反馈矩阵R的异构信息网络G

任务：为每个用户$u \in \mathcal{U}$，生成用户感兴趣的推荐列表

The framework :

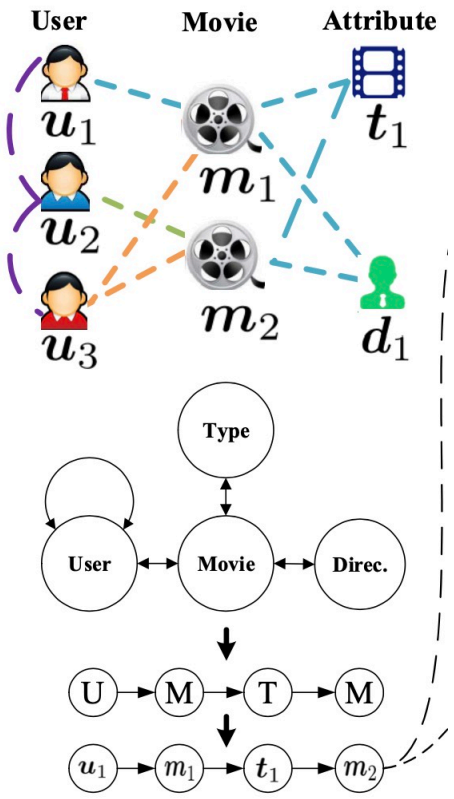


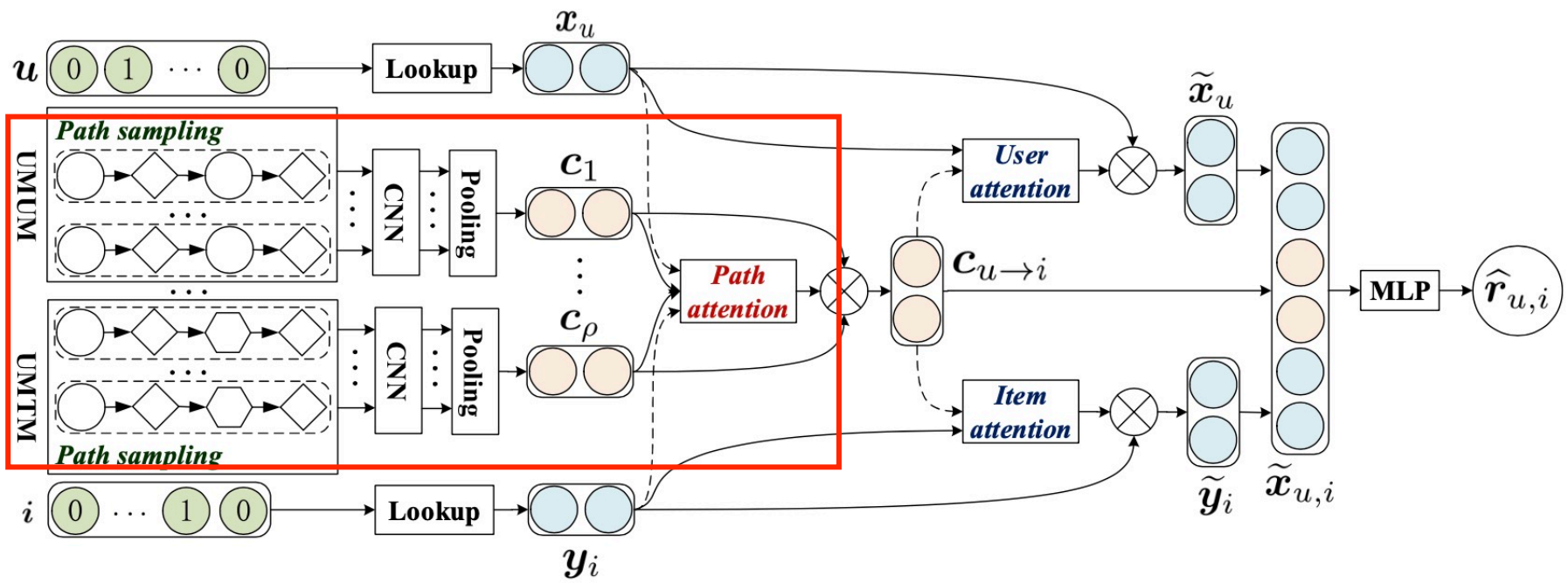
User and item embedding:

$$x_u = \mathrm{P}^T \cdot p_u$$

$$y_i = Q^T \cdot q_i$$

The framework :

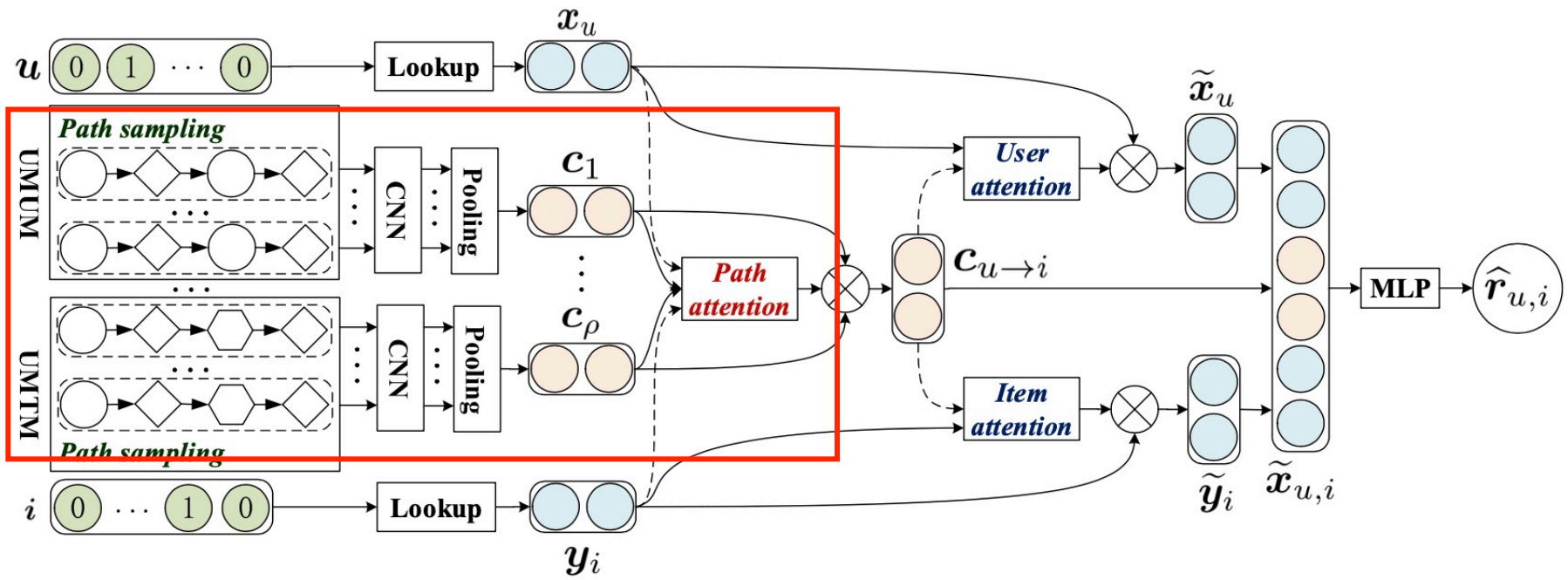




获得user和item之间的元路径信息：

1. 通过基于权重的随机游走策略采样路径实例，给定一个元路径，只保留平均优先级最高的前K个路径实例。

The framework :



获得user和item之间的元路径信息：

2. 获得k个路径实例之后，把节点序列输入CNN，得到每条路径实例的嵌入 h_p。

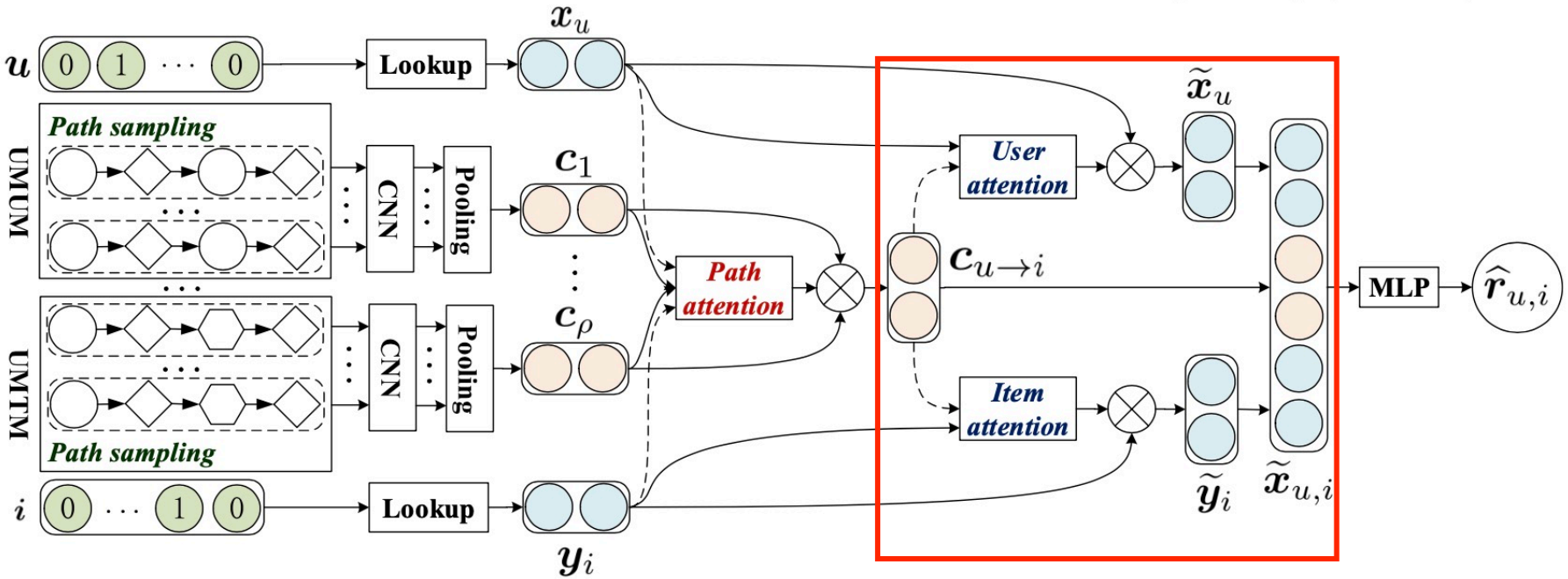
3. 将得到的k个路径实例嵌入进行max-pooling操作，得到元路径嵌入c_p。

4. 针对u和i之间所有的元路径嵌入c_p，通过两层计算注意力权重结构，最终获得user u和item i之间的上下文信息表示：

$$\mathbf{c}_{u\to i} = \sum_{\rho\in\mathcal{M}_{u\to i}} \alpha_{u,i,\rho}\cdot\mathbf{c}_\rho,$$

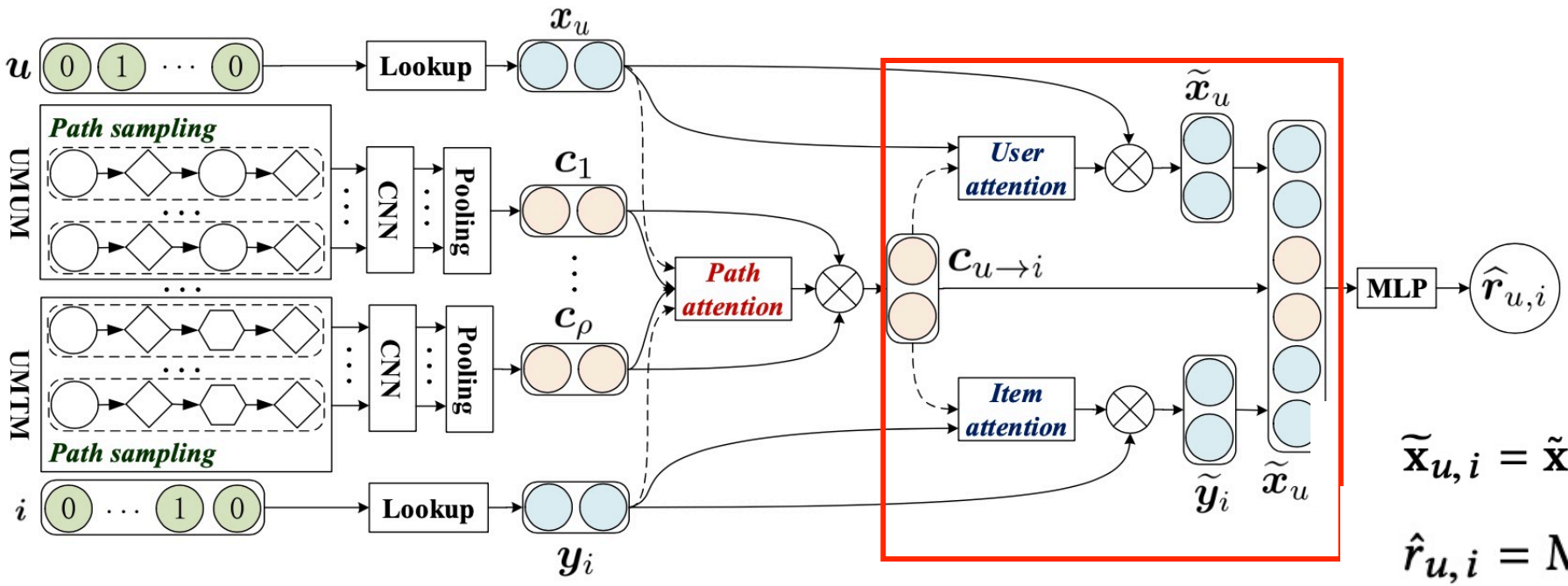
The framework :

$$\boldsymbol{\beta}_u = f(\mathbf{W}_u \mathbf{x}_u + \mathbf{W}_{u \to i} \mathbf{c}_{u \to i} + \mathbf{b}_u),$$
$$\boldsymbol{\beta}_i = f(\mathbf{W}'_i \mathbf{y}_i + \mathbf{W}'_{u \to i} \mathbf{c}_{u \to i} + \mathbf{b}'_i),$$



The framework :

$$\boldsymbol{\beta}_u = f(\mathbf{W}_u \mathbf{x}_u + \mathbf{W}_{u\to i} \mathbf{c}_{u\to i} + \mathbf{b}_u),$$
$$\boldsymbol{\beta}_i = f(\mathbf{W}'_i \mathbf{y}_i + \mathbf{W}'_{u\to i} \mathbf{c}_{u\to i} + \mathbf{b}'_i),$$



$$\widetilde{\mathbf{x}}_{u,i} = \tilde{\mathbf{x}}_u \oplus \mathbf{c}_{u\to i} \oplus \tilde{\mathbf{y}}_i,$$

$$\hat{r}_{u,i} = \mathrm{MLP}(\widetilde{\mathbf{x}}_{u,i}),$$

$$\tilde{\mathbf{x}}_u = \boldsymbol{\beta}_u \odot \mathbf{x}_u,$$
$$\tilde{\mathbf{y}}_i = \boldsymbol{\beta}_i \odot \mathbf{y}_i.$$

Performance:

Model	Movielens			LastFM			Yelp		
	Prec@10	Recall@10	NDCG@10	Prec@10	Recall@10	NDCG@10	Prec@10	Recall@10	NDCG@10
ItemKNN	0.2578	0.1536	0.5692	0.4160	0.4513	0.7981	0.1386	0.5421	0.5378
BRP	0.3010	0.1946	0.6459	0.4129	0.4492	0.8099	0.1474	0.5504	0.5549
MF	0.3247	0.2053	0.6511	0.4364	0.4634	0.7921	0.1503	0.5350	0.5322
NeuMF	0.3293*	0.2090	0.6587	0.4540	0.4678	0.8104	0.1504	0.5857	0.5713
SVDFeature$_{hete}$	0.3171	0.2021	0.6445	0.4576	0.4841	0.8290*	0.1404	0.5613	0.5289
SVDFeature$_{mp}$	0.3109	0.1929	0.6536	0.4391	0.4651	0.8116	0.1524	0.5932	0.5974*
HeteRS	0.2485	0.1674	0.5967	0.4276	0.4489	0.8026	0.1423	0.5613	0.5600
FMG$_{rank}$	0.3256	0.2165*	0.6682*	0.4630*	0.4916*	0.8263	0.1538*	0.5951*	0.5861
MCRec$_{rand}$	0.3223	0.2104	0.6650	0.4540	0.4795	0.8002	0.1510	0.5842	0.5718
MCRec$_{avg}$	0.3270	0.2111	0.6631	0.4645	0.4914	0.8311	0.1595	0.5933	0.6021
MCRec$_{mp}$	0.3401	0.2200	0.6828	0.4662	0.4924	0.8428	0.1655	0.6303	0.6228
MCRec	**0.3451**$^{\#}$	**0.2256**$^{\#}$	**0.6900**$^{\#}$	**0.4807**$^{\#}$	**0.5068**$^{\#}$	**0.8526**$^{\#}$	**0.1686**$^{\#}$	**0.6326**$^{\#}$	**0.6301**$^{\#}$

Path-Based Methods

2019：Explainable Reasoning over Knowledge Graphs for Recommendation（KPRN）

通过组合实体和关系的语义来生成路径表示，并且利用路径中的顺序依赖关系，对路径进行有效的推理，从而推断用户对项目的偏好。

Problem formulation :

给定用户u、目标物品i、一组连接u和i的路径P

任务：预测用户u和物品i之间交互的分数

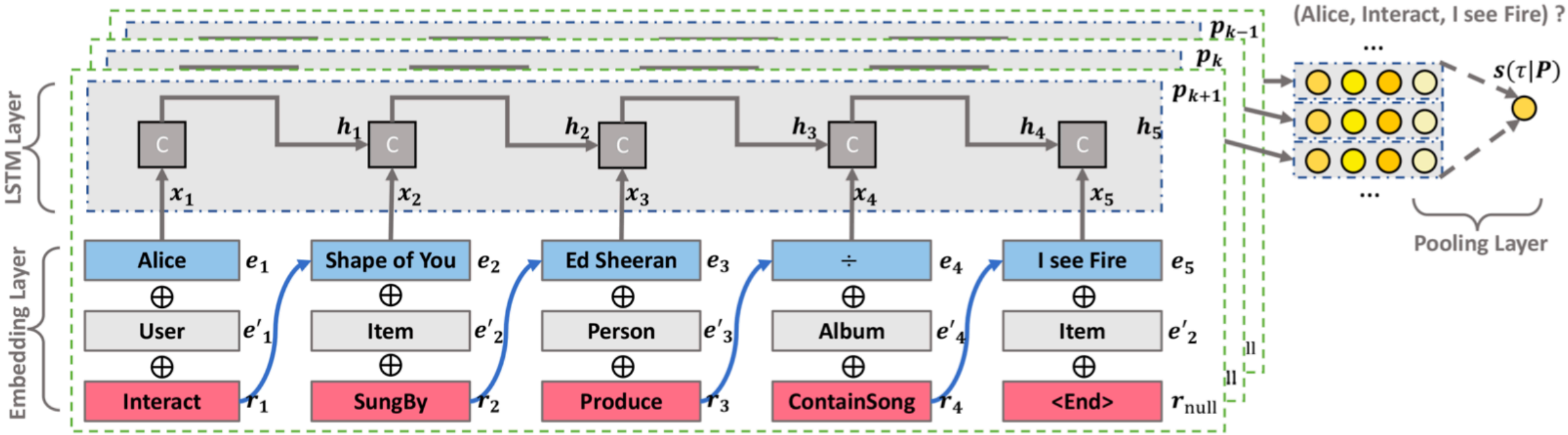
The framework:

Embedding layer

LSTM layer

Pooling layer

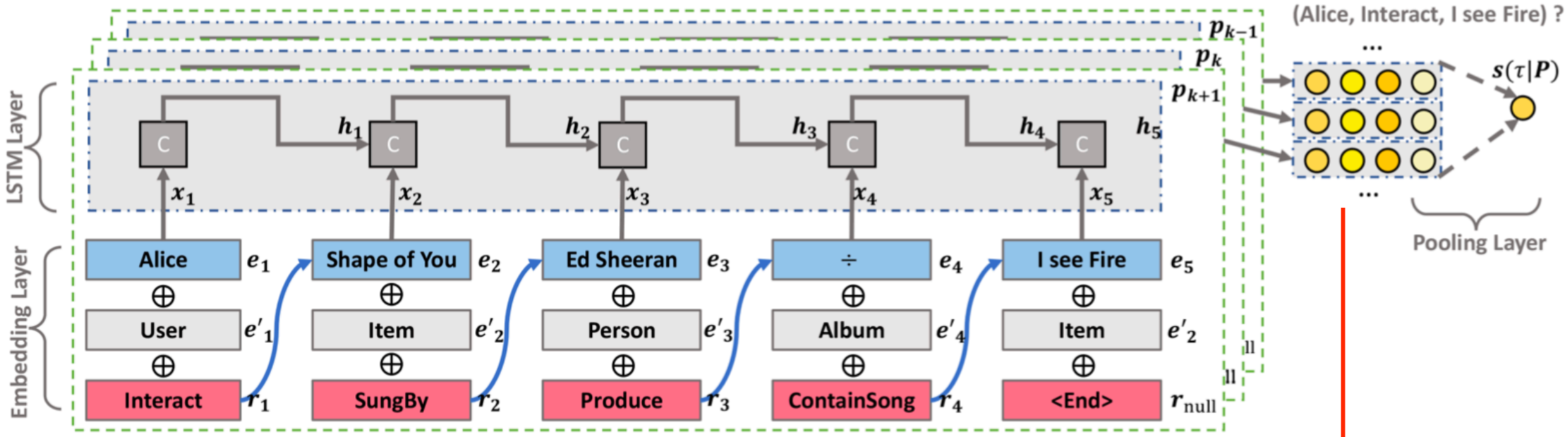
The framework:



Embedding layer:

用每一条路径p_k中的实体和关系的嵌入向量，来表示p_k的嵌入序列：

$$[e_1, r_1, e_2, \cdots, r_{L-1}, e_L]$$



LSTM layer:

在第$l-1$步的输入：

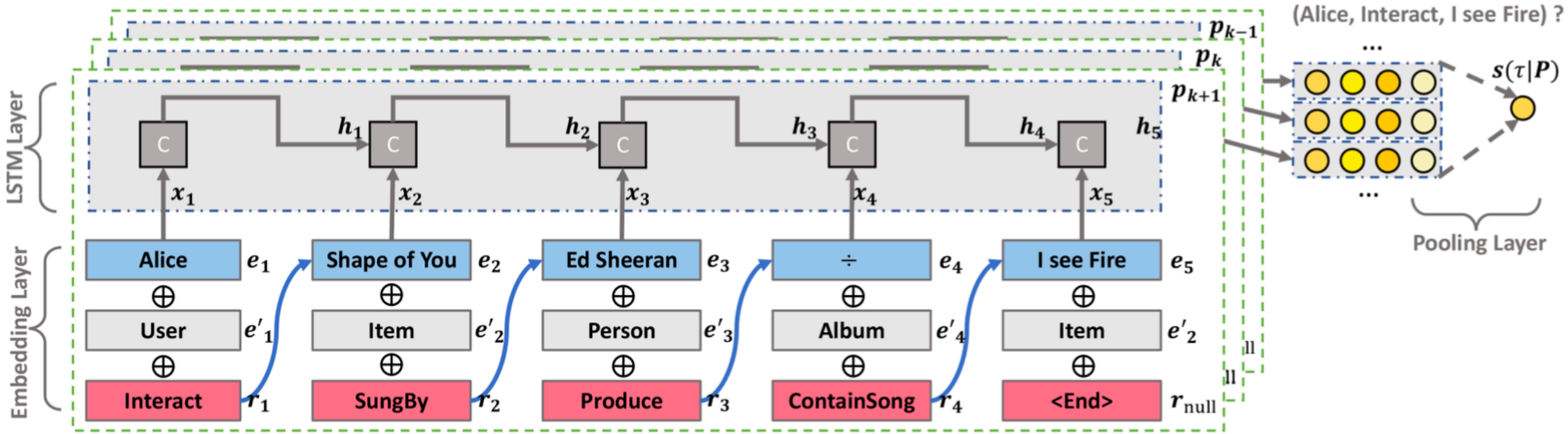
$$\mathbf{x}_{l-1} = \mathbf{e}_{l-1} \oplus \mathbf{e}'_{l-1} \oplus \mathbf{r}_{l-1},$$

整条路径p_k由最后一个状态h_L表示：

$$\begin{aligned}
\mathbf{z}_l &= \tanh(\mathbf{W}_z\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_z) \\
\mathbf{f}_l &= \sigma(\mathbf{W}_f\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_f) \\
\mathbf{i}_l &= \sigma(\mathbf{W}_i\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_i) \\
\mathbf{o}_l &= \sigma(\mathbf{W}_o\mathbf{x}_l + \mathbf{W}_h\mathbf{h}_{l-1} + \mathbf{b}_o) \\
\mathbf{c}_l &= \mathbf{f}_l \odot \mathbf{c}_{l-1} + \mathbf{i}_l \odot \mathbf{z}_l \\
\mathbf{h}_l &= \mathbf{o}_l \odot \tanh(\mathbf{c}_l)
\end{aligned}$$

每条路径得到相应的路径表示后，输入到两层全连接层，得到每条路径的分数：

$$s(\tau|\mathbf{p}_k) = \mathbf{W}_2^\top \mathrm{ReLU}(\mathbf{W}_1^\top \mathbf{p}_k),$$



Weight Pooling layer:

得到K条路径的分数$S = \{s_1, s_2, \cdots, s_K\}$后，进行权重池化操作来聚合所有路径的分数：

$$g(s_1, s_2, \cdots, s_K) = \log \left[\sum_{k=1}^{K} \exp \left(\frac{s_k}{\gamma} \right) \right],$$

$$\hat{y}_{ui} = \sigma(g(s_1, s_2, \cdots, s_K)),$$

Unified Methods

- 基于嵌入的方法利用KG中user/item的语义表示进行推荐
- 基于路径的方法使用语义连接性信息
- 统一的方法将实体和关系的语义表示和连接信息结合起来，基于嵌入传播的思想。

Unified Methods

2019：KGAT: Knowledge Graph Attention Network for Recommendation

利用KG中的高阶信息来提供更好的推荐。

Problem formulation :

给定一个协作知识图G（包括user-item二部图G_1和知识图G_2）

任务：预测用户u会选择物品i的概率

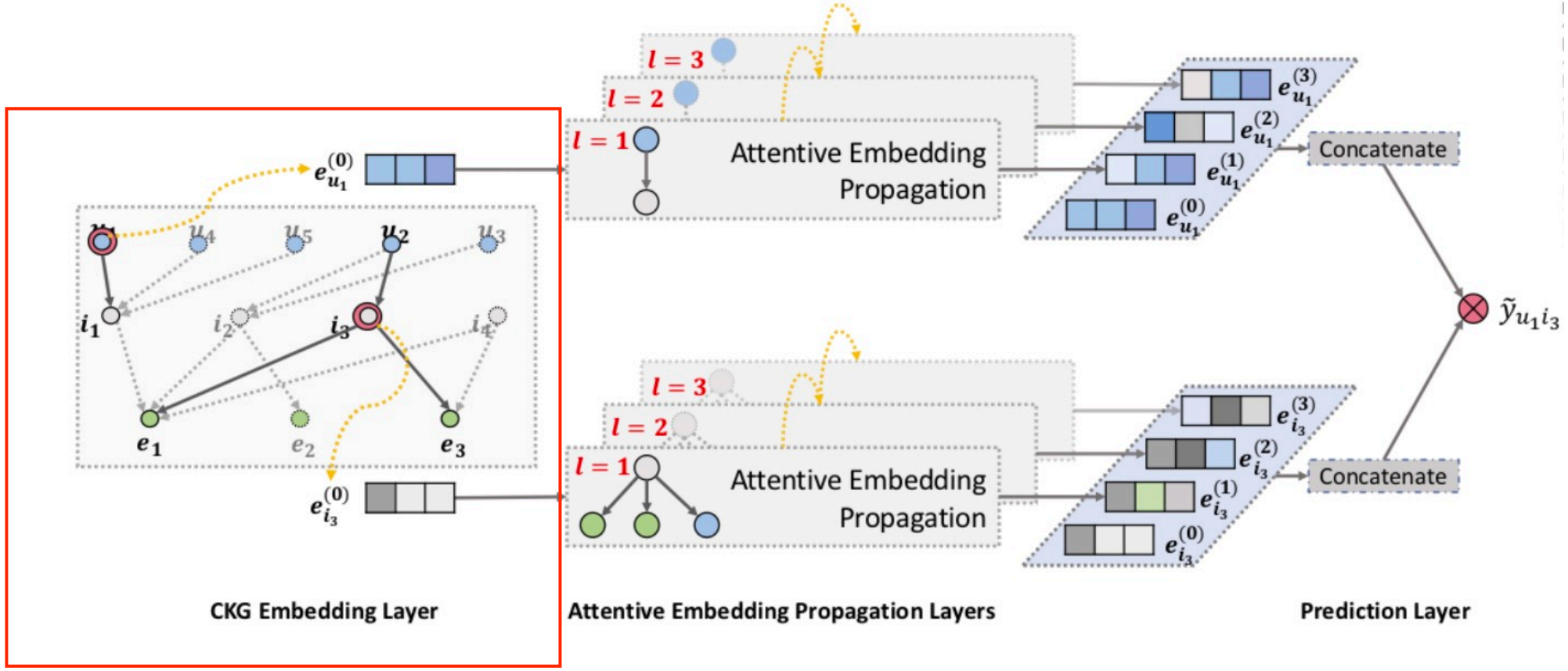
The framework:

Embedding layer

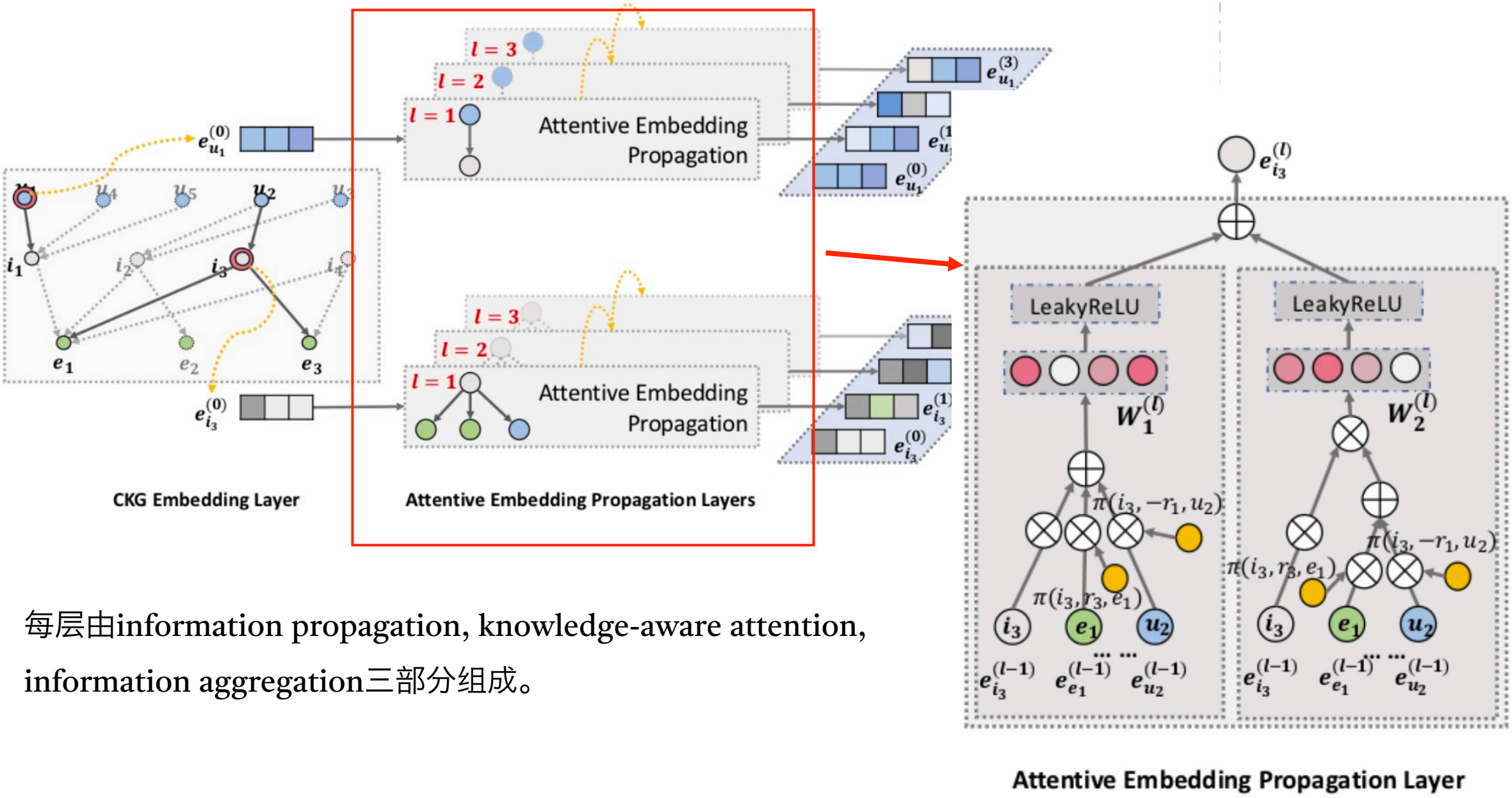
Attentive embedding propagation layer

Prediction layer

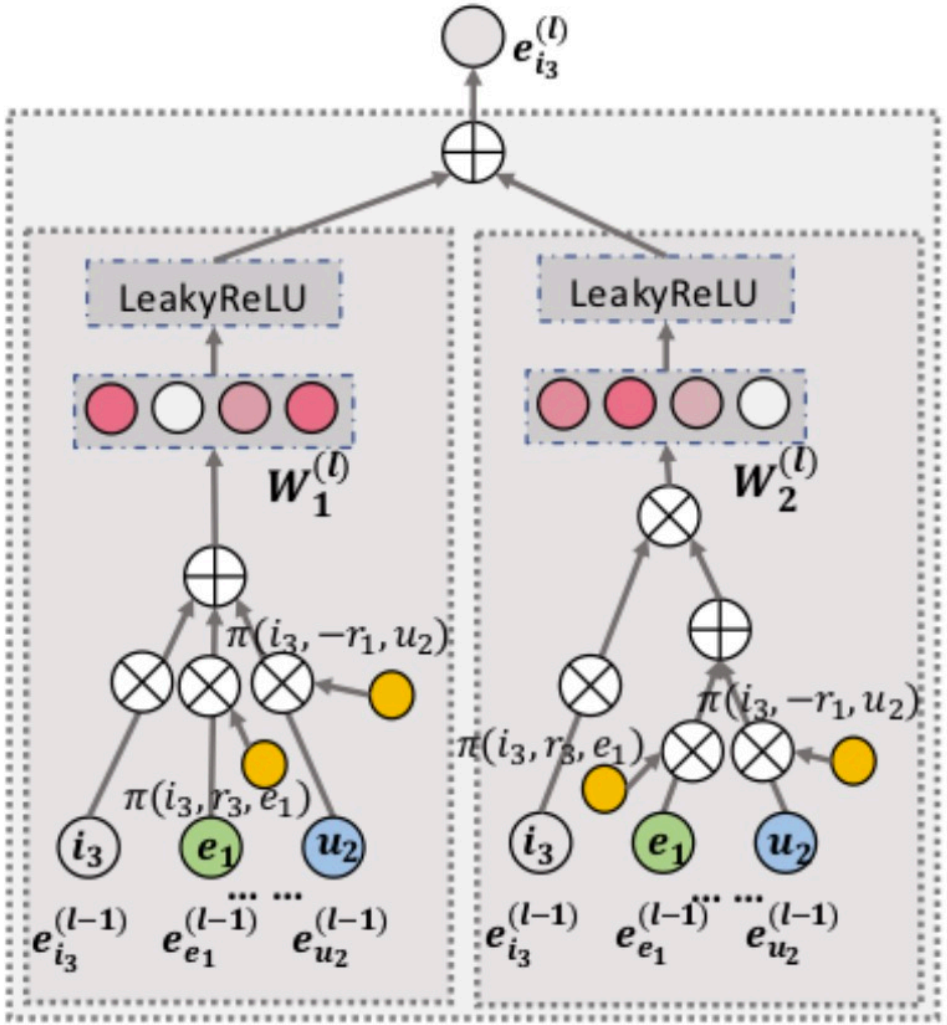
The framework:



使用TransR模型，得到实体和关系的向量表示，同时保留图结构。



每层由information propagation, knowledge-aware attention, information aggregation三部分组成。



Attentive Embedding Propagation Layer

information propagation :

一个实体可以包含在多个三元组中，因此可以作为两个不同三元组的桥梁。例如：

$$e_1 \xrightarrow{r_2} i_2 \xrightarrow{-r_1} u_2$$

$$e_2 \xrightarrow{r_3} i_2 \xrightarrow{-r_1} u_2$$

i2可以将自己的属性e1和e2来丰富自身的特性，然后为u2提供偏好信息。

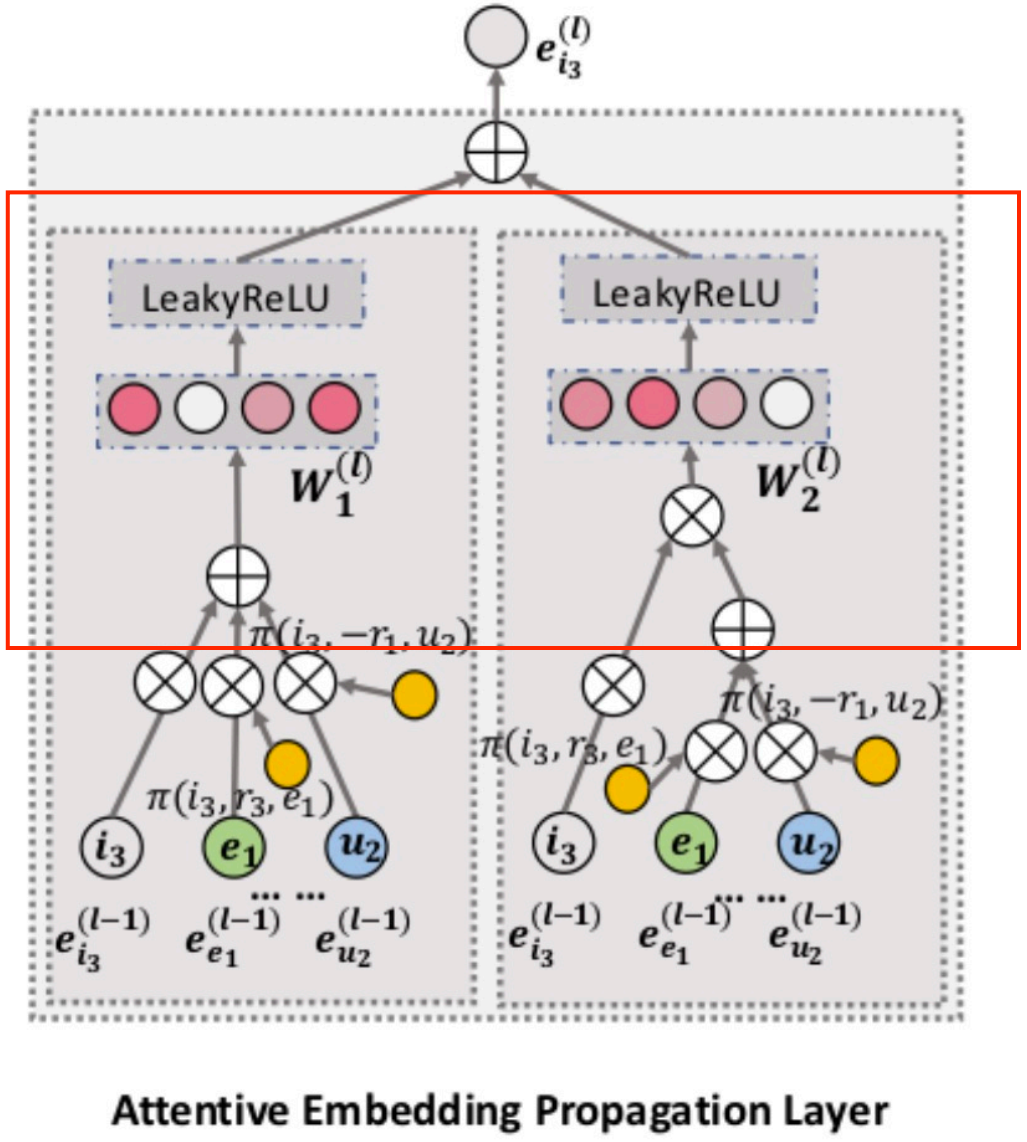
给定一个实体h，把将h作为头实体的所有三元组的集合称为h的ego-network，并表示为

$$\mathbf{e}_{\mathcal{N}_h} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathbf{e}_t,$$

其中：

$$\pi(h,r,t) = (\mathbf{W}_r\mathbf{e}_t)^\top \tanh\Big((\mathbf{W}_r\mathbf{e}_h + \mathbf{e}_r)\Big),$$

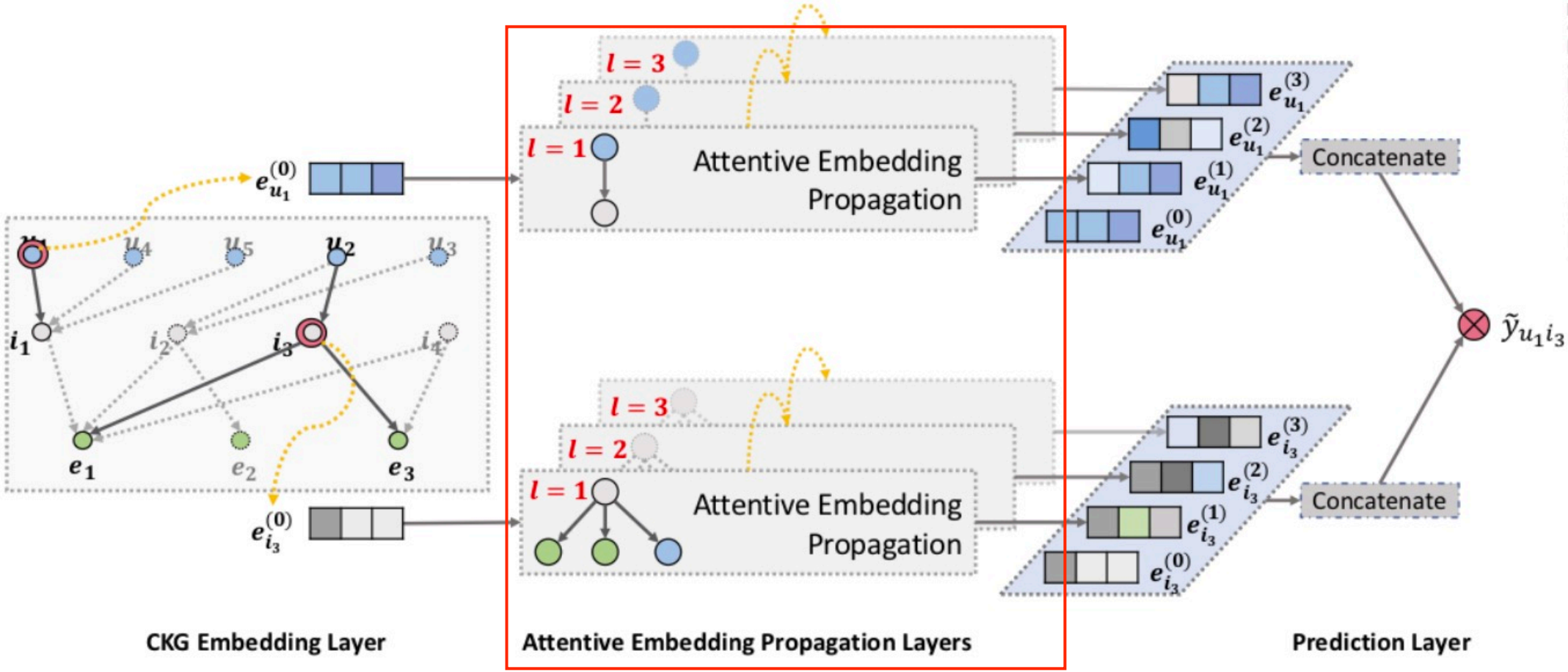
为更接近的实体传播更多的信息



Attentive Embedding Propagation Layer

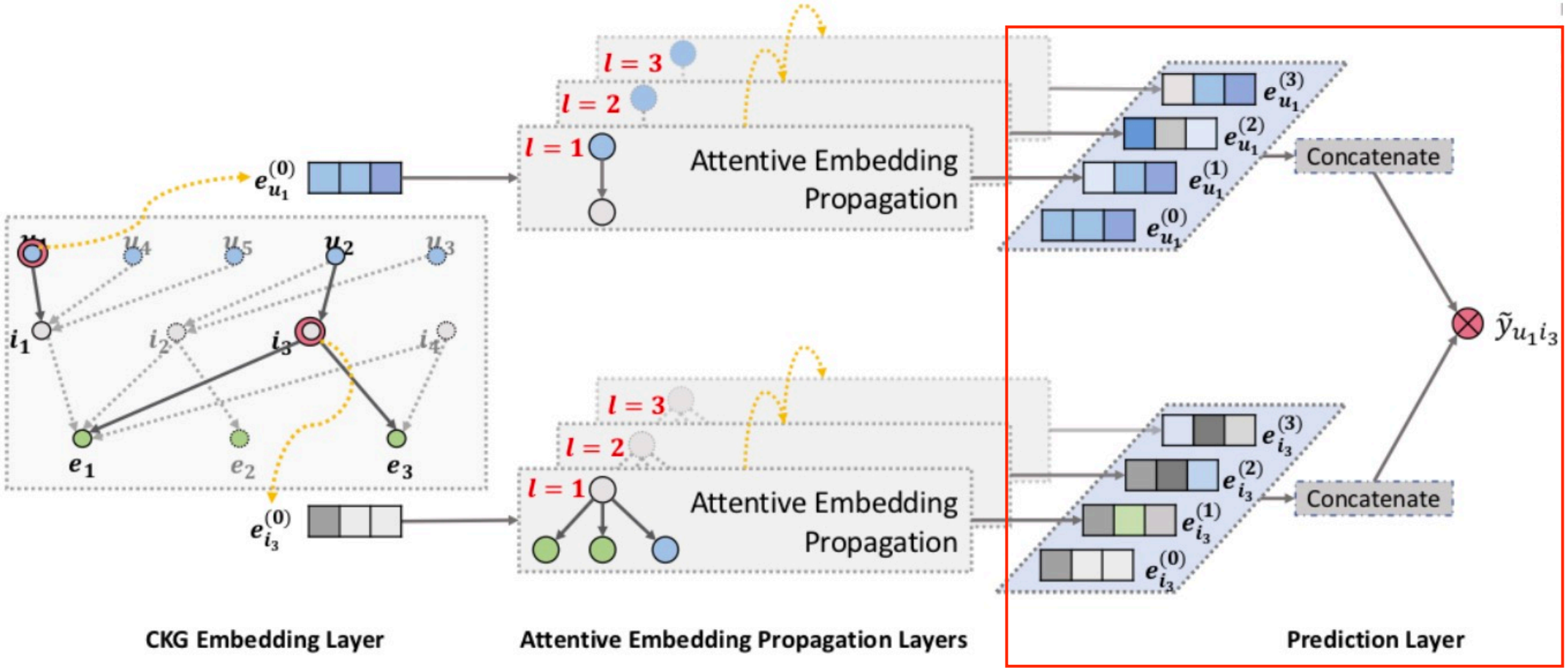
Information Aggregation :

$$f_{\text{Bi-Interaction}} = \text{LeakyReLU}\left(\mathbf{W}_1(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})\right) + \text{LeakyReLU}\left(\mathbf{W}_2(\mathbf{e}_h \odot \mathbf{e}_{\mathcal{N}_h})\right),$$



$$\mathbf{e}_{\mathcal{N}_h}^{(l-1)} = \sum_{(h,r,t)\in\mathcal{N}_h} \pi(h,r,t)\mathbf{e}_t^{(l-1)},$$

$$\mathbf{e}_h^{(l)} = f\left(\mathbf{e}_h^{(l-1)}, \mathbf{e}_{\mathcal{N}_h}^{(l-1)}\right),$$



$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \| \cdots \| \mathbf{e}_u^{(L)}, \quad \mathbf{e}_i^* = \mathbf{e}_i^{(0)} \| \cdots \| \mathbf{e}_i^{(L)},$$

$$\hat{y}(u, i) = \mathbf{e}_u^{*\top} \mathbf{e}_i^*.$$

Performance :

	Amazon-Book		Last-FM		Yelp2018	
	recall	ndcg	recall	ndcg	recall	ndcg
FM	0.1345	0.0886	0.0778	0.1181	0.0627	0.0768
NFM	**0.1366**	0.0913	**0.0829**	0.1214	0.0660	0.0810
CKE	0.1343	0.0885	0.0736	0.1184	0.0657	0.0805
CFKG	0.1142	0.0770	0.0723	0.1143	0.0522	0.0644
MCRec	0.1113	0.0783	-	-	-	-
RippleNet	0.1336	0.0910	0.0791	0.1238	**0.0664**	**0.0822**
GC-MC	0.1316	0.0874	0.0818	**0.1253**	0.0659	0.0790
KGAT	**0.1489***	**0.1006***	**0.0870***	**0.1325***	**0.0712***	**0.0867***
%Improv.	8.95%	10.05%	4.93%	5.77%	7.18%	5.54%

Unified Methods

2020：Attentive Knowledge Graph Embedding for Personalized Recommendation (AKGE)

通过挖掘语义子图来描述KG，结合注意图神经网络，更好地学习子图中实体的语义嵌入来进行推荐。

Problem formulation :

给定一个知识图谱G，用户集U

任务：为每个用户生成一个用户感兴趣的物品列表

The framework:

Subgraph Construction

AGNN

MLP Prediction

Subgraph Construction :

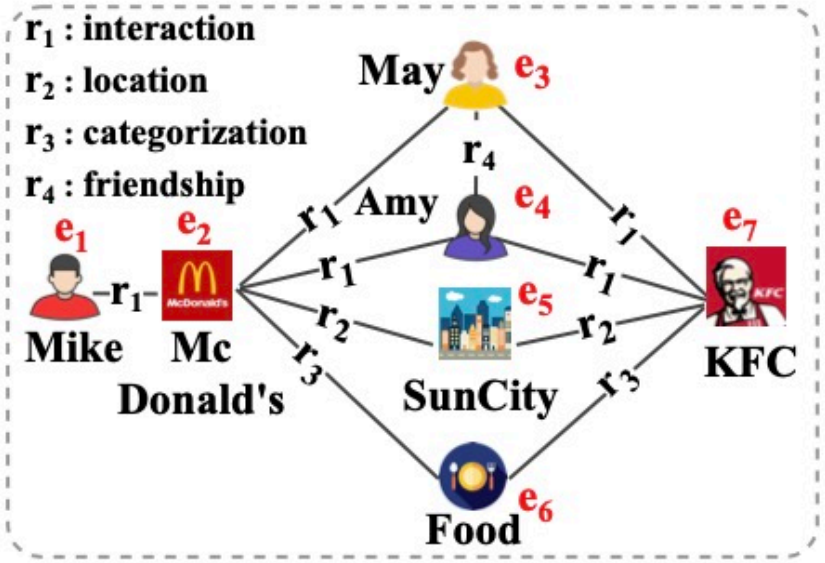
首先，在图中用如下策略进行路径采样：

假设距离较短的邻居有助于反映更可靠的连接。因此， walker的下一步是由当前实体和候选实体之间的距离决定的。

具体做法：

将所有实体作为嵌入投影到欧几里德空间中，并通过相应嵌入的欧几里德距离来测量它们的距离。同时，利用TransR对KG中实体的嵌入进行了预训练。这样，我们可以计算出一条路径上任意两个连续实体的成对距离，并将所有这些距离相加为整条路径的距离。最后，只保留距离最短的K条路径来构造语义子图。

Subgraph Construction :



(a) An example of the subgraph

	e_1	e_2	e_3	e_4	e_5	e_6	e_7
e_1	0	1	0	0	0	0	0
e_2	1	0	1	1	1	1	0
e_3	0	1	0	1	0	0	1
e_4	0	1	1	0	0	0	1
e_5	0	1	0	0	0	0	1
e_6	0	1	0	0	0	0	1
e_7	0	0	1	1	1	1	0

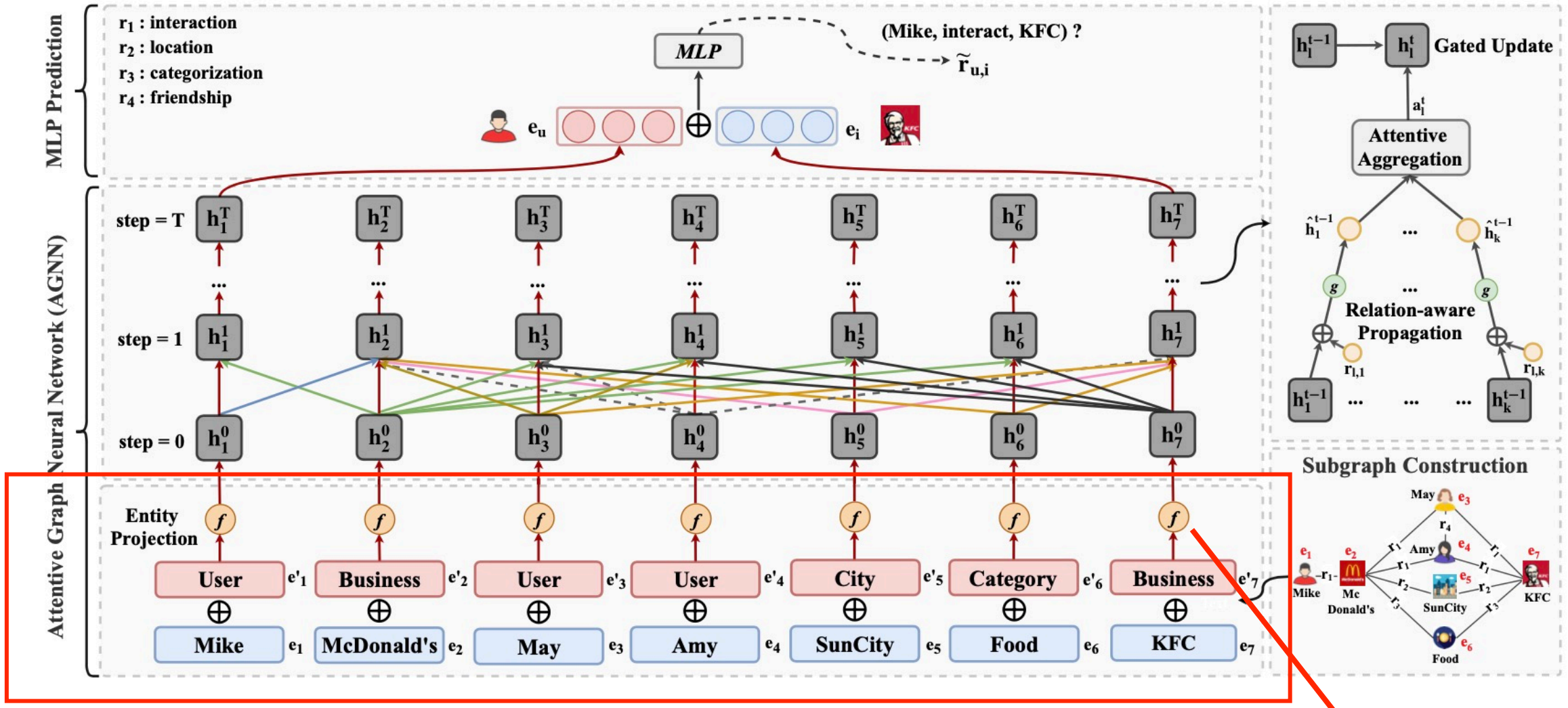
(b) Adjacency matrix (A)

遍历路径P（u，i），将每个对象和连接分别作为实体和关系沿着路径映射到子图(a)中。

Mike、McDonald’s、May、KFC映射为实体：e1、e2、e3、e7。

并且将子图的结构信息编码为一个邻接矩阵A。

Attentive Graph Neural Network :

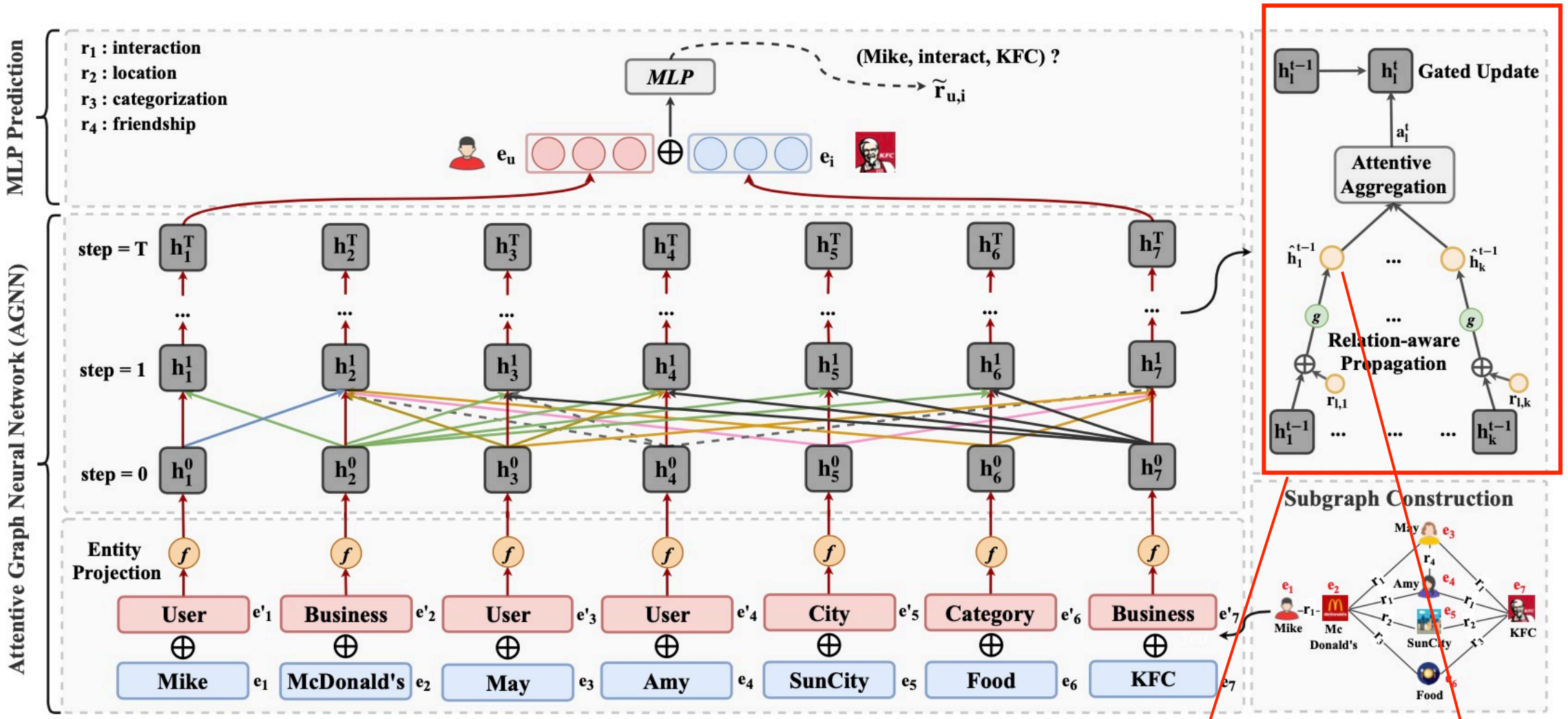


在实体映射阶段，将实体自身和实体的类型向量进行拼接，
输入到look-up层，生成一个低维表示向量。

$$\hat{\mathbf{e}}_l = f(\mathbf{e}_l \oplus \mathbf{e}'_l)$$

$$f(\mathbf{x}) = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b});$$

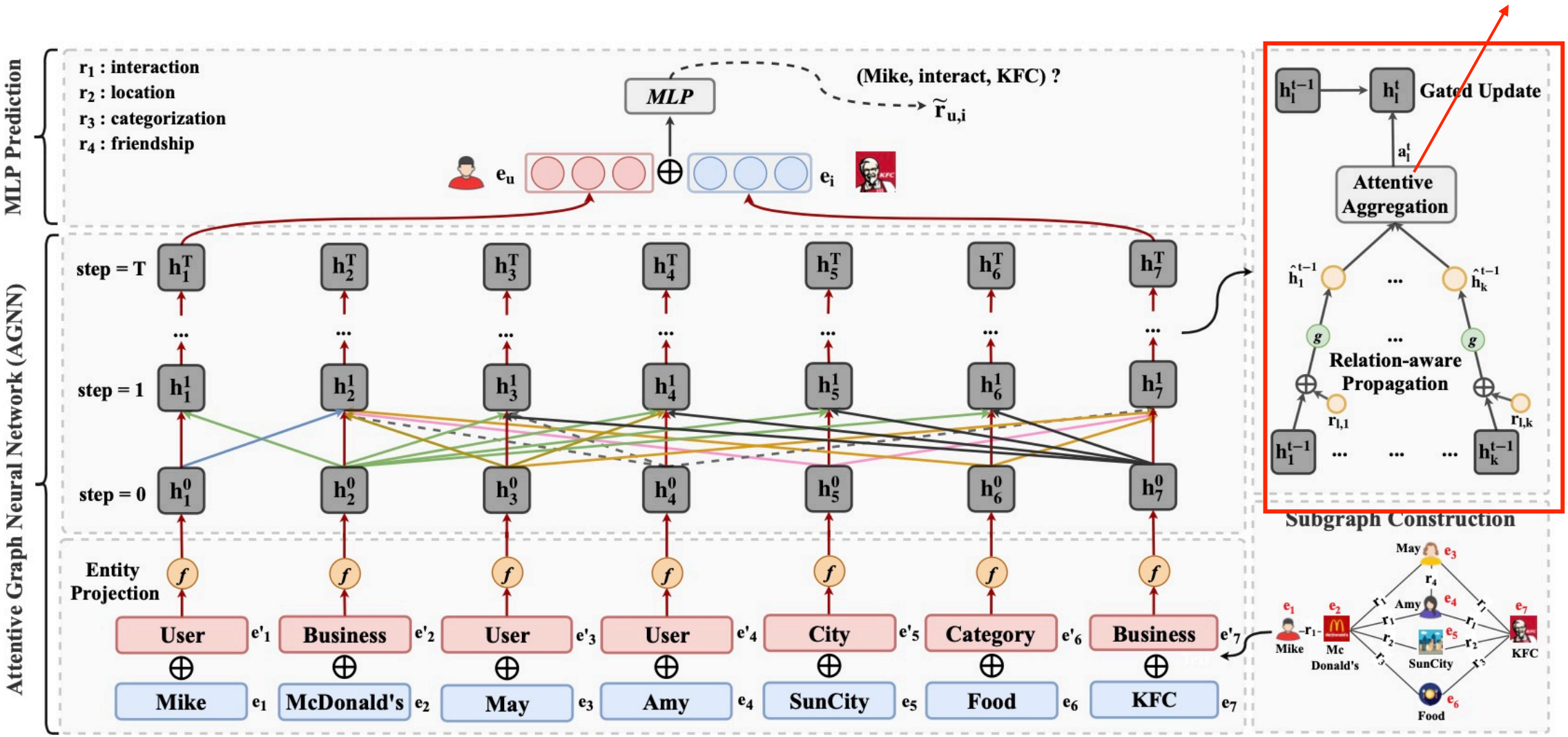
Attentive Graph Neural Network :



$$\mathbf{h}_l^0 = \hat{\mathbf{e}}_l \qquad \hat{\mathbf{h}}_k^t = g(\mathbf{h}_k^t \oplus \mathbf{r}_{l,k})$$

Attentive Graph Neural Network :

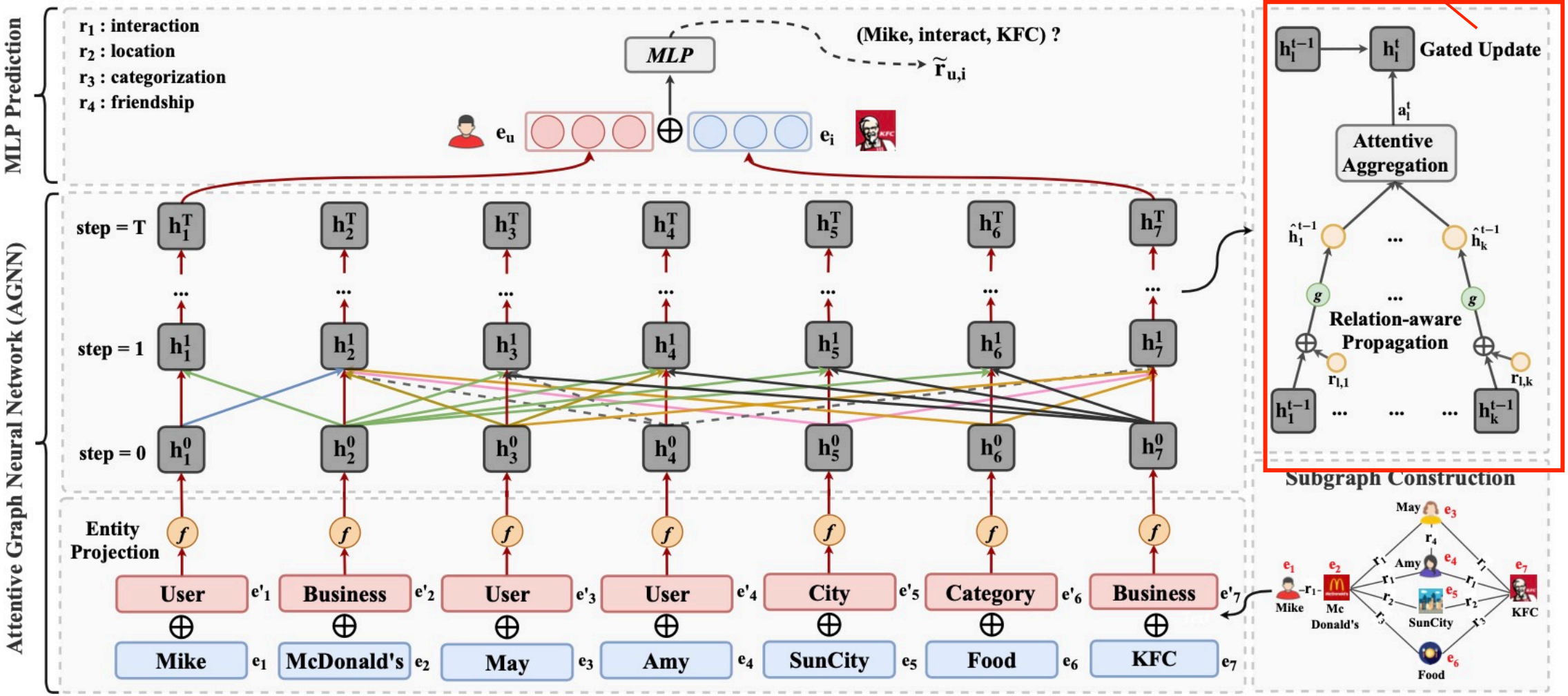
$$\mathbf{a}_l^t = (\mathbf{A}_{s_l} \odot \mathbf{Q}_l^t) \left[\hat{\mathbf{h}}_1^{t-1}, \cdots, \hat{\mathbf{h}}_{|\mathcal{E}_s|}^{t-1} \right]^\top + \mathbf{b}$$



Attentive Graph Neural Network :

$$\mathbf{z}_l^t = \sigma\left(\mathbf{W}_z \mathbf{a}_l^t + \mathbf{U}_z \mathbf{h}_l^{t-1}\right)$$

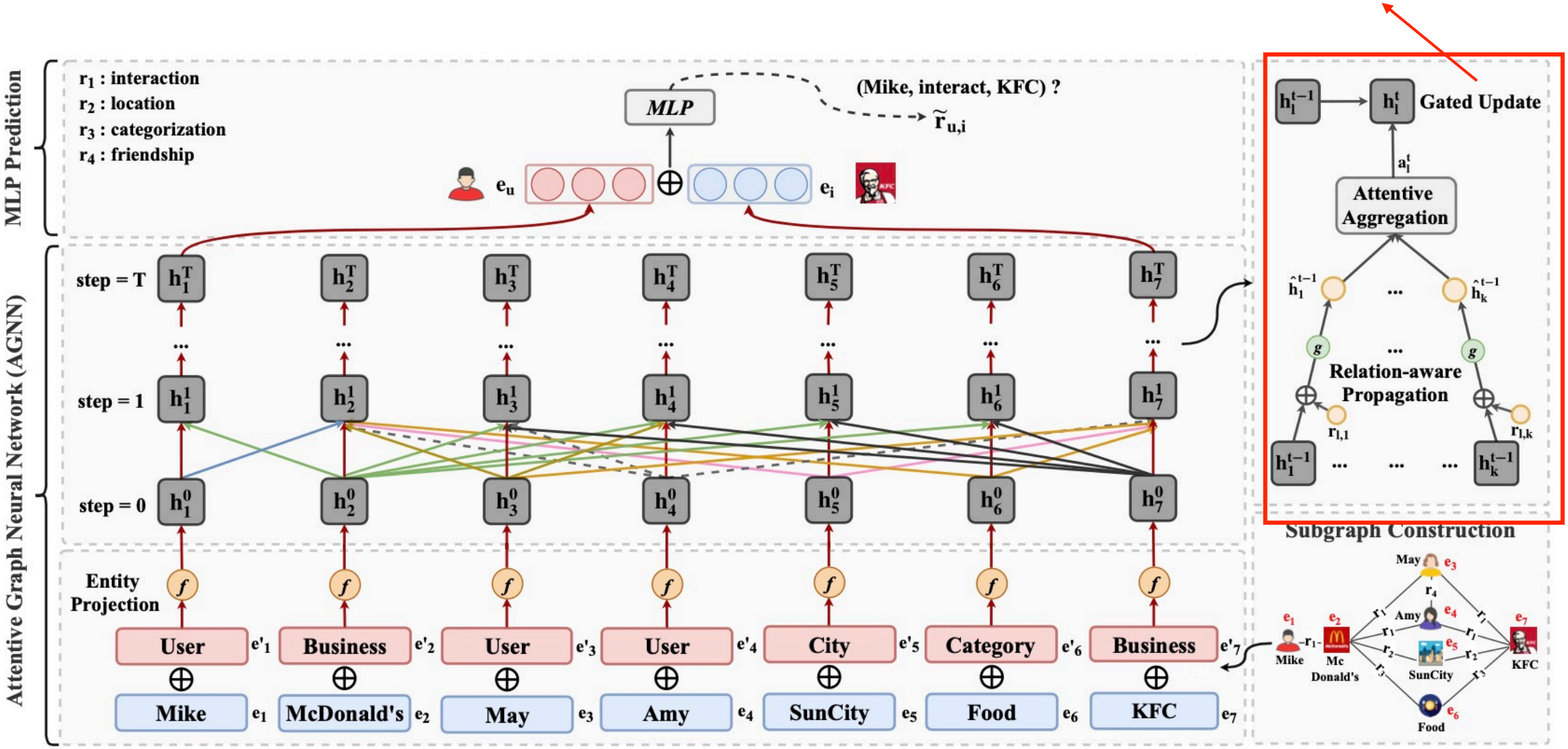
$$\mathbf{r}_l^t = \sigma\left(\mathbf{W}_r \mathbf{a}_l^t + \mathbf{U}_r \mathbf{h}_l^{t-1}\right)$$



Attentive Graph Neural Network :

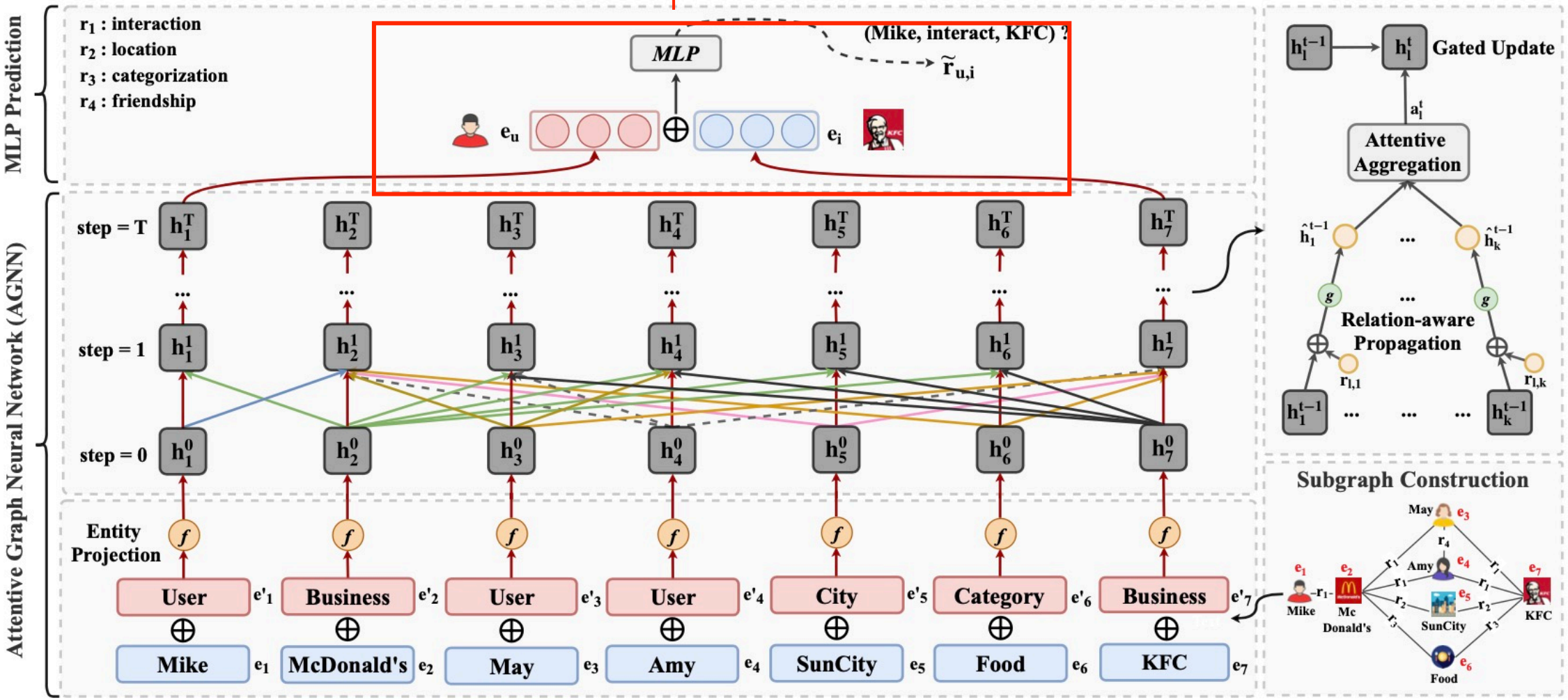
$$\widetilde{\mathbf{h}}_l^t = \tanh\left(\mathbf{W}_h \mathbf{a}_l^t + \mathbf{U}_h\left(\mathbf{r}_l^t \odot \mathbf{h}_l^{t-1}\right)\right)$$

$$\mathbf{h}_l^t = \left(1 - \mathbf{z}_l^t\right) \odot \mathbf{h}_l^{t-1} + \mathbf{z}_l^t \odot \widetilde{\mathbf{h}}_l^t$$



Attentive Graph Neural Network :

$$\tilde{r}_{u,i} = MLP(\mathbf{e}_u \oplus \mathbf{e}_i)$$



Unified Methods

2020：Jointly Non-Sampling Learning for Knowledge Graph Enhanced Recommendation

使用非抽样学习，在每次参数更新时考虑了所有样本；并且提出了高效优化算法，加速训练过程。

Negative Sampling vs Non-sampling :

	优点	缺点
负采样	训练样例少，速度快	鲁棒性差，模型很难达到最优
非采样	考虑了所有样例，模型训练效效果好	传统的非采样方法效率低

Efficient Non-sampling Knowledge Graph Embedding :

传统非采样学习使用加权回归的Loss function:

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

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

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

Efficient Non-sampling Knowledge Graph Embedding :

传统非采样学习使用加权回归的Loss function:

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

$$g_{uv} = \begin{cases} 1, & if\ interaction(user\ u, item\ v)\ is\ observed; \\ 0, & otherwise. \end{cases}$$ ，可以将已知项化为常数

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

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

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

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

non-observed数据可以由所有数据减去正样本来表示

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

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

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

基于对内积的解耦操作，重新排列

DistMult :

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

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

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

Independent from each other

The final efficient non- sampling loss for KG embedding learning :

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

由于实际数据中，正反馈的数量远小于全部样本数量，所以理论上复杂度降低为：

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

The framework :

KG embedding part

Attentive user-item preference modeling part

Joint learning part

User-Item Preference Modeling:

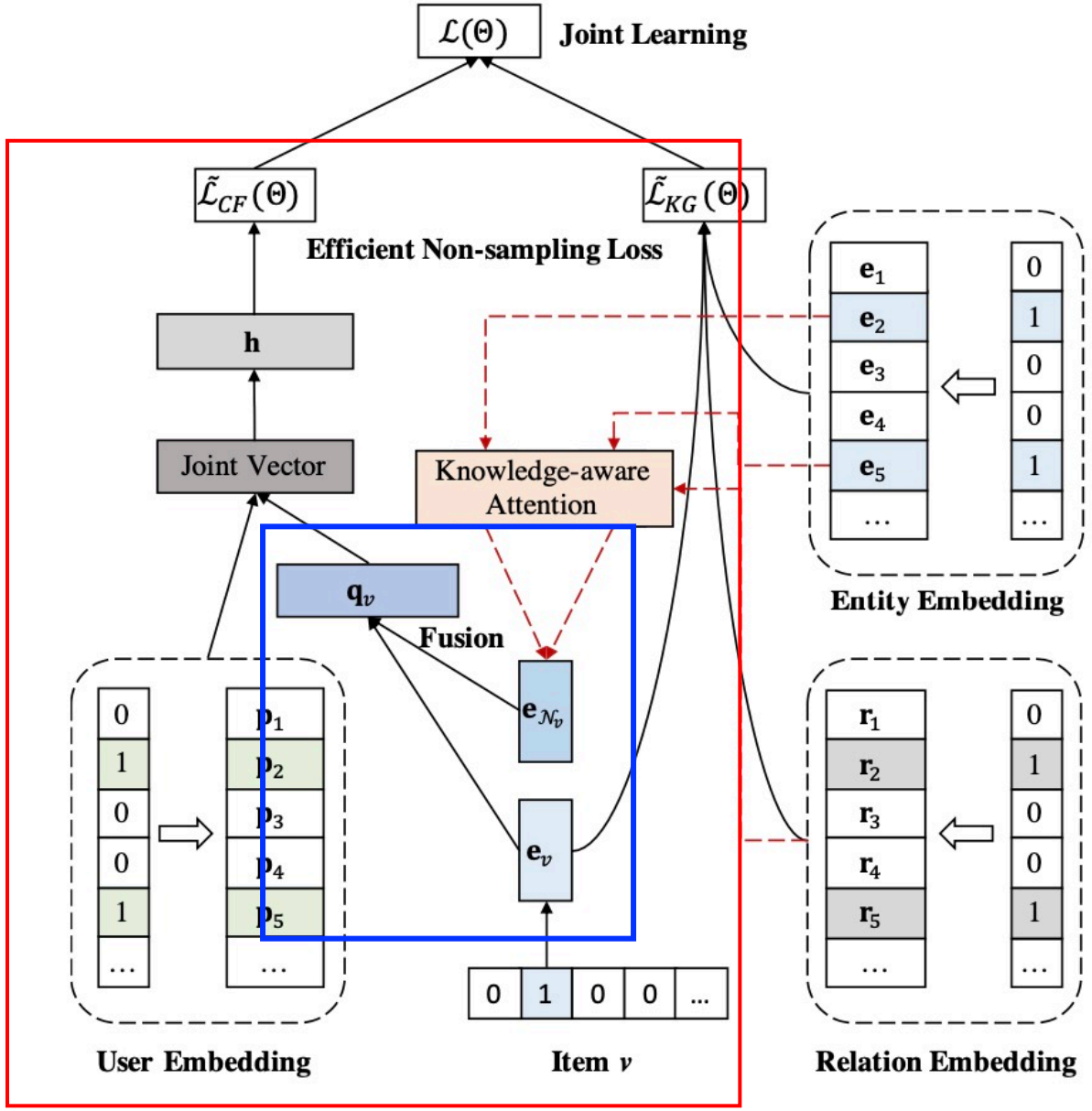
$$\hat{y}_{uv} = \mathbf{h}^T (\mathbf{p}_u \odot \mathbf{q}_v)$$

P_u: user u的embedding

q_v: 融合v自身和邻居实体的embedding

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

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



The framework :

KG embedding part

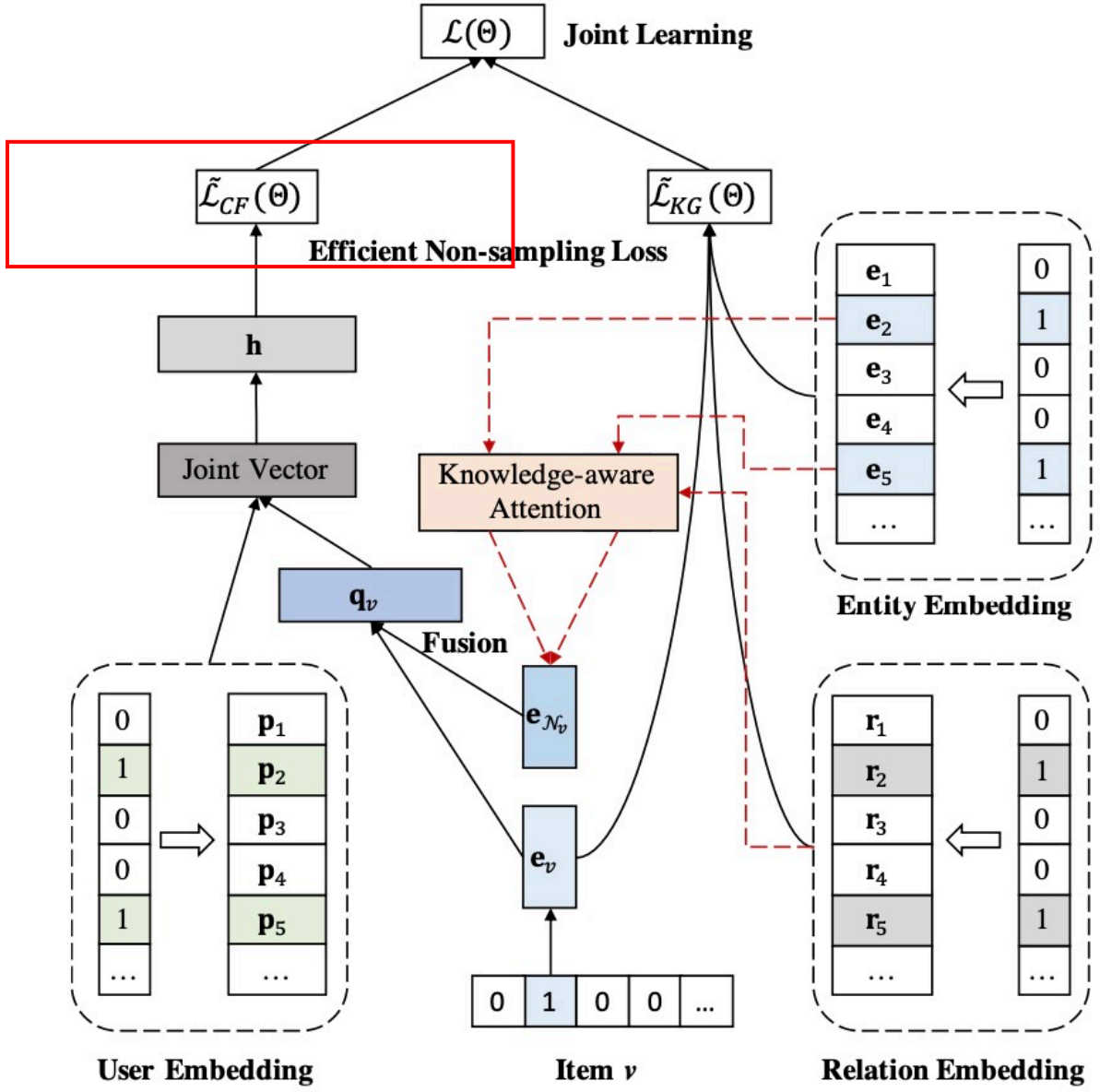
Attentive user-item preference modeling part

Joint learning part

User-Item Preference Modeling:

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

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



The framework :

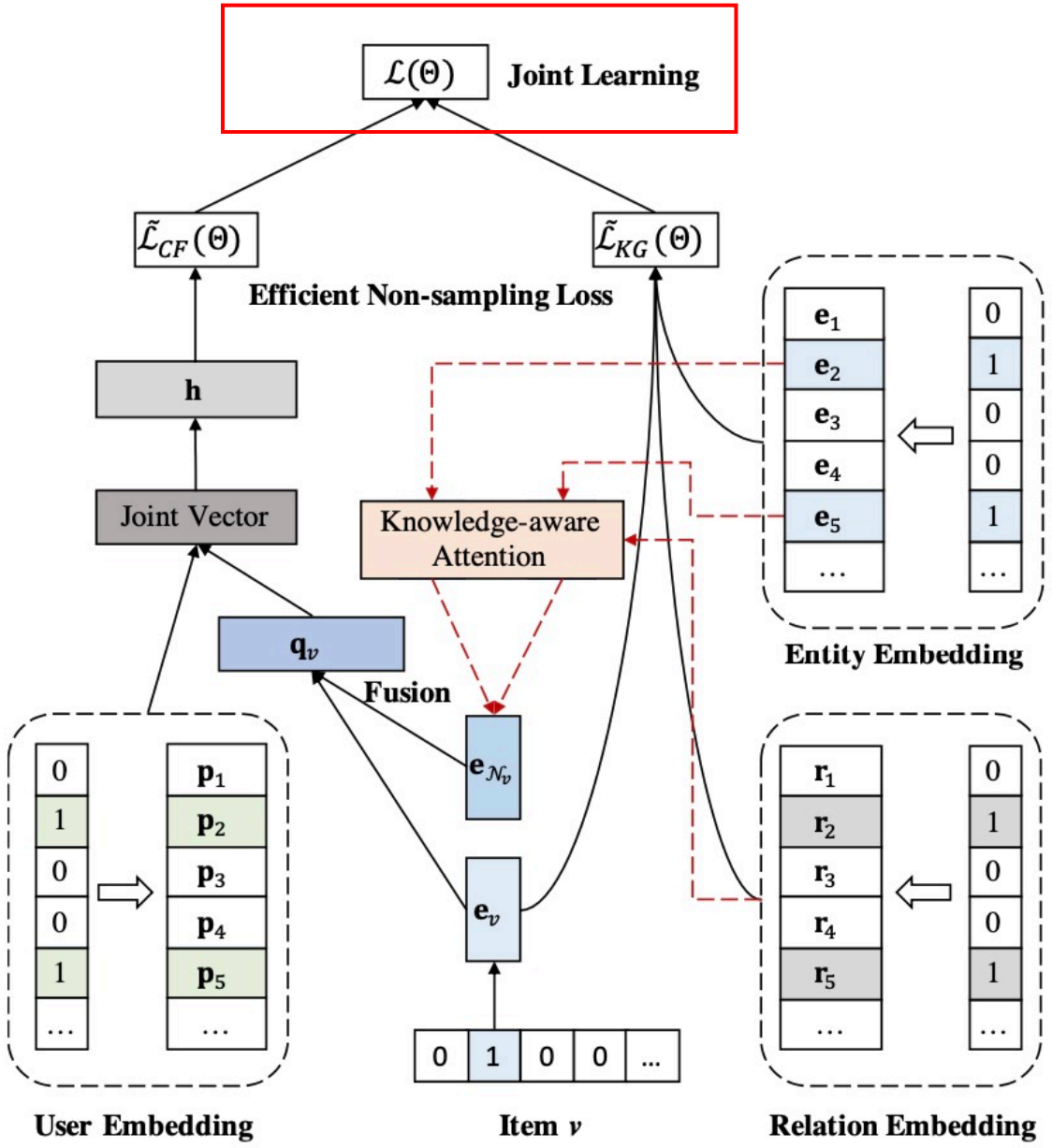
KG embedding part

Attentive user-item preference modeling part

Joint learning part

Joint learning :

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



Performance :

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.

Future Directions

- Dynamic Recommendation
- Multi-task Learning
- Cross-Domain Recommendation