Knowledge Graph Aware Recommender System

ZENG YUAN

Dec. 24th, 2020

Recommender System

- RS intend to address the information explosion by finding a small set of items for users to meet their personalized interests.

- Two categories
  - Rating prediction — explicit feedback
  - Click-through rate prediction — implicit feedback

Collaborative Filtering

- Suppose similar users have similar preferences

	Item 1	2	3	4	Similarity with user 4
user 1	2	3	3	2	0.7
user 2	3	1	1	4	0.1
user 3	2	5	4	5	0.2
user 4	?	3	4	1	

? = 0.7*2 + 0.1*3 + 0.2*2 = 2.1

Side Information

- A KG usually consists of triples (head, relation, tail)
- Items in recommender systems are also nodes in KGs

item Sherlock $\xrightarrow{\textit{starred by}}$ Benedict Cumberbatch *non-item entity*

non-item entity Benedict Cumberbatch $\xrightarrow{\textit{nationality}}$ UK *non-item entity*

1. **DKN: Deep Knowledge-Aware Network for News Recommendation**

   {Hongwei Wang et al, WWW 2018 }

2. **Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems**

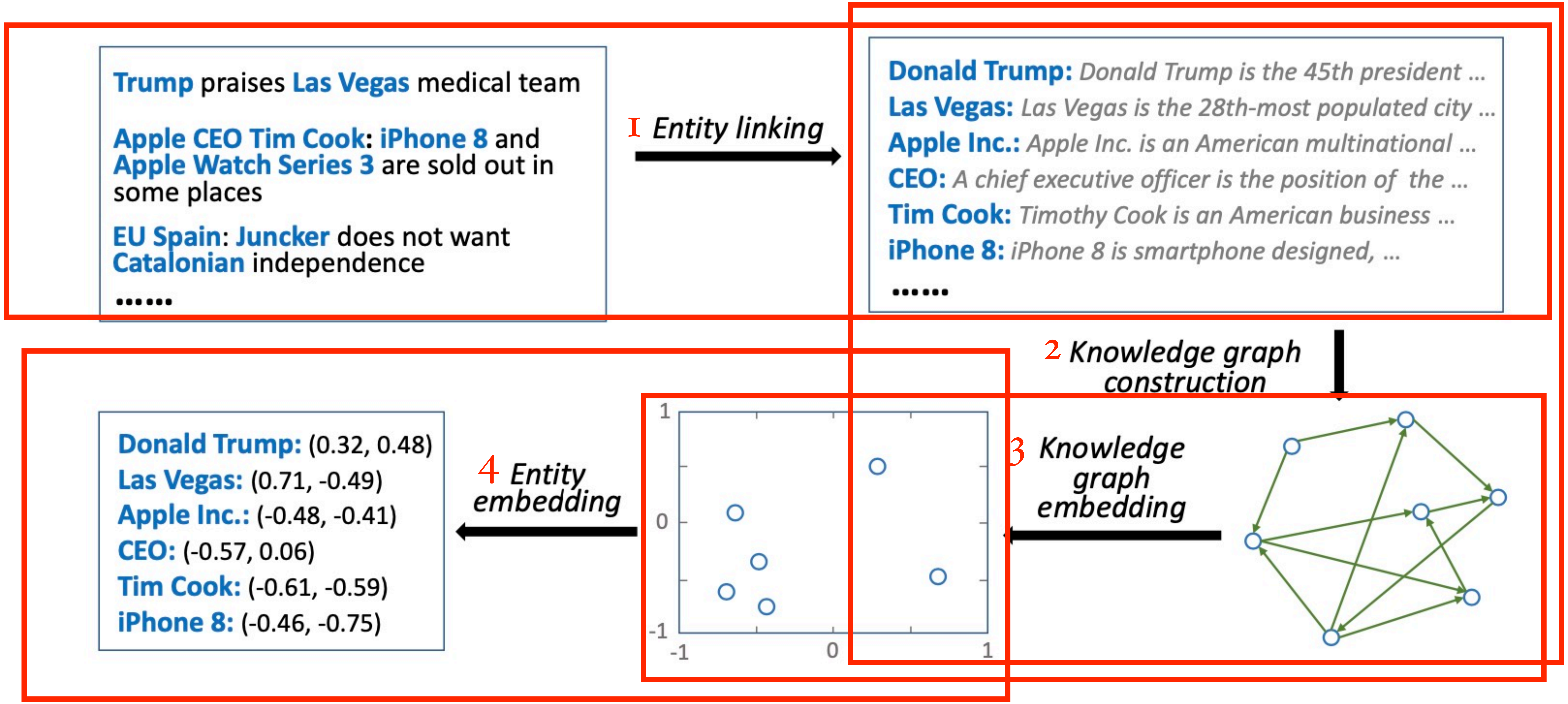
   {Hongwei Wang et al, KDD 2019 }

DKN: Deep Knowledge-Aware Network for News Recommendation

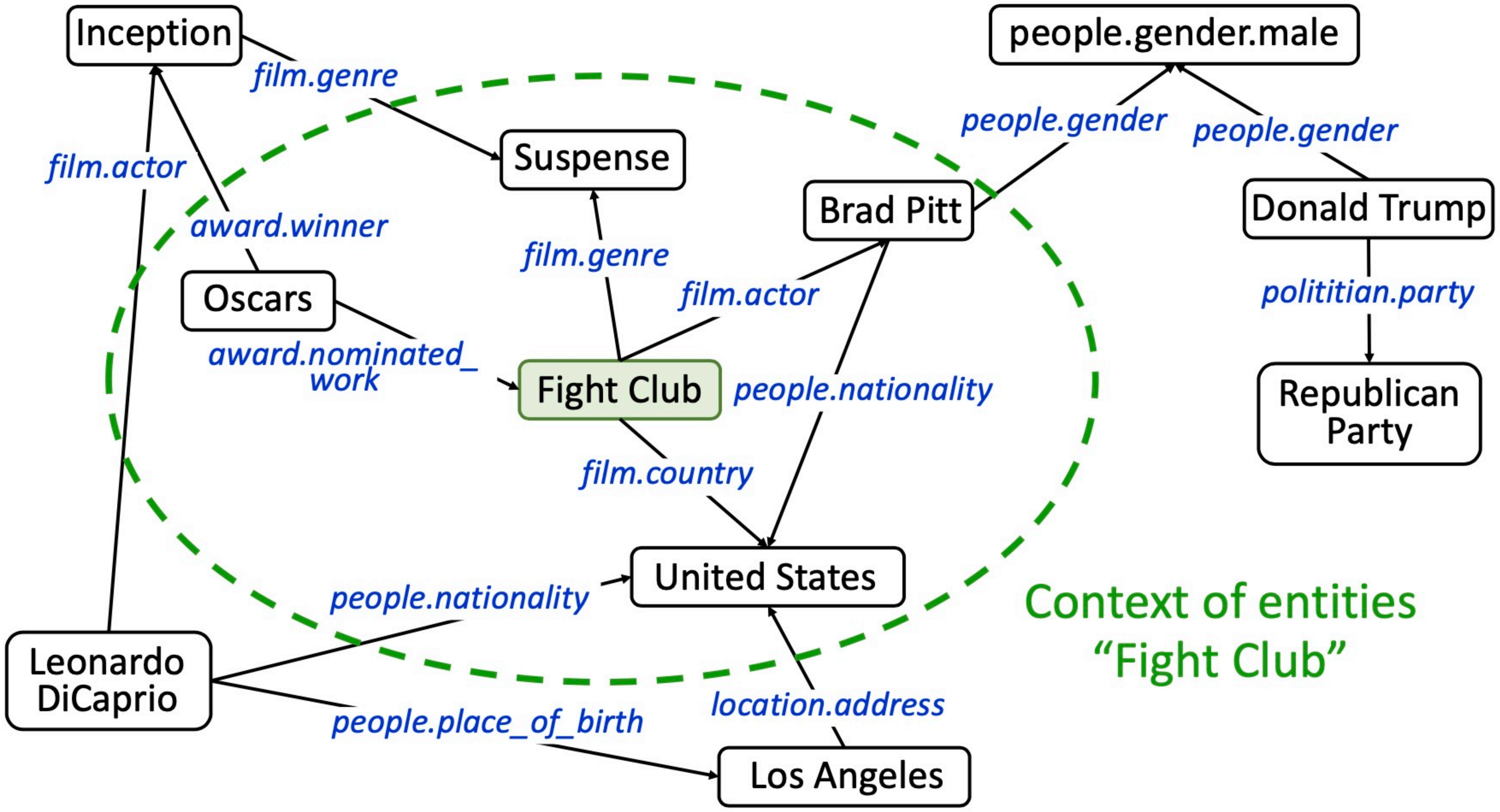
Task

DKN is a content-based model for click-through rate(CTR) prediction, which takes one piece of candidate news and one user's click history as input, and outputs the probability of the user clicking the news.

Knowledge Distillation



Context Embedding



Context Embedding

$$\bar{\mathbf{e}} = \frac{1}{|context(e)|} \sum_{e_i \in context(e)} \mathbf{e}_i,$$

$$context(e) = \{e_i \mid (e, r, e_i) \in \mathcal{G} \text{ or } (e_i, r, e) \in \mathcal{G}\},$$

Knowledge-aware CNN

$$w_{1:n} = [\textit{Donald Trump praises Las Vegas medical team}]$$

$\mathbf{w}_{1:n} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \ldots \ \mathbf{w}_n] \in \mathbb{R}^{d \times n}$ denote the word embedding matrix of the title

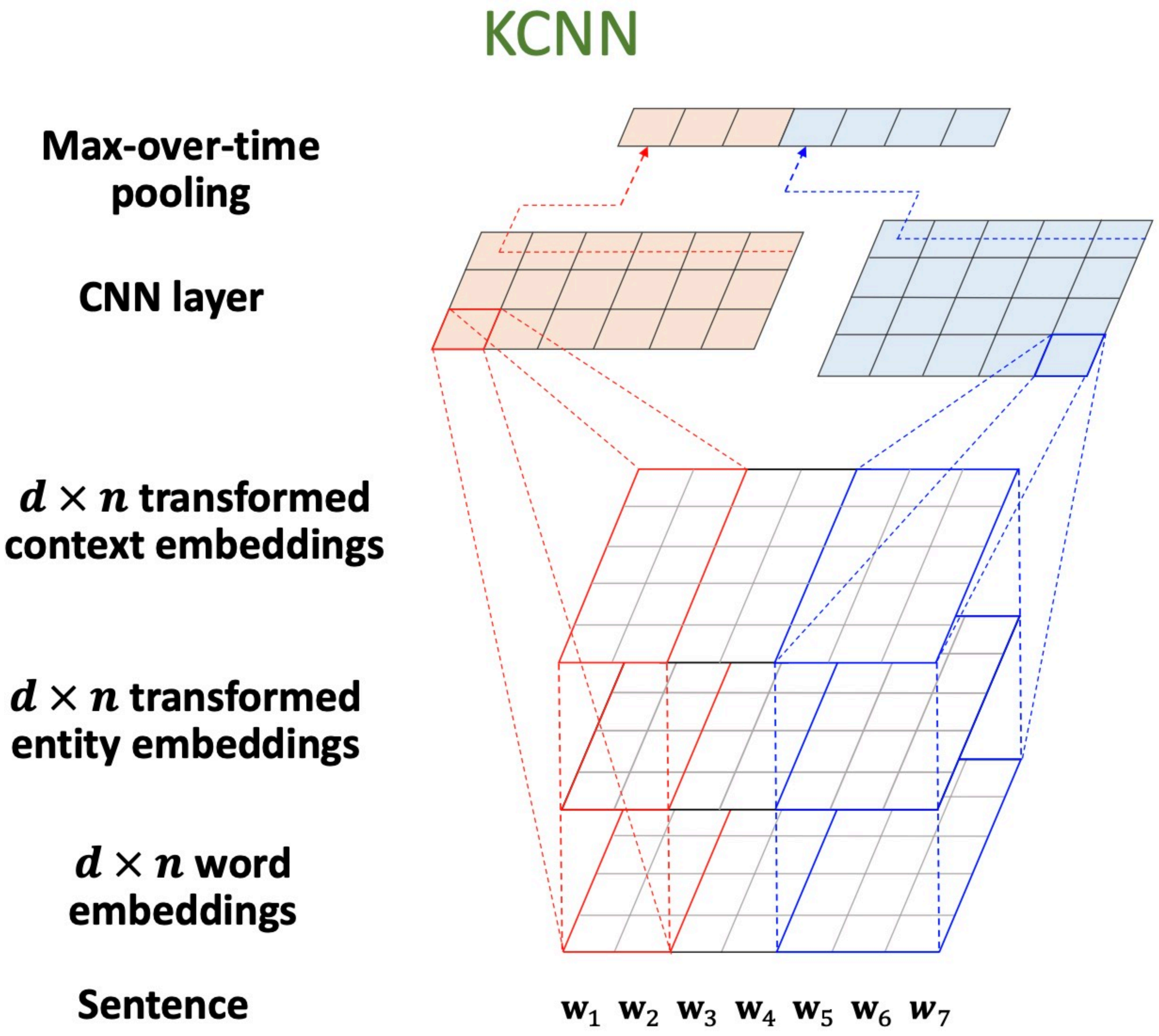
w_i

Entity embedding $e_i \in \mathbb{R}^{k \times 1}$

Context embedding $\bar{e}_i \in \mathbb{R}^{k \times 1}$

Word embedding $\boldsymbol{w}_i$

Knowledge-aware CNN



Linear or non-linear

$$g(\mathbf{e}) = \mathbf{M}\mathbf{e}$$

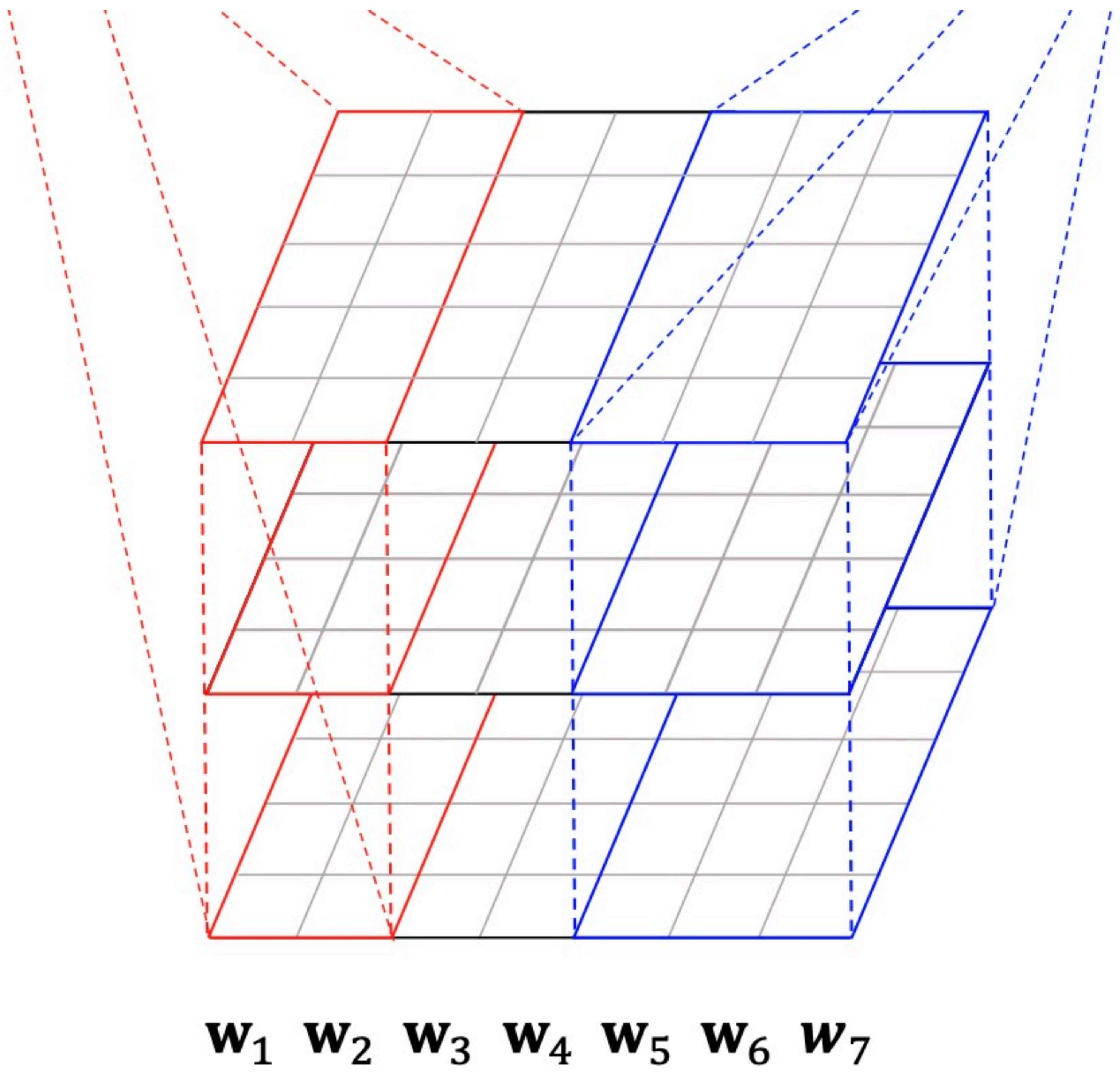
$$g(\mathbf{e}) = \tanh(\mathbf{M}\mathbf{e} + \mathbf{b}),$$

***d* × *n* transformed context embeddings**

***d* × *n* transformed entity embeddings**

***d* × *n* word embeddings**

Sentence



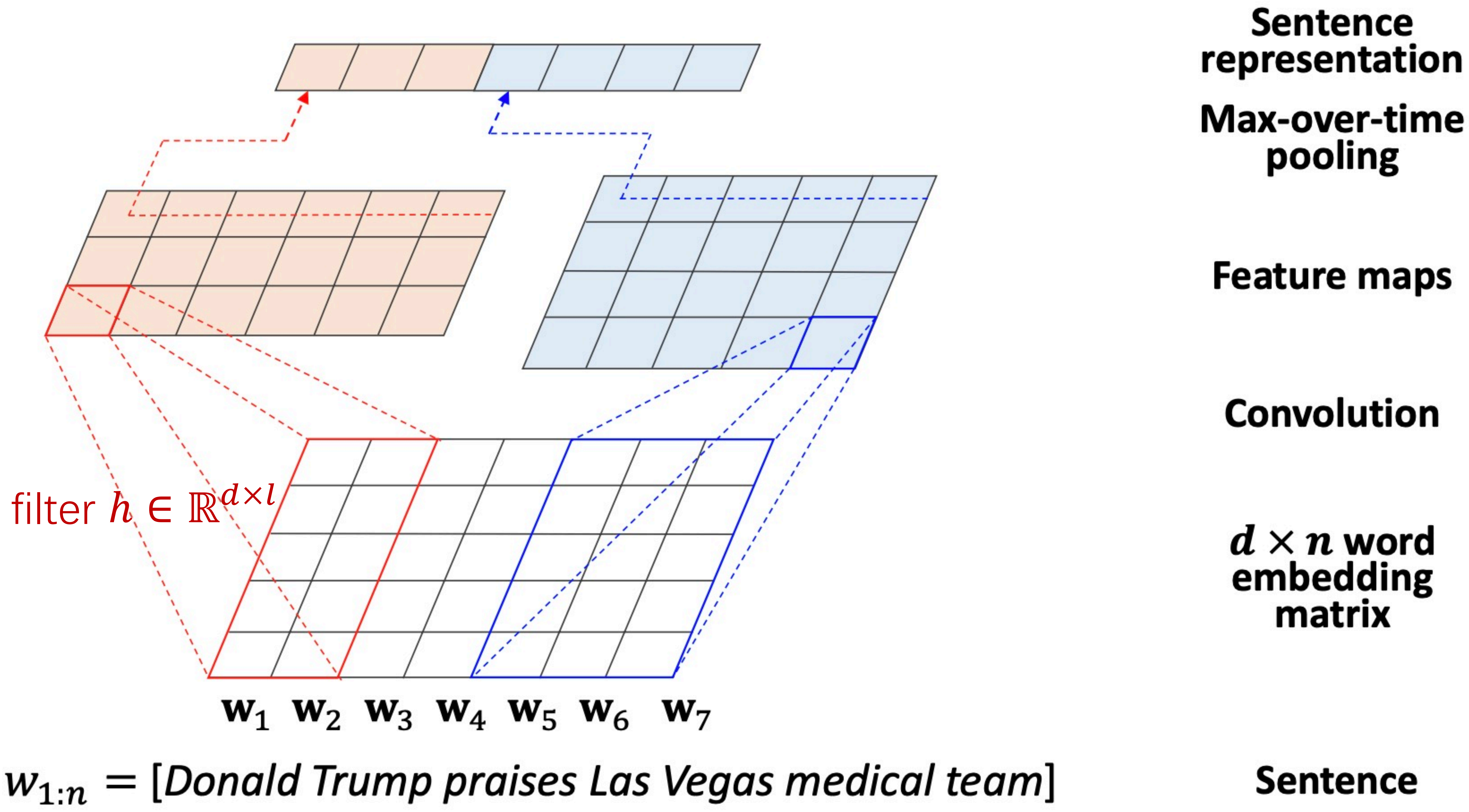
$$g(\bar{\mathbf{e}}_{1:n}) = [g(\bar{\mathbf{e}}_1)\ g(\bar{\mathbf{e}}_2)\ \ldots\ g(\bar{\mathbf{e}}_n)]$$

$$g(\mathbf{e}_{1:n}) = [g(\mathbf{e}_1)\ g(\mathbf{e}_2)\ \ldots\ g(\mathbf{e}_n)]$$

$$\mathbf{w}_{1:n} = [\mathbf{w}_1\ \mathbf{w}_2\ \ldots\ \mathbf{w}_n]$$

$$W = [[w_1 g(\mathrm{e}_1) g(\bar{e}_1)][w_2 g(e_2) g\ (\bar{e}_2)] \ldots [e_n g(e_n) g(\bar{e}_n)]] \in \mathbb{R}^{d\times n\times 3}$$

Kim CNN

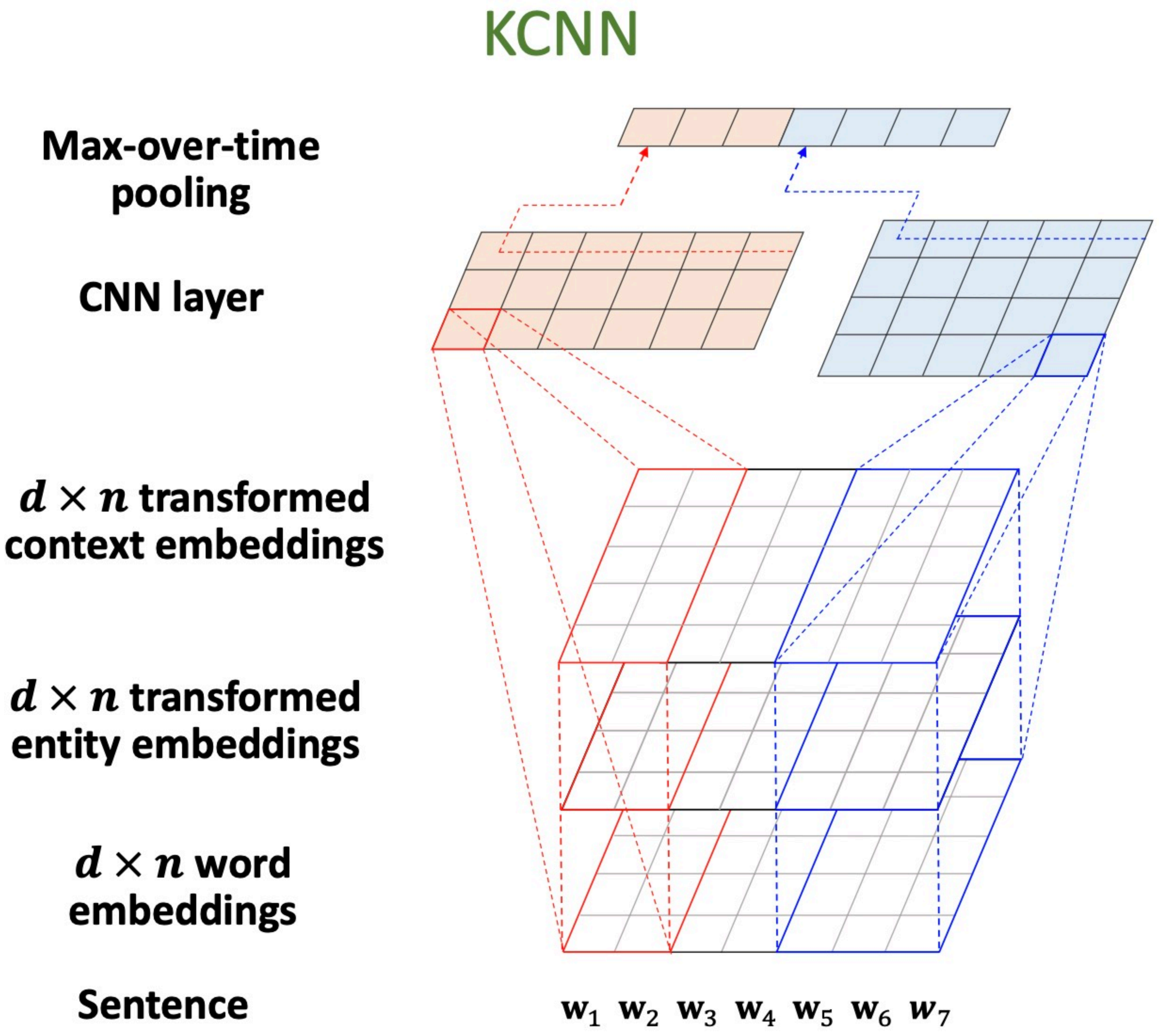


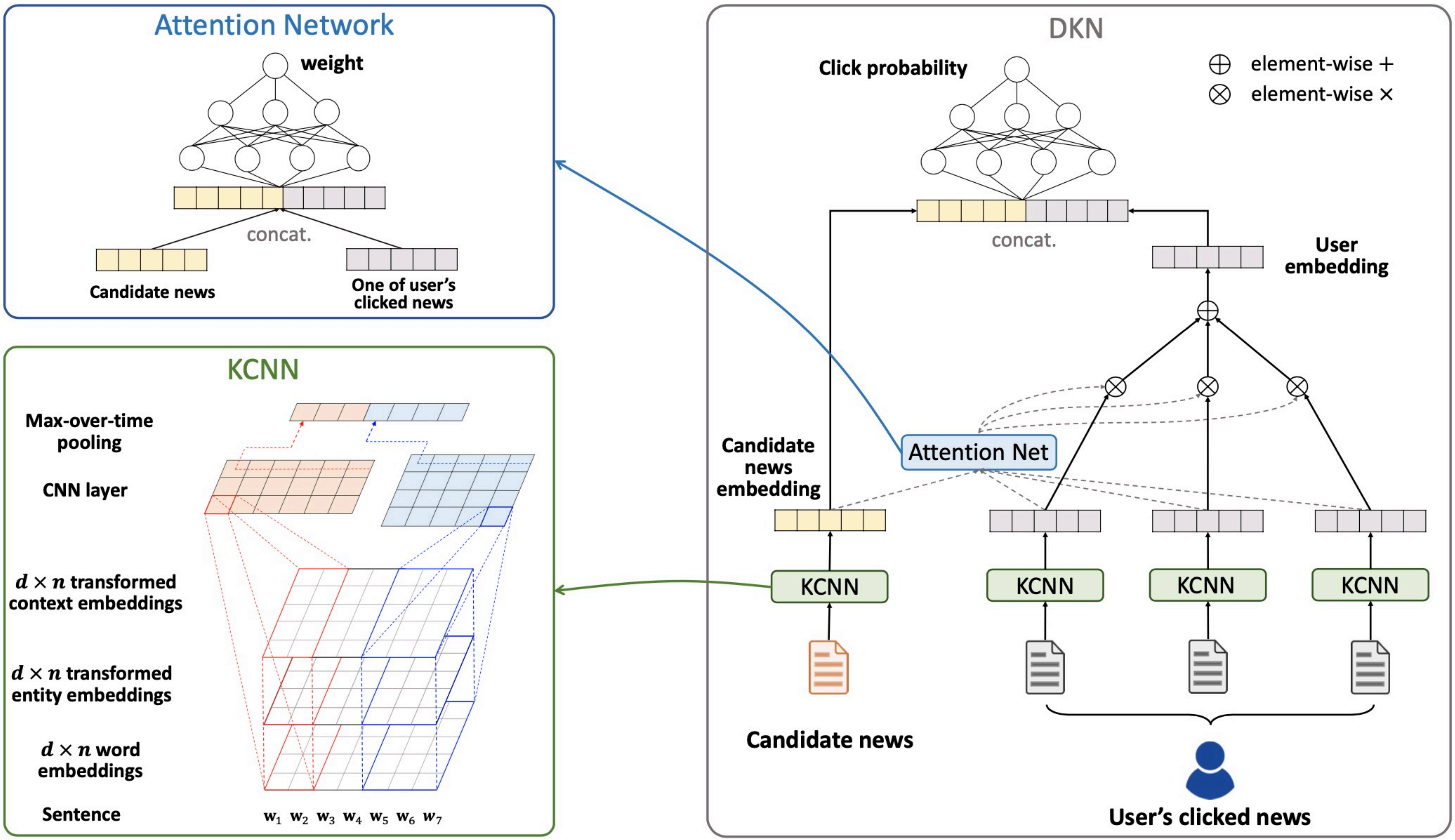
$$c_i^h = f(\boldsymbol{h} * \mathbf{W}_{i:i+l-1} + b),$$

$$\tilde{c}^h = \max\{c_1^h, c_2^h, \ldots, c_{n-l+1}^h\}.$$

$$\mathbf{e}(t) = [\tilde{c}^{h_1} \ \tilde{c}^{h_2} \ \ldots \ \tilde{c}^{h_m}],$$

Knowledge-aware CNN

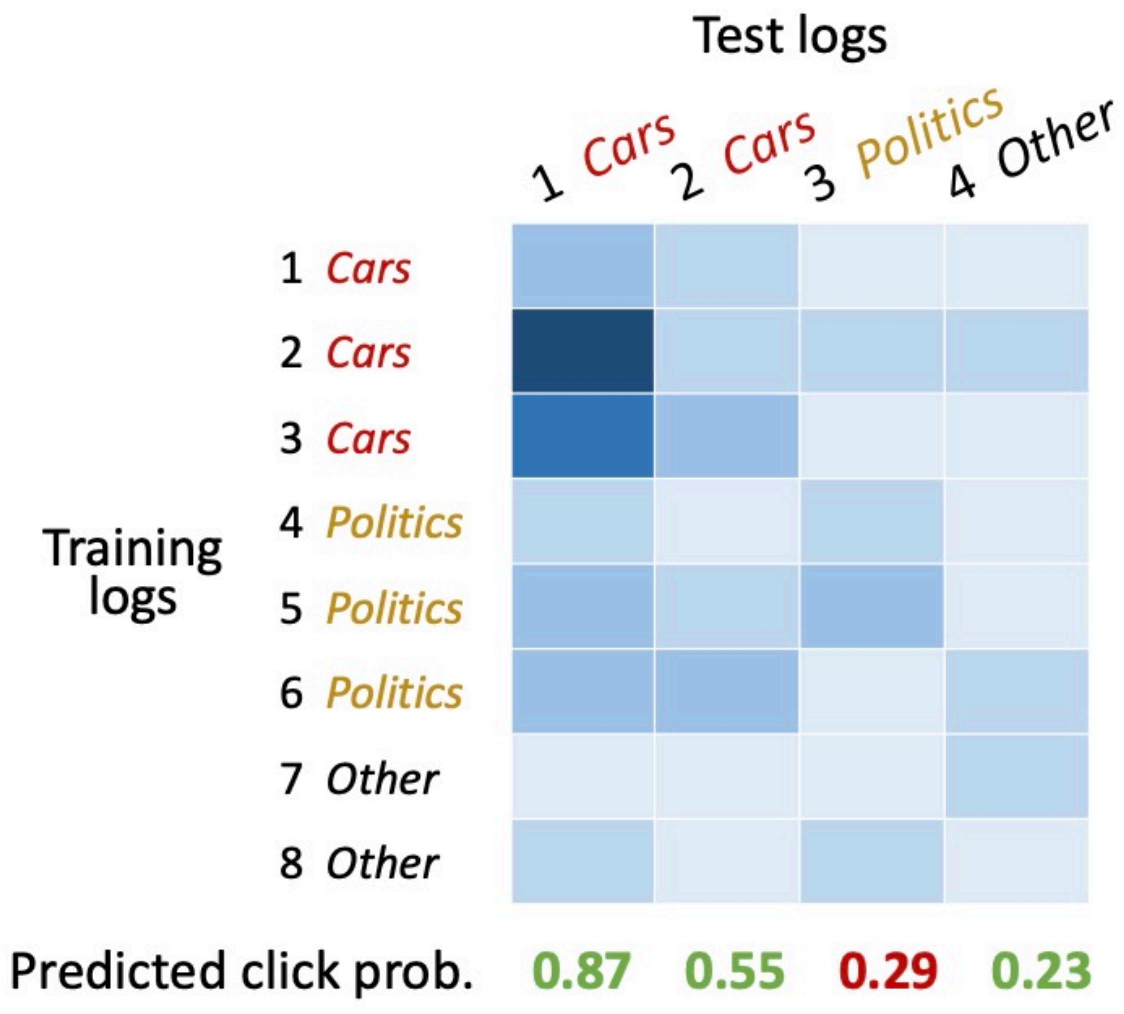




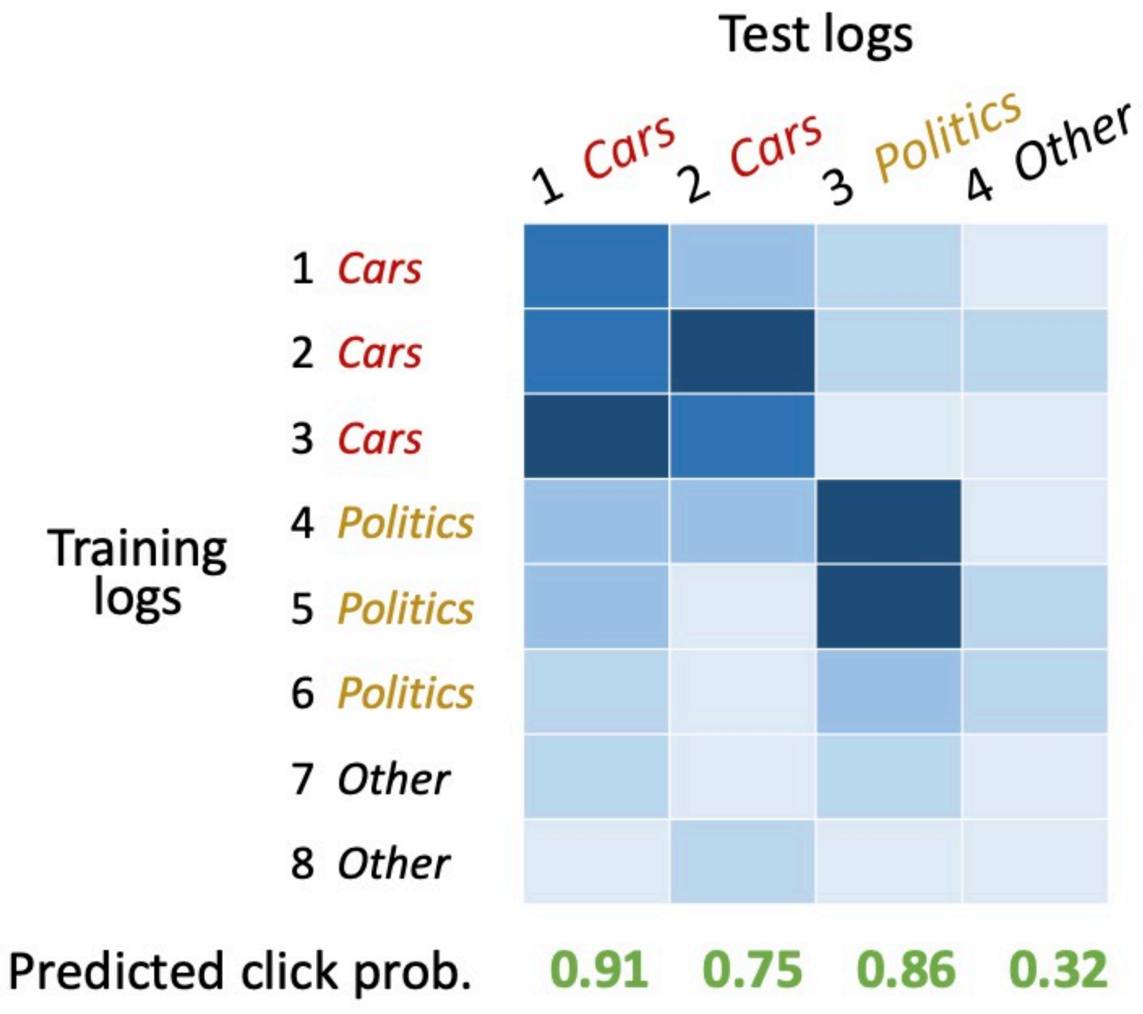
Attention Network
weight
concat.
Candidate news
One of user's clicked news
KCNN
Max-over-time pooling
CNN layer
$d \times n$ transformed context embeddings
$d \times n$ transformed entity embeddings
$d \times n$ word embeddings
Sentence
w_1 w_2 w_3 w_4 w_5 w_6 w_7
DKN
Click probability
element-wise +
element-wise ×
concat.
User embedding
Candidate news embedding
Attention Net
KCNN
KCNN
KCNN
KCNN
Candidate news
User's clicked news

Case Study

	No.	Date	News title	Entities	Label	Category
training	1	12/25/2016	Elon Musk teases huge upgrades for Tesla's supercharger network	Elon Musk; Tesla Inc.	1	Cars
	2	03/25/2017	Elon Musk offers Tesla Model 3 sneak peek	Elon Musk; Tesla Model 3	1	Cars
	3	12/14/2016	Google fumbles while Tesla sprints toward a driverless future	Google Inc.; Tesla Inc.	1	Cars
	4	12/15/2016	Trump pledges aid to Silicon Valley during tech meeting	Donald Trump; Silicon Valley	1	Politics
	5	03/26/2017	Donald Trump is a big reason why the GOP kept the Montana House seat	Donald Trump; GOP; Montana	1	Politics
	6	05/03/2017	North Korea threat: Kim could use nuclear weapons as "blackmail"	North Korea; Kim Jong-un	1	Politics
	7	12/22/2016	Microsoft sells out of unlocked Lumia 950 and Lumia 950 XL in the US	Microsoft; Lumia; United States	1	Other
	8	12/08/2017	6.5 magnitude earthquake recorded off the coast of California	earthquake; California	1	Other
			......			
test	1	07/08/2017	Tesla makes its first Model 3	Tesla Inc; Tesla Model 3	1	Cars
	2	08/13/2017	General Motors is ramping up its self-driving car: Ford should be nervous	General Motors; Ford Inc.	1	Cars
	3	06/21/2017	Jeh Johnson testifies on Russian interference in 2016 election	Jeh Johnson; Russian	1	Politics
	4	07/16/2017	"Game of Thrones" season 7 premiere: how you can watch	Game of Thrones	0	Other



(a) without knowledge graph



(b) with knowledge graph

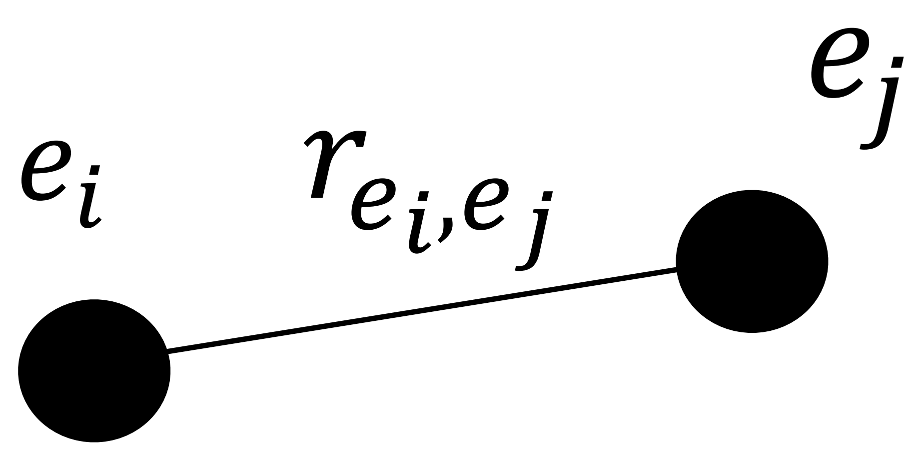
Knowledge Graph Convolutional Networks for Recommender Systems

Task

Given user-item interaction matrix Y, knowledge graph G, our task is to predict whether User u has potential interest in item v with which he/she has not engaged before.

Relation Scoring Function

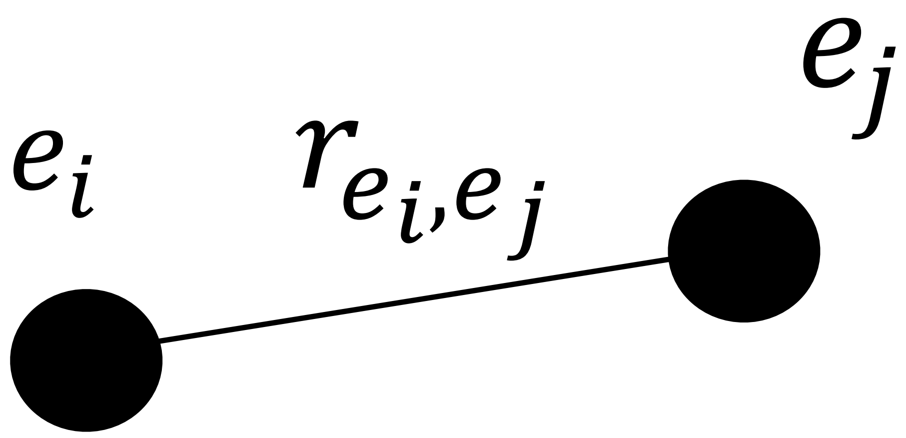
$$S_u(r) = \mathrm{g}(\mathrm{u}, \mathrm{r})$$



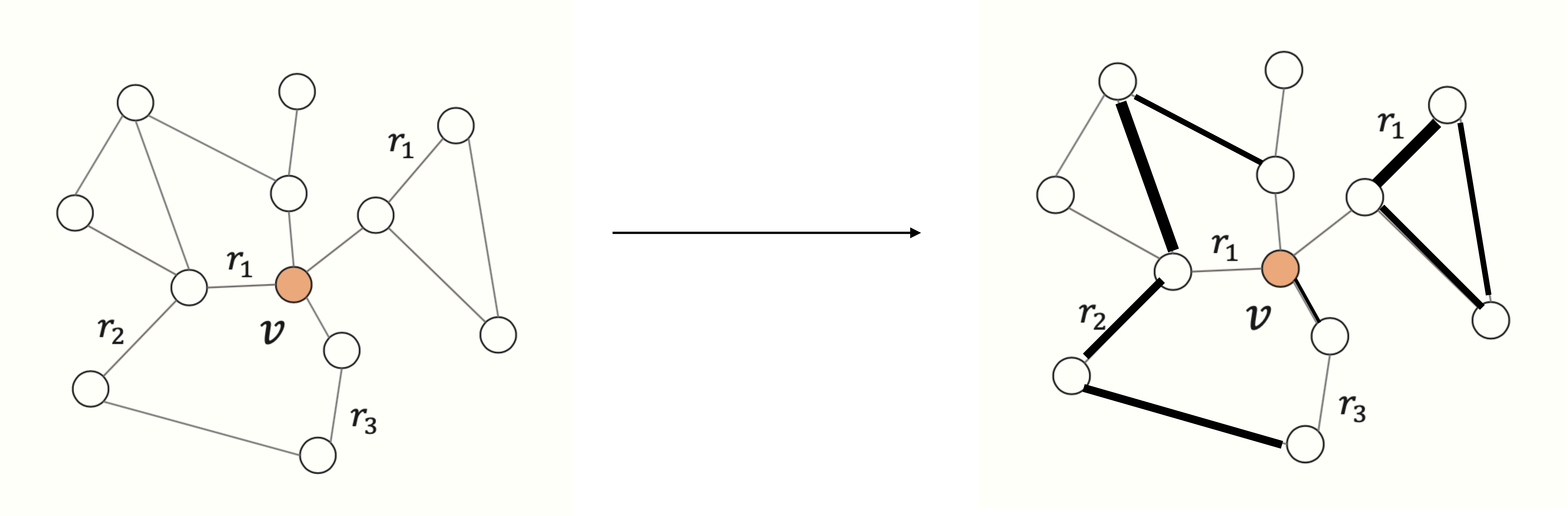
- u: a user, r:a type of relation.
- g is a differentiable function. E.g.: $S_u(r) = \boldsymbol{u}^T \boldsymbol{r}$

Relation Scoring Function

$$S_u(r) = \mathrm{g}(\mathrm{u}, \mathrm{r})$$



- u: a user, r:a type of relation.
- g is a differentiable function. E.g.: $S_u(r) = \boldsymbol{u}^T \boldsymbol{r}$



A_{u}

$A_u^{ij} = s_u(r_{e_i,e_j})$

Knowledge-aware Graph Neural Networks

- Layer-wise forward propagation:

$$H_{l+1} = \sigma\left(D_u^{-1/2} A_u D_u^{-1/2} H_l W_l\right), l = 0,1,\dots,L-1$$

Trainable transformation matrix

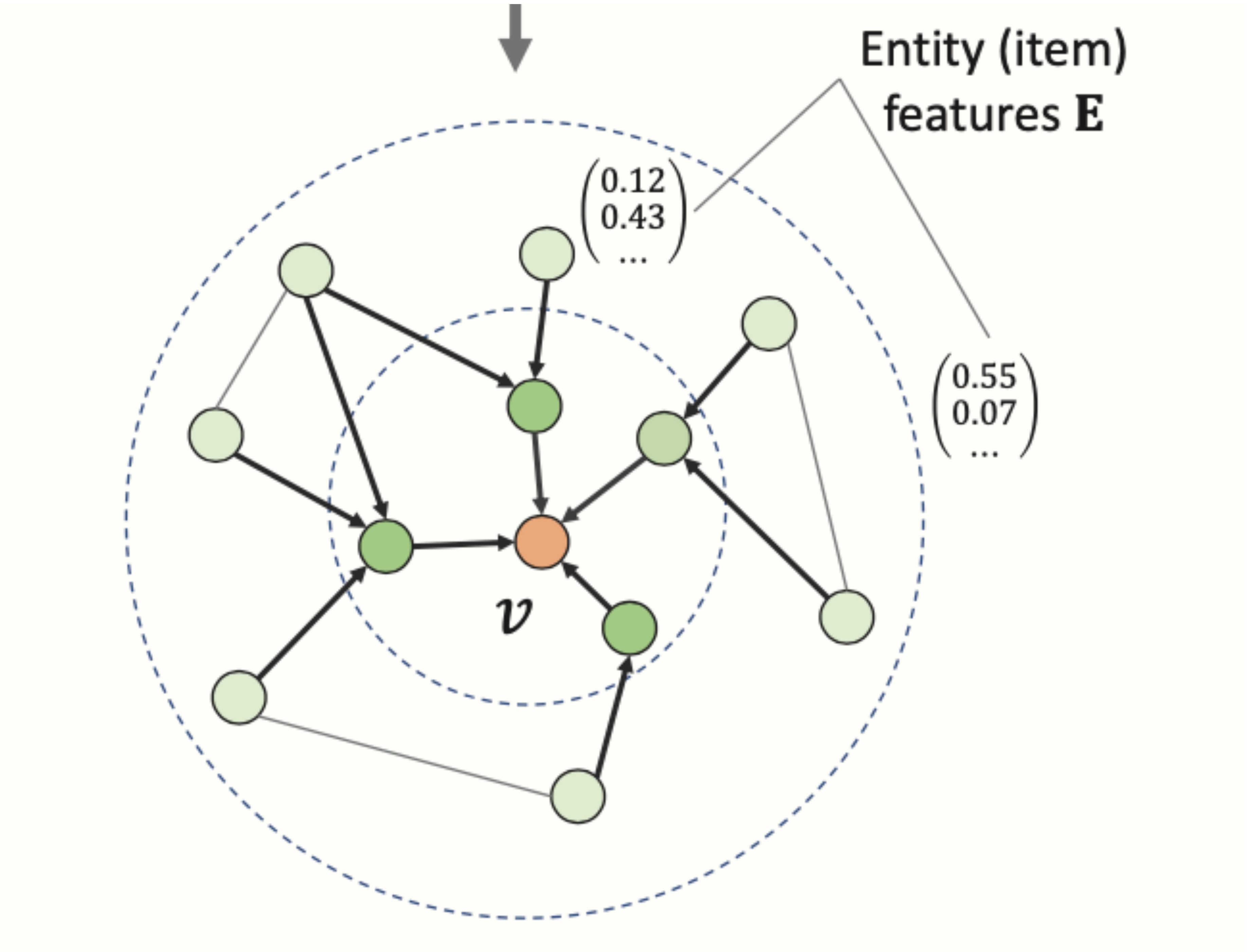
Diagonal degree matrix of Au

Adjacency matrix of the KG for particular user u

Knowledge-aware Graph Neural Networks

- Layer-wise forward propagation:

$$H_{l+1} = \sigma\left(D_u^{-1/2} A_u D_u^{-1/2} H_l W_l\right), l = 0,1, \dots, L-1$$



Predicting Engagement Probability

$$\hat{y}_{uv} = f(\boldsymbol{u}, \boldsymbol{v}_u)$$

- **u**: user embedding
- **V**u: entity(item) embedding from the last KGNN layer

Traditional GNN

Fixed

$$H_{l+1} = \sigma\left(D_u^{-1/2} A_u D_u^{-1/2} H_l W_l\right), l = 0,1, \dots, L-1$$

User Engagement Labels

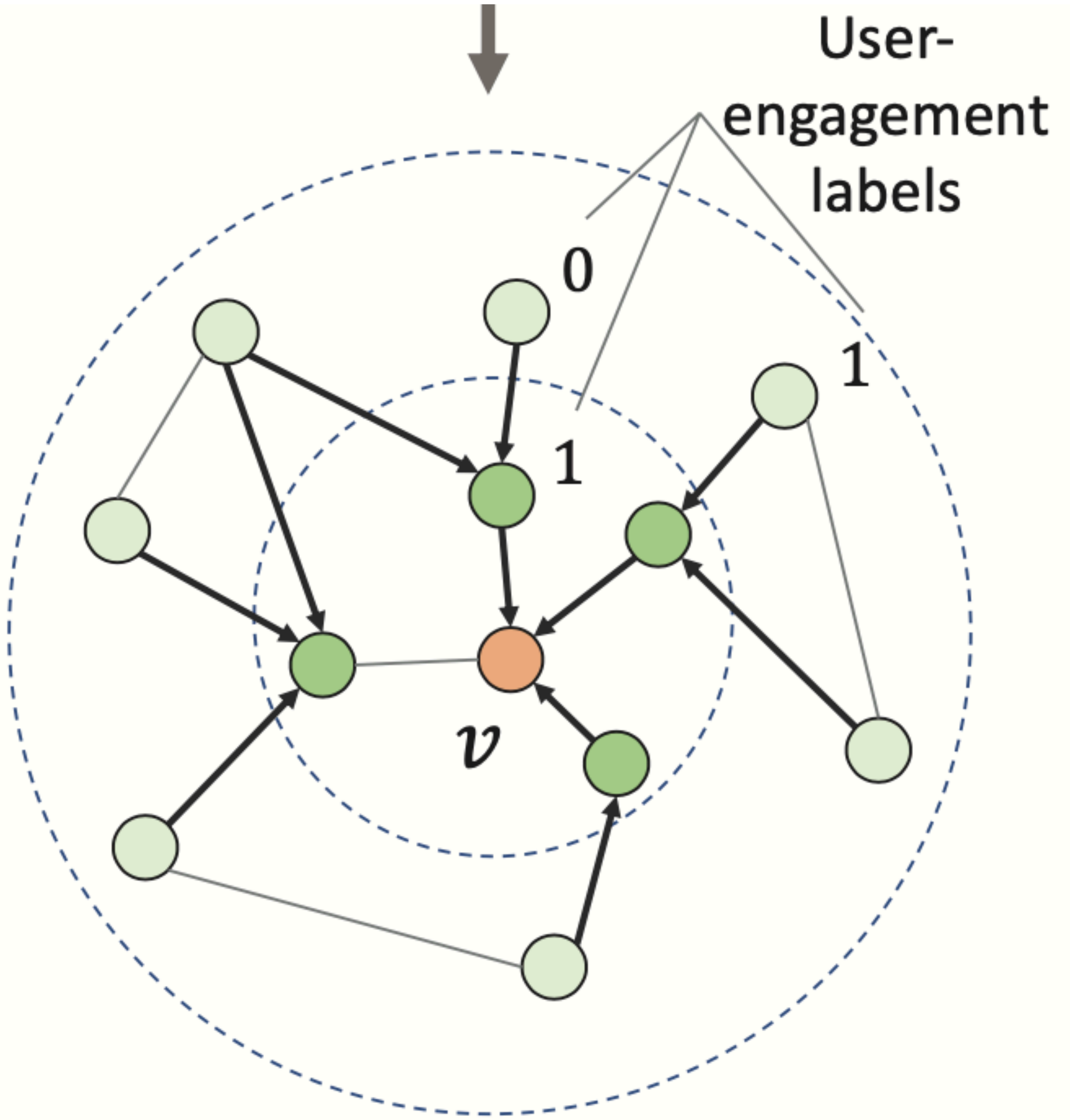
- Positive items: 1
- Negative items: 2
- Non-item entities: unlabeled

Label Smoothness Assumption

$$L = \frac{1}{2} \sum_{i,j \in \varepsilon} A_u[i,j](\bar{y}_{ui} - \bar{y}_{uj})^2$$

- For a given node, take the weighted average of its neighborhood labels as its own label

Label Smoothness Regularization



Label smoothness regularization on KG weights: $R(\mathbf{A}_u)$ (label propagation)

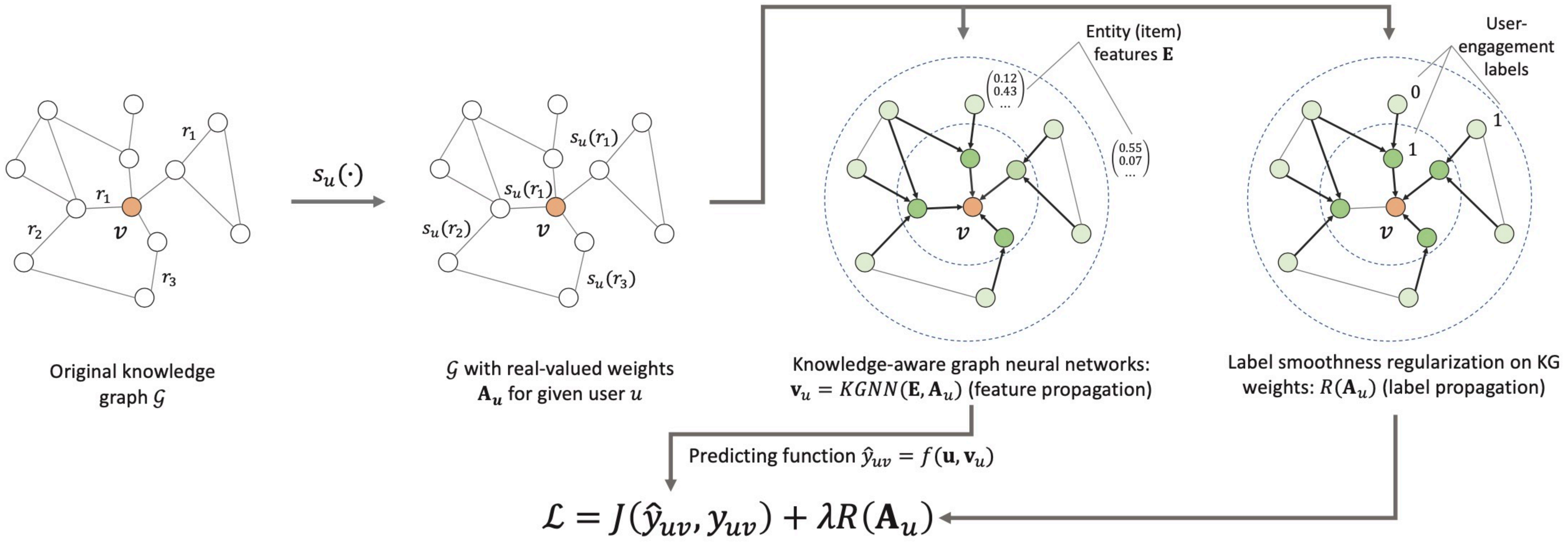
- Predict the label of v by label propagation algorithm

$$\bar{y}_{uv} \Longleftrightarrow y_{uv}$$

- Cross-entropy loss

$$J(y_{uv}, \bar{y}_{uv})$$

$$R(A) = \sum_{\mathrm{u}} R(A_u) = \sum_{u} \sum_{v} J(y_{uv}, \bar{y}_{uv})$$



$s_u(\cdot)$
r_1
r_1
r_2
r_3
v
Original knowledge graph $\mathcal{G}$
$s_u(r_1)$
$s_u(r_1)$
$s_u(r_2)$
$s_u(r_3)$
v
$\mathcal{G}$ with real-valued weights $\mathbf{A}_u$ for given user u
Entity (item) features $\mathbf{E}$
$\begin{pmatrix} 0.12 \\ 0.43 \\ \dots \end{pmatrix}$
$\begin{pmatrix} 0.55 \\ 0.07 \\ \dots \end{pmatrix}$
v
Knowledge-aware graph neural networks: $\mathbf{v}_u = KGNN(\mathbf{E}, \mathbf{A}_u)$ (feature propagation)
User-engagement labels
0
1
1
v
Label smoothness regularization on KG weights: $R(\mathbf{A}_u)$ (label propagation)
Predicting function $\hat{y}_{uv} = f(\mathbf{u}, \mathbf{v}_u)$
$\mathcal{L} = J(\hat{y}_{uv}, y_{uv}) + \lambda R(\mathbf{A}_u)$

Experience

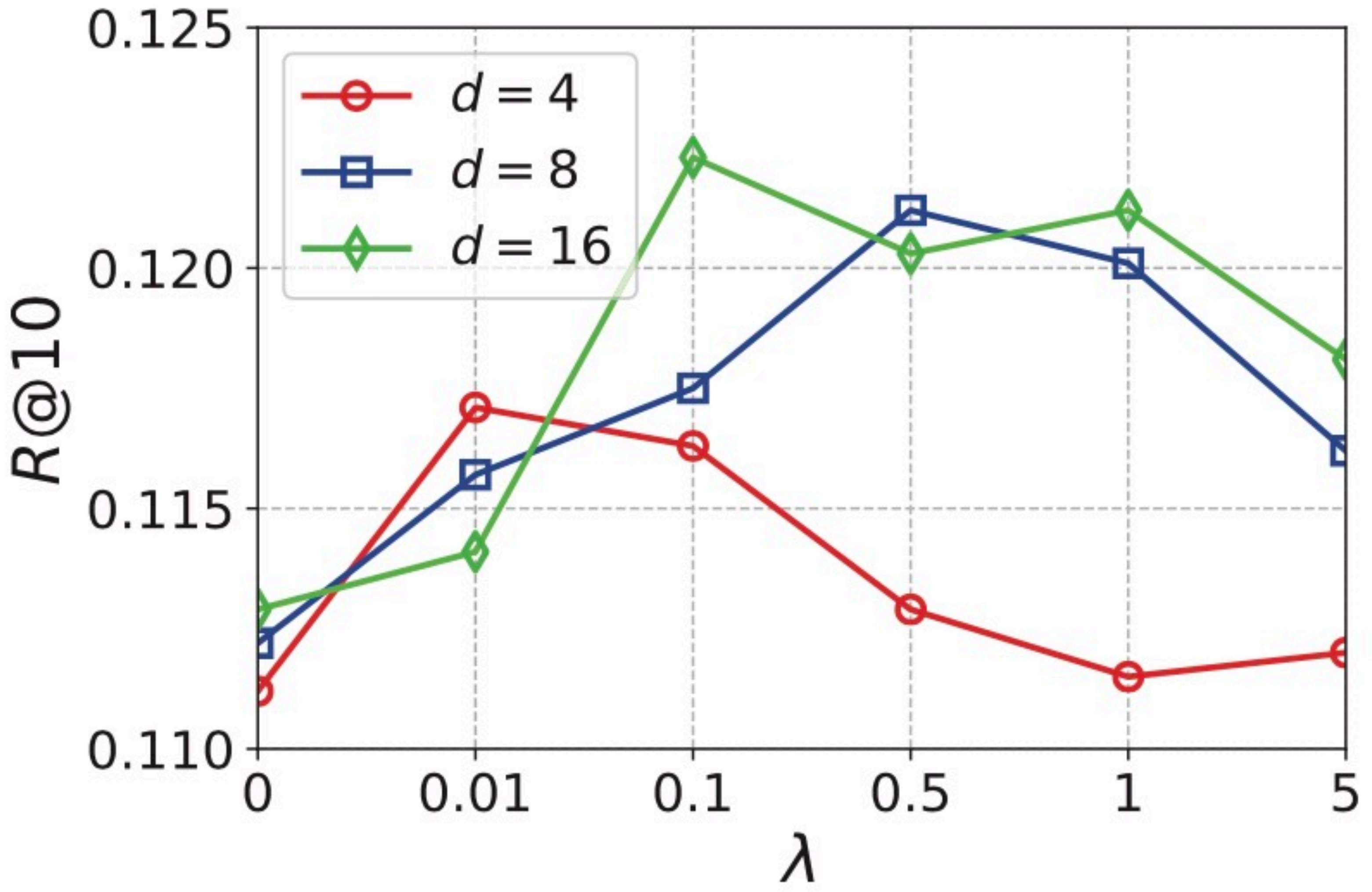
Model	MovieLens-20M				Book-Crossing				Last.FM				Dianping-Food			
	R@2	*R@10*	*R@50*	*R@100*	*R@2*	*R@10*	*R@50*	*R@100*	*R@2*	*R@10*	*R@50*	*R@100*	*R@2*	*R@10*	*R@50*	*R@100*
SVD	0.036	0.124	0.277	0.401	0.027	0.046	0.077	0.109	0.029	0.098	0.240	0.332	0.039	0.152	0.329	0.451
LibFM	0.039	0.121	0.271	0.388	0.033	0.062	0.092	0.124	0.030	0.103	0.263	0.330	0.043	0.156	0.332	0.448
LibFM + TransE	0.041	0.125	0.280	0.396	0.037	0.064	0.097	0.130	0.032	0.102	0.259	0.326	0.044	0.161	**0.343**	0.455
PER	0.022	0.077	0.160	0.243	0.022	0.041	0.064	0.070	0.014	0.052	0.116	0.176	0.023	0.102	0.256	0.354
CKE	0.034	0.107	0.244	0.322	0.028	0.051	0.079	0.112	0.023	0.070	0.180	0.296	0.034	0.138	0.305	0.437
RippleNet	**0.045**	0.130	0.278	0.447	0.036	0.074	0.107	0.127	0.032	0.101	0.242	0.336	0.040	0.155	0.328	0.440
KGNN-LS	0.043	**0.155**	**0.321**	**0.458**	**0.045**	**0.082**	**0.117**	**0.149**	**0.044**	**0.122**	**0.277**	**0.370**	**0.047**	**0.170**	0.340	**0.487**

Table 3: The results of *Recall@K* in top-K recommendation.

Model	Movie	Book	Music	Restaurant
SVD	0.963	0.672	0.769	0.838
LibFM	0.959	0.691	0.778	0.837
LibFM + TransE	0.966	0.698	0.777	0.839
PER	0.832	0.617	0.633	0.746
CKE	0.924	0.677	0.744	0.802
RippleNet	0.960	0.727	0.770	0.833
KGNN-LS	**0.979**	**0.744**	**0.803**	**0.850**

Table 4: The results of *AUC* in CTR prediction.

Experience



r	20%	40%	60%	80%	100%
SVD	0.882	0.913	0.938	0.955	0.963
LibFM	0.902	0.923	0.938	0.950	0.959
LibFM+TransE	0.914	0.935	0.949	0.960	0.966
PER	0.802	0.814	0.821	0.828	0.832
CKE	0.898	0.910	0.916	0.921	0.924
RippleNet	0.921	0.937	0.947	0.955	0.960
KGNN-LS	**0.961**	**0.970**	**0.974**	**0.977**	**0.979**

Table 5: *AUC* of all methods w.r.t. the ratio of training set *r*.