Rethinking Graph Convolutional Networks in Knowledge Graph Completion

Zhanqiu Zhang[1], Jie Wang[1,2,*], Jieping Ye[3,4], Feng Wu[1]
[1]CAS Key Laboratory of Technology in GIPAS
University of Science and Technology of China
[2]Institute of Artificial Intelligence
Hefei Comprehensive National Science Center
zqzhang@mail.ustc.edu.cn,{jiewangx,fengwu}@ustc.edu.cn
[3]University of Michigan
[4]KE Holdings Inc.
jpye@umich.edu

Haobo Xu

2022/10/26

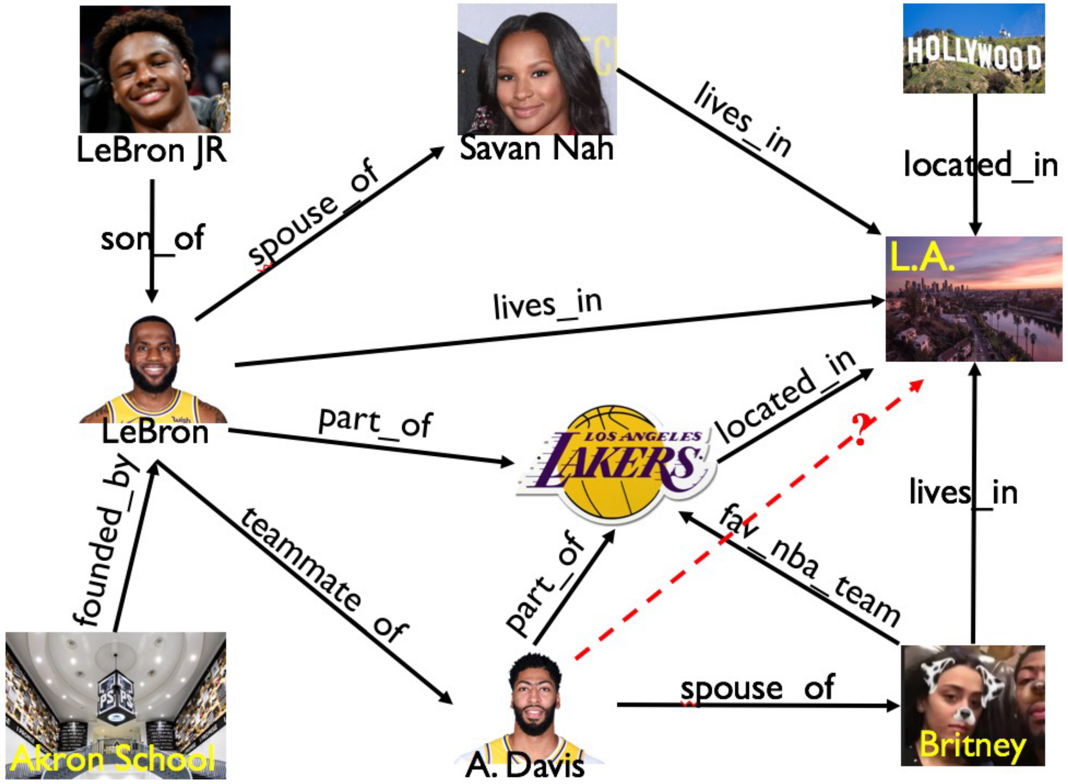
Outlines

- Background
  - ➢ KGC
  - ➢ GCN-based KGC
- Rethinking
  - ➢ Do GCNs work in KGC?
  - ➢ Which factor works?
- Framework & Experiments
- Conclusion
- Discussion

Zhang, Z., Wang, J., Ye, J., & Wu, F. Rethinking Graph Convolutional Networks in Knowledge Graph Completion. WWW *2022*.

Background

- Knowledge Graph Completion
  - Methods
    - KGE
    - GNN for KG
    - PLM
    - …
  - Subtasks
    - Link prediction, relation prediction, triple classification



Background

- Motivation for GCN-based KGC methods
  - Inspired by GNN message passing

• Message function:

$$\boldsymbol{m}_{(u,v)}^{k} = \phi(\boldsymbol{h}_u^k, \boldsymbol{h}_v^k)$$

• Aggregate function:

$$\bar{\boldsymbol{m}}_v^k = Agg^k(\{\boldsymbol{m}_{(u,v)}^k : u \in \mathcal{N}(v)\})$$

• Update function:

$$\boldsymbol{h}_v^{k+1} = Comb^k(\boldsymbol{h}_v^k, \bar{\boldsymbol{m}}_v^k)$$

  - Aggregate neighbor & relation information for KGC

• Message function:

$$\boldsymbol{m}_{(u,r,v)}^{k} = \phi(\boldsymbol{h}_u^k, \boldsymbol{h}_r^k, \boldsymbol{h}_v^k)$$

• Aggregate function

$$\bar{\boldsymbol{m}}_v^k = Agg^k(\{\boldsymbol{m}_{(u,r,v)}^k : (u,r) \in \mathcal{N}(v)\})$$

• Update function

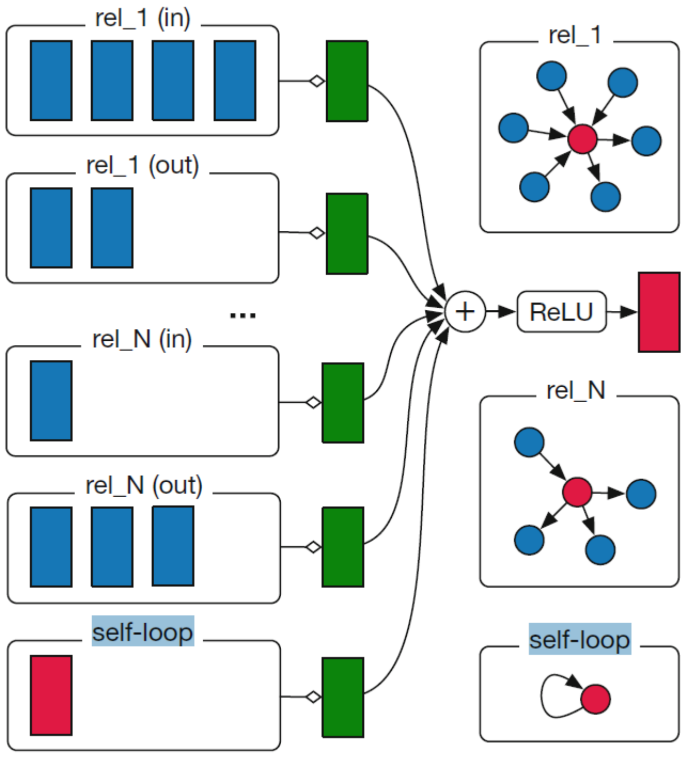
$$\boldsymbol{h}_v^{k+1} = Comb^k(\boldsymbol{h}_v^k, \bar{\boldsymbol{m}}_v^k)$$
$$\boldsymbol{h}_r^{k+1} = \varphi(\boldsymbol{h}_r^k)$$

Background

- RGCN
  - Message function $$\boldsymbol{m}^{k}_{(u,r,v)} = W_r^{k} \boldsymbol{h}_u^{k}$$
  - Aggregate & Update function

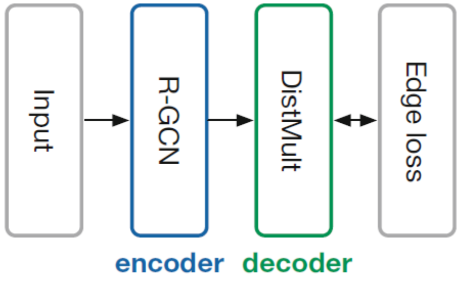
$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right)$$

Including **inverse edges** and **self loop**



(a) Single R-GCN layer

- Subtask: link prediction
  - R-GCN: encoder
  - KGE model: decoder



(c) Link prediction model

R-GCN: Modeling Relational Data with Graph Convolutional Networks. ESWC 2018.

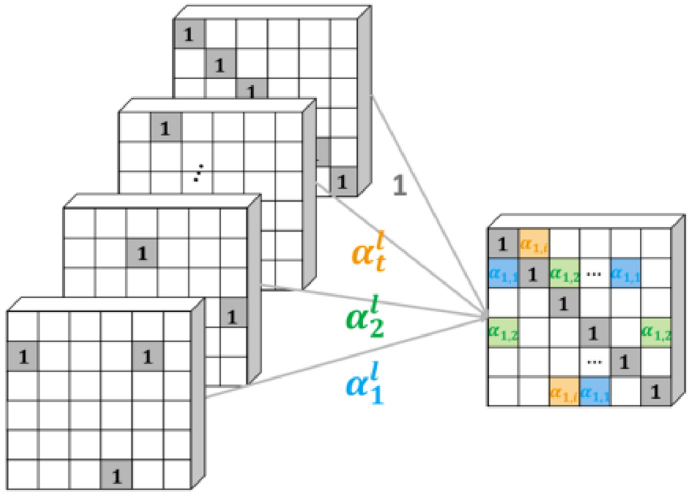
Background

- WGCN
  - Motivation:

    RGCN over-parameterization for relation representation
  - Methods: $\bar{\boldsymbol{m}}_v^k = \sigma\left(\sum_{(u,r)\in\mathcal{N}(v)} \alpha_r^k W^k \boldsymbol{h}_s^k\right).$

    Multi-relational Graph

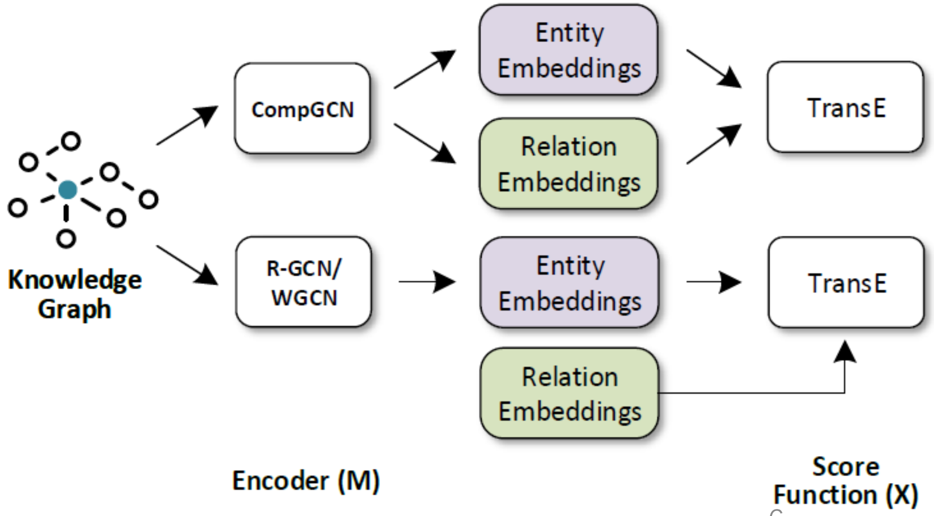
    → weighted multiple single-relational subgraphs



- CompGCN
  - Methods:

    Message function $\phi(\boldsymbol{h}_u^k, \boldsymbol{h}_r^k, \boldsymbol{h}_v^k)$: a composition operator
    - Subtraction (TransE): $\phi(\cdot) = W_{dir(r)}^{(k)}(\boldsymbol{h}_u^k - \boldsymbol{h}_r^k)$;
    - Multiplication (DistMult): $\phi(\cdot) = W_{dir(r)}^{(k)}(\boldsymbol{h}_u^k * \boldsymbol{h}_r^k)$;
    - Circular-correlation (HolE): $\phi(\cdot) = W_{dir(r)}^{(k)}(\boldsymbol{h}_u^k \star \boldsymbol{h}_r^k)$.

    Relation update function: $\boldsymbol{h}_r^{k+1} = W_{rel}\boldsymbol{h}_r^k$.



WGCN: End-to-end Structure-Aware Convolutional Networks for Knowledge Base Completion. AAAI 2019.
CompGCN: Composition-based Multi-Relational Graph Convolutional Networks. ICLR 2020.

Rethinking

- Overview
  - Q1: Do GCNs Really Bring Performance Gain?
    - Conclusion: GCNs do bring performance gain.
  - Q2: Which Factor of GCNs is Critical in KGC?
    - Conclusion:

      GCNs based methods have three main parts
    - 1) graph structures -> not necessary
    - 2) transformations for relation -> not necessary
    - 3) aggregated entity representation -> necessary

Neighborhood Information

Self-loop Information

Have the same effect.
Can substitute for each other.

Rethinking

- Q1: Do GCNs Really Bring Performance Gain?
  - Fact

    GCNs utilize training strategies, while original KGE models not, which may be unfair

  - Experiment

O: original papers, R: reproduce; T: TransE, D: DistMult, C: ConvE

	FB237			WN18RR		
	T	D	C	T	D	C
w/o GCN (R)	.332	.279	.319	.205	.410	.462
RGCN (R)	.324	.332	.337	-	-	-
WGCN (R)	.272	.329	.340	.222	.422	.462
CompGCN (R)	.335	.342	.353	.206	.430	.469

  - Conclusion: GCNs do bring performance gain.

Rethinking

- Q2: Which Factor of GCNs is Critical in KGC?
  - Consensus: Main advantage of GCNs is ability to model graph structure
  - Review of GCNs for KGC

	Entity Aggregation	Entity Update	Relation
RGCN [24]	$\mathbf{m}_e^{l+1} = \sum_{(h,r)\in\mathcal{N}_{in}(e)} \mathbf{W}_r^l \mathbf{h}^l + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \mathbf{W}_r^l \mathbf{t}^l + \mathbf{W}_0^l \mathbf{e}^l$	$\mathbf{e}^{l+1} = \sigma(\mathbf{m}_e^{l+1})$	$\mathbf{r}^{l+1} = \mathbf{r}^l$
WGCN [25]	$\mathbf{m}_e^{l+1} = \sum_{(h,r)\in\mathcal{N}_{in}(e)} \mathbf{W}^l (\alpha_r^l \mathbf{h}^l) + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \mathbf{W}^l (\alpha_r^l \mathbf{t}^l) + \mathbf{W}_0^l \mathbf{e}^l$	$\mathbf{e}^{l+1} = \sigma(\mathbf{m}_e^{l+1})$	$\mathbf{r}^{l+1} = \mathbf{r}^l$
CompGCN [31]	$\mathbf{m}_e^{l+1} = \sum_{(h,r)\in\mathcal{N}_{in}(e)} \mathbf{W}_r^l \phi_{in}(\mathbf{h}^l, \mathbf{r}^l) + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \mathbf{W}_r^l \phi_{out}(\mathbf{t}^l, \mathbf{r}^l) + \mathbf{W}_0^l \mathbf{e}^l$	$\mathbf{e}^{l+1} = \sigma(\mathbf{m}_e^{l+1})$	$\mathbf{r}^{l+1} = \mathbf{W}_{rel}^l \mathbf{r}^l$

Graph Structure?

Self-loop Information?

Neighborhood Information?

Relations Update?

Rethinking

- Graph Structure Ablation Experiment
  - Break graph structure
  - Use the random adjacency tensors in message passing
- Neighbor Information Ablation Experiment
  - Remove neighbor information term in aggregation function

Table 4: MRR for GCNs with random adjacency tensors (RAT) and without neighbor information (WNI).

	FB237			WN18RR		
	T	D	C	T	D	C
RGCN	.324	.332	.337	-	-	-
RGCN + RAT	.322	.331	.333	-	-	-
RGCN + WNI	.324	.332	.335	-	-	-
WGCN	.272	.329	.340	.222	.422	.462
WGCN + RAT	.325	.330	.354	.213	.426	.470
WGCN + WNI	.337	.334	.355	.235	.425	.473
CompGCN	.335	.342	.353	.206	.430	.469
CompGCN + RAT	.336	.336	.351	.208	.420	.467
CompGCN + WNI	.339	.335	.352	.214	.435	.465

Conclusion: Neither graph structure nor neighbor information is critical.

Rethinking

- Self Loop Ablation Experiment
  - Remove self loop term in aggregation function
  - Result: (X+WSI) Keeping only neighbor information achieves comparative results

Conclusion: Self Loop Information is not critical.

- Self Loop + Random Neighbors Experiment
  - Result: (X+WSI+RAT) Only aggregating randomly generated neighbor information achieve comparative results

Conclusion: Keeping only randomly selected neighbor information still works.

Table 5: MRR results for GCNs without self-loop information (WSI) and random adjacency tensors (RAT).

	FB237			WN18RR		
	T	D	C	T	D	C
RGCN	.324	.332	.337	-	-	-
RGCN + WSI	.323	.337	.339	-	-	-
RGCN + WSI+RAT	.320	.317	.318	-	-	-
WGCN	.272	.329	.340	.222	.422	.462
WGCN + WSI	.263	.330	.341	.181	.408	.426
WGCN + WSI + RAT	.322	.319	.340	.168	.353	.408
CompGCN	.335	.342	.353	.206	.430	.469
CompGCN + WSI	.320	.338	.352	.181	.408	.426
CompGCN + WSI + RAT	.317	.297	.342	.168	.353	.408
w/o GCN (R)	.332	.279	.319	.205	.410	.462

Rethinking

- Guesswork:

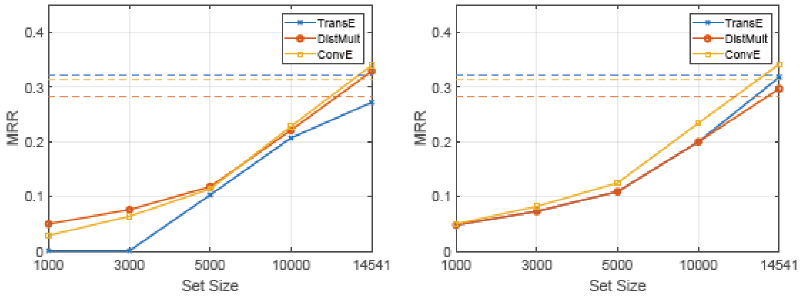
  The followings contain semantical information, which can distinguish entities from others

  - 1) self-loop
    - Reason: The representation of each entity is independent.
  - 2) neighbor information
    - Reason: Entities with different neighbors have different semantics.
  - 3) randomly generated neighbor
    - Reason: Assign different neighbors for different entities with a high possibility, thus different entities have different sets of neighbors (just like a hash number).

Any of the above (that can distinguish entities) has the same effect on the final performance.

Rethinking

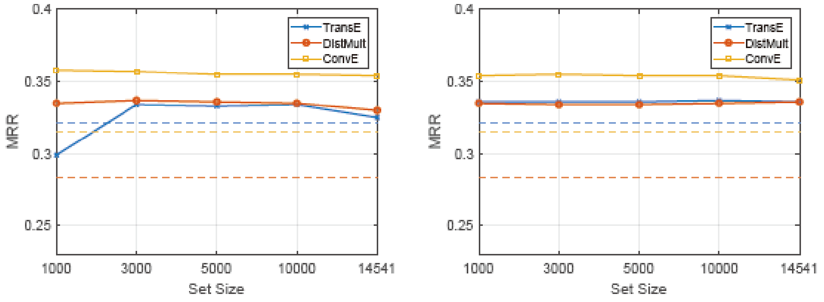
- Experiment:
  - Methods: Randomly sample neighbors in a given set.
  - Expectation: When <u>set size becomes smaller</u>, entities get harder to distinguish themselves from others, thus <u>the performance get worse</u>



(a) WGCN+X (b) CompGCN+X

Figure 2: MRR results for no self-loop information and random sampled neighbor entities on FB237

Without self-loop + Random sampled neighbors (agree with expectation)



(a) WGCN+X (b) CompGCN+X

Figure 3: MRR for random sampled neighbors on FB237, where the dash lines indicate performance without GCNs.

With self-loop + Random sampled neighbors (disagree with expectation)

Conclusion: Self-loop / neighbor Info both can distinguish entities to bring performance gain

Rethinking

- Linear Transformations for Relations (CompGCN)

$$\mathbf{r}^{l+1} = \mathbf{W}_{rel}^{l}\mathbf{r}^{l}$$

		TransE	DistMult	ConvE
FB237	with LTR	0.335	0.342	0.353
	w/o LTR	0.336	0.343	0.352
WN18RR	with LTR	0.206	0.430	0.469
	w/o LTR	0.190	0.433	0.468

Conclusion: Linear transformation is not necessary

Rethinking

- Summary

	Entity Aggregation	Entity Update	Relation
RGCN [24]	$\mathbf{m}_e^{l+1} = \sum_{(h,r)\in\mathcal{N}_{in}(e)} \mathbf{W}_r^l \mathbf{h}^l + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \mathbf{W}_r^l \mathbf{t}^l + \mathbf{W}_0^l \mathbf{e}^l$	$\mathbf{e}^{l+1} = \sigma(\mathbf{m}_e^{l+1})$	$\mathbf{r}^{l+1} = \mathbf{r}^l$
WGCN [25]	$\mathbf{m}_e^{l+1} = \sum_{(h,r)\in\mathcal{N}_{in}(e)} \mathbf{W}^l (\alpha_r^l \mathbf{h}^l) + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \mathbf{W}^l (\alpha_r^l \mathbf{t}^l) + \mathbf{W}_0^l \mathbf{e}^l$	$\mathbf{e}^{l+1} = \sigma(\mathbf{m}_e^{l+1})$	$\mathbf{r}^{l+1} = \mathbf{r}^l$
CompGCN [31]	$\mathbf{m}_e^{l+1} = \sum_{(h,r)\in\mathcal{N}_{in}(e)} \mathbf{W}_r^l \phi_{in}(\mathbf{h}^l, \mathbf{r}^l) + \sum_{(r,t)\in\mathcal{N}_{out}(e)} \mathbf{W}_r^l \phi_{out}(\mathbf{t}^l, \mathbf{r}^l) + \mathbf{W}_0^l \mathbf{e}^l$	$\mathbf{e}^{l+1} = \sigma(\mathbf{m}_e^{l+1})$	$\mathbf{r}^{l+1} = \mathbf{W}_{rel}^l \mathbf{r}^l$

GCNs based methods have three main parts

- 1) graph structures -> not necessary
- 2) transformations for relation -> not necessary
- 3) aggregated entity representation -> necessary

Neighborhood Information

Self-loop Information

Both of them can distinguish entity to bring performance gain

Framework & Experiments

- Linear Transformed Entity(LTE) – KGE Model
  - Framework $$f(g_h(W_h\mathbf{h}), \mathbf{r}, g_t(W_t\mathbf{t})),$$
    - W_h, W_t can share the same parameters
    - g_h, g_t $\in$ {Identity function, activation functions, **batch normalization, dropout**, …}
    - Compared to TransR $$-\|\mathbf{M}_r\mathbf{h} + \mathbf{r} - \mathbf{M}_r\mathbf{t}\|_2^2$$
      - Memory-efficient and flexible to incorporate non-linear operations
  - Relationship with GCNs (g = Id): The gradient of loss function $$\sum_{r\in R_h}\sum_{t:(h,r,t)\in S}\log f(W\mathbf{h}, \mathbf{r}, W\mathbf{t}),$$
    - LTE-TransE $$\sum_{r\in R_h}\sum_{t:(h,r,t)\in S} -\frac{W^\top}{\|W\mathbf{h}+\mathbf{r}-W\mathbf{t}\|_2^2}W\mathbf{t} + \frac{W^\top(\mathbf{r}+W\mathbf{h})}{\|W\mathbf{h}+\mathbf{r}-W\mathbf{t}\|_2^2}.$$
    - LTE-DistMult $$\sum_{r\in R_h}\sum_{t:(h,r,t)\in S}\frac{W^\top\mathbf{R}}{((W\mathbf{h})^\top\mathbf{R}(W\mathbf{t}))}W\mathbf{t}$$
    - LTE-ConvE $$\sum_{r\in R_h}\sum_{t:(h,r,t)\in S}\lambda(\mathbf{h},\mathbf{r},\mathbf{t})W\mathbf{t}.$$

$$\sum_{r\in R_h}\sum_{t:(h,r,t)\in S} a(\mathbf{h},\mathbf{r},\mathbf{t})W\mathbf{t} + b(\mathbf{h},\mathbf{r},\mathbf{t}),$$

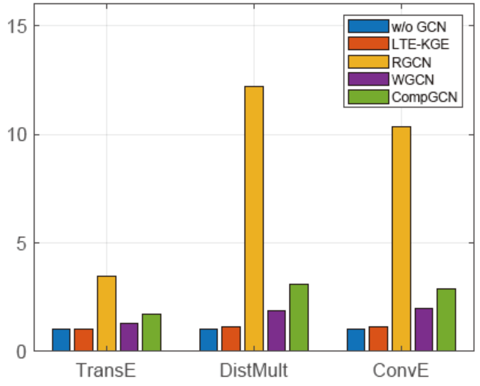
Same as 1–Layer GCN aggregation function

Framework & Experiments

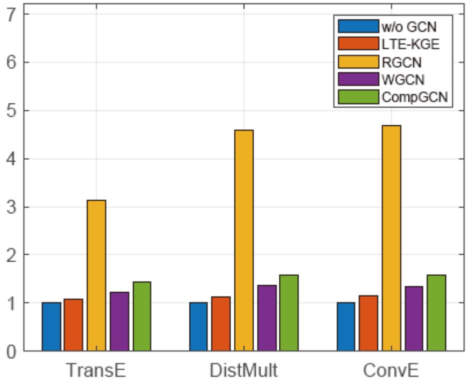
- Linear Transformed Entitv(LTE) – KGE Model
  - Experiments

	FB237					WN18RR				
	MRR	MR	H@1	H@3	H@10	MRR	MR	H@1	H@3	H@10
RotatE	.338	177	.241	.375	.533	.476	3340	.428	.492	.571
TuckER	.358	-	.266	.394	.544	.470	-	.433	.482	.526
TransE†	.332	182	.240	.368	.516	.205	3431	.022	.347	.519
DistMult†	.279	392	.202	.306	.433	.410	7970	.389	.420	.450
ConvE†	.319	276	.232	.351	.492	.462	4888	.431	.476	.525
CompGCN + TransE†	.335	205	.247	.369	.511	.206	3182	.064	.281	.502
CompGCN + DistMult†	.342	200	.252	.372	.520	.430	4559	.395	.439	.513
CompGCN + ConvE†	.353	221	.261	.388	.536	.469	3065	.433	.480	.543
LTE-TransE	.334	182	.241	.370	.519	.211	3290	.022	.362	.521
LTE-DistMult	.335	238	.246	.367	.517	.437	4485	.403	.447	.517
LTE-ConvE	.355	249	.264	.389	.535	.472	3434	.437	.485	.544

1 – Layer GCN



(a) Training Time



(b) Test Time

Conclusion

- The transformations for entity embeddings which can distinguish different entities account for the performance improvements for GCN-based methods.

Discussion

- RGCN/WGCN/CompGCN slightly outperform KGE only by transformation of entity embeddings (not by aggregating neighbor information)

  -> Maybe the entity transformation contains global information?

- RED-GNN v.s. CompGCN

	RED-GNN	CompGCN
Message	propagation	aggregation
Representation	query-dependent	node-dependent
Decoder	node-level readout	scoring function
entity embedding	×	√

  - This article mainly focus on aggregation-like message passing model with entity embeddings, while RED-GNN propagate message without entity embeddings.
  - Path (not simply aggregating neighbor information) seems important to bring performance gain.

Thanks!