Inductive Relation Prediction Using Analogy Subgraph Embeddings

Jiarui Jin[1], Yangkun Wang[1], Kounianhua Du[1], Weinan Zhang[1], Quan Gan[2], Zheng Zhang[2], Yong Yu[1], David Wipf[2]

[1]Shanghai Jiao Tong University, [2]Amazon

Ph.D. Candidate, Shanghai Jiao Tong University
Applied Scientist Intern, Amazon

Knowledge Graph

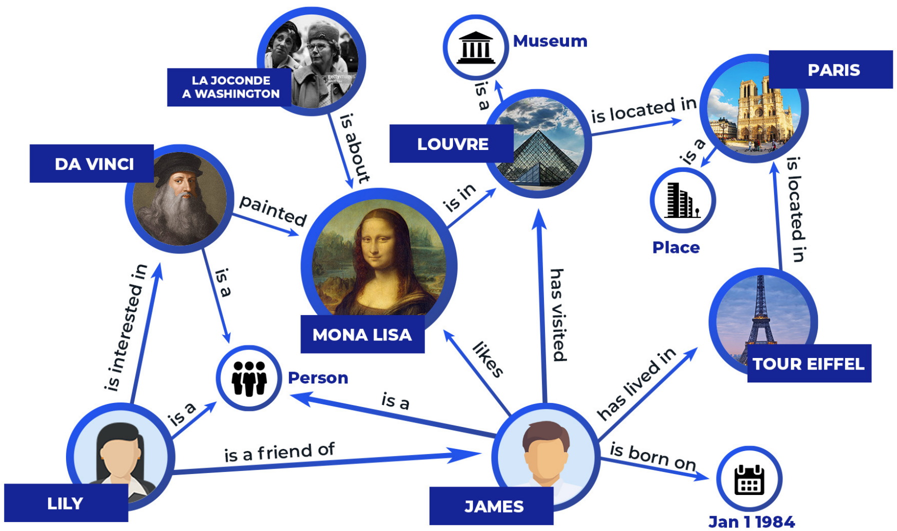


Figure: Knowledge Graph

- $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ consisting of an entity set $\mathcal{V}$ and a relation set $\mathcal{E}$.
- KG is a special heterogeneous graph.
- A set of facts are represented as triples (head entity, relation, tail entity).

Inference: Mona Lisa is in Louvre, Louvre is located in Paris → Mona Lisa is in Paris.

Logics and Graph Patterns

- Traditional models in heterogeneous information network always focus on the neighborhood information.
- These methods ignore analogy patterns (logic) in the whole graph.

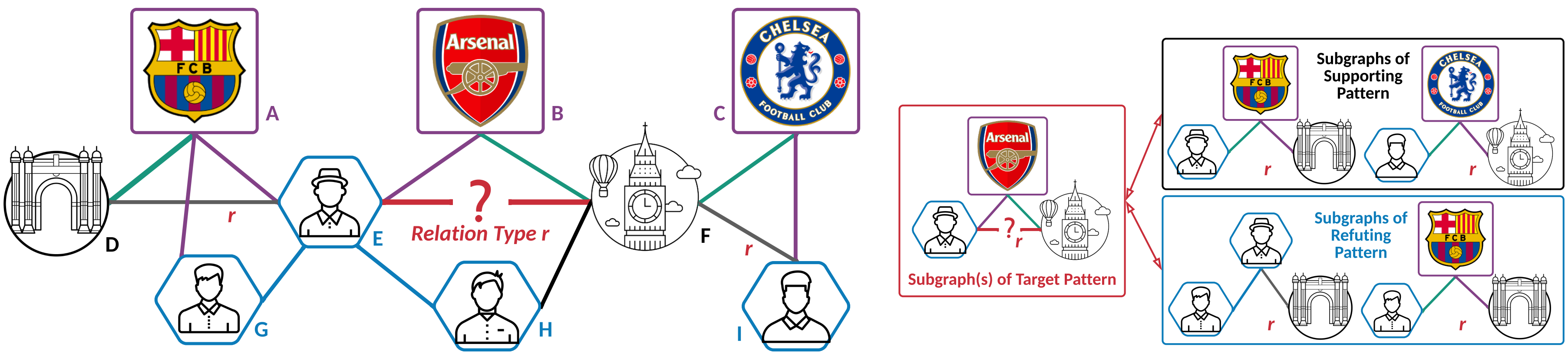


Figure: An illustrated example of Heterogeneous Graph.

Overall Framework

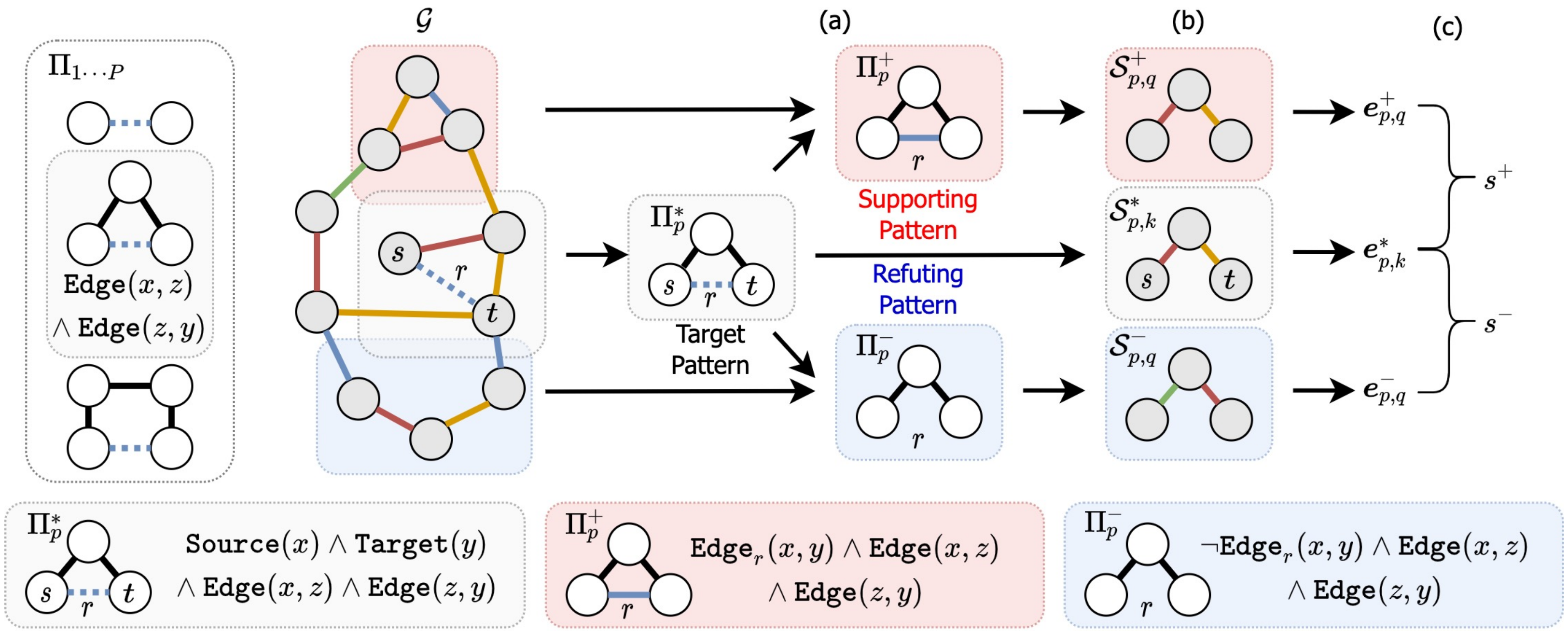


Figure: Overview of GraphANGEL (**AN**alogy sub**G**raph **E**mbedding **L**earning) where different edge colors in the graph represent different relation types, and dashed edge in graph represent the triplet we wish to predict. The left box shows the patterns considered in our implementation, where black edges mean matching edges irrespective of relation types. The bottom boxes show the logical function of the three patterns.

Search and Retrieval Module

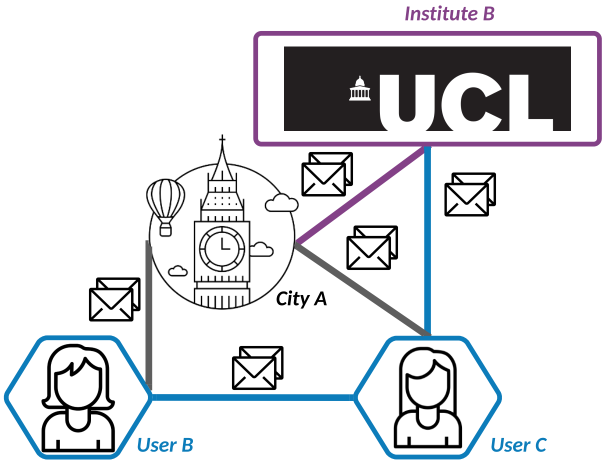


Figure: An illustrated example of Heterogeneous Graph.

Pairs

$\boldsymbol{u_B}: <u_B, u_C>, <u_B, c_A>$

$\boldsymbol{u_C}: <u_C, u_B>, <u_C, c_A>, <u_C, i_B>$

$\boldsymbol{c_A}: <c_A, u_B>, <c_A, u_C>, <c_A, i_B>$

$\boldsymbol{i_B}: <i_B, c_A>, <i_B, u_C>$

Triangles

$\boldsymbol{u_B}: <u_B, u_C, c_A>, <u_B, u_C, i_B>, <u_B, c_A, u_C>, <u_B, c_A, i_B>$

$\boldsymbol{u_C}: <u_C, u_B, c_A>, <u_C, c_A, u_B>, <u_C, c_A, i_B>, <u_C, i_B, c_A>$

$\boldsymbol{c_A}: <c_A, u_B, u_C>, <c_A, u_C, u_B>, <c_A, u_C, i_B>, <c_A, i_B, u_C>$

$\boldsymbol{i_B}: <i_B, c_A, u_B>, <i_B, c_A, u_C>, <i_B, u_C, u_B>, <i_B, u_C, c_A>$

Quadrangles

… …

Search and Retrieval Algorithms

Theorem 1. (Time Complexity of Retrieval and Sampling) *Given a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *and a graph pattern* $\Pi_{\texttt{3-cycle}}$ *in 3-cycle and a graph pattern* $\Pi_{\texttt{4-cycle}}$ *in 4-cycle, the time complexity of retrieving all the (supporting) subgraphs satisfying* $\Pi_{\texttt{3-cycle}}$ *is* $O(|\mathcal{E}|^{\frac{3}{2}})$*, of retrieving all the (supporting) subgraphs satisfying* $\Pi_{\texttt{4-cycle}}$ *is* $O(\max(|\mathcal{E}|^{\frac{3}{2}}, N_{\texttt{4-cycle}}))$ *where* $N_{\texttt{4-cycle}}$ *is the number of quadratic cycles in* $\mathcal{G}$ *and also the trivial lower bound of the time complexity. For uniform sampling algorithms, the time complexity of sampling* $n_{\texttt{4-cycle}}$ *supporting cases of* $\Pi_{\texttt{4-cycle}}$ *is* $O(|\mathcal{E}|^{\frac{3}{2}} + n_{\texttt{4-cycle}})$*. As there are usually more refuting cases than supporting ones, the time complexity of sampling* n *refuting cases of* $\Pi_{\texttt{3-cycle}}$ *or* $\Pi_{\texttt{4-cycle}}$ *is* $O(|\mathcal{V}| + |\mathcal{E}| + n)$.

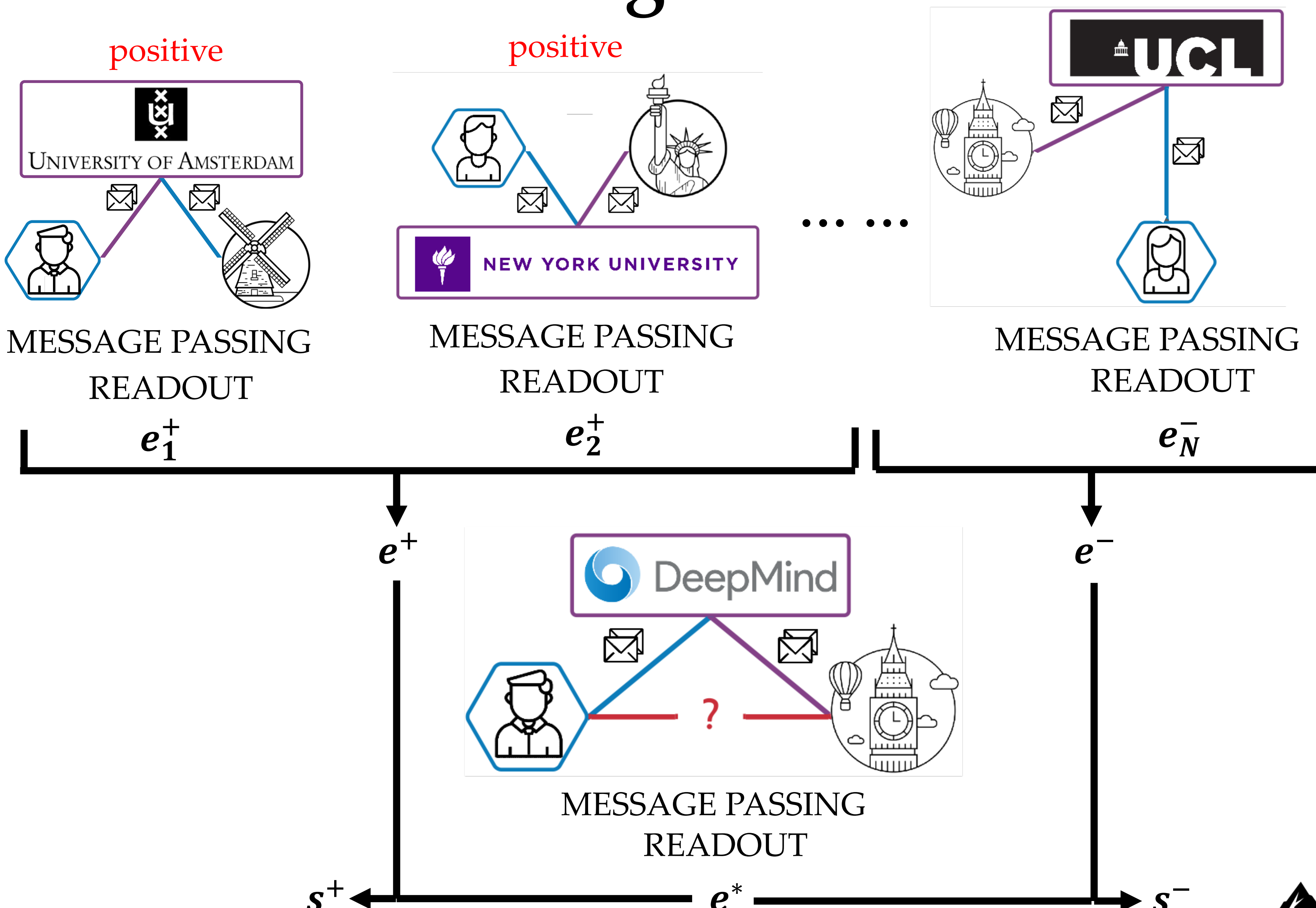
Algorithm 3: Search and Retrieval Algorithm B for $\Pi_{\texttt{3-cycle}}$

$\mathcal{B} \leftarrow \{\}$
foreach $u \in \mathcal{V}$ **do**
  foreach $v \in \mathcal{N}_{\mathcal{G}}(u)$ where $d_v \leq d_u$ **do**
    foreach $w \in \mathcal{N}_{\mathcal{G}}(v)$ where $d_w \leq d_v$ **do**
      if $\langle v, w \rangle \in \mathcal{E}$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(u, v, w)\}$
      end
    end
  end
end

Algorithm 4: Search and Retrieval Algorithm for $\Pi_{\texttt{4-cycle}}$

$\mathcal{B} \leftarrow \{\}$
foreach $u \in \mathcal{V}$ **do**
  $\mathcal{T}_x \leftarrow \{\}$ for each $x \in \mathcal{V}$
  foreach $v \in \mathcal{N}_{\mathcal{G}}(u)$ where $d_v \leq d_u$ **do**
    foreach $w \in \mathcal{N}_{\mathcal{G}}(v)$ where $d_w \leq d_u$ **do**
      foreach $x \in \mathcal{T}_w$ **do**
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(u, v, w, x)\}$
      end
      $\mathcal{T}_w \leftarrow \mathcal{T}_w \cup \{v\}$
    end
  end
end

Encoding Module



Amazon Web Services

Apex Data & Knowledge Management Lab

Encoding Algorithms

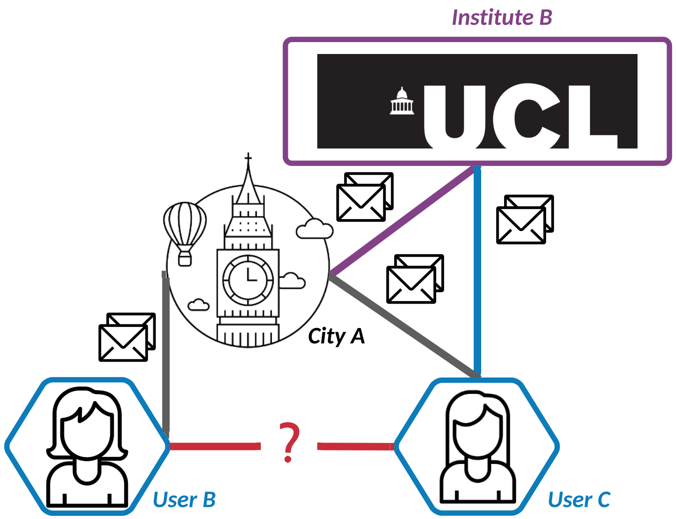


Figure: An illustrated example of Heterogeneous Graph.

MESSAGE PASSING:

$$m_v^{t+1} = \sum_{w \in \mathcal{N}(v)} M_t(h_v^t, h_w^t, e_{vw})$$

$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$

M_t is message function and U_t is vertex update function, hidden states h_v^t at each node in the graph are updated based on messages m_v^{t+1}.

READOUT:

$$y = R(\{h_v^T | v \in G\})$$

The message function M_t , vertex update function U_t, and readout function R are all learned differentiable function.

Ref: Justin Gilmer, et al. Neural Message Passing for Quantum Chemistry. ICML, 2017.

Experiments

Table 2: Patterns Π_p considered in our experiments.

Task	Pair	3-cycle (with type)	4-cycle (with type)
Knowledge Graph Completion	`true`	$\texttt{Edge}(x,z) \land \texttt{Edge}(z,y)$	$\texttt{Edge}(x,z) \land \texttt{Edge}(z,w) \land \texttt{Edge}(w,y)$
Heterogeneous Graph Recommendation	`true`	$\texttt{Edge}_a(x,z) \land \texttt{Edge}_b(z,y)$	$\texttt{Edge}_a(x,z) \land \texttt{Edge}_b(z,w) \land \texttt{Edge}_c(w,y)$

Table 3: Result comparisons with baselines on heterogeneous graph recommendation task.

Models	LastFM			Yelp			Amazon			Douban Book		
	AUC	ACC	F1	AUC	ACC	F1	AUC	ACC	F1	AUC	ACC	F1
HetGNN	0.7936	0.7258	0.7177	0.9083	0.8297	0.8205	0.7744	0.7108	0.7109	0.8737	0.7912	0.7915
HAN	0.8915	0.8337	0.8296	0.9156	0.8488	0.8426	0.8487	0.7682	0.7572	0.9244	0.8501	0.8458
TAHIN	0.8910	0.8463	0.8337	0.9067	0.8490	0.8393	0.8535	0.7718	0.7644	0.9253	0.8497	0.8373
HGT	0.8394	0.7939	0.7882	0.9006	0.8375	0.8334	0.7125	0.6482	0.6296	0.9132	0.8364	0.8222
R-GCN	0.8526	0.8393	0.8341	0.9098	0.8427	0.8323	0.8130	0.7408	0.7366	0.9203	0.8413	0.8271
GraphANGEL$_{\texttt{3-cycle}}$	0.8934	0.8519	0.8465	0.9167	0.8498	0.8514	0.8601	0.7746	0.7746	0.9256	0.8512	0.8479
GraphANGEL$_{\texttt{4-cycle}}$	0.8961	0.8514	0.8467	0.9201	0.8506	0.8521	0.8609	0.7752	0.7716	0.9242	0.8502	0.8378
GraphANGEL	0.8979	0.8524	0.8469	0.9231	0.8512	0.8533	0.8611	0.7790	0.7753	0.9311	0.8601	0.8543
GraphANGEL*	**0.9001**	**0.8611**	**0.8589**	**0.9337**	**0.8701**	**0.8577**	**0.8700**	**0.7810**	**0.7813**	**0.9410**	**0.8640**	**0.8591**

Experiments

Table 4: Result comparisons with baselines on knowledge graph completion task.

Models	FB15k-237					WN18RR				
	MR	MRR	Hit@1	Hit@3	Hit@10	MR	MRR	Hit@1	Hit@3	Hit@10
pLogicNet	173	0.332	0.237	0.367	0.524	3408	0.441	0.398	0.446	0.537
TransE	181	0.326	0.229	0.363	0.521	3410	0.223	0.235	0.401	0.531
ConvE	244	0.325	0.237	0.356	0.501	4187	0.430	0.400	0.440	0.520
ComplEx	339	0.247	0.158	0.275	0.428	5261	0.440	0.410	0.460	0.510
MLN	1980	0.098	0.067	0.103	0.160	11549	0.259	0.191	0.322	0.361
RotatE	177	0.338	0.241	0.375	0.533	3340	0.476	0.428	0.492	0.571
RNNLogic	232	0.344	0.252	0.380	0.530	4615	0.483	0.446	0.497	0.558
ComplEx-N3	159	0.370	0.272	0.400	0.561	3452	0.491	0.440	0.500	0.581
GraIL	205	0.322	0.223	0.361	0.520	3539	0.401	0.352	0.438	0.501
QuatE	**87**	0.348	0.248	0.382	0.550	**2314**	0.488	0.438	0.508	0.582
GraphANGEL$_{\texttt{3-cycle}}$	159	0.366	0.270	0.398	0.560	2919	0.492	0.463	0.497	0.590
GraphANGEL$_{\texttt{4-cycle}}$	165	0.351	0.239	0.381	0.548	2914	0.493	0.465	0.502	0.587
GraphANGEL	151	**0.374**	**0.275**	**0.408**	**0.564**	2834	**0.504**	**0.470**	**0.515**	**0.598**
	±3	±0.003	±0.002	±0.004	±0.004	±25	±0.003	±0.002	±0.004	±0.004

Experiments

Table 5: Result comparisons with baselines on generalization setting by randomly removing 20% relations. See Appendix A6.2 for full version and Appendix A6.2 for results of dropping 5%, 10%, 15%. The numbers in brackets show the descent degree.

Models	FB15k-237			WN18RR		
	Hit@1	Hit@3	Hit@10	Hit@1	Hit@3	Hit@10
pLogicNet*	0.112(52.7%↓)	0.179(51.2%↓)	0.257(51.0%↓)	0.141(64.6%↓)	0.222(50.2%↓)	0.267(50.3%↓)
TransE*	0.101(55.9%↓)	0.163(55.1%↓)	0.246(52.8%↓)	0.072(46.7%↓)	0.200(50.1%↓)	0.260(51.0%↓)
ConvE*	0.104(56.1%↓)	0.178(50.0%↓)	0.247(50.7%↓)	0.201(49.8%↓)	0.223(49.3%↓)	0.268(48.5%↓)
ComplEx*	0.078(50.6%↓)	0.142(48.4%↓)	0.226(47.2%↓)	0.214(47.8%↓)	0.236(48.7%↓)	0.267(47.6%↓)
MLN*	0.031(53.7%↓)	0.049(52.4%↓)	0.070(56.3%↓)	0.092(51.8%↓)	0.154(52.2%↓)	0.178(50.7%↓)
RotatE*	0.121(49.8%↓)	0.187(50.1%↓)	0.271(49.1%↓)	0.238(44.3%↓)	0.260(47.1%↓)	0.296(48.2%↓)
RNNLogic*	0.124(50.7%↓)	0.172(54.7%↓)	0.240(54.6%↓)	0.244(45.2%↓)	0.260(47.6%↓)	0.281(49.7%↓)
ComplEx-N3*	0.142(47.2%↓)	0.208(49.6%↓)	0.289(48.5%↓)	0.250(43.2%↓)	0.269(46.2%↓)	0.311(46.4%↓)
GraIL*	0.125(43.9%↓)	0.185(48.8%↓)	0.263(49.4%↓)	0.195(44.7%↓)	0.222(49.3%↓)	0.267(46.8%↓)
QuatE*	0.127(48.7%↓)	0.190(50.3%↓)	0.282(48.7%↓)	0.248(43.3%↓)	0.255(49.8%↓)	0.308(47.0%↓)
GraphANGEL$_{3-\mathtt{cycle}}$	0.168(37.6%↓)	0.230(42.2%↓)	0.333(40.5%↓)	0.277(40.2%↓)	0.291(**41.4%**↓)	0.329(44.3%↓)
GraphANGEL$_{4-\mathtt{cycle}}$	0.147(38.7%↓)	0.222(41.7%↓)	0.328(40.2%↓)	0.278(40.2%↓)	0.291(42.1%↓)	0.326(44.4%↓)
GraphANGEL	**0.173**(**37.2%**↓)	**0.238**(**41.5%**↓)	**0.337**(**40.1%**↓)	**0.284**(**39.5%**↓)	**0.299**(41.8%↓)	**0.334**(**44.1%**↓)

Conclusion

- Traditional graph-based models usually condense the neighborhood connectivity pattern of each node into a node-specific low-dimensional embedding and exploit such local connectivity.
- By bridging logical expressions and graph patterns, GraphANGEL predicts relations between each node pair by checking whether the subgraphs containing the pair are similar to other (analogy) subgraphs containing
  the considered relation .
- Each graph pattern explicitly represents a specific logical rule, which contributes to an inductive bias that facilitates generalization to unseen relation types and leads to more explainable predictive models.

Thanks for Your Listening

Email: jinjiarui97@sjtu.edu.cn
jiaruiji@amazon.com