Inductive Relation Prediction by Subgraph Reasoning

Presenter: Zhanke Zhou

2021. 11. 12

Background – About the paper

- Title: Inductive Relation Prediction by Subgraph Reasoning
- Authors: Komal K. Teru, Etienne Denis, William L. Hamilton
- Conference: ICML'20
- Affiliation: McGill University, Mila
- Paper: http://proceedings.mlr.press/v119/teru20a/teru20a.pdf
- Code: https://github.com/kkteru/grail

Outline

- Background
- The proposed framework: GraIL
- Key takeaways and research directions

Outline

- Background
  - Knowledge graph embedding (KGE)
  - Two learning paradigms in KGE
- The proposed framework: GraIL
- Key takeaways and research directions

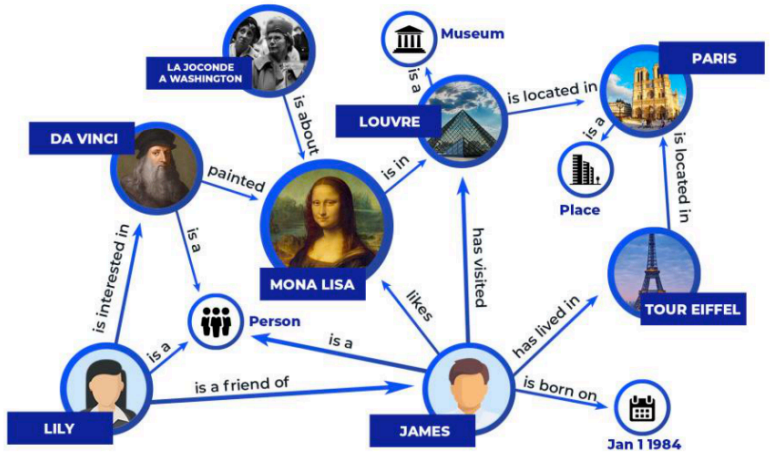
Background – Knowledge Graph (KG)

A knowledge graph

- Mainly describe real world entities and relations, organized in a graph
- Allows potentially interacting arbitrary entities with each other

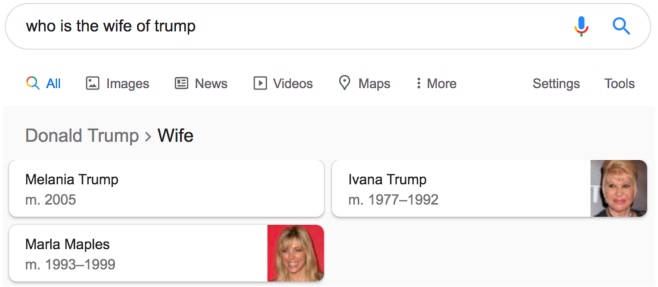
Preliminaries

- Graph representation: $\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{F})$
- Entities $\mathcal{E}$
  - real world objects or concepts
- Relations $\mathcal{R}$
  - interactions between entities
- Facts $\mathcal{F}$
  - the basic unit in form of (s, r, o)
  - (subject entity, relation, object entity)

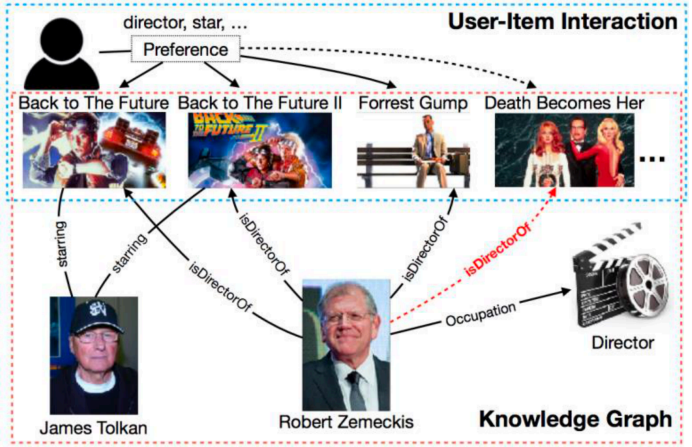


Applications

KGQA:

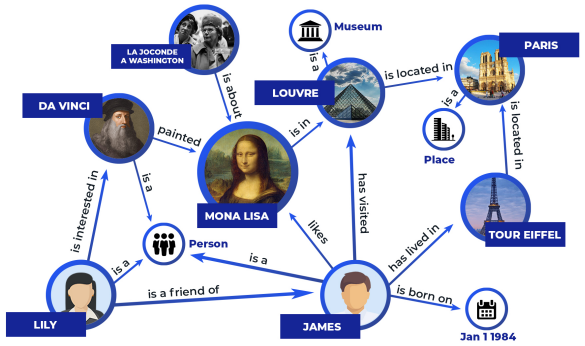


Recommendation:



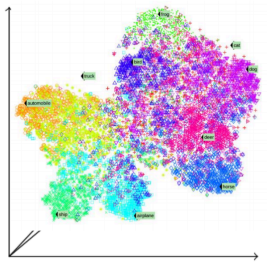
Background – Knowledge Graph Embedding (KGE)

- Knowledge Graph Embedding
  - Encode entities and relations in KG into low-dimensional vectors space
  - while capturing nodes' and edges' connection properties



vectorize

entities & relations

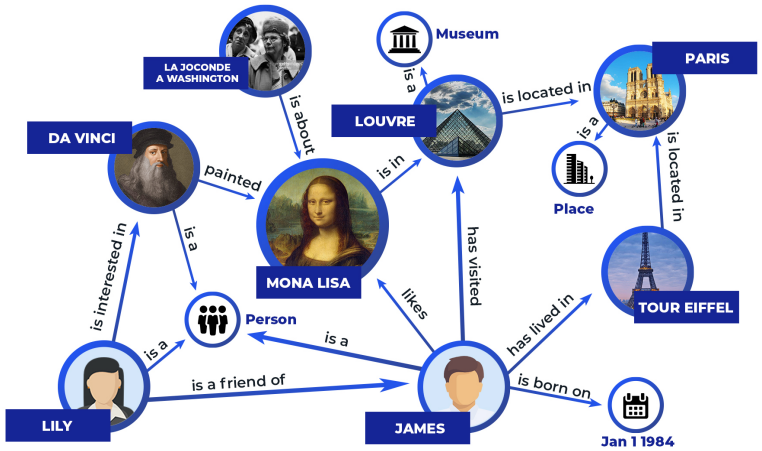


Two kinds of information
- **Semantical**
- **Topological**

- Most KGE models based on embeddings define a scoring function ϕ to estimate the plausibility of any fact (s, r, o) using their embeddings:

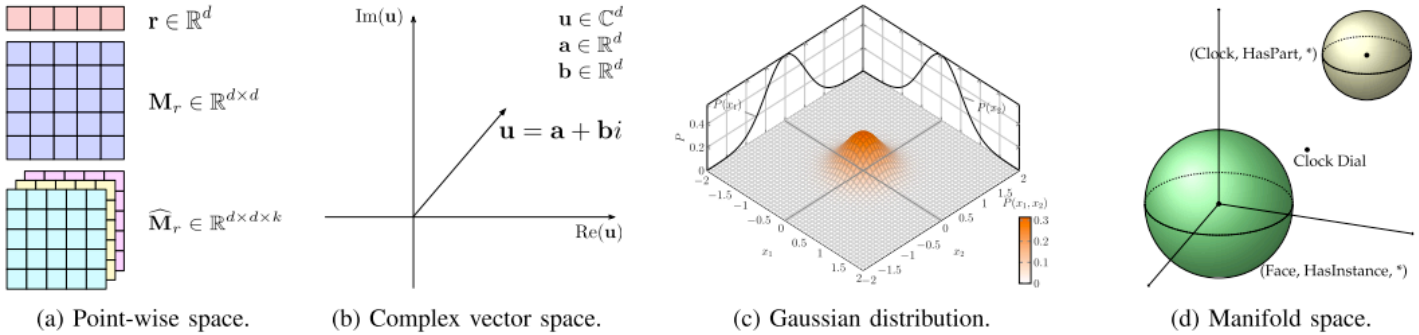
$$\phi(\boldsymbol{s}, \boldsymbol{r}, \boldsymbol{o})$$

Background – Knowledge Graph Embedding (KGE)

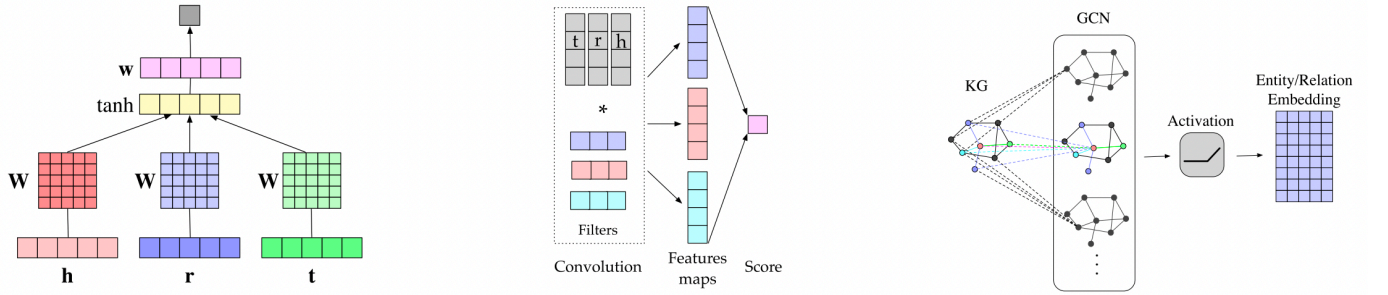


Main Challenge:

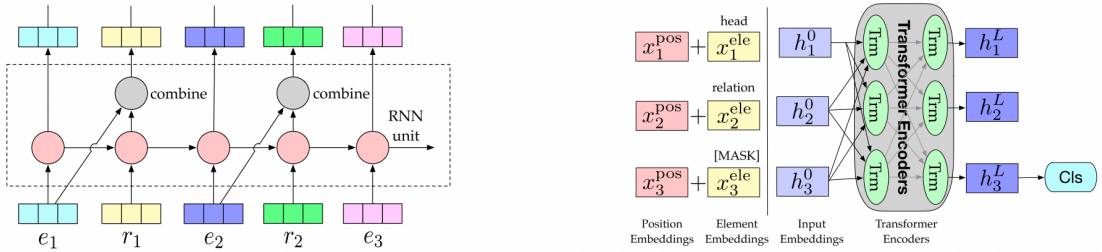
How to **extract and utilize**
the semantical / topological information
in the original KG ?



(a) Point-wise space. (b) Complex vector space. (c) Gaussian distribution. (d) Manifold space.



(a) MLP. (b) CNN. (c) GCN.

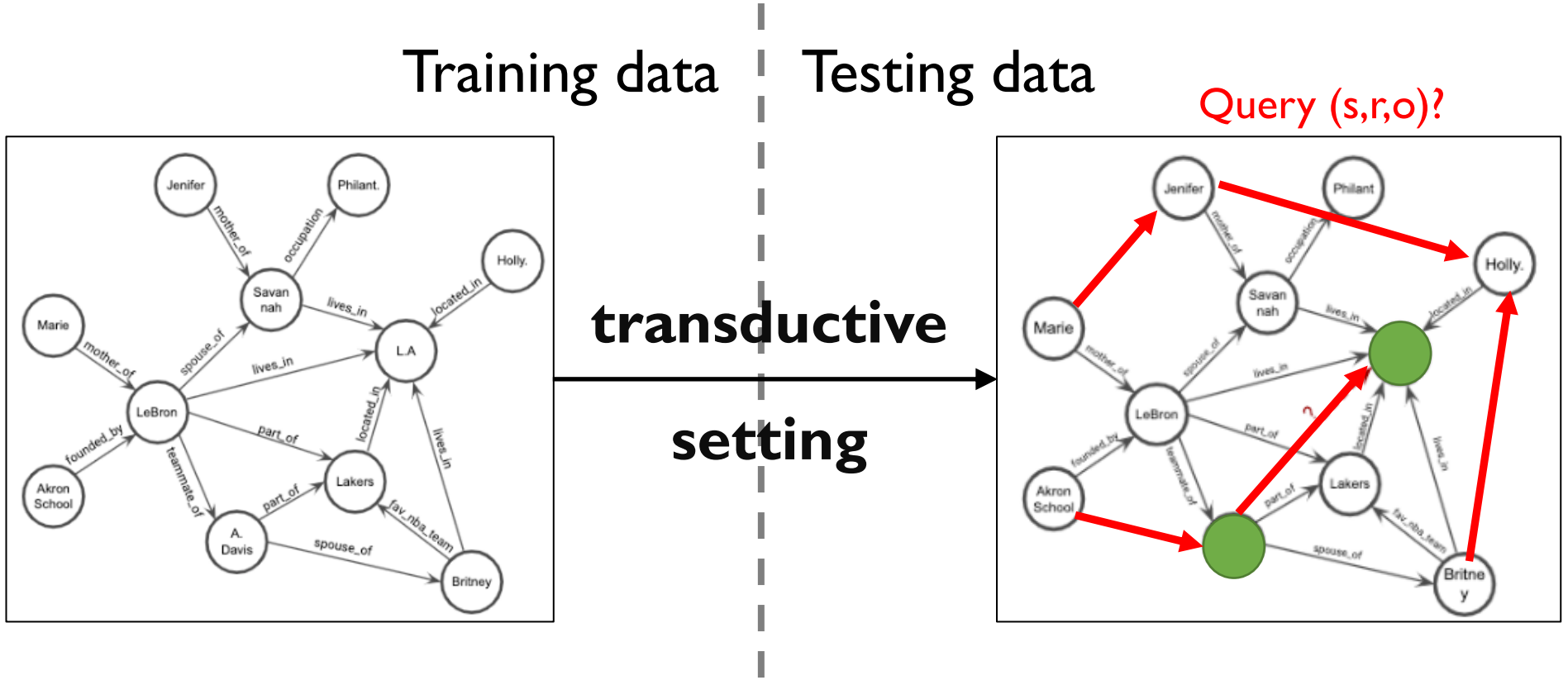


(d) RSN. (e) Transformer.

$$\phi(s, r, o)$$

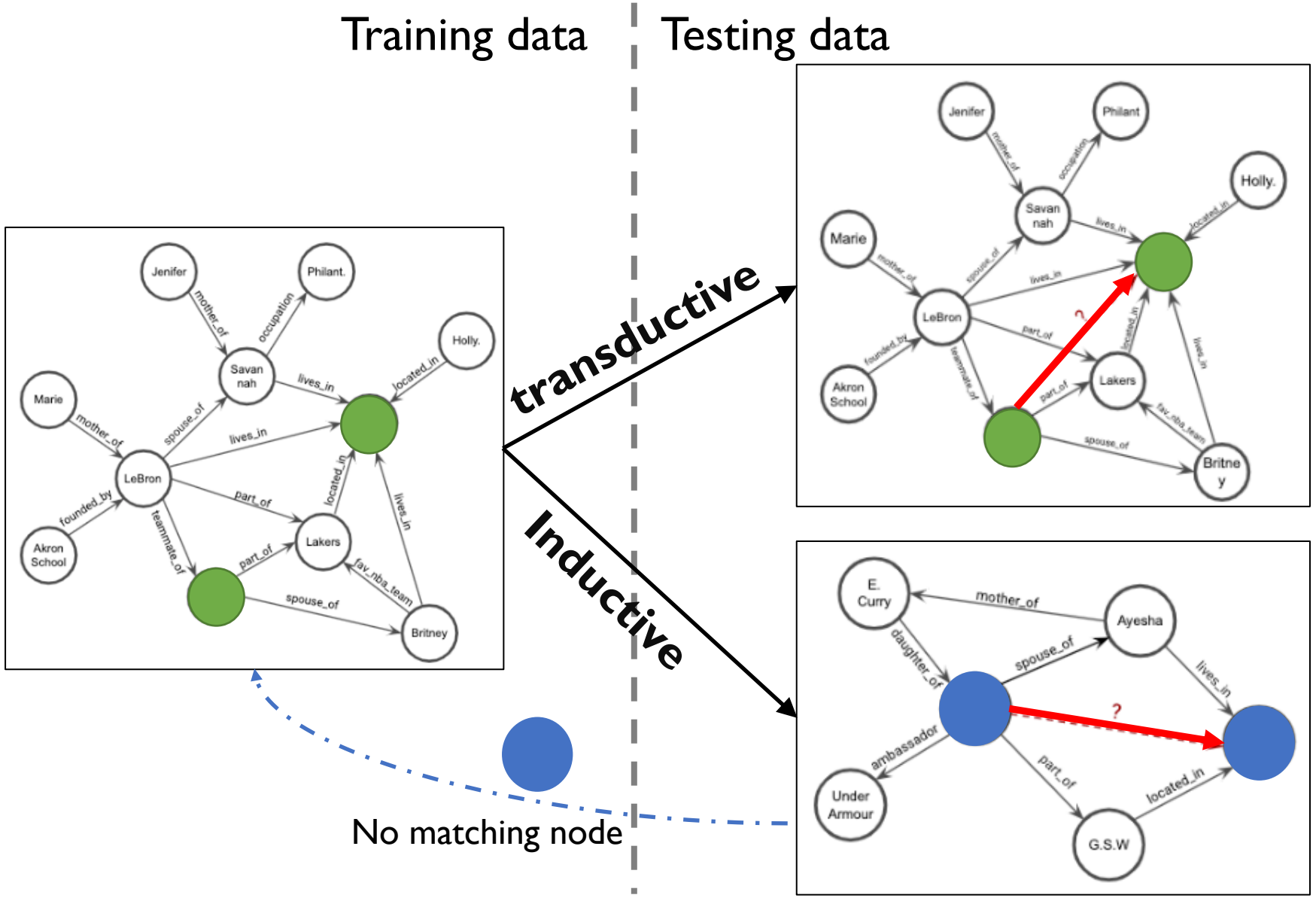
Background – Knowledge Graph Embedding (KGE)

Transductive learning | Popular learning paradigm in KGE



The same set of entities (nodes)
for the training and testing data

Two learning paradigms: Inductive and transductive



The same set of entities (nodes) as the training data

Different set of entities to the training data

e.g.,

- new users and products on e-commerce platforms
- new molecules in biomedical KG

Two learning paradigms: Inductive and transductive

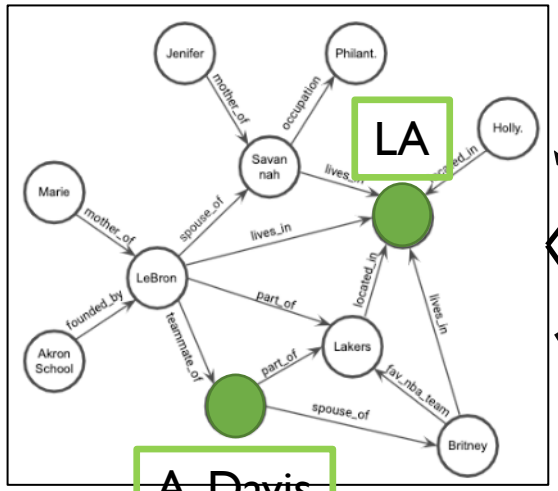
Inductive learning

- is how AI systems attempt to use a generalized rule to carry out observations [1]
- e.g., to find symbolic descriptions expressed in high-level terms that people can understand [2]

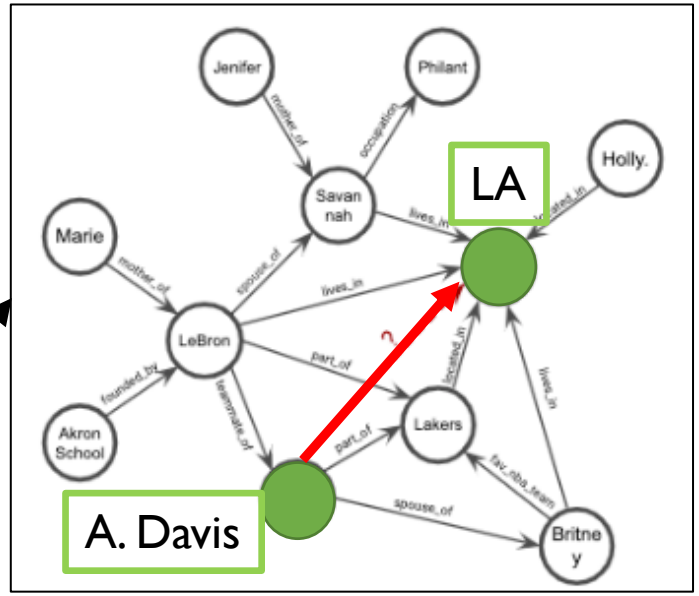
Inductive learning	Transductive learning
Train the model and label unlabelled points which we have never encountered.	Train the model and label unlabelled points which we have already encountered.
Builds a predictive model. If new unlabelled points are encountered, we can use the initially built model.	Does not build a predictive model. If new unlabelled points are encountered, we will have to re-run the algorithm.
Can predict any point in the space of points beyond the unlabelled points.	Can predict only the points in the encountered testing dataset based on the observed training dataset.

[1] https://towardsdatascience.com/introduction-to-inductive-learning-in-artificial-intelligence-dafc2796405b#:~:text=Inductive%20Learning%2C%20also%20known%20as,If%20this%20then%20that%E2%80%9D%20format.
[2] Applying inductive learning to enhance knowledge-based expert systems. MJ Shaw. 1987
[3] https://towardsdatascience.com/inductive-vs-transductive-learning-e608e786f7d

Two learning paradigms: Inductive and transductive



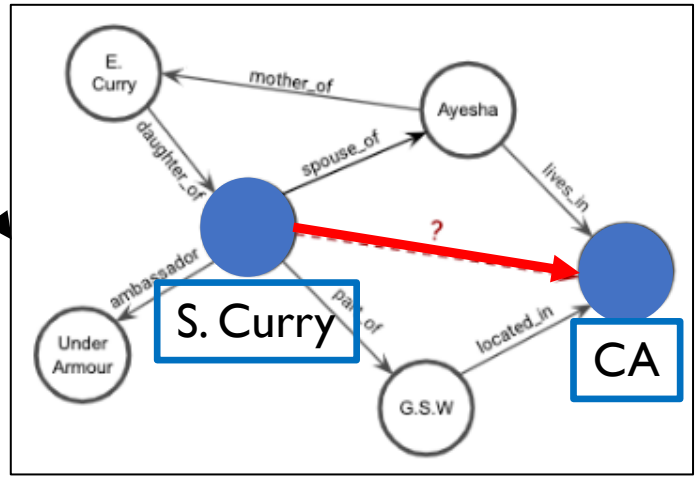
transductive



Query = (A. Davis, lives in, LA) ?

Semantic node embedding of [Davis] / [lives in] / [LA]

Inductive

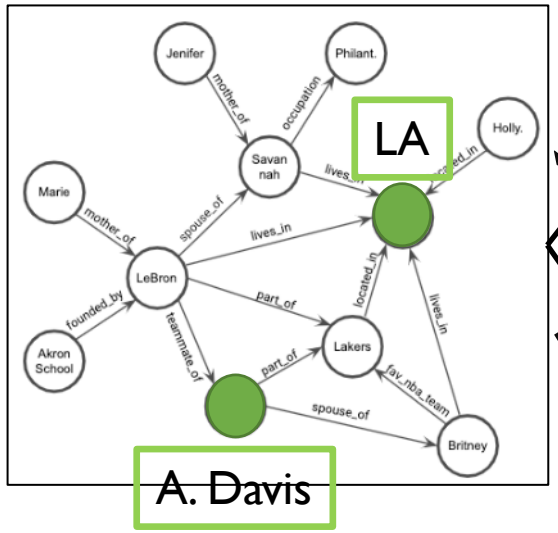


Query = (S. Curry, lives in, CA) ?

(X, spouse of, Y) ∧ (Y, lives in, Z) → (X, lives in, Z)

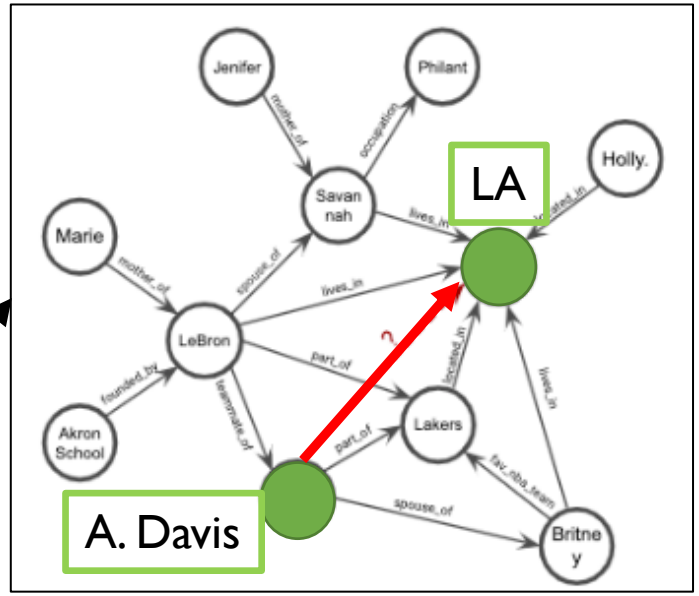
(X, part of, Y) ∧ (Y, located in, Z) → (X, lives in, Z)

Two learning paradigms: Inductive and transductive



transductive

Inductive

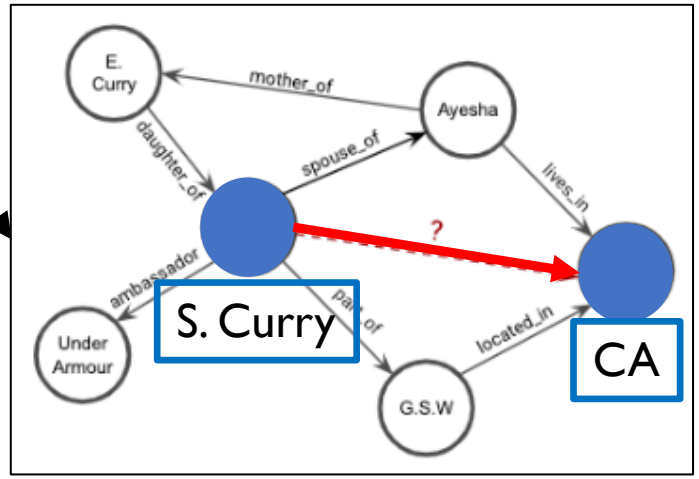


Query = (A. Davis, lives in, LA) ?

Semantic embedding of [Davis] / [lives in] / [LA]

Embedding models
- ✓ semantics
- ? topologies

Representatives
- TransE
- RotatE
- ComplEx
- DistMult



Query = (S. Curry, lives in, CA) ?

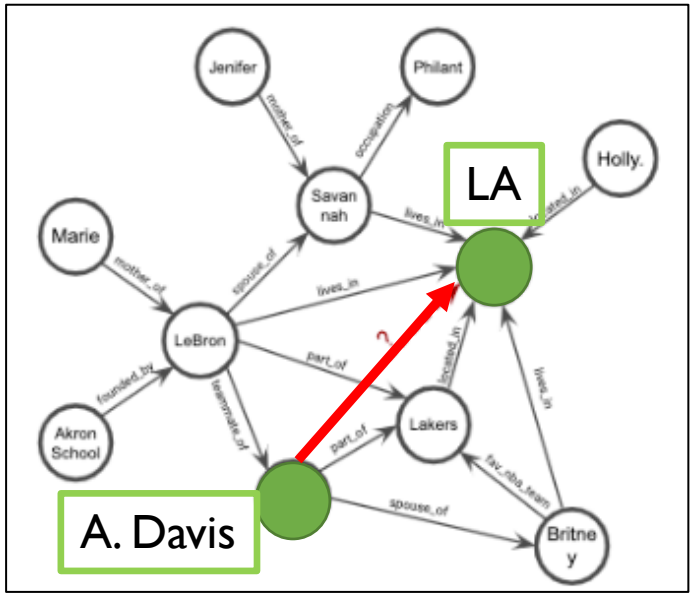
(X, spouse of, Y) ∧ (Y, lives in, Z) → (X, lives in, Z)

Rule mining models
- ? semantics
- ✓ topologies

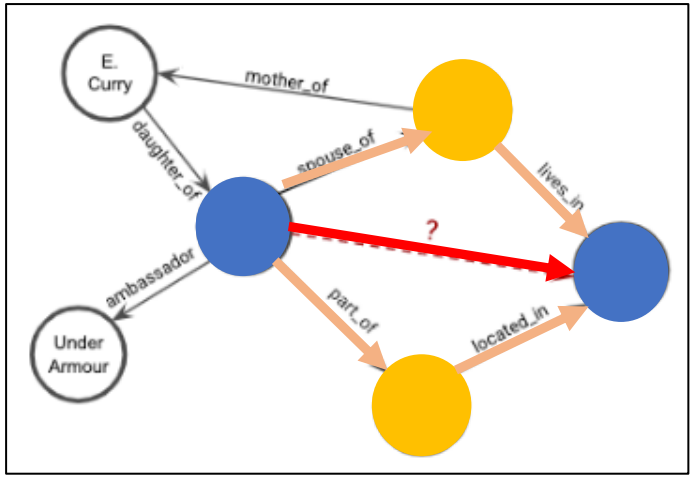
Representatives
- Neural LP
- DRUM
- RuleN

Rethink existing works

- Embedding models
  - Pros
    - simple, powerful, and popular
    - can be enhanced by other models (e.g., RNN, GNN)
  - Cons
    - the learned embedding is not understandable to human
    - do not explicitly capture the compositional logical rules
    - limited to the transductive setting



- Rule mining models (structure learning models)
  - Pros
    - more explainable, easier to understand
  - Cons
    - more difficult to train and optimize
    - suffer from scalability issues



Outline

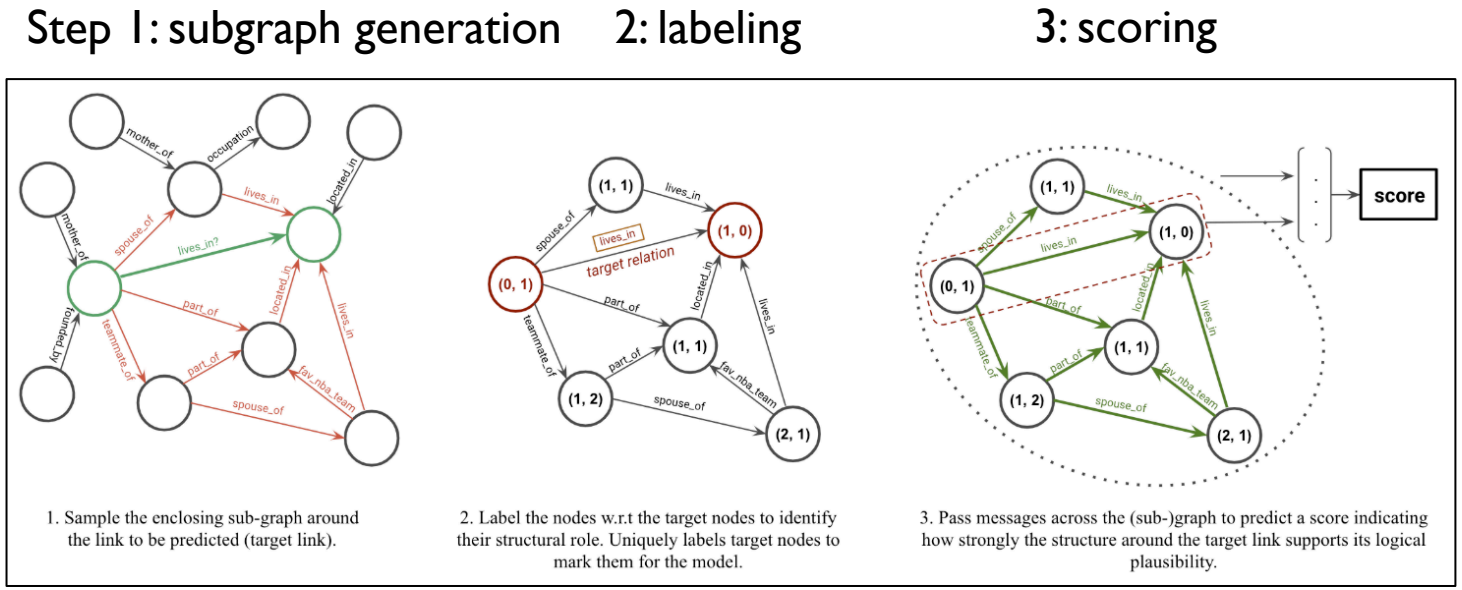
- Background
- The proposed framework: GraIL
  - Overview
  - Technical details
  - Experiments
- Key takeaways and research directions

Graph Inductive Learning (GraIL)

Overview

- a GNN-based relation prediction framework
- reasons over local subgraph structures
- has the inductive bias to learn entity-independent relational semantics
  - can generalize to unseen entities and graphs after training

Two key components
- subgraph preparing (step1 & 2)
- subgraph scoring (step 3)



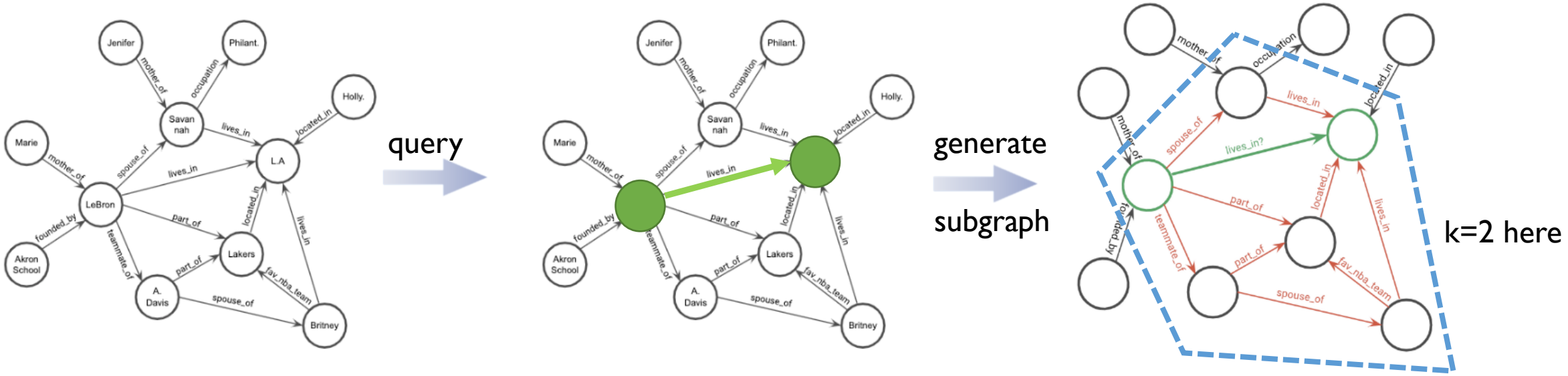
Outline

- Background
- The proposed framework: GraIL
  - Overview
  - Technical details
    - step1: subgraph generation
    - step2: subgraph labeling
    - step3: subgraph scoring
  - Experiments
- Key takeaways and research directions

GraIL | Technical details

Step 1: Subgraph generation

- Assumption
  - local graph neighborhood of a particular triplet in the KG contains the logical evidence
- Subgraph generation
  - *enclosing subgraph*: the graph induced by all the nodes that occur on a path between u and v. (by taking the intersection, $\mathcal{N}_k(u) \cap \mathcal{N}_k(v)$)

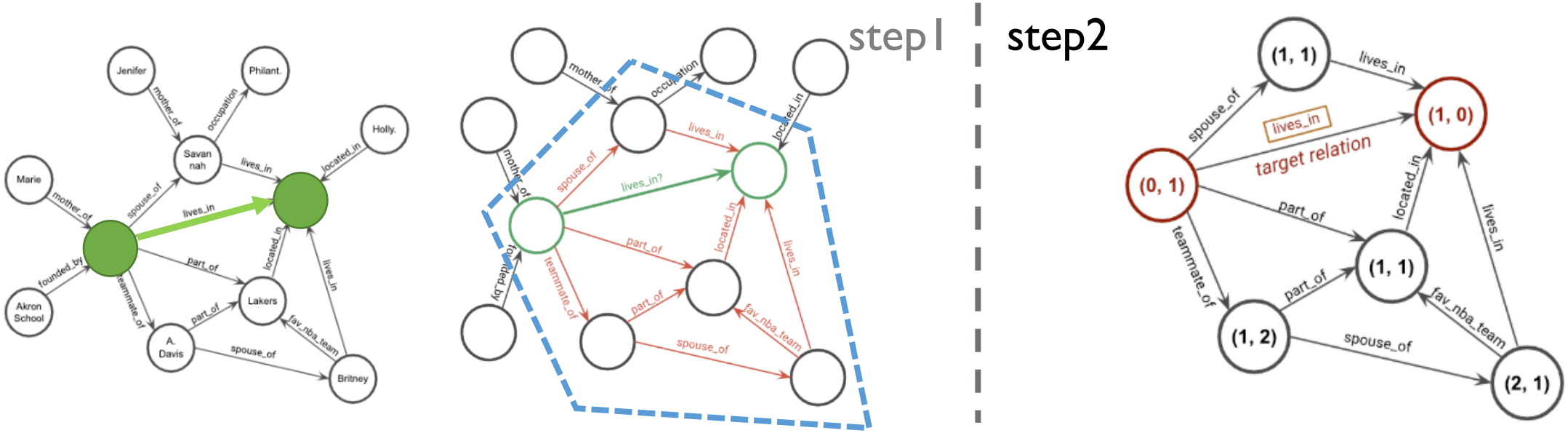


GraIL | Technical details

Two key components
- subgraph preparing (step1 & 2)
- subgraph scoring (step 3)

Step 2: Subgraph labeling

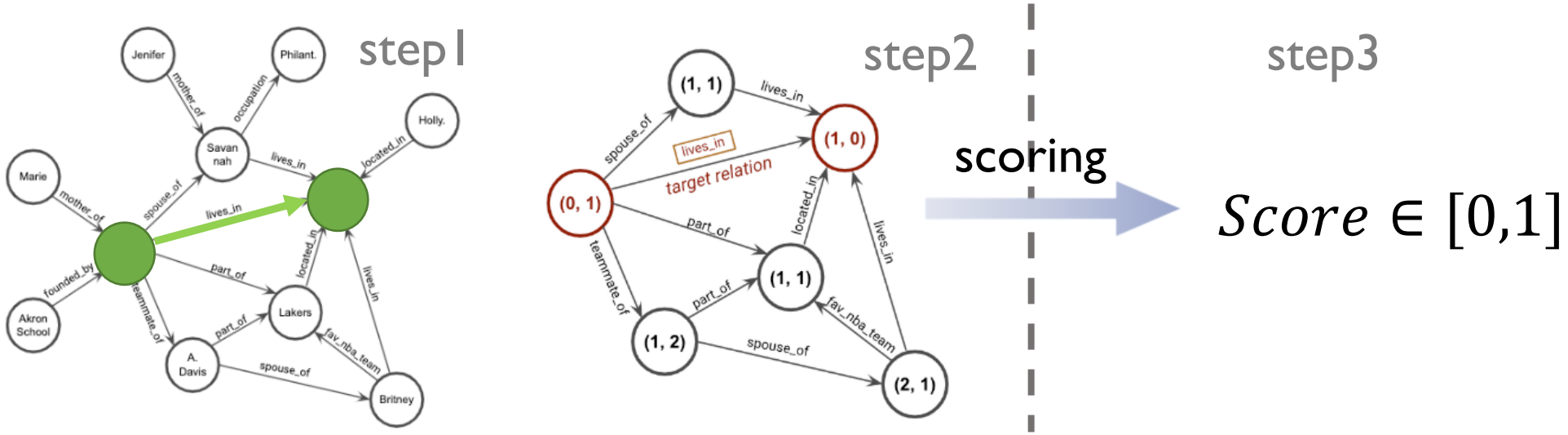
- aims to give entity-independent node feature
- each node i is labeled with (d(i, u), d(i, v))
  - d(i, u) denotes the shortest distance between nodes i and u
    - counted by Dijkstras algorithm
  - The node features are [one-hot(d(i, u)) ⊕ one-hot(d(i, v))]



GraIL | Technical details

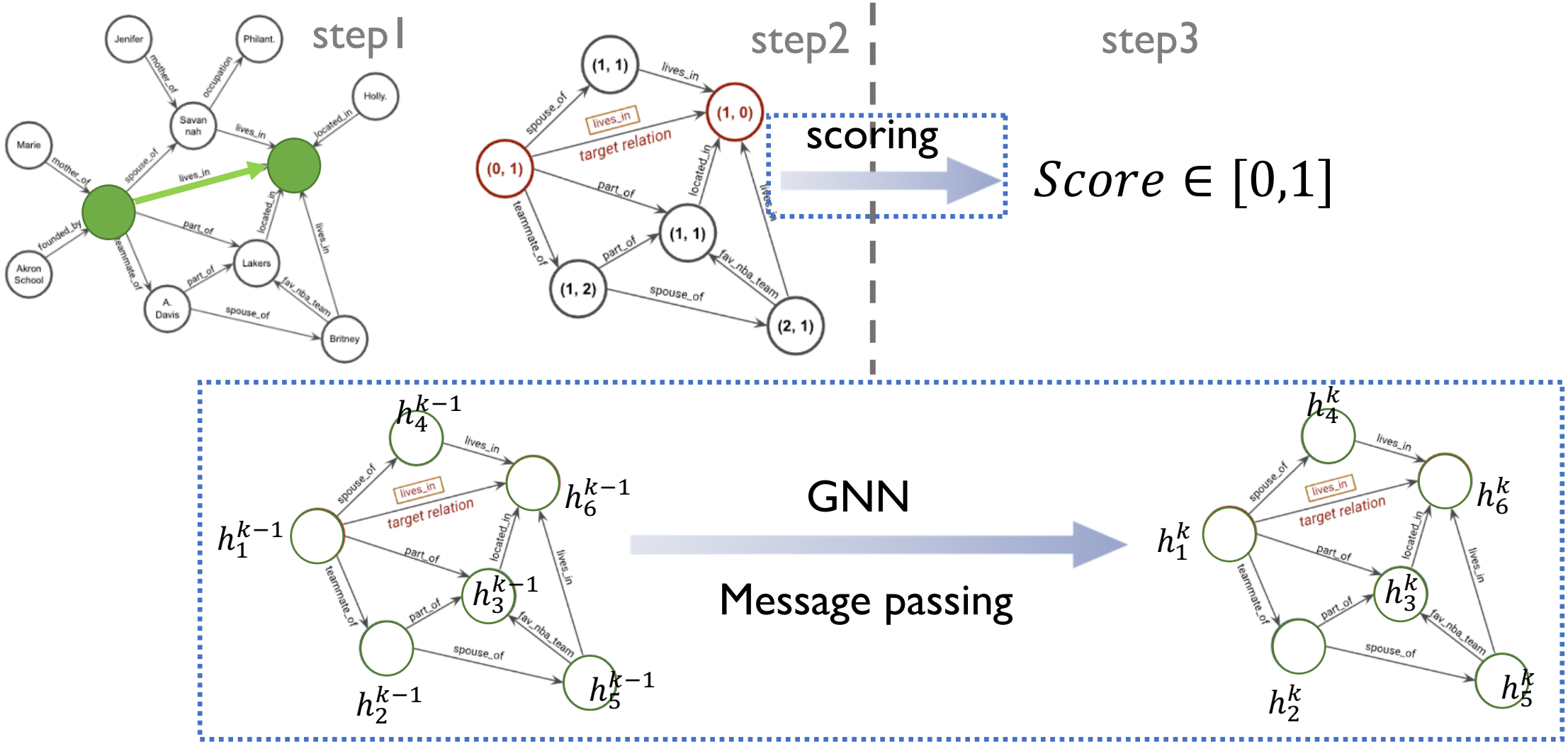
Step 3: Subgraph scoring

Two key components
- subgraph preparing (step1 & 2)
- subgraph scoring (step 3)



GraIL | Technical details

Step 3: Subgraph scoring

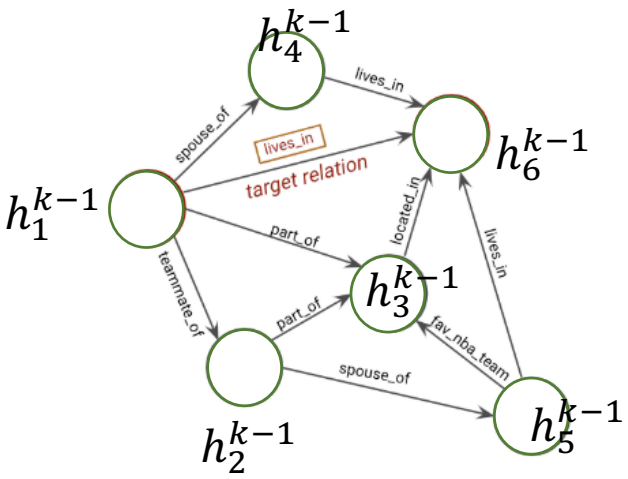


GraIL | Technical details

Step 3: Subgraph encoding and scoring

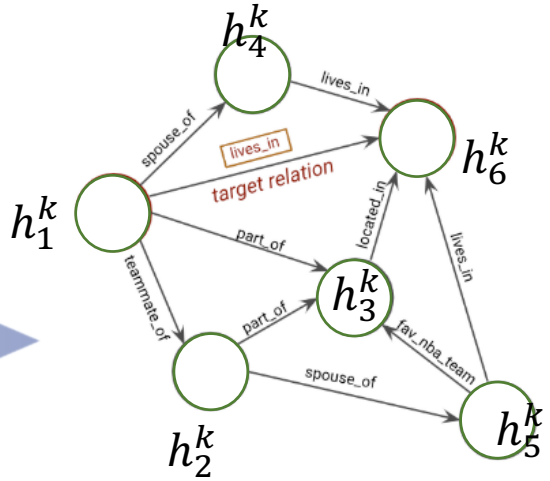
h_t^{k-1} : representation of node t at layer (k-1)

h_t^k : representation of node t at layer k



$$\mathbf{a}_t^k = \text{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right)$$
$$\mathbf{h}_t^k = \text{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right),$$

Message passing

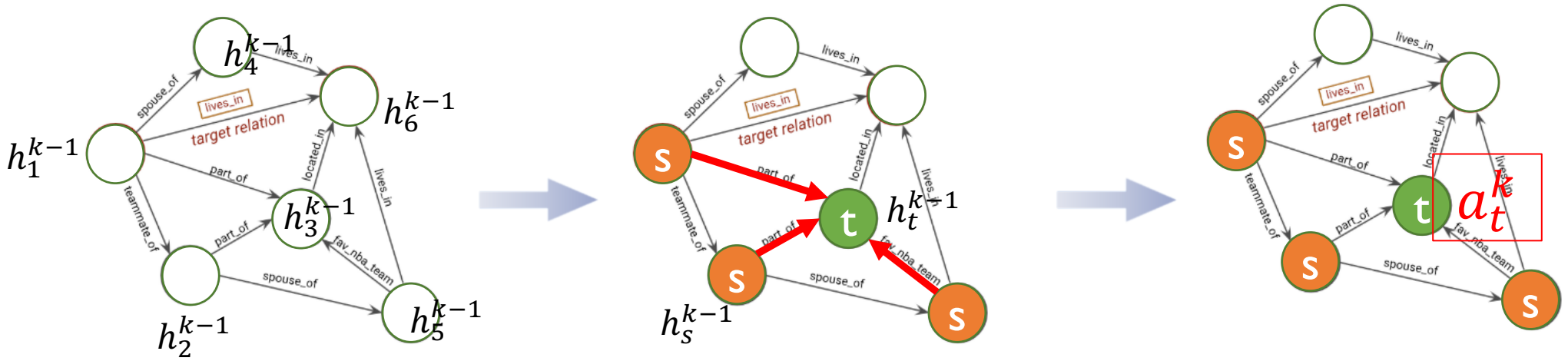


GraIL | Technical details

$$\mathbf{h}_v^{(k+1)} = \text{AGG}\left(\left\{\text{ACT}\left(\text{DROPOUT}\left(\text{BN}(\mathbf{W}^{(k)}\mathbf{h}_u^{(k)} + \mathbf{b}^{(k)})\right)\right), u \in \mathcal{N}(v)\right\}\right)$$

Design space for GNN. NeurIPS'20

Step 3: Subgraph encoding and scoring



Aggregated message at k th layer (for node t)

$$\mathbf{a}_t^k = \text{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right)$$

$$\mathbf{h}_t^k = \text{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right),$$

$$\mathbf{a}_t^k = \sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha_{rr_tst}^k \mathbf{W}_r^k \mathbf{h}_s^{k-1}$$

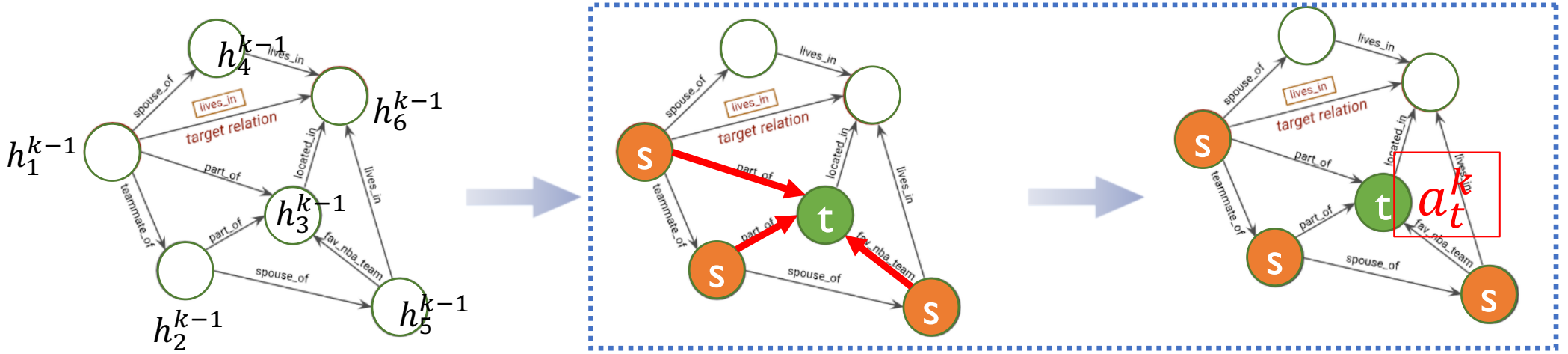
$$\mathbf{s} = \text{ReLU}\left(\mathbf{A}_1^k[\mathbf{h}_s^{k-1} \oplus \mathbf{h}_t^{k-1} \oplus \mathbf{e}_r^a \oplus \mathbf{e}_{r_t}^a] + \mathbf{b}_1^k\right)$$

$$\alpha_{rr_tst}^k = \sigma\left(\mathbf{A}_2^k \mathbf{s} + \mathbf{b}_2^k\right)$$

GraIL | Technical details

Step 3: Subgraph encoding and scoring

Aggregate operation



Aggregated message at k th layer (for node t)

$$\mathbf{a}_t^k = \text{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right)$$

$$\mathbf{h}_t^k = \text{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right),$$

$$\mathbf{a}_t^k = \sum_{r=1}^{R} \sum_{s \in \mathcal{N}_r(t)} \alpha_{rr_tst}^k \mathbf{W}_r^k \mathbf{h}_s^{k-1}$$

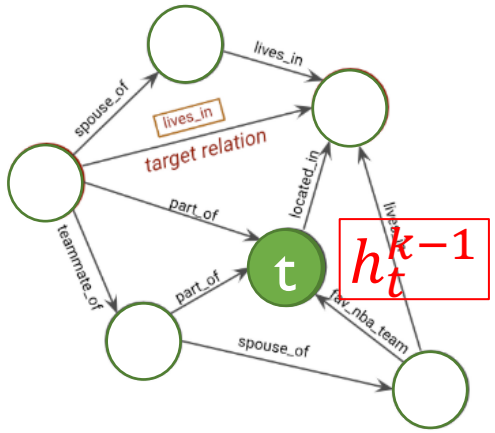
$$\mathbf{s} = \text{ReLU}\left(\mathbf{A}_1^k[\mathbf{h}_s^{k-1} \oplus \mathbf{h}_t^{k-1} \oplus \mathbf{e}_r^a \oplus \mathbf{e}_{r_t}^a] + \mathbf{b}_1^k\right)$$

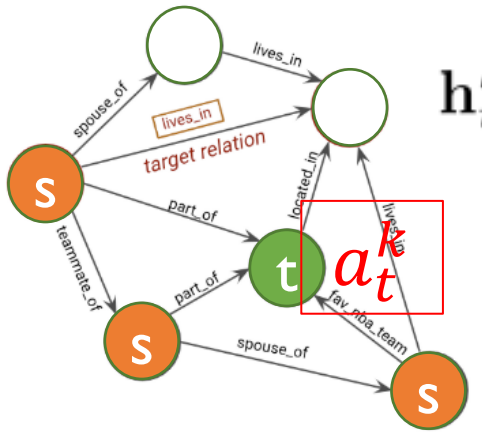
$$\alpha_{rr_tst}^k = \sigma\left(\mathbf{A}_2^k \mathbf{s} + \mathbf{b}_2^k\right)$$

Learnable parameters

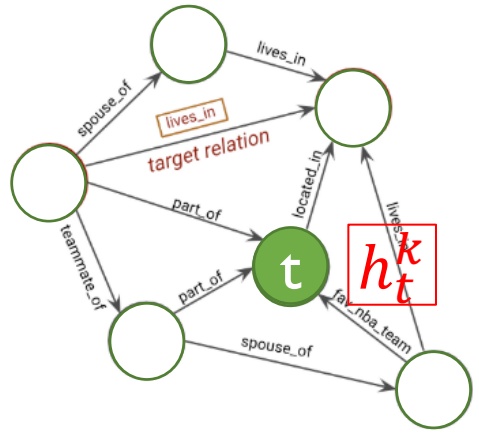
GraIL | Technical details

Step 3: Subgraph encoding and scoring





$$\mathbf{h}_t^k = \mathrm{ReLU}\left(\mathbf{W}_{self}^k \mathbf{h}_t^{k-1} + \mathbf{a}_t^k\right)$$

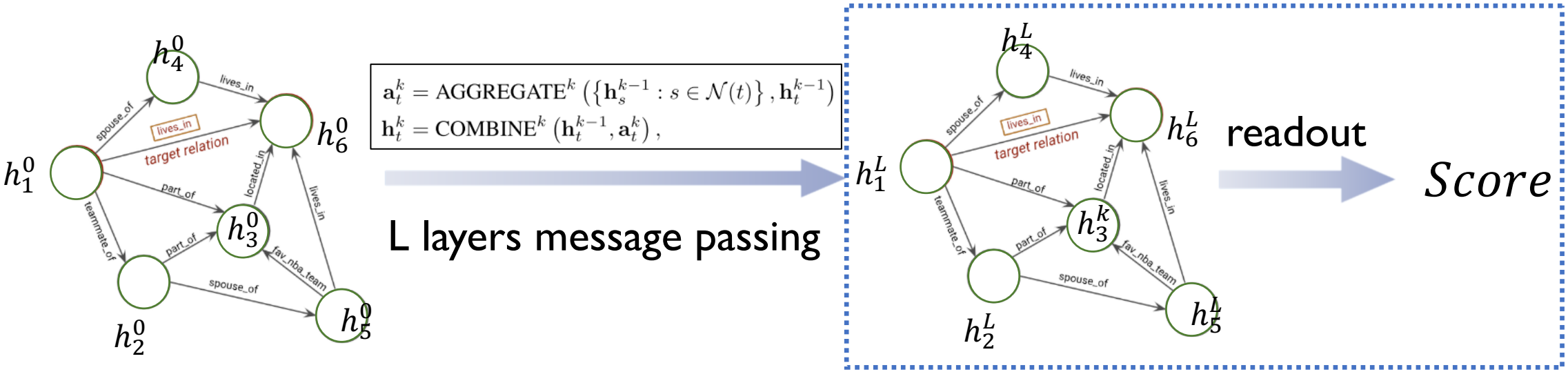


$$\mathbf{a}_t^k = \mathrm{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right)$$

$$\mathbf{h}_t^k = \mathrm{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right),$$

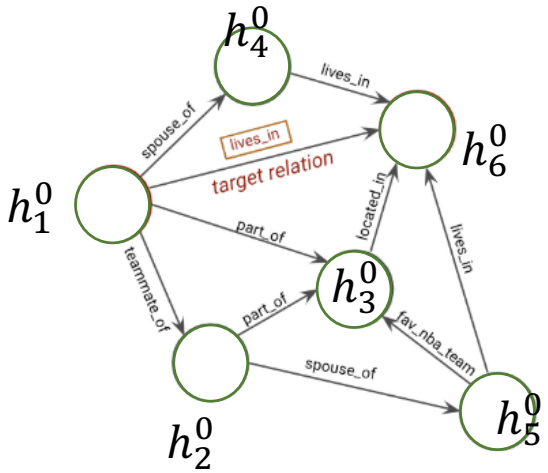
GraIL | Technical details

Step 3: Subgraph encoding and scoring



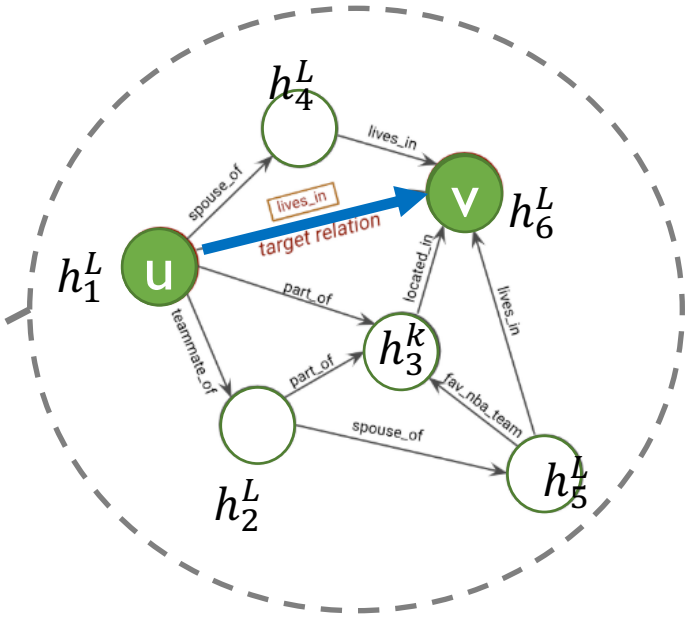
GraIL | Technical details

Step 3: Subgraph encoding and scoring



$$\mathbf{a}_t^k = \text{AGGREGATE}^k\left(\left\{\mathbf{h}_s^{k-1} : s \in \mathcal{N}(t)\right\}, \mathbf{h}_t^{k-1}\right)$$
$$\mathbf{h}_t^k = \text{COMBINE}^k\left(\mathbf{h}_t^{k-1}, \mathbf{a}_t^k\right),$$

L layers message passing



Get global information:

$$\mathbf{h}_{\mathcal{G}_{(u,v,r_t)}}^L = \frac{1}{|\mathcal{V}|}\sum_{i\in\mathcal{V}} \mathbf{h}_i^L$$

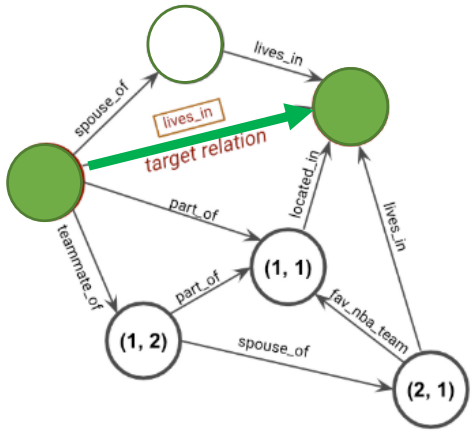
Compute the final score:

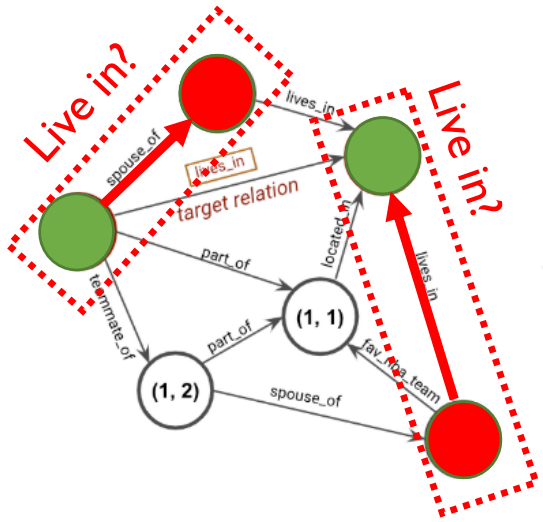
$$\text{score}(u, r_t, v) = \mathbf{W}^T[\mathbf{h}_{\mathcal{G}_{(u,v,r_t)}}^L \oplus \mathbf{h}_u^L \oplus \mathbf{h}_v^L \oplus \mathbf{e}_{r_t}]$$

GraIL | Technical details

Optimization

- sample a negative triplet by replacing the head (or tail)
- train the model to score positive triplets **higher than** negative triplets





$$\mathcal{L} = \sum_{i=1}^{|\mathcal{E}|} \max(0, \text{score}(n_i) - \text{score}(p_i) + \gamma)$$

Outline

- Background
- The proposed framework: GraIL
  - Overview
  - Technical details
  - Experiments
- Key takeaways and research directions

GraIL | Experiments

Three major parts

- Inductive relation prediction
  - *Q: How does GraIL perform in comparison to existing methods?*
- Transductive relation prediction
  - *Q: Can the complementary inductive bias improve existing KGE methods?*
- Ablation study
  - *Q: How important are the various components?*

GraIL | Experiments

Part1 | Inductive relation prediction

- Observation
  - GraIL significantly outperforms the inductive baselines across all datasets

Variant KGs for inductive setting

Table 1. Inductive results (AUC-PR)

	WN18RR				FB15k-237				NELL-995			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
Neural-LP	86.02	83.78	62.90	82.06	69.64	76.55	73.95	75.74	64.66	83.61	87.58	85.69
DRUM	86.02	84.05	63.20	82.06	69.71	76.44	74.03	76.20	59.86	83.99	87.71	85.94
RuleN	90.26	89.01	76.46	85.75	75.24	88.70	91.24	91.79	84.99	88.40	87.20	80.52
GraIL	**94.32**	**94.18**	**85.80**	**92.72**	**84.69**	**90.57**	**91.68**	**94.46**	**86.05**	**92.62**	**93.34**	**87.50**

Table 2. Inductive results (Hits@10)

	WN18RR				FB15k-237				NELL-995			
	v1	v2	v3	v4	v1	v2	v3	v4	v1	v2	v3	v4
Neural-LP	74.37	68.93	46.18	67.13	52.92	58.94	52.90	55.88	40.78	78.73	82.71	**80.58**
DRUM	74.37	68.93	46.18	67.13	52.92	58.73	52.90	55.88	19.42	78.55	82.71	**80.58**
RuleN	80.85	78.23	53.39	71.59	49.76	77.82	**87.69**	85.60	53.50	81.75	77.26	61.35
GraIL	**82.45**	**78.68**	**58.43**	**73.41**	**64.15**	**81.80**	82.83	**89.29**	**59.50**	**93.25**	**91.41**	73.19

GraIL | Experiments

Part2 | Transductive relation prediction

- Observation
  - GraIL is not SOTA for transductive prediction
    - optimized for the inductive setting
  - significant gains obtained by ensembling GraIL with various KGE methods

Late fusion

Table 3. Late fusion ensemble results on WN18RR (AUC-PR)

	TransE	DistMult	ComplEx	RotatE	GraIL
T	93.73	93.12	92.45	93.70	94.30
D		93.08	93.12	93.16	95.04
C			92.45	92.46	94.78
R				93.55	94.28
G					90.91

Early fusion

Table 6. Early fusion ensemble with TrasnE results (AUC-PR)

	WN18RR	FB15k-237	NELL-995
GraIL	90.91	92.06	97.79
GraIL++	96.20	93.91	98.11

GraIL | Experiments

Part3 | Ablation study

- Observation
  - the three components are all effective to GraIL
  - components of subgraph generation are more important

Table 7. Ablation study of the proposed framework (AUC-PR)

	FB (v3)	NELL (v3)
GraIL	**91.68**	**93.34**
GraIL w/o enclosing subgraph	84.25	85.89
GraIL w/o node labeling scheme	82.07	84.46
GraIL w/o attention in GNN	90.27	87.30

Subgraph generation

Subgraph scoring

GraIL | Theoretical analysis

- GraIL can represent a useful subset of first-order logic

Theorem 1. *Let $\mathcal{R}$ be any logical rule (i.e., clause) on binary predicates of the form:*

$$r_t(X, Y) \leftarrow r_1(X, Z_1) \wedge r_2(Z_1, Z_2) \wedge ... \wedge r_k(Z_{k-1}, Y),$$

where $r_t, r_1, ..., r_k$ are (not necessarily unique) relations in the knowledge graph, $X, Z_1, ..., Z_k, Y$ are free variables that can be bound by arbitrary unique entities. For any such $\mathcal{R}$ there exists a parameter setting Θ for a GraIL model with k GNN layers and where the dimension of all latent embeddings are $d = 1$ such that

$$score(u, r_t, v) \neq 0$$

if and only if $\exists Z_1, ..., Z_k$ where the body of $\mathcal{R}$ is satisfied with $X = u$ and $Y = v$.

Corollary 1. *Let $\mathcal{R}_1 ..., \mathcal{R}_m$ be a set of logical rules with the same structure as in Theorem 1 where each rule has the same head $r_t(X, Y)$. Let*

$$\beta = |\{\mathcal{R}_i \ : \exists Z_1, ..., Z_k \text{ where } \mathcal{R}_i = true \text{ with } X = u \text{ and } Y = v\}|.$$

Then there exists a parameter setting for GraIL with the same assumptions as Theorem 1 such that

$$score(u, r_t, v) \propto \beta.$$

Rules discovered by other rule-mining methods

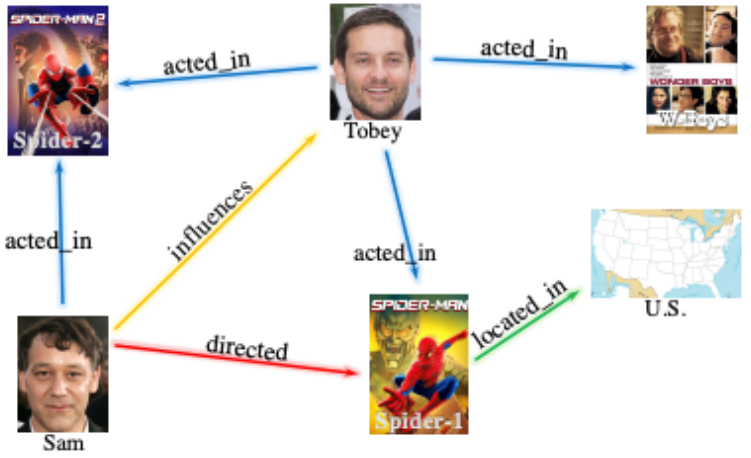
brother(B, A) ← sister(A, B)
brother(C, A) ← sister(A, B), sister(B, C)
brother(C, A) ← brother(A, B), sister(B, C)
brother(C, A) ← nephew(A, B), uncle(B, C)
brother(C, A) ← nephew(A, B), nephew(C, B)
brother(C, A) ← brother(A, B), sister(B, C)

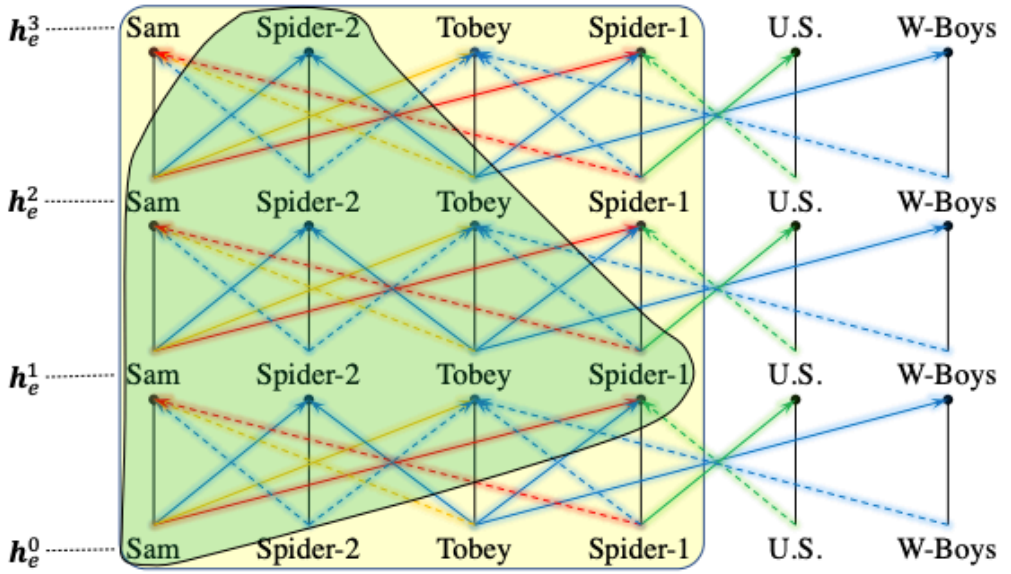
GraIL | Disadvantages

- Efficiency $\mathcal{O}(\log(\mathcal{V})\mathcal{E} + \mathcal{R}dk)$
  - Time-consuming node labeling

- Effectiveness
  - The information flow inside the enclosing subgraph is mixed

    (making it hard to learn the dependency among relations)
  - Not interpretable or traceable





Knowledge Graph Reasoning with Relational Directed Graph. Arxiv 2021.

Outline

- Background
- The proposed framework: GraIL
- Key takeaways and research directions

Key takeaways and Research directions

Summary of GraIL

- Presents a GNN framework that has a strong inductive bias to learn entity-independent relational semantics.
- Introduces a series of benchmark tasks for the inductive relation prediction problem.
- Empirically justify the effectiveness of the proposed method in both of transductive and inductive settings.

Main contributions
- GNN + inductive for KG reasoning
- Variant KGs for inductive setting
- SOTA in inductive setting

Following works:

[1] Graph meta learning via local subgraphs. 2020.

[2] Inductive Learning on Commonsense Knowledge Graph Completion. 2021.

[3] Standing on the Shoulders of Predecessors: Meta-Knowledge Transfer for Knowledge Graphs. 2021.

Direction: Decoupling of structure generation and scoring

Summary of existing works

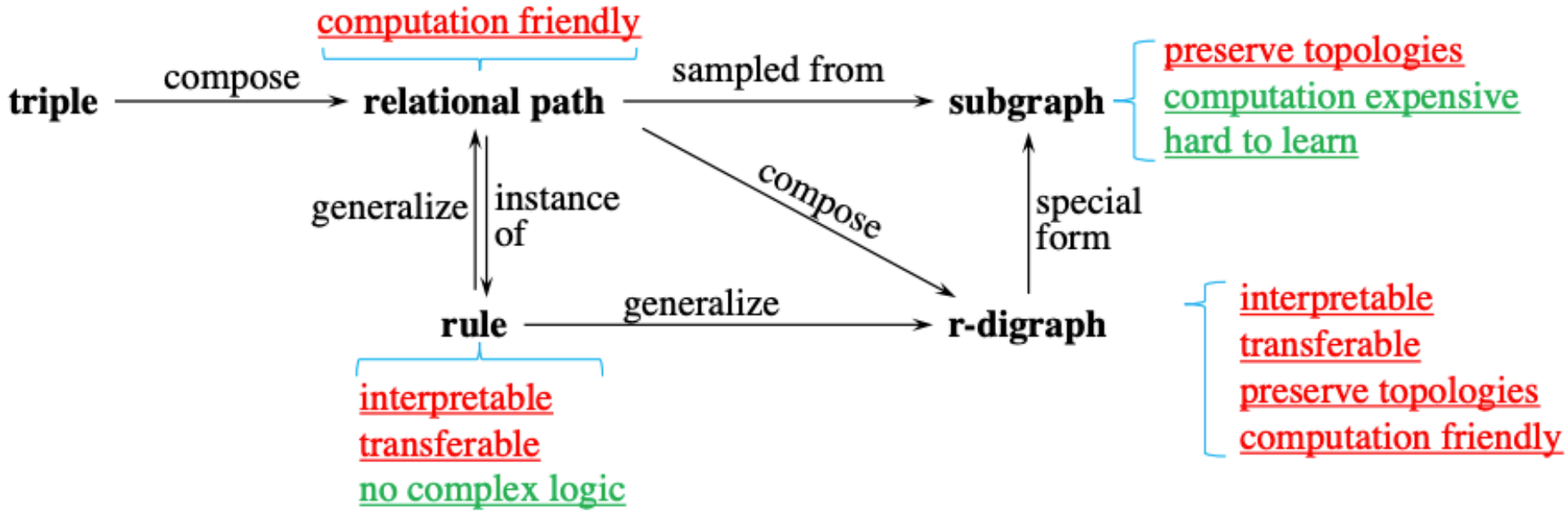
	Structure Generation	Structure Scoring

Forward Inference	• Path
	  • using RL agent to pick up next edge
	  • select path with max length [PathCon]
	• Subgraph
	  • enclosing subgraph from h to t [GraIL]
	  • relation di-graph [RED-GNN]
	  • generalized Bellman-Ford algorithm [NBFNet]
		• RNN / (MLP)
		  • used as policy network in RL agent → encodes the state (i.e., history of the walked path) and then take action (decode)
		• GNN
		  • uniquely label nodes and pass messages to obtain final score [GraIL]
		  • assign an independent embedding s_p for each path $P \in P_{h \to t}$, combine context presentation with message passing [PathCon]
		  • local+global with {indicator + message + aggregate} functions [NBFNet]

Backward Optimization	• RL agent (sparse reward / slow convergence)
	  • BFS + Monte-Carlo policy gradient (REINFORCE)
	• EM algorithm
	  • generate rules from a pre-defined distribution
	  • to select good generated rules and then optimize the rule generator [RNNLogic]
	  • EM + pure embedding model [pLogicNet]
		• RNN
		• GNN
		  • [negative sampling] sampled a negative triple and minimize a MR loss [GraIL]
		  • 1/k vs all training

Direction: Decoupling of structure generation and scoring

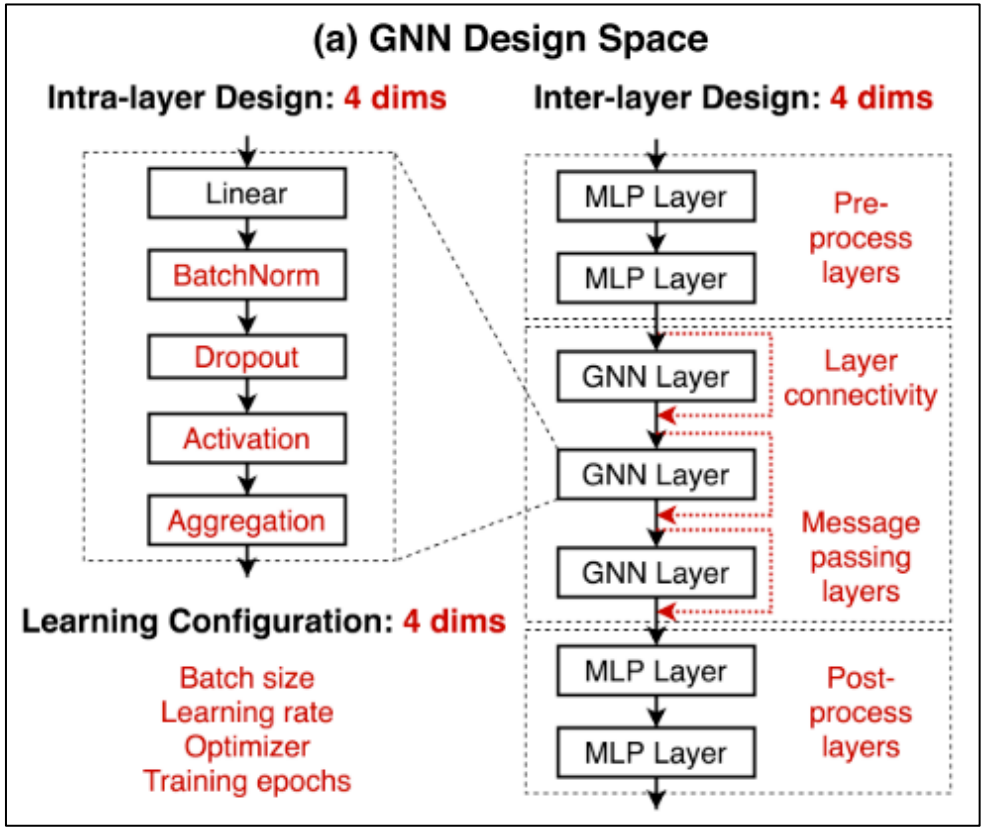
1. From the aspect of structure generation:



Knowledge Graph Reasoning with Relational Directed Graph. Arxiv 2021.

Direction: Decoupling of structure generation and scoring

2. From the aspect of structure scoring:



[1]

Algorithm 1 Neural Bellman-Ford Networks

Input: source node u, query relation q, #layers T
Output: pair representations $\boldsymbol{h}_q(u, v)$ for all $v \in \mathcal{V}$

for $v \in \mathcal{V}$ **do** ▷ Boundary condition
  $\boldsymbol{h}_v^{(0)} \leftarrow \textsc{Indicator}(u, v, q)$
end for
for $t \leftarrow 1$ to T **do** ▷ Bellman-Ford iteration
  for $v \in \mathcal{V}$ **do**
    $\mathcal{M}^{(t)} \leftarrow \left\{\boldsymbol{h}_v^{(0)}\right\}$ ▷ Message augmentation
    for $(x, r, v) \in \mathcal{E}(v)$ **do**
      $\boldsymbol{m}_{(x,r,v)}^{(t)} \leftarrow \textsc{Message}^{(t)}(\boldsymbol{h}_x^{(t-1)}, \boldsymbol{w}_q(x, r, v))$
      $\mathcal{M}^{(t)} \leftarrow \mathcal{M}^{(t)} \cup \left\{\boldsymbol{m}_{(x,r,v)}^{(t)}\right\}$
    end for
    $\boldsymbol{h}_v^{(t)} \leftarrow \textsc{Aggregate}^{(t)}(\mathcal{M}^{(t)})$
  end for
end for
return $\boldsymbol{h}_v^{(T)}$ as $\boldsymbol{h}_q(u, v)$ for all $v \in \mathcal{V}$

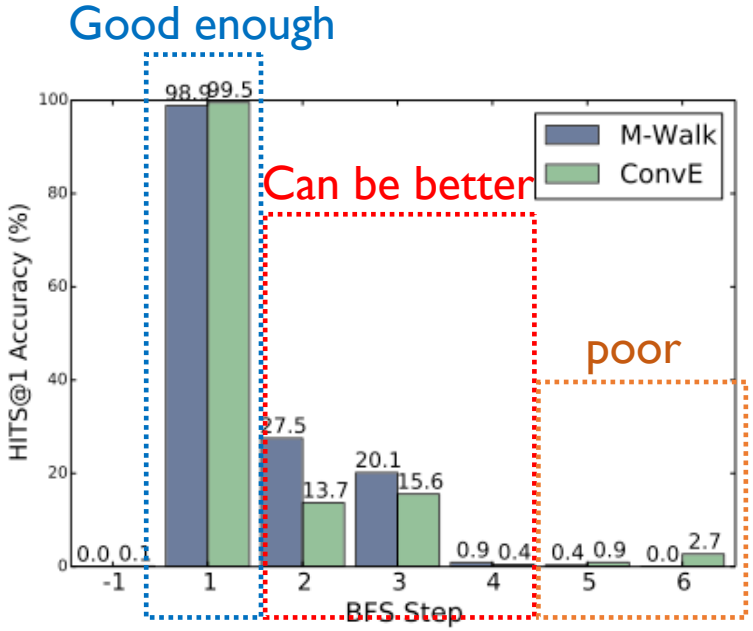
[2]

[1] Design Space for Graph Neural Networks
[2] Neural Bellman-Ford Networks: A General Graph Neural Network Framework for Link Prediction

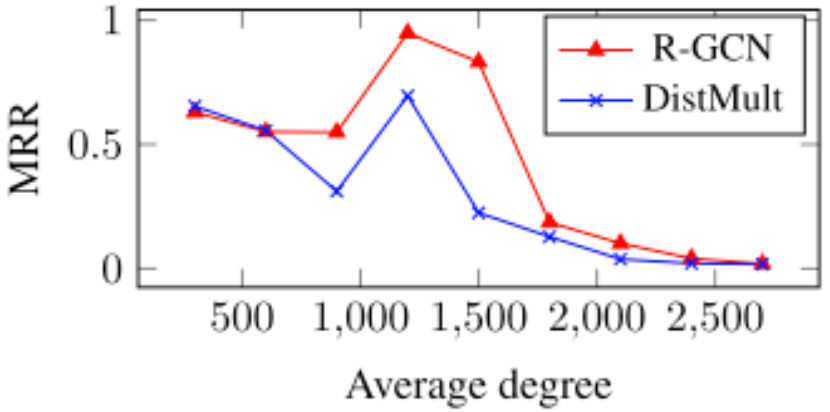
Direction: Decoupling of structure generation and scoring

Potential advantages for structure learning models

- more robust and effective to distant triplets (e.g., hop>1)
- performs better for nodes with high degree (where contextual information is abundant).



[1]



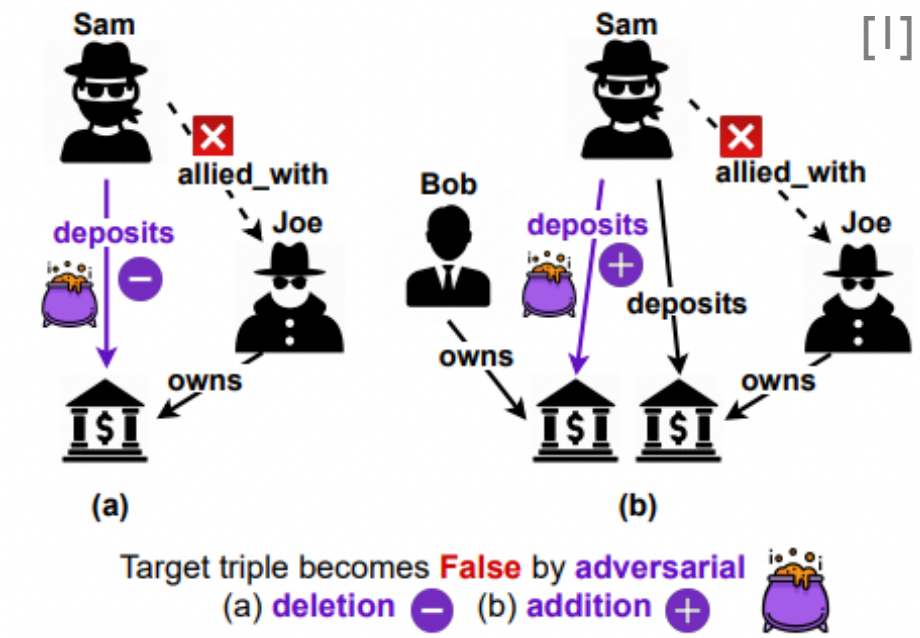
[2]

[1] M-Walk: Learning to Walk over Graphs using Monte Carlo Tree Search. 2018.
[2] Modeling Relational Data with Graph Convolutional Networks. 2017.

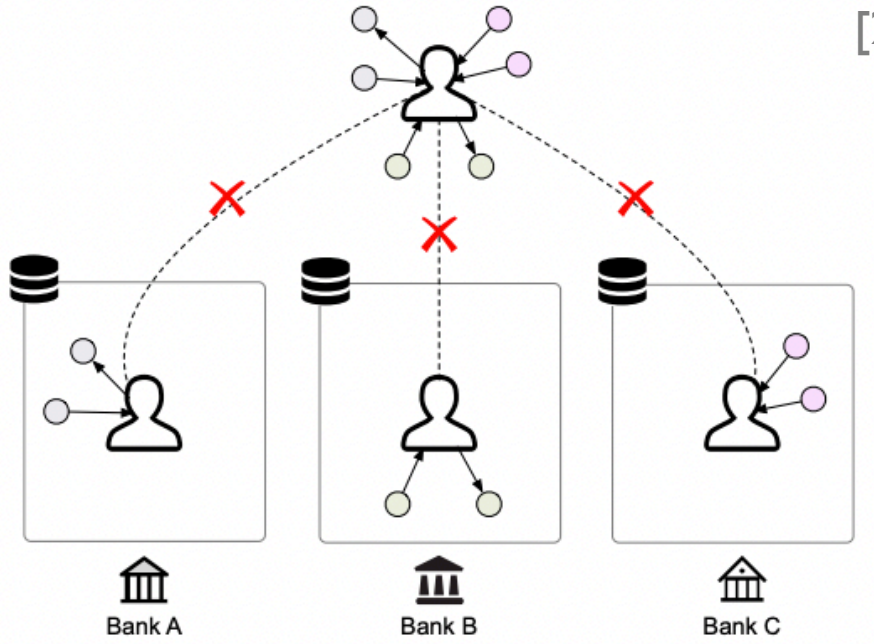
Direction: Decoupling of structure generation and scoring

3. From the aspect of task:

- Adversarial attack on KG
- Transfer / federated learning with KG



[1]



[2]

[1] Adversarial Attacks on Knowledge Graph Embeddings via Instance Attribution Methods. EMNLP 2021.
[2] FedE: Embedding Knowledge Graphs in Federated Setting. Arxiv 2020.

Q&A

Thanks for your attention!

Appendix: data statistics

Table 13. Statistics of inductive benchmark datasets

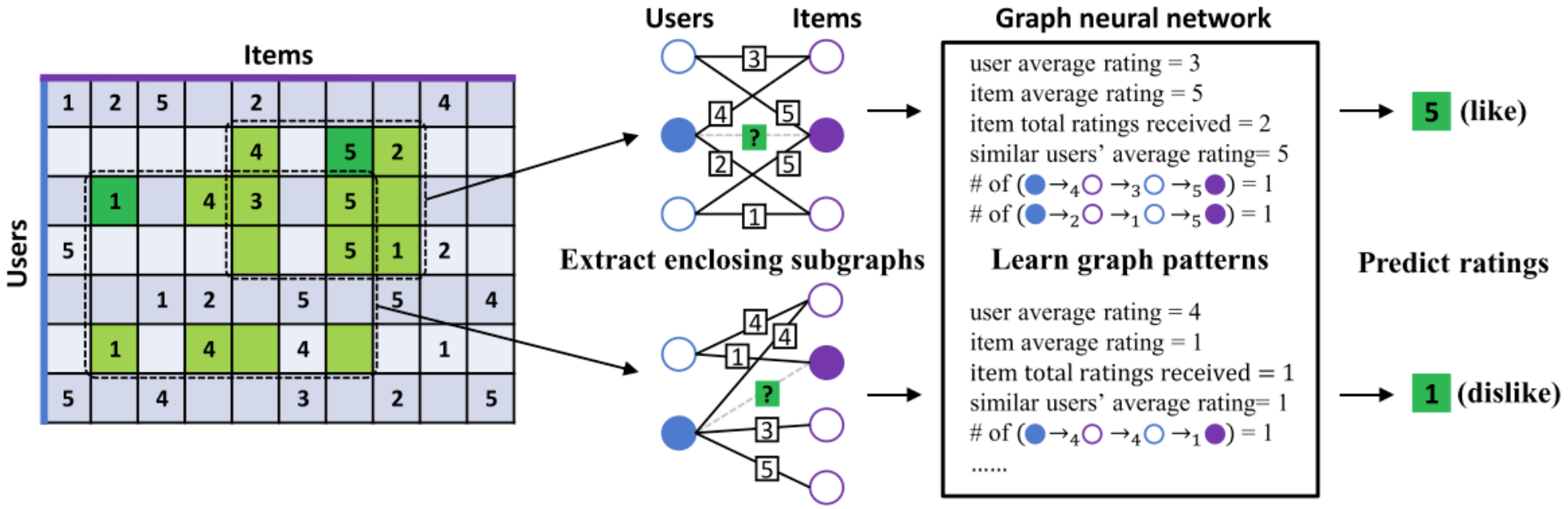
		WN18RR			FB15k-237			NELL-995		
		#relations	#nodes	#links	#relations	#nodes	#links	#relations	#nodes	#links
v1	train	9	2746	6678	183	2000	5226	14	10915	5540
	ind-test	9	922	1991	146	1500	2404	14	225	1034
v2	train	10	6954	18968	203	3000	12085	88	2564	10109
	ind-test	10	2923	4863	176	2000	5092	79	4937	5521
v3	train	11	12078	32150	218	4000	22394	142	4647	20117
	ind-test	11	5084	7470	187	3000	9137	122	4921	9668
v4	train	9	3861	9842	222	5000	33916	77	2092	9289
	ind-test	9	7208	15157	204	3500	14554	61	3294	8520

Appendix: readout function

$$\text{score}(u, r_t, v) = \mathbf{W}^T [\mathbf{h}^L_{\mathcal{G}_{(u,v,r_t)}} \oplus \mathbf{h}^L_u \oplus \mathbf{h}^L_v \oplus \mathbf{e}_{r_t}]$$

$$\text{score}(u, r_t, v) = \mathbf{W}^T \bigoplus_{i=1}^{L} [h^i_{\mathcal{G}_{(u,v,r_t)}} \oplus h^i_u \oplus h^i_v \oplus e_{r_t}]$$

Appendix: introduction



[1] INDUCTIVE MATRIX COMPLETION BASED ON GRAPH NEURAL NETWORKS. ICLR 2020.

Appendix: KGE training and testing

- Training
  - S^+: $positive\ samples$ S^-: $negative\ samples$
  - Objectives: $max\ \phi^+(S^+) + \phi^-(S^-)$
- Prediction
  - For link prediction, given an incomplete triple $(h, r, ?)$
  - the missing tail is inferred as the entity that results in the highest score:

$$t = \underset{e \in \mathcal{E}}{\operatorname{argmax}}\, \phi(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{e})$$

- Evaluation
  - Three common metrics
  - ***q*:** the rank of correct entity

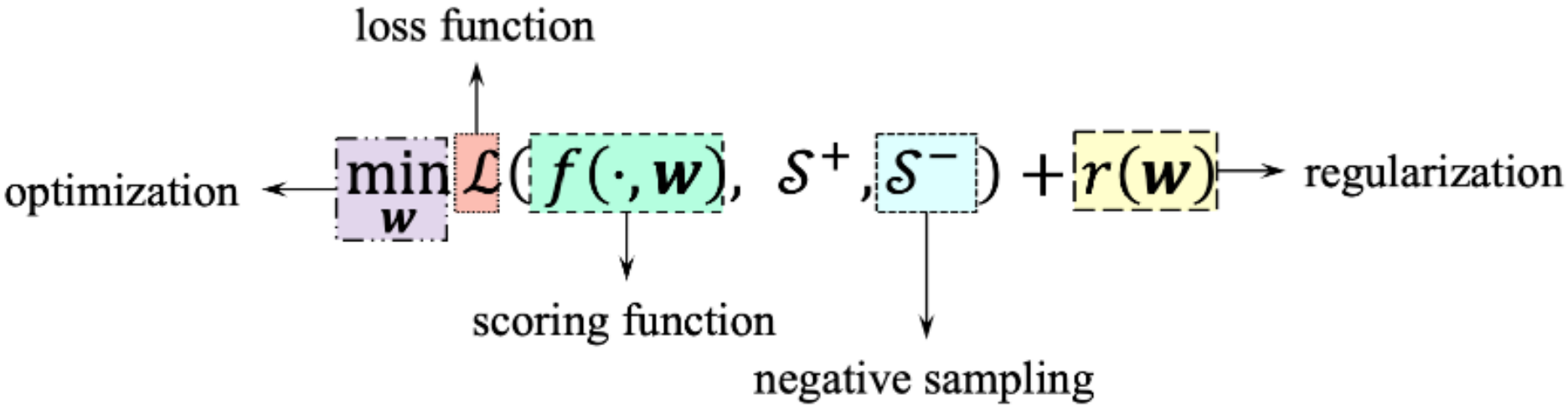
$$MR = \frac{1}{|Q|} \sum_{q \in Q} q \qquad MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{q} \qquad H@K = \frac{|\{q \in Q : q \leq K\}|}{|Q|}$$

Appendix: KGE training and testing

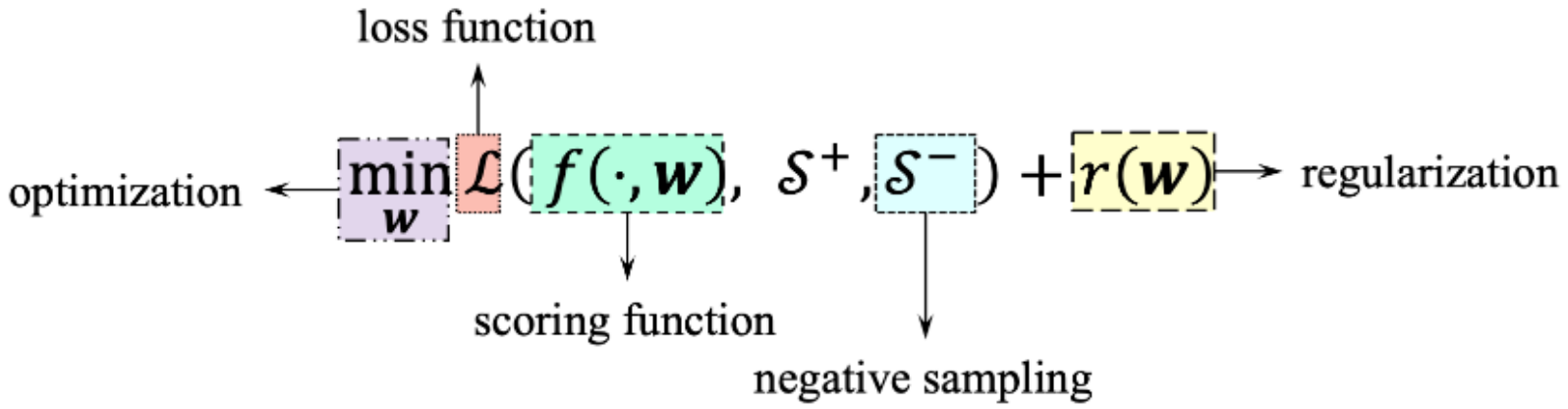
For obtaining embeddings of entities and relations, and finishing KGE task (e.g., predict potential facts)

- **the scoring function f** is expected to discriminate positive/negative factual triples
- a sampling scheme is needed to generate **negative samples S^-**
- a **loss function L** and **regularization r** are required for defining learning problem
- a **optimization** strategy is needed for convergence procedure

Considering the above 5 factors,
we can formulate the KGE learning framework as:



Appendix: KGE training and testing



5 KGE components:

Component	Role	Inputs	Outputs	Example
Scoring function $f()$	Plausibility estimation	Factual triples	Scores for each triple	TransE / CP
Loss function $L()$	Learning problem definition	Scores and labels	Loss value	BCE / CE
Negative sampling	Budget tradeoff	Positive samples S^+	Negative samples S^-	Uniform
Regularization $r()$	Avoid overfitting	Learnable parameters	Regularization value	L2 / N3
Optimization	Convergence control	-	-	-