Learning Symbolic Logic Rules for Reasoning on Knowledge Graphs

Jian Tang

Mila-Quebec AI Institute

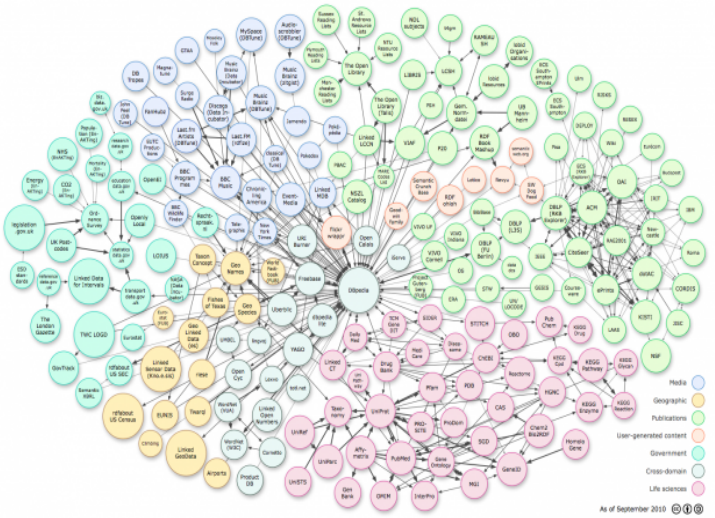
CIFAR AI Chair, HEC Montreal

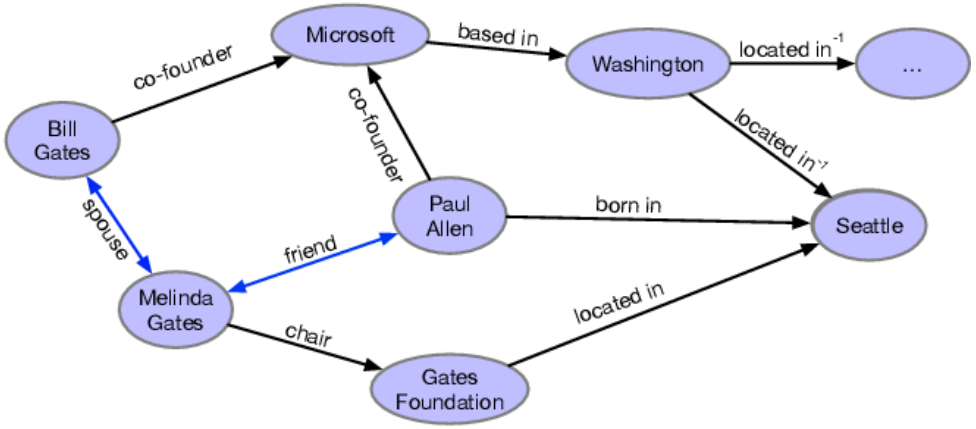
www.jian-tang.com

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, Jian Tang. **RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs**. ***To appear*** **in ICLR'21.**

Knowledge Graphs

- Knowledge graphs : a set of facts represented as triplets
  - <head entity, relation, tail entity>
- Important in a variety of applications
  - Online search
  - Question answering
  - Recommender systems





Google Knowledge Graph

Microsoft 悟 SATORI

NELL: Never-Ending Language Learning

Freebase

DBpedia

yago select knowledge

OpenIE (Reverb, OLLIE)

Reasoning on Knowledge Graphs

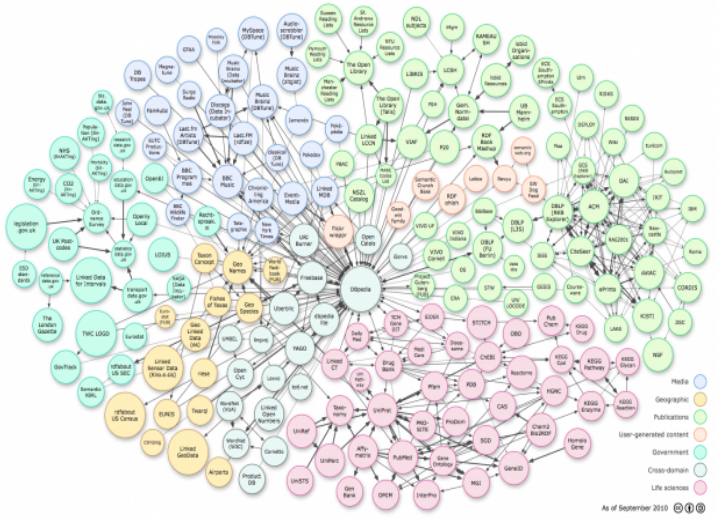
- Knowledge graphs are usually incomplete. Many facts are missing
- A fundamental task: **predicting missing links (or facts) by reasoning on existing facts**
- The Key Idea: leverage **logic rules** for reasoning on knowledge graphs implicitly or explicitly
- Example:

A's husband is B => B's wife is A

Parents of Parents are Grandparents

Logical Rule Induction/Learning

- Logical rules are usually not available, how to infer logical rules from knowledge graphs?
  - Inductive logic programming
  - Neural logic programming



$$\texttt{Appears\_in\_TV\_Show}(X,Y) \leftarrow \texttt{Has\_Actor}(X,Y)$$
$$\texttt{Appears\_in\_TV\_Show}(X,Y) \leftarrow \texttt{Creator\_of}(X,U) \wedge \texttt{Has\_Producer}(U,V) \wedge \texttt{Appears\_in\_TV\_Show}(V,Y)$$

$$\texttt{ORG.\_in\_State}(X,Y) \leftarrow \texttt{ORG.\_in\_City}(X,U) \wedge \texttt{City\_Locates\_in\_State}(U,Y)$$
$$\texttt{ORG.\_in\_State}(X,Y) \leftarrow \texttt{ORG.\_in\_City}(X,U) \wedge \texttt{Address\_of\_PERS.}(U,V) \wedge \texttt{Born\_in}(V,W) \wedge \texttt{Town\_in\_State}(W,Y)$$

$$\texttt{Person\_Nationality}(X,Y) \leftarrow \texttt{Born\_in}(X,U) \wedge \texttt{Place\_in\_Country}(U,Y)$$
$$\texttt{Person\_Nationality}(X,Y) \leftarrow \texttt{Student\_of\_Educational\_Institution}(X,U) \wedge \texttt{ORG.\_Endowment\_Currency}(U,V) \wedge \texttt{Currency\_Used\_in\_Region}(V,W) \wedge \texttt{Region\_in\_Country}(W,Y)$$

Related Work

- **Inductive Logic Programming**:
  - Key idea: generate-and-test
  - Generate a set of candidate logic rules for reasoning
  - Choose the most useful logic rules from all candidates
- Limitations: inability to handle noisy, erroneous or ambiguous data

Related Work

- **Neural inductive logic programming**: combines the advantages of ILP and neural network-based systems:
  - data efficient
  - Robust to noisy and ambiguous data
- Differentiable ILP (Evans et al. 2017)
  - Consider a large number of candidate logic rules
  - Learn the weights of these rules jointly
- Limitations:
  - Large search space
  - The weights may not reflect the important of rules precisely

RNNLogic (Qu and Chen et al. 2020)

- A new logic rule learning approach called RNNLogic:
  - Treating a set of logic rules as **a latent variable**
  - A **rule generator** for generating candidate logic rules (prior)
  - A **reasoning predictor** with logic rules (likelihood)
- RNNLogic is able to effectively perform search in the search space
- An effective EM algorithm for optimizing RNNLogic
- Outperforms many competitive rule learning methods and knowledge graph embedding methods on several benckmark datasets

Qu, Meng*, Chen, Junkun*, Xhonneux Louis-Pascal, Bengio Yoshua, and Tang, Jian. **"RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs."** To appear ICLR'21.

Chain-like Rules

- Rules with a chain structure:
  - $\mathrm{r}(X_0, X_l) \leftarrow \mathrm{r}_1(X_0, X_1) \wedge \mathrm{r}_2(X_1, X_2) \wedge \cdots \wedge \mathrm{r}_l(X_{l-1}, X_l)$
- Example:
  - $\mathrm{Nationality}(X, Y) \leftarrow \mathrm{LiveIn}(X, Z) \wedge \mathrm{CityOf}(Z, Y)$
  - $\mathrm{GrandFather}(X, Y) \leftarrow \mathrm{Father}(X, Z) \wedge \mathrm{Father}(Z, Y)$
- Chain-like rules capture:
  - Composition
  - Symmetric relations $\mathrm{r}(X, Y) \leftarrow \mathrm{r}^{-1}(X, Y)$ with r^{-1} the inverse relation of r
  - Inverse relations $\mathrm{r}(X, Y) \leftarrow \mathrm{r}_I^{-1}(X, Y)$ with r_I^{-1} the inverse relation of r_I

Probabilistic Formalization

- Problem:
  - Input: a query $\boldsymbol{q} = (h, \mathrm{r}, ?)$, a background knowledge graph $\mathcal{G}$
  - Output: the answer $\boldsymbol{a} = t$
  - The goal is to model $p(\boldsymbol{a}|\mathcal{G}, \boldsymbol{q})$
- Probabilistic formalization:
  - Treat a set of chain-like logic rules as a latent variable $\boldsymbol{z}$

$$p_{w,\theta}(\boldsymbol{a}|\mathcal{G}, \boldsymbol{q}) = \sum_{\boldsymbol{z}} p_w(\boldsymbol{a}|\mathcal{G}, \boldsymbol{q}, \boldsymbol{z}) p_\theta(\boldsymbol{z}|\boldsymbol{q}) = \mathbb{E}_{p_\theta(\boldsymbol{z}|\boldsymbol{q})}[p_w(\boldsymbol{a}|\mathcal{G}, \boldsymbol{q}, \boldsymbol{z})]$$

Likelihood from a Reasoning Predictor p_w — Prior from a Rule Generator p_θ

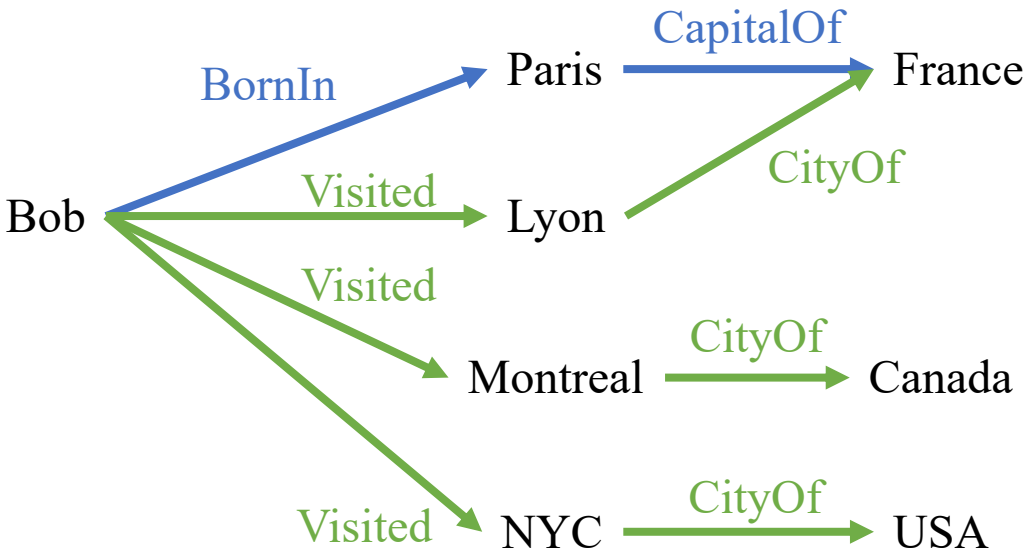
  - Objective function: $\max_{w,\theta} \mathcal{O}(w, \theta) = \mathbb{E}_{(\mathcal{G},\boldsymbol{q},\boldsymbol{a})\sim p_{\text{data}}}[\log p_{w,\theta}(\boldsymbol{a}|\mathcal{G}, \boldsymbol{q})]$

Rule Generator $p_\theta(z|q)$

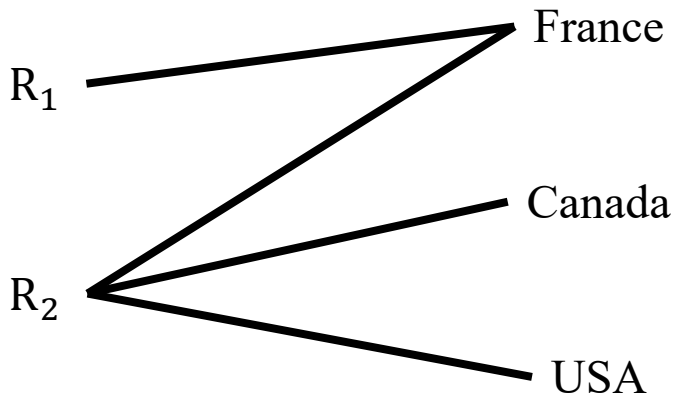
- Each chain-like rule can be represented as a sequence of relations:
  - $\mathrm{r}(X_0, X_l) \leftarrow \mathrm{r}_1(X_0, X_1) \wedge \mathrm{r}_2(X_1, X_2) \wedge \cdots \wedge \mathrm{r}_l(X_{l-1}, X_l)$
  - $[\mathrm{r}, \mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_l, \mathrm{r}_{\mathrm{END}}]$ where $\mathrm{r}_{\mathrm{END}}$ is a special ending relation
- Such sequences can be effectively generated by an RNN
  - The probability of each rule can be simultaneously computed
  - $p(rule) = \mathrm{RNN}_\theta(rule|\mathrm{r})$
- For a query $\boldsymbol{q} = (h, \mathrm{r}, ?)$, define the prior over a set of rules $\boldsymbol{z}$ as:
  - $p_\theta(\boldsymbol{z}|\boldsymbol{q}) = \mathrm{Mu}(\boldsymbol{z}|N, \mathrm{RNN}_\theta(\cdot\,|\mathrm{r}))$ where Mu is multinomial distribution
  - Generative process of $\hat{\boldsymbol{z}} \sim p_\theta(\boldsymbol{z}|\boldsymbol{q})$:
    - Generate N chain-like rules with RNN_θ, form $\hat{\boldsymbol{z}}$ with these rules.

Reasoning Predictor $p_w(a|\mathcal{G}, q, z)$

- For each query $q = (h, \mathrm{r}, ?)$, we can use rules in z to get a search tree:
  - Query: $q = (\mathrm{Bob}, \mathrm{Nationality}, ?)$
  - Logic rules in z:
    - R_1: Nationality ← BornIn ∧ CapitalOf R_2: Nationality ← Visited ∧ CityOf



Each logic rule finds some candidate answers



Reasoning Predictor $p_w(a|\mathcal{G}, q, z)$

- Assign a score to each candidate answer according to the corresponding logic rules:
  - Bob $\rightarrow$ R_1: BornIn $\wedge$ CapitalOf $\rightarrow$ France
  - Bob $\rightarrow$ R_2: Visited $\wedge$ CityOf $\rightarrow$ France

$$\text{Score}(\text{France}) = \psi_w(R_1)\phi_w(\text{Bob}, \text{BornIn}, \text{CapitalOf}, \text{France}) + \psi_w(R_2)\phi_w(\text{Bob}, \text{Visited}, \text{CityOf}, \text{France})$$

Scalar weight of each rule

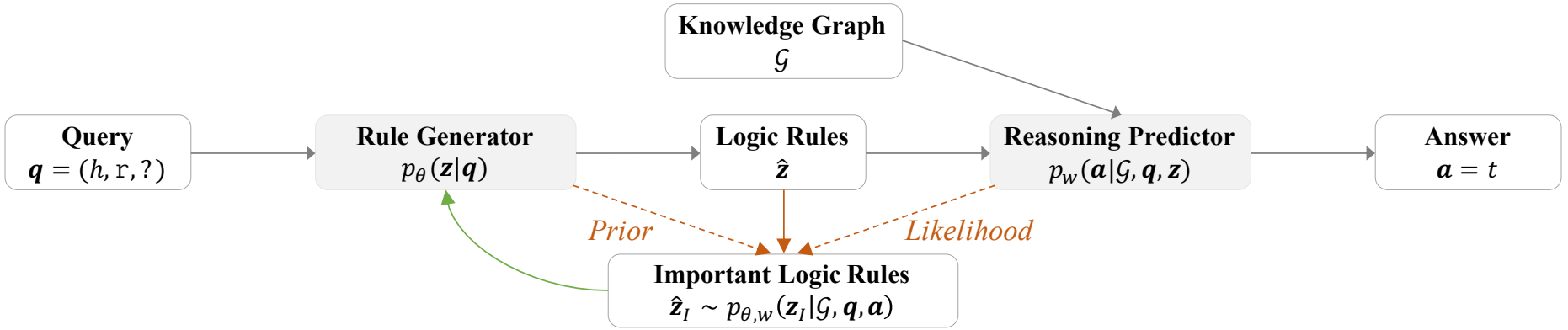
Score of each relational path, either a constant or computed with embeddings

$$p_w(\boldsymbol{a} = \text{France}|\mathcal{G}, \boldsymbol{q}, \boldsymbol{z} = (R_1, R_2)) = \frac{\exp\big(\text{Score}(\text{France})\big)}{\exp\big(\text{Score}(\text{France})\big) + \exp\big(\text{Score}(\text{Canada})\big) + \exp\big(\text{Score}(\text{USA})\big)}$$

Softmax over all candidate answers

Optimization

- An EM algorithm:



  - In each iteration:
    - Explore a set of logic rules $\hat{\boldsymbol{z}}$ from the rule generator p_θ
    - E-step: Identify a subset of important rules based on posterior $p_{\theta,w}(\boldsymbol{z}_I|\mathcal{G}, \boldsymbol{q}, \boldsymbol{a})$
    - M-step: Update p_θ and p_w according to the selected important rules

Optimization E-step

- Goal of E-step:
  - Identify a set of most important rules
- Posterior inference:
  - Compute the posterior distribution ($\boldsymbol{z}_I \subset \hat{\boldsymbol{z}}$ is a subset of all the generated rules):

$$p_{\theta,w}(\boldsymbol{z}_I|\mathcal{G},\boldsymbol{q},\boldsymbol{a}) \propto p_w(\boldsymbol{a}|\mathcal{G},\boldsymbol{q},\boldsymbol{z}_I)p_\theta(\boldsymbol{z}_I|\boldsymbol{q})$$

Posterior — Likelihood from p_w — Prior from p_θ

  - Infer $\hat{\boldsymbol{z}}_I = \arg\max_{\boldsymbol{z}_I} p_{\theta,w}(\boldsymbol{z}_I|\mathcal{G},\boldsymbol{q},\boldsymbol{a})$ as the most important rules
    - A set of logic rules with the maximum posterior probability

Optimization M-step

- Goal of M-step:
  - Use the identified important rules $\hat{\boldsymbol{z}}_I$ to update the reasoning predictor p_w and rule generator p_θ
- For each query $\boldsymbol{q} = (h, \mathrm{r}, ?)$ and answer $\boldsymbol{a} = t$:
  - Reasoning predictor:
    - Maximize $\log p_w(\boldsymbol{a} = t|\mathcal{G}, \boldsymbol{q}, \hat{\boldsymbol{z}}_I)$
  - Rule generator:
    - Maximize $\log p_\theta(\hat{\boldsymbol{z}}_I|\boldsymbol{q}) = \sum_{rule \in \hat{\boldsymbol{z}}_I} \log \mathrm{RNN}_\theta(rule|r)$

Increase the probability of each identified important logic rule

Experimental Setup

- Data:
  - A set of (h, r, t)-triplets $\mathcal{T}$
- Training:
  - Randomly sample a $(h, r, t) \in \mathcal{T}$
  - Form the question and answer as $\boldsymbol{q} = (h, \mathrm{r}, ?)$ and $\boldsymbol{a} = t$
  - Form the background knowledge graph as $\mathcal{G} = \mathcal{T} \setminus (h, r, t)$
  - Treat $(\mathcal{G}, \boldsymbol{q}, \boldsymbol{a})$ as each training instance
- Testing:
  - Form the background knowledge graph as $\mathcal{G} = \mathcal{T}$

Main Results on FB15k-237 and WN18RR

- RNNLogic outperforms all rule learning methods
- RNNLogic achieves comparable results to state-of-the-art knowledge graph embedding methods

Category	Algorithm	FB15k-237					WN18RR				
		MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
No Rule Learning	TransE*	357	0.294	-	-	46.5	3384	0.226	-	-	50.1
	DistMult*	254	0.241	15.5	26.3	41.9	5110	0.43	39	44	49
	ComplEx*	339	0.247	15.8	27.5	42.8	5261	0.44	41	46	51
	ComplEx-N3*	-	**0.37**	-	-	**56**	-	**0.48**	-	-	**57**
	ConvE*	244	0.325	23.7	35.6	50.1	4187	0.43	40	44	52
	TuckER*	-	0.358	**26.6**	**39.4**	54.4	-	0.470	44.3	48.2	52.6
	RotatE*	**177**	0.338	24.1	37.5	53.3	**3340**	0.476	42.8	49.2	**57.1**
Rule Learning	PathRank	-	0.087	7.4	9.2	11.2	-	0.189	17.1	20.0	22.5
	NeuralLP†	-	0.237	17.3	25.9	36.1	-	0.381	36.8	38.6	40.8
	DRUM†	-	0.238	17.4	26.1	36.4	-	0.382	36.9	38.8	41.0
	NLIL*	-	0.25	-	-	32.4	-	-	-	-	-
	MINERVA*	-	0.293	21.7	32.9	45.6	-	0.415	38.2	43.3	48.0
	M-Walk*	-	0.232	16.5	24.3	-	-	0.437	41.4	44.5	-
RNNLogic	w/o emb.	538	0.288	20.8	31.5	44.5	7527	0.455	41.4	47.5	53.1
	with emb.	232	0.344	25.2	38.0	53.0	4615	**0.483**	**44.6**	**49.7**	55.8

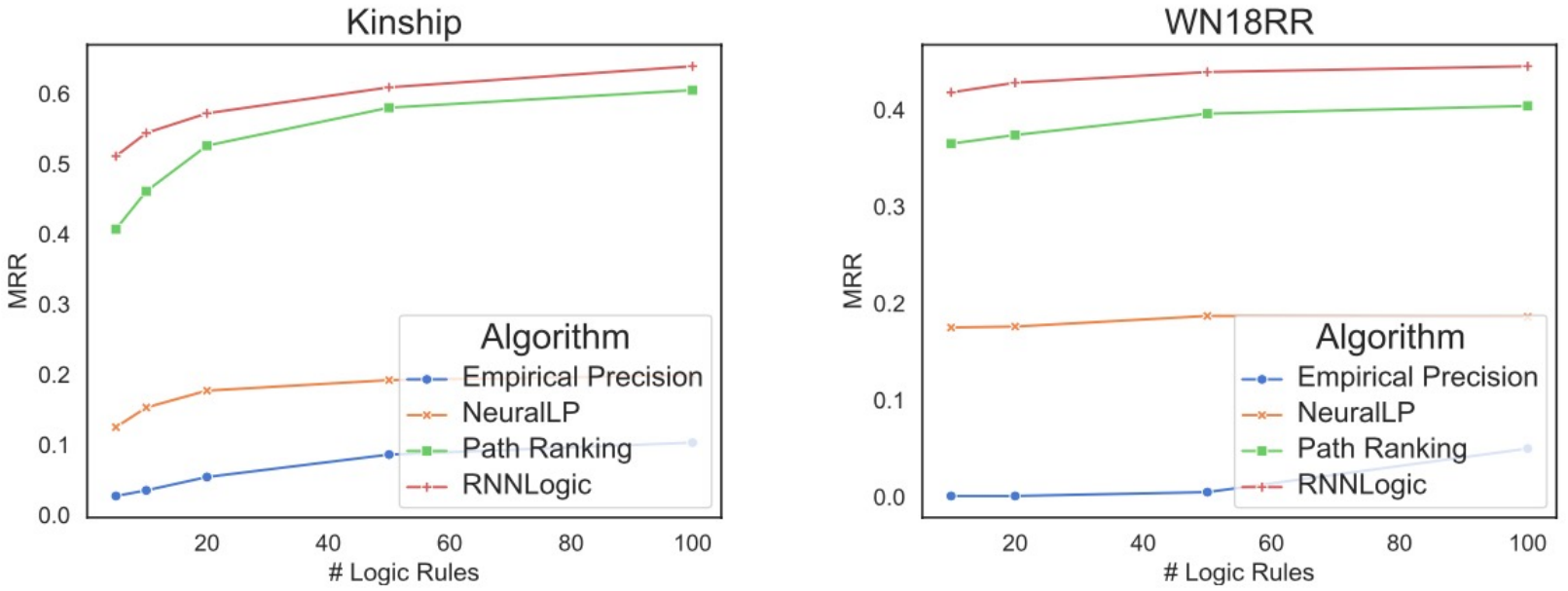
Main Results on Kinship and UMLS

- RNNLogic outperforms all the methods
- RNNLogic achieves comparable results to state-of-the-art knowledge graph embedding methods even without using embedding in predictors

Category	Algorithm	Kinship					UMLS				
		MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
No Rule Learning	DistMult	8.5	0.354	18.9	40.0	75.5	14.6	0.391	25.6	44.5	66.9
	ComplEx	7.8	0.418	24.2	49.9	81.2	13.6	0.411	27.3	46.8	70.0
	ComplEx-N3	-	0.605	43.7	71.0	92.1	-	0.791	68.9	87.3	95.7
	TuckER	6.2	0.603	46.2	69.8	86.3	5.7	0.732	62.5	81.2	90.9
	RotatE	3.7	0.651	50.4	75.5	93.2	4.0	0.744	63.6	82.2	93.9
Rule Learning	MLN	10.0	0.351	18.9	40.8	70.7	7.6	0.688	58.7	75.5	86.9
	Boosted RDN	25.2	0.469	39.5	52.0	56.7	54.8	0.227	14.7	25.6	37.6
	PathRank	-	0.369	27.2	41.6	67.3	-	0.197	14.8	21.4	25.2
	NeuralLP	16.9	0.302	16.7	33.9	59.6	10.3	0.483	33.2	56.3	77.5
	DRUM	11.6	0.334	18.3	37.8	67.5	8.4	0.548	35.8	69.9	85.4
	MINERVA	-	0.401	23.5	46.7	76.6	-	0.564	42.6	65.8	81.4
	CTP	-	0.335	17.7	37.6	70.3	-	0.404	28.8	43.0	67.4
RNNLogic	w/o emb.	3.9	0.639	49.5	73.1	92.4	5.3	0.745	63.0	83.3	92.4
	with emb.	**3.1**	**0.722**	**59.8**	**81.4**	**94.9**	**3.1**	**0.842**	**77.2**	**89.1**	**96.5**

Performace w.r.t. the Number of Rules

- Generate different numbers of logic rules with different methods
- Train reasoning predictors with these rules to evaluate the results
- RNNLogic achieves competitive results even with 10 rules per relation



Case Study

- The logic rules generated by RNNLogic are meaningful and diverse
  - Rule 1 is a subrelation rule
  - Rule 3&4 are two-hop compositional rules
  - Others have more complicated forms

$\texttt{Appears\_in\_TV\_Show}(X,Y) \leftarrow \texttt{Has\_Actor}(X,Y)$
$\texttt{Appears\_in\_TV\_Show}(X,Y) \leftarrow \texttt{Creator\_of}(X,U) \wedge \texttt{Has\_Producer}(U,V) \wedge \texttt{Appears\_in\_TV\_Show}(V,Y)$
$\texttt{ORG.\_in\_State}(X,Y) \leftarrow \texttt{ORG.\_in\_City}(X,U) \wedge \texttt{City\_Locates\_in\_State}(U,Y)$
$\texttt{ORG.\_in\_State}(X,Y) \leftarrow \texttt{ORG.\_in\_City}(X,U) \wedge \texttt{Address\_of\_PERS.}(U,V) \wedge \texttt{Born\_in}(V,W) \wedge \texttt{Town\_in\_State}(W,Y)$
$\texttt{Person\_Nationality}(X,Y) \leftarrow \texttt{Born\_in}(X,U) \wedge \texttt{Place\_in\_Country}(U,Y)$
$\texttt{Person\_Nationality}(X,Y) \leftarrow \texttt{Student\_of\_Educational\_Institution}(X,U) \wedge \texttt{ORG.\_Endowment\_Currency}(U,V) \wedge \texttt{Currency\_Used\_in\_Region}(V,W) \wedge \texttt{Region\_in\_Country}(W,Y)$

Take Away

- Predicting missing facts (or reasoning) on knowledge graphs is a very important problem
  - Learning the logic rules is the key
- A new algorithm called **RNNLogic** for learning logic rules for reasoning on knowledge graphs
  - For reach query, treating a set of meaningful logic rules as a latent variable
  - A RNN generator to generate meaningful logic rules (prior distribution)
  - A logic rule-based reasoning predictor (likelihood)
- Future: extend the whole framework to the few-shot learning setting

Thanks!

Contact: jian.tang@hec.ca

More Examples of Learned Rules

Relation	←	Rule (*Explanation*)
$X \xrightarrow{\texttt{Appears\_in\_TV\_Show}} Y$	←	$X \xleftarrow{\texttt{Actor\_of}} Y$ (*Definition. An actor of a show appears in the show, obviously.*)
	←	$X \xrightarrow{\texttt{Creator\_of}} U \xleftarrow{\texttt{Producer\_of}} V \xrightarrow{\texttt{Appears\_in\_TV\_Show}} Y$ (*The creator X and the producer V of another show U are likely to appear in the same show Y.*)
	←	$X \xleftarrow{\texttt{Actor\_of}} U \xleftarrow{\texttt{Award\_Nominated}} V \xleftarrow{\texttt{Winner\_of}} Y$
	←	$X \xrightarrow{\texttt{Writer\_of}} U \xleftarrow{\texttt{Creater\_of}} V \xrightarrow{\texttt{Actor\_of}} Y$
	←	$X \xrightarrow{\texttt{Student\_of}} U \xleftarrow{\texttt{Student\_of}} V \xrightarrow{\texttt{Appears\_in\_TV\_Show}} Y$ (*Two students X and V in the same school U are likely to appear in the same show Y.*)
$X \xrightarrow{\texttt{ORG.\_in\_State}} Y$	←	$X \xrightarrow{\texttt{ORG.\_in\_City}} U \xrightarrow{\texttt{City\_in\_State}} Y$ (*Use the city to indicate the state directly.*)
	←	$X \xrightarrow{\texttt{ORG.\_in\_City}} U \xleftarrow{\texttt{Lives\_in}} V \xrightarrow{\texttt{Born\_in}} W \xrightarrow{\texttt{Town\_in\_State}} Y$ (*Use the person living in the city to induct the state.*)
	←	$X \xleftarrow{\texttt{Sub-ORG.\_of}} U \xrightarrow{\texttt{ORG.\_in\_State}} Y$
	←	$X \xrightarrow{\texttt{Sub-ORG.\_of}} U \xleftarrow{\texttt{Sub-ORG.\_of}} V \xrightarrow{\texttt{ORG.\_in\_State}} Y$
	←	$X \xrightarrow{\texttt{ORG.\_in\_City}} U \xleftarrow{\texttt{ORG.\_in\_City}} V \xrightarrow{\texttt{ORG.\_in\_State}} Y$ (*Two organizations in the same city are in the same state.*)

$X \xrightarrow{\texttt{Person\_Nationality}} Y$	←	$X \xrightarrow{\texttt{Born\_in}} U \xrightarrow{\texttt{Place\_in\_Country}} Y$ (*Definition.*)
	←	$X \xrightarrow{\texttt{Spouse}} U \xrightarrow{\texttt{Person\_Nationality}} Y$ (*By a fact that people are likely to marry a person of same nationality.*)
	←	$X \xrightarrow{\texttt{Student\_of}} U \xrightarrow{\texttt{ORG.\_Endowment\_Currency}} V \xleftarrow{\texttt{Region\_Currency}} W \xrightarrow{\texttt{Region\_in\_Country}} Y$ (*Use the currency to induct the nationality.*)
	←	$X \xrightarrow{\texttt{Born\_in}} U \xleftarrow{\texttt{Born\_in}} V \xrightarrow{\texttt{Person\_Nationality}} Y$
	←	$X \xrightarrow{\texttt{Politician\_of}} U \xleftarrow{\texttt{Politician\_of}} V \xrightarrow{\texttt{Person\_Nationality}} Y$
$X \xrightarrow{\texttt{Manifestation\_of}} Y$	←	$X \xleftarrow{\texttt{Treats}} U \xrightarrow{\texttt{Prevents}} V \xleftarrow{\texttt{Precedes}} Y$
	←	$X \xleftarrow{\texttt{Complicates}} U \xleftarrow{\texttt{Precedes}} Y$
	←	$X \xrightarrow{\texttt{Location\_of}} U \xrightarrow{\texttt{Is\_a}} V \xleftarrow{\texttt{Precedes}} Y$
	←	$X \xleftarrow{\texttt{Complicates}} U \xrightarrow{\texttt{Precedes}} V \xleftarrow{\texttt{Occurs\_in}} Y$
	←	$X \xrightarrow{\texttt{Location\_of}} U \xleftarrow{\texttt{Occurs\_in}} V \xleftarrow{\texttt{Occurs\_in}} Y$
	←	$X \xrightarrow{\texttt{Precedes}} U \xleftarrow{\texttt{Occurs\_in}} V \xleftarrow{\texttt{Degree\_of}} Y$
$X \xleftarrow{\texttt{Affects}} Y$	←	$X \xrightarrow{\texttt{Result\_of}} U \xrightarrow{\texttt{Occurs\_in}} V \xrightarrow{\texttt{Precedes}} Y$
	←	$X \xleftarrow{\texttt{Precedes}} U \xrightarrow{\texttt{Produces}} V \xleftarrow{\texttt{Occurs\_in}} Y$
	←	$X \xleftarrow{\texttt{Prevents}} U \xrightarrow{\texttt{Disrupts}} V \xrightarrow{\texttt{Co-occurs\_with}} Y$
	←	$X \xleftarrow{\texttt{Result\_of}} U \xrightarrow{\texttt{Complicates}} V \xrightarrow{\texttt{Precedes}} Y$
	←	$X \xleftarrow{\texttt{Assesses\_Effect\_of}} U \xrightarrow{\texttt{Method\_of}} V \xrightarrow{\texttt{Complicates}} Y$
	←	$X \xrightarrow{\texttt{Process\_of}} U \xrightarrow{\texttt{Interacts\_with}} V \xrightarrow{\texttt{Causes}} Y$
	←	$X \xleftarrow{\texttt{Assesses\_Effect\_of}} U \xleftarrow{\texttt{Result\_of}} V \xrightarrow{\texttt{Precedes}} Y$

Optimization E-step

- Approximation:
  - For a query $\boldsymbol{q} = (h, \mathrm{r}, ?)$ and answer $\boldsymbol{a} = t$, compute $H(rule)$ for each $rule \in \hat{\boldsymbol{z}}$:

$$H(rule) = \left\{ \text{score}(t|rule) - \frac{1}{|\mathcal{A}|} \sum_{e \in \mathcal{A}} \text{score}(e|rule) \right\} + \log \text{RNN}_\theta(rule|r)$$

The score that *rule* assigns to the correct answer in the reasoning predictor

The mean score that *rule* assigns to all candidate answers in the reasoning predictor

Prior probability of *rule* from the rule generator

  - $H(rule)$ reflects how important each *rule* is for a pair of $(\boldsymbol{q}, \boldsymbol{a})$
  - $\hat{\boldsymbol{z}}_I$ can be formed by K rules with the maximum $H(rule)$