Neural and Symbolic Logical Reasoning on Knowledge Graphs

Jian Tang

Mila-Quebec AI Institute

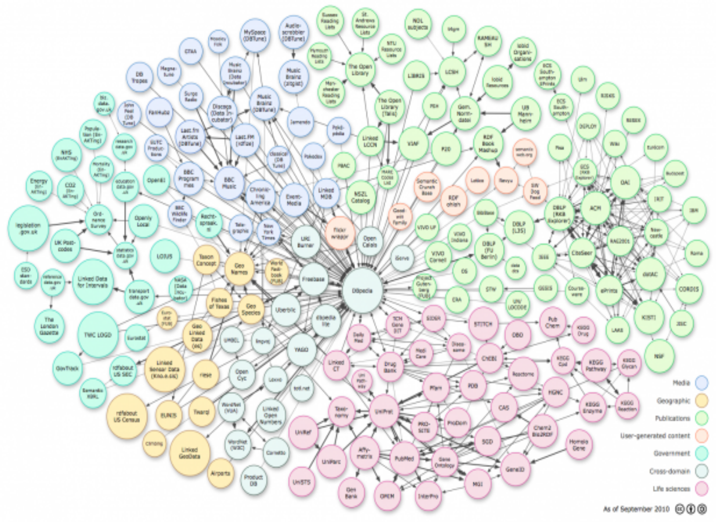
CIFAR AI Chair, HEC Montreal

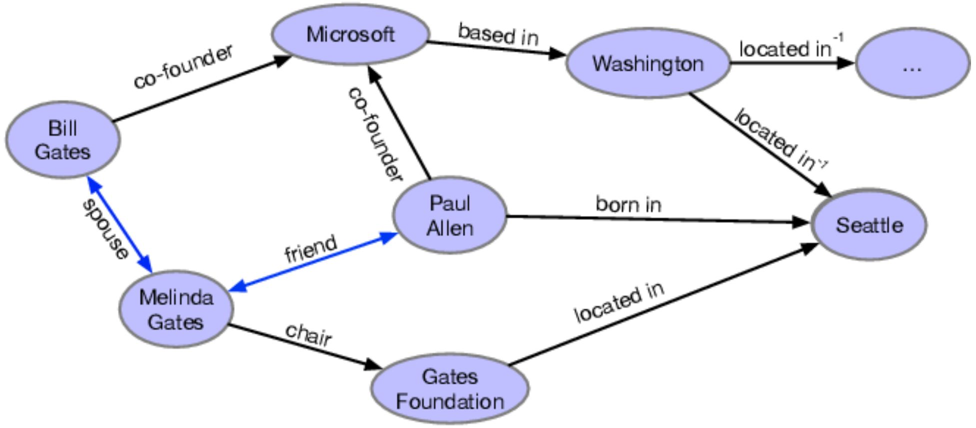
Homepage: www.jian-tang.com

Acknowledgements: Meng Qu, Yoshua Bengio, Zhiqing Sun, Zhaocheng Zhu, Junkun Chen, Louis-Pascal Xhonneux

Knowledge Graphs

- Knowledge graphs are **heterogeneous** graphs
  - Multiple types of relations
- A set of facts represented as triplets
  - (head entity, relation, tail entity)





Google
Knowledge Graph

Microsoft
SATORI

Browse the Knowledge Base!
NELL: Never-Ending Language Learning

Freebase

DBpedia

yago select knowledge

OpenIE
(Reverb, OLLIE)

Recommendation in E-commerce

- Suggest relevant items to users

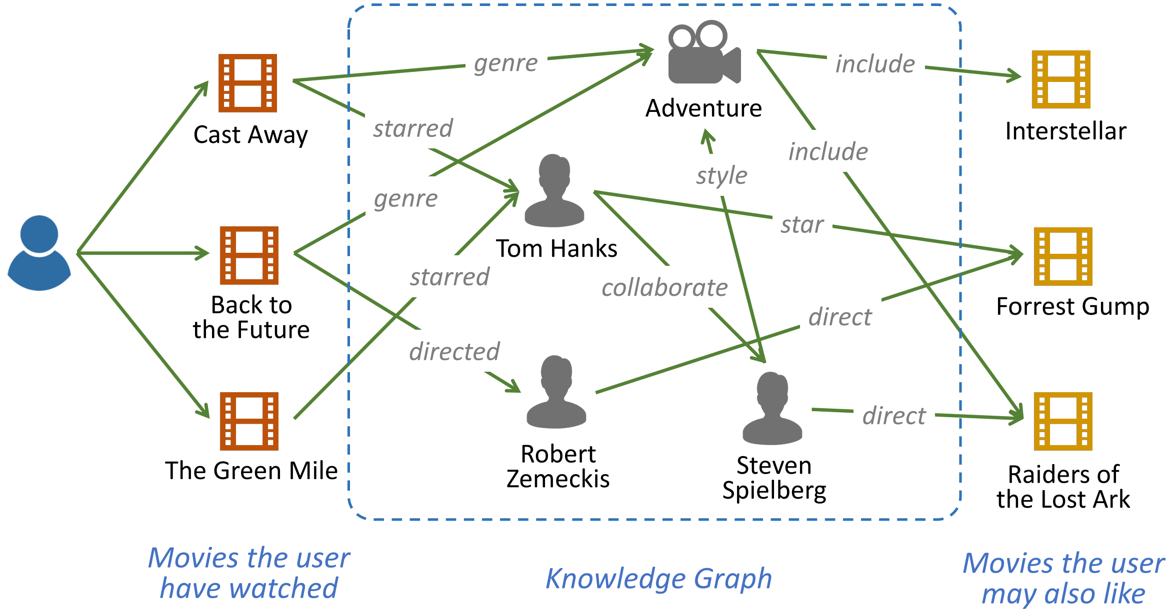
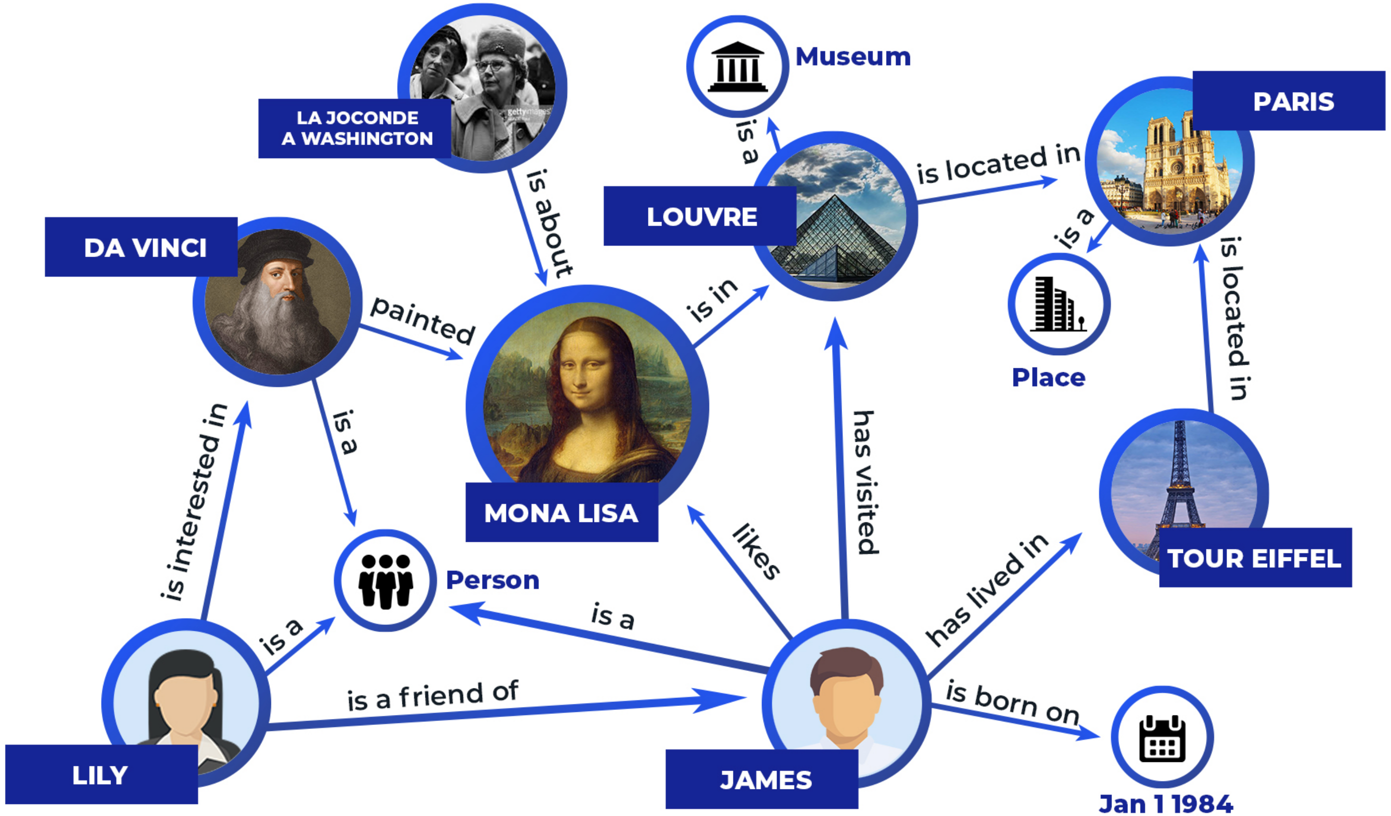


Figure from Wang et al. 2018

Question Answering

Question: **“What are all the country capitals in Africa?”**



Drug Repurposing

- Predicting effective (approved) drugs given a disease

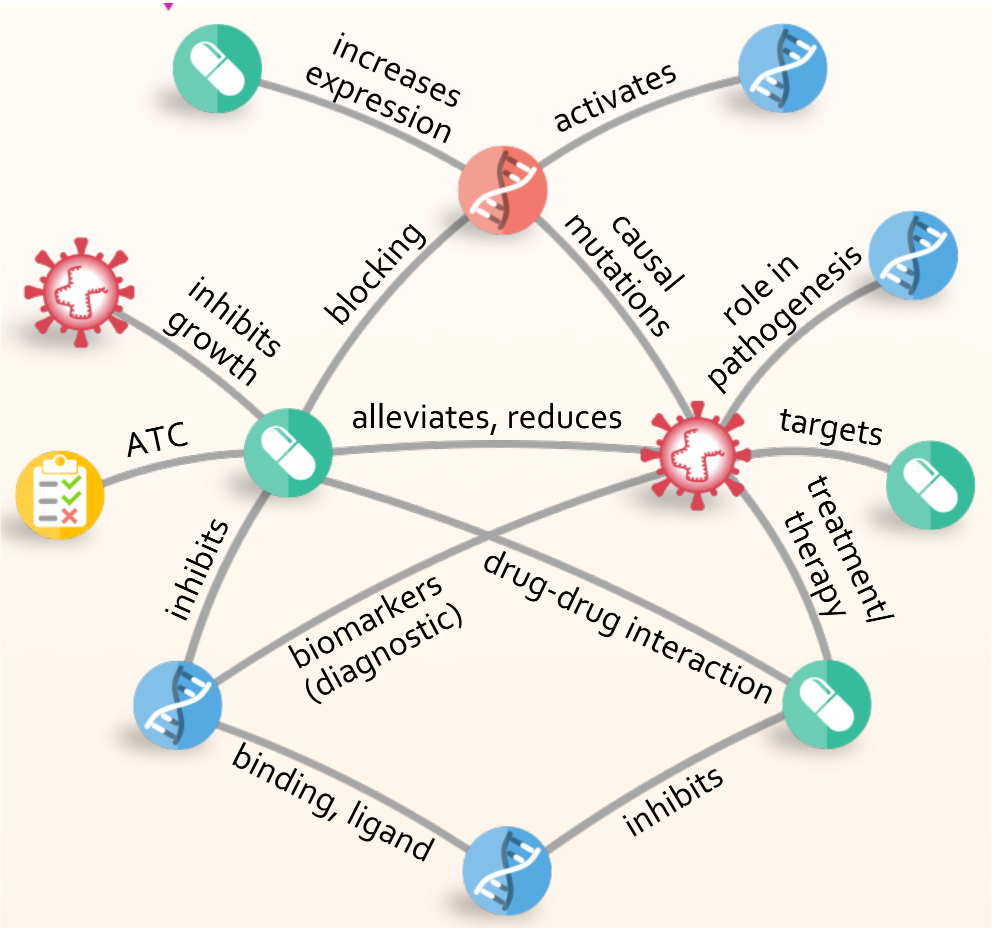


Figure from Zeng et al. 2019

Information Retrieval

- Knowledge graphs are used to understand the meanings of query terms and identify documents that match the meanings

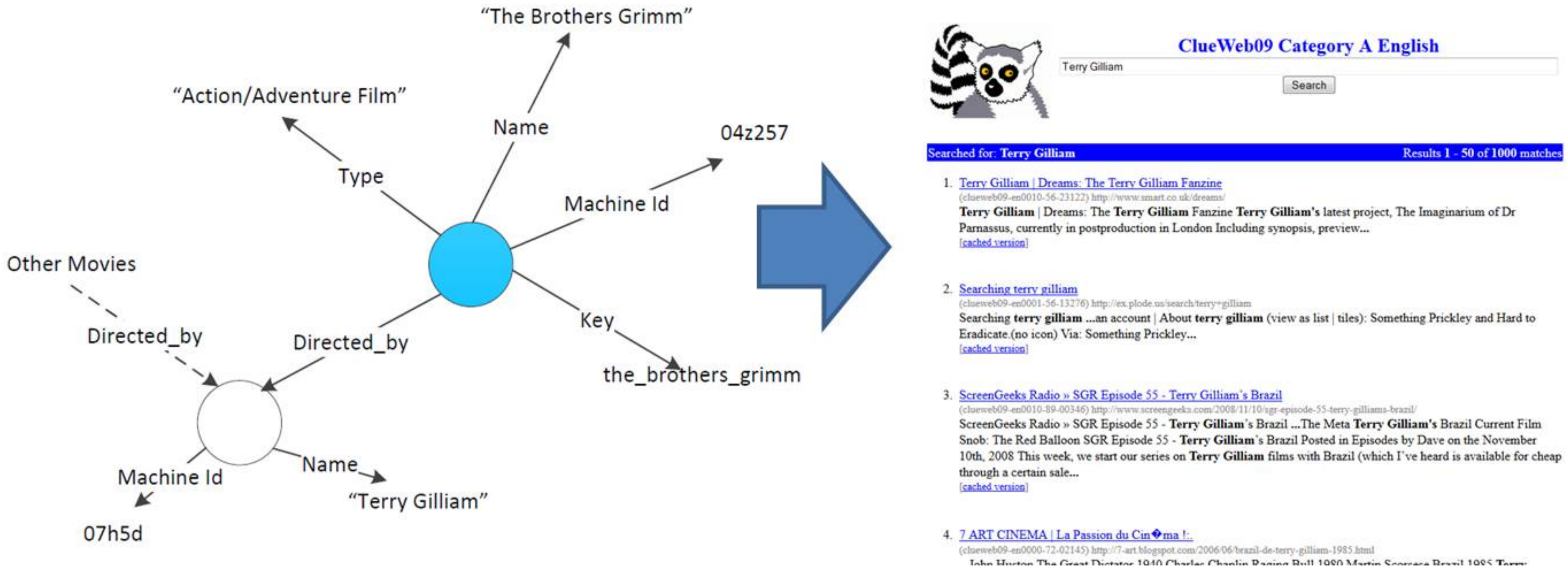


Figure from http://www.cs.cmu.edu/~callan/Projects/IIS-1422676/

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:

Barack_Obama **BornIn** United_States

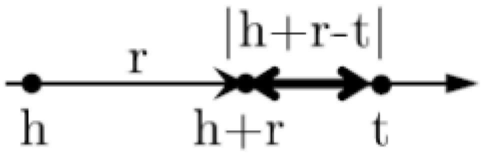
↓

Barack_Obama **Nationality** American

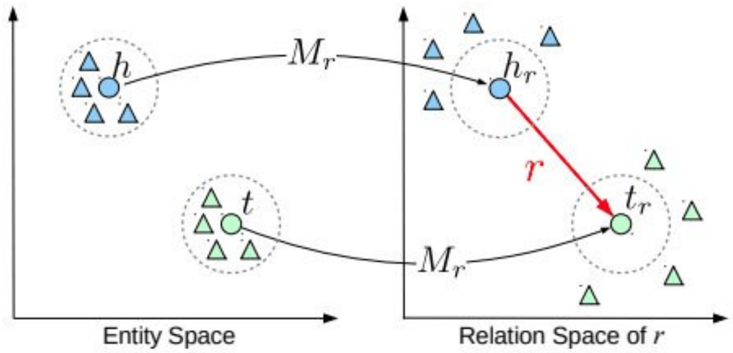
Parents of Parents are Grandparents

Reasoning in Continuous Space

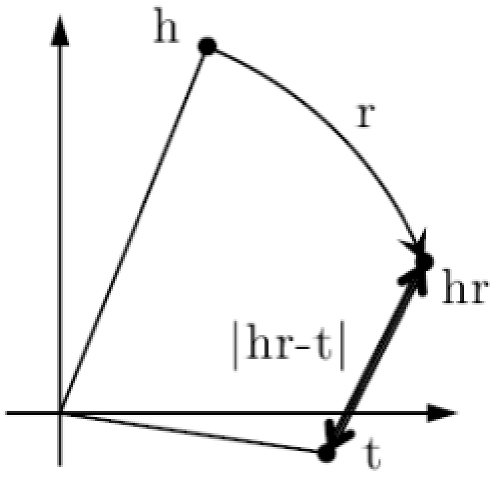
- Knowledge graph embedding methods
  - Map entities and relations into continuous space, and reasoning in the continuous spaces
  - TransE, TransH, TransR, ComplEx, RotatE, ….



TransE



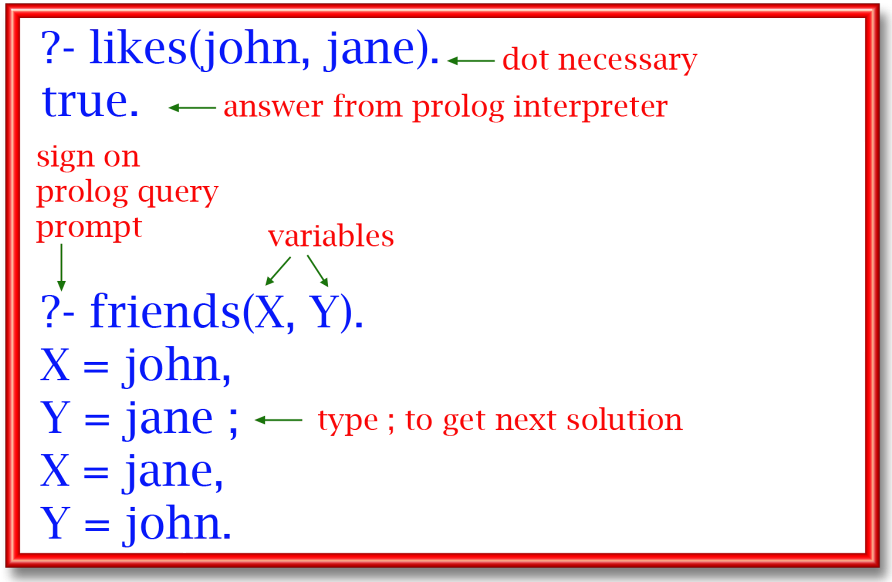
TransR



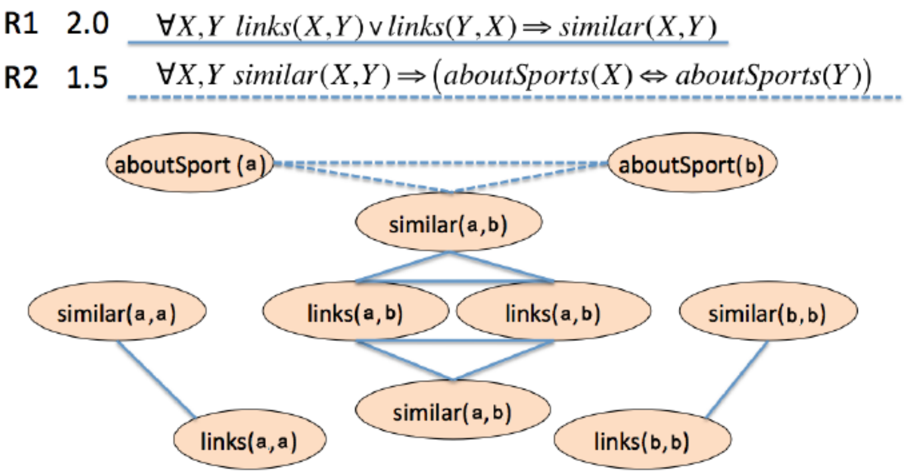
RotatE

Reasoning in Symbolic Space

- Symbolic logical rule based methods
  - Logic programming (e.g., Prolog)
  - Markov Logic Network
  - ….



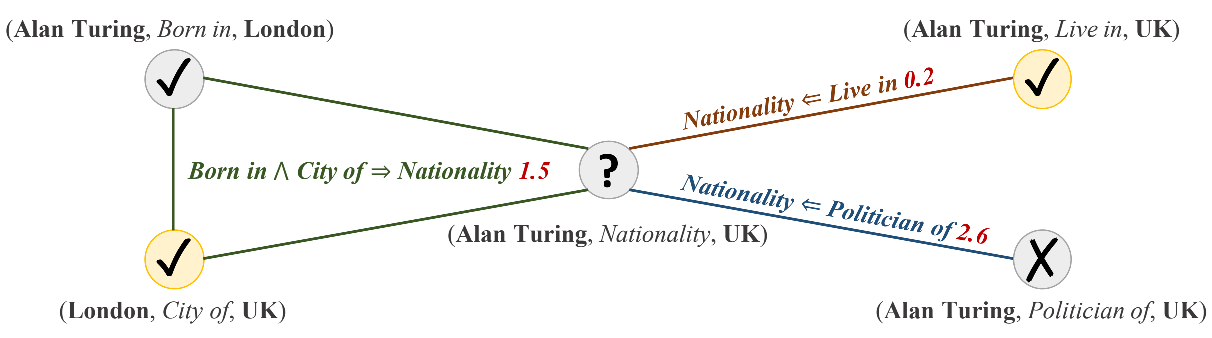
Prolog



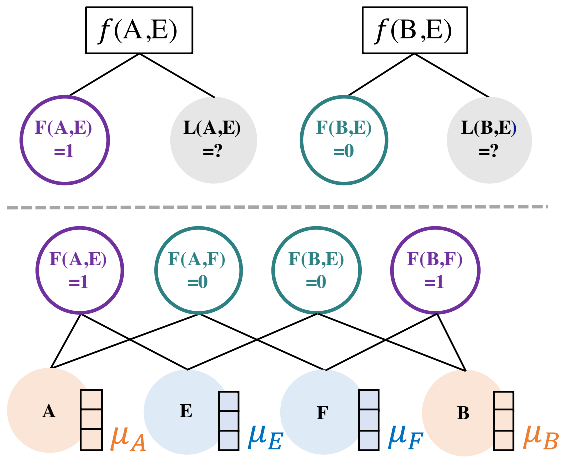
Markov Logic Networks

Neural-Symbolic Reasoning

- Reasoning in both continuous and symbolic space
- pLogicNet (Qu and Tang, 2019)
- ExpressGNN (Zhang et al. 2019)



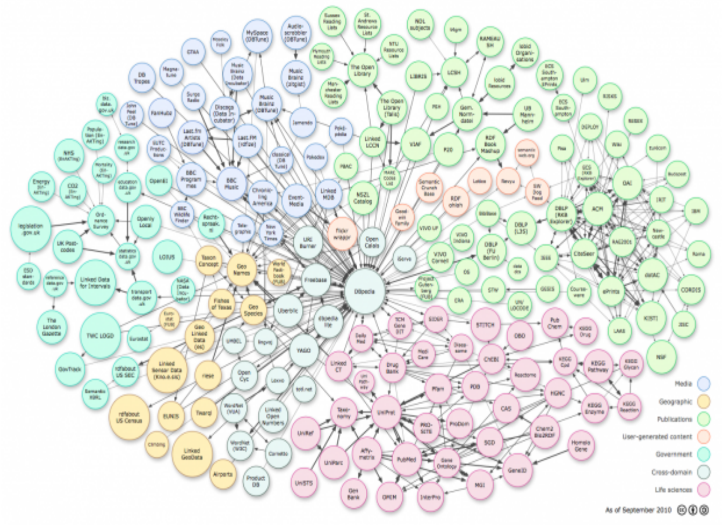
pLogicNet



ExpressGNN

Logical Rule Induction/Learning

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



Appears_in_TV_Show$(X,Y) \leftarrow$ Has_Actor(X,Y)
Appears_in_TV_Show$(X,Y) \leftarrow$ Creator_of$(X,U) \wedge$ Has_Producer$(U,V) \wedge$ Appears_in_TV_Show(V,Y)
ORG._in_State$(X,Y) \leftarrow$ ORG._in_City$(X,U) \wedge$ City_Locates_in_State(U,Y)
ORG._in_State$(X,Y) \leftarrow$ ORG._in_City$(X,U) \wedge$ Address_of_PERS.$(U,V) \wedge$ Born_in$(V,W) \wedge$ Town_in_State(W,Y)
Person_Nationality$(X,Y) \leftarrow$ Born_in$(X,U) \wedge$ Place_in_Country(U,Y)
Person_Nationality$(X,Y) \leftarrow$ Student_of_Educational_Institution$(X,U) \wedge$ ORG._Endowment_Currency$(U,V) \wedge$ Currency_Used_in_Region$(V,W) \wedge$ Region_in_Country(W,Y)

Roadmap

- Part I: Reasoning in Continuous Space
- Part II: Symbolic Logic Reasoning
- Part III: Neural-Symbolic Logic Reasoning
- Part IV: Logic Rule Induction/Learning

Logical Rules

- **Symmetric/Antisymmetric** Rule
  - Symmetric: e.g., Marriage
  - Antisymmetric: e.g., Filiation
- Formally:

r is **Symmetric**: $\underbrace{r^{-1}(\mathrm{X}, Y)}_{\text{Rule Head}} \leftarrow \underbrace{r(X, Y)}_{\text{Rule Body}} \forall X, Y$

r is **Antisymmetric**: $\neg r^{-1}(X, Y) \leftarrow r(X, Y) \text{if } \forall X, Y$

Logical Rules

- **Inverse** Rule
  - Hypernym and hyponym
  - Husband and wife
- Formally:

r_1 **is inverse to relation** r_2**:** $r_1^{-1}(X, Y) \leftarrow r_2(X, Y)$ if $\forall\, X, Y$

Logical Rules

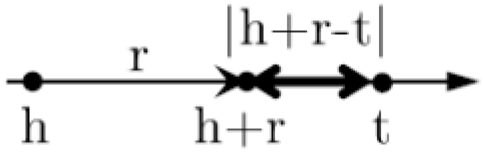
- **Composition** Rule
  - My mother's husband is my father
- Formally:

r_1 is a **composition** of relation r_2 and relation r_3: $r_1(X, Z) \leftarrow r_2(X, Y) \wedge r_3(Y, Z)$ if $\forall X, Y, Z$

TransE (Bordes et al. 2013)

- Each entity and relation is embedded as a low-dimensional vector
- Relation **r** defined as a **translation** from the head entity **h** to the tail entity **t**.

$$\mathbf{t} = \mathbf{h} + \mathbf{r}$$



- Scoring function:

$$-||\mathbf{h} + \mathbf{r} - \mathbf{t}||$$

Question

- What kinds of logical rules TransE can model and infer?

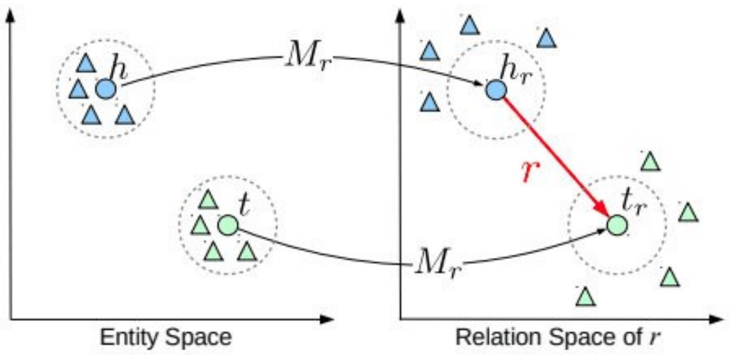
TransR (Lin et al. 2015)

- Limitations of TransE: entities and relations are assumed to be lie in the same space, which might not be true
- Map entities to the semantic space of relations through a projection

$$\mathbf{h_r} = \mathbf{hM_r} \qquad \mathbf{t_r} = \mathbf{tM_r}$$

- Scoring function:

$$-||\mathbf{h_r} + \mathbf{r} - \mathbf{t_r}||$$



RotatE (Sun et al. 2019)

- Representing head and tail entities in complex vector space, i.e., $\mathbf{h}, \mathbf{t} \in \mathbb{C}^{\boldsymbol{k}}$
- Define each relation **r** as an element-wise rotation from the head entity **h** to the tail entity **t,** i.e.,

$$\mathbf{t} = \mathbf{h} \circ \boldsymbol{r}, \quad \text{where } |r_i|=1$$

- $\circ$ is the element-wise product. More specifically, we have $\mathrm{t_i} = \mathrm{h_i r_i}$, and

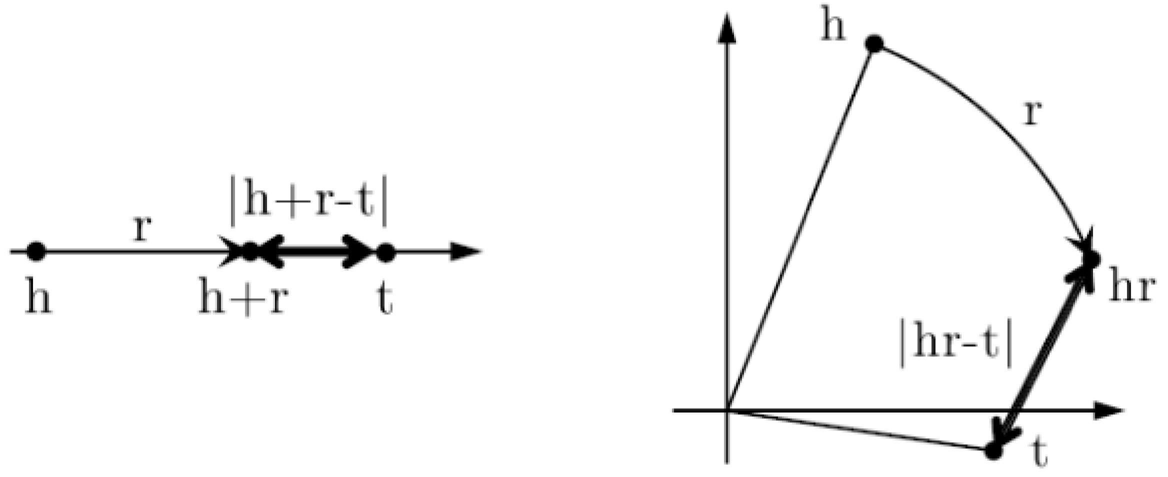
$$\mathrm{r_i} = e^{i\theta_{r,i}},$$

- where $\theta_{r,i}$ is the phase angle of **r** in the i-th dimension.

Geometric Interpretation

- Define the distance function of RotatE as

$$d_r(\boldsymbol{h}, \boldsymbol{t}) = ||\mathbf{h} \circ \mathbf{r} - \mathbf{t}||$$



(a) TransE models $\mathbf{r}$ as translation in real line.

(b) RotatE models $\mathbf{r}$ as rotation in complex plane.

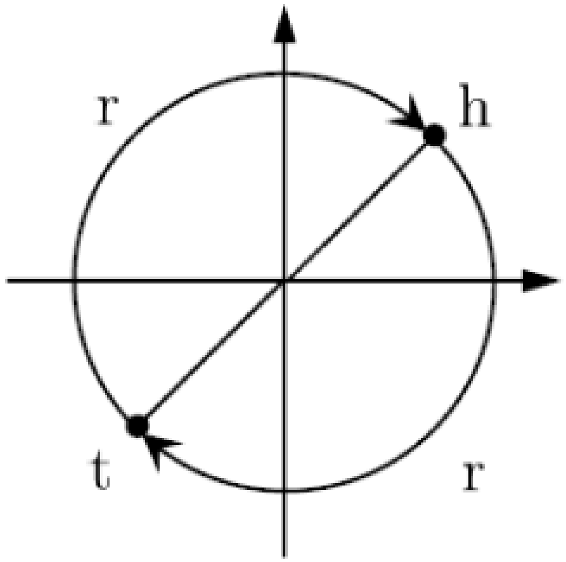
Modeling the Relation Patterns with RotatE

- A relation **r** is **symmetric** if and only if $r_i = \pm 1$, i.e.,

$$\theta_{r,i} = 0 \; or \; \pi$$

- An example on the space of $\mathbb{C}$

$$r_i = -1 \text{ or } \theta_{r,i} = \pi$$



Modeling the Relation Patterns with RotatE

- A relation r is **antisymmetric** if and only if $\mathbf{r} \circ \mathbf{r} \neq \mathbf{1}$

- Two relations r_1 and r_2 are **inverse** if and only if $\mathbf{r}_2 = \bar{\mathbf{r}}_1$, i.e.,

$$\theta_{2,i} = -\theta_{1,i}$$

- A relation $\boldsymbol{r}_3 = e^{i\boldsymbol{\theta}_3}$ is a **composition** of two relations $\boldsymbol{r}_1 = e^{i\boldsymbol{\theta}_1}$ and $\boldsymbol{r}_2 = e^{i\boldsymbol{\theta}_2}$ if only if $\boldsymbol{r}_3 = \boldsymbol{r}_1 \circ \boldsymbol{r}_2$, i.e.,

$$\boldsymbol{\theta}_3 = \boldsymbol{\theta}_1 + \boldsymbol{\theta}_2$$

Optimization (Sun et al. 2019)

- Negative sampling loss

$$L = -\log \sigma\big(\gamma - d_r(\boldsymbol{h}, \boldsymbol{t})\big) - \sum_{i=1}^{k} \frac{1}{k} \log \sigma(d_r(\boldsymbol{h}'_i, \boldsymbol{t}'_i) - \gamma)$$

- γ is a fixed margin, σ is the sigmoid function, and $(\boldsymbol{h}'_i, \boldsymbol{r}, \boldsymbol{t}'_i)$ is the i-th negative triplet.

Self-adversarial Negative Sampling (Sun et al. 2019)

- Traditionally, the negative samples are drawn in an uniform way
  - Inefficient as training goes on since many samples are obviously false
  - Does not provide useful information
- A self-adversarial negative sampling
  - Sample negative triplets according to the current embedding model
  - Starts from easier samples to more and more difficult samples
  - Curriculum Learning

$$p(h'_j, r, t'_j | \{(h_i, r_i, t_i)\}) = \frac{\exp \alpha f_r(\mathbf{h}'_j, \mathbf{t}'_j)}{\sum_i \exp \alpha f_r(\mathbf{h}'_i, \mathbf{t}'_i)}$$

- α is the temperature of sampling. $f_r(h'_j, t'_j)$ measures the salience of the triplet

The Final Objective

- Instead of sampling, treating the sampling probabilities as weights.

$$L = -\log \sigma(\gamma - d_r(\mathbf{h}, \mathbf{t})) - \sum_{i=1}^{n} p(h'_i, r, t'_i) \log \sigma(d_r(\mathbf{h}'_i, \mathbf{t}'_i) - \gamma)$$

Other Approaches

- TransH (Wang et al. 2014)
- STransE (Nguyen et al. 2016)
- DisMult (Yang et al. 2014)
- ComplEx (Trouillon et al. 2016)
- HolE (Nickel et al. 2016)
- ConvE (Dettmers et al. 2017)
- QuaE (Zhang et al. 2019)
- …

Analysis on Inferring Different Types of Logical Rules

Model	Score Function	Symmetry	Antisymmetry	Inversion	Composition
SE	$-\|\boldsymbol{W}_{r,1}\mathbf{h} - \boldsymbol{W}_{r,2}\mathbf{t}\|$	✗	✗	✗	✗
TransE	$-\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|$	✗	✓	✓	✓
TransX	$-\|g_{r,1}(\mathbf{h}) + \mathbf{r} - g_{r,2}(\mathbf{t})\|$	✓	✓	✗	✗
DistMult	$\langle \mathbf{h}, \mathbf{r}, \mathbf{t} \rangle$	✓	✗	✗	✗
ComplEx	$\mathrm{Re}(\langle \mathbf{h}, \mathbf{r}, \overline{\mathbf{t}} \rangle)$	✓	✓	✓	✗
RotatE	$-\|\mathbf{h} \circ \mathbf{r} - \mathbf{t}\|$	✓	✓	✓	✓

Benchmark Data Sets

- **FB15K**: a subset of Freebase. The main relation types are **symmetry/antisymmetry** and **inversion** patterns.
- **WN18**: a subset of WordNet. The main relation types are **symmetry/antisymmetry** and **inversion** patterns.
- **FB15K-237**: a subset of FB15K, where inversion relations are deleted. The main relation types are **symmetry/antisymmetry** and **composition** patterns.
- **WN18RR**: a subset of WN18, where inversion relations are deleted. The main relation types are **symmetry/antisymmetry** and **composition** patterns.

Dataset	#entity	#relation	#training	#validation	#test
FB15k	14,951	1,345	483,142	50,000	59,071
WN18	40,943	18	141,442	5,000	5,000
FB15k-237	14,541	237	272,115	17,535	20,466
WN18RR	40,943	11	86,835	3,034	3,134

Results on FB15k and WN18

- RotatE performs the best
- pRotatE performs similarly to RotatE

	FB15k					WN18				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
TransE [♥]	-	.463	.297	.578	.749	-	.495	.113	.888	.943
DistMult [♦]	42	.798	-	-	**.893**	655	.797	-	-	.946
HolE	-	.524	.402	.613	.739	-	.938	.930	.945	.949
ComplEx	-	.692	.599	.759	.840	-	.941	.936	.945	.947
ConvE	51	.657	.558	.723	.831	374	.943	.935	.946	.956
pRotatE	43	**.799**	**.750**	.829	.884	**254**	.947	.942	.950	.957
RotatE	**40**	.797	.746	**.830**	.884	309	**.949**	**.944**	**.952**	**.959**

Results on FB15k-237 and WN18RR

- RotatE performs the best
- RotatE performs significantly better than pRotatE
  - A lot of composition patterns on the two data sets
  - Modulus information are important for modeling the composition patterns

	FB15k-237					**WN18RR**				
	MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
TransE [♥]	357	.294	-	-	.465	3384	.226	-	-	.501
DistMult	254	.241	.155	.263	.419	5110	.43	.39	.44	.49
ComplEx	339	.247	.158	.275	.428	5261	.44	.41	.46	.51
ConvE	244	.325	.237	.356	.501	4187	.43	.40	.44	.52
pRotatE	178	.328	.230	.365	.524	**2923**	.462	.417	.479	.552
RotatE	**177**	**.338**	**.241**	**.375**	**.533**	3340	**.476**	**.428**	**.492**	**.571**

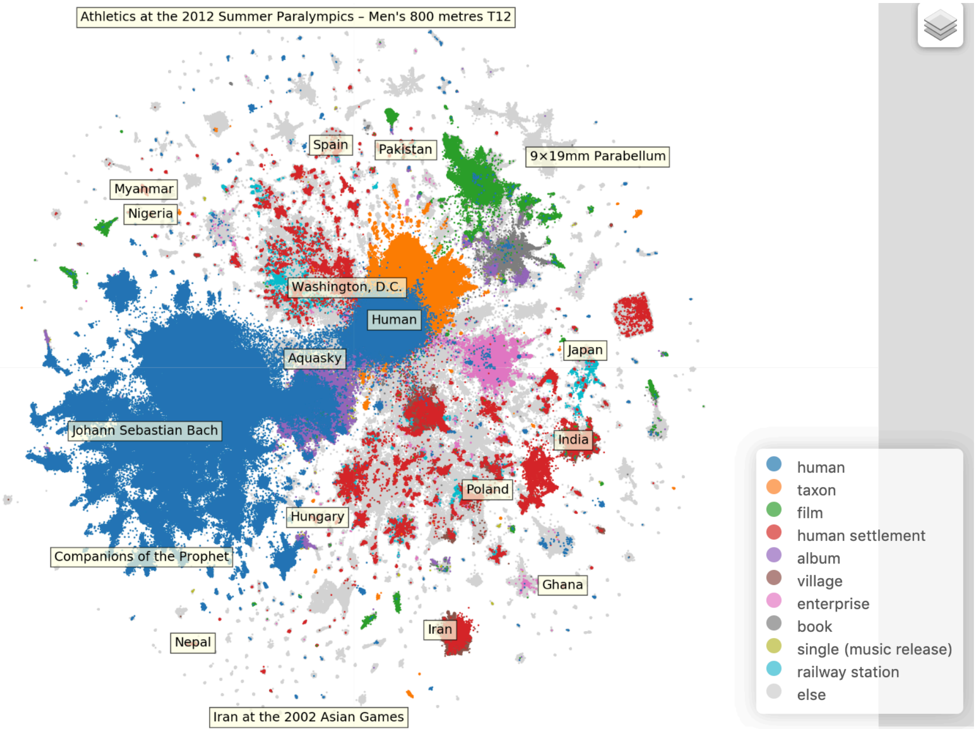
Results on Countries (Bouchard et al. 2015)

- A carefully designed dataset to explicitly test the capabilities for modeling the composition patterns
  - Three subtasks S1, S2, S3
  - From easy to difficult

	Countries (AUC-PR)			
	DistMult	ComplEx	ConvE	RotatE
S1	$\mathbf{1.00 \pm 0.00}$	0.97 ± 0.02	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
S2	0.72 ± 0.12	0.57 ± 0.10	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$
S3	0.52 ± 0.07	0.43 ± 0.07	0.86 ± 0.05	$\mathbf{0.95 \pm 0.00}$

Wikidata5M: a Large-scale Knowledge Graph (Wang et al. 2019)

- Contains 5 million entities and also the the descriptions of entities
- Pretrained knowledge graph embeddings with Wikidata5M: https://graphvite.io/pretrained_models



Open Source Package

- **OpenKE** by Prof. Zhiyuan Liu's group: https://github.com/thunlp/OpenKE
- **KnowldgeGraphEmbedding** by Prof. Jian Tang's group: https://github.com/DeepGraphLearning/KnowledgeGraphEmbedding
- **GraphVite** by Prof. Jian Tang's group: https://graphvite.io/
- **DGL-KGE** by Amazon: https://github.com/awslabs/dgl-ke

Roadmap

- Part I: Reasoning in Continuous Space
- **Part II: Symbolic Logic Reasoning**
- Part III: Neural-Symbolic Logic Reasoning
- Part IV: Logic Rule Induction/Learning

Logic Programming

- Logic programs consist of clauses
- Each clause can be viewed as a **first-order logic rule**
- Example:
  - $\underbrace{\forall X, Y, Z \quad \text{Grandfather}(X, Y)}_{\text{Rule Head}} \leftarrow \underbrace{\text{Father}(X, Z) \wedge \text{Father}(Z, Y)}_{\text{Rule Body}}$
- Apply logic rules to existing facts to **infer new facts**

Inference Algorithms

- Two fundamental algorithms:
  - Forward chaining algorithm:
    - Repeatly apply given **logic rules** to the **current set of facts**, until the fact set converges.
    - Strength: able to find a large number of facts every time
    - Weakness: inefficient and high memory cost
  - Backward chaining algorithm:
    - For each query, use the given **logic rules** and depth-first search to construct a **search tree** to infer the answer.
    - Strength: efficient
    - Weakness: focus on each individual query

Inference Algorithms

- Examples:
  - Given facts: $\text{Father}(a, b)$ $\text{Father}(b, c)$ $\text{Father}(c, d)$
  - Given logic rule: $\forall X, Y, Z$ $\text{Grandfather}(X, Y) \leftarrow \text{Father}(X, Z) \land \text{Father}(Z, Y)$

Forward Chaining

Iteration 0: $\text{Father}(a, b)$ $\text{Father}(b, c)$ $\text{Father}(c, d)$

Iteration 1: $\text{Father}(a, b)$ $\text{Father}(b, c)$ $\text{Father}(c, d)$ $\text{Grandfather}(a, c)$ $\text{Grandfather}(b, d)$

Iteration 2: $\text{Father}(a, b)$ $\text{Father}(b, c)$ $\text{Father}(c, d)$ $\text{Grandfather}(a, c)$ $\text{Grandfather}(b, d)$

Convergence

Backward Chaining

Query: $\text{Grandfather}(?, c)$

↓ Apply the given rule

$\text{Father}(?, Z) \land \text{Father}(Z, c)$

↓ Replace Z with b

$\text{Father}(?, b)$

↓ Replace ? with a

$? = a$

Logic Programming in Probabilistic Ways

- Combine **first-order logic** with **probabilistic models**
  - Model logic rules in a probabilistic way, yielding soft rules.
  - Handle the uncertainty of logic rules

- Representative methods:
  - Markov logic programming (Richardson and Domingos, 2006):
    - Markov Logic Networks (Richardson and Domingos, 2006)
  - Stochastic logic programming (Cussens, 2001) :
    - TensorLog (Cohen et al. 2017)

Markov Logic Programming (Richardson and Domingos, 2006)

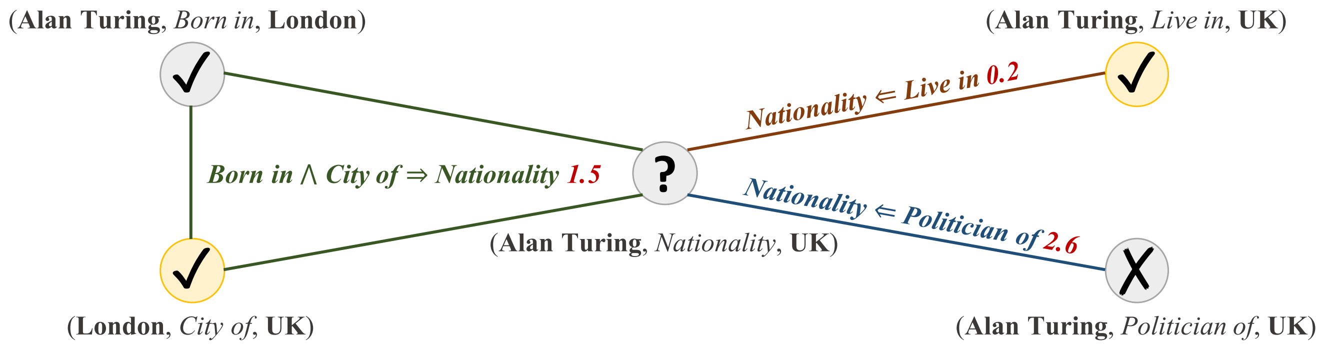
- Associate a **scalar weight** to each logic rule

- Apply the given **logic rules** to the given facts, and use the **forward chaining** algorithm to find **a collection of relevant facts**.

- Build a **Markov network** and perform inference to predict the value of each fact (true/false)

Markov Logic Programming (Richardson and Domingos, 2006)

- Example:
  - Rules:
    - R1: $\forall X, Y$ Nationality$(X, Y) \leftarrow$ LoveIn(X, Y) weight 0.2
    - R2: $\forall X, Y$ Nationality$(X, Y) \leftarrow$ PoliticianOf(X, Y) weight 2.6
    - R3: $\forall X, Y$ Nationality$(X, Z) \leftarrow$ BornIn$(X, Y) \wedge$ CityOf(Y, Z) weight 1.5
  - All obtained facts and the graph structure:



$$p(\mathbf{v}_O, \mathbf{v}_H) = \frac{1}{Z} \exp\left(\sum_{l \in L} w_l n_l(\mathbf{v}_O, \mathbf{v}_H)\right)$$

$\mathbf{v}_O$: Observed facts

$\mathbf{v}_H$: Hidden facts inferred by forward chaining

w_l: Weight of rule l

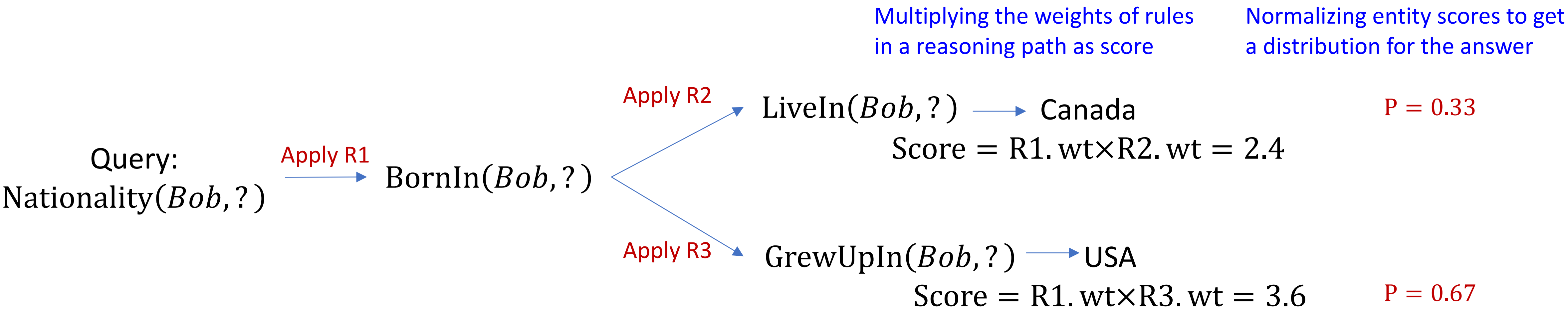
n_l: Number of times l is satisfied

Stochastic Logic Programming (Cussens, 2001)

- Associate a **scalar weight** to each logic rule
- For each query, use the given **logic rules** and **backward chaining** algoritm to build a **search tree**.
- Infer the answer according to **rule weights** and **tree structure**

Stochastic Logic Programming (Cussens, 2001)

- Example:
  - Rules:
    - R1: $\forall X, Y$ Nationality$(X, Y) \leftarrow$ BornIn(X, Y) weight 3.0
    - R2: $\forall X, Y$ BornIn$(X, Y) \leftarrow$ LiveIn(X, Y) weight 0.8
    - R3: $\forall X, Y$ BornIn$(X, Y) \leftarrow$ GrewUpIn(X, Y) weight 1.2



Other Formalizations

- Bayesian logic programming (Kersting and De Raedt et al. 2001):
  - Model each logic rule as a conditional distribution
  - Methods:
    - DeepProbLog (Manhaeve et al. 2018)
    - SPLog (Skryagin et al. 2020)

Roadmap

- Part I: Reasoning in Continuous Space
- Part II: Symbolic Logic Reasoning
- **Part III: Neural-Symbolic Logic Reasoning**
- Part IV: Logic Rule Induction/Learning

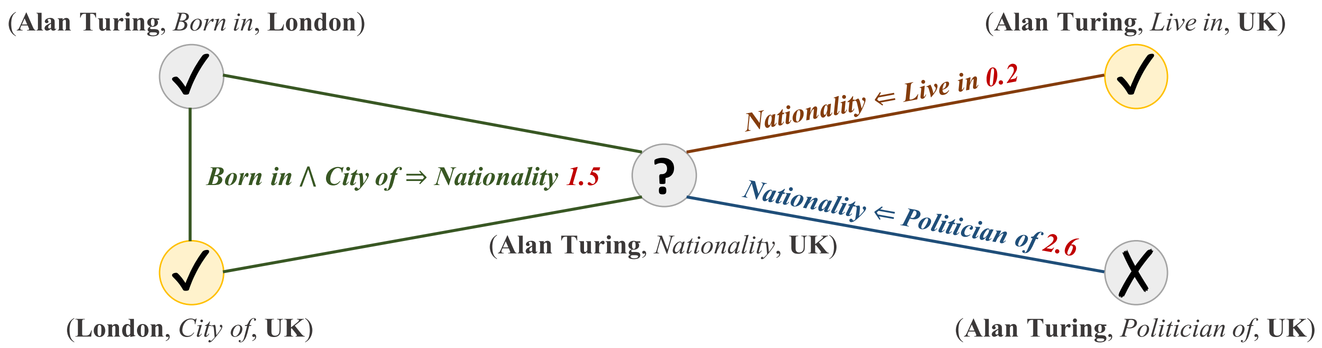
Markov Logic Networks (Richardson and Domingos, 2006)

- Combines first-order logic and probabilistic graphical models

0.2 Live(X, Y) => Nationality (X, Y)

2.6 Politician_of(X, Y) => Nationality (X, Y)

1.5 Born(X,Y)∧City_of (Y,Z) => Nationality(X, Z)



$$p(\mathbf{v}_O, \mathbf{v}_H) = \frac{1}{Z} \exp\left(\sum_{l \in L} w_l \sum_{g \in G_l} \mathbb{1}\{g \text{ is true}\} \right) = \frac{1}{Z} \exp\left(\sum_{l \in L} w_l n_l(\mathbf{v}_O, \mathbf{v}_H) \right)$$

V_O: observed facts

V_H: unobserved/hidden facts

w_l: weight of logic rule l

$n_l(V_O, V_H)$: number of true grounds of the logic rule l

Pros and Cons of Markov Logic Networks

- Pros
  - Effectively leverage domain knowledge with logic rules
  - Handle the uncertainty
- Limitation
  - Inference is difficult due to complicated graph structures
  - Recall is low since many facts are not covered by any logic rules

Knowledge Graph Embeddings

- Learning the entity and relation embeddings for predicting the missing facts (e.g., TransE, ComplEx, DisMult, RotatE)
- Defining the joint distribution of all the facts

$$p(\mathbf{v}_O, \mathbf{v}_H) = \prod_{(h,r,t)\in O\cup H} \mathrm{Ber}(\mathbf{v}_{(h,r,t)} | f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t)),$$

An example:

$$\mathrm{Ber}(\mathbf{v}_{(h,r,t)} | f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t)) = \sigma(\gamma - ||\mathbf{x}_h + \mathbf{x}_r - \mathbf{x}_t||)$$

σ is the sigmoid function, γ is a fixed margin

- Trained by treating V_O as positive facts and V_H as negative facts

Pros and Cons

- Pros
  - Can be effectively and efficiently trained by SGD
  - High recall of missing link prediction with entity and relation embeddings
- Cons
  - Hard to leverage domain knowledge (logic rules)

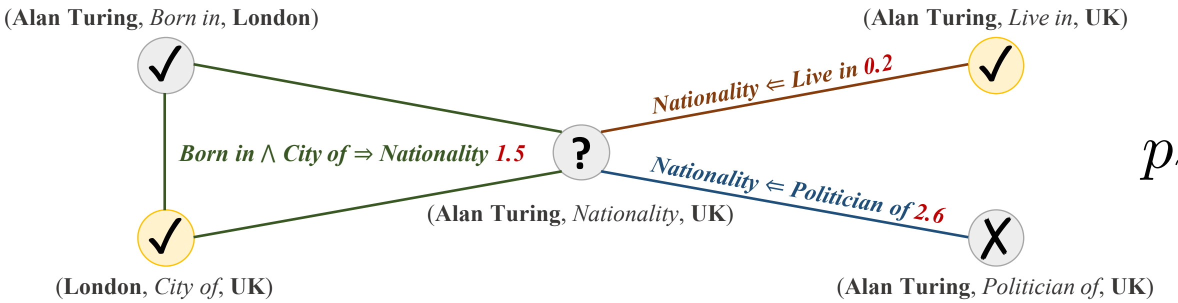
Probabilistic Logic Neural Networks for Reasoning (Qu and Tang, NeurIPS'19.)

- Towards combining Markov Logic Networks and knowledge graph embedding
  - Leverage logic rules and handling their uncertainty
  - Effective and efficient inference
- Define the joint distribution of facts with Markov Logic Network
- Optimization with variational EM
  - Parametrize the variational distribution with knowledge graph embedding methods

Meng Qu and Jian Tang. "Probabilistic Logic Neural Networks for Reasoning." In NeurIPS'2019.

pLogicNet

- Define the joint distribution of facts with an MLN



$$p_w(\mathbf{v}_O, \mathbf{v}_H) = \frac{1}{Z} \exp\left(\sum_l w_l n_l(\mathbf{v}_O, \mathbf{v}_H)\right)$$

- Learning by maximizing the variational lower-bound of the log-likelihood of observed facts

$$\log p_w(\mathbf{v}_O) \geq \mathcal{L}(q_\theta, p_w) = \mathbb{E}_{q_\theta(\mathbf{v}_H)}[\log p_w(\mathbf{v}_O, \mathbf{v}_H) - \log q_\theta(\mathbf{v}_H)]$$

Inference

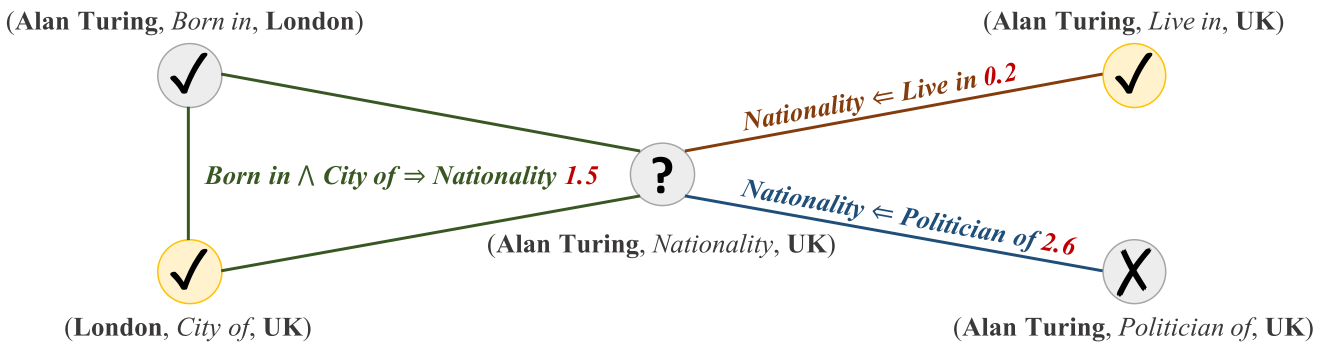
- Amortized mean-field variational inference
  - Use knowledge graph embedding model to parameterize the variational distribution

$$q_\theta(\mathbf{v}_H) = \prod_{(h,r,t)\in H} q_\theta(\mathbf{v}_{(h,r,t)}) = \prod_{(h,r,t)\in H} \mathrm{Ber}(\mathbf{v}_{(h,r,t)} | f(\mathbf{x}_h, \mathbf{x}_r, \mathbf{x}_t)),$$

Learning

- Optimize pseudo-likelihood function
  - Update the weights of logic rules

$$\ell_{PL}(w) \triangleq \mathbb{E}_{q_\theta(\mathbf{v}_H)}[\sum_{h,r,t} \log p_w(\mathbf{v}_{(h,r,t)}|\mathbf{v}_{O\cup H\setminus(h,r,t)})] = \mathbb{E}_{q_\theta(\mathbf{v}_H)}[\sum_{h,r,t} \log p_w(\mathbf{v}_{(h,r,t)}|\mathbf{v}_{\text{MB}(h,r,t)})],$$



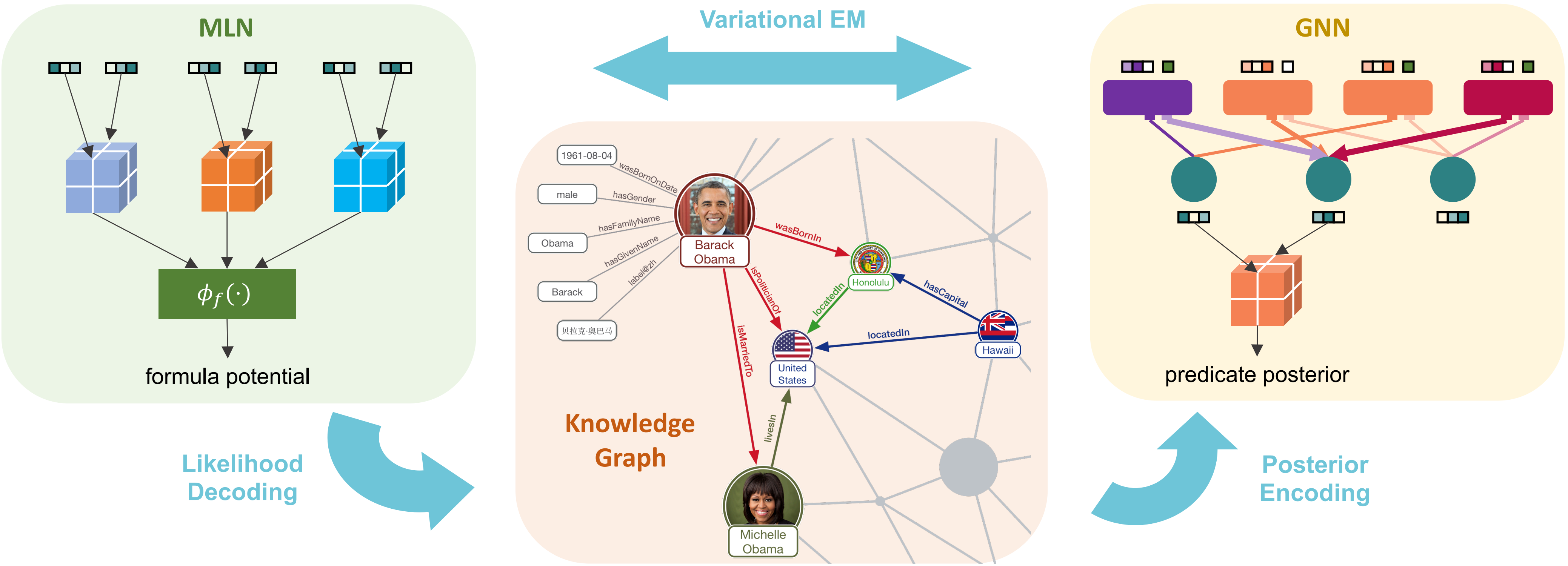
Performance of Link Prediction

- **Datasets:** benchmark knowledge graphs
  - FB15K, WN18, FB15K-237, WN18-RR
- Logic rules:
  - Composition rules (e.g., Father of Father is GrandFather)
  - Inverse rules (e.g., Husband and Wife)
  - Symmetric rules (e.g., Similar)
  - Subrelation rules (e.g., Man => Person)

Category	Algorithm	FB15k MR	FB15k MRR	FB15k H@1	FB15k H@3	FB15k H@10	WN18 MR	WN18 MRR	WN18 H@1	WN18 H@3	WN18 H@10
KGE	TransE [3]	40	0.730	64.5	79.3	86.4	272	0.772	70.1	80.8	92.0
	DistMult [17]	42	0.798	-	-	89.3	655	0.797	-	-	94.6
	HolE [26]	-	0.524	40.2	61.3	73.9	-	0.938	93.0	94.5	94.9
	ComplEx [41]	-	0.692	59.9	75.9	84.0	-	0.941	93.6	94.5	94.7
	ConvE [8]	51	0.657	55.8	72.3	83.1	374	0.943	93.5	94.6	95.6
Rule-based	BLP [7]	415	0.242	15.1	26.9	42.4	736	0.643	53.7	71.7	83.0
	MLN [32]	352	0.321	21.0	37.0	55.0	717	0.657	55.4	73.1	83.9
Hybrid	RUGE [15]	-	0.768	70.3	81.5	86.5	-	-	-	-	-
	NNE-AER [9]	-	0.803	76.1	83.1	87.4	-	0.943	**94.0**	94.5	94.8
Ours	pLogicNet	**33**	0.792	71.4	85.7	90.1	255	0.832	71.6	94.4	95.7
	pLogicNet*	**33**	**0.844**	**81.2**	**86.2**	**90.2**	**254**	**0.945**	93.9	**94.7**	**95.8**

ExpressGNN (Zhang et al. 2019)

- Inference with graph neural networks



Source Codes

- pLogicNet: https://github.com/DeepGraphLearning/pLogicNet
- ExpressGNN: https://github.com/expressGNN/ExpressGNN

Roadmap

- Part I: Reasoning in Continuous Space
- Part II: Symbolic Logic Reasoning
- Part III: Neural-Symbolic Logic Reasoning
- **Part IV: Logic Rule Induction/Learning**

Learning Logic Rules

- Methods introduced so far:
  - Require given logic rules as input
  - Unable to discover logic rules automatically

- Learning logic rules:
  - Learn useful logic rules from existing knowledge graphs
- Foundation:
  - Inductive logic programming

Inductive Logic Programming

- Problem description:
  - Given: background facts B, positive examples P, negative examples N
  - Output: first-order logic rules H such that $B \wedge H \vDash P \quad B \wedge H \nvDash N$
    - Applying H to B yields all positive examples in P
    - Applying H to B yields none of negative examples in N
- Key idea: generate-and-test
  - Generate a set of candidate logic rules for reasoning
  - Choose the most useful logic rules from all candidates

Inductive Logic Programming

- Example:
  - Background facts: $\text{Father}(a, b)$ $\text{Father}(b, c)$ $\text{Father}(c, d)$
  - Positive facts: $\text{GrandFather}(a, c)$
  - Negative facts: $\text{GrandFather}(a, d)$

Rule Template

$$\forall X, Y, Z \quad \text{Grandfather}(X, Y) \leftarrow \text{Father}(X, Z) \wedge \text{Father}(Z, Y)$$

Consistent with pos/neg facts → Useful Rule

$$\forall X, Y, Z \quad \text{Grandfather}(X, Y) \leftarrow \text{Father}(X, U) \wedge \text{Father}(U, V) \wedge \text{Father}(V, Y)$$

Conflict with pos/neg facts → Unuseful Rule

Limitations of Traditional ILP

- Inability to handle noisy, erroneous or ambiguous data
  - E.g., mislabeled data in the positive or negative examples
- Neural ILP: combines the advantages of ILP and neural network-based systems:
  - data efficient
  - able to learn explicit human-readable symbolic rules
  - Robust to noisy and ambiguous data

Differentiable ILP (Evans et al. 2017)

- Key ideas:
  - Generate **candidate logic rules** according to pre-defined templates
  - Assign a scalar weight to each candidate rule
  - Perform **differentiable forward chaining** for reasoning
  - Choose **rules with large weights** as useful ones

Differentiable ILP (Evans et al. 2017)

- A differentiable extension of inductive logic programming:
  - Inductive logic programming:
    - The value of each ground atom is discrete (true/false)
    - The logic operators are discrete ($\neg \wedge \vee$)

  - Differentiable ILP:
    - Approximate the value of ground atoms with a continuous value in [0,1]
    - Approximate logic operators with differentiable operators
      - $x \vee y \approx \max\{x, y\}$ or $x \vee y \approx x + y - x \cdot y$ with $x, y \in [0,1]$
      - $x \wedge y = x \cdot y$
      - $\neg x = 1 - x$

Differentiable ILP (Evans et al. 2017)

- Apply forward chaining and all the candidate logic rules to the given facts, yielding a collection of new facts and predicted values.
  - Example:
    - Rules:
      - R1: $\text{Nationality}(X, Y) \leftarrow \text{BornIn}(X, Y)$ R2: $\text{Nationality}(X, Y) \leftarrow \text{LiveIn}(X, Y)$
    - Given facts: $\text{BornIn}(Bob, Canada)$ $\text{LiveIn}(Bob, USA)$
    - New facts: $\text{Nationality}(Bob, Canada)$ $\text{Nationality}(Bob, USA)$
- The value of each new fact is a function of rule weights
  - $\text{value}\big(\text{Nationality}(Bob, Canada)\big) = f_1(w)$
  - $\text{value}\big(\text{Nationality}(Bob, USA)\big) = f_2(w)$

Differentiable ILP (Evans et al. 2017)

- Adjust rule weights to minimize the difference between the ground-truth atom value and predicted atom value
  - Example:
    - Positive example (the value is 1): $\text{Nationality}(Bob, Canada)$
    - Negative example (the value is 0): $\text{Nationality}(Bob, USA)$
    - Predicted values:
      - $\text{value}(\text{Nationality}(Bob, Canada)) = f_1(w)$
      - $\text{value}(\text{Nationality}(Bob, USA)) = f_2(w)$
    - Cross-entropy loss:
      - $\ell(w) = -\{\log(f_1(w)) + \log(1 - f_2(w))\}$

Neural LP (Yang et al. 2017)

- Key ideas:
  - Generate **chain-like logic rules** up to a certain length as candidates
  - Assign a weight to each candidate with an **attention mechanism**
  - **Integrate all the candidate logic rules** for reasoning
  - Choose **rules with large weights** as useful ones

Neural LP (Yang et al. 2017)

- Chain-like logic rules:

$$\alpha \quad \mathrm{query}(\mathrm{Y},\mathrm{X}) \leftarrow \mathrm{R}_1(\mathrm{Y},\mathrm{Z}_1) \wedge \cdots \wedge \mathrm{R}_\mathrm{n}(\mathrm{Z}_\mathrm{n},\mathrm{X})$$

  - $\alpha \in [0,1]$: the confidence associated with this rule
  - n: the length of this rule

- Example:
  - $\mathrm{Nationality}(X,Y) \leftarrow \mathrm{LiveIn}(X,Z) \wedge \mathrm{CityOf}(Z,Y)$
  - $\mathrm{GrandFather}(X,Y) \leftarrow \mathrm{Father}(X,Z) \wedge \mathrm{Father}(Z,Y)$

Neural LP (Yang et al. 2017)

- Reasoning by matrix multiplication:
  - Assign an interger index to each entity
    - Let v_i be a one-hot vector with the entry of entity i being 1
    - Let M_R be a matrix in $\{0,1\}^{|E|\times|E|}$ such that the (i,j)-entry is 1 if and only if $R(i,j)$ is a given fact
  - During reasoning, for a rule $R(Y,X) \leftarrow P(Y,Z) \wedge Q(Z,X)$ and query $R(?,X)$, the answer can be obtained by:
    - Computing $s = M_P \cdot M_Q \cdot v_x$
    - Retrieving entities whose entries are nonzeros as answers

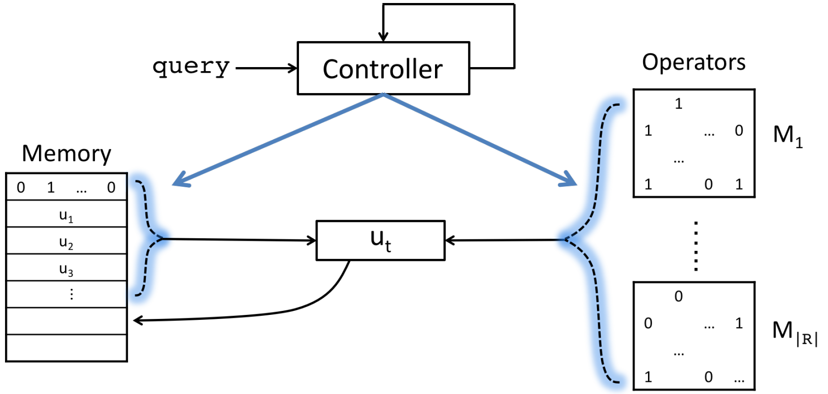
Neural LP (Yang et al. 2017)

- Integrating multiple rules for reasoning:
  - Consider:
    - A query R(? , X)
    - A set of logic rules $\{(\alpha_l, \beta_l = \mathrm{R}(\mathrm{Y},\mathrm{X}) \leftarrow \mathrm{R}_1(\mathrm{Y},\mathrm{Z}_1) \wedge \cdots \wedge \mathrm{R}_\mathrm{n}(\mathrm{Z}_\mathrm{n},\mathrm{Y}))\}_l$

  - Apply **backward chaining** for reasoning:
    - Each rule *l* gives a score over all entities $\mathrm{s}_l = \alpha_l(\prod_{\mathrm{R}_\mathrm{k} \in \mathrm{Body}(\beta_l)} \mathrm{M}_{\mathrm{R}_\mathrm{k}})\mathrm{v}_\mathrm{x}$
    - Combing all rules yields $\mathrm{s} = \sum_l \mathrm{s}_l = \sum_l(\alpha_l(\prod_{\mathrm{R}_\mathrm{k} \in \mathrm{Body}(\beta_l)} \mathrm{M}_{\mathrm{R}_\mathrm{k}})\mathrm{v}_\mathrm{x})$
    - The value of the i-entry in s is the score received by entity i

Neural LP (Yang et al. 2017)

- Maintain a set of auxiliary memory vectors $\mathbf{u}_t$
- Memory attention vector $\mathbf{b}_t$
- Operator attention vector $\mathbf{a}_t$



$$\mathbf{u_0} = \mathbf{v_x}$$

$$\mathbf{u_t} = \sum_{\mathrm{k}}^{|\mathbf{R}|} a_t^{\mathrm{k}} \mathbf{M}_{\mathrm{R_k}} \left(\sum_{\tau=0}^{t-1} b_t^{\tau} \mathbf{u}_{\tau} \right) \quad \text{for } 1 \leq t \leq T$$

$$\mathbf{u_{T+1}} = \sum_{\tau=0}^{T} b_{T+1}^{\tau} \mathbf{u}_{\tau}$$

$$\mathbf{h_t} = \text{update}\left(\mathbf{h_{t-1}}, \text{input}\right)$$

$$\mathbf{a_t} = \text{softmax}\left(W\mathbf{h_t} + b\right)$$

$$\mathbf{b_t} = \text{softmax}\left(\left[\mathbf{h_0}, \ldots, \mathbf{h_{t-1}}\right]^T \mathbf{h_t}\right)$$

Neural LP (Yang et al. 2017)

- Main results:
  - Neural LP outperforms many knowledge graph embedding methods

	WN18		FB15K		FB15KSelected	
	MRR	Hits@10	MRR	Hits@10	MRR	Hits@10
Neural Tensor Network	0.53	66.1	0.25	41.4	-	-
TransE	0.38	90.9	0.32	53.9	-	-
DistMult [29]	0.83	94.2	0.35	57.7	**0.25**	**40.8**
Node+LinkFeat [25]	0.94	94.3	**0.82**	87.0	0.23	34.7
Implicit ReasoNets [23]	-	**95.3**	-	**92.7**	-	-
Neural LP	**0.94**	94.5	0.76	83.7	0.24	36.2

Neural LP (Yang et al. 2017)

- Case study:
  - The learned logic rules are quite intuitive

1.00	partially_contains (C, A) ← contains (B, A) ∧ contains (B, C)
0.45	partially_contains (C, A) ← contains (A, B) ∧ contains (B, C)
0.35	partially_contains (C, A) ← contains (C, B) ∧ contains (B, A)
1.00	marriage_location (C, A) ← nationality (C, B) ∧ contains (B, A)
0.35	marriage_location (B, A) ← nationality (B, A)
0.24	marriage_location (C, A) ← place_lived (C, B) ∧ contains (B, A)
1.00	film_edited_by (B, A) ← nominated_for (A, B)
0.20	film_edited_by (C, A) ← award_nominee (B, A) ∧ nominated_for (B, C)

Neural LP (Yang et al. 2017)

- Inductive knowledge graph reasoning (Hit@10):
  - The learned rules can be used in other knowledge graphs for reasoning

	WN18	FB15K	FB15KSelected
TransE	0.01	0.48	0.53
Neural LP	**94.49**	**73.28**	**27.97**

Limitation

- Idea:
  - Consider a large number of candidate logic rules
  - Learn the weights of these rules jointly
- Limitation:
  - High dimensionality
  - The weights may not reflect the important of rules precisely

RNNLogic (Qu and Chen et al. 2020)

- A new rule learning approach 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." *arXiv preprint arXiv:2010.04029* (2020).

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

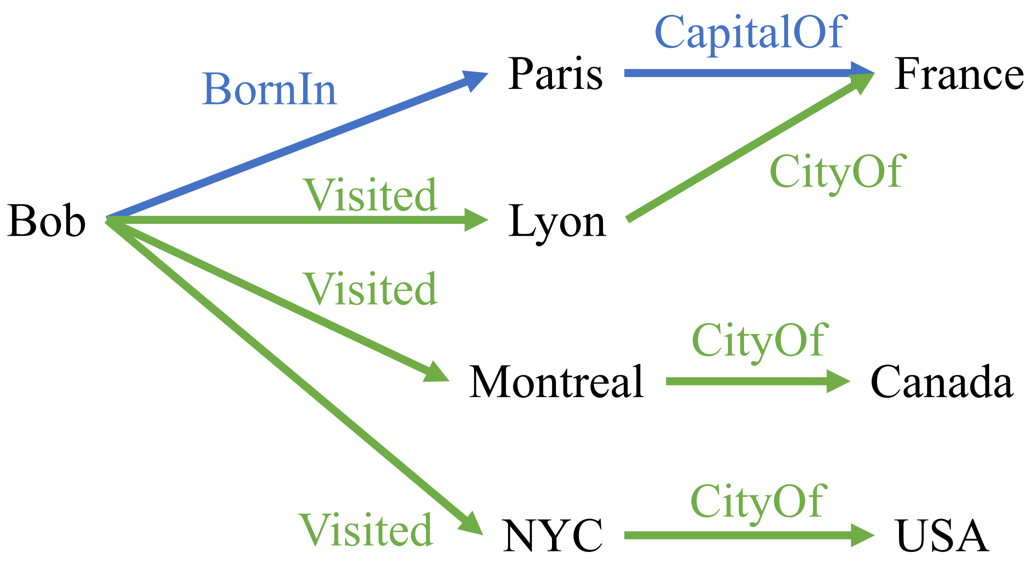
  - Objective function: $\max_{w,\theta} \mathcal{O}(w, \theta) = \mathbb{E}_{(\mathcal{G},\boldsymbol{q},\boldsymbol{a})\sim p_{\mathrm{data}}}\left[\log p_{w,\theta}(\boldsymbol{a}|\mathcal{G}, \boldsymbol{q})\right]$

Rule Generator $p_\theta(\mathbf{z}|\mathbf{q})$

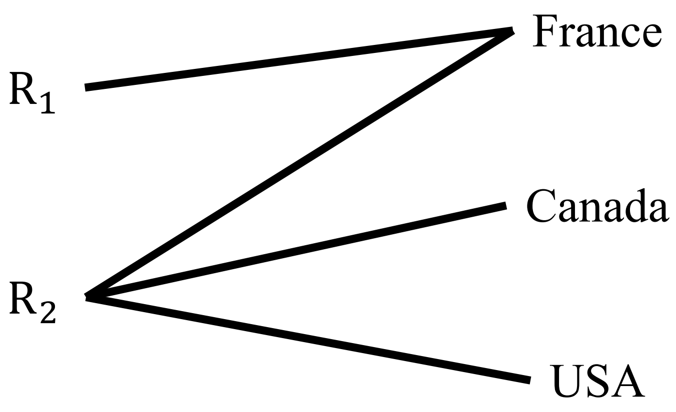
- Each chain-like rule can be represented as a sequence of relations:
  - $\mathrm{r}(X_0, X_l) \leftarrow \mathrm{r}_1(X_0, X_1) \wedge \mathrm{r}_2(X_1, X_2) \wedge \cdots \wedge \mathrm{r}_l(X_{l-1}, X_l)$
  - $[\mathrm{r}, \mathrm{r}_1, \mathrm{r}_2, \ldots, \mathrm{r}_l, \mathrm{r}_{\mathrm{END}}]$ 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 $\boldsymbol{q} = (h, \mathrm{r}, ?)$, we can use rules in $\boldsymbol{z}$ to get a search tree:
  - Query: $\boldsymbol{q} = (\mathrm{Bob}, \mathrm{Nationality}, ?)$
  - Logic rules in $\boldsymbol{z}$:
    - 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

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

Scalar weight of each rule

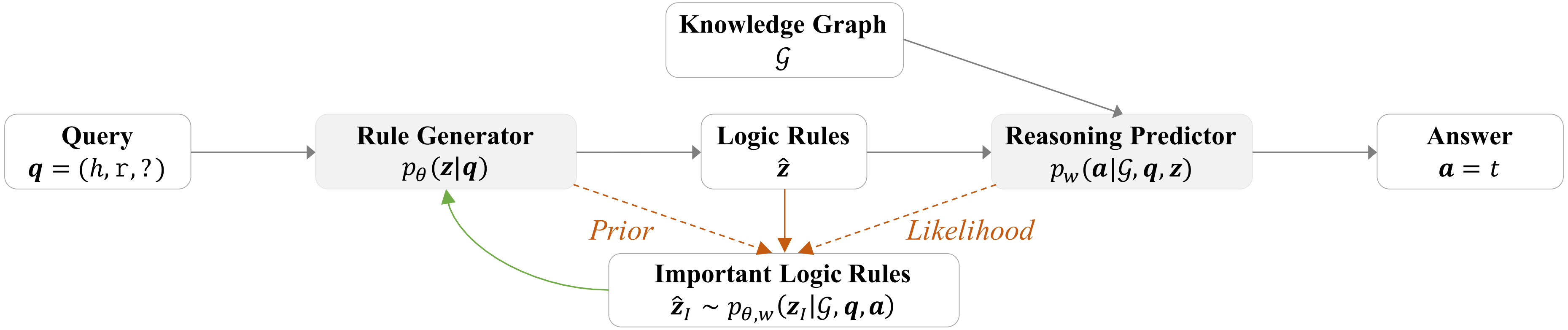
Score of each relational path, either a constant or computed with embeddings

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

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 E-step

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

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

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 $H(rule)$

Optimization M-step

- Goal of M-step:
  - Use the identified important rules $\hat{\mathbf{z}}_I$ 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 $\log p_w(\boldsymbol{a} = t | \mathcal{G}, \boldsymbol{q}, \hat{\mathbf{z}}_I)$
  - Rule generator:
    - Maximize $\log p_\theta(\hat{\mathbf{z}}_I | \boldsymbol{q}) = \sum_{rule \in \hat{\mathbf{z}}_I} \log \mathrm{RNN}_\theta(rule | r)$

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

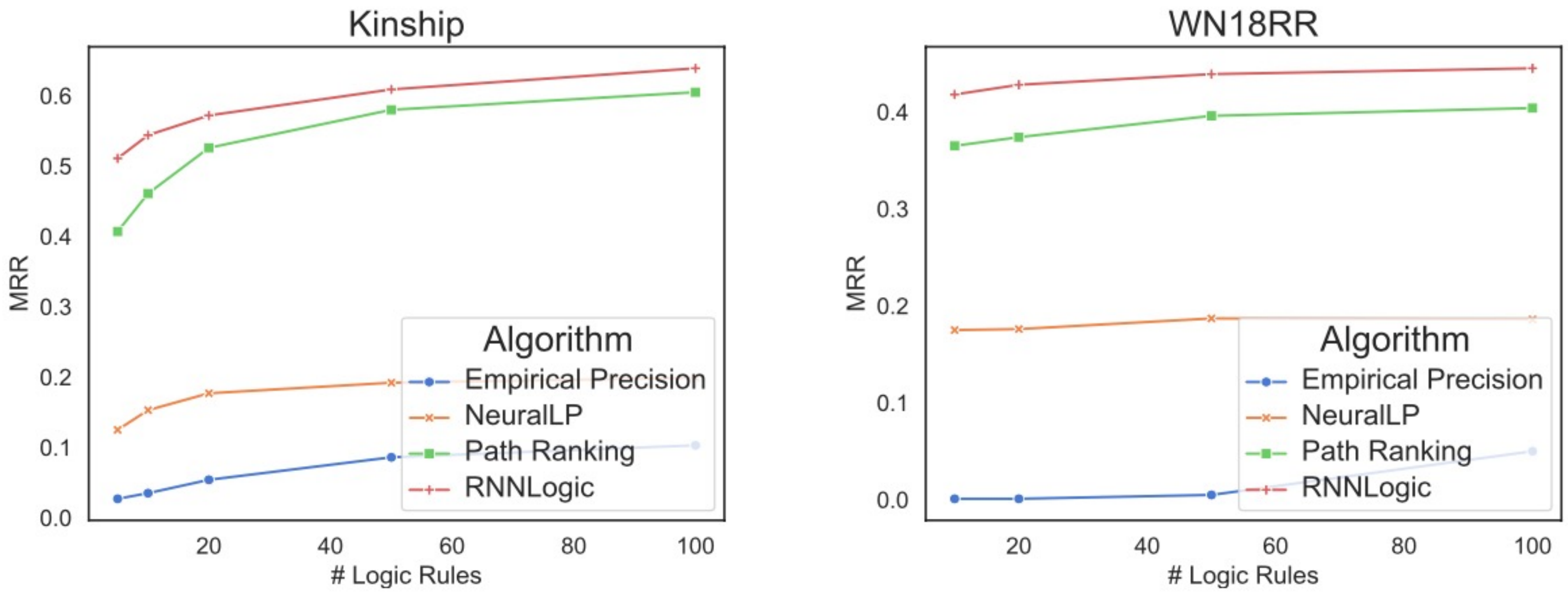
Main Results on Kinship and UMLS

- 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

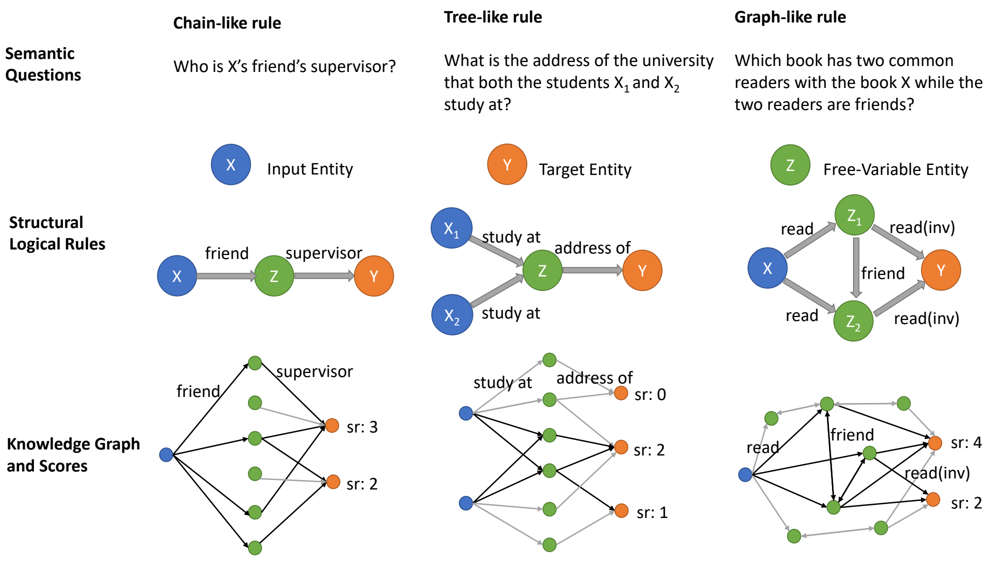
Appears_in_TV_Show$(X,Y) \leftarrow$ Has_Actor(X,Y)
Appears_in_TV_Show$(X,Y) \leftarrow$ Creator_of$(X,U) \wedge$ Has_Producer$(U,V) \wedge$ Appears_in_TV_Show(V,Y)
ORG._in_State$(X,Y) \leftarrow$ ORG._in_City$(X,U) \wedge$ City_Locates_in_State(U,Y)
ORG._in_State$(X,Y) \leftarrow$ ORG._in_City$(X,U) \wedge$ Address_of_PERS.$(U,V) \wedge$ Born_in$(V,W) \wedge$ Town_in_State(W,Y)
Person_Nationality$(X,Y) \leftarrow$ Born_in$(X,U) \wedge$ Place_in_Country(U,Y)
Person_Nationality$(X,Y) \leftarrow$ Student_of_Educational_Institution$(X,U) \wedge$ ORG._Endowment_Currency$(U,V) \wedge$ Currency_Used_in_Region$(V,W) \wedge$ Region_in_Country(W,Y)

More Examples of Learned Rules

Relation	$\leftarrow$	Rule (*Explanation*)
$X \xrightarrow{\texttt{Appears\_in\_TV\_Show}} Y$	$\leftarrow$	$X \xleftarrow{\texttt{Actor\_of}} Y$ (*Definition. An actor of a show appears in the show, obviously.*)
	$\leftarrow$	$X \xrightarrow{\texttt{Creator\_of}} U \xleftarrow{\texttt{Producer\_of}} V \xrightarrow{\texttt{Appears\_in\_TV\_Show}} Y$ (*The creator X and the producer V of another show U are likely to appear in the same show Y.*)
	$\leftarrow$	$X \xleftarrow{\texttt{Actor\_of}} U \xleftarrow{\texttt{Award\_Nominated}} V \xleftarrow{\texttt{Winner\_of}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Writer\_of}} U \xleftarrow{\texttt{Creater\_of}} V \xrightarrow{\texttt{Actor\_of}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Student\_of}} U \xleftarrow{\texttt{Student\_of}} V \xrightarrow{\texttt{Appears\_in\_TV\_Show}} Y$ (*Two students X and V in the same school U are likely to appear in the same show Y.*)
$X \xrightarrow{\texttt{ORG.\_in\_State}} Y$	$\leftarrow$	$X \xrightarrow{\texttt{ORG.\_in\_City}} U \xrightarrow{\texttt{City\_in\_State}} Y$ (*Use the city to indicate the state directly.*)
	$\leftarrow$	$X \xrightarrow{\texttt{ORG.\_in\_City}} U \xleftarrow{\texttt{Lives\_in}} V \xrightarrow{\texttt{Born\_in}} W \xrightarrow{\texttt{Town\_in\_State}} Y$ (*Use the person living in the city to induct the state.*)
	$\leftarrow$	$X \xleftarrow{\texttt{Sub-ORG.\_of}} U \xrightarrow{\texttt{ORG.\_in\_State}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Sub-ORG.\_of}} U \xleftarrow{\texttt{Sub-ORG.\_of}} V \xrightarrow{\texttt{ORG.\_in\_State}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{ORG.\_in\_City}} U \xleftarrow{\texttt{ORG.\_in\_City}} V \xrightarrow{\texttt{ORG.\_in\_State}} Y$ (*Two organizations in the same city are in the same state.*)
$X \xrightarrow{\texttt{Person\_Nationality}} Y$	$\leftarrow$	$X \xrightarrow{\texttt{Born\_in}} U \xrightarrow{\texttt{Place\_in\_Country}} Y$ (*Definition.*)
	$\leftarrow$	$X \xrightarrow{\texttt{Spouse}} U \xrightarrow{\texttt{Person\_Nationality}} Y$ (*By a fact that people are likely to marry a person of same nationality.*)
	$\leftarrow$	$X \xrightarrow{\texttt{Student\_of}} U \xrightarrow{\texttt{ORG.\_Endowment\_Currency}} V \xleftarrow{\texttt{Region\_Currency}} W \xrightarrow{\texttt{Region\_in\_Country}} Y$ (*Use the currency to induct the nationality.*)
	$\leftarrow$	$X \xrightarrow{\texttt{Born\_in}} U \xleftarrow{\texttt{Born\_in}} V \xrightarrow{\texttt{Person\_Nationality}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Politician\_of}} U \xleftarrow{\texttt{Politician\_of}} V \xrightarrow{\texttt{Person\_Nationality}} Y$
$X \xrightarrow{\texttt{Manifestation\_of}} Y$	$\leftarrow$	$X \xleftarrow{\texttt{Treats}} U \xrightarrow{\texttt{Prevents}} V \xleftarrow{\texttt{Precedes}} Y$
	$\leftarrow$	$X \xleftarrow{\texttt{Complicates}} U \xleftarrow{\texttt{Precedes}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Location\_of}} U \xrightarrow{\texttt{Is\_a}} V \xleftarrow{\texttt{Precedes}} Y$
	$\leftarrow$	$X \xleftarrow{\texttt{Complicates}} U \xrightarrow{\texttt{Precedes}} V \xleftarrow{\texttt{Occurs\_in}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Location\_of}} U \xleftarrow{\texttt{Occurs\_in}} V \xleftarrow{\texttt{Occurs\_in}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Precedes}} U \xleftarrow{\texttt{Occurs\_in}} V \xleftarrow{\texttt{Degree\_of}} Y$
$X \xleftarrow{\texttt{Affects}} Y$	$\leftarrow$	$X \xrightarrow{\texttt{Result\_of}} U \xrightarrow{\texttt{Occurs\_in}} V \xrightarrow{\texttt{Precedes}} Y$
	$\leftarrow$	$X \xleftarrow{\texttt{Precedes}} U \xrightarrow{\texttt{Produces}} V \xleftarrow{\texttt{Occurs\_in}} Y$
	$\leftarrow$	$X \xleftarrow{\texttt{Prevents}} U \xrightarrow{\texttt{Disrupts}} V \xrightarrow{\texttt{Co-occurs\_with}} Y$
	$\leftarrow$	$X \xleftarrow{\texttt{Result\_of}} U \xrightarrow{\texttt{Complicates}} V \xrightarrow{\texttt{Precedes}} Y$
	$\leftarrow$	$X \xleftarrow{\texttt{Assesses\_Effect\_of}} U \xrightarrow{\texttt{Method\_of}} V \xrightarrow{\texttt{Complicates}} Y$
	$\leftarrow$	$X \xrightarrow{\texttt{Process\_of}} U \xrightarrow{\texttt{Interacts\_with}} V \xrightarrow{\texttt{Causes}} Y$
	$\leftarrow$	$X \xleftarrow{\texttt{Assesses\_Effect\_of}} U \xleftarrow{\texttt{Result\_of}} V \xrightarrow{\texttt{Precedes}} Y$

Beyond Chain-like Rules

- Tree-like rules:
  - Learn to Explain Efficiently via Neural Logic Inductive Learning
  - (Yang and Song, 2020)
- Graph-lile rules:
  - Differentiable Learning of Graph-like Logical Rules from Knowledge Graphs
  - (ICLR 2021 anomalous submission)



Other Rule Learning Approaches

- Neural logic machines (Dong et al. 2019)
- Neural theorem provers (Rocktäschel and Riedel, 2017)
- Relation-set following (Cohen et al, 2019)
- Path ranking (Lao and Cohen, 2010)
- DeepPath (Xiong et al. 2017)
- DIVA (Chen et al. 2018)
- Probabilistic personalized page rank (Wang et al. 2013)
- AMIE+ (Galárraga et al. 2015)

Conclusion

- Part I: Reasoning in Continuous Space
  - TransE, TransR, RotatE
- Part II: Symbolic Logic Reasoning
  - Logic programming
  - Probabilistic logic programming (Markov Logic Networks)
- Part III: Neural-Symbolic Logic Reasoning
  - pLogicNet, ExpressGNN
- Part IV: Logic Rule Induction/Learning
  - Inductive logic programming
  - Neural logic programming
  - RNNLogic

Future Directions

- Few-shot Learning
  - Can we reason with a few limited number of facts for each relation
- Integrate text + knowledge graph for reasoning
  - Unstructured data are huge but noisy
- Combining System I and II reasoning
  - Knowledge graph reasoning are mainly System II reasoning
  - How to integrate with system I (perception)
- …

References

- Knowledge Graph Embedding

  - Bordes, Antoine, et al. "Translating embeddings for modeling multi-relational data." *Advances in neural information processing systems*. 2013.
  - Sun, Zhiqing, et al. "Rotate: Knowledge graph embedding by relational rotation in complex space." *arXiv preprint arXiv:1902.10197* (2019).
  - Wang, Zhen, et al. "Knowledge graph embedding by translating on hyperplanes." *Aaai*. Vol. 14. No. 2014. 2014.
  - Nguyen, Dat Quoc, et al. "Stranse: a novel embedding model of entities and relationships in knowledge bases." *arXiv preprint arXiv:1606.08140* (2016).
  - Yang, Bishan, et al. "Embedding entities and relations for learning and inference in knowledge bases." *arXiv preprint arXiv:1412.6575* (2014).
  - Trouillon, Théo, et al. "Complex embeddings for simple link prediction." International Conference on Machine Learning (ICML), 2016.
  - Nickel, Maximilian, Lorenzo Rosasco, and Tomaso Poggio. "Holographic embeddings of knowledge graphs." *arXiv preprint arXiv:1510.04935* (2015).
  - Dettmers, Tim, et al. "Convolutional 2d knowledge graph embeddings." *arXiv preprint arXiv:1707.01476* (2017).
  - Zhang, Shuai, et al. "Quaternion knowledge graph embeddings." *Advances in Neural Information Processing Systems*. 2019.

References

- Symbolic Logic Reasoning

  - Cussens, James. "Parameter estimation in stochastic logic programs." *Machine Learning* 44.3 (2001): 245-271.
  - Kersting, Kristian, and Luc De Raedt. "Bayesian logic programs." *arXiv preprint cs/0111058* (2001).
  - Richardson, Matthew, and Pedro Domingos. "Markov logic networks." *Machine learning* 62.1-2 (2006): 107-136.
  - Cohen, William W., Fan Yang, and Kathryn Rivard Mazaitis. "Tensorlog: Deep learning meets probabilistic dbs." *arXiv preprint arXiv:1707.05390* (2017).
  - Manhaeve, Robin, et al. "Deepproblog: Neural probabilistic logic programming." *Advances in Neural Information Processing Systems*. 2018.
  - Skryagin, Arseny, et al. "Sum-Product Logic: Integrating Probabilistic Circuits into DeepProbLog."

References

- Neural & Symbolic Logic Reasoning

  - Qu, Meng, and Jian Tang. "Probabilistic logic neural networks for reasoning." *Advances in Neural Information Processing Systems*. 2019.
  - Zhang, Yuyu, et al. "Efficient probabilistic logic reasoning with graph neural networks." *arXiv preprint arXiv:2001.11850* (2020).

References

- Logic rule induction/learning

  - Qu, Meng*, Chen, Junkun*, Xhonneux Louis-Pascal, Bengio Yoshua, and Tang, Jian. "RNNLogic: Learning Logic Rules for Reasoning on Knowledge Graphs." *arXiv preprint arXiv:2010.04029* (2020).
  - Lao, Ni, Tom Mitchell, and William Cohen. "Random walk inference and learning in a large scale knowledge base." *Proceedings of the 2011 conference on empirical methods in natural language processing*. 2011.
  - Wang, William Yang, Kathryn Mazaitis, and William W. Cohen. "Programming with personalized pagerank: a locally groundable first-order probabilistic logic." *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2013.
  - Galárraga, Luis, et al. "Fast rule mining in ontological knowledge bases with AMIE $$$$." *The VLDB Journal* 24.6 (2015): 707-730.
  - Rocktäschel, Tim, and Sebastian Riedel. "End-to-end differentiable proving." *Advances in Neural Information Processing Systems*. 2017.
  - Xiong, Wenhan, Thien Hoang, and William Yang Wang. "Deeppath: A reinforcement learning method for knowledge graph reasoning." *arXiv preprint arXiv:1707.06690* (2017).
  - Evans, Richard, and Edward Grefenstette. "Learning explanatory rules from noisy data." *Journal of Artificial Intelligence Research* 61 (2018): 1-64.
  - Yang, Fan, Zhilin Yang, and William W. Cohen. "Differentiable learning of logical rules for knowledge base reasoning." *Advances in Neural Information Processing Systems*. 2017.
  - Chen, Wenhu, et al. "Variational knowledge graph reasoning." *arXiv preprint arXiv:1803.06581* (2018).

References

- Logic rule induction/learning

  - Yang, Yuan, and Le Song. "Learn to Explain Efficiently via Neural Logic Inductive Learning." *arXiv preprint arXiv:1910.02481* (2019).
  - Dong, Honghua, et al. "Neural logic machines." *arXiv preprint arXiv:1904.11694* (2019).
  - Cohen, William W., et al. "Scalable Neural Methods for Reasoning With a Symbolic Knowledge Base." *arXiv preprint arXiv:2002.06115* (2020).

Thanks!

Contact: jian.tang@hec.ca