RNNLogic- Learning Logic Rules for Reasoning on Knowledge Graphs

Published as a conference paper at ICLR 2021

Meng Qu, Junkun Chen, Louis-Pascal Xhonneux, Yoshua Bengio, Jian Tang

2024-09-12

HoonUi Lee

Contents

Previous work

End-to-End differentiable model

이산적 공간의 structure와 연속적 공간의 score를

Gradient based optimization을 통해 학습시키자

NeuralLP, DRUM

End-to-End differentiable model한 RNN 기반 모델

large search space -> High-quality의 logic rule을 찾기 어려움

RNNLogic

Rule Generator p_θ

다양한 rule body의 rule 생성

Reasoning Predictor p_w

생성된 rule에 대한 scoring

+ EM algorithm

Rule Generator, Reasoning Predictor의

parameter를 단계적으로 학습

RNNLogic

Objective function

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

G: background knowledge graph (triplets)
q: query (h, r, ?)
a: answer (a = t)
z: rule generator가 정의한 latent rules

주어진 G, q에 대하여 z로 얻은 answer가

실제 answer **a**를 예측할 확률을 최대화

RNNLogic

Rule generator

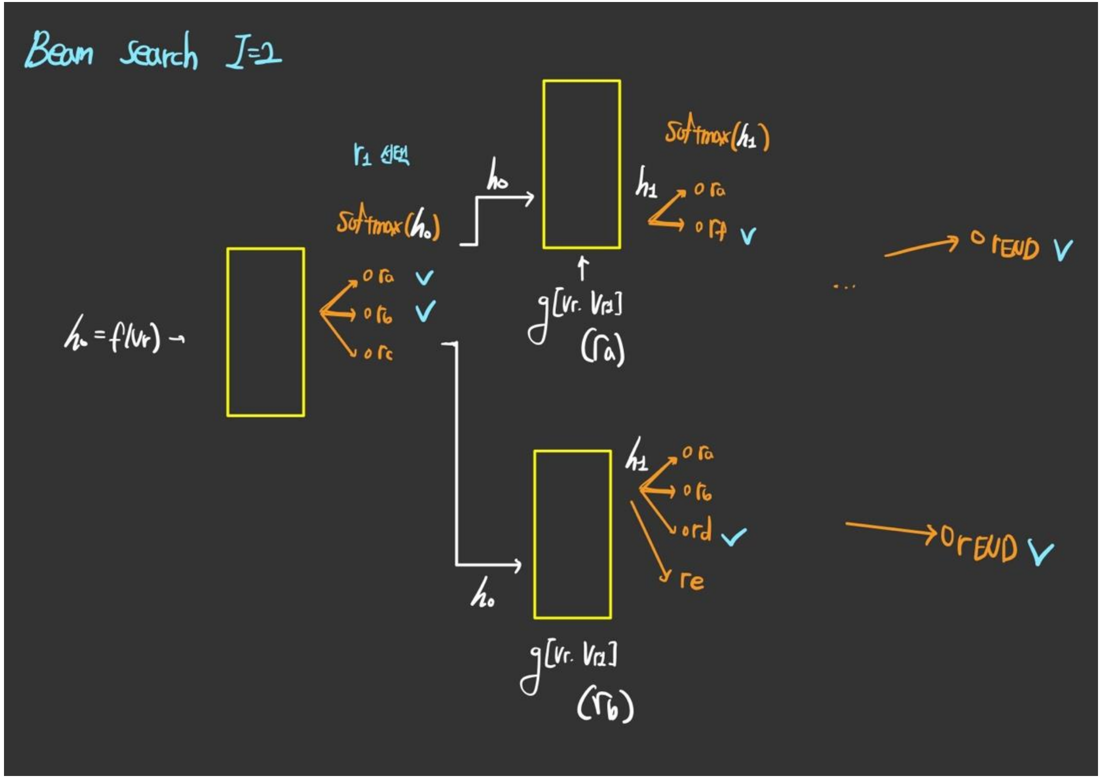
$$p_\theta(\boldsymbol{z}|\boldsymbol{q}) = \mathbf{Mu}(\boldsymbol{z}|N, \mathbf{RNN}_\theta(\cdot|\mathrm{r}))$$

Rule Generator는 LSTM 기반 RNN 모델 사용

(Hyperparameter) N개의 rule에 대한 multinomial distribution 정의

=> 규칙들이 각기 다른 확률을 가지고 sampling 됨

RNNLogic



$$\boldsymbol{h}_0 = f(\boldsymbol{v}_{\mathrm{r}}),$$

$$\mathrm{softmax}(o(\boldsymbol{h}_{t+1}))$$

$$\boldsymbol{h}_t = \mathrm{LSTM}(\boldsymbol{h}_{t-1}, g([\boldsymbol{v}_{\mathrm{r}}, \boldsymbol{v}_{\mathrm{r}_t}]),$$

RNNLogic

$$\left.\begin{array}{ll} \text{top 1.} & z_1 \, [r. \; r_{11} \; r_{z12} \cdots , r_{END}] \\ \text{top.2} & z_2 \, [r. \; r_{z21} \; r_{z22} \cdots , r_{END}] \end{array}\right\} \hat{z}$$

RNNLogic

Reasoning Predictor

$$\texttt{score}_w(e) = \sum_{rule \in \boldsymbol{z}} \texttt{score}_w(e|rule) = \sum_{rule \in \boldsymbol{z}} \sum_{path \in \mathcal{P}(h, rule, e)} \psi_w(rule) \cdot \phi_w(path)$$

Rule에 대한 learnable parameter와 Path에 대한 score 존재

Path score

RotatE를 사용하여 h부터 rule body에 대한 embedding을 움직여

Candidate answer e의 embedding과 거리로 점수 계산

$$\phi_w(path) = \sigma(\delta - d(\boldsymbol{x}_{e_0} \circ \boldsymbol{x}_{r_1} \circ \boldsymbol{x}_{r_2} \circ \cdots \circ \boldsymbol{x}_{r_l}, \boldsymbol{x}_{e_l}))$$

RNNLogic

Reasoning Predictor

$$p_w(\boldsymbol{a} = e|\mathcal{G}, \boldsymbol{q}, \boldsymbol{z}) = \frac{\exp(\mathtt{score}_w(e))}{\sum_{e' \in \mathcal{A}} \exp(\mathtt{score}_w(e'))}$$

Softmax function을 통해

다른 candidate answer들 간 score에 대한 확률 계산

RNNLogic

EM algorithm

E-step: 주어진 **현재 parameter**를 **바탕**으로 latent 변수의 **기댓값** 계산

M-step: E-step에서 계산된 latent 변수의 기댓값을 이용해 **parameter**를 **최대화**

RNNLogic

EM algorithm

E-step : 현재 step에서 rule generator가 생성한 z hat에서 high quality의 rule 식별

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

G, q, a가 주어졌을 때, 좋은 rule z를 찾는 과정

by posterior probability

RNNLogic

EM algorithm

but, Rule에 대한 multinomial distribution 형식으로 나타내기 어려움

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

$$\left| \log p_{\theta,w}(\boldsymbol{z}_I|\mathcal{G},\boldsymbol{q},\boldsymbol{a}) - \left(\sum_{\textit{rule}\in\boldsymbol{z}_I} H(\textit{rule}) + \gamma(\boldsymbol{z}_I) + \mathrm{const} \right) \right| \leq s^2 + O(s^4)$$

$$q(\boldsymbol{z}_I) \propto \exp\left(\sum_{\textit{rule}\in\boldsymbol{z}_I} H(\textit{rule}) + \gamma(\boldsymbol{z}_I)\right)$$

RNNLogic

EM algorithm

$$\exp(H(rule))/(\sum_{rule' \in \hat{\boldsymbol{z}}} \exp(H(rule')))$$

$\Downarrow$

Top – K 개의 rule이 sampling되어 $\hat{\boldsymbol{z}}_I$ 구성

RNNLogic

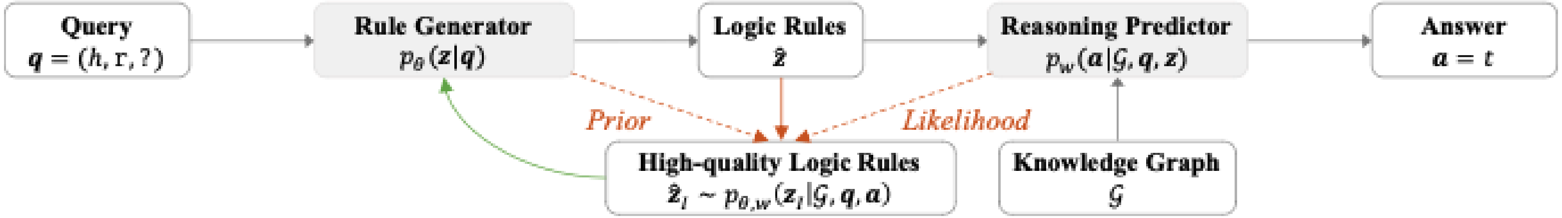
EM algorithm

M-step : high quality의 rule을 통해 Rule Generator를 update

$$\mathcal{O}_{(\mathcal{G},\boldsymbol{q},\boldsymbol{a})}(\theta) = \log p_\theta(\hat{\boldsymbol{z}}_I|\boldsymbol{q}) = \sum_{rule \in \hat{\boldsymbol{z}}_I} \log \mathrm{RNN}_\theta(rule|r) + \text{const.}$$

Objective function: Rule generator가 high quality rule을 생성할 확률을 최대화

RNNLogic



Query
$q = (h, r, ?)$
Rule Generator
$p_\theta(z|q)$
Logic Rules
$\hat{z}$
Reasoning Predictor
$p_w(a|\mathcal{G}, q, z)$
Answer
$a = t$
Prior
Likelihood
High-quality Logic Rules
$\hat{z}_I \sim p_{\theta,w}(z_I|\mathcal{G}, q, a)$
Knowledge Graph
$\mathcal{G}$

Experiment

Evaluation Metrics: (h, r, ?), (t, r^-1, ?)

RNNLogic+ : Aggregator + MLP + KGE 활용한 scoring 선택

$$\mathrm{score}_w(e) = \mathrm{MLP}(\mathrm{AGG}(\{\mathbf{v}_{rule}, |\mathcal{P}(h, rule, e)|\}_{rule \in \hat{z}_I}) + \eta\, \mathrm{KGE}(h, r, e).$$

Inverse relation 제거된 datset

$$\phi_w(path) = 1$$

$$\phi_w(path) = \sigma(\delta - d(\boldsymbol{x}_{e_0} \circ \boldsymbol{x}_{r_1} \circ \boldsymbol{x}_{r_2} \circ \cdots \circ \boldsymbol{x}_{r_l}, \boldsymbol{x}_{e_l}))$$

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	-	-	-	-	-
	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
RNNLogic+	w/o emb.	480	0.299	21.5	32.8	46.4	7204	0.489	45.3	50.6	56.3
	with emb.	178	0.349	25.8	38.5	53.3	4624	**0.513**	**47.1**	**53.2**	**59.7**

relation / inverse relation 에 대한 평가

Experiment

Evaluation Metrics

Category	Algorithm	FB15k-237					WN18RR				
		MR	MRR	H@1	H@3	H@10	MR	MRR	H@1	H@3	H@10
Rule Learning	MINERVA*	-	0.293	21.7	32.9	45.6	-	0.448	41.3	45.6	51.3
	MultiHopKG*	-	0.407	32.7	-	56.4	-	0.472	43.7	-	54.2
RNNLogic	w/o emb.	459.0	0.377	28.9	41.2	54.9	7662.8	0.478	43.8	50.3	55.3
	with emb.	**146.1**	**0.443**	**34.4**	**48.9**	**64.0**	**3767.0**	**0.506**	**46.3**	**52.3**	**59.2**

(h, r, ?)만 사용하는 모델들과도 비교

Experiment

Evaluation Metrics

Category	Algorithm	Kinship MR	Kinship MRR	Kinship H@1	Kinship H@3	Kinship H@10	UMLS MR	UMLS MRR	UMLS H@1	UMLS H@3	UMLS 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**

Experiment

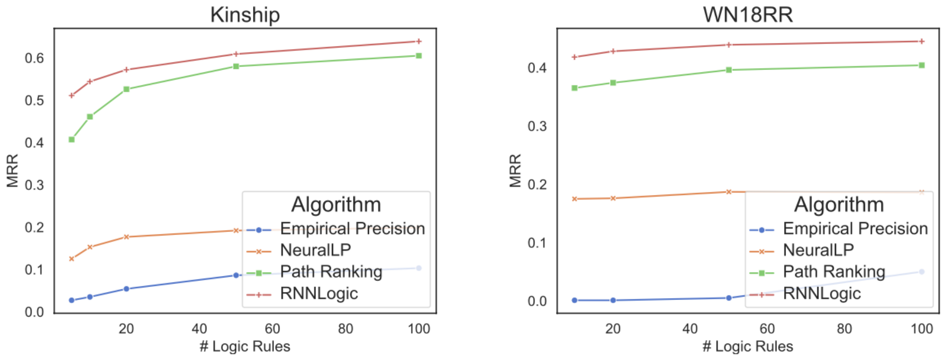


Figure 2: Performance w.r.t. # logic rules. RNNLogic achieves competitive results even with 10 rules per query relation.

주어진 query relation r에 대해 10 개의 rule 생성하여 추론한 결과

Experiment

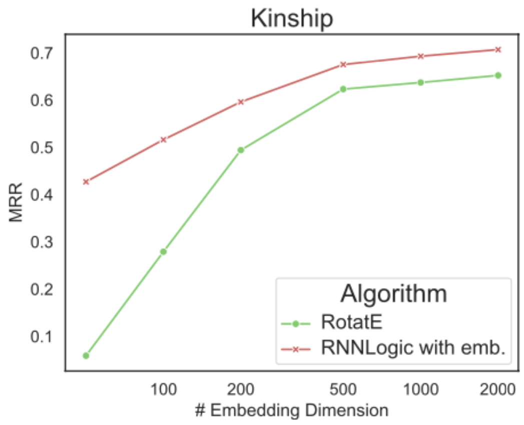


Figure 3: Performance w.r.t. embedding dimension.

모든 embedding 차원에서 RotatE 능가

Experiment

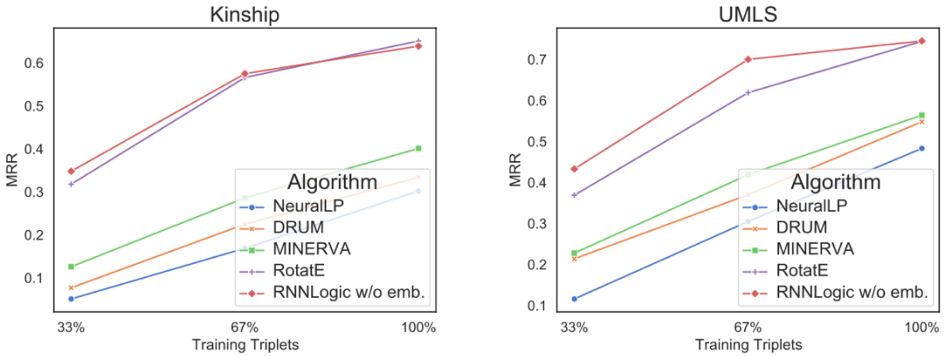


Figure 4: Performance w.r.t. # training triplets. RNNLogic is more robust to data sparsity even without using embeddings.

훈련시킨 triplet 개수에 대한 효율성

Conclusion

NeuralLP, DRUM에서 채택한 end-to-end differentiable은 search space가 커지는 문제 발생

RNNLogic은 Rule Generator, Reasoning Predictor를 통해 rule 생성과 scoring을 분리

EM algorithm을 활용하여 parameter 최적화

Experiment를 통해 high-quality의 rule을 통한 answer 추론이 효과적임