Go for a walk and arrive at the answer: reasoning over paths in knowledge bases using reinforcement learning

October 11, 2019

Overview

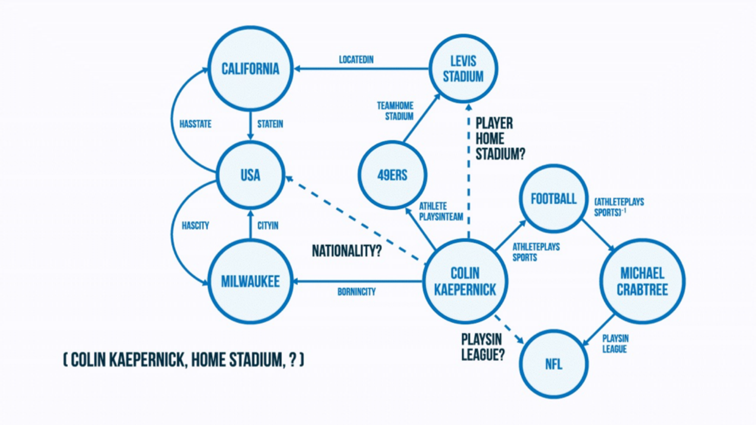
- Objective : Graph completion in knowledge base/ knowledge graph
- Related works
- Task and Model
- Experiments
- Discussion

Objective

- Problem in knowledge graph (KG): Incomplete
- Graph completion: It is a task to determine the potential relation between entities. A popular approach to KG completion is to infer new relations by combinatory reasoning over the information found along other paths connecting a pair of entities, or evaluating the truth of a proposed triple.
- Practical task: Query answering where relation is known with only one entity.

Objective

Example of inferring for graph completion :



where solid edges are observed and dashed edges are part of queries.

Objective

Question: "What is the nationality of Colin Kaepernick?"
Query answering form: (Colin Kaepernick, Nationality, ?)
This paper: Authors proposed a neural reinforcement learning approach (MINERVA[1]) which learns how to navigate the graph conditioned on the input query to find predictive paths.

[1] Meandering In Networks of Entities to Reach Verisimilar Answers

Related work

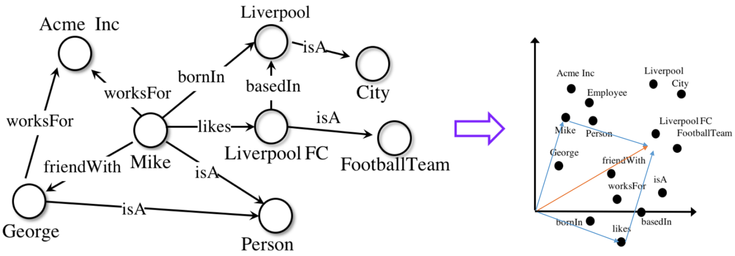
Symbolic representations methods:
Stanley Kok and Pedro Domingos. Statistical predicate invention. In ICML, 2007
Ni Lao, Tom Mitchell, and William Cohen. Random walk inference and learning in a large scale knowledge base. In EMNLP, 2011.
$\rightarrow$ Poor performance

Related work

Symbolic representations methods:
Stanley Kok and Pedro Domingos. Statistical predicate invention. In ICML, 2007
Ni Lao, Tom Mitchell, and William Cohen. Random walk inference and learning in a large scale knowledge base. In EMNLP, 2011.
→ Poor performance
Graph embedding methods:
Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In NIPS, 2013.
Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. In NIPS, 2013.

Related work

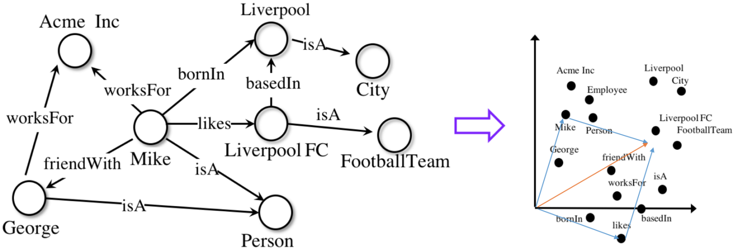
Simple illustration for graph embedding methods (TranE):



Query answering form: (Mike, likes, ?)

Related work

Simple illustration for graph embedding methods (TranE):



Query answering form: (Mike, likes, ?)

$\rightarrow$ good performance but unable to capture chains of reasoning expressed by KB paths.

Task and Model

Notation :

- Set of entities : $\mathscr{E}$
- Set of binary relations : $\mathscr{R}$
- Triplet (e_1, r, e_2) : $e_i \in \mathscr{E}, i = 1, 2,\ r \in \mathscr{R}$
- Inverse triplet (e_1, r^{-1}, e_2).
- Knowledge graph: $G = (\mathscr{E}, E, \mathscr{R})$ where $E \subset \mathscr{E} \times \mathscr{R} \times \mathscr{E}$
- Query answer problem: complete $(e_{1q}, r_q, ?)$.

Task and Model

Reinforcement learning model: Authors specified a deterministic partially observed Markov decision process (POMDP), which is a 5-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, R)$, from the KG.

- **State** $\mathcal{S}$ consists of all valid combinations in $\mathscr{E} \times \mathscr{E} \times \mathscr{R} \times \mathscr{E}$. For $S \in \mathcal{S}$, S can be denoted as $(e_t, e_{1q}, r_q, e_{2q})$, where e_{1q}, r_q is the query, e_{2q} is the answer, e_t is the current location of the the RL agent

Task and Model

Reinforcement learning model: Authors specified a deterministic partially observed Markov decision process (POMDP), which is a 5-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, R)$, from the KG.

- **State** $\mathcal{S}$ consists of all valid combinations in $\mathscr{E} \times \mathscr{E} \times \mathscr{R} \times \mathscr{E}$. For $S \in \mathcal{S}$, S can be denoted as $(e_t, e_{1q}, r_q, e_{2q})$, where e_{1q}, r_q is the query, e_{2q} is the answer, e_t is the current location of the the RL agent
- **Observations** $\mathcal{O}$: Agent knows its current location e_t and query e_{1q}, r_q but not the answer e_{2q}. Therefore, $\mathcal{O} : \mathcal{S} \rightarrow \mathscr{E} \times \mathscr{E} \times \mathscr{R}$, i.e. $\mathcal{O}(s = (e_t, e_{1q}, r_q, e_{2q})) = (e_t, e_{1q}, r_q)$.

Task and Model

Reinforcement learning model: Authors specified a deterministic partially observed Markov decision process (POMDP), which is a 5-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, R)$, from the KG.

- **State** $\mathcal{S}$ consists of all valid combinations in $\mathscr{E} \times \mathscr{E} \times \mathscr{R} \times \mathscr{E}$. For $S \in \mathcal{S}$, S can be denoted as $(e_t, e_{1q}, r_q, e_{2q})$, where e_{1q}, r_q is the query, e_{2q} is the answer, e_t is the current location of the the RL agent
- **Observations** $\mathcal{O}$: Agent knows its current location e_t and query e_{1q}, r_q but not the answer e_{2q}. Therefore, $\mathcal{O} : \mathcal{S} \rightarrow \mathscr{E} \times \mathscr{E} \times \mathscr{R}$, i.e. $\mathcal{O}(s = (e_t, e_{1q}, r_q, e_{2q})) = (e_t, e_{1q}, r_q)$.
- **Action** $\mathcal{A}_S = \{(e_t, r, v) \in E : S = (e_t, e_{1q}, r_q, e_{2q}), r \in \mathscr{R}, v \in \mathscr{E}\} \cup \{(s, \emptyset, s)\}$.

Task and Model

Reinforcement learning model: Authors specified a deterministic partially observed Markov decision process (POMDP), which is a 5-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, R)$, from the KG.

- **Transition** $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defined by $\mathcal{P}(S, A) = (v, e_{1q}, r_q, e_{2q})$ where $S = (e_t, e_{1q}, r_q, e_{2q})$ and $A = (e_t, r, v)$

Task and Model

Reinforcement learning model: Authors specified a deterministic partially observed Markov decision process (POMDP), which is a 5-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{P}, R)$, from the KG.

- **Transition** $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ defined by $\mathcal{P}(S, A) = (v, e_{1q}, r_q, e_{2q})$ where $S = (e_t, e_{1q}, r_q, e_{2q})$ and $A = (e_t, r, v)$
- **Reward** R: $R_T = 1$ if, in the final state, the current location e_t is equal to e_{2q}. Otherwise, reward is zero.

Task and Model

To solve the POMDP, authors designed a randomized non-stationary history-dependent policy $\pi = (d_1, d_2, .., d_{T-1})$, where $d_t : H_t \rightarrow \mathcal{A}_{S_t}$ and the history $H_t = (H_t, A_{t-1}, O_t)$ is the sequence of observations and actions taken.

Task and Model

To solve the POMDP, authors designed a randomized non-stationary history-dependent policy $\pi = (d_1, d_2, .., d_{T-1})$, where $d_t : H_t \to \mathcal{A}_{S_t}$ and the history $H_t = (H_t, A_{t-1}, O_t)$ is the sequence of observations and actions taken.

Technique Detail:

In practice, authors restricted to policies parameterized by long short-term memory network (LSTM).

Suppose the agent encode the history H_t as $\mathbf{h}_t \in \mathbb{R}^{2d}$, relations are embedded as $\mathbf{r} \in \mathbb{R}^{|\mathscr{R}| \times d}$ and entities are embedded as $\mathbf{e} \in \mathbb{R}^{|\mathscr{E}| \times d}$. The history embedding for $H_t = (H_t, A_{t-1}, O_t)$ is updated according to LSTM dynamics:

$$\mathbf{h}_t = LSTM(\mathbf{h}_{t-1}, [\mathbf{a}_{t-1}; \mathbf{o}_{t-1}])$$

where $\mathbf{a}_{t-1} = \mathbf{r}_{A_{t-1}}$ and $\mathbf{o}_t = \mathbf{e}_{e_t}$ if $O_t = (e_t, e_{1q}, r_q)$.

Task and Model

To solve the POMDP, authors designed a randomized non-stationary history-dependent policy $\pi = (d_1, d_2, .., d_{T-1})$, where $d_t : H_t \rightarrow \mathcal{A}_{S_t}$ and the history $H_t = (H_t, A_{t-1}, O_t)$ is the sequence of observations and actions taken.

Technique Detail:

Based on the history embedding $\mathbf{h}_t$, the policy network makes the decision to choose an action from all available actions. Typically,

$$\mathbf{d}_t = softmax(\mathbf{A}_t(W_2 \; Relu(W_1[\mathbf{h}_t; \mathbf{o}_t; \mathbf{r}_q])))$$
$$A_t \sim Categorical(\mathbf{d}_t)$$

where $\mathbf{A}_t$ is obtained from the embedding matrix of the available actions for state S_t; W_1, W_2 are trainable variables.

Reinforcement learning: training

Author maximized the expected reward:

$$J(\theta) = \mathbb{E}_{(e_1,r,e_2)\sim D}\mathbb{E}_{A_1,..,A_{T-1}\sim\pi_\theta}[R(S_T)|S_1 = (e_1, e_1, r, e_2)]$$

where assuming there is a true underlying distribution $(e_1, r, e_2) \sim D$, θ denotes all trainable variable in policy. Specially, authors use the REINFORCE algorithms (Williams, 1992) to solve the optimization problem.

Experiment

Criteria:

- Hit Ratio ($HR@k$): $HR@k$ is defined as $\frac{\#hit}{n}$, where n is the number of tests and hit is the number of cases that the hidden entity in the test case is ranked in top-k in the produced ranking list by a recommender system.
- Mean reciprocal rank (MRR) : The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of n queries: $MRR = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{rank_i}$

Experiment

Illustration for criteria:

Query	Proposed Results	Correct response	Rank	Reciprocal rank
cat	catten, cati, **cats**	cats	3	1/3
tori	torii, **tori**, toruses	tori	2	1/2
virus	**viruses**, virii, viri	viruses	1	1

$$HR@1 = \frac{1}{3}$$
$$HR@2 = \frac{2}{3}$$
$$HR@3 = \frac{3}{3}$$
$$MRR = (1/3 + 1/2 + 1)/3 = 11/18$$

Information of dataset:

Dataset	#entities	#relations	#facts	#queries	#degree	
					avg.	median
COUNTRIES	272	2	1158	24	4.35	4
UMLS	135	49	5,216	661	38.63	28
KINSHIP	104	26	10686	1074	82.15	82
WN18RR	40,945	11	86,835	3134	2.19	2
NELL-995	75,492	200	154,213	3992	4.07	1
FB15K-237	14,505	237	272,115	20,466	19.74	14
WikiMovies	43,230	9	196,453	9952	6.65	4

Table 1: Statistics of various datasets used in experiments.

Experiment

Result for small dataset:

Data	Metric	ComplEx	ConvE	DistMult	NTP	NTP-λ	NeuralLP	MINERVA
KINSHIP	HITS@1	0.754	0.697	0.808	0.500	0.759	0.475	0.605
	HITS@3	0.910	0.886	0.942	0.700	0.798	0.707	0.812
	HITS@10	0.980	0.974	0.979	0.777	0.878	0.912	0.924
	MRR	0.838	0.797	0.878	0.612	0.793	0.619	0.720
UMLS	HITS@1	0.823	0.894	0.916	0.817	0.843	0.643	0.728
	HITS@3	0.962	0.964	0.967	0.906	0.983	0.869	0.900
	HITS@10	0.995	0.992	0.992	0.970	1.000	0.962	0.968
	MRR	0.894	0.933	0.944	0.872	0.912	0.778	0.825

Table 3: Query answering results on KINSHIP and UMLS datasets.

Experiment

Result for large dataset:

Data	Metric	ComplEx	ConvE	DistMult	NeuralLP	Path-Baseline	MINERVA
WN18RR	HITS@1	0.382	0.403	0.410	0.376	0.017	0.413
	HITS@3	0.433	0.452	0.441	0.468	0.025	0.456
	HITS@10	0.480	0.519	0.475	0.657	0.046	0.513
	MRR	0.415	0.438	0.433	0.463	0.027	0.448
FB15K-237	HITS@1	0.303	0.313	0.275	0.166	0.169	0.217
	HITS@3	0.434	0.457	0.417	0.248	0.248	0.329
	HITS@10	0.572	0.600	0.568	0.348	0.357	0.456
	MRR	0.394	0.410	0.370	0.227	0.227	0.293
NELL-995	HITS@1	0.612	0.672	0.610	-	0.300	0.663
	HITS@3	0.761	0.808	0.733	-	0.417	0.773
	HITS@10	0.827	0.864	0.795	-	0.497	0.831
	MRR	0.694	0.747	0.680	-	0.371	0.725

Table 4: Query answering results on WN18RR, FB15K-237 and NELL-995 datasets. NeuralLP does not scale to NELL-995 and hence the entries are kept blank.

Ability to learn chain:

(i) **Can learn general rules:**

(S1) LocatedIn(X, Y) $\leftarrow$ LocatedIn(X, Z) & LocatedIn(Z, Y)
(S2) LocatedIn(X, Y) $\leftarrow$ NeighborOf(X, Z) & LocatedIn(Z, Y)
(S3) LocatedIn(X, Y) $\leftarrow$ NeighborOf(X, Z) & NeighborOf(Z, W) & LocatedIn(W, Y)

(ii) **Can learn shorter path:** Richard F. Velky $\xrightarrow{\text{WorksFor}}$?

Richard F. Velky $\xrightarrow{\text{PersonLeadsOrg}}$ Schaghticokes $\xrightarrow{\text{NO-OP}}$ Schaghticokes $\xrightarrow{\text{NO-OP}}$ Schaghticokes

(iii) **Can recover from mistakes:** Donald Graham $\xrightarrow{\text{WorksFor}}$?

Donald Graham $\xrightarrow{\text{OrgTerminatedPerson}}$ TNT Post $\xrightarrow{\text{OrgTerminatedPerson}^{-1}}$ Donald Graham $\xrightarrow{\text{OrgHiredPerson}}$ Wash Post

Table 8: A few example of paths found by MINERVA on the COUNTRIES and NELL. MINERVA can learn general rules as required by the COUNTRIES dataset (example (i)). It can learn shorter paths if necessary (example (ii)) and has the ability to correct a previously taken decision (example (iii))

Discussion

- A new way of automated reasoning on large KG by training the agent to walk in KG with RL.
- Achieve state-of-the-art results on multiple benchmark knowledge base completion tasks.
- MINERVA can learn long chains-of-reasoning.
- Future research directions include applying more sophisticated RL techniques and working directly on textual queries and documents.