Efficient Distributed Knowledge Representation Learning for Large Knowledge Graphs

Lele Chai[1] Xin Wang[1,2,*] Baozhu Liu[1] Yajun Yang[1,2]

1 School of Computer Science and Technology, Tianjin University, China

2 Tianjin Key Laboratory of Cognitive Computing and Application, China
lelechai@tju.edu.cn

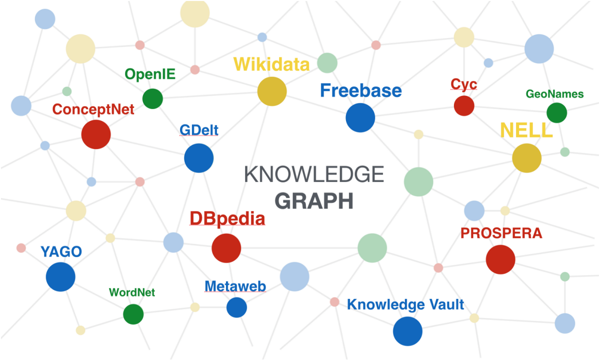
August 2, 2019

Contents

Contents

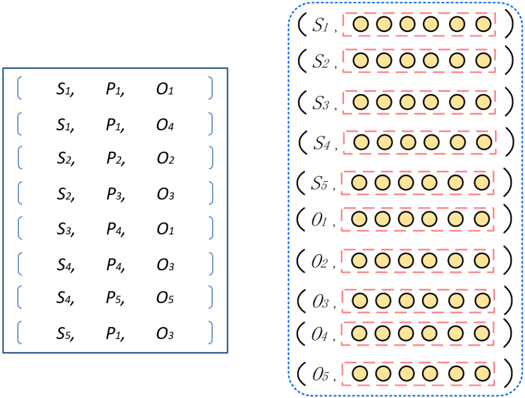
Motivation

1. With the massive growth of linked Web data, the scale of knowledge graphs has been growing dramatically
2. The traditional triple representations can behave well in maintaining the KG structures and expressiveness
3. Difficulties in executing the downstream tasks, such as link prediction
4. Low computing performance and high sparsity issue



Motivation

1. The existing knowledge embedding algorithms cannot be applied efficiently to large-scale KG datasets
2. Lacking a unified framework to integrate current KRL models to facilitate the realization of embeddings for various applications



Related Work

The existing research work on representative KRL models:

1. **Traditional KRL Models**

2. **The Uniform Training Framework**

Related Work

The existing research work on representative KRL models:

1. **Traditional KRL Models**
   - Structured Embedding, distance based **[AAAI 2011]**
   - Neural Network Model, nonlinear operation is adopted **[NIPS 2013]**
   - Semantic energy Model, projection matrices are defined **[ML 2013]**
   - Translational Model, translation based **[NIPS 2013]**
2. **The Uniform Training Framework**
   - KB2E, single machine based **[AAAI 2015]**
   - OpenKE, high efficiency of a GPU depended **[EMNLP 2015]**
   - SANSA, low accuracy and efficiency **[ISWC 2017]**

Related Work

The existing research work on representative KRL models:

1. **Traditional KRL Models**
   - Structured Embedding, distance based **[AAAI 2011]**
   - Neural Network Model, nonlinear operation is adopted **[NIPS 2013]**
   - Semantic energy Model, projection matrices are defined **[ML 2013]**
   - Translational Model, translation based **[NIPS 2013]**
2. **The Uniform Training Framework**
   - KB2E, single machine based **[AAAI 2015]**
   - OpenKE, high efficiency of a GPU depended **[EMNLP 2015]**
   - SANSA, low accuracy and efficiency **[ISWC 2017]**

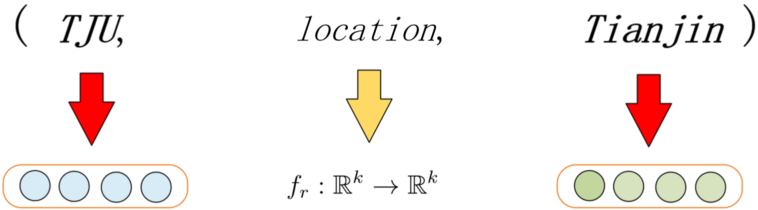
However, the existing approaches have not been completely to take full advantage of the distributed learning scenario of the uniform training framework

Contributions

1. A distributed algorithm template framework for knowledge representation learning by using Spark.
2. The training process in the DKRL framework can meet large-scale datasets and high dimensional requirement for the translational models
3. The extensive experiments for verifying the efficiency of the proposed method

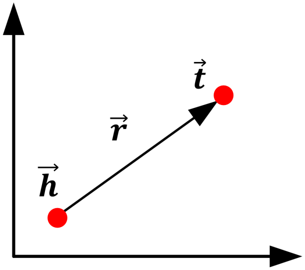
Contents

- An example representation learning for embedding

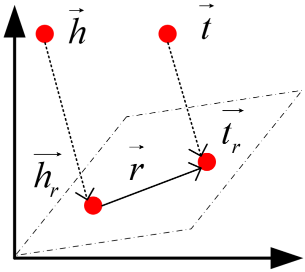


Contents

- Traditional KRL embedding methods



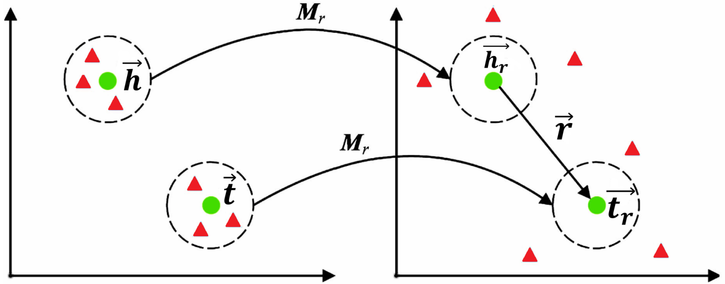
TransE [NIPS 2013]



TransH [AAAI 2014]

KRL embedding methods

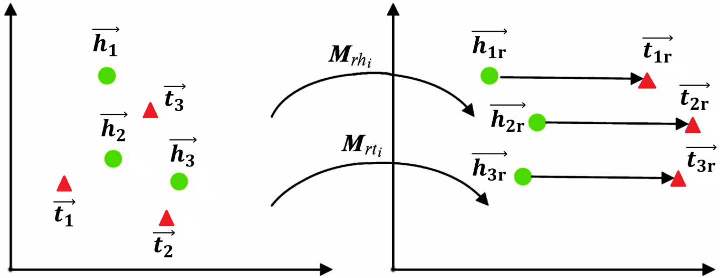
- Traditional KRL embedding methods



TransR [AAAI 2015]

KRL embedding methods

- Traditional KRL embedding methods



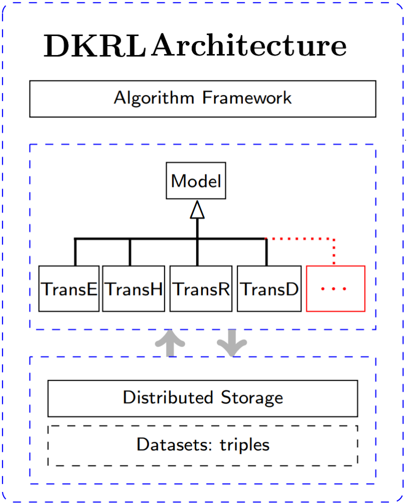
TransD [ACL 2015]

KRL embedding methods

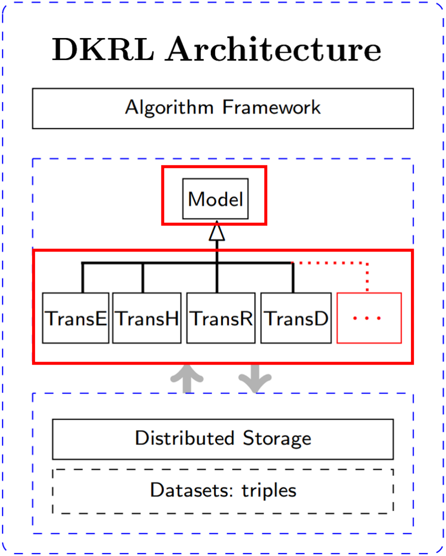
- The framework function signatures

Function	Description
$dataPrepare(S)$	To load data, where S is the input triple set.
$dataInit(DR, X)$	To initialize DR according to the model type X.
$trainRun(\gamma, \alpha, X)$	To start to train model X, γ is the loss function parameter, and α is the learning rate.
$sample(IS, seed)$	To generate positive samples from the dataset IS, IS denotes that its data type is *Integer*, *seed* represents the random seed.
$negative(IS, V, seed)$	To generate negative samples from the dataset IS.
$D(T)$	To calculate the distance of those triples T.

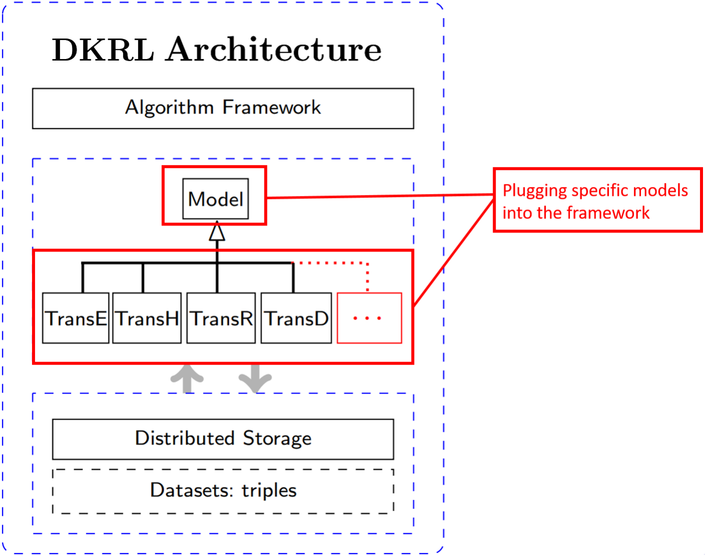
The DKRL uniform algorithm framework



The DKRL uniform algorithm framework

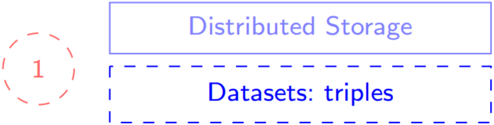


The DKRL uniform algorithm framework



DKRL Training Procedure

Training Procedure



DKRL Training Procedure

Training Procedure

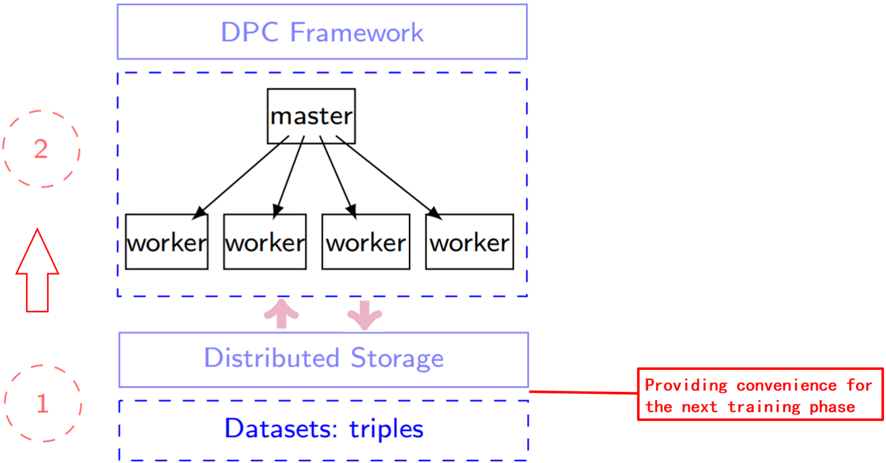
1

Distributed Storage

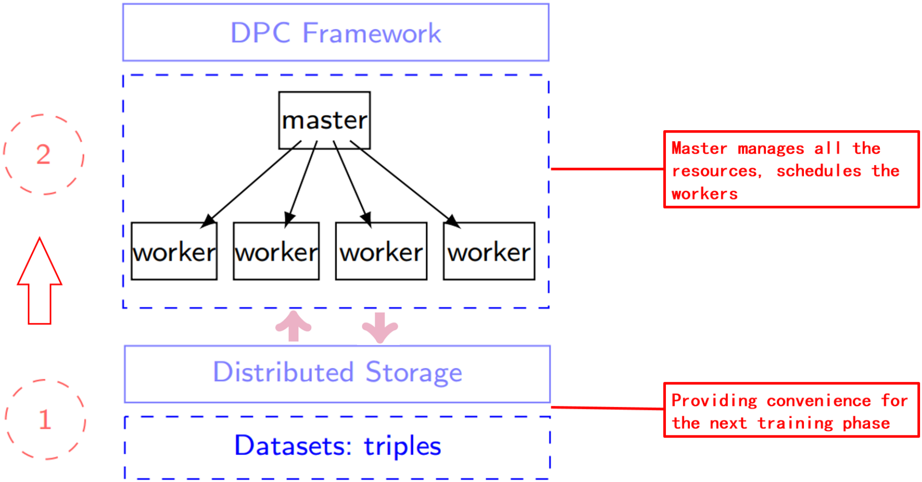
Datasets: triples

Providing convenience for the next training phase

Training Procedure

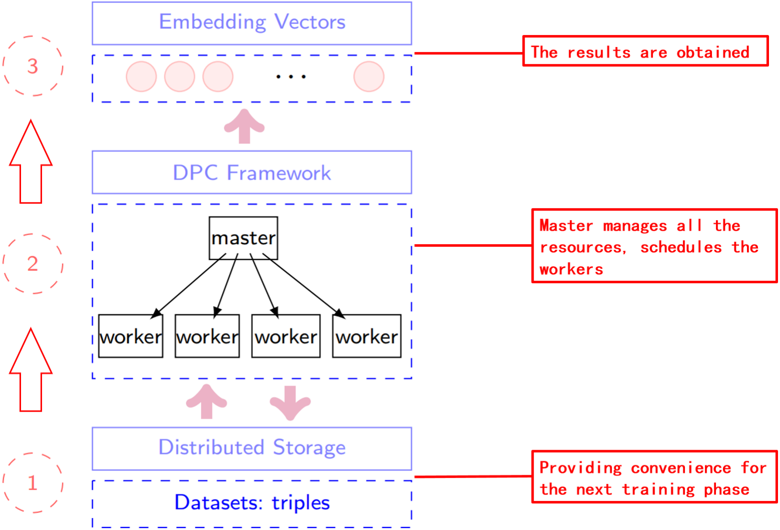


Training Procedure



DKRL Training Procedure

Training Procedure



Functions of the Framework

Trans(X) \ Function	**dataInit()**	**negative()**	**D(T)**
TransE [NIPS 2013]	$\vec{t}$: tail vector $\vec{h}$: head vector $\vec{r}$: relation vector	• Uniform	$-\Vert\vec{h}+\vec{r}-\vec{t}\Vert_{L_1/L_2}$
TransH [AAAI 2014]	$\vec{t}$: tail vector $\vec{h}$: head vector $\vec{n}_r$: norm vector $\vec{r}$: relation vector	• Uniform • Bernoulli	$\vec{h}_p = \vec{n}_r{}^\top \cdot \vec{h} \cdot \vec{n}_r$ $\vec{t}_p = \vec{n}_r{}^\top \cdot \vec{t} \cdot \vec{n}_r$ $-\Vert\vec{h}_p+\vec{r}-\vec{t}_p\Vert_{L_1/L_2}$
TransR [AAAI 2015]	$\vec{t}$: tail vector $\vec{h}$: head vector $\vec{r}$: relation vector M_r : projection matrix	• Uniform • Bernoulli	$-\Vert\vec{M}_r \cdot \vec{h}+\vec{r}-\vec{M}_r \cdot \vec{t}\Vert_{L_1/L_2}$
TransD [ACL 2015]	$\vec{t}$: tail vector $\vec{h}$: head vector $\vec{r}$: relation vector $\vec{W}_h$: mapping vector $\vec{W}_t$: mapping vector $\vec{W}_r$: mapping vector	• Uniform • Bernoulli	$\vec{h}_p = \vec{W}_r \cdot \vec{W}_h^\top \cdot \vec{h}$ $\vec{t}_p = \vec{W}_r \cdot \vec{W}_t^\top \cdot \vec{t}$ $\Vert\vec{h}_p+\vec{r}-\vec{t}_p\Vert_{L_1/L_2}$

DKRL Algorithm

Algorithm 1: DKRL-Traning `/*train in each site in parallel*/`

Input : RDF triple Sets $S = \{(h, r, t)\}$, where $h, t \in V$ and $r \in E$

Parameter: max training iterations epo, embedding dimension k, learning rate α, margin γ

Output : Embedding results of S: $Vec(S) = \{(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})\}$

$Vec(S) \leftarrow \emptyset$;

Function dataPrepare (S)

  $rdd : RDD[DT] \leftarrow$ load S;

  $DR(Ih, Ir, It) \leftarrow$ convert RDD to DR;

dataInit $(DR(Ih, Ir, It), Trans(X))$; /* $X \in \{E, H, R, D\}$ */

Function trainRun $(\gamma, \alpha, Trans(X))$

  foreach *computing site* s_i **do**

    repeat

      $T_p \leftarrow$ sample$(IS, rand_seed[s_i])$;

      $T_n \leftarrow$ negative$(IS, V, rand_seed[s_i])$;

      $t \leftarrow \mid T_p \mid$;

      while $t > 0$ **do**

        $T \leftarrow T \setminus DR(Ih, Ir, It)$; /* $T \in \{T_p, T_n\}$ */

        map$(\emptyset, DR)$ s.t. $DR \in T$;

        D$(T) \leftarrow$ reduce$(\emptyset, (N_1, N_2))$; /* $N_1, N_2 \subseteq T \wedge N_1 \cap N_2 = \emptyset$ */

      Loss $\leftarrow \gamma +$ D(T_p) $-$ D(T_n);

      if $Loss > 0$ **then**

        $\{(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})\} \leftarrow$ update $\{(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})\}$ w.r.t. GD;

      else

        $\{(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})\}$;

    until epo;

$Vec(S) \leftarrow Vec(S) \cup \{(\boldsymbol{h}, \boldsymbol{r}, \boldsymbol{t})\}$;

return $Vec(S)$;

The training type Trans(*X*) is regarded as a parameter

Contents

Experiment Setting

- Our methods were implemented in Scala using Spark, which were deployed on an 8-site cluster.
    - 4-core CPU and 16GB memory
    - 64-bit CentOS Linux operating system
    - Java 1.8, Scala 2.11, Hadoop 2.7.4, and Spark 2.2.0

Dataset

- **Datasets used in the experiments**

Dataset	#Rel	#Ent	#Train	#Valid	#Test
FB15K	1,345	14,951	483,142	50,000	59,071
WN18	18	40,943	141,142	5,000	5,000
DBpedia	663	5,526,330	18,295,010	50,000	50,000

Evaluation Metrics

- **Raw Mean Rank & Hits@N**
  - Suppose there are n triples in the dataset
  - Replace the head or tail entity in a triple t, generate n triples
  - Calculate the energy values $\vec{h} + \vec{r} - \vec{t}$, obtain n energy values
  - Sort the n energy values in ascending order
  - Record the sequence number of k the energy value of the triplet t
  - Repeat the process for all triples
  - Average out the sequence numbers of each correct triple
  - Calculate the ratio of the energy sequence of the correct triples to less than N

- **Filtered Mean Rank**
  - Suppose there are n triples in the dataset
  - Replace the head or tail entity in a triple t, generate n triples
  - Calculate the energy values $\vec{h}+\vec{r}-\vec{t}$, obtain n energy values
  - Sort the n energy values in ascending order
  - Record the sequence number k of the energy value of the triplet t
  - If m triples in the first $k-1$ energy corresponding to the triple are correct, the serial number of the triple t is to $k-m$
  - Repeat the process for all triples
  - Average out the sequence numbers of each correct triple
  - Calculate the ratio of the energy sequence of the correct triples to less than N

Effectiveness of The Algorithms in Accuracy

- **Link prediction effectiveness results on WN18 and FB15K**

Dataset	**WN18**		**FB15K**	
metrics	**Mean Rank**		**Mean Rank**	
	Raw	**Filt**	**Raw**	**Filt**
TransE (unif)	247	254	206	117
TransE (bern)	232	302	195	102
TransH (unif)	287	295	259	127
TransH (bern)	304	382	215	94
TransR (unif)	263	245	232	106
TransR (bern)	242	238	197	89
TransD (unif)	253	241	224	84
TransD (bern)	243	218	189	96

Effectiveness of The Algorithms in Accuracy

- **Hits@N effectiveness results on DBpedia**

Metric	Hits@10	Hits@20	Hits@50	Hits@100
Head	4.8	5.1	5.2	5.7
Tail	21.9	25.0	32.3	44.6

- The methods SANSA and KB2E report **out-of-memory**

Effectiveness of The Algorithms in Accuracy

- **Raw Mean Rank effectiveness results on FB15K**

Metrics	**Raw Mean Rank**			
Methods	**Baseline**	**SANSA**	**KB2E**	**DKRL**
TransE	243	7438	210	**206**
TransH	**211**	—	221	215
TransR	226	—	208	**197**
TransD	194	—	**184**	189

Efficiency of The Algorithms in Time

- **Time results on FB15K**

Models	**Time (s)**
TransE (SANSA)	36596.940
TransE (DKRL)	**7080.633**

Contents

Conclusion

- An efficient unified training distributed framework DKRL for KG embedding training
  - Incorporating various existing KRL models
  - Accelerating training under the strategy of distributed experimental settings
  - A set of primitive interface functions is defined
- The extensive experiments were conducted on both synthetic and real-world datasets, which have verified the effectiveness and efficiency of our method

THANK YOU!

Q&A