Mining Inference Formulas by Goal-Directed Random Walks

Zhuoyu Wei[1,2], **Jun Zhao**[1,2] and **Kang Liu**[1]
[1] National Laboratory of Pattern Recognition, Institute of Automation,
Chinese Academy of Sciences, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
{zhuoyu.wei, jzhao, kliu}@nlpr.ia.ac.cn

Abstract

Deep inference on a large-scale knowledge base (KB) needs a mass of formulas, but it is almost impossible to create all formulas manually. Data-driven methods have been proposed to mine formulas from KBs automatically, where random sampling and approximate calculation are common techniques to handle big data. Among a series of methods, Random Walk is believed to be suitable for knowledge graph data. However, a pure random walk without goals still has a poor efficiency of mining useful formulas, and even introduces lots of noise which may mislead inference. Although several heuristic rules have been proposed to direct random walks, they do not work well due to the diversity of formulas. To this end, we propose a novel goal-directed inference formula mining algorithm, which directs random walks by the specific inference target at each step. The algorithm is more inclined to visit benefic structures to infer the target, so it can increase efficiency of random walks and avoid noise simultaneously. The experiments on both WordNet and Freebase prove that our approach is has a high efficiency and performs best on the task.

1 Introduction

Recently, various knowledge bases (KBs), such as Freebase (Bollacker et al., 2008), WordNet (Miller, 1995), Yago (Hoffart et al., 2013), have been built, and researchers begin to explore how to make use of structural information to promote performances of several inference-based NLP applications, such as text entailment, knowledge base completion, question and answering. Creating useful formulas is one of the most important steps in inference, and an accurate and high coverage formula set will bring a great promotion for an inference system. For example, *Nationality(x, y)* $\wedge$ *Nationality(z, y)* $\wedge$ *Language(z, w)* $\Rightarrow$ *Language(x, w)* is a high-quality formula, which means people with the same nationality probably speak the same language. However, it is a challenge to create formulas for open-domain KBs, where there are a great variety of relation types and it is impossible to construct a complete formula set by hand.

Several data-driven methods, such as Inductive Logic Programming (ILP) (Muggleton and De Raedt, 1994) and Markov Logic Network (MLN) (Richardson and Domingos, 2006), have been proposed to mine formulas automatically from KB data, which transform frequent sub-structures of KBs, e.g., paths or loops, into formulas. Figure 1.a shows a sub-graph extracted from Freebase, and the formula mentioned above about *Language* can be generated from the loop in Figure 1.d. However, the running time of these traditional probabilistic inference methods is unbearable over large-scale KBs. For example, MLN needs grounding for each candidate formula, i.e., it needs to enumerate all paths. Therefore, the computation complexity of MLN increases exponentially with the scale of a KB.

In order to handle large-scale KBs, the random walk is usually employed to replace enumerating all possible sub-structures. However, random walk is inefficient to find useful structures due to its completely randomized mechanism. For example in Fig-

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1379–1388,
Austin, Texas, November 1-5, 2016.

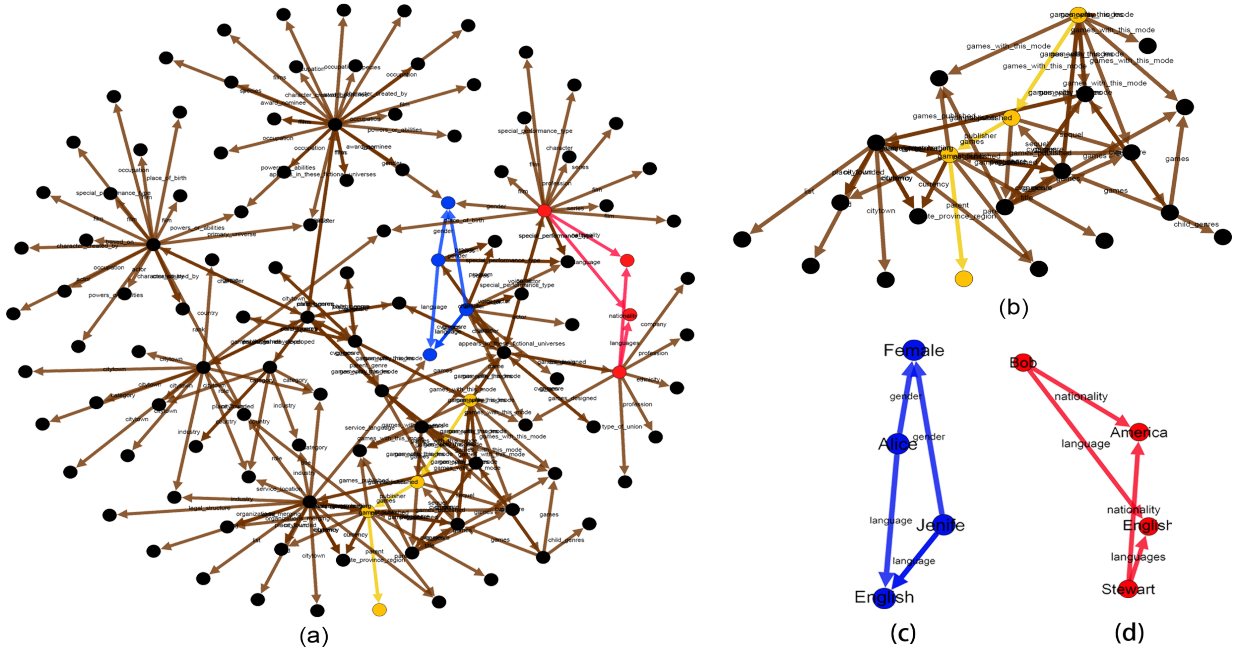

Figure 1: a) shows a subgraph extracted from Freebase. b) shows the searching space of finding the yellow path. c) shows a loop which can generate a false formula. d) shows a loop which can generate a true formula.

ure 1.b, the target path (yellow one) has a small probability to be visited, the reason is that the algorithm may select all the neighboring entity to transfer with an equal probability. This phenomenon is very common in KBs, e.g., each entity in Freebase has more than 30 neighbors in average, so there will be about 810,000 paths with length 4, and only several are useful. There have been two types of methods proposed to improve the efficiency of random walks, but they still meet serious problems, respectively.

1) **Increasing rounds of random walks.** More rounds of random walks will find more structures, but it will simultaneously introduce more noise and thus generate more false formulas. For example, the loop in Figure 1.c exists in Freebase, but it produces a false formula, *Gender(x, y)* $\wedge$ *Gender(z, y)* $\wedge$ *Language(z, w)* $\Rightarrow$ *Language(x, w)*, which means people with the same gender speak the same language. This kind of structures frequently occur in KBs even the formulas are mined with a high confidence, because there are a lot of sparse structures in KBs which will lead to inaccurate confidence. According to our statistics, more than 90 percent of high-confidence formulas produced by random walk are noise.

2) **Employing heuristic rules to direct random walks.** This method directs random walks to find useful structures by rewriting the state transition probability matrix, but the artificial heuristic rules may only apply to a little part of formulas. For example, PRA (Lao and Cohen, 2010; Lao et al., 2011) assumes the more narrow distributions of elements in a formula are, the higher score the formula will obtain. However, formulas with high scores in PRA are not always true. For example, the formula in Figure 1.c has a high score in PRA, but it is not true. Oppositely, formulas with low scores in PRA are not always useless. For example, the formula, $Father(x, y) \wedge Father(y, z) \Rightarrow Grandfather(x, t)$, has a low score when x and y both have several sons, but it obviously is the most effective to infer $Grandfather$. According to our investigations, the situations are common in KBs.

In this paper, we propose a Goal-directed Random Walk algorithm to resolve the above problems. The algorithm employs the specific inference target as the direction at each step in the random walk process. In more detail, to achieve such a goal-directed mechanism, at each step of random walk, the algorithm dynamically estimates the potentials for each neighbor by using the ultimate goal, and assigns higher probabilities to the neighbors with higher potentials. Therefore, the algorithm is more inclined to visit structures which are beneficial to infer

the target and avoid transferring to noise structures. For example in Figure 1, when the inference target is *what language a person speaks*, the algorithm is more inclined to walk along *Nationality* edge than *Gender*, because *Nationality* has greater potential than *Gender* to infer *Language*. We build a real potential function based on low-rank distributional representations. The reason of replacing symbols by distributional representations is that the distributional representations have less parameters and latent semantic relationship in them can contribute to estimate potentials more precisely. In summary, the contributions of this paper are as follows.

• Compared with the basic random walk, our approach direct random walks by the inference target, which increases efficiency of mining useful formulas and has a great capability of resisting noise.

• Compared with the heuristic methods, our approach can learn the strategy of random walk automatically and dynamically adjust the strategy for different inference targets, while the heuristic methods need to write heuristic rules by hand and follow the same rule all the time.

• The experiments on link prediction task prove that our approach has a high efficiency on mining formulas and has a good performance on both WN18 and FB15K datasets.

The rest of this paper is structured as follows, Section 2 introduces the basic random walk for mining formulas. Section 3 describes our approach in detail. The experimental results and related discussions are shown in Section 4. Section 5 introduces some related works, and finally, Section 6 concludes this paper.

2 Mining Formulas by Random Walk

2.1 Frequent Pattern Mining

Mining frequent patterns from source data is a problem that has a long history, and for different specific tasks, there are different types of source data and different definitions of pattern. Mining formulas is more like frequent subgraph mining, which employs paths or loops as frequent patterns and mines them from a KB. For each relation type R, the algorithm enumerates paths from entity H to entity T for each triplet $R(H,T)$. These paths are normalized to formulas by replacing entities to variables. For example, the loop in Figure 1.d, *Nationality(Bob, America)* $\wedge$ *Nationality(Stewart, America)* $\wedge$ *Language(Bob, English)* $\Rightarrow$ *Language(Stewart, English)*, can be normalized to the formula, *Nationality(x, y)* $\wedge$ *Nationality(z, y)* $\wedge$ *Language(z, w)* $\Rightarrow$ *Language(x, w)*. Support and confidence are employed to estimate a formula, where the support value of a formula $f : X \Rightarrow Y$, noted as S_f, is defined as the proportion of paths in the KB which contains the body X, and the confidence value of $X \Rightarrow Y$, noted as C_f, is defined as the proportion of the paths that contains X which also meets $X \Rightarrow Y$. C_f is calculated as follows,

$$C_f = \frac{N_f}{N_X} \tag{1}$$

where N_f is the total number of instantiated formula f and N_X is the total number of instantiated X.

2.2 Random Walk on Knowledge Graph

Enumerating paths is a time consuming process and does not apply to large-scale KBs. Therefore, random walk on the graph is proposed to collect frequent paths instead of enumerating. Random walk randomly chooses a neighbor to jump unlike enumerating which needs to search all neighbors. To estimate a formula f, the algorithm employs f's occurrence number during random walks $N_f^{'}$ to approximate the total number N_f in Equation (1), and similarly employs $N_X^{'}$ to approximate N_X. Therefore, f's confidence C_f can be approximatively estimated by $N_f^{'}$ and $N_X^{'}$, noted as $C_f^{'}$.

Random walk maintains a state transition probability matrix P, and P_{ij} means the probability of jumping from entity i to entity j. To make the confidence $C_f^{'}$ as close to the true confidence C_f as possible, the algorithm sets P as follows,

$$P_{ij} = \begin{cases} 1/d_i, & j \in Adj(i) \\ 0, & j \notin Adj(i) \end{cases} \tag{2}$$

where d_i is the out-degree of the entity i, $Adj(i)$ is the set of adjacent entities of i, and $\sum_{j=1}^{N} P_{ij} = 1$. Such a transition matrix means the algorithm may jump to all the neighboring entities with an equal probability. Such a random walk is independent from the inference target, so we call this type of random walk as a goalless random walk. The goalless mechanism causes the inefficiency of mining useful structures. When we want to mine paths for $R(H,T)$, the algorithm cannot arrive at T from H

in the majority of rounds. Even though the algorithm recalls several paths for $R(H,T)$, some of them may generate noisy formulas for inferring $R(H,T)$.

To solve the above problem, several methods direct random walks by statically modifying P. For example, PRA sets $P_{r_{ij}} = \frac{P(j|i;r)}{|R_i|}$, $P(j|i;r) = \frac{r(i,j)}{r(i,*)}$, where $P(j|i;r)$ is the probability of reaching node j from node i under the specific relation r, $r(i,*)$ is the number of edges from i under r, and R_i is the number of relation types from i. Such a transition matrix implies the more narrow distributions of elements in a formula are, the higher score the formula will obtain, which can be viewed as the heuristic rule of PRA.

3 Our Approach

3.1 Goal-Directed Random Walk

We propose to use the inference target, $\rho = R(H,T)$, to direct random walks. When predicting ρ, our approach always directs random walks to find useful structures which may generate formulas to infer ρ. For different ρ, random walks are directed by modifying the transition matrix P in different ways. Our approach dynamically calculates $P_{r_{ij}}$ when jumping from entity i to entity j under relation r as follows,

$$P_{r_{ij}} = \begin{cases} \dfrac{\Phi(r(i,j),\rho)}{\sum_{k\in Adj(i)}\Phi(r(i,k),\rho)}, & j \in Adj(i) \\ 0, & j \notin Adj(i) \end{cases} \tag{3}$$

where $\Phi(r(i,j),\rho)$ is the $r(i,j)$'s potential which measures the potential contribution to infer ρ after walking to j.

Intuitively, if $r(i,j)$ exits in a path from H to T and this path can generate a benefic formula to infer $R(H,T)$, the probability of jumping from i to j should larger and thus $\Phi(r(i,j),\rho)$ also should be larger. Reversely, if we cannot arrive at T within the maximal steps after jumping to j, or if the path produces a noisy formula leading to a wrong inference, P_{ij} and $\Phi(r(i,j),\rho)$ should both be smaller.

To explicitly build a bridge between the potential Φ and the inference goal ρ, we maximize the likelihood of paths which can infer ρ. First, we recursively define the likelihood of a path from H to t as $P_{p_{Ht}} = P_{p_{Hs}} \cdot P_{r_{st}}$, where $P_{r_{st}}$ is defined in Equation (3). We then classify a path p_{Ht} into three separate categories: a) $t = T$ and p_{Ht} can produce a benefic formula to infer $R(H,T)$; b) $t \neq T$; c) $t = T$ but p_{Ht} may generate a noisy formula which misleads inference. Finally, we define the likelihood function as follows,

$$\max P_{\mathbb{P}} = \prod_{p_{Ht}\in\mathbb{P}} P_{p_{Ht}}^{a}(1-P_{p_{Ht}})^{b+c} \tag{4}$$

where $\mathbb{P}$ is all paths found in the process of performing random walks for $R(H,T)$, and t may be equal to T or not. a, b, c are three 0-1 variables corresponding to the above categories a), b), c). Only one in a, b, c can be 1 when P_{Ht} belongs to the corresponding category. We then transform maximizing $P_{\mathbb{P}}$ to minimizing $L_{rw} = -\log P_{\mathbb{P}}$ and employ SGD to train it. In practice, there is not a clear-cut boundary between a) and c), so we divide the loss into two parts: $L_{rw} = L_{rw}^{t} + \lambda L_{rw}^{inf}$. L_{rw}^{t} is the loss of that $t \neq T$, and L_{rw}^{inf} is the loss of that p_{HT} generates a noisy formula leading to a wrong inference. λ is a super parameter to balance the two losses. L_{rw}^{t} and L_{rw}^{inf} have the same expression but are optimized in different stages. L_{rw}^{t} can be optimized during random walks, while L_{rw}^{inf} should be optimized in the inference stage. We rewrite L_{rw} for a specific path p as follows,

$$L_{rw}(p) = -y\log P_p - (1-y)\log(1-P_p) \tag{5}$$

where y is the label of the path p and $y = 1$ if p is beneficial to infer ρ. To obtain the best Φ, we compute gradients of L_{rw} as follows,

$$\begin{aligned} &\nabla L_{rw}(p) = (\nabla L_{rw}(r_{12}), \nabla L_{rw}(r_{23}), ...) \\ &\nabla L_{rw}(r_{ij}) = (\frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ij}}}, \frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ik_1}}}, \frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ik_2}}}, ...) \\ &\frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ij}}} = \frac{(P_p - y)\cdot(1-P_{r_{ij}})}{\Phi_{r_{ij}}\cdot(1-P_p)} \\ &\frac{\partial L_{rw}(r_{ij})}{\partial \Phi_{r_{ik}}} = -\frac{(P_p - y)\cdot P_{r_{ij}}}{\Phi_{r_{ij}}\cdot(1-P_p)} \end{aligned} \tag{6}$$

where $\nabla L_{rw}(r_{ij})$ is the component of $\nabla L_{rw}(p)$ at r_{ij}. $\Phi(r(i,j),\rho)$ and $\Phi(r(i,k),\rho)$ are the potentials for all triplets $r(i,j) \in p$ and $r(i,k) \notin p$, and r_{ij} is short for $r(i,j)$. After iteratively updating $\Phi_{r_{ij}}$ and $\Phi_{r_{ik}}$ by the gradient of L_{rw}^{t}, the random walks can

be directed to find more paths from H to T, and consequently it increases efficiency of the random walk. After updating $\Phi_{r_{ij}}$ and $\Phi_{r_{ik}}$ by the gradient of L_{rw}^{inf}, random walk is more likely to find high-quality paths but not noise. Therefore, the goal-directed random walk increases efficiency of mining benefic formulas and has a great capability of resisting noise.

3.2 Distributional Potential Function

The potential $\Phi(r(i,j),\rho)$ measures an implicit relationship between two triplets in the KB, so the total number of parameters is the square of the KB size. It is hard to precisely estimate all Φ because of the sparsity of training data. To reduce the number of parameters, we represent each entity or relation in the KB as a low-rank numeric vector which is called embeddings (Bordes et al., 2013), and then we build a potential function Ψ on embeddings as $\Phi(r(i,j),\rho) = \Psi(E_{r(i,j)}, E_{R(H,T)})$, where $E_{r(i,j)}$ and $E_{R(H,T)}$ are the embeddings of triplets. In practice, we set $E_{r(i,j)} = [E_r, E_j]$ and $E_{R(H,T)} = [E_R, E_T]$ because E_i is the same for all triplets $r(i,*)$, where [] is a concatenation operator.

In the view of the neural network, our goal-directed mechanism is analogous to the attention mechanism. At each step, the algorithm estimates attentions for each neighboring edges by Ψ. Therefore, there are several existing expressions of Ψ, e.g., the dot product (Sukhbaatar et al., 2015) and the single-layer perceptron (Bahdanau et al., 2015). We will not compare different forms of Ψ, the detail comparison has been presented in the work (Luong et al., 2015). We directly employ the simplest dot product for Ψ as follows,

$$\Psi(E_{r(i,j)}, E_{R(H,T)}) = \sigma(E_{r(i,j)} \cdot E_{R(H,T)}) \quad (7)$$

where σ is a nonlinear function and we set it as an exponential function. Ψ has no parameters beside KB embeddings which are updated during the training period.

3.3 Integrated Inference Model

To handle the dependence between goal-directed random walk and subsequent inference, we combine them into an integrated model and optimize them together. To predict $\rho = R(H,T)$, the integrated model first collects formulas for $R(H,T)$, and then merges estimations of different formulas as features into a score function χ,

$$\chi(\rho) = \sum_{f \in F_\rho} \delta(f) \quad (8)$$

where F_ρ is the formula set obtained by random walks for ρ, and $\delta(f)$ is an estimation of formula f. The original frequent pattern mining algorithm employs formulas' confidence as $\delta(f)$ directly, but it does not produce good results (Galárraga et al., 2013). There are two ways to solve the problem: one is selecting another more suitable measure of f as $\delta(f)$ (Tan et al., 2002); the other is attaching a weight to each formula and learning weights with supervision, e.g., MLN (Richardson and Domingos, 2006) . We employ the latter method and set $\delta(f) = w_f \cdot n_f$. Finally, we employ a logistic regression classifier to predict $R(H,T)$, and the posterior probability of $R(H,T)$ is shown as follows,

$$\begin{aligned} P(\rho = y|\chi) &= \mathcal{F}(\chi)^y (1 - \mathcal{F}(\chi))^{1-y} \\ \mathcal{F}(\chi) &= \frac{1}{1+e^{-\chi}} \end{aligned} \quad (9)$$

where y is a 0-1 label of ρ. Similar to L_{rw}^t in Equation (5), we treat the negative logarithm of $P(\rho = y|\chi)$ as the loss of inference, $L_{inf} = -\log P(\rho = y|\chi)$, and turn to minimize it. Moreover, the loss L_{rw}^{inf} of the above goal-directed random walk is influenced by the result of predicting $R(H,T)$, so $\Phi_{r_{ij}}$ and $\Phi_{r_{ik}}$ will be also updated. Algorithm 1 shows the main process of training, where Ξ is the triplet set for training, Ψ is the potential function in Equation (7), F is the formula set, f_p is

Algorithm 1: Train Integrated Inference Model

Input: KB, Ξ
Output: Ψ, W, F
For $\rho = R(H,T) \in \Xi$
  Repeat ρ-directed Random Walk from H to t
    Update Ψ by L_{rw}^t
    If $t = T$, then $F = F \cap f_p$
  Calculate L_{inf} and L_{rw}^{inf} by ρ
  Update W by L_{inf}
  Update Ψ by L_{rw}^{inf}
Remove $f \in F$ with little w_f
Output Ψ, W, F

Dataset	Relation	Entity	Train	Valid	Test
WN18	18	40,943	141,442	5,000	5,000
FB15K	1,345	14,951	483,142	50,000	59,071

Table 1: Statistics of WN18 and FB15K

a formula generated from the path p, and H, T, t are entities in the KB. To predict $\rho = R(H, T)$, the algorithm first performs multi rounds of random walks, and each random walk can find a path p_{Ht} (at line 2). Then the algorithm decides to update Ψ by L^t_{rw} based on whether t is T (at line 3), and adds the formula p_f into the formula set when $t = T$ (at line 4). After random walks, the inference model predicts ρ, and computes L_{inf} and L^{inf}_{rw} according to the prediction result (at line 5). Finally W and Ψ are updated by L_{inf} and L^{inf}_{rw} (at line 6-7), respectively. After training by all triplets in Ξ, the algorithm removes formulas with low weights from F (at line 8) and outputs the model (at line 9). When we infer a new triplet by this model, the process is similar to Algorithm 1.

4 Experiments

We first compare our approach with several state-of-art methods on link prediction task to explore our approach's overall ability of inference. Subsequently, we evaluate formulas mined by different random walk methods to explore whether the goal-directed mechanism can increase efficiency of mining useful structures. Finally, we dive deep into the formulas generated by our approach to analyze the characters of our approach.

4.1 Datasets and Evaluation Setup

We conduct experiments on both WN18 and FB15K datasets which are subsets sampled from WordNet (Miller, 1995) and Freebase (Bollacker et al., 2008), respectively, and Table 1 shows the statistics of them. For the link prediction task, we predict the missing h or t for a triplet $r(h, t)$ in test set. The detail evaluation method is that t in $r(h, t)$ is replaced by all entities in the KB and methods need to rank the right answer at the top of the list, and so does h in $r(h, t)$. We report the mean of those true answer ranks and the Hits@10 under both 'raw' and 'filter' as TransE (Bordes et al., 2013) does, where Hits@10 is the proportion of correct entities ranked in the top 10.

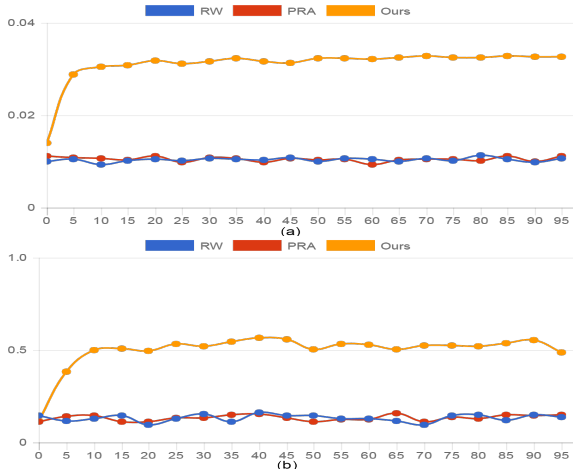

Figure 2: $Arr@10$ of three random walk algorithms and the horizontal axis represents epochs and the vertical axis represents $Arr@10$. Figure 2.a shows results on relation *_derivationally_related_form* in WN18, and Figure 2.b shows results on relation *form_of_government* in FB15K.

4.2 Baselines

We employ two types of baselines. One type is based on random walks including: a) the basic random walk algorithm whose state transition probability matrix is shown in Equation (2); b) PRA in (Lao et al., 2011) which is a typical heuristic random walk algorithm. The other type is based on KB embeddings including TransE (Bordes et al., 2013), Rescal (Nickel et al., 2011), TransH (Wang et al., 2014b), TransR (Lin et al., 2015b). These embedding-based methods have no explicit formulas, so we will not evaluate their performances on mining formulas.

4.3 Settings

We implement three random walk methods under a unified framework. To predict $r(h, *)$ quickly, we first select Top-K candidate instances, $t_{1 \to K}$, by TransE as (Wei et al., 2015), and then the algorithm infers each $r(h, t_i)$ and ranks them by inference results. We adjust parameters for our approach with the validate dataset, and the optimal configurations are set as follows. The rounds of random walk is 10, learning rate is 0.0001, training epoch is 100, the size of candidate set is 500 for WN18 and 100 for FB15K, the embeddings have 50 dimensionalities for WN18 and 100 dimensionalities for FB15K, and the embeddings are initialized by TransE. For some relations, random walk truly finds no practicable formulas, so we employ TransE to improve per-

	Dataset	WN18				FB15K			
	Metric	Mean Rank		Hits@10(%)		Mean Rank		Hits@10(%)	
		Raw	Filt	Raw	Filt	Raw	Filt	Raw	Filt
2.a	RESCAL	1,180	1,163	37.2	52.8	828	683	28.4	44.1
	TransE	263	251	75.4	89.2	243	125	34.9	47.1
	TransH	401	388	73.0	82.3	212	87	45.7	64.4
	TransR	238	225	79.8	92.0	198	77	48.2	68.7
2.b	RW	28*	17*	84.40	94.89	37*	28*	37.04	51.13
	PRA	28*	17*	**84.43**	**94.90**	37*	29*	36.72	50.73
2.d	Our approach	**28***	**17***	84.40	94.86	**34***	**25***	**53.47**	**74.75**

Table 2: Link Prediction Results on both WN18 and FB15K

formance for these relations. For embedding-based methods, we use reported results directly since the evaluation datasets are identical.

4.4 Results on Link Prediction

We show the results of link prediction for our approach and all baselines in Table 2 (* means the mean of ranks for random walk methods are evaluated in the Top-K subset), and we can obtain the following observations:

1) Our approach achieves good performances on both WN18 and FB15K. On the FB15K, our approach outperforms all baselines. It indicates that our approach is effective for inference. On the WN18, three random walk methods have similar performances. The reason is that most entities in WN18 only have a small number of neighbors, so RW and PRA can also find useful structures in a few rounds.

2) For FB15K, the performances of RW and PRA are both poor and even worse than a part of embedding-based methods, but the performance of our approach is still the best. The reason is that there are too many relation types in FB15K, so goalless random walks introduce lots of noise. Oppositely, our approach has a great capability of resisting noise for the goal-directed mechanism.

3) RW and PRA have similar performances on both datasets, which indicates the heuristic rule of PRA does not apply to all relations and formulas.

4.5 Paths Recall by Random Walks

To further explore whether the goal-directed mechanism can increase efficiency of mining paths, we compare the three random walk methods by the number of paths mined. For each triplet $R(H, T)$ in the training set, we perform 10 rounds of random walks from H and record the number of times which arrive at T, noted as Arr@10. We respectively select one relation type from WN18 and FB15K and show the comparison result in Figure 2. We can obtain the following observations:

1) With the increase of training epochs, $Arr@10$ of the goal-directed random walk first increases and then stays around a high value on both WN18 and FB15K, but the $Arr@10$ of RW and PRA always stay the same. This phenomenon indicates that the goal-directed random walk is a learnable model and can be trained to find more useful structures with epochs increasing, but RW and PRA are not.

2) RW and PRA always have similar $Arr@10$, which means PRA has not found more formulas. This indicates that the heuristic rule of PRA is not always be beneficial to mining more structures for all relations.

4.6 Example Formulas

In Table 3, we show a small number of formulas mined by our approach from FB15K, and the formulas represent different types. Some formulas contain clear logic, e.g, Formula 1 means that if the writer x contributes a story to the film y and y is adapted from the book z, x is the writer of the book z. Some formulas have a high probability of being satisfied, e.g., Formula 3 means the wedding place probably is also the burial place for some people, and Formula 7 means the parent of the person x died of the disease and thus the person x has a high risk of suffering from the disease. Some formulas depend on synonyms, e.g., *story_by* and *works_written* have the similar meaning in Formula 2. However, there are still useless formulas, e.g, Formula 8 is useless be-

Relation	Formula
works_written	
1	film_story_contributor(x,y) ∧ adapted_from(y,z) ⇒ works_written(x,z)
2	story_by(y,x) ⇒ works_written(x,y)
place_of_burial	
3	place_of_death(x,y) ⇒ place_of_burial(x,y)
4	marriage_type_of_union(x,y) ∧ marriage_location_of_ceremony(y,z) ⇒ place_of_burial(x,z)
service_language	
5	service_location(x,y) ∧ imported_from(y,z) ∧ official_language(z,w) ⇒ service_language(x,w)
6	service_location(x,y) ∧ exported_to(y,z) ∧ languages_spoken(z,w) ⇒ service_language(x,w)
disease_risk_factors	
7	parent_cause_of_death(x,y) ∧ disease_risk_factors(y,z) ⇒ disease_risk_factors(x,z)
8	disease_risk_factors(x,y)∧ -disease_risk_factors(y,x) ⇒ disease_risk_factors(x,y)

Table 3: Example Formulas Obtained by Goal-directed Random Walk

cause the body of the formula is same as the head. Such useless formula can be removed by a super-rule, which is that the head of a formula cannot occur in its body.

5 Related Work

Our work has two aspects, which are related to mining formula automatically and inference on KBs, respectively.

Inductive Logic Programming (ILP) (Muggleton and De Raedt, 1994) and Association Rule Mining (ARM) (Agrawal et al., 1993) are both early works on mining formulas. FOIT (Quinlan, 1990) and SHERLOCK (Schoenmackers et al., 2010) are typical ILP systems, but the former one usually need a lot of negative facts and the latter one focuses on mining formulas from text. AMIE (Galárraga et al., 2013) is based on ARM and proposes a new measure for formulas instead of the confidence. Several structure learning algorithms (Kok and Domingos, 2005; Kok and Domingos, 2009; Kok and Domingos, 2010) based on Markov Logic Network (MLN) (Richardson and Domingos, 2006) can also learn first order logic formulas automatically, but they are too slow to run on large KBs. ProPPR (Wang et al., 2013; Wang et al., 2014a) performs structure learning by depth first searching on the knowledge graph, which is still not efficient enough to handle web-scale KBs. PRA (Lao and Cohen, 2010; Lao et al., 2011) is a method based on random walks and employs heuristic rules to direct random walks. PRA is closely related to our approach, but unlike it, our approach dynamically calculates state transition probabilities. Another method based on random walks (Wei et al., 2015) merges embedding similarities of candidates into the random walk as a priori, while our approach employs KB embeddings to calculate potentials for neighbors.

The majority of mining formula methods can perform inference on KBs, and besides them, a dozen methods based KB embeddings can also achieve the inference goal, and the typical ones of them are TransE (Bordes et al., 2013), Rescal (Nickel et al., 2011), TransH (Wang et al., 2014b), TransR (Lin et al., 2015b). These embedding-based methods take advantage of the implicit relationship between elements of the KB and perform inference by calculating similarities. There are also methods which combine inference formulas and KB embeddings, such as PTransE (Lin et al., 2015a) and ProPPR+MF (Wang and Cohen, 2016).

6 Conclusion and Future Works

In this paper, we introduce a goal-directed random walk algorithm to increase efficiency of mining useful formulas and decrease noise simultaneously. The approach employs the inference target as the direction at each steps in the random walk process and is more inclined to visit structures helpful to inference. In empirical studies, we show our approach achieves good performances on link prediction task over large-scale KBs. In the future, we are interested in exploring mining formulas directly in the distributional spaces which may resolve the sparsity of formulas.

7 Acknowledgments

This work was supported by the Natural Science Foundation of China (No. 61533018), the National Basic Research Program of China (No. 2014CB340503) and the National Natural Science Foundation of China (No. 61272332). And this work was also supported by Google through focused research awards program.

References

Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. ACM.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *Advances in Neural Information Processing Systems*, pages 2787–2795.

Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web*, pages 413–422. International World Wide Web Conferences Steering Committee.

Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, and Gerhard Weikum. 2013. Yago2: A spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 194:28–61.

Stanley Kok and Pedro Domingos. 2005. Learning the structure of markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pages 441–448. ACM.

Stanley Kok and Pedro Domingos. 2009. Learning markov logic network structure via hypergraph lifting. In *Proceedings of the 26th annual international conference on machine learning*, pages 505–512. ACM.

Stanley Kok and Pedro Domingos. 2010. Learning markov logic networks using structural motifs. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 551–558.

Ni Lao and William W Cohen. 2010. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, 81(1):53–67.

Ni Lao, Tom Mitchell, and William W Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 529–539. Association for Computational Linguistics.

Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2015a. Modeling relation paths for representation learning of knowledge bases. *arXiv preprint arXiv:1506.00379*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, pages 2181–2187.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Conference on Empirical Methods in Natural Language Processing*.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Stephen Muggleton and Luc De Raedt. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 809–816.

J. Ross Quinlan. 1990. Learning logical definitions from relations. *Machine learning*, 5(3):239–266.

Matthew Richardson and Pedro Domingos. 2006. Markov logic networks. *Machine learning*, 62(1-2):107–136.

Stefan Schoenmackers, Oren Etzioni, Daniel S Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1088–1098. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, pages 2431–2439.

Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. 2002. Selecting the right interestingness measure for association patterns. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 32–41. ACM.

William Yang Wang and William W Cohen. 2016. Learning first-order logic embeddings via matrix factorization. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016)*.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2013. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2129–2138. ACM.

William Yang Wang, Kathryn Mazaitis, and William W Cohen. 2014a. Structure learning via parameter learning. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1199–1208. ACM.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014b. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, pages 1112–1119. Citeseer.

Zhuoyu Wei, Jun Zhao, Kang Liu, Zhenyu Qi, Zhengya Sun, and Guanhua Tian. 2015. Large-scale knowledge base completion: Inferring via grounding network sampling over selected instances. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1331–1340. ACM.