Large-scale Knowledge Base Completion: Inferring via Grounding Network Sampling over Selected Instances

Zhuoyu Wei[1,2], Jun Zhao[1], Kang Liu[1], Zhenyu Qi[2], Zhengya Sun[1] and Guanhua Tian[2]
[1] National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences
[2] Interactive Digital Media Technology Research, Institute of Automation, Chinese Academy of Sciences
{zhuoyu.wei, zhenyu.qi, zhengya.sun, guanhua.tian}@ia.ac.cn, {jzhao, kliu}@nlpr.ia.ac.cn

ABSTRACT

Constructing large-scale knowledge bases has attracted much attention in recent years, for which Knowledge Base Completion (KBC) is a key technique. In general, inferring new facts in a large-scale knowledge base is not a trivial task. The large number of inferred candidate facts has resulted in the failure of the majority of previous approaches. Inference approaches can achieve high precision for formulas that are accurate, but they are required to infer candidate instances one by one, and extremely large candidate sets bog them down in expensive calculations. In contrast, the existing embedding-based methods can easily calculate similarity-based scores for each candidate instance as opposed to using inference, so they can handle large-scale data. However, this type of method does not consider explicit logical semantics and usually has unsatisfactory precision. To resolve the limitations of the above two types of methods, we propose an approach through Inferring via Grounding Network Sampling over Selected Instances. We first employ an embedding-based model to make the instance selection and generate much smaller candidate sets for subsequent fact inference, which not only narrows the candidate sets but also filters out part of the noise instances. Then, we only make inferences within these candidate sets by running a data-driven inference algorithm on the Markov Logic Network (MLN), which is called Inferring via Grounding Network Sampling (INS). In this process, we especially incorporate the similarity priori generated by embedding-based models into INS to promote the inference precision. The experimental results show that our approach improved Hits@1 from 32.911% to 71.692% on the FB15K dataset and achieved much better AP@n evaluations than state-of-the-art methods.

Categories and Subject Descriptors

E.1 [**Data**]: DATA STRUCTURES—*Graphs and networks*

Keywords

Knowledge Base Completion; Embedding; Inference

CIKM'15 October 19-23, 2015, Melbourne, VIC, Australia

DOI: http://dx.doi.org/10.1145/2806416.2806513.

1. INTRODUCTION

Automatically extracting facts from texts and constructing large-scale knowledge bases (KB) have grown vigorously in recent years. As a result, several typical knowledge bases have been built, such as Freebase [2], Nell [6], YAGO [10], and Knowledge Vault [7]. However, these extracted repositories are far from completion. To complete the constructed KBs, according to the conclusion in [7], using an existing knowledge base to complete itself is an important supplement for automatic knowledge extraction to increase the number of facts in KBs and cannot be substituted by other techniques. Therefore, this paper focuses on the large-scale knowledge base completion (KBC) and is committed to predicting the missing links in the existing knowledge base.

In general, according to the process of KBC, there are two types of approaches: inference-based approaches and embedding-based approaches.

First, inference-based approaches [22, 16, 24] usually employ logic formulas to infer the missing links among existing entities in a KB. They manually or automatically construct various logic formulas and learn the weight of each formula by sampling or counting groundings from existing KBs. These weighted formulas are viewed as the long-range interactions across several relations. The biggest limitation of such approaches is the computation complexity. These methods need to infer knowledge one by one, which implies the computation complexity is linearly growing with the size of candidate sets. However, usually, there are extremely large candidate sets for some specific relations in large-scale KBC, and in each candidate set, only one or a few are actually correct. For example, *Barack Obama's mother* is missing in a KB, and we need to find out who *Barack Obama's mother* is. All persons or females in the KB are candidates, but only one is the correct selection. The huge candidate set brings inference-based approaches to an unacceptable running time. Although some methods have avoided this issue through simple operations, such as only adding a small part of false facts to testing sets [13, 23], this strategy is too coarse to obtain precise inference results. On the other hand, there are some noise candidates, which may violate formulas and mislead inference algorithms. Therefore, existing inference methods that rely on formulas cannot remove the noises by themselves, and the noises may result in the decrease of the performance.

In contrast, embedding-based methods [21, 5, 12, 4, 3, 26, 18] are not affected by huge candidate sets because they can easily calculate similarity-based scores for each candidate instance after learning representations of entities and rela-

Table 1: Several embedding approach results on FB15K

Embedding Model	Hits@1(%)	Hits@10(%)
Uns[4]	0.384	15.573
SE[5]	28.633	61.026
SME-lin[3]	29.807	68.386
SME-bil[3]	32.911	68.506
TransE[4]	29.401	73.71

tions. Unfortunately, embedding-based approaches do not consider explicit logical semantics and cannot capture the interaction between different relations well enough. Most of them simply model the direct interaction between relations by entity embeddings, apart from the long-range interaction across several relations. Thus, their results are universally unsatisfactory. Table 1 shows several typical embedding models' Hits@1 and Hits@10 evaluations[1] on the FB15K dataset [4]. Although many embedding-based methods could obtain high ratings in Top N results (N>1), they usually have low performance in Top 1 results (the more useful metric than Top N in KBC).

To resolve the limitations of the above two types of methods, we propose a novel approach by inferring via grounding network sampling over selected instances. We first employ an embedding-based model to perform instances selection. In specific, we employ TransE [4] to learn the representations of entities and relations in the KB. We calculate similarity scores between candidates and the input query, and we select Top-N instances to constitute a new smaller candidate set for subsequent fact inference. In this way, we not only filter out a part of the noise but also improve the efficiency of the inference algorithm. Meanwhile, the corresponding similarity scores are recorded and viewed as the prior to supervise the subsequent inference. We perform logical inference over selected instances by exploiting a data-driven inference algorithm on a Markov Logic Network (MLN), which is called Inferring via Grounding Network Sampling (**INS**). INS could consider the long-range interaction across several relations that were ignored in existing embedding-based methods. Moreover, we improve INS and propose a **INS-ES** algorithm (Inferring via ground Network Sampling Merging Embedding Similarity Priori), which could consider the probability of the transition between states when performing network sampling for inference. In this way, the recorded similarity scores can be incorporated naturally into our model and the inference process can be supervised in some way, which is beneficial for improving the inference precision. In general, our method is an comprehensive combination of two types of methods for knowledge inference, and it could not only avoid the disadvantage of inference-based methods (cannot handle large-scale knowledge bases) but also could improve the weakness of embedding-based methods (cannot obtain explicit logical semantics and cannot sufficiently capture the interaction between different relations). Experiments show that our approach has significant improvement compared with state-of-the-art methods, increasing Hits@1 from 32.911% to 71.692% on the FB15K.

The main contributions of this work are summarized as follows.

- We propose employing embedding-based methods to select instances and generate much smaller candidate sets for knowledge inference, which drastically reduces the running time of inferring on large-scale KBs.
- We employ MLN to model the large-scale KB and propose an inferring algorithm **INS** to infer overselected instances. In this way, the explicit semantics and long-range interaction across different relations could be captured.
- We specially incorporate the similarity scores generated by the embedding-based model into INS-ES to promote the inference precision.

[1]Hits@n means the proportion of correct entities ranked in the top N

2. PRELIMINARIES

Knowledge Base is a structural information storage system, e.g., Freebase, that usually contains entities, relations, and facts. A KB can be viewed as a directed graph $G(V, E, R)$. The vertex set V contains all entities in KB, and the edge set E contains all facts. R is the set of relations that can be viewed as edge labels. We only consider binary relation in this paper. $r(h, t)$ is an example of a fact, and there is an edge from h to t with label r.

Knowledge Base Completion is the prediction of additional true facts using only an existing database[23]. Given a KB, noted as $G(V, E, R)$, our goal is to predict a new fact set $E^{'}$ with high precision and recall, and V and R remain unchanged. For example, 93.8% of persons included in Freebase have no place of birth, and 78.5% of them have no nationality[20]; KBC is committed to predicting them.

Embedding means representing each entity in KB as a low-dimension numeric vector, and different dimensions of the vector may implicitly represent different aspects of an entity. Relations in KB usually have relevant representations, such as vectors, matrixes and tensors. Entities interact under a specific relation by performing arithmetical operations between entity embeddings and relation's representation.

3. FRAMEWORK

The sketch of our approach's framework is shown in Figure 1. In general, it contains two main parts: (a) selecting instances and (b) inferring instances.

Selecting instances ((a) in Figure 1). First, we choose an embedding-based model to learn distributed representations for both entities and relations in the existing knowledge base. For most of the existing embedding-based models, the objective is always to make $r(h, t)$ in the KB more similar than other $r(h, t^{'})$ or $r(h^{'}, t)$ generated randomly. For example, if *BornIn(Barack Obama, Honolulu)* exists in KB but *BornIn(Barack Obama, Washington)* does not, *(Barack Obama, Honolulu)* should be more similar than *(Barack Obama, Washington)* under the relation *BornIn*. Second, we employ learned representations of entities and relations to calculate similarity scores for each instance and sort them in descending order to pick the Top N to form a new smaller candidate set. To the query *Mother(Barack Obama, ?)*, there is an entity set $\mathcal{X}$, and each $x \in \mathcal{X}$ can replace *?* in *Mother(Barack Obama, ?)*, where x can be any person or female. *Mother(Barack Obama, $\mathcal{X}$)* denotes the entire candidate set $\mathcal{S}$, and we calculate all similarity scores for each *Mother(Barack Obama, x)* to pick N instances with the highest similarity scores for subsequent fact inference.

Inferring instances ((b) in Figure 1). In this component, we employ MLN to model KB and infer selected

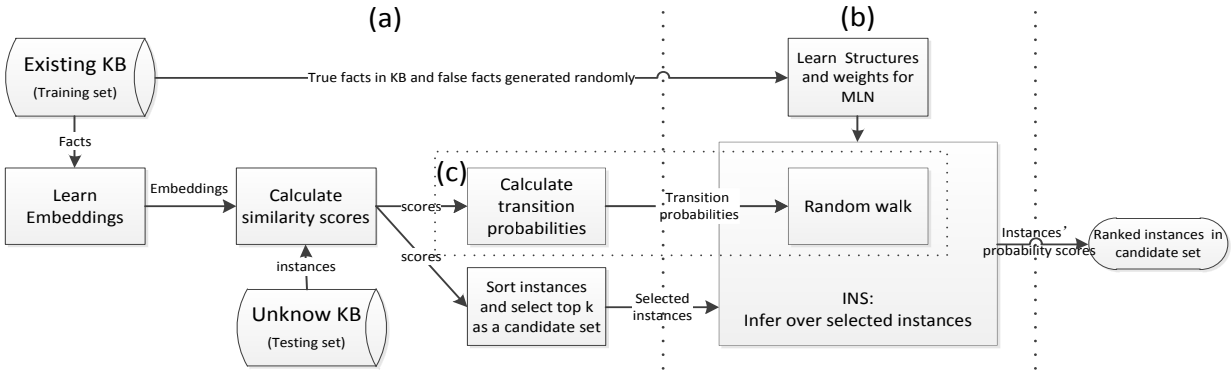


Figure 1: The sketch of the framework: (a) selecting instances; (b) inferring instances; (c) incorporating similarity priori generated by embedding-based methods.

instances on it. Before performing inference, we need to learn the structure and weights for MLN. Specifically, we employ the method in [24] to learn structure and weights simultaneously by performing random walks. Then, we propose an inferring algorithm, called Inferring via Ground Network Sampling or **INS** for short. INS performs random walks on the Partial Order Graph (POG) to generate paths and changes them to grounding formulas. For example, for *Mother(Barack Obama, Ann Dunham)*, we can obtain paths such as[*Barack Obama*] $\rightarrow$ [*Honolulu*] $\rightarrow$[*Ann Dunham*] from random walks and turn it into a formula such as *BornIn(Barack Obama,Honolulu)* $\wedge$ *LiveIn(Ann Dunham, Honolulu)* $\Rightarrow$ *Mother(Barack Obama, Ann Dunham)*. For each selected instance, we count formulas in which the instance occurs and treat counts of formulas as the instance features. Then, instance features are used to calculate probabilities for each instance. These probabilities can be viewed as final predictions for outputting.

Furthermore, we replace the uniform transition probability in random walks with the function of similarities calculated by the embedding model for the purpose of improving precision ((c) in Figure 1). Originally, each state adjacent to the current state has an equal transition probability, so the inference is completely independent from the previous instance selection. However, the similarity scores obtained from the embedding-based method should not only contribute in building candidate sets but should also be used in the following inferring process as a priori. The improved method employs a function from the similarity score to the transition probability, noted as $p_{trans} = f(s_{sim})$, where p_{trans} means the probability of transition from one state to an adjacent state in random walks and s_{sim} means the similarity score. For example, for *Mother(Barack Obama, x)*, if an state contains a selected instance, e.g., *Ann Dunham*, we assign $f(s_{sim}(Mother(BarackObama, AnnDunham)))$ to this state; otherwise, a small default probability is assigned to the adjacent state.

4. SELECTING INSTANCES BY USING EMBEDDINGS

To narrow candidate sets for subsequent fact inference, we make instance selection for each entire candidate set $\mathcal{S}$, for example, *Mother(Barack Obama, X)*. We propose employing embedding-based methods to do selection and describe the process as follows.

We first choose an existing embedding-based model to learn distributed representations for both entities and relations in the KB. Almost all embedding-based models embed entities into a relatively low (e.g., 50) dimensional embedding vector space $\mathbb{R}^k$ while representing relations in different ways. Therefore, for a specific fact $r(h,t)$, we represent h and t as vectors E_h and E_t, respectively, and abstract the representation of r as E_r. Meanwhile, different embedding-based models design different similarity-score-functions $f_s(E_h, E_t, E_r)$ to measure the similarity of entities under a specific representation of relations. For example, SE [5] represents a relation as two matrixes $R_r^{lhs} \in \mathbb{R}^{k\times k}$ and $R_r^{rhs} \in \mathbb{R}^{k\times k}$ and employs a kernel density estimation following the Gaussian kernel as the similarity-score-function; SME [3] represents a relation as a matrix, and its similarity-score-function is merged into a neural network; TransE [4] represents a relation as a vector with the same dimension as the entities and simply employs the negative 1-norm of the vector $(E_h + E_r - E_t)$.

Most of the embedding-based models design their objectives to make $f_s(E_h, E_t, E_r)$ larger than other $f_s(E_h, E_{t'}, E_r)$ or $f_s(E_{h'}, E_t, E_r)$, where $r(h,t)$ exists in the KB but $r(h,t^{'})$ or $r(h^{'},t)$ does not. Formally, we can define a uniform pairwise loss function as follows:

$$\mathcal{L} = \sum_{r(h,t)\in KB} \sum_{r(h^{'},t^{'})\in KB^{'}_{r(h,t)}} [\gamma + f_s(r(h^{'},t^{'})) - f_s(r(h,t))]_+ \tag{1}$$

where $[x]_+$ denotes the positive part of x. $KB^{'}_{r(h,t)}$ is sampled randomly from $\{r(h,t^{'})\} \cup \{r(h^{'},t)\}$, and $\gamma > 0$ is a margin hyperparameter.

Specifically, we employ TransE as the embedding-based model, which has been proved to be effective for knowledge base embedding [4]. We mainly apply TransE to generate candidate sets for inference in our experiments. TransE [4] is a simple and practical model, and it embeds all entities and relations into the same k-dimensional vector space by viewing r in $r(h,t)$ as a translation. TransE thinks E_t should be the nearest neighbor of $E_h + E_r$ in the same vector space and designs the similarity-score-function as $f_s(E_h, E_t, E_r) = -|E_h + E_r - E_t|$. In essence, TransE replaces the complex matrix multiplications with a vector addition operation.

After the learning process, we obtain embeddings for entities and relations in KB that contain semantic similarities. For example, the embedding of *Barack Obama* is close to

the embedding of *Ann Dunham* under *Mother* relation. We calculate similarity scores for each candidate $r(h,t)$ by employing the similarity-score-function $f_s(E_h, E_t, E_r)$, which is simple enough to be completed in a short running time. Then, we sort the candidate instances in descending order by their similarity scores and pick the Top N to form a new smaller candidate set. For example, for the query *Who is Barack Obama's mother*, we calculate all f_s(Mother(Barack Obama, x)) and pick the Top 10 persons as a new candidate set. We only require that *Ann Dunham* (the true answer) be in the Top 10 regardless of its place, which can be handled sufficiently by the existing embedding-based methods. In this way, the subsequent logical inference would be constrained by a smaller selected instance space, and the computational difficulty of inference-based methods is avoided.

5. INFERRING INSTANCES WITHIN CANDIDATE SET

The network structure and logical interaction in KBs make it natural to apply MLN to model KBs. Performing inference on MLN to predict missing links in KB is in accordance with KBC. In this section, we describe details of the inference approaches based on MLN, which includes two parts: 1) using MLN to model the exiting knowledge base for the task of KBC; 2) inferring overselected instances obtained from embedding-based methods. In terms of inference-based methods based on MLN, the huge candidate set is not the only reason that may cause the expensive calculation. The vast facts and the large grounding space also cause the unacceptable running time for both the learning and inferring process. In particular, learning structures for MLN suffer from a severe scale issue [15, 14, 13, 19, 11], which leads to the result that MLN-based approaches are scarcely applied to the task of large-scale KBC. We propose an inferring algorithm via grounding network sampling, noted as INS. INS employs the learning method in [24] to learn structure and weights for MLN simultaneously and performs random walks on the Partial Order Graph (POG) [1] instead of the original KB, which treats paths as POG nodes. INS is able to take all Horn clauses with the query at the place of both head and body, which can capture more logical factors and cover more useful formulas. As the INS extension, INS-ES, short for Inferring via ground Network Sampling Merging Embedding Similarity Priori, takes advantage of similarity priori obtained from previous embedding-based models to achieve a further promotion.

5.1 Modelling KBC by MLN

To obtain new knowledge from the existing KB, we regard all facts in the existing KB as evidences and regard candidate facts as queries. First of all, we need to model the existing KB and candidate facts in a MLN, and then rank candidates by their likelihood.

Markov logic can be viewed as a probabilistic extension of first-order logic by attaching weights to formulas. Each weight reflects the relative strength or importance of the corresponding formula. Higher weight indicates greater reward to a world that satisfies the formula. More formally, let X be the set of all true facts in KBs, F be the set of all first-order formulas in the MLN, w_i be the weight associated with formula $f_i \in F$, and G_{f_i} be the set of all possible groundings of formula f_i; then the probability of a possible world x is defined as:

$$\begin{aligned} P_w(X=x) &= \frac{1}{Z} exp(\sum_{f_i \in F} w_i \sum_{g \in G_{f_i}} g(x)) \\ &= \frac{1}{Z} exp(\sum_{f_i \in F} w_i n_i(x)) \end{aligned} \tag{2}$$

where $Z = \sum_{x' \in X} exp(\sum_{f_i \in F} w_i n_i(x^{'}))$ is the normalizing constant, w represents a set of formula weights, $g(x)$ equals 1 if g is satisfied and 0 otherwise, and $n_i(x)$ denotes the number of true groundings of f_i in x.

To model a knowledge base by MLN, we first view the KB from the perspective of Markov Networks: Facts can be true or false, so we treat all facts in the existing KB as binary random variables. We add them to a Markov Network as nodes and set their truth values as true. All facts not in the KB are not added to the Markov Network explicitly, and only a small part of them are generated when necessary. To represent interdependence between random variables, we add undirected edges for each two facts that share one common entity. There are no hyperedges in Markov Networks. When more than two nodes share one common entity, we add edges for each two of them to represent the dependence between two variables and use cliques [2] to represent dependence between all of them. MLN simplifies the dependence between more than two facts by splitting cliques into logical formulas. MLN generalizes formulas by replacing entities with variables and replacing facts with relation types. Finally, MLN learns weights for each conceptualized formula by counting groundings and treats weight formulas as features to predict missing links.

To obtain useful formulas and precise weights for one specific query $\mathcal{Q}$, we discriminatively learn structures and weights of the MLN for $\mathcal{Q}$, which means only facts under $\mathcal{Q}$ are treated as queries. Therefore, we care about the probability of the possible world only containing the specific query facts and treat other facts as evidence. We note the query as Y and compute its conditional probability given evidence X as follows:

$$P_w(Y=y|X=x) = \frac{1}{Z_y} exp(\sum_{f_i \in F_Y} w_i n_i(x,y)) \tag{3}$$

Different from Equation (2), each f_i in F_Y must involve the query Y, and groundings of f_i are counted when the Y is set as true and false, respectively. The normalizing constant Z_y aggregates the probabilities for all possible y, and $Z_y = \sum_{y' \in Y} exp(\sum_{f_i \in F_Y} w_i n_i(x, y^{'}))$.

For brevity, only non-recursive formulas are considered, and all queries become independent after evidence is given. The normalizing constant Z_y can be simplified as the sum of two parts when $Y=1$ and $Y=0$. Therefore, for one specific grounding query Y_j, its conditional probability is simplified as follows:

$$P_w(Y_j = y_j|X=x) = \frac{exp(\sum_{f_i \in F_{Y_j}} w_i n_i(x, y_{[Y_j=y_j]}))}{exp(\sum_{f_i \in F_{Y_j}} w_i n_i(x, y_{[Y_j=0]})) + exp(\sum_{f_i \in F_{Y_j}} w_i n_i(x, y_{[Y_j=1]}))} \tag{4}$$

[2] en.wikipedia.org/wiki/Clique_(graph_theory)

where $n_i(x, y_{[Y_j=y_j]})$ is the number of true groundings of the ith formula when all the evidence facts in X and the query Y_j are set to their truth values. Analogously, $n_i(x, y_{[Y_j=0]})$ and $n_i(x, y_{[Y_j=1]})$ are the numbers when Y_j is set as 0 and 1, respectively.

For each MLN model under a specific query relation $\mathcal{Q}$, we employ the algorithm described in [24] to learn structure and weights discriminatively. Algorithm [24] performs random walks starting from a number of initial nodes sampled randomly and collects paths constituted by linking entities during random walks. In this process, we generate grounding formulas by replacing each two adjacent nodes in a path with facts containing them in the existing KB. Therefore, one path obtained from random walks can be changed to various grounding formulas because there are possibly several facts under different relations between two adjacent entities.

After heuristic pruning, we only keep grounding formulas containing a true fact under $\mathcal{Q}$, and conceptualize them by replacing entities by variables. For example, the grounding formula *BornIn(Barack Obama,Honolulu)* $\wedge$ *LiveIn(Ann Dunham,Honolulu)* $\Rightarrow$ *Mother(Barack Obama, Ann Dunham)* is conceptualized to *BornIn*(x_1, x_2) $\wedge$ *LiveIn*(x_3, x_2) $\Rightarrow$ *Mother* (x_1, x_3). For one specific fact under $\mathcal{Q}$, all formulas attached to it are counted as its features, and the fact with counts of formulas are used to learn formula weights by Equation (4).

5.2 Inferring Over Selected Instances

For a specific query relation $\mathcal{Q}$, we have obtained candidate sets and an MLN learned under $\mathcal{Q}$. We exploit an inferring algorithm to independently calculate the probability of each selected instance being true, called **INS** (Inferring via ground Network Sampling). Additionally, we will describe its extension, **INS-ES** (Inferring via ground Network Sampling Merging Embedding Similarity Priori).

5.2.1 INS algorithm

INS is a data-driven algorithm and follows 4 steps: Performing random walks on POG; generating grounding formulas; counting formulas for instances; calculating probabilities for each instance. Algorithm 1 shows the detail of INS, where **KB** is the training set, **C** is the candidate fact set, $\mathcal{Q}$ is the query relation, **F** is the formula set with weights learned previously, **S** is the seed set, F_p is the grounding formula set, **q** is one query in **C**, and **QDI** is a map from query to data instance.

First, INS performs random walks on the POG to generate entity paths. We collect all entities occurring in selected instances under $\mathcal{Q}$ to form a set of initial nodes by employing each entity to build a single node path. Then, we perform random walks from each initial node to obtain paths of entities. There are formulas learned with a query at the place of body. For example, for *BornIn*(x_1,x_2) $\wedge$ *LiveIn*(x_3,x_2) $\Rightarrow$ *Mother*(x_1,x_3), when *BornIn* or *LiveIn* is treated as the query relation $\mathcal{Q}$, we regard it as a formula with the query in body. To obtain this type of formula, we assume all candidates are true and put them in the existing KB before performing random walks because the truth values of all candidates are unknown, and INS randomly walks to them only when they exist in the KB. The above process is presented in Line 1 to 7 of Algorithm 1; Line 5 to 7 is the core strategy of random walks which is detailed in Section 5.2.2.

Second, INS generates grounding formula bodies from each sampled path: We first transform adjacent nodes into grounding facts, which only occur in the existing KB. Then, to add heads to these formulas, we link the head and tail of the path with several possible relations, which can be true or false. After obtaining grounding formulas, all of them are conceptualized to generate a great deal of conceptualized formulas, which never have to be learned or pruned, and INS simply ignores them. Line 8 to 15 shows the generating and conceptualizing process.

Third, all formulas that remain are counted as features for a specific selected instance $r(h, t)$ that occurs in these formulas. INS travels all instances in the map of **QDI** and finally employs Equation (4) to calculate the probability for each candidate instance, which corresponds to Line 16 to 17. In particular, for some selected instance without any formulas, their scores are set to 0 directly.

Algorithm 1: INS (KB, C, $\mathcal{Q}$, F).

```
Input: KB, C, Q, F
Output: Probabilities for each instance in C.
1: Set q ∈ C to true, add them to KB.
2: QDI ← building pairs <q, empty instance>
3: S ← entities in q, q ∈ C.
4: Foreach s ∈ S:
5:     Initialize Path State p from s.
6:     Repeat Until MaxIterator:
7:         p ← RandomToNextPathState(p).
8:         F_p ← GenerateFormulas(p).
9:         Foreach f_p ∈ F_p.
10:            If q in f_p
11:                Conceptualize f_p.
12:                If f_p ∈ F
13:                    Add f_p truth counting to QDI(q)
14:                Else Ignore f_p.
15:            Else Ignore f_p.
16: Foreach q ∈ C:
17:    Calculate P(q|KB,F) by q and weights of
       F using Equation (4)
```

5.2.2 Grounding Network Sampling

We apply grounding network sampling to obtain entity paths uniformly and count formulas for each selected instance. To sample grounding network, we perform random walks on a POG.

POG is a directed graph and its nodes are subgraphs in the KB which is viewed as a graph. For KBC, we simplify it by only considering simple paths but not subgraphs. Edges in POG can be divided into three categories: super-edge, sub-edge and restart-edge. We take an example to explain them: There are two nodes, n_1 =[*Barack Obama*] $\rightarrow$[*Honolulu*] and n_2 = [*Ann Dunham*] $\rightarrow$ [*Barack Obama*] $\rightarrow$ [*Honolulu*]. The edge from n_1 to n_2 is a super-edge, while the edge from n_2 to n_1 is a sub-edge, and edges from n_1 or n_2 to the initial node are restart-edges, where the initial node can be [*Barack Obama*] or [*Honolulu*]. We make an further explanation for three types of edges, and they can be viewed as three operations: (1) Add one node to the existing path, e.g., add [*Ann Dunham*] to n_1 at the far left; (2) Remove one node from the existing path, e.g., remove [*Ann Dunham*] from n_2; (3) Restart from the initial node which is determined before the random walk starting, e.g., both n_1 and n_2 can restart from [*Barack Obama*] which is in query *Mother(Barack Obama,x)*.

To achieve a uniform distribution for all the simple paths, we exploit ideas from Metropolis-Hastings (MH) algorithm to perform random walks on the graph [1, 17]. Note the stationary distribution of node u as $\pi(u)$. At the current state $X_t = u$, the next state X_{t+1} is proposed with a proposal probability Q(u, v)(u $\neq$ v). The proposed transition to v is accepted with an acceptance probability $A(u, v)$, that is,

$$A(u,v) = min\{1, \frac{\pi(v)Q(v,u)}{\pi(u)Q(u,v)}\} \tag{5}$$

and rejected with probability $1 - A(u, v)$ in which case the state X_{t+1} remained unchanged.

The purpose of random walk on POG is to perform an unbiased graph sampling for MLN's structure and weight learning, so INS proposes that the target stationary distribution should be set to the uniform distribution $\pi = (1/n, 1/n, ..., 1/n)$, where $n = |N_{next}|$ represents the number of possible next states. Corresponding to super-edge and sub-edge, we also divide N_{next} into N_+ and N_-, and denote $d_+(u) = |N_+(u)|$ and $d_-(u) = |N_-(u)|$. Additionally, INS sets a fixed probability of restarting noted as γ. Exploring effect of adjusting γ to result is beyond focus of this paper, so we directly set $\gamma = 0.35$ in our experiment. The proposal probability is defined as follows:

$$Q(u,v) = \begin{cases} \gamma & , if\ restart \\ \frac{1-\gamma}{2d_+(u)}, if\ v \in N_+(u) \\ \frac{1-\gamma}{2d_-(u)}, if\ v \in N_-(u) \end{cases} \tag{6}$$

This process corresponds to the RandomToNextPathState(**p**) in Algorithm 1 (Line 7).

5.2.3 INS-ES algorithm

To incorporate the similarity a priori of candidates obtained from the previous embedding method, we exploit another inferring algorithm, noted as **INS-ES**. INS-ES replaces the uniform transition probability with the function of similarity scores obtained from the embedding-based method. It benefits from the framework of the MH algorithm, which employs a type of proposal probability $Q(u, v)$ to obtain a desired stationary distribution π for random walks. $Q(u, v)$ can be viewed as a state transition probability of an arbitrary irreducible Markov chain on the whole state space $\mathcal{N}$, with constraints: $Q(u, v) > 0$ if and only if $Q(v, u) > 0$, and $Q(u, v) = 0$ for all $(u, v) \notin \mathcal{E}(u \neq v)$, where $\mathcal{E}$ is the set of edges [17]. In INS, it assigns equal value to $Q(u, v), v \in N_{next}(u)$, which can achieve an unbiased graph sampling. However, now, we expect to utilize similarity a priori obtained from embedding-based methods in random walk process. We assign distinguishing $Q(u, v)$ to different adjacent nodes to break the unbiased graph sampling. This strategy can guide random walks to nodes containing selected instances with larger probabilities than others. More specifically, the instance, being at the more frontal position in the candidate set, is more likely to be visited.

In the previous step of making the instance selection, each selected instance has been assigned a similarity score for ranking. The embedding-based methods treat the similarity scores as the final results for KBC. Although Table 1 shows that these results are insufficient for the task of KBC, we believe there are still some priori knowledge that imply that one instance with a higher score is more likely to be true.

To replace the uniform probabilities by similarity priori, we just need to reassign transition probabilities only for super-edges and keep probabilities unchanged for sub-edges and restart-edges. During random walks, we use entities added to paths to represent candidate states for simplicity. When the current state is u and we expect to travel to the next state, $|N_+(u)|$ includes all entities adjacent to two marginal entities in the current path. For example, when the current state contains a path, $n_1 = [BarackObama] \rightarrow [Honolulu]$, $N_+(n_1)$ is composed of all entities adjacent to *[Barack Obama]* or *[Honolulu]*, and we assign diverse transition probabilities to states in $N_+(n_1)$. We split $N_+(n_1)$ into two parts: $N_{+cand}(n_1)$ and $N_{+usal}(n_1)$. $N_{+cand}(n_1)$ represents the subset, in which nodes all come from selected instances, and other nodes form $N_{+usal}(n_1)$. We assign distinguishing $Q(u, v)$ to entities in $N_{+cand}(u)$ and assign default $Q(u, v)$ for $N_{+usal}(u)$ as follows:

$$Q(u,v) = \begin{cases} \frac{(1-\gamma)f(score_v)}{2Z}, v \in N_{+cand}(u) \\ \frac{1-\gamma}{2Z} \qquad\quad , v \in N_{+usal}(u) \end{cases} \tag{7}$$

where Z is a normalizing constant and $Z = |N_{+usal}(u)| + \sum_{j \in N_{+cand}(u)} f(score_v)$, $score_v$ is the similarity score of $\mathcal{Q}(u, v)$ and $f(score_v)$ is a function to map similarity to probability. We design a heuristic equation for $f(score_v)$:

$$f(score_v) = \lambda \cdot \frac{score_v}{|score|_{max}} + 1 \tag{8}$$

To normalize $score_v$, we first divide it by $|score|_{max}$, which is the maximum absolute value of the similarity score under $\mathcal{Q}$. Then, multiply by a coefficient λ, which can adjust the strength of the influence from the embedding priori. With the increase in λ, an adjacent node that occurs in a selected instance has larger and larger transition probability from the current state, which means the strength of the embedding similarity priori is stronger and stronger. However, an overlarge λ means that other nodes that never occur in any selected instance are ignored easily and are never visited when λ is infinite. We add 1 to the function as a base value, so that the candidate entity with $score_v = 0$ has the same transition probability as ordinary ones. When $\lambda = 1$, $f(score_v)$ is always 1, and INS-ES degrades into INS.

6. EXPERIMENTS AND EVALUATIONS

We perform all experiments on an FB15K dataset, which is sampled from Freebase [4]. It contains 592,213 true facts with 14,951 entities and 1345 relation types. It has been split randomly into training, validation, and testing sets. These sets only contain true facts. For the learning process, both embedding-based methods and inference-based methods have their own strategies to generate false facts. However, for testing, false facts are uniformly generated for all methods as described in [4]. For each true fact $r(h, t)$ in the testing set, the tail t is replaced by every entity in the existing KB, and these are mixed with the true one, forming the Left Testing Subset. Then, the head h is replaced in the same way, noted as the Right Testing Subset.

Two parts of experiments are performed:

- Explore the effectiveness of embedding-based methods on KBC and discuss which embedding-based method performs best on instance selection, which is important

for subsequent logical inference based on our INS and INS-ES algorithms.

- Explore the effectiveness of INS-ES algorithm for KBC. We perform a comparison between our approaches and several state-of-the-art methods, including embedding-based methods and inference-based methods.

All following experiments are performed by a single thread on the servers with 24 Intel Xeon E5-2620 clocked at 2.00GHZ, with 64GB RAM, running Linux 2.6.32.

6.1 Instance Selection

We compare several embedding-based methods with a traditional approach to selecting different sizes of candidate sets. Hits@n can be viewed as the true facts coverage in n-size candidate sets because we believe that the more the true facts are covered, the better the selection is.

6.1.1 Compared Methods

BFS is a traditional approach based on an adjacent hypothesis, which means an entity t being adjacent or reachable via one skip from h implies $r(h,t)$ is more likely to be true. BFS is treated as a baseline.

Embedding-based methods described previously, **SE** [5], **SME**[3](linear and bilinear), **TransE** [4], are as compared. As a baseline for TransE, **Unstructured** (**Uns**, for short), which is a naive version of TransE that ignores the effect of various relations, is treated as a baseline as well.

6.1.2 Set up

We implement BFS in java by ourselves, but for Uns, SE, SME-lin, SME-bil and TransE, we use the codes on TransE's homepage[3] and keep the default setting for parameters. Similar to evaluation methods in [4, 26, 18], we evaluate Left Testing Subsets and Right Testing Subsets for each true fact $r(h,t)$ in testing KB. There may be other true facts in Left and Right Testing Subsets in addition to $r(h,t)$, but previous evaluation methods ignore them. For example, in the Left Testing Subset of $r(h,t)$, there is a true fact $r(h,t^{'})$ also occurring in testing the KB, and we should treat it the same as $r(h,t)$ when calculating Hits@n and AP@n. Our evaluation considers such $r(h,t^{'})$ in testing the KB, so that the Hits@10 value is different from those reported in [4, 26, 18].

6.1.3 Results

Figure 2 shows that, except for Uns, all embedding-based methods are better than BFS, the traditional approach. It indicates that most embedding-based methods have advantages on instance selection.

In Figure 2, the best Hits@1 is 32.911%, achieved by SME-bil, but after n increases to 10, TransE begins to outperform the other methods. In addition, TransE is the simplest method, and its running time is the shortest. Therefore, TransE is the most suitable method for generating smaller candidate sets for the subsequent logical inference.

TransE's Hits@50 has exceeded 90%, and Hits@100 is 94.568%, which is sufficient for subsequent fact inference. On the other hand, when n>100, all methods' Hits@n gradually approach 100%, and the performance of BFS starts to reach values that are close to the performance of embedding-based methods. The advantages of embedding-based methods disappear with the increase in n, and too large candidate sets would slow down the subsequent inferring algorithm. Therefore, we set the size of candidate set to 100 for subsequent experiments.

[3]https://everest.hds.utc.fr/doku.php?id=en:transe

6.2 Knowledge Base Completion

In this subsection, we compare our approaches with state-of-the-art methods including embedding-based methods and inference-based methods. Additionally, we explore the effect of similarity obtained a priori by the embedding-based method by comparing INS-ES with INS.

We employ Hits@n and AP@n as evaluations for the following experiments. Unlike instance selection, we employ Hits@1 as the most important metric because a good Hits@n with a large n cannot indicate that an approach has a good performance. AP@n is used as the other metric for the supplementary of Hits@n. AP@n can give a reasonable evaluation for $\mathcal{Q}(h,?)$, which has more than one true answer.

6.2.1 Compared Methods

Embedding-based methods We employ Uns, SE, SME-lin, SME-bil and TransE to represent embedding-based approaches in the same way as the methods in the experimental subsection of instance selection. However, we treat the output from embedding-based models as the final results.

Inference-based methods There are many methods that belong to this category. Most of them infer instances through logical formulas, but few of them can be used in KBC for their computation complexity. Here, we employ the INS algorithm to infer potential facts in the testing set and treat it as representative of inference-based methods.

Our approaches We apply the INS algorithm to infer overselected instances obtained from different embedding-based models. As **Embedding-based methods**, we choose Uns, SE, SME-lin, SME-bil and TransE to select instances and refer to the methods inferring over them as Uns-INS, SE-INS, SME-lin-INS, SME-bil-INS, TransE-INS, respectively.

Our approach with a priori We continue to expand TransE-INS by incorporating the similarity generated a priori by the embedding-based model into INS, noted as TransE-INS-ES. As an important super-parameter of TransE-INS-ES, λ in (in Equation 8) controls the strength of the effect of the similarity priori. We use TransE-INS-ES-λ to denote our method with different λ, where n denotes different λ.

6.2.2 Set up

We keep the default parameter setting for all embedding-based methods as in the instance selection. The number of selected instances is set to 100. For the INS algorithm, we set the maximum formula length as $l_f = 5$. The size of the initial node random walk starting from M_{random} is 500. The random walk times from one starting point T is 50 for learning and 1000 for inference. The restarting probability in random walk $P_{restart}$ is 0.35. The maximum number of iterations of the weight learner $M_{learner}$ is 100, and the L2-norm parameter C is 1. We employ the TransE-INS with 1000-size candidate sets to approximate INS because it cannot run completely to infer each instance directly, which is also the main reason that we make instance selection before inference. The implement is available online.[4]

[4]https://github.com/ZhuoyuWei/fpMLN

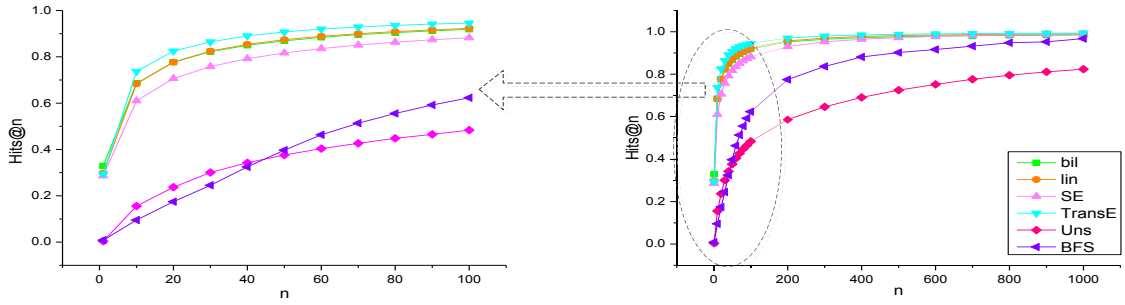


Figure 2: Hit curves of various methods for instance selection that generates candidate sets with different sizes. The range of n in the right graph is [0,1000], and the left graph is an enlarged view of the part at [0, 100].

Table 2: AP@n of various methods on FB15K

	Methods	Hits@1(AP@1)	Hits@10	AP@2	AP@3	AP@5	AP@7	AP@10
	Uns	0.00384	0.15573	0.01305	0.01747	0.02128	0.02336	0.02571
	SME-bil	0.32911	0.68506	0.32119	0.3208	0.32655	0.32951	0.33348
2.a	SME-lin	0.29807	0.68386	0.29136	0.29011	0.29426	0.29805	0.30102
	SE	0.28633	0.61026	0.28015	0.27996	0.28194	0.28480	0.28737
	TransE	0.29401	0.73710	0.30263	0.30835	0.31691	0.32099	0.32529
2.b	INS^*	0.32540	0.51917	0.30472	0.29668	0.28869	0.28519	0.28232
	Uns-INS	0.32899	0.39388	0.30518	0.29349	0.28302	0.28102	0.28288
	SME-bil-INS	0.69212	0.83464	0.67224	0.66194	0.65233	0.65051	0.65483
2.c	SME-lin-INS	0.68143	0.82478	0.65615	0.64645	0.63599	0.63568	0.64042
	SE-INS	0.67333	0.79938	0.65196	0.64158	0.63029	0.62823	0.63236
	TransE-INS	**0.69303**	**0.84894**	**0.67540**	**0.66566**	**0.65781**	**0.65550**	**0.66114**
2.d	TransE-INS-ES4.5	**0.71692**	**0.86633**	**0.70146**	**0.69320**	**0.68386**	**0.68100**	**0.68557**

6.2.3 *Ours method vs. State-of-the-art methods*

Table 2 shows the results of the comparison between our methods and state-of-the-art methods for KBC. From the results, we can obtain the following observations.

1) Our methods (except Uns-INS) in Table 2.c and 2.d achieve performance improvement over other methods in Table 2.a and 2.b. This indicates that our method based on Inferring via Grounding Network Sampling over Selected Instances is effective for the task of KBC.

2) Our methods, which perform inference after select instances by embedding-based methods in Table 2.c, all outperform only the corresponding embedding-based models in Table 2.a. For example, TransE's Hits@1 is 29.401%, which was the state-of-the-art method for KBC, while TransE-INS promotes Hits@1 to 69.303%, and TransE-INS also achieves much higher Hits@10 and AP@n. This proves the effectiveness of inferring instances following instance selection by embedding-based models, and the significant improvement is derived from the fact that our approaches consider explicit logical semantics and interaction between different relations, which is ignored in embedding-based models.

3) Our methods, which first make instances selection and generate small-scale candidate sets for INS in Table 2.c, outperform only INS in Table 2.b. The reason is that there are some noise instances, which may mislead the INS algorithm, but they are filtered by instance selection via embedding-based methods. It further indicates that embedding-based models can capture implicit factors that are ignored in inference methods, and both of these two methods cannot be substituted by each other.

4) Our extensional method, which incorporates the similarity generated a priori by the embedding-based model into INS in Table 2.d, outperforms our methods without the similarity priori in Table 2.c. This proves that the similarities obtained from embedding-based methods are helpful for promoting the inference precision as a priori.

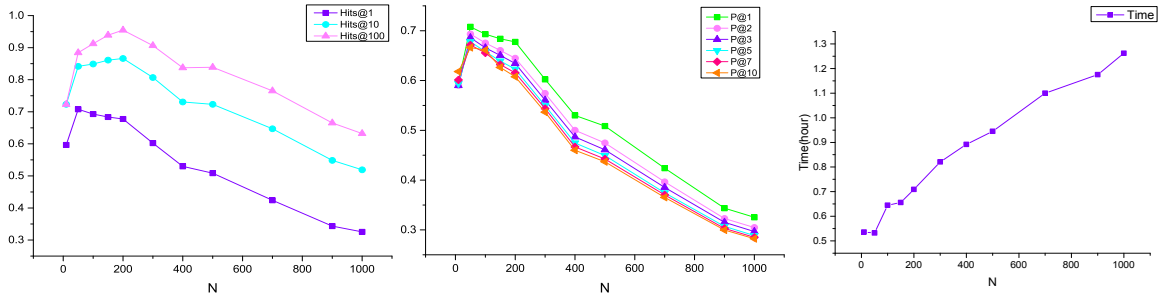
6.2.4 *Detailed Discussion*

In this part, we engage in a detailed discussion on our proposed approaches.

Effect of the candidate set size We explore the effect of the number of instances for subsequent fact inference, and we employ TransE-INS as the experimental method. Figure 3 shows the results. The horizontal axis represents the number of selected instances, noted as N. The vertical axes of (a), (b) represent Hits@n, AP@n, respectively. Hits@1,Hits@10 and Hits@100 are presented in Figure 3.a. The best Hits@1 occurs at N=50, while the best Hits@10 and Hits@100 occur at N=200. When N>200, the performance keeps decreasing with an increasing N, and the AP@n in Figure 3.b has the similar appearance. This indicates that making instance selection before inferring to generate small-scale candidate sets is an effective mechanism, which not only narrows the inferring space but also filters out part of the noise.

Effect to running time Figure 3.c shows the running time of performing inference, which indicates the computation complexity of inference is linearly growing with the increase in the candidate set size. This proves that inference on large-scale KBs without first reducing candidates would take an extremely long period of time. Therefore, selecting instances to narrow the inferring space in our approach is necessary and ensures that the inferring algorithm is completed in a short time.

Effect of the similarity priori We are going to prove the effectiveness of incorporating the similarity priori generated by the embedding-based model into INS by comparing TransE-INS with several TransE-INS-ES with different λ. Figure 4 shows the results of INS and INS-ES over 100 se-



(a) Hits@n (b) AP@n (c) Times

Figure 3: Hits@n, AP@n and Time curves of TransE-INS with different size candidate sets.

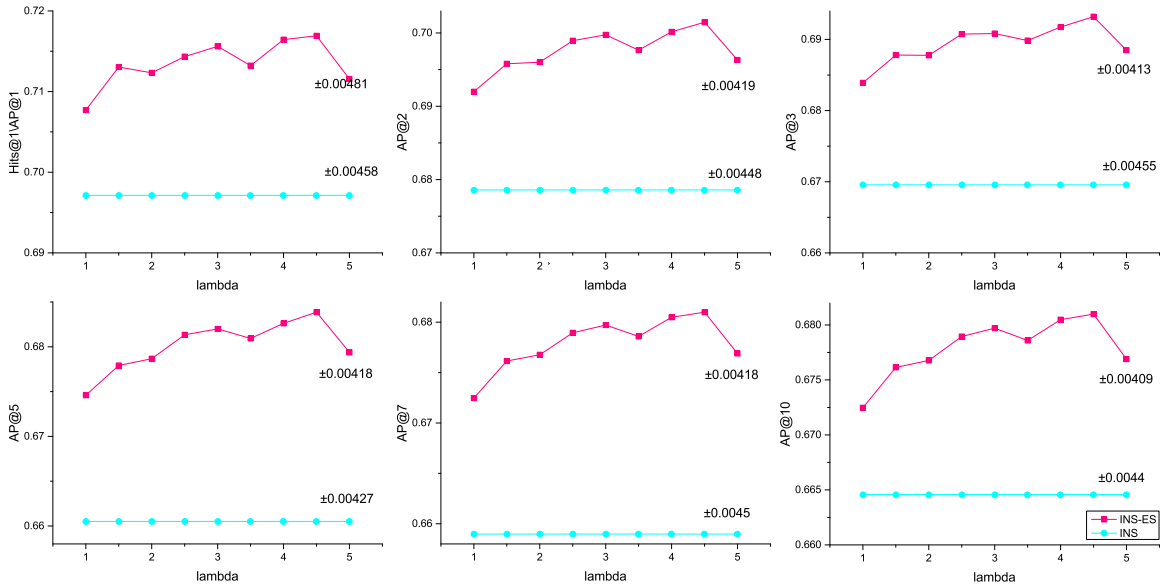


Figure 4: Comparing TransE-INS and TransE-INS-ES with different λ

lected instances generated by TransE. The horizontal axes represent λ, and the vertical axes in six sub-figures represent AP@1 (equal to Hits@1/100), AP@2, AP@3, AP@5, AP@7, AP@10. To reduce the randomness and highlight the effect of similarity priori, this experiment is repeated 5 times, and we report the mean and variance of AP@n. Regardless of the λ, INS-ES outperforms INS on Hits@1 and AP@n, and the best performance of INS-ES is achieved when $\lambda = 4.5$. This indicates that incorporating the similarity priori into a random walk process can achieve better performance. The curves also show that the performance is becoming better with the effect of similarity priori increase until $\lambda = 4.5$, and then the performance starts to become worse. The performance decrease may be caused by an excessively strong effect of similarity priori easily ignoring the nodes, which never occurs in selected facts, but these nodes may be accompanied by useful information.

7. RELATED WORK

Our work is related to two main categories of KBC methods: embedding-based approaches [21, 5, 12, 23, 4, 3, 26, 18] and inference-based approaches [22, 16, 24].

Nearly all embedding-based approaches represent entities as low dimensional vectors, but the representation for relations is multifarious. SE [5] represents each relation as two matrixes to capture the left and right interaction with one entity. RESCAL [21] represents relations as only one matrix and employs the product of three elements in a fact as similarity. SME[3] also represents relations as one matrix but employs an energy function to transform the interaction between the relation and entity. In the simplest way, TransE [4] represents relations in the same vector space as entities. The extensions of TransE, TransH [26] and TransR [18] also represent relations as low dimensional vectors, but there are differences between them: The former interprets a relation as a translating operation on a hyperplane to address one-to-many, many-to-one, and many-to-many relations, but the latter represents entities and relations in distinct spaces and projects entities from entity space to relation space. Tranc [9] and [12] take 2-way interactions between one entity and one relation into account. As a more complicated model, NTN [23] represents relations as tensors and matrixes simultaneously. However, these embedding-based methods do not consider explicit logical semantics and cannot sufficiently capture the interaction between different relations. Our approach applies a subsequent fact inference to overcome these shortages.

Inference-based methods usually view a KB as a graph, and seek, count or sample sub-structures (e.g., paths, subgraphs) to judge or calculate probability for a new fact. There are some typical inferring methods for KBC: ARM (association rule mining) [8], ILP (inductive logic programming) [25], and MLN [22]. They are all limited by the large

size of the candidate sets, but our approach speeds up MLN and employs it as a subsequent inference model for its high precision. MLN is an appropriate model for KBC because it applies weighted formulas to capture long-range interaction across several relations. The main issue of MLN is the computation complexity of the learning structure and inferring; thus, the methods based on MLN cannot be extended to large-scale knowledge bases. [15, 14, 13, 19, 11] are all proposed methods to speed up the learning and inferring process, but most of them still need grounding facts or formulas, so that the issue of computation complexity has still not been adequately addressed. PRA [16] employs random walks, but unlike our approach, it simplifies MLN by only keeping the horn clause with the query at the place of head. A promising algorithm is proposed by Sun [24], which is also a data-driven algorithm based on random walks that learns structure and weights simultaneously. Our approach takes advantage of this and employs a similar mechanism for inference and replaces the uniform transition probability for promotion.

8. CONCLUSIONS

We proposed employing embedding-based models to make instance selection for subsequent fact inference and exploited INS and INS-ES (incorporating embedding a priori) algorithms to infer facts on MLN, which improved Hits@1 from 32.911% to 71.692% for the FB15K dataset.

9. ACKNOWLEDGMENTS

The authors are supported by the National High Technology Research and Development Program of China (No. 2015AA015402) and the National Natural Science Foundation of China (No. 61272332, 61202329, 61303179 and 613031 72).

10. REFERENCES

[1] M. Al Hasan and M. J. Zaki. Output space sampling for graph patterns. *In VLDB*, 2009.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *SIGMOD*, 2008.

[3] A. Bordes, X. Glorot, J. Weston, and Y. Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 2013.

[4] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko. Translating embeddings for modeling multi-relational data. In *NIPS*, 2013.

[5] A. Bordes, J. Weston, R. Collobert, Y. Bengio, et al. Learning structured embeddings of knowledge bases. In *AAAI*, 2011.

[6] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[7] X. L. Dong, K. Murphy, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, T. Strohmann, S. Sun, and W. Zhang. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In *KDD*, 2014.

[8] L. A. Galárraga, C. Teflioudi, K. Hose, and F. Suchanek. Amie: association rule mining under incomplete evidence in ontological knowledge bases. In *WWW*, 2013.

[9] A. García-Durán, A. Bordes, and N. Usunier. Effective blending of two and three-way interactions for modeling multi-relational data. In *ECML/PKDD*. 2014.

[10] J. Hoffart, F. M. Suchanek, K. Berberich, and G. Weikum. Yago2: a spatially and temporally enhanced knowledge base from wikipedia. *Artificial Intelligence*, 2013.

[11] T. N. Huynh and R. J. Mooney. Discriminative structure and parameter learning for markov logic networks. In *ICML*, 2008.

[12] R. Jenatton, N. L. Roux, A. Bordes, and G. R. Obozinski. A latent factor model for highly multi-relational data. In *NIPS*, 2012.

[13] T. Khot, S. Natarajan, K. Kersting, and J. Shavlik. Learning markov logic networks via functional gradient boosting. In *ICDM*, 2011.

[14] S. Kok and P. Domingos. Learning the structure of markov logic networks. In *ICML*, 2005.

[15] S. Kok and P. Domingos. Learning markov logic networks using structural motifs. In *ICML*, 2010.

[16] N. Lao, T. Mitchell, and W. W. Cohen. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, 2011.

[17] C.-H. Lee, X. Xu, and D. Y. Eun. Beyond random walk and metropolis-hastings samplers: why you should not backtrack for unbiased graph sampling. In *SIGMETRICS*, 2012.

[18] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu. Learning entity and relation embeddings for knowledge graph completion. In *AAAI*, 2015.

[19] L. Mihalkova and R. J. Mooney. Bottom-up learning of markov logic network structure. In *ICML*, 2007.

[20] B. Min, R. Grishman, L. Wan, C. Wang, and D. Gondek. Distant supervision for relation extraction with an incomplete knowledge base. In *HLT-NAACL*, 2013.

[21] M. Nickel, V. Tresp, and H.-P. Kriegel. A three-way model for collective learning on multi-relational data. In *ICML*, 2011.

[22] M. Richardson and P. Domingos. Markov logic networks. *Machine learning*, 2006.

[23] R. Socher, D. Chen, C. D. Manning, and A. Ng. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*, 2013.

[24] Z. Sun, Z. Wei, J. Wang, and H. Hao. Scalable learning for structure in markov logic networks. In *AAAI 2014 Workshop on StarAI*, 2014.

[25] W. Y. Wang, K. Mazaitis, and W. W. Cohen. Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In *CIKM*, 2013.

[26] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *AAAI*, 2014.