Capturing Semantic and Syntactic Information for Link Prediction in Knowledge Graphs

Changjian Wang[1,2], Minghui Yan[1,2], Chuanrun Yi[1,2], and Ying Sha[1,2,3](✉)

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{wangchangjian,yanminghui,yichuanrun,shaying}@iie.ac.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] College of Informatics, Huazhong Agricultural University, Wuhan, China

Abstract. Link prediction has recently been a major focus of knowledge graphs (KGs). It aims at predicting missing links between entities to complement KGs. Most previous works only consider the triples, but the triples provide less information than the paths. Although some works consider the semantic information (i.e. similar entities get similar representations) of the paths using the Word2Vec models, they ignore the syntactic information (i.e. the order of entities and relations) of the paths. In this paper, we propose RW-LMLM, a novel approach for link prediction. RW-LMLM consists of a random walk algorithm for KG (RW) and a language model-based link prediction model (LMLM). The paths generated by RW are viewed as pseudo-sentences for LMLM training. RW-LMLM can capture the semantic and syntactic information in KGs by considering entities, relations, and order information of the paths. Experimental results show that our method outperforms several state-of-the-art models on benchmark datasets. Further analysis shows that our model is highly parameter efficient.

Keywords: Knowledge graph embedding · Link prediction · Random walk · Language model

1 Introduction

Knowledge graphs (KGs) are databases that contain facts about the world. Each fact in KGs is represented as a triple $\langle head\ entity, relation, tail\ entity \rangle$ denoted as $\langle h, r, t \rangle$, e.g., $\langle Washington, capitalOf, USA \rangle$. Recent years, several KGs such as YAGO [35], Freebase [3], NELL [5], and DBpedia [16] have been constructed. These KGs are extremely useful resources for many real-world applications such as question answering, information extraction, recommendation, etc. However, KGs are usually far from complete, i.e., missing links between entities, which hinders their usefulness in the above applications. Therefore, *link prediction* or *knowledge base completion* is proposed to solve this problem.

C. Ghidini et al. (Eds.): ISWC 2019, LNCS 11778, pp. 664–679, 2019.
https://doi.org/10.1007/978-3-030-30793-6_38

Recently, many methods have been proposed for link prediction. The most successful models are embedding-based, such as TorusE [8], SimplE [14], and ConvE [6]. In these models, entities are represented as vectors while relations are represented as vectors or matrices. By using the scoring function which is defined on the representation of each triple, these models can measure the likelihood of each candidate triple being a fact. However, these models only consider the triples, so the information they can use is limited.

Compared with the triples, the paths that are connected by multiple triples provide more information. Similar to DeepWalk [27], some works [10,11,21] treat the paths as context information and use the Word2Vec [22] models to learn the latent representations of entities or relations on KGs. Since Word2Vec models discard the word order information, these methods discard the order information of the paths either. If we treat the paths as natural language sentences, these methods can capture the semantic information (similar entities get similar representations) but cannot capture the syntactic information (the order of entities and relations) of the paths.

However, the syntactic information is very useful, it can tell us what the next word is given previous words, which is exactly the goal of link prediction (predicting the next entity given previous entity and relation). For example, given a path $A \xrightarrow{sonOf} B \xrightarrow{wifeOf} C$, if we capture the semantic information (B and C have similar representations) and the syntactic information (B is the next entity of A and $sonOf$), there is a high probability that $\langle A, sonOf, C\rangle$ is a fact.

To capture the semantic information and the syntactic information simultaneously, we propose RW-LMLM, a novel approach for link prediction. Figure 1 shows the overview of our method. The first part of our method is a random walk algorithm for KG (RW) and the second part is a language model-based link prediction model (LMLM) which is constructed by the Transformer Decoder [20,40]. In order to obtain the paths more conveniently, the triples in KG are converted to a graph. RW generates a set of paths by performing random walks along the outgoing direction of entities on the graph. The paths generated by RW consist of entities and relations while maintaining their order. These paths will be viewed as pseudo-sentences to train LMLM like the standard language model. Benefitting from the masked self-attention mechanism and the positional encoding of LMLM, the previous entities, relations and their order are considered when predicting the next entity, which makes LMLM have the ability to capture both the semantic and syntactic information. We evaluate our method on four benchmark datasets: WN18 [4], FB15k [4], WN18RR [6], and FB15k-237 [38]. Experimental results show that our method outperforms several state-of-the-art models.

In summary, our contributions are as follows:

- We propose RW-LMLM— a novel approach for link prediction. RW-LMLM can capture the semantic and the syntactic information in KGs.
- We analyze the parameter sensitivity of RW and find that we do not need too many walk steps to get the best performance, for FB15k-237, 5 steps is

enough. This may guide some works to choose a more reasonable path length to improve their performance.
– LMLM that utilizes the path information is more parameter efficient than some methods that only use the triples information. Compared with two state-of-the-art models ConvE and DistMult, LMLM is 2x parameter efficient than ConvE and at least 4x than DistMult.

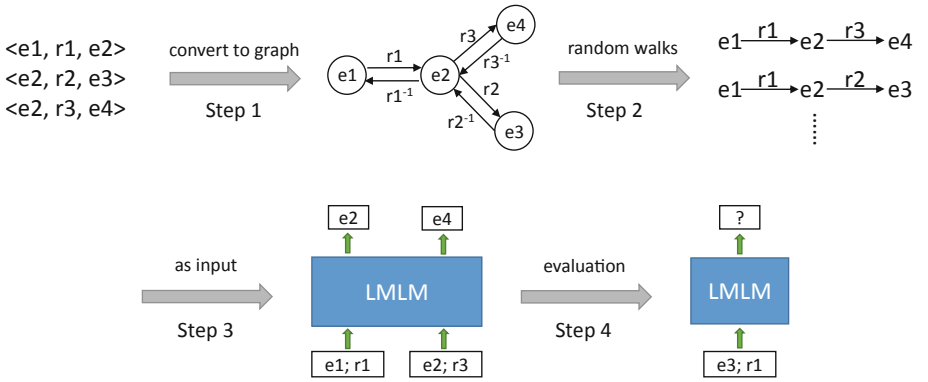

Fig. 1. Overview of our method. Step 1 converts triples to a graph; step 2 performs random walks on the graph to generate the paths; step 3 uses each path to train LMLM (taking $e1 \xrightarrow{r1} e2 \xrightarrow{r3} e4$ as an example); step 4 uses the trained LMLM to do link prediction task (taking $\langle e3, r1, ? \rangle$ as an example).

2 Related Work

Embedding models for link prediction have been quite popular in recent years. They usually use the triples to get the representations of entities and relations. We roughly divide them into three categories: translation-based, bilinear, and neural network-based. TransE [4] is the first translation-based model. It model the relation as a translational vector to correlate the head and tail entity embeddings. Many works extend TransE by projecting the head and tail embeddings into the relation vector space using projection vectors or matrices, such as TransH [42], and TransR [18]. Using the same principle as TransE, TorusE [8] embeds entities and relations on a torus. Unlike translation-based models, bilinear models represent the relation as a matrix. RESCAL [26] has no restrictions on the relation matrix. DistMult [43] restricts relations to diagonal matrices, and ComplEx [39] is DistMult's extension in complex space. SimplE [14] is a simple interpretable fully expressive bilinear model. The first two types of models are simple, efficient, and easy to expand, but they are less expressive than neural network-based models. NTN [33] has a neural network architecture, which allows mediated interaction of entity vectors via a tensor. MLP [7] is a simplified version of NTN where each relation is associated with one vector and then a

standard multi layer perceptron is used to capture interaction terms. ConvE [6] is a highly parameter efficient model which uses 2D convolution over embeddings and multiple layers of non-linear features to model KGs.

The information provided by the triples is limited. Many works try to exploit richer context information. Some utilize neighbor information of entitie, such as GAKE [10], TransE-NMM [24], and R-GCN [32]. Relation paths between two entities are more deep information for link prediction. PRA [15] is an early work. Recent research usually combines the relation path into a new relation between two entities by addition, multiplication, or RNN, such as PTransE [17], TransE-COMP [12], and Bilinear-COMP [12]. Our method also uses the relation paths. Instead of treating them as new relations, we treat them as pseudo-sentences together with the entity paths, which can make full use of the intermediate information of the paths.

Our work is closely related to DeepWalk [27] which uses Skip-gram [22] on the information generated by random walks to get the latent representations of vertices on social graphs. Luo et al. [21], Goikoetxea et al. [11], and GAKE [10] use the similar idea on KGs. Our method differs from these in several aspects. (1) DeepWalk and Goikoetxea et al. only consider the entities, and they all do not consider the order information. Our method considers all three aspects of the path: entities, relations, and order information. (2) All these works use the Word2Vec models (CBOW or Skip-gram), but we use the multi-layer Transformer Decoder language model, which is more expressive, more suitable for ordered data, and better at capturing high-level information especially the syntactic information. (3) They are just embedding models which aim at obtaining the latent representations of entities or relations. Our model is not only an embedding model, but also a link prediction model which can be used directly for link prediction task.

3 Problem Definition

A KG is represented as a set of triples (facts) $\mathcal{O} = \{\langle h, r, t\rangle\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Each triple $\langle h, r, t\rangle$ denotes a relation $r \in \mathcal{R}$ between head entity $h \in \mathcal{E}$ and tail entity $t \in \mathcal{E}$, where $\mathcal{E}$ and $\mathcal{R}$ are the sets of entities and relations respectively. We convert KG $\mathcal{O}$ to a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. We treat each entity e as a vertex, each relation r as a directed edge from h to t. We also add a directed edge from t to h to represent the inverse relation r^{-1}, which is widely used to make full use of the structural information of KG [10].

Link prediction usually refers to the task of predicting another entity for a given entity and relation, i.e., predicting t given h and r ($\langle h, r, ?\rangle$) or h given r and t ($\langle ?, r, t\rangle$). For example, $\langle Washington, capitalOf, ?\rangle$ means to predict which country's capital is Washington, and $\langle ?, capitalOf, USA\rangle$ means to predict which city is the capital of the USA. We unify the two tasks by converting the latter to the former, i.e., predicting head entity given tail entity and inverse relation ($\langle t, r^{-1}, ?\rangle$). Unlike other methods which learn the scoring function $f_r(h, t)$, we directly learn the conditional probability distribution of the

target entity $P(T|I, R; \Theta)$ where I and R denote h and t or t and r^{-1}, and Θ denotes parameters which learned by LMLM.

4 Methodology

In this section, we will describe the two parts of our method (RW and LMLM) in detail. The purpose of RW is to obtain the paths in KGs. These paths will be viewed as pseudo-sentences to train LMLM. Just like the standard language model, the objective of LMLM is to maximize the probability of the entities in the paths.

4.1 RW: Random Walks on KG

We perform random walks on G to get a set of paths $\mathcal{P}_{\mathcal{ER}}$ for training. For the sake of convenience, we fix the length l of a path P_{ER}. Random walks are along the outgoing direction of entities, so the entities in the paths are in the form of head-to-tail and the relations are in the middle, such as $e_0 \xrightarrow{r_0} e_1 \xrightarrow{r_1} \cdots \xrightarrow{r_{l-1}} e_l$. A path P_{ER} in $\mathcal{P}_{\mathcal{ER}}$ contains $l + 1$ entities and l relations

Algorithm 1. Random Walks on KG

Input: graph $G = (V, E)$
number of iterations t
walk length l
Output: a set of paths $\mathcal{P}_{\mathcal{ER}}$

```
for i = 1 to t do
  V = Shuffle(V)
  for each v ∈ V do
    add v to path P_ER
    for j = 1 to l do
      randomly choose an outgoing adjacent vertex e of v
      randomly choose an edge r between v and e
      add r and e to P_ER
      v = e
    end for
    add P_ER to P_ER
  end for
end for
return P_ER
```

Algorithm 1 shows our approach. We perform t iterations on V. At the start of each iteration, we generate a random ordering $\mathcal{V}$ of V. Random walks are performed staring from each vertex in $\mathcal{V}$. A random walk walks l steps along the outgoing direction of a start vertex to get a path P_{ER}. For each step, we randomly choose an outgoing adjacent vertex firstly, then choose the edge between the two

vertices. If there are multiple edges between the two vertices, we will randomly choose one. Finally, we will get a set of paths $\mathcal{P}_{\mathcal{ER}}$.

This algorithm is similar to the algorithm proposed by [27]. The main difference is that our random walks are performed on directed graphs with relation information. Instead of randomly choosing an adjacent edge and the vertex on the edge, we first choose an adjacent vertex and then randomly choose an edge between the two vertices. Since there may be multiple edges (relations) between two vertices (entities), the former method may break the balance of the number of entities in the paths.

4.2 LMLM: Language Model-Based Link Prediction Model

Inspired by the ability of the standard language model to capture the semantic and syntactic information of natural language sentences as well as the ability to predict the next word given previous words, we construct the link prediction model LMLM based on the standard language model.

The standard language model usually defines a probability distribution over a sequence of tokens: $P(w_1, w_2, \ldots, w_n) = \prod_i P\left(w_i | w_1, \ldots, w_{i-1}\right)$. The goal of language modeling is to maximize this probability. The conditional probabilities $P\left(w_i | w_1, \ldots, w_{i-1}\right)$ can be learned by neural networks [2,23]. Recent years, the Transformer Decoder (TD) [20], the decoder part of Transformer [40], has been widely used for language modeling [20,30,31]. Our model is also constructed using the TD.

TD. Figure 2 shows the architecture of TD. The TD is mainly composed of two parts: masked Multi-Head Attention layer and Feed-Forward layer. The masked Multi-Head Attention consists of multiple scaled dot-product attention [40] which is a commonly used attention function [9,37,41]. The self-attention in this layer is masked to computes the hidden representation of each position by considering the representations of previous positions and itself. The Feed-Forward layer is a fully connected position-wise Feed-Forward Network (FFN) which consists of two linear transformations. It is applied to each position separately and identically. In addition, the TD uses residual connection [13] and layer normalization [1] between every two layers.

LMLM. We treat a path as a pseudo-sentence, and the set of paths is a corpus. We use the corpus to train our model. Formally, given a path $P_{ER} = e_0 \xrightarrow{r_0} e_1 \xrightarrow{r_1} \cdots \xrightarrow{r_{l-1}} e_l$ where $e_i \in \mathcal{E}$ and $r_i \in \mathcal{R}$, the input of our model is $\left(\left(e_0, r_0\right), \left(e_1, r_1\right), \ldots, \left(e_{l-1}, r_{l-1}\right)\right)$ and the target is $\left(e_1, e_2, \ldots, e_l\right)$. Similar to the standard language model, our objective is to maximize the following probability:

$$P(e_1, e_2, \ldots, e_l) = \prod_{i=1}^{l} P\left(e_i | \left(e_0, r_0\right), \ldots, \left(e_{i-1}, r_{i-1}\right)\right). \tag{1}$$

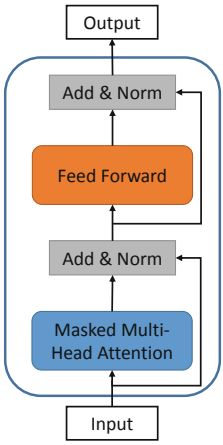

Fig. 2. TD architecture

We divide the input into two parts, the entities input $(e_0, e_2, \ldots, e_{l-1})$ and the relations input $(r_0, r_1, \ldots, r_{l-1})$. They are represented as the one-hot matrices $M_E \in \mathbb{R}^{l\times|\mathcal{E}|}$ and $M_R \in \mathbb{R}^{l\times|\mathcal{R}|}$ respectively. Each position of the input has a positional encoding, which is represented as a position embedding matrix $W_p \in \mathbb{R}^{l\times(d_E+d_R)}$ where d_E and d_R are the embedding dimension of entities and relations respectively. We use the fixed position embedding matrix proposed by [40]:

$$W_p(i,j) = \begin{cases} \sin\left(i/10000^{j/(d_E+d_R)}\right) & \text{if } j \bmod 2 = 0 \\ \cos\left(i/10000^{(j-1)/(d_E+d_R)}\right) & \text{if } j \bmod 2 = 1. \end{cases} \tag{2}$$

We use the TD to get the conditional probability distribution of the i-th target entity $y_i = \hat{P}\left(T|\left(e_0, r_0\right), \ldots, \left(e_{i-1}, r_{i-1}\right)\right)$:

$$h_0 = [M_E W_E; M_R W_R] + W_p, \tag{3}$$

$$h_k = TD(h_{k-1}), k \in [1, n], \tag{4}$$

$$y_i = softmax(h_n^i W_h W_E^T), i \in [1, l], \tag{5}$$

where $W_E \in \mathbb{R}^{|\mathcal{E}|\times|d_E|}$ and $W_R \in \mathbb{R}^{|\mathcal{R}|\times|d_R|}$ are the entity embedding matrix and the relation embedding matrix respectively; $[;]$ is a concatenation operator on row vector of two matrices; h_k^i is the i-th hidden representation of k-th TD layer; n is the number of layers of the TD; $W_h \in \mathbb{R}^{(d_E+d_R)\times d_E}$ is a linear transformation matrix.

Figure 3 shows the architecture of LMLM. Firstly, the model obtains the d_E-dimensional (d_R-dimensional) representations of entities (relations) by W_E (W_R). Then the model concatenates the representations of entities and relations. After adding the positional encoding, the representations are used as the initial input of the TD. After the multi-layer TD, the representations are projected

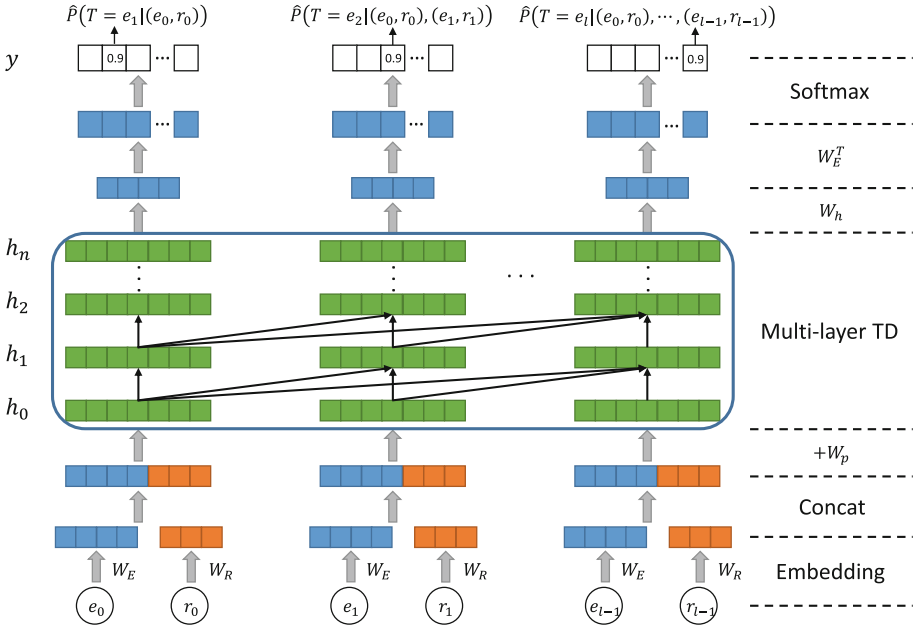

Fig. 3. LMLM architecture

to d_E-dimensional space using a linear transformation matrix W_h. The model then projects the representations to $|\mathcal{E}|$-dimensional space using W_E^T (i.e. output embedding [28]) and gets the probability distribution of the next entity using the softmax function. Since the positional encoding and the masked self-attention of TD, the previous i-1 entities, relations and their order are considered when predicting the i-th entity.

We train our model by minimizing the following loss function:

$$\mathcal{L} = -\sum_{i=1}^{l} \log \hat{P}\left(T = e_i | \left(e_0, r_0\right), \ldots, \left(e_{i-1}, r_{i-1}\right)\right). \tag{6}$$

We adopt stochastic gradient descent to train our model. To reduce overfitting, we regularise our model by using label smoothing [36] and dropout [34]. In particular, we use dropout on the embeddings and the FFN layers of TD.

5 Experiments

5.1 Datasets

We evaluate our method on four benchmark datasets: WN18 [4], FB15k [4], WN18RR [6], and FB15k-237 [38]. WN18 is a subset of Wordnet. FB15k is a subset of Freebase. WN18RR and FB15k-237 are subsets of WN18 and FB15k

respectively. WN18RR and FB15k-237 are created by removing inverse relations to form more challenging, realistic datasets. All these datasets consist of three parts: training set, validation set, and testing set. Table 1 presents the statistics of the four datasets.

Table 1. Statistics of the experimental datasets. #train, #valid, and #test represent the number of triples in training set, validation set, and testing set, respectively.

Dataset	$\|\mathcal{E}\|$	$\|\mathcal{R}\|$	#train	#valid	#test
WN18	40,943	18	141,442	5,000	5,000
FB15k	14,951	1,345	483,142	50,000	59,071
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

5.2 Evaluation Protocol

The purpose of link prediction is to predict the target entity given an input entity I and relation R. We can get the probability distribution of the target entity $P(T|I,R)$ by the trained LMLM. We rank the probability values in descending order. The top ranked entity is more likely to be the target entity which we want to predict.

Given an entity e_1 and a relation r_1, if the predicted entity e_2' ranks higher than the target entity e_2, but $\langle e_1, r_1, e_2' \rangle$ is a fact in KG, this is not wrong. In order to avoid this misleading behavior, we remove such type entities that exist in training, validation, or testing set before ranking. We call the original one *raw*, the filtered one *filt*. [4].

To measure the performance of different methods in link prediction, we employ several common evaluation metrics: Hits@N, Mean Rank (MR), and Mean Reciprocal Rank (MRR). Hits@N denotes the proportion of the target entities that are ranked within top N. MR is the mean of the target entities' rankings. MRR is the mean of multiplicative inverse of the target entities' rankings. Higher Hits@N, lower MR, and higher MRR indicate better performance.

5.3 Experimental Setup

We first utilize RW on the training sets to generate the paths for model training. For WN18 and WN18RR, the number of iterations is 50 and walk length is 10. For FB15k and FB15k-237, the number of iterations is 200 and walk length are 10 and 5 respectively.

We use grid search to select the hyperparameters of LMLM. The ranges of hyperparameters are as follows: entity embedding dimension d_E in {50, 100, 200}, relation embedding dimension d_R in {10, 30, 50}, FFN layer dimension d_f in {500, 800, 1000}, the number of TD layers n in {1, 2, 4, 5}, embedding dropout dp_e in {0.1, 0.2, 0.3}, batch size bs in {64, 128}. We fix some parameters: learning

Table 2. Link prediction results on WN18 and FB15k (*raw*)

Method	WN18			FB15k		
	MR	MRR	Hits@10	MR	MRR	Hits@10
TransE [4]	263	—	0.754	243	—	0.349
STransE [25]	217	0.469	0.809	219	0.252	0.516
GAKE [10]	—	—	—	228	—	0.445
ANALOGY [19]	—	0.657	—	—	0.253	—
R-GCN [32]	—	0.553	—	—	0.251	—
TransAt(asy,bern) [29]	**169**	—	0.814	**185**	—	0.529
TorusE [8]	—	0.619	—	—	0.256	—
RW-LMLM	318	**0.664**	**0.852**	211	**0.322**	**0.572**

rate is 0.01, label smoothing is 0.2, attention heads is 4, and FFN layer dropout is 0.1. We find the following hyperparameters work well on the four datasets: $d_E = 100$, $d_R = 10$, $d_f = 500$, $n = 2$, $dp_e = 0.1$, $bs = 128$ on WN18; $d_E = 100$, $d_R = 30$, $d_f = 500$, $n = 4$, $dp_e = 0.2$, $bs = 128$ on WN18RR; $d_E = 100$, $d_R = 30$, $d_f = 800$, $n = 2$, $dp_e = 0.1$, $bs = 128$ on FB15k; $d_E = 100$, $d_R = 30$, $d_f = 500$, $n = 4$, $dp_e = 0.3$, $bs = 64$ on FB15k-237. Best models are selected by using early stopping according to Hits@10 on the validation sets, with up to 30 epochs over the set of paths. Our code is available online[1].

5.4 Results

Table 2 shows the results of several methods on WN18 and FB15k under the *raw* setting. Our method achieves the best MRR and Hits@10 on the both datasets. Table 3 shows the results of several methods on the two datasets under the *filt.* setting. We evaluate methods on Hits@N in more detail, including Hits@1, Hits@3, and Hits@10. The results show that our method obtains the best MRR and all Hits@N. Compared with GAKE which is a work similar to ours but does not consider the order information (i.e. missing syntactic information), our method achieves the relative improvement of 7%/29% in MR/Hits@10 on FB15k (*raw*) and 37%/35% in MR/Hits@10 on FB15k (*filt.*)

It has been noted by [38] that many testing triples are inverse triples of training triples in WN18 and FB15k, which makes these triples easy to learn. Our method may benefit from it on the two datasets. To demonstrate the superiority of our method, we evaluate our method on more challenging datasets: WN18RR and FB15k-237 which remove inverse relations in WN18 and FB15k. Table 4 shows the results. We achieve the best MR and all Hits@N on WN18RR and the best MRR, Hits@3, and Hits@10 on FB15k-237. The results of MRR on WN18RR and Hits@1 on FB15k-237 are close to the best.

[1] https://github.com/chjianw/RW-LMLM.

Table 3. Link prediction results on WN18 and FB15k (*filt.*)

Method	WN18					FB15k				
	MR	MRR	Hits@N			MR	MRR	Hits@N		
			1	3	10			1	3	10
TransE [4]	**251**	—	—	—	0.892	125	—	—	—	0.471
ComplEx [39]	—	0.941	0.936	0.945	0.947	—	0.692	0.599	0.759	0.840
GAKE [10]	—	—	—	—	—	119	—	—	—	0.648
ANALOGY [19]	—	0.942	0.939	0.944	0.947	—	0.725	0.646	0.785	0.854
R-GCN [32]	—	0.814	0.686	0.928	0.955	—	0.651	0.541	0.736	0.825
SimplE [14]	—	0.942	0.939	0.944	0.947	—	0.727	0.660	0.773	0.838
ConvE [6]	504	0.942	0.935	0.947	0.955	**64**	0.745	0.670	0.801	0.873
RW-LMLM	308	**0.949**	**0.944**	**0.951**	**0.957**	75	**0.762**	**0.694**	**0.809**	**0.877**

Table 4. Link prediction results on WN18RR and FB15k-237 (*filt.*). Results marked * are taken from [6].

Method	WN18RR					FB15k-237				
	MR	MRR	Hits@N			MR	MRR	Hits@N		
			1	3	10			1	3	10
DistMult [43]*	5110	0.43	0.39	0.44	0.49	254	0.241	0.155	0.263	0.419
Node+LinkFeat [38]	—	—	—	—	—	—	0.226	—	—	0.347
Neural LP [44]	—	—	—	—	—	—	0.24	—	—	0.362
R-GCN [32]	—	—	—	—	—	—	0.248	0.153	0.258	0.414
ConvE [6]	5277	**0.46**	0.39	0.43	0.48	**246**	0.316	**0.239**	0.350	0.491
RW-LMLM	**4286**	0.45	**0.42**	**0.47**	**0.51**	358	**0.321**	0.231	**0.352**	**0.507**

We note that our method performs well on most metrics except MR. One explanation for this is that our method targets the most accurate entities given previous entities and relations, while MR reflects the average performance of methods, and a single bad ranking of target entity can greatly affect MR even the others perform well. Compared with MR, MRR is more reasonable and robust. It uses the mean of multiplicative inverse of the target entities' rankings, so the effect of bad triples is reduced, and the rankings of target entities can be distinguished, i.e., lower rankings will have higher scores. Our method achieves the best MRR on three of the four datasets and is on par with ConvE on WN18RR.

6 Analysis

We analyze our method on FB15k-237 in several aspects, including parameter sensitivity of RW, parameter efficiency of LMLM, and ablation studies. The experimental setup is the same as in Sect. 5.3 unless otherwise specified.

6.1 Parameter Sensitivity of RW

We investigate the effect of the two parameters (i.e. the number of iterations t and the walk length l) of RW on Hits@10. We experiment on the FB15k-237 dataset.

Figure 4 shows the results. We note that Hits@10 increases as t increases, and the trend slows down. This is intuitive. More iterations can get more information to help the improvement of performance. $l = 5$ has an improvement on Hits@10 compared to $l = 3$, but from $l = 5$ to $l = 10$ Hits@10 has almost no change. This means that we do not need too many walk steps to get the best performance on FB15k-237.

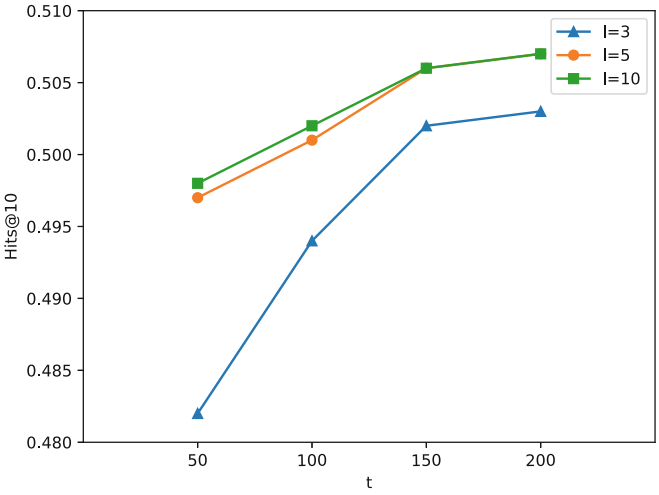

Fig. 4. Results of different parameters in RW. t is the number of iterations and l is the walk length.

6.2 Parameter Efficiency of LMLM

We compare the number of parameters with a bilinear model DistMult and a neural network-based model ConvE to demonstrate the parameter efficiency of LMLM.

Table 5 shows the results on FB15k-237. We can see that LMLM performs better than DistMult and ConvE with the same number of parameters. LMLM with 0.95M parameters performs better than ConvE with 1.89M parameters on Hits@10 and is the same on MRR. Similar results are also reported on LMLM with 0.46M parameters and ConvE with 0.95M parameters. LMLM with 0.46M parameters still performs better than DistMult with 1.89M parameters on both Hits@10 and MRR. Overall, LMLM is 2x parameter efficient than ConvE, at least 4x than DistMult.

LMLM is more parameter efficient than ConvE and DistMult, probably because LMLM utilizes the path information while the other two only utilize the triples, so even with fewer parameters, LMLM can still capture enough information.

Table 5. Parameter comparison on FB15k-237. Results of DistMult and ConvE are taken from [6].The embedding size refers to the entity embedding size and the numbers in brackets are the relation embedding sizes. For DistMult and ConvE, their relation embedding size and entity embedding size are the same.

Model	Parameter count	Embedding size	MRR	Hits@10
DistMult	1.89M	128	0.23	0.41
	0.95M	64	0.22	0.39
ConvE	1.89M	96	0.32	0.49
	0.95M	54	0.30	0.46
	0.46M	28	0.28	0.43
LMLM	1.89M	83(30)	0.32	0.50
	0.95M	43(20)	0.32	0.50
	0.46M	21(12)	0.29	0.46

6.3 Ablation Studies

We perform two ablation studies on FB15k-237 and the results are shown in Table 6.

First, we investigate the effect of missing the relation information on performance by removing relation embedding in LMLM. The model without relation information has a dramatic decline in performance compared to the full model. This is in line with expectations, since the relation information is one of the most important information in KGs and it is critical to performance. Second, we investigate the effect of missing the order information on performance by removing the position embedding and disordering the triples in LMLM. Compared with the full model, the performance of the model without the order information declines on all metrics, up to 8% relative decrease on Hits@1. The results demonstrate that the order information (i.e. the syntactic information) contributes to the performance improvement of link prediction.

Table 6. Ablation studies on FB15k-237

Model	MRR	Hits@1	His@3	Hits@10
Full model	0.321	0.231	0.352	0.507
w/o relation	0.035(↓ 0.286)	0.007(↓ 0.224)	0.017(↓ 0.335)	0.061(↓ 0.446)
w/o order	0.301(↓ 0.020)	0.212(↓ 0.019)	0.329(↓ 0.023)	0.484(↓ 0.023)

7 Conclusion and Future Work

This paper proposes a novel method RW-LMLM for link prediction in KGs. RW-LMLM consists of two parts, including RW—a random walk algorithm for KG, and LMLM—a language model-based link prediction model. The paths generated by RW are treated as pseudo-sentences and they are used to train LMLM like the standard language model. RW-LMLM has the ability to capture the semantic and syntactic information in KGs since it considers entities, relations, and order information of the paths. Experimental results on four datasets show that our method outperforms previous state-of-the-art models. Compared to some methods that only utilize the triples, our method that utilizes the path information is more parameter efficient. We also analyze the parameter sensitivity of RW and we find that more walk steps may not always necessary. This may help other works to choose a reasonable path length when they want to improve link prediction performance by the path information. Our work is an attempt to solve the problem in KGs using natural language processing method. Experimental results show the competitiveness of this way.

In the future, we plan to explore the following directions: (1) although the effect of the dimensions of entities and relations on performance is reflected in Sect. 6.2, more specific study is necessary. (2) We plan to study the relationship between the optimal path length and the number of entities or relations on more datasets.

Acknowledgements. This work is supported by the National Key Research and Development Program of China (2017YFB0803301).

References

1. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv: 1607.06450(2016)
2. Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C.: A neural probabilistic language model. J. Mach. Learn. Res. **3**(Feb), 1137–1155 (2003)
3. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1247–1250. ACM (2008)
4. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in Neural Information Processing Systems, pp. 2787–2795 (2013)
5. Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E.R., Mitchell, T.M.: Toward an architecture for never-ending language learning. In: Twenty-Fourth AAAI Conference on Artificial Intelligence (2010)
6. Dettmers, T., Minervini, P., Stenetorp, P., Riedel, S.: Convolutional 2d knowledge graph embeddings. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
7. Dong, X., et al.: Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 601–610. ACM (2014)

8. Ebisu, T., Ichise, R.: Toruse: Knowledge graph embedding on a lie group. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
9. Fang, S., Xie, H., Zha, Z.J., Sun, N., Tan, J., Zhang, Y.: Attention and language ensemble for scene text recognition with convolutional sequence modeling. In: 2018 ACM Multimedia Conference on Multimedia Conference, pp. 248–256. ACM (2018)
10. Feng, J., Huang, M., Yang, Y., et al.: Gake: graph aware knowledge embedding. In: Proceedings of COLING 2016 the 26th International Conference on Computational Linguistics: Technical Papers, pp. 641–651 (2016)
11. Goikoetxea, J., Soroa, A., Agirre, E.: Random walks and neural network language models on knowledge bases. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1434–1439 (2015)
12. Guu, K., Miller, J.J., Liang, P.: Traversing knowledge graphs in vector space. In: Empirical Methods in Natural Language Processing, pp. 318–327 (2015)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.,pp. 770–778 (2016)
14. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. In: Advances in Neural Information Processing Systems, pp. 4284–4295 (2018)
15. Lao, N., Cohen, W.W.: Relational retrieval using a combination of path-constrained random walks. Mach. Learn. **81**(1), 53–67 (2010)
16. Lehmann, J., et al.: Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. Semant. Web **6**(2), 167–195 (2015)
17. Lin, Y., Liu, Z., Luan, H., Sun, M., Rao, S., Liu, S.: Modeling relation paths for representation learning of knowledge bases. In: Empirical Methods in Natural Language Processing, pp. 705–714 (2015)
18. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: Twenty-ninth AAAI Conference on Artificial Intelligence (2015)
19. Liu, H., Wu, Y., Yang, Y.: Analogical inference for multi-relational embeddings. In: Proceedings of the 34th International Conference on Machine Learning, volu. 70, pp. 2168–2178. JMLR. org (2017)
20. Liu, P.J., et al.: Generating wikipedia by summarizing long sequences. arXiv preprint arXiv:1801.10198 (2018)
21. Luo, Y., Wang, Q., Wang, B., Guo, L.: Context-dependent knowledge graph embedding. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1656–1661 (2015)
22. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. arXiv: Computation and Language (2013)
23. Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association (2010)
24. Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M.: Neighborhood mixture model for knowledge base completion. In: Conference on Computational Natural Language Learning, pp. 40–50 (2016)
25. Nguyen, D.Q., Sirts, K., Qu, L., Johnson, M.: Stranse: a novel embedding model of entities and relationships in knowledge bases. arXiv preprint arXiv:1606.08140 (2016)
26. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. ICML **11**, 809–816 (2011)

27. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM (2014)
28. Press, O., Wolf, L.: Using the output embedding to improve language models. arXiv preprint arXiv:1608.05859 (2016)
29. Qian, W., Fu, C., Zhu, Y., Cai, D., He, X.: Translating embeddings for knowledge graph completion with relation attention mechanism. In: IJCAI, pp. 4286–4292 (2018)
30. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training. URL https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/languageunsupervised/languageunderstandingpaper. pdf (2018)
31. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
32. Schlichtkrull, M., Kipf, T.N., Bloem, P., van den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 593–607. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_38
33. Socher, R., Chen, D., Manning, C.D., Ng, A.: Reasoning with neural tensor networks for knowledge base completion. In: Advances in Neural Information Processing Systems, pp. 926–934 (2013)
34. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
35. Suchanek, F.M., Kasneci, G., Weikum, G.: Yago: a core of semantic knowledge. In: Proceedings of the 16th International Conference on World Wide Web, pp. 697–706. ACM (2007)
36. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2818–2826 (2016)
37. Tan, Z., Wang, M., Xie, J., Chen, Y., Shi, X.: Deep semantic role labeling with self-attention. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
38. Toutanova, K., Chen, D.: Observed versus latent features for knowledge base and text inference. In: Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality, pp. 57–66 (2015)
39. Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., Bouchard, G.: Complex embeddings for simple link prediction. In: International Conference on Machine Learning, pp. 2071–2080 (2016)
40. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
41. Vemula, A., Muelling, K., Oh, J.: Social attention: modeling attention in human crowds. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–7. IEEE (2018)
42. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: Twenty-Eighth AAAI Conference on Artificial Intelligence (2014)
43. Yang, B., Yih, W.t., He, X., Gao, J., Deng, L.: Embedding entities and relations for learning and inference in knowledge bases. arXiv preprint arXiv:1412.6575 (2014)
44. Yang, F., Yang, Z., Cohen, W.W.: Differentiable learning of logical rules for knowledge base reasoning. In: Advances in Neural Information Processing Systems, pp. 2319–2328 (2017)