Information Sciences 609 (2022) 679–690

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Approximate inferring with confidence predicting based on uncertain knowledge graph embedding

Shihan Yang [a], Weiya Zhang [b,*], Rui Tang [a], Mingkai Zhang [a], Zhensheng Huang [c]

[a] Faculty of Management and Economics, Kunming University of Science and Technology, Kunming 650500, China
[b] School of History and Administration, Yunnan Normal University, Kunming 650500, China
[c] College of Electronic Information, Guangxi University for Nationalities, Nanning 530006, China

ARTICLE INFO

Article history:
Received 29 November 2020
Received in revised form 8 April 2022
Accepted 17 July 2022
Available online 20 July 2022

2022 MSC:
00-01
99-00

Keywords:
Knowledge presentation
Knowledge graphs
Uncertainty embedding
Approximate inferring
Recurrent neural network

ABSTRACT

Uncertainty is a natural character of knowledge, while it is still tough to be encoded into the knowledge graph embedding space that can be employed for machine learning tasks. However, the approximate inference could be performed in the embedding space, if confidence, real-value representation of the uncertainty of knowledge facts, can be learned by neural networks. To tackle this, a simple yet effective confidence predicting method is proposed, and several approximate inferring are efficiently performed based on these predictions. The model is a two-step model: knowledge elements embedding step, in which knowledge facts regarded as short sentences are fed into the natural language model to get entity and relation embedding vectors; and confidence learning step, in which the confidence distribution of knowledge facts in the knowledge graph are learned utilizing the recurrent neural network in order to carry out approximate inference. The experience demonstrates that the model achieves better results than state-of-the-art on the link prediction task over uncertain knowledge graph embedding. Uncertainty inferring grounded on predicted confidence is more accurate, feasible, and meaningful for serval knowledge inferring tasks: transitivity, composition inferring, and probabilistic soft logic inferring. Likewise, the proposed approach achieves the best tradeoff between efficiency and accuracy of uncertain knowledge graph embedding and inferring, and can be used to handle large size knowledge graphs at lower time consumption because of the simplicity.

1. Introduction

Knowledge graphs (KG), a kind of knowledge base represented by graphs, is a collection of facts that have the form of (*head*, *relation*, *tail*), also known as triples. The confidence tells the uncertainty of facts, with which approximate knowledge inferring can be carried out. According to the value range of the confidence of those facts, knowledge graphs can be divided into two categories: (1) Deterministic knowledge graphs, such as Freebase [1] and YAGO [2], which the confidence of a fact is 1 or 0 (1 means that the fact occurs in the KG, 0 does not); (2) Uncertain knowledge graphs or non-deterministic knowledge graphs, such as NELL [3] and ConceptNet [4], in which each fact associates with a confidence score between 0 and 1 that

* Corresponding author.
E-mail addresses: dr.yangsh@kust.edu.cn (S. Yang), dr.zhangwy@aliyun.com (W. Zhang), tangrui@kust.edu.cn (R. Tang), 690055397@qq.com (M. Zhang), 285912615@qq.com (Z. Huang).

https://doi.org/10.1016/j.ins.2022.07.098
0020-0255/

represents the trust degree of the fact to be true. Knowledge graph embedding (KGE), where entities (heads and tails) and relations are encoded into a continuous low-dimension vector space that can then be employed for machine learning tasks, has become one of the most promising approaches for knowledge graphs analysis [5] [6]. Knowledge graph embedding has shown amazing solutions on downstream tasks, such as link prediction, recommendation, question answering, and triplet classification. However, most recent works have focussed on deterministic knowledge graph embedding. In fact, uncertain knowledge graph embedding should be more focused on due to the intrinsic incompleteness of knowledge bases and the uncertain nature of knowledge [7]. Literature [8] argues that it is critical to obtain and preserve uncertain semantic information in the embedding space for several reasons. Originally, uncertainty is an essential part of knowledge. Uncertain knowledge representation has great benefits for various knowledge-driven applications, such as uncertain decision-making, and auto-answering systems. Secondly, considering uncertainty enhances inference in knowledge-driven tasks, such as text understanding in the Natural Language Processing field, which often entails interpreting real-world concepts that are always ambiguous or intrinsically vague. As far as we know, capturing the uncertain semantic information of uncertain knowledge graphs is still an open problem. Seldom works have focused on the uncertain knowledge graphs embedding research. There are several challenges to this situation:

- In the embedding space, the uncertainty information is hard to be adequately preserved, neither are relationship rules, for example, implication and equivalence.
- Approximate inferring grounded on the confidence of facts is frequently unexplainable and not intuitive in the embedding space.
- The cost of time and space of training uncertain knowledge graph embedding is actually expensive.
- It is difficult to suitably estimate the uncertainty of unseen facts, instead of regarding them as zero when they do not occur in the knowledge graph like deterministic knowledge graph embedding models do.

To tackle these problems, we have proposed a model UKG_sE (Uncertain Knowledge Graph simple-but-effective Embedding). This model performs confidence predicting fast and effectively. The main idea is to treat each knowledge fact as a short sentence with only several words (three or more) to fast encode all entities and relations into embedding vectors by Word2-Vec [9], then train an LSTM (Long Short-Term Memories) [10] neural network to learn the confidence distribution of all facts in the knowledge graph embedding space. When considering a relation fact as a sentence, the knowledge base can be regarded as a document including all short-length sentences, some graph structures can be regarded as paragraphs, and all entities and relations are combined into a corpus. So the model is a two-step training framework: knowledge elements (i.e. entities and relations) embedding and confidence learning. The first step learns knowledge elements embedding from the knowledge base document with equal treatment of entities and relations. Each knowledge element embedding is an entity embedding vector or a relation embedding vector collected from word embedding vectors, which are the pre-training parts of the approach. And we also believe that the well pre-trained can lead to high-quality embedding as [11] argued. The second step learns the confidence score distribution of all facts by the LSTM neural network, which can memorize hidden fact-sentence structure information and confidence of facts. After training the model UKG_sE, we performed the confidence prediction of facts seen or unseen in the knowledge graph. We have achieved a better result in link prediction on many uncertain knowledge graph benchmark datasets, and better evaluating on the mean squared error for testing datasets than the state of the art. Furthermore, we obtained these results with lower training time and less computing power, which means the approach can also be used to address very large-size uncertain knowledge graph embedding requirements with available accuracy.

2. Related works

In this section, we will consider two aspects of related works: uncertainty embedding of knowledge graphs and approximate knowledge inferring in the embedded space.

A probabilistic embedding method, KG2E [12], has been proposed, which models the (un) certainties of entities and relations in KG based on a Bayesian framework, and considers the uncertainty of knowledge at a word-level, not at a sentence-level when treating a triple as a sentence, so fewer knowledge structures are embedded. [13] has proposed an embedding uncertain networks method based on matrix factorization, URGE (UnceRtain Graph Embedding), which can generate only node embeddings for a given general graph. In addition, this model is not specific to uncertain knowledge graphs, because it can handle the uncertainty of knowledge by only considering the node proximity, not explicit relations of knowledge graphs. Knowledge graphs are heterogeneous graphs with different kinds of relationships. [14] has focused on partial order embedding over concept spaces and extended the order embedding by probability theory to address the uncertainty of knowledge. This method limits the partial order of general graphs. However, in most cases, the knowledge map goes beyond the partial order of a single kind of concept. The literature [15] argues that there is still a lack of methods for quantifying predictive uncertainty in a knowledge graph embedding representation, and has proposed a highly scalability probabilistic model, LIM (Latent Information Model) and LFM (Latent Fact Model) based on neural variational inference, to evaluate the predictive uncertainty during link prediction tasks. This approach has roughly the same goal as confidence learning, the second step of our framework, but our model is more intuitive, concise, and faster to perform. Recent work [8] has proposed

firstly the uncertain knowledge graph embedding problem definition: Given an uncertain knowledge graph, the embedding model aims to encode entities and relations in a low-dimensional real space in which structure information and confidence scores of relation facts are preserved as much as possible. And they have proposed an embedding approach based on vector encoding. Furthermore, the continued work [16] has proposed another method based on probabilistic box embeddings [17]. We follow this problem definition and develop a natural but effective method and get some better results in learning confidence of uncertain knowledge graphs than those they reported.

Thanks to the construction of several large-scale uncertain knowledge bases, such as ConceptNet, NELL, uncertain knowledge graphs have enabled lots of knowledge-driven applications recently. For example, [18] has used NELL to improve the quality and efficiency of an open-domain question answering system. [19] has employed ConceptNet as background knowledge to greatly improve the automated theorem prover, CoRg system.

Knowledge inferring tasks, whose goal is to automatically discover new knowledge from known knowledge, are the core of knowledge-driven applications, while inferring in the embedding space has been often unexplainable and obscure. Recent works have shown that embedding logic rules as background knowledge into the continuous space can improve the performance in downstream tasks [20] [21]. However, rule-embedding is an obstacle [22] because that rule learning has been often by grounding rules, where the grounding is time-consuming and is hard to complete due to the incompleteness of KGs. Neural approaches, like in nature language inferring (NLI), attempt to use the powerful learning ability of neural networks to represent the facts in knowledge graphs and thereby obtain better reasoning ability [23], which employ convolutional neural networks (CNN) [24], recurrent neural networks (RNN) and attention mechanism [25] [26], reinforcement learning [27], and so on. Today, neural approaches top virtually all leaderboards for NLI tasks [28], so should do for knowledge reasoning. For predicting the uncertainty of facts in knowledge graphs, we adopt recurrent neural network LSTM to train our model. The difficulty of approximate knowledge graph reasoning based on the confidence of relation facts lies in fast calculating and accurately evaluating the confidence. In another way, [29–31] augment triples (actually increase the size of knowledge data) by mining rules to improve the quality of embedding.

In addition, [22] argues that encoding rules are as important as providing practical solutions for encoding rules, and so do we. Further, we think the practice of inferring in embedding spaces is more important. Section 6 demonstrates the approximate inferring based on predicted confidence in several examples, including symmetry and asymmetry inference, and inference based on probabilistic logic.

3. Uncertain knowledge graphs embedding

Uncertain knowledge graphs associate each relation fact with a confidence score so-called an uncertain fact. Formally, let $l = (s, p, o, c)$ be an uncertain fact of an uncertain knowledge graph K, containing a subject (head entity) s, a predicate (relation) p, an object (tail entity) o, and a confidence score c. For any fact $(s, p, o, c) \in E \times R \times E \times [0, 1]$, where E is the set of entities, R is the set of relations, and the confidence score is a real value between 0 and 1, which is interpreted as a probability of this uncertain fact to be true. Uncertain knowledge graph embedding aims to encode each entity and relation into a real low-dimensional vector space, in which semantical information of the original graph including its structures and uncertainties of facts are preserved as much as possible. A representation function F is defined as

$$F : E \cup R \rightarrow \mathbb{R}^d \tag{1}$$

assigning a real vector with dimensionality d to every entity and relation. And a confidence function

$$C : \mathbb{R}^d \times \mathbb{R}^d \times \mathbb{R}^d \rightarrow [0, 1] \tag{2}$$

assigns each relation fact in the embedding space with a real value between 0 and 1. That is, for any $l \in K$, existing $l_v = (s_v, p_v, o_v, c\prime)$ where $s_v, p_v, o_v \in \mathbb{R}^d$ and $c\prime \in [0, 1], c\prime \approx c$. We focus on discovering simple yet effective functions F and C to preserve much structural information and confidence distribution in the embedding space enough so as to make the valuable approximate inference. We also consider the following problems when constructing functions F and C:

- Trading off between generating embedding at a high rate and preserving more uncertain information as much as possible;
- Validating the distributional hypothesis of confidences of uncertain facts in the uncertain knowledge graph embedding space;
- Effectively performing approximate inference in the embedding space for some specific domain knowledge tasks.

4. UKG$_s$E Model

We propose a two-step framework, UKG$_s$E, consisting of a pre-training step and a confidence learning step. In the pre-training step, function F is performed by regarding each relation fact without confidence $l = (s, p, o)$ as a short sentence with only three (or more, but short) words, and employing language models to train the entity and relation embedding. The function F means language models, such as the skip-gram model and the CBOW model [32], and generates fact embedding vectors $l_v = (s_v, p_v, o_v) = F(s, p, o)$. In the confidence learning step, function C is performed by the recurrent neural network

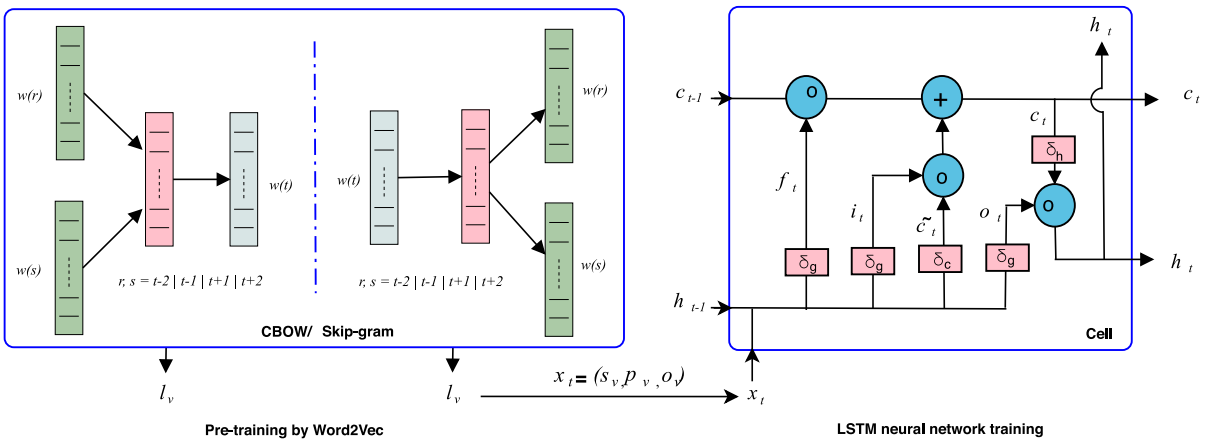

Fig. 1. Schematic diagram of the training process.

LSTM [33] [10] [34], which learns the confidence score of each uncertain fact $c\prime = C(l_v)$. The function C is LSTM. Fig. 1 shows the schema of UKG$_s$E.

4.1. Entities and relations embedding

In the natural language processing field, the continuous bag of words model (CBOW) uses a continuously distributed representation of the context to predict the current word. On the other hand, the continuous skip-gram model uses each current word as an input to predict words within a certain range (window) before and after it.

With the skip-gram model [35], combing all entities and relations into a corpus $E \cup R$, we consider a triple without confidence as a short sentence of three phrases over the corpus, $T = \{u_s, u_p, u_o\}$, and aim to maximize the average log probability

$$\frac{1}{3}\sum_{u\in T}\sum_{u\prime\in T\setminus u} log\ p(u\prime|u) \tag{3}$$

where the context window is 2 as for the word sequence size is always $|T| = 3, T \setminus u$ is a difference set of T and u. And the probability is defined by the soft maximum function

$$p(u\prime|u) = \frac{exp(v^{O^\top}_{u\prime} \cdot v^I_u)}{\sum_{x\in E\cup R} exp(v^{O^\top}_x \cdot v^I_u)} \tag{4}$$

where v^O_x and v^I_x are respectively output vectors representation and input vectors representation of the word x, and $\top$ represents the transpose of a matrix. During the actual model training, x selects all members from a batch samples set, and five negative samples are sampled by corrupting the object strategy.

The CBOW model, the mirror of the skip-gram, predicts the current word from context, and the context window of it is likewise 2. The optimization aims to minimize the following loss function

$$L = -\sum_{u\prime\in T\setminus u} \log(p(u|u\prime)) \tag{5}$$

where $u\prime$ is in the context of $u \in T$. In the experiments, we found that there was a slight difference between basing on these two different models. For one dataset, adopting CBOW in the first step, the prediction of confidence has better results; for another dataset, the skip-gram leads to better prediction. We consider this later in Section 5.

4.2. Confidence learning

Employing an LSTM neural network to evaluate the confidence scoring function (2), and feeding a sequence of embedding vectors $l_v = (s_v, p_v, o_v)$ for each fact $(s, p, o, c) \in K$ into the neural network, the model outputs a confidence score c after a dense hidden layer connected to a single output neuron with sigmoid activation function. Adam optimizer is chosen in the model training and the early-stopping strategy is adopted.

LSTM is an artificial recurrent neural network architecture, which has not only feed-forward like a standard neural network but also feedback connections. The advantage of an LSTM network compared to a common recurrent unit is its cell unit [10], the memory part of the LSTM, which usually includes three gates: an input gate, an output gate and a forget gate. The cell vector can encapsulate the notion of forgetting part of its previously-stored memory, as well as adding part of the new

information. Intuitively, the cell is responsible for keeping track of the dependencies between the elements in the input sequence. The input gate controls the extent to which a new value flows into the cell, the forget gate controls the extent to which a value remains in the cell and the output gate controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit. The compact forms of the equations for the forward pass of an LSTM unit with a forget gate are:

$$\begin{aligned} f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\ \tilde{c_t} &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c_t} \\ h_t &= o_t \circ \sigma_h(c_t) \end{aligned} \tag{6}$$

where the lowercase variables represent vectors, matrices W and U contain the weights of the input and recurrent connections, for which the subscript can either be the forget gate f, the input gate i, output gate o or the memory cell c, depending on the activation being calculated. The subscript t indexes the time step, and x_t is the input vector of the LSTM cell unit, h_t the output vector of that. When the d and h refer to the number of input features and number of hidden units, respectively, $x_t \in \mathbb{R}^d, h_t, f_t, i_t, o_t, \tilde{c_t}, c_t \in \mathbb{R}^h, W \in \mathbb{R}^{h\times d}, U \in \mathbb{R}^{h\times h}$ and $b \in \mathbb{R}^h$, bias vectors. Furthermore, the activation function σ_g is sigmoid function, σ_c hyperbolic tangent function, and σ_h the peephole connection function, $\sigma_h(x) = x$. The operator $\circ$ denotes element-wise product. In our model, the initial values are $c_0 = \mathbf{0}$ and $h_0 = \mathbf{0}$, and d is three times length of the dimension of embedding vectors, h equals to the dimension. The right hand side of Fig. 1 shows the schema of the LSTM unit.

5. Experiment and results

To evaluate the capability of the model, we experimented with and performed it on the link prediction task. Over two datasets, CN15k and PPI5k, the subgraph of two different uncertain knowledge graphs, UKG$_s$E is performed in Python 3.6 with the *Gensim* and *PyTorch* libraries, on which the MSE and MAE evaluations have been done, runtime also has been analyzed. Sources, datasets, and execution logging are available online [1]. All experiments were carried out on the laptop computer with Win10, 8G RAM, Intel 4-cores CPU @ 2.30 GHz, and without GPUs.

5.1. Metrics and baselines

In the community, the knowledge graph embedding is usually evaluated on the link prediction task (or knowledge base completion). Mean square error (MSE) and mean absolute error (MAE) are the simple evaluation criteria between original confidence values and confidence scores computed by the model in the test data set. The smaller the value of MSE and MAE is, the better the model is. Meanwhile, we treat the relational facts that appear in the knowledge base and the relational facts that do not appear in the same way. Furthermore, we consider the runtime performance, which evaluates the handling of a total number of triples per second.

An uncertain knowledge graph embedding is a knowledge graph embedding with a confidence distribution of all relation facts, in which the inference is far beyond the link prediction since the inferring task is often heavily dependent on external algorithms [36]. Regrading evaluation of inference tasks in uncertain KGE, we predict the confidence of relation facts by the function that outputs scores, $c \approx LSTM(s_v, p_v, o_v)$ with (s_v, p_v, o_v) inputs. It is quite different from link predictions that are calculated by embedding vectors addition $s_v + p_v \approx o_v$ in KGE, or by a neural network $NN(s_v, p_v) \approx o_v$ with (s_v, p_v) inputs.

For a query $(s, p, ?\underline{o})$, we rank all the entities in the vocabulary as object candidates and evaluate the ranking performance by means of Normalized Discounted Cumulative Gain (NDCG) [37]. After defining the gain in retrieving an object o_0 as the ground truth confidence score $c(s, p, o_0)$, we take the mean NDCG over the test query set as the ranking metric. Two evaluating versions of the gain are reported: linear gain and exponential gain, in the next section. On the accurate evaluation of implication inference in the embedding space, we illuminate them by case studies in Section 6.

To compare with our results, for MSE, MAE, mean NDCG values, and case studies, we select baselines UKGE$_{rect}$ and UKGE$_{logi}$ in [8], URGE in [13] and BEUrRE in [16], which are state of the art results on uncertain KG embedding. For runtime, we adopt the baseline KG2Vec in [36], which is a state of the art in quickly generating Knowledge Graph embedding.

5.2. Datasets

CN15k dataset is extracted from ConceptNet [4], and matches the number of nodes with FB15k [38], the widely used benchmark dataset for deterministic Knowledge Graph Embeddings. CN15k is a subgraph of the commonsense knowledge base ConceptNet, contains 15,000 entities, 36 relations, and 234,675 uncertain relation facts (delete the duplicate 6,483 tri-

[1] https://github.com/ShihanYang/UKGsE

ples from a total 241,158). The original confidence scores vary from 0.1 to 22, and [8] normalized them into $[0.1, 1.0]$ by min–max normalization after bound the confidence values to $[0.1, 3.0]$. The average and standard deviation of all confidence values are 0.627 and 0.234, respectively. We get the dataset from the website [2] and cut the duplicate samples off.

PPI5k dataset is taken from the literature [8]. The Protein–Protein Interaction Knowledge Base STRING [39] labels the interactions between proteins with occurrence probabilities. PPI5k is a subset of STRING, a denser graph with fewer entities, but more relation facts than CN15k, containing 271,666 uncertain relation facts for 4,999 entities and 7 relations.

Table 1 shows the overview of these two datasets, where *Avg(c)* and *Std(c)* are the average and standard deviation of the confidence values, respectively. *Ratio* is the number of facts to the total number of entities and relations. Both datasets have been cut into three parts at the confidence learning step, 80% for training, 10% for validation, 10% for testing. For evaluating the ability to handle negative triples, which are also generated by corrupting the object of each fact in test datasets.

5.3. Setup

At the **pre-training** step, regarding each entity and relation as a word and each fact as a sentence, we implement a CBOW/skip-gram model trained by Word2Vec of the natural language processing library Gensim [40] with windows 2 (maximum distance between the current and predicted word within a sentence), dimensionality of the word vectors 64,100,128 or 200 according to the needs of the next step, ignores words with total frequency lower than 1, negative sampling number 5, initial learning rate of 0.025, iteration epochs 10, all entities and relations as the vocabulary, and all facts as the corpus.

At the **confidence learning** step, inputting word vectors of each fact encoded at the first step into the LSTM model with original confidence as labels, outputting a confidence score between 0 and 1, we train LSTM with epochs 100, batch size 64, mean squared error loss function, Adam optimizer, for which we set the exponential decay rates $\beta_1 = 0.9$ and $\beta_2 = 0.99$, and negative triples generated by corrupting the object of each fact.

Regarding hyper-parameters, we select all combinations from the following sets of values: embedding dimensionality $ed \in \{64, 100, 128, 200\}$, batch size $bs \in \{64, 128, 256, 512\}$, and word2vec strategy $ws \in \{skip-gram, CBOW\}$. LSTM training was stopped by using an early-stop strategy on the validation set. The best combination on PPI5k is $\{ed = 100, bs = 64, ws = CBOW\}$, and on CN15K it is $\{ed = 128, bs = 128, ws = skip-gram\}$, according to our experience. We report results based on their best hyper-parameter settings.

We hypothesized that different word2vec strategies lead to different confidence prediction effectiveness because distinct datasets have different ratios of facts (samples or triples) to nodes (entities and relations). As for PPI5k, the ratio equals about 54.3, which gets more samples to train vectors from their context. For CN15k, the ratio is only 15.6, which is more suitable for forecasting context from the node. Table 1 shows the ratios in the last column. The more facts in the knowledge base, the more contexts that can be trained, the more suitable word2vec strategy is CBOW. Moreover, we assume that the critical value of this ratio is above 50.0, but this needs further experimental verification.

5.4. Runtime

At the first step of the model, wording embedding performs very fast and effective enough. Same as the majority of existing literature, hits@1, hits@3, and hits@10 of relation facts without confidence in test datasets are shown in Table 2, where Hits@X is calculated with substituting the object of each fact in the testing dataset for all different entities in the corpus.

Although there is nothing so reasonable for evaluating hits@X in UKGE, the first step of the approach shows us the results are plausible that pre-trained embedding vectors already contain much structural semantical information. The hits@X is not so bad compared to the state of the art (results are compared in Table 3 with [36]), however the time cost of performance is much lower than others. For dimension 100 embedding vectors on PPI5k dataset, the model just takes 13.00 s on a laptop computer without a GPU, which trained just 10 epochs with 249,946 facts and a corpus size of 5,006. The handling rate of 17,911 triples per second is also very impressive, and the best rate reported in [36] is 10,887 triples per second. The results mean that the approach achieves a better tradeoff between higher efficiency and more accuracy of knowledge graph embedding; and that it can be used to deal with large-size knowledge bases expressed in KG or UKG.

As for confidence scoring of relation facts in the second phase of the approach, we learn them utilizing the LSTM neural network, which is the main time-consuming part of the approach. Table 4 shows the performance details of LSTM trained through early stopping strategy with monitor *validation loss*, minimal delta 0.0001 and patience 10, where training samples include half of the positive samples and half of the negative samples, *Avg. loss* (%) is the mean squared error of losses on the validated dataset. We can see that good results are obtained in a short time. On the CN15k dataset, the average loss of 7.71, which is comparable to the best result (8.61 of UKGE_{rect} and 7.8 of the latest BEUrRE in Table 5), only needs 19 training epochs. On the PPI5k dataset, we got the average loss of 1.239 after 29 training epochs. It is also known from the Table 4 that with the increase in training times, better results can be obtained. The performance rate (triples/s) was also reported. As the dimension increases, the processing rate decreases, but the precision will also be improved. So the trade-off between efficiency and accuracy is always a problem worth considering in practice KG-based applications. When adapting the hyper-

[2] https://github.com/stasl0217/UKGE

Table 1
Datasets overview.

Dataset	CN15k	PP15k
entities	15,000	4,999
relations	36	7
facts	234,675	271,666
Avg(c)	0.627	0.415
Std(c)	0.234	0.213
ratio	15.6	54.3

Table 2
Runtime(Rt.) and accuracy for embedding relation facts without confidence in KG and hits@X rate value (%), Dimen. denotes dimensionality, Triples/s is the processing rate.

Dataset	CN15k			PPI5k		
Dimen.	64	100	128	64	100	128
Rt. (s)	14.75	15.18	16.35	12.76	13.00	13.79
Triples/s	14,611	14,197	13,180	18,640	**17,911**	16,669
hits@1	0.61	0.56	0.63	2.689	2.767	2.5
hits@3	1.03	1.01	1.1	5.18	5.34	4.953
hits@10	2.20	2.23	2.31	12.58	12.61	12.38

Table 3
Filtered hits@X values (%) for the query $(subject, predicate, ?object)$ on PPI5k using several different strategies.

Approaches	hits@1	hits@3	hits@10
LSTM + corrupted	3.84	9.79	19.23
LSTM + random	1.39	4.89	10.49
The first step of UKG$_s$E	**2.767**	**5.34**	**12.61**

Table 4
Performance details of learning confidence of relation facts.

Dataset	CN15k			PPI5k		
Num. of triples		443,730			499,892	
Num. of vectors		15,036			5,006	
Dimensionality	64	100	128	64	100	128
Stop@epoch	12	18	**19**	22	22	**29**
Runtime(s)	1,646	3,499	4,639	2,640	4,211	6,754
Rate(Triples/s)	3,253	2,283	1,930	3,849	2,413	1,983
Avg. loss	8.473	7.86	**7.71**	1.454	1.395	**1.239**

Table 5
Mean squared error (MSE) and mean absolute error (MAE) of relation facts confidence predication (%) on testing dataset with 21,720 triples on PPI5k and 19,293 on CN15k.

Dataset	CN15k			PPI5k		
Metrics	MSE	MAE	epochs	MSE	MAE	epochs
URGE	10.32	22.72	-	1.44	6.00	-
UKGE$_{rect}$	8.61	19.90	>100	0.95	3.79	>100
UKGE$_{logi}$	9.86	20.74	>100	0.96	4.07	>100
BEUrRE	7.8	20.03	-	-	-	-
UKG$_s$E	**7.71**	21.34	19	**0.98**	5.98	80

parameters of early-stopping training, we can train more epochs and get more accuracy, but consume more performance time.

The recent work [36] has handled with deterministic KG embedding without confidence scoring, so we compare with the reported results in [8,16] in the next subsection.

5.5. Confidence prediction

Evaluating confidence prediction with the mean squared error (MSE) and the mean absolute error (MAE), we perform the model on testing datasets with unseen relation facts. UKG$_s$E gets the best MSE value of 7.71 over 19,293 testing samples on

Table 6
Mean NDCG for fact ranking over the CN15k dataset.

model	linear gain	exponential gain
URGE	0.572	0.570
$UKGE_{rect}$	0.773	0.775
$UKGE_{logi}$	0.789	0.788
BEUrRE	0.796	0.795
UKG_sE	0.780	**0.795**

Table 7
Predicting objects of a query $(s, p, ?o)$ over dataset CN15k in Example 1.

subject	predicate	true objects	confidence	predicted objects	predicted confidence	true confidence
rush	relatedto	fast	0.968	**fast**	**0.709**	0.968
		run	0.709	**run**	**0.701**	0.709
		rapid	0.709	action	**0.676**	0.659
		pass	0.709	time	0.669	0.105
		act	0.709	pass	0.661	0.709
hotel	usedfor	sleeping	1.0	**sleeping**	**0.856**	1.0
		rest	0.984	**rest**	**0.761**	0.984
		sleep	0.893	have sex	**0.694**	0.709
		have sex	0.709	sleep in	0.689	0.709
		sleep in	0.709	sleep	0.637	0.893

the CN15k dataset after 19 epochs of training with 128 dimensionality embedding vectors, and gets the third-best MSE (with 0.03 difference to the best) 0.98 over 21,720 unseen test samples on the PPI5k dataset after 80 epochs of training with 100 dimensionality embedding vectors. The results are shown in the Table 5 compared with URGE, $UKGE_{rect}$ and $UKGE_{rect}$ reported in [8], and BEUrRE reported in [16].

On the CN15k dataset, the model gets better results both on MSE and MAE metrics. To get the best results of $UKGE_{rect}$ and $UKGE_{logi}$, we train the model for more than 100 epochs by running the programs they provided, while UKG_sE get smaller MSE values on stopping train after epochs 19 and 80 for two datasets respectively. The UKG_sE is a better tradeoff between accuracy and time consumed.

Table 6 shows the mean NDCG over 19,293 test queries of CN15k for several compared models. It observes that UKGE ($UKGE_{rect}$ and $UKGE_{logi}$) has generally better performance than URGE, as URGE only considers the node proximity in the graph with no relations and only generates node embeddings. $UKGE_{logi}$ is better than $UKGE_{rect}$ because that $UKGE_{logi}$ can handle the transitivity of synonym relations in the knowledge graph through injecting probabilistic soft logic rules, over which the order of the observed relation facts can be preserved in the embedding space to a certain degree. We achieve a value of the mean NDCG with a linear gain that is very close to that of the $UKGE_{logi}$ model and the BEUrRE model, and a value of the mean NDCG with an exponential gain is equal to that of the BEUrRE model, the best one. Furthermore, the results are carried out very quickly and do not need strong computing power. So, we hypothesized that the order in a triple (lexical relation) and the order between triples (syntactic relation) are partially preserved in the embedding space when treating the triple as a sentence and handling them by NLP technologies. And, we explain why this approach works effectively as follows:

- The embedding vector trained by Word2Vec can preserve the hidden structural-semantic information when treating a triple as a short sentence. The simpler the sentence structure is, the easier and more features of the structure are encoded. Here the short triple-sentence is a three-word sentence (*subject, predicate, object*), which is simple enough.
- Pre-training is indeed very important and effective for uncertain knowledge graph embedding. Better pre-trained embedding vectors get a better confidence prediction. Moreover, high-quality pre-training can greatly improve the convergence speed of confidence learning.
- It is better to consider a relation fact as a whole (a sentence) than to train it with $head + relation \approx tail$ as the objective function.
- The step-by-step processing method is also more practical when dealing with large-scale uncertain knowledge graph embedding.

6. Approximate inferring and case studies

Inferring tasks are the core of knowledge graph representation and natural language understanding. Approximate inference utilizing confidence predicting based on neural networks provides a promising method for some uncertain knowledge graph tasks, such as unseen facts predicting, knowledge base completing, knowledge discovering, knowledge integrating and fusing, and so on. In this section, several approximate inferring examples based on the confidence prediction by UKG_sE are performed and analyzed, which are predicting entities of a given incomplete fact, transitivity inferring and composition

Table 8
Transitivity inferring based on confidence prediction in Example 2.

triple	relation	pred. conf.	true conf.
fork at. kitchen	at.	0.834	1.0
kitchen at. apartment	at.	0.840	0.709
fork ?p apartment	**at.**	**0.814**	-
	locatednear	0.810	-
	hasa	0.750	-

Notes: at. is the abbreviation of atlocation, *pred.* predicted and *conf.* confidence.

inferring over the knowledge graph, as well as classical knowledge reasoning based on propositional logic rules (so-called probabilistic soft logic).

Example 1. The top 5 predicted objects and corresponding original objects are shown in the Table 7, given a query $(s, p, ?o)$. Predicted objects are sorted by their score in descending order. This prediction result is more accurate than that in literature [8]. Firstly, the predicted top one object of the two examples is both true, but in [8] just one example is correct; furthermore, the UKG$_s$E correctly predicts the top two objects for both queries, while the literature has just reported the top one for one query. Secondly, the confidence score of the top one is higher than that in [8], *(0.709>0.703, 0.856>0.849)*. Thirdly, it is most important that we preserve the order of the first three objects, which is not found in [8]. The true ranking of objects for the query *(rush, relatedto, ?objects)* is *fast (0.968)* $\succ$ *run (0.709)* $\succ$ *action (0.659)*, and the predicting ranking of that is the same order, *fast (0.709)* $\succ$ *run (0.701)* $\succ$ *action (0.676)*. This indicates that the ability of knowledge-discovering by the UKG$_s$E model is stronger than that of the previous models, and it also has certain inferring abilities in the embedding space.

Example 2. For transitivity inferring, $(A, R, B)\Lambda(B, R, C) \rightarrow (A, R, C)$, the query $(A, ?p, C)$ on the relation R should gain the highest confidence score for all relations of the knowledge base, given triples (A, R, B) and (B, R, C) in the knowledge graph. For example, *(fork, atlocation, kitchen)* and *(kitchen, atlocation, apartment)* imply *(fork, atlocation, apartment)* with the highest confidence score among all relationships between *fork* and *apartment* in KG. Table 8 shows the result, where "-" means there is not a triple in the knowledge base. This is also effective for synonym inference.

Example 3. For composition inferring, $(A, R_1, B)\Lambda(B, R_2, C) \rightarrow (A, R_3, C), R_3$ should be the first top relation of ranking results of the query $(A, ?p, C)$. Predicting *(fork, isa, tool, 0.917)* and *(tool, synonym, use, 0.785)*, we execute the query *(fork, ?p, use)*, and gain the list (only top 3 are listed) of R_3 (*[usedfor:0.907, causes:0.872, hasprerequisite:0.857]*), so the result of inferring should be *(fork, usedfor, use)* with confidence 0.907, which means fork is something that can be used. These inferences are more in line with people's common sense. Table 9 shows these results.

Example 4. Considering probabilistic logic as in [22]: let a, b be two grounded triples, and let the truth values (or confidence scores) of a and b are denoted by $P(a)$ and $P(b)$ respectively. Modelling the truth values of negation, conjunction, disjunction, and implication of a and b as follows:

$$P(\neq ga) = 1 - P(a),$$

$$P(a\Lambda b) = P(a) \cdot P(b),$$

$$P(a \vee b) = P(a) + P(b) - P(a) \cdot P(b),$$

$$P(a \rightarrow b) \Longleftrightarrow P(a) \leqslant P(b),$$

We can make general inferences in some sense. For example, if *(fork, atlocation, kitchen,* c_1*)*, *(fork, isa, tool,* c_2*)* and *(tool, madeof, metal,* c_3*)* can imply *(kitchen, hasa, metal,* c_4*)*, the model should predict confidence scores meeting $c_1 \cdot c_2 \cdot c_3 \leqslant c_4$. Table 10 shows the calculated results.

These examples demonstrated above show that uncertainty inferring based on the predicting confidence of triples that occurred in the knowledge graph is feasible and meaningful. We hypothesize that when treating triples as sentences, the embedding vectors trained by NLP technologies can inject more hidden structural-semantic information into the embedding space. Moreover, the simpler the sentence structure is, the easier and the more features of the structure are preserved. We also argue that pre-training is of particular relevance and effectiveness to uncertain knowledge graph embedding. Better pre-trained embedding vectors get a better confidence prediction. Furthermore, high-quality pre-training can dramatically improve the convergence speed of confidence learning.

Table 9
Transitivity inferring based on confidence prediction in Example 3.

triple	relation	predicted conference
fork isa tool	isa	0.917
tool synonym use	synonym	0.785
fork 2p use	**usedfor**	**0.907**
	causes	0.872
	hasprerequisite	0.857

Table 10
Probabilistic logic inferring based on confidence prediction in Example 4.

	triples	predicted confidence
	$t_1 = (fork, atlocation, kitchen)$	$c_1 = 0.834$
	$t_2 = (fork, isa, tool)$	$c_2 = 0.917$
	$t_3 = (tool, madeof, metal)$	$c_3 = 0.747$
	$t_4 = (kitchen, hasa, metal)$	$c_4 = \mathbf{0.749}$
premise	$t_1 \Lambda t_2 \Lambda t_3$	$c_1 \cdot c_2 \cdot c_3 = \mathbf{0.571}$
conclusion	$t_1 \Lambda t_2 \Lambda t_3 \rightarrow t_4$	$c_1 \cdot c_2 \cdot c_3 \leqslant c_4$

7. Analysis and discussion

7.1. On pre-training

The two-step pattern of the model is pre-training, which services the next step with generating embedding vectors, and fine-tuning, which is trained by deep neural networks for specific domain tasks, is the same as what BERT does [11]. The pre-training idea spirited by natural language models in the NLP community is proved to be effective for some knowledge-based applications when we look back at the structured knowledge graphs as normal languages, even for knowledge inferring tasks. The pre-training model is promising in the approximate inference utilizing the confidence of uncertain knowledge graphs. Firstly, neural networks can learn more hidden structural semantics of knowledge facts when they are treated as standard short sentences (triples). Moreover, richer semantics could also be drawn when training a group of similar knowledge facts as a whole (such as many triples with the same relation) at one time. Secondly, logic rules could be injected in the first step, where all grounding of the same logic rules are trained as a group, by which we believe the model can learn some hidden features of logic rules. Logic rules can be regarded as high-level structured semantics of knowledge graphs in a sense, so we call the injection of logic rules this way *the local structural injection* into an embedding space. Thirdly, the global structural semantics injection can be considered when the whole knowledge base is treated as a structured document including many sentences.

7.2. On confidence learning based on RNN

Recurrent neural networks, such as LSTM, Attention mechanism, Transformer, and BERT, have abilities of memories, which are the real reasons that some structural-semantic information of knowledge graphs are preserved in the embedding space. What the network remembers is the most likely to keep that in the embedding space. What the network wants to remember depends largely on the specific downstream tasks, and what the network can remember depends completely on the development of deep learning technologies. It is believed that the development of the natural language model will greatly benefit the (un) certain knowledge graph embedding, such as BERT, which should help preserve the global semantics of knowledge graphs into embedding spaces to improve the ability to handle downstream tasks.

7.3. On explainability

Generally, knowledge graphs are often employed to increase neural networks' explainability because knowledge graphs are open and explainable through relationships with other entities. Concerning knowledge graph embeddings, explainability has a difficult interpretation: knowledge graph embeddings are often vector representations in the vector space, thus losing the original interpretability that comes from logic. In fact, mapping vector space representations with logic rules is intrinsically difficult, and even some rules are impossible to learn with knowledge graph embeddings [41,42]. However, under an uncertain knowledge graph embedding scenario, confidence values can intuitively illustrate some semantics in the embedding space. Example 2 shows that the maximal confidence score of (A, R, C) illuminates that it is the logical result of (A, R, B) and (B, R, C), and so does Example 3.

And Example 4 is more a kind of formula validation than a kind of approximate knowledge inference. Actually, the formula $t_1 \Lambda t_2 \Lambda t_3 \rightarrow t_4$ explains the prediction based on confidence calculating. Many candidates for t_4 could be also used as an

inferential conclusion of $t_1 \Lambda t_2 \Lambda t_3$ as long as their confidence scores are greater than 0.571. Although the interpretability is very weak, it does express or validate that the t_4 is the conclusion of that formula with the least possibility of 0.571.

7.4. On approximate inferring: Probabilistic models vs. Deep network models

One of the biggest difficulties of inferring from structured probabilistic graph models is to calculate the posterior probability, especially when the graph's size is too big, such as in uncertain knowledge graphs. In general, with modelling n discrete variables, each having k values, the computational complexity is often exponential, such as $O(k^n)$. However, the parameter size of the learning confidence score utilizing deep network models only depends on the depth, that is, the scale of the network parameters. On the other hand, deep neural network models need distributed representation as inputs, and the difficulty of that distributed representation is to keep richer semantics. So the two-step approach is the compromise between probabilistic graph models and deep neural network models in a sense. It is that knowledge graphs are encoded into an embedding space with preserved semantics as much as possible, and confidence scores are learned with deep networks to avoid complex calculating of posterior possibilities. There are grounds to believe that the approach is promising for improving the ability to handle some kinds of uncertain knowledge-graph-based tasks.

8. Conclusion and future works

An approximate inferring approach based on confidence prediction of uncertain knowledge graph embedding, UKG_sE, is proposed. After quickly encoding an uncertain knowledge graph into the embedding space with natural language models, it learns the confidence distribution of facts in that knowledge graph utilizing deep neural networks. On the one hand, the model achieves a better tradeoff between performance and accuracy than baselines, of which the MSE accuracy is improved by up to 25.3% and the training time is reduced by more than 5 times. On the other hand, some kinds of uncertain knowledge inferring can be effectively performed grounded on predicted confidence scores, which not only benefits some uncertain knowledge-graph-based tasks but also provides weak explainability for deep neural networks in a sense.

In knowledge graph applications, logic rules are the heart of discovering and applying knowledge, but they are not easily injected into the knowledge graph embedding space. Embedding logic rules and more complex structural-semantic information is still an open problem and is also our research interesting. We will further devote ourselves to combining natural language inferring technologies and formal reasoning technologies into the uncertain knowledge graph embedding to perform complex inferring tasks, such as non-analogical inferring, abductive inferring, and commonsense inferring. Shortly, we will also conduct further experimentation to validate veritably large-size uncertain knowledge bases.

CRediT authorship contribution statement

Shihan Yang: Methodology, Software, Formal analysis. **Weiya Zhang:** Conceptualization, Resources, Writing - original draft. **Rui Tang:** Visualization, Validation, Software. **Mingkai Zhang:** Software, Validation. **Zhensheng Huang:** Data curation, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Thanks reviewers for their valuable comments. This research is partly supported by the Talents Introduction Project of KUST (No.20180025).

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD '08, Association for Computing Machinery, New York, NY, USA, 2008, pp. 1247–1250, https://doi.org/10.1145/1376616.1376746.
[2] T. Rebele, F. Suchanek, J. Hoffart, J. Biega, E. Kuzey, G. Weikum, Yago: A multilingual knowledge base from wikipedia, wordnet, and geonames, in: P. Groth, E. Simperl, A. Gray, M. Sabou, M. Krötzsch, F. Lecue, F. Flöck, Y. Gil (Eds.), The Semantic Web – ISWC 2016, Springer International Publishing, Cham, 2016, pp. 177–185.
[3] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E. Platanios, A. Ritter, M. Samadi, B. Settles, R. Wang, D. Wijaya, A. Gupta, X. Chen, A. Saparov, M. Greaves, J. Welling, Never-ending learning, Commun. ACM 61 (5) (2018) 103–115, https://doi.org/10.1145/3191513.
[4] R. Speer, J. Chin, C. Havasi, Conceptnet 5.5: An open multilingual graph of general knowledge (2018). arXiv:1612.03975.
[5] F. Bianchi, G. Rossiello, L. Costabello, M. Palmonari, P. Minervini, Knowledge Graph Embeddings and Explainable AI (2020) arXiv:2004.14843.
[6] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. de Melo, et al., Knowledge Graphs (2020) abs/2003.02320. arXiv:2003.02320.

[7] M. Nayyeri, C. Xu, J. Lehmann, H. Shariat Yazdi, LogicENN: A Neural Based Knowledge Graphs Embedding Model with Logical Rules (2019) arXiv:1908.07141.
[8] X. Chen, M. Chen, W. Shi, Y. Sun, C. Zaniolo, Embedding uncertain knowledge graphs (2019). arXiv:1811.10667.
[9] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient Estimation of Word Representations in Vector Space (2013) arXiv:1301.3781.
[10] F.A. Gers, J.A. Schmidhuber, F.A. Cummins, Learning to forget: Continual prediction with lstm, Neural Comput. 12 (10) (2000) 2451–2471.
[11] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (2018) arXiv:1810.04805.
[12] S. He, K. Liu, G. Ji, J. Zhao, Learning to represent knowledge graphs with gaussian embedding, CIKM '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 623–632.
[13] J. Hu, R. Cheng, Z. Huang, Y. Fang, S. Luo, On embedding uncertain graphs, CIKM '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 157–166.
[14] L. Vilnis, X. Li, S. Murty, A. McCallum, Probabilistic Embedding of Knowledge Graphs with Box Lattice Measures (2018) arXiv:1805.06627.
[15] A.I. Cowen-Rivers, P. Minervini, T. Rocktaschel, M. Bosnjak, S. Riedel, J. Wang, Neural variational inference for estimating uncertainty in knowledge graph embeddings (2019). arXiv:1906.04985.
[16] X. Chen, M. Boratko, M. Chen, S.S. Dasgupta, X.L. Li, A. McCallum, Probabilistic box embeddings for uncertain knowledge graph reasoning, in: Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Association for Computational Linguistics, 2021, pp. 882–893, https://doi.org/10.18653/v1/2021.naacl-main.68, URL:https://aclanthology.org/2021.naacl-main.68.
[17] L. Vilnis, X. Li, S. Murty, A. McCallum, Probabilistic embedding of knowledge graphs with box lattice measures, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2018, 2018, pp. 263–272.
[18] A. Abujabal, R. Saha Roy, M. Yahya, G. Weikum, Never-ending learning for open-domain question answering over knowledge bases, in: Proceedings of the 2018 World Wide Web Conference, WWW '18, International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 2018, p. 1053–1062. doi:10.1145/3178876.3186004.
[19] C. Schon, S. Siebert, F. Stolzenburg, Using conceptnet to teach common sense to an automated theorem prover, Electronic Proceedings in Theoretical Computer, Science 311 (2019) 19–24, https://doi.org/10.4204/eptcs.311.3.
[20] S. Guo, Q. Wang, L. Wang, B. Wang, L. Guo, Knowledge graph embedding with iterative guidance from soft rules, in: AAAI, 2018.
[21] B. Ding, Q. Wang, B. Wang, L. Guo, Improving knowledge graph embedding using simple constraints, in: ACL, 2018.
[22] M. Nayyeri, C. Xu, J. Lehmann, H.S. Yazdi, Logicenn: A neural based knowledge graphs embedding model with logical rules (2019). arXiv:1908.07141.
[23] X. Chen, S. Jia, Y. Xiang, A review: Knowledge reasoning over knowledge graph, Expert Systems Applications 141 (2020), https://doi.org/10.1016/j.eswa.2019.112948 112948.
[24] T. Dettmers, P. Minervini, P. Stenetorp, S. Riedel, Convolutional 2d knowledge graph embeddings (2018). arXiv:1707.01476.
[25] Z. Wang, L. Li, D.D. Zeng, Y. Chen, Attention-based multi-hop reasoning for knowledge graph, in: 2018 IEEE International Conference on Intelligence and Security Informatics (ISI), 2018, pp. 211–213. doi:10.1109/ISI.2018.8587330.
[26] L. Guo, Q. Zhang, W. Ge, W. Hu, Y. Qu, Dskg: A deep sequential model for knowledge graph completion (2018), ArXiv abs/1810.12582.
[27] W. Xiong, T. Hoang, W.Y. Wang, Deeppath: A reinforcement learning method for knowledge graph reasoning (2018). arXiv:1707.06690.
[28] S. Storks, Q. Gao, J.Y. Chai, Recent advances in natural language inference: A survey of benchmarks, resources, and approaches (2020). arXiv:1904.01172.
[29] L.S. Vannur, L. Nagalapatti, B. Ganesan, H. Patel, Data augmentation for personal knowledge graph population (2020), CoRR abs/2002.10943. arXiv:2002.10943. URL:https://arxiv.org/abs/2002.10943
[30] F. Zhao, H. Sun, L. Jin, H. Jin, Structure-augmented knowledge graph embedding for sparse data with rule learning, Computer Communications 159 (2020) 271–278, https://doi.org/10.1016/j.comcom.2020.05.017, URL:https://www.sciencedirect.com/science/article/pii/S0140366419319310.
[31] G. Li, Z. Sun, L. Qian, Q. Guo, W. Hu, Rule-based data augmentation for knowledge graph embedding, AI Open 2 (2021) 186–196, https://doi.org/10.1016/j.aiopen.2021.09.003, URL:https://www.sciencedirect.com/science/article/pii/S2666651021000267.
[32] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space (2013) (2013). arXiv:1301.3781.
[33] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780, https://doi.org/10.1162/neco.1997.9.8.1735.
[34] M. Sundermeyer, R. Schlüter, H. Ney, LSTM Neural Networks for Language Modeling, in: 13th Annual Conference of the International Speech Communication Association, September 9–13, 2012, Portland, Oregon, Portland, Or., 2012, pp. 194–197.
[35] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, p. 3111–3119.
[36] T. Soru, S. Ruberto, D. Moussallem, A. Valdestilhas, A. Bigerl, E. Marx, D. Esteves, Expeditious generation of knowledge graph embeddings (2018). arXiv:1803.07828.
[37] T.-Y. Liu, Learning to rank for information retrieval, SIGIR '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 904.
[38] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, NIPS'13, Curran Associates Inc., Red Hook, NY, USA, 2013, pp. 2787–2795.
[39] D. Szklarczyk, J.H. Morris, H. Cook, M. Kuhn, S. Wyder, M. Simonovic, A. Santos, N.T. Doncheva, A. Roth, P. Bork, L.J. Jensen, C. von Mering, The STRING database in 2017: quality-controlled protein–protein association networks, made broadly accessible, Nucleic Acids Research 45 (D1) (2016) D362–D368, https://doi.org/10.1093/nar/gkw937.
[40] R. Řehuřek, P. Sojka, Software Framework for Topic Modelling with Large Corpora, ELRA, Valletta, Malta, 2010, pp. 45–50.
[41] S.M. Kazemi, D. Poole, Simple embedding for link prediction in knowledge graphs, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2018, pp. 4289–4300.
[42] V. Gutiérrez-Basulto, S. Schockaert, From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules, in: Sixteenth International Conference on Principles of Knowledge Representation and Reasoning (KR2018), 2018.