Discriminator-based adversarial networks for knowledge graph completion

Abdallah Tubaishat[1] Tehseen Zia[2] · Rehana Faiz[2] · Feras Al Obediat[1] · Babar shah[1] David Windridge[3]

Abstract
Knowledge graphs (KGs) inherently lack reasoning ability which limits their effectiveness for tasks such as question–answering and query expansion. KG embedding (KGE) is a predominant approach where proximity between relations and entities in the embedding space is used for reasoning over KGs. Most existing KGE approaches use structural information of triplets and disregard contextual information which could be crucial to learning long-term relations between entities. Moreover, KGE approaches mostly use discriminative models which require both positive and negative samples to learn a decision boundary. KGs, by contrast, contain only positive samples, necessitating that negative samples are generated by replacing the head/tail of predicates with randomly chosen entities. They are thus usually irrational and easily discriminable from positive samples, which can prevent the learning of sufficiently robust classifiers. To address the shortcomings, we propose to learn contextualized KGE using pre-trained adversarial networks. We assume multi-hop relational paths(mh-RPs) as textual sequences for competitively learning discriminator-based KGE against the negative mh-RP generator. We use a pre-trained ELECTRA model and feed it with relational paths. We employ a generator to corrupt randomly chosen entities with plausible alternatives and a discriminator to predict whether an entity is corrupted or not. We perform experiments on multiple benchmark knowledge graphs, and the results show that our proposed KG-ELECTRA model outperforms BERT in link prediction.

Keywords Knowledge graph completion · Pre-trained language model · Transformer model

Tehseen Zia
tehseen.zia@comsats.edu.pk

Abdallah Tubaishat
abdallah.tubaishat@zu.ac.ae

Rehana Faiz
rehanafaiz8@gmail.com

Feras Al Obediat
feras.al-obeidat@zu.ac.ae

Babar shah
babar.shah@zu.ac.ae

David Windridge
d.windridge@mdx.ac.uk

[1] College of Technological Innovation, Zayed University, Abu Dhabi, UAE

[2] Department of Computer Science, COMSATS University Islamabad, Islamabad, Pakistan

[3] Department of Computer Science, Middlesex University, London, United Kingdom

1 Introduction

Knowledge bases (KBs) such as WordNet [1], Freebase [2] and Yago [3] have become reference resources for various logic-oriented tasks such as query expansion [4], co-reference resolution [5], question answering and information retrieval, etc. Such KBs are typically incomplete (in the classical sense of the term 'knowledge base') in that they lack a reasoning capability, which thus restricts their applicability. This has stimulated research on KB completion methods [6]. Within this context, a number of studies have focused on using representation learning for its ability to model semantic features useful for generalization [7–9]. The goal in these approaches is to represent KB entities and relations using vectors such that similarities between them (proximities, inner-product relations) can be used to make logical inferences.

Most of existing approaches, however, leverage from structure information in available triplets and hence are unable to cope with sparsity of KBs. Further, these approaches learn static representations of entities and relations in different triples [10] and hence are unable to cater different contexts. This is unlike to word embeddings where words are represented differently in different contexts. Furthermore, these approaches rely on given entities and relations or word co-occurrence with entities and hence do not fully utilize the rich syntactic and semantic information available in the large-scale text data [10, 11]. To address the issues, researchers recently use pre-trained language models (i.e., transformers) and learn KB representations as a downstream task. While this approach has achieved some impressive results, the model is computationally expensive to fine-tune [12] due to its masked language modeling-based training procedure. To address that, we use a pre-trained discriminative model called ELECTRA, [13] to learn downstream KB representations. Unlike predicting masked entities and relations, this discriminative model uses a computationally efficient procedure [13] token replacement detection which learn KB representations by detecting replaced tokens (i.e., entities and relations) in relational paths. Typically, as KBs consist of only positive samples, negative samples are generated synthetically by replacing the head or tail of a predicate with a randomly chosen entity in order to learn discriminative models. Such negative samples, however, tend to be easily discriminable from positive samples, leading to fragile KB representations. In [14], generative adversarial networks (GANs) have been shown to be useful for generating negative samples from positive samples, rather than composing random triplets.

As generative and discriminative modeling approach representation learning from very different perspectives, it is possible to take a broader perspective on representation learning by reciprocally allowing one modeling paradigm to be guided by the other [14]. Embodying this principle, generative adversarial networks (GANs) [15] have emerged as powerful framework within which generator and discriminator play a game-theoretic, 2-player min-max game [16]. Such networks have displayed notable capability in image generation, sequence generation, domain adaptation and information retrieval. Relevant here, GANs have been shown to be useful for generating negative samples from positive samples [14]. While the approach of [14] uses GANs to generate negative samples only, the current study will exclusively focus on the use of GANs to train discriminator for KB completion tasks. Thus, while [14] is a framework for modeling distributions over negative samples, we are here concerned with using pre-trained discriminator for downstream KB completion, in order to achieve a model for extensible logical reasoning and the querying of hypotheticals. In previous work [17], we use GANs to generate relational paths for reasoning over KBs. However, while the approach [17] use a generator to compose a relational path and discriminator to train generator by discriminating between positive and negative paths, this study aims to use generator to generate negative samples and discriminator to learn KB representations.

Stimulated by the GAN concept [15], we thus propose Knowledge Completion Discriminator-based Adversarial Networks (KCGAN) [17], a novel framework leveraging both generative- and discriminative-based approaches. Specifically, the KCGAN architecture seeks to learn two models: (1) A Generator G, which takes a masked relational path and predicts masked entities and relations to complete the path. (2) A Discriminator D, which aims to detect replaced entities and relations in the generated relational path. In the proposed KCGAN model, G and D contest each other as follows: G attempts to fool D by generating an indistinguishable invalid relational path through completion of the masked entities and relations, while D attempts to discriminate between original and replaced relations. This contention results in both models improving as the game progresses, until a convergence point is reached in which the generator is (to the discriminator) indistinguishable from the true relational distribution and the discriminator is maximally effective at distinguishing between original and replaced entities and relations. Novelty of the work is twofolded as follows:

1. We propose a novel token replacement detection procedure for KB completion called KG-ELECTRA
2. We use a discriminator-based language model for downstream KB completion
3. We perform extensive experiments on WN18-SQuAD, WN18RR-SQuAD and FB15K-SQuAD.

2 Related work

The completion of KGs remained an interesting research area in the past few years. Many different approaches have been developed in these studies; the main idea behind all these studies is to make KGs machine readable and apply different reasoning techniques to them. The translational distance approach (TransE) [6] is one of the initial approaches used for KG completion. This approach is used to convert KGs into vectors. This approach introduced simple and basic ways for KG reasoning and gives the scores to triples according to the distance between them [6]. Although TransE provides bases for KG reasoning, this approach also has many limitations, for example, it cannot be used for large-scale KGs and leads to large dimensional vectors conversion; it is limited to only 1-hop relations and

cannot be used for multi-hop relational paths (mh-RPs). TransG [18] is the extension model of TransE that can perform KG embedding for mh-RP still it is limited to use only structural information of triplets.

Semantic matching is another approach introduced for KG completion uses a similarity-based scoring function for triples plausibility, for example, the DistMult model [7]. DistMult-HRS [19] is the extension model of DistMult that introduced hierarchical relation information for KG completion. However, these approaches are limited to using only structural information of triplets and cannot incorporate external information for better KG reasoning. Many convolutional neural network-based approaches have been used for KGC [20–23], also limited to the use of structural knowledge only. Although all these approaches show promising results for KGC, these approaches lack the ability of intelligent reasoning over KG because they only used structured information about triples. To improve the KG reasoning, many approaches incorporated external textual data like entity types and relations descriptions [9]. However, these approaches learn the same embedding for entity/relation pairs in different triples and do not incorporate the weight and meaning of descriptions in a textual representation, for example, for triple 1(*Ali, lives in, Islamabad*) and for triple 2 (*Ali, wrote, a book*). These approaches give the same weights to "Ali" irrespective of the given relations and external information.

2.1 Adversarial models

Many research studies trained multiple networks in adversarial way like Generative Adversarial Networks (GANs) [15]. These studies introduced different frameworks that consist of a generator and a discriminator. The generator aimed to generate samples just like input samples. And the discriminator is aimed to predict whether the generated samples are real or fake (generated sample). These approaches focused on designing better generators that can fool the discriminator. For example, Knowledge Base GAN (KBGAN) [14] is a framework that trains already introduced KG embedding models in an adversarial manner. KB-GAN uses translational embedding models as generators and semantic matching models as discriminators. This framework generates negative triplet samples to improve the training of existing semantic matching-based KG embedding models or discriminators. Here, the generator takes 20 different negative triples as input (real samples); then, it ranks these triplets and generates a negative triple (fake sample) similar to a most probable input triple while the discriminator has to discriminate between real and fake triple. This approach only focuses on fooling the discriminator and limits the strength of the generator by limiting the amount of input sample. GenKGC[24] is also a generative approach that tries to complete KG by generating a sequential task trough pre-trained model.

Another generative adversarial approach is called Knowledge Completion GAN (KCGAN) [17]. This approach also trains a generator and a discriminator. It trains a generator that generates mh-RPs from a given entity/relation pair and a discriminator that discriminates the generated samples from real samples. This model acts as a game between generator and discriminator because the generator tries to generate samples similar to the input samples so that discriminator cannot discriminate it from the original samples while the discriminator tries to discriminate samples and give rewards accordingly to the generator. Although this approach performs better by using mh-RPs, this approach also has some limitations. This model focuses on training a better generator rather than a better discriminator.

2.2 Pre-trained models

To overcome the previously mentioned issues, many new studies introduced pre-training approaches for KG completion, for example, KG-BERT [12]. This approach fine-tuned a pre-trained language model called BERT [25] for KG completion tasks like link prediction, relation prediction and triples classification. BERT is a general-purpose language representation learning model that can be fine-tuned for multiple downstream tasks. It is a transformer-based model that trains transformer encoders in two stages of pre-training and fine-tuning. BERT [25] generated state-of-the-art results for different representation learning tasks. However, BERT is for KG completion tasks like link prediction, relation prediction and triples classification. BERT is a general-purpose language representation learning model that can be fine-tuned for multiple downstream tasks. It is a transformer-based model that trains transformer encoders in two stages pre-training and fine-tuning. BERT generated state-of-the-art results for different representation learning tasks. However, BERT [25] model is a deep bidirectional model; therefore, it utilizes a large amount of computation to perform. While ELECTRA [13] is computationally efficient for example, an ELECTRA-small model outperforms the comparable BERT-small by 5 points on GLUE [26] by training on single GPU and utilizing $1/20_{th}$ the parameters and $1/135_{th}$ the pre-training compute of a transformer-based BERT-large. ELECTRA [13] model shows comparative results with much less computation and outperforms BERT when given equal computational resources.

Hence, the BERT model cannot be used for simple downstream tasks due to the expense of large computation requirements. As KG-BERT [12] fine-tuned the BERT

model for KG completion, therefore, it also consumed large computational resources to perform. This model performs many tasks of KG completion as link prediction, relation prediction and triples classification. Triples classification and relation prediction are simple tasks as compared to link prediction. Because link prediction is a very time and computation-consuming task. Therefore, fine-tuning BERT [25] for these tasks requires less computation as compared to the link prediction. For link prediction, KG-BERT at first replaced each entity of a triplet with every other entity present in the KG and then ranked according to the plausibility of a triplet. It, later on, calculates the probability of plausibility for each triplet, and the most probable triplet is predicted as output. However, BERT is a costly model that requires a large number of parameters along with huge computational resources to perform link prediction.

3 Method

3.1 Generator-based adversarial network for knowledge graph completion

Previously, we used generator-based adversarial network for KGC [17]. We posed KGC as a relation path (RP) generation problem and adopted an auto-regressive modeling approach to learn to generate RPs [27]. Thus, given a dataset of N relation paths $D = P_{(1)}, \ldots P_{(N)}$, where a RP $P(h,t) = (h, r_1, \ldots, r_L, t)$ connects pairs of entities h and t through L relations. We trained generator G to compose P and use the corresponding discriminator D to process the generated paths and provide supervision to the generator. To model G, we employed a Recurrent Neural Network (RNN) to greedily selects an entity and relation at each time step and produces an output entity; relations are kept within the input space and entities are embedded in the hidden latent space. This model designed by modifying the RNN's recursive function as follows:

$$\hat{v}_{e_l} = (W[\hat{v}_{e_{l-1}}; v_{r_l}]) \tag{1}$$

$$v_{e_l} = \text{softmax}\hat{v}_{e_l} \tag{2}$$

where v_{e_l} and $\hat{v}_{e_l}$, respectively, denote the predefined and modeled representation of entity e at position l, and v_{r_l} is the given vector representation of relation r. To initialize the model, we set $\hat{v}_{e_l} = v_h$.

To model D, we used a Convolutional Neural Network (CNN) where we perform convolution to achieve a feature map v^i. We thus represent a relation path $p_{1:L} = (h, r_1, \ldots, t$ as:

$$\epsilon_{1:T} = (v_h * v_{r1} *, \ldots, * v_t) \tag{3}$$

where * is a concatenation operator applied in building a matrix $\epsilon_{1:T}$. The convolution is performed by applying a filter $\omega \epsilon R^l \cdot k$ to a window of l tokens in order to produce a feature map v_i as:

$$v_i = g(\omega \circledast \epsilon_{i:i+l-1} + b) \tag{4}$$

The convolution is followed by max-over-time pooling over feature maps as:

$$\hat{v} = \max v_1, v_2, ..., v_{T-l+1} \tag{5}$$

The pooling layer connects to a fully connected (FC) layer and finally to a sigmoid unit to produce the inferred probability of the relation path being real.

Here the generator is used to generate mh-RPs while the discriminator is used to distinguish between real and generated paths. The model (*path-KCGAN* [27]) improved KGC as compared to the traditional KGE techniques. However, recent studies [13] show that a discriminator-based the approach performs better than a generator-based approach for language modeling. Therefore, in this study, we aim to use a discriminator-based adversarial network to further improve the efficacy of adversarial networks for KGC.

3.2 Discriminator-based adversarial network for knowledge graph completion

In this study, we aim to improve KGC adversarial networks by leveraging rich contextual information available in the related text datasets. We also aim to analyze the efficacy of discriminator-based KGC adversarial network as compared to our previous study on generator-based model [27]. To achieve these goals, we build our model using an adversarial network pre-trained on a text dataset. We introduce pre-trained adversarial network in Sect. 3.2.1 and proposed methodology in Sect. 3.2.2.

3.2.1 Efficiently learning an encoder that classifies token replacements accurately (ELECTRA)

ELECTRA is a transformer-based pre-train model while it trains the transformer encoders as discriminators rather than generators as in BERT [25]. The generator in ELECTRA generates the samples similar to the original input samples while the BERT model [25] generates the missing tokens [MASK] in the given input. Therefore, BERT requires a large and complex network for generating original text for these masked tokens. Along with masked token generation, BERT also performs the task of next sentence prediction in which it predicts the relation

between two sentences by pointing out whether the next sentence is related to the previous sentence or not. Therefore, BERT [25] model requires a large amount of time and computation to perform. However, the ELECTRA model [13] corrupts the input by replacing some of the input tokens (15%) with tokens generated by a generator and then discriminates between the replaced and original tokens by a discriminator. ELECTRA uses a lightweight generator to train the discriminator in the pre-training phase. After training, the generator got discarded while the discriminator retains for inference. As ELECTRA [13] is a lightweight language model as compared to BERT [25] and requires less computation even in fine-tuning, here we employ the ELECTRA model for KGC therefore, we named it Knowledge Graph ELECTRA (KG-ELECTRA). This model performs competitively with KG-BERT [12] with a lesser number of parameters and computational resources.

3.2.2 ELECTRA for knowledge graph completion (KG-ELECTRA)

We fine-tune the ELECTRA model [13] and use it's the original classification and question-answering output layers for our triples classification task and link prediction task, respectively. The architecture of KG-ELECTRA that models triples classification called KG-ELECTRA (a) as shown in Fig. 1. The architecture of KG-ELECTRA (a) is similar to the ELECTRA [13] except for the input layer. During the fine-tuning of ELECTRA for triples classification, we represent the sequences of triples (h, r, t) as a combination of the head, relation and tail sequences rather than single sequence sentence as in ELECTRA. Each model input the sequence starts with [*CLS*] which is a special classification token. The head, relation and tail sequences are separated by a special token [*SEP*]. The head entity tokens are represented as $Tok_{h1}, \ldots, Tok_{hn}$ while relation and tail entity tokens are represented as $Tok_{r1}, \ldots, Tok_{rn}$ and $Tok_{t1}, \ldots, Tok_{tn}$, respectively.

The input representation of each token is constructed by adding the corresponding token embedding, its segment embedding and its positional embedding. All the tokens in the head/tail sentence have the same segment embedding E_A while all relation sentence tokens have the same segment embedding E_B. The position embedding of all tokens remains same in a single position i that could be any one from 1, 2, 3, ..., 512. E_i is the embedded input representation of each input token I, and these inputs are fed into the KG-ELECTRA (a). The output layer of the model shows output in the form of a label, i.e., label 1 for true triplet and 0 for false triplet. The final hidden vectors RH represent output as C and T_i that corresponds to special token [*CLS*] and i_{th} input token, respectively. Here H is the size of hidden states in the pre-trained ELECTRA model [13]. For computing triples classification score, the final hidden vector C is used as the aggregate sequence representation. In this fine-tuning, the only new parameters were introduced in the form of weights for classification layer W that belongs to R^{2H}. We use cross-entropy loss for triples classification and updated it via gradient descent.

The score function of the triples classification task gives a score to the most plausible triple. The most plausible triple has a score of 1 or equal to one while the least plausible triple gets a zero or almost zero scores. The score function for this task is given in Eq. (6) where g is a sigmoid non-linear function, C is the aggregate sequence representation while W^T is the weight matrix of hidden layers.

$$f_{\text{Classification}} = g(CW^T) \tag{6}$$

The architecture of our KG-ELECTRA for link prediction is known as KG-ELECTRA (b) shown in Fig. 2. In this architecture of our model, both input and output layers are the same as the layers in the ELECTRA model for question answering. In this task, the input the sequence is represented as the combination of questions (i.e., head entity/relation pair) and $mh - RP$ context passage. Here, the entity/relation pair use segment embedding E_A while $mh - RP$ use E_B. In fine-tuning, two new vectors S(start-span vector) and E(end-span vector) are introduced that predicts the start–position and end position of tail entity (answer) from the given $mh - RP$(these vectors were not present during pre-training). The objective of this task is to calculate the correct sum of the start and end positions of the answer. The correct sum of start/end span vectors is considered matched answer while the wrong sum is considered the mismatched answer. The output for link prediction is further illustrated by different results given in Table 1.

The table shows two examples from our designed dataset WN18RR-SQuAD. The first column of the table contains $mh - RP$ the paragraph that is generated from a question (entity/relation pair) by appending a triple starting with the tail entity of question and so on. The second column contains a question for which the model has to predict the answer from the paragraph. The third column contains the answer predicted by the model. The first row of Table 1 represents an example for predicting an answer for directly related triples while the second row represents an example for predicting the answer of indirectly related triples (further explained in ablation study experiments of indirect relations). The score function of the link prediction task was used to measure the sum of the start position and end position of the answer in the paragraph. The most probable answer gets the high score, i.e., accurate sum of

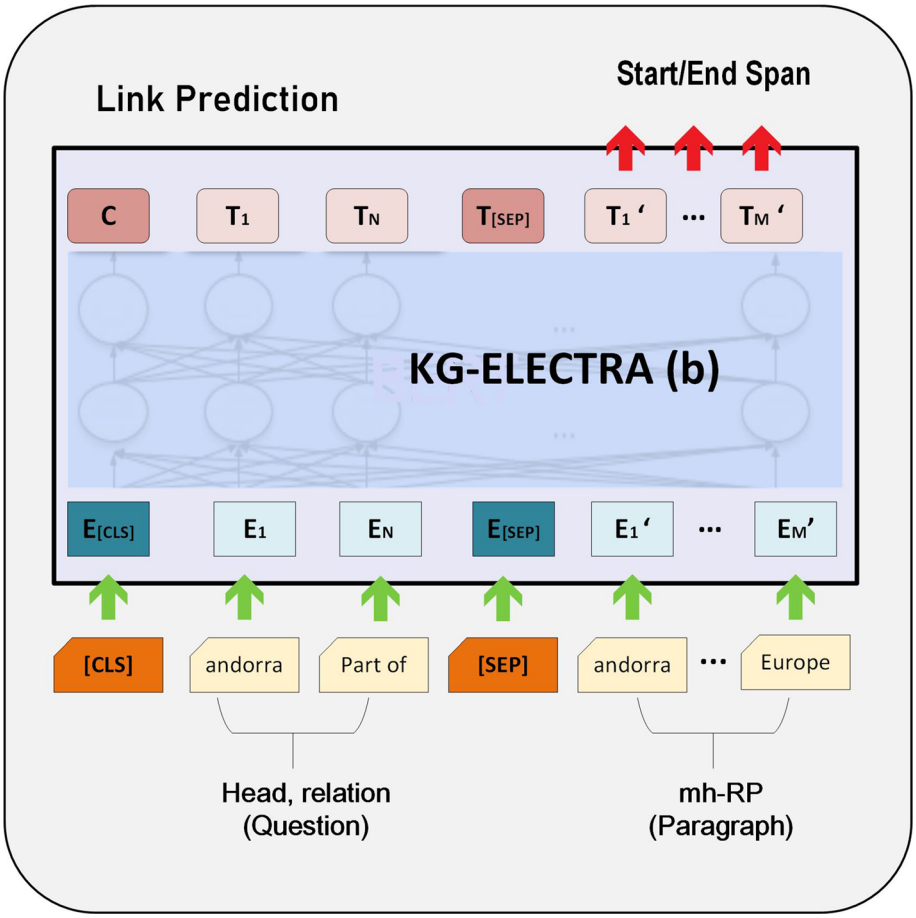

Fig. 1 Illustration of KG-ELECTRA fine-tuning for triples classification task (KG-ELECTRA (a)). The motivation for this architecture is the KG-BERT model [12]. The model input sequence starts with a special token [*CLS*]. The head entity tokens *('Ali', 'Faiza')*, relation tokens *('gender', 'gender')* and tail entity tokens *('male', 'male')* are separated by [*SEP*] token. *C* is the final hidden vector that is used for triples label generation

start and end positions. The score function for the link prediction task is given in Eq. (7) where S represents the start vector, E represents the end vector and T_i represents the embedded token at i_{th} position.

$$f_{Link-Prediction} = S \cdot T_i + E \cdot T_i \tag{7}$$

3.3 Data preparation for link prediction

We designed new datasets for the task of link prediction.[1] We prepared these datasets by using the triples of Wordnet [1] and Freebase [2] datasets. We used these triples to prepare the *SQuADV*1.1 [28] like datasets named WN18-SQuAD, WN18RR-SQuAD and FB15K-SQuAD. For the preparation of these datasets, first, we selected any random triple from the original KGs (i.e. WN18, WN18RR and FB15K) and then appended the triple that starts with the tail entity of chosen triple and so on appended up to 25 triples. These multihop triples of specific length form the paragraph while any head/relation (h, r) pair from that paragraph act as the question, and its tail entity would be the answer that needs to be predicted. The link prediction task predicts the true tail entity (answer) from the paragraph by providing a multi-hop relational path as paragraph and h, r pair as the question. The flowchart of the data preparation for the link prediction task is given in Fig. 3.

4 Experiments

For the evaluation of our proposed model KG-ELECTRA, we performed different experiments for link prediction and triples classification. Through our different experiments, we tried to determine the following concepts related to knowledge graphs (KGs):

- Can KG-ELECTRA determine that the given unseen triple is either true or false?
- Can KG-ELECTRA locate the correct entity for a given pair of entity/relation, from the given $(mh - RP)$?

4.1 Datasets

We used different benchmark KGs for our experiments. We used WN11 [1] and FB13 [2] datasets for the triples classification task. For the task of link prediction, we

[1] https://github.com/RehanaFaiz/KG-Electra.

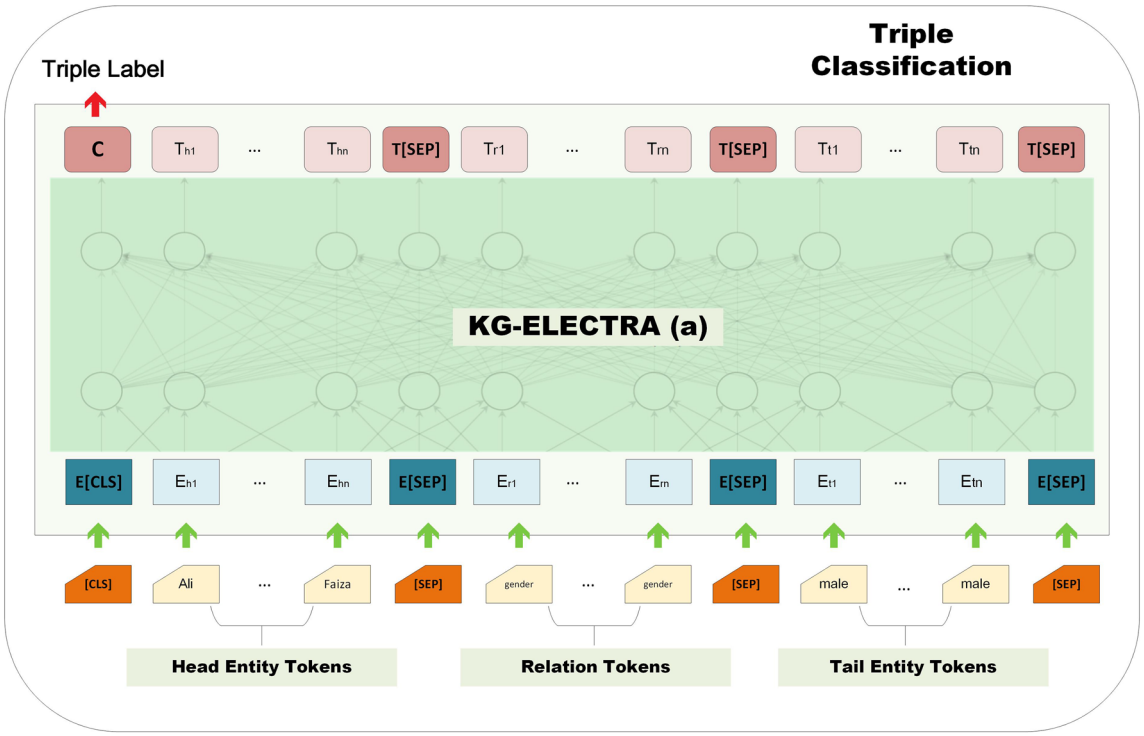

Fig. 2 Illustration of KG-ELECTRA fine-tuning for link prediction task. The motivation for this architecture is the BERT model [25]. The pair of head entity and relation tokens *('andora', 'part of')* is packed with $mh - RP$ *('andora',..., 'Europe')* in the form of input sequence sentence for KG-ELECTRA (b) input. The tail entity (answer) is predicted by predicting the correct start and end span from $mh - RP$

Table 1 Examples of link prediction from WN18-SQuAD

Multi-hop relational path (Paragraph)	Entity–relation pair (Question)	Predicted entity (Answer)
Investigation, derivationally related form, investigate, derivationally related form, inquirer	Investigate, derivationally related form	Inquirer
Picket, synset domain topic of, military, hypernym, force	Picket, hypernym	Force

designed three different datasets of KG on the pattern of SQuAD v1.1 [28]. FB15K-SQuAD, WN18-SQuAD and WN18RRSQuAD are the datasets that we designed for the experiments of the link prediction task. We designed these datasets by generating mh-RP as a paragraph, head entity/relation pairs as questions and tail entity as answers. We used the original triples from FB15K, WN18 and WN18RR datasets for generating their SQuAD versions. We generated paragraphs by selecting any random triple from the KG and then appending the triple that has the tail entity of the previous triple as its head entity and so on. WN11, WN18 and WN18RR are the subsets of WordNet [1] which is an English lexical and grammar-based KG. FB13 and FB15K are the subsets of Freebase [2] KG which consists of general world facts. The test sets used for the classification task contain both positive and negative examples while the test sets used for the link prediction task only contain positive examples of triples. The train sets used for link prediction contain only one answer for each question while the dev set contains three answers for a question out of which there exist only one correct answer and two incorrect answers except the last triple of paragraph, i.e., the wrong answers are entities from next triple of paragraph; hence, the last triple does not have further triples ahead; therefore, it contains all three correct answers. The statistics of the data sets used are given in Table 2.

4.2 Baseline models

We compared KG-ELECTRA with KG-BERT [12] for the task of triples classification because it outperforms all other state-of-the-art models such as TransE [6] and it's all other

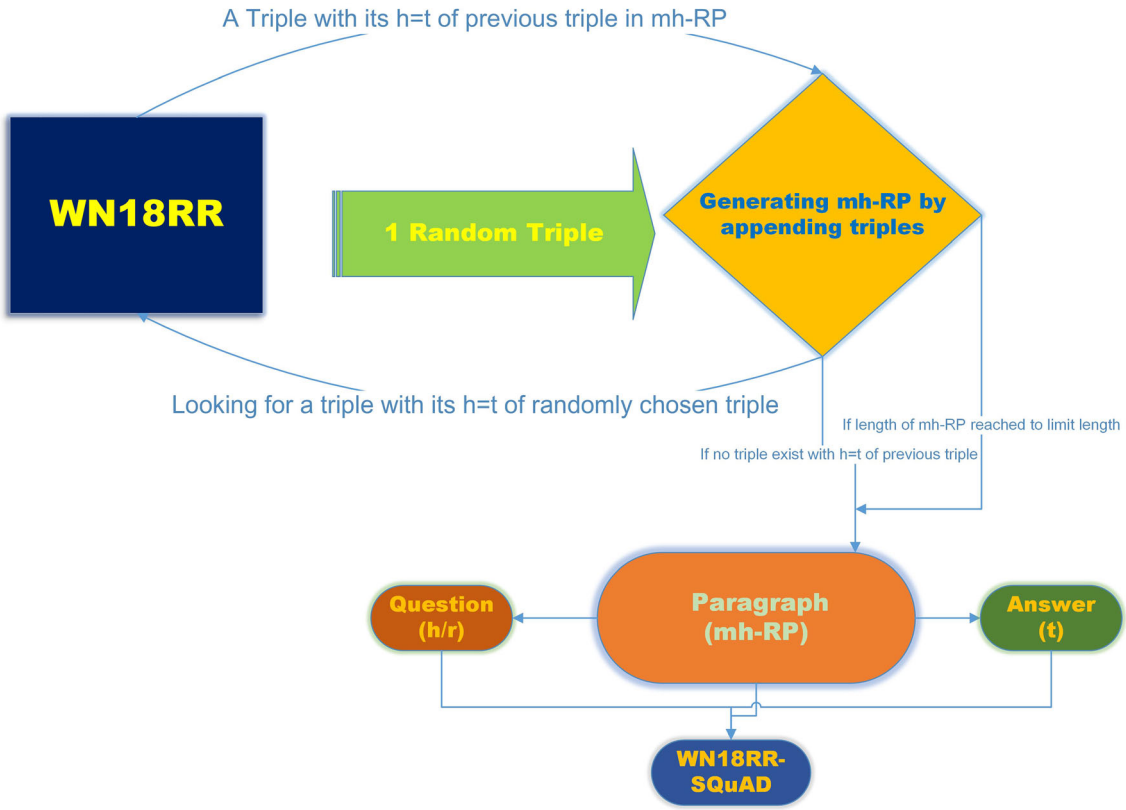

Fig. 3 Workflow of data preparation for link-prediction task

extensions, the tensor network NTN [9] and its version ProjE [29], CNN models: ConvKB [20], ConvE [21] and R-GCN [22], some knowledge incorporating models: TEKE [30], DKRL [31], SSP [32], AATE [33], Contextualized KG embeddings: DOLORES [34], Complex-valued KG embeddings ComplEx [35] and RotatE [36] and adversarial learning framework KBGAN [14]. For the task of link prediction which is based on question answering task, we compared our proposed model with the BERT model [25].

4.3 Experimental settings

For our experiments, we initialized KG-ELECTRA with the base version of pre-trained ELECTRA [13] that has 12 layers, 768 hidden sizes and 110 M parameters.[2] We also experimented by fine-tuning other versions of ELECTRA, but we found results of ELECTRA-Small lesser as compared to ELECTRA-Base; however, ELECTRA-Large performs better on the cost of huge computation as compared to ELECTRABase. Therefore, in our experiments, we fine-tune the pre-trained ELECTRA-Base for the tasks of classification and link prediction. We also followed ELECTRA-Base [13] for the hyper-parameters of KG-ELECTRA. The used fine-tune hyper-parameters are given in Table 3. We also tried other hyper-parameters but found no difference in results; however, the parameters we used make the fine-tuning process faster than others. For the link prediction task, we tried different number of paragraphs for question answering and found that KG-ELECTRA (b) performs well on 200 paragraphs for both train and dev sets of FB15K-SQuAD and WN18-SQuAD while on 500 paragraphs of the WN18RR-SQuAD dataset. We found that by increasing the number of paragraphs (1000, 3000, 5000, 10000) the results decrease; therefore, we found 200 number of paragraphs best for both train and dev sets of link prediction are further explained in ablation experiments section.

4.4 Triples classification

The task of triples classification differentiates the true triples from the false or negative triples of KG. The negative triples are generated by replacing either the head or the tail entity of already present positive triples in the respective datasets. The accuracy results for different models on the datasets of WN11 and FB13 are given in Table 4. The analysis of results represents that TransE [6] performs

[2] https://github.com/google-research/electra.

Table 2 Dataset statistics for KG-ELECTRA experiments

Dataset	Training examples	Dev examples	Test examples
WN11	112581	5218	21088
FB13	316232	11816	47466
FB15K-SQuAD	2369	2324	
WN18-SQuAD	1206	1182	
WN18RR-SQuAD	2133	2097	

Table 3 Fine-tuning hyper-parameters

Hyper-parameters	Values
Adam	1e-6
Epochs	2 for link prediction, 3 for classification
Dropout	0.1
Batch Size	8
Learning rate	1e-4
Learning rate decay	Linear
Maximum sequence length	128
Layer-wise learning rate decay	0.8

badly as compared to its extensions (TransG [18], TransH [37], TransR [38], TransD [39] and TranSparse-S [40]) because TransE cannot deal with multi-relational paths while its variants use relation-specific parameters and outperform TransE. The DistMult-HRS [19] outperforms its original model DistMult [7] because DistMult lacks the structure capabilities for hierarchical relations. DOLORES [34] the model performs better than ConvKB [20] by introducing the contextual information along with entity–relation text. TEKE [30] and AATE [33] models also perform better than TransE and DistMult due to incorporating external information; however, they still lack the utilization of rich language patterns. KG-BERT [12] model outperforms all these mentioned models because it fully utilized rich linguistic patterns of large incorporated external data; however, it utilized large number of parameters and computational costs. Our model KG-ELECTRA (a) performs comparatively to the KG-BERT model with minimum computational resources and parameters.

4.5 Link prediction

In the link prediction task, we give the entity/relation pair (as a question) along with the mh-RP paragraph to KG-ELECTRA (b) model and predict the right entity (answer) as output. We designed a specific dataset for this task on the bases of SQuAD [28] dataset. Our dataset contains mh-RP as a paragraph and head/relation pair as a question while the tail entity as the answer. We first train the model on correct answers for each question which means there is only one answer for each question in the training dataset. While we test the model by providing three answers for each question, two answers are wrong and only one answer is correct. The model has to find the correct answer for each question by locating the correct start position and end position of the answer from the mh-RP. The wrong answers are also taken from the mh-RP but from another triple which means the wrong answers given in the dev set are the tail entities of a triple next to the question triple.

We experimented on different numbers of mh-RP paragraphs such as 200, 500, 1000, 3000 while we found that model performs better on 200 paragraphs, and the results decrease by increasing the number of paragraphs. Therefore, we found the 200 paragraphs for KG-ELECTRA (b) most appropriate for our experiments. In these experiments, we find the effects of mh-RP on link prediction task and we also find that how our model effectively locates the correct entity (answer) from the mh-RP paragraph.

In our experiments, we predicted the missing entity for the triples already present in the KG. We also experimented on finding the answers for indirect relations that are not present in the KG. For example, there is a triple in KG like James and lives in England, and there is another triple England, language, English so we generated an indirect question like, James, language, English. A further illustration of these experiments is given in the section on ablation studies. However, the results of these indirect questions are lesser than the results of direct questions that are already present in the KG and further explained in the ablation studies section. Therefore, in our experiments, we focused only on finding the answers to direct questions and the illustration of which is given in Fig. 4. However, the

Table 4 Triples classification accuracy for latest embedding models. The baseline results are obtained from KG-BERT paper [12]

Model	WN11 (Accu. %)	FB13 (Accu. %)	Average (Accu. %)
NTN	86.2	90.0	88.1
TransE	75.9	81.5	78.7
TransH	78.8	83.3	81.1
TransR	85.9	82.5	84.2
TransD	86.4	89.1	87.8
TEKE	86.1	84.2	85.2
TransG	87.4	87.3	87.4
TranSparse-S	86.4	88.2	87.3
DistMult	87.1	86.2	86.7
DistMult-HRS	88.9	89.0	89.0
AATE	88.0	87.2	87.6
ConvKB	87.6	88.8	88.2
DOLORES	87.5	89.3	88.4
KG-BERT(a)	93.5	90.4	91.9
KG-ELECTRA (a)	91.14	90.18	90.66

results for the link prediction task are given in Table 5. We compare our model with BERT [25] and *path-KCGAN*, and the results show that our model outperforms both comparative models within comparatively fewer computational resources.

4.6 Ablation studies

In a further evaluation of KG-ELECTRA, we conducted different ablation studies for the task of link prediction to find the effects of these studies on the performance of our proposed model. These ablation studies are as follows:

4.6.1 Paragraph length

In this ablation study, we experimented by changing the length of the mh-RP paragraph. Here, we conducted experiments by changing the paragraphs to a length of the different number of triples. We experimented on different paragraphs with the length of 10 triples, 25 triples, 50 triples, 100 triples and 125 triples. The results of these experiments show that the most effective paragraph length for the task of link prediction lies between 25 and 50. The datasets with a paragraph length of 10 show lesser results as compared to the dataset with a paragraph-length of 25. All the datasets having examples with paragraph-length greater than 25 also show lesser results as compared to the examples with paragraph-length of 25 except WN18RR, i.e., the WN18RR shows f1 score 89.18 with paragraphs of 25 triples while showing a slightly higher f1 score of 90.55 with paragraph-length of 50 triples. Therefore, in our experiments, we used the paragraph length up to 25 triples.

4.6.2 Number of paragraphs

In this study, we evaluated KG-ELECTRA by changing the number of mh-RP paragraphs in train and dev sets. We experimented on datasets with different numbers of paragraphs, e.g., 200, 500, 1000. The results of these different experiments show that KG-ELECTRA performs better on datasets with 200 paragraphs than on datasets with paragraphs greater than 200. We experimented on all different datasets of link prediction with these variable numbers of paragraphs, and the results show that for all datasets 200 paragraphs are most suitable than more than 200. Therefore, in all our experiments we used 200 number of paragraphs for both train and dev sets Tables (6, 7, 8 and 9).

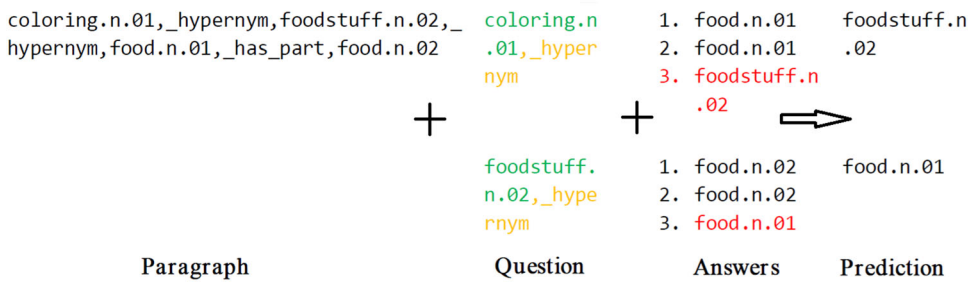

Fig. 4 Example of link prediction from WN18RR-SQuAD dataset. The green color here represents the head entity, yellow color represents relation, and the red color represents tail entity

Table 5 Results for experiments of link prediction

Model	WN18-SQuAD (F1)	WN18RR-SQuAD (F1)	FB15K-SQuAD (F1)
path-KCGAN	78.08	70.19	87.34
BERT	82.26	71.51	89.40
KG-ELECTRA (b)	83.03	89.18	92.37

Table 6 Effect of various paragraph length on performance of KG-ELECTRA

Paragraph length (No. of triples)	F1 score
10	82.42
25	89.14
50	90.55
100	88.55
125	86.93

Table 7 Effect of different number of paragraphs in datasets on the performance of KG-ELECTRA

Number of paragraphs	F1 score
200	89.18
500	85.17
1000	79.89

4.6.3 Position of correct answer in dev set

For the task of link prediction, the dev set contains three answers for each question. There exists only one correct answer out of these three answers. In this ablation study, we experimented on changing the position of the correct answer in the sequence of answers. Firstly, we experimented with a dev set that contains all the examples with their correct answer at the same position (e.g., first answer is correct while the second and third, are both wrong answers). Then, we experimented with a dev set that contains examples with two different positions of correct answer, i.e., half examples contain first answer as correct answers following two wrong answers while half examples contain the third answer as the correct answer followed by two wrong answers. The results of this ablation study show that a dev set with the only 1-type correct answer position performs lesser than a dev set with 2-type correct answer position.

Table 8 Effect of correct answer position in dev set on the performance of KG-ELECTRA

Position of correct answer	F1 score
1-Type position	82.52
2-Type position	89.14

Table 9 Results for performance of KG-ELECTRA on indirect link prediction

Types of triples	F1 score
Directly related	89.14
Indirectly related	77.71

4.6.4 Indirect relations

In this ablation study, we conducted experiments on a dataset with only indirect relation examples. In a KG, we usually predict the direct link between two entities; however, in this study, we tried to find the indirect link between the two entities of a KG. For example, predicting the indirect link between a person and his country by using a triple that shows a direct link between the person and his city of birth (i.e., "person, born in, Islamabad" and "Islamabad, the city in, Pakistan" are the two triples with direct relations; however, "person, born in, Pakistan" is a triple with indirect relation that is not given in KG). The results of this study show that predicting indirect link between two entities using KG-ELECTRA shows lesser results that predict the direct link Therefore, in our original experiments of KG-ELECTRA, we only used directly related triples for link prediction task.

5 Conclusion and future work

KGs have many different applications in AI such as semantic search, question answering and recommendation systems; however, KGs often suffer from incompleteness.

Many literature studies introduced different approaches for the completion of KGs, but these approaches also have many limitations. In our research study, we introduced a lightweight framework called KG-ELECTRA for better completion of KGs. Our framework performs two tasks for KG completion which are, triples classification and link prediction. Therefore, it has two different representations, i.e., KG-ELECTRA (a) for triples classifications and KG-ELECTRA (b) for the link prediction task. We performed triples classification on WN11 and FB13 datasets; however, we designed new datasets WN18-SQuAD, WN18RR-SQuAD and FB15-KSQuAD for the link prediction task. The results of experiments show that our proposed model KG-ELECTRA performs approximately equal to state-of-art models in triples classification tasks; however, it outperforms state-of-the-art models in link prediction tasks with much lesser computation consumption. In the future, we will work on investigating the performance of our proposed model KG-ELECTRA for other tasks of KG completion is like relation prediction. We will also try to work on increasing the performance of KG-ELECTRA for indirectly related triples of the KG.

Declarations

Conflict of interest The authors have no conflict of interest.

References

1. Miller GA (1995) Wordnet: a lexical database for english. Commun ACM 38(11):39–41
2. Bollacker K, Evans C, Paritosh P, et al (2008) Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD, pp 1247–1250
3. Suchanek FM, Kasneci G, andWeikum G (2007) Yago: a core of semantic knowledge. In: WWW. ACM, pp 697–706
4. Cui W, Xiao Y, Wang H et al (2017) KBQA: learning question answering over qa corpora and knowledge bases. Proc VLDB Endow 10(5):565–576
5. Zhang, F, Yuan NJ, Lian D, et al (2016) Collaborative knowledge base embedding for recommender systems. In: KDD. ACM, pp 353–362
6. Yang B, Yih WT, He X, et al (2015) Embedding entities and relations for learning and inference in knowledge bases. In: ICLR
7. Bordes A, Usunier N, Garcia DA, et al (2013) Translating embeddings for modeling multi-relational data. In: NIPS, pp 2787–2795
8. Nickel M, Tresp V, Kriegel HP (2011) A three way model for collective learning on multi-relational data. In: ICML, pp 809–816
9. Socher R, Chen D, Manning CD, Ng A (2013) Reasoning with neural tensor networks for knowledge base completion. In: NIPS, pp 926–934
10. Xie R, Liu Z, Jia J, et al (2016) Representation learning of knowledge graphs with entity descriptions. In: AAAI
11. Xie R, Liu Z, Sun M (2016) Representation learning of knowledge graphs with hierarchical types. In: IJCAI, pp 2965–2971
12. Liang Y, Chengsheng M, Luo Y (2019) KG-BERT: BERT for Knowledge Graph Completion. arXiv:1909.03193v2 [cs.CL]
13. Clark K, Luong MT, Le QV, et al (2020) Manning ELECTRA: Pre-training Text Encoders as Discriminator rather than Generator. arXiv:2003.10555v1 [cs.CL]
14. Cai L, andWang WY (2018) KBGAN: Adversarial learning for knowledge graph embeddings. In: NAACL, pp 1470–1480
15. Goodfellow I, Pouget-Abadie J, Mirza M, et al (2014) Generative adversarial nets. Adv Neural Inf Process Syst
16. Wang J, Yu L, Zhang W, et al (2017) Irgan: a minimax game for unifying generative and discriminative information retrieval models. In: The 40th international ACM SIGIR conference on research and development in information retrieval
17. Zia T, Zahid U, Windridge D (2019) A generative adversarial strategy for modeling relation paths in knowledge base representation learning. In: 33rd Conference on neural information processing systems (NeuraIPS 2019), Vancouver, Canada
18. Xiao H, Huang M, Zhu X (2016) TransG: a generative model for knowledge graph embedding. ACL 1:2316–2325
19. Zhang Z, Zhuang F, Qu M, et al (2018) Knowledge graph embedding with hierarchical relation structure. In: EMNLP, pp 3198–3207
20. Dettmers T, Minervini P, Stenetorp P, Riedel S (2018) Convolutional 2d knowledge graph embeddings In: AAAI, pp 1811–1818
21. Nguyen DQ, Nguyen TD, Phung D (2018) A convolutional neural network-based model for knowledge base completion and its application to search personalization Semantic Web
22. Schlichtkrull M, Kipf TN, Bloem P, et al (2018) Modeling relational data with graph convolutional networks. In: ESWC, pp 593–607
23. Zhou Z, Wang C, Feng Y, Chen D (2022) JointE: jointly utilizing 1D and 2D convolution for knowledge graph embedding. Knowl Bases Syst 240:108100
24. Xie X, Zhang N, Li Z et al (2022) From Discrimination to Generation:Knowledge Graph Completion with Generative Transformer. arXiv:2202.02113v6 [cs.CL] 29 Mar 2022
25. Devlin J, Chang MW, Lee K,Toutanova K (2019) Bert: pre-training of deep bidirectional transformers for language understanding. In: NAACL, pp 4171–4186
26. Wang A, Singh A, Michael J et al (2019) GLUE: a multi-task benchmark and analysis platform for natural language understanding. In: ICLR
27. Zia T, Windridge D (2021) A generative adversarial network for single and multi-hop distributional knowledge base completion. Neurocomputing 461:543–551
28. Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016) Squad: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 conference on empirical methods in natural language processing pages, pp 2383–2392
29. Shi B, Weninger T (2017) ProjE: embedding projection for knowledge graph completion. In: AAAI
30. Wang Z, Li JZ (2016) Text-enhanced representation learning for knowledge graph. In: IJCAI, pp 1293–1299
31. Wang Z, Li JZ (2016) Text-enhanced representation learning for knowledge graph. In: IJCAI, pp 1293–1299
32. Xiao H, Huang M, Meng L, Zhu X (2017) SSP: semantic space projection for knowledge graph embedding with text descriptions In: AAAI
33. An B, Chen B, Han X, Sun L (2018) Accurate text-enhanced knowledge graph representation learning. In: NAACL, pp 745–755

34. Wang H, Kulkarni V, Wang WY (2018) Dolores: Deep contextualized knowledge graph embeddings. arXiv preprint arXiv:1811.00147
35. Trouillon T, Welbl J, Riedel S, et al (2016) Complex embeddings for simple link prediction. In: ICML, pp 2071–2080
36. Sun Z, Deng ZH, Nie JY, Tang J (2019) Rotate: knowledge graph embedding by relational rotation in complex space. In: ICLR
37. Wang Z, Zhang J, Feng J, Chen Z (2014) Knowledge graph embedding by translating on hyperplanes. In AAAI
38. Lin Y, Liu Z, Sun M, et al (2015) Learning entity and relation embeddings for knowledge graph completion. In: AAAI
39. Ji G, He S, Xu L, et al (2015) Knowledge graph embedding via dynamic mapping matrix. In: ACL, J, pp 687–696
40. Ji G, Liu K, He S, Zhao J (2016) Knowledge graph completion with adaptive sparse transfer matrix. In: AAAI