RAILD: Towards Leveraging Relation Features for Inductive Link Prediction In Knowledge Graphs

Genet Asefa Gesese
genet-asefa.gesese@fiz-karlsruhe.de
FIZ Karlsruhe – Leibniz Institute for Information Infrastructure
Germany
Karlsruhe Institute of Technology, Institute AIFB
Germany

Harald Sack
harald.sack@fiz-karlsruhe.de
FIZ Karlsruhe – Leibniz Institute for Information Infrastructure
Germany
Karlsruhe Institute of Technology, Institute AIFB
Germany

Mehwish Alam
mehwish.alam@fiz-karlsruhe.de
FIZ Karlsruhe – Leibniz Institute for Information Infrastructure
Germany
Karlsruhe Institute of Technology, Institute AIFB
Germany

ABSTRACT

Due to the open world assumption, Knowledge Graphs (KGs) are never complete. In order to address this issue, various Link Prediction (LP) methods are proposed so far. Some of these methods are inductive LP models which are capable of learning representations for entities not seen during training. However, to the best of our knowledge, none of the existing inductive LP models focus on learning representations for unseen relations. In this work, a novel Relation Aware Inductive Link preDiction (RAILD) is proposed for KG completion which learns representations for both unseen entities and unseen relations. In addition to leveraging textual literals associated with both entities and relations by employing language models, RAILD also introduces a novel graph-based approach to generate features for relations. Experiments are conducted with different existing and newly created challenging benchmark datasets and the results indicate that RAILD leads to performance improvement over the state-of-the-art models. Moreover, since there are no existing inductive LP models which learn representations for unseen relations, we have created our own baselines and the results obtained with RAILD also outperform these baselines.

CCS CONCEPTS

• **Computing methodologies** → **Artificial intelligence**; **Machine learning**.

KEYWORDS

Knowledge graphs, Inductive link prediction, Textual descriptions, Entity representations, Relation representations

ACM Reference Format:
Genet Asefa Gesese, Harald Sack, and Mehwish Alam. 2018. **RAILD**: Towards Leveraging Relation Features for Inductive Link Prediction In Knowledge Graphs. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX).* ACM, New York, NY, USA, 9 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recently, Knowledge Graphs (KGs) have gained massive attention for use in various applications such as question answering, information retrieval, recommender systems, etc [24]. Due to the open-world assumption KGs are never complete [6] and hence, there arises the need for automated KG Completion (KGC) systems. In order to tackle this problem, various works [8, 12] have been done so far which are either rule-based techniques or distributed representation (embedding) learning methods. Many of the KGC approaches which are based on representation learning techniques utilize the well known Link Prediction (LP) task, i.e., the task of predicting missing links in the KG. The benefit of using an embedding-based LP task over rule-based approaches is that the embeddings of entities and relations learned in the LP task could also be leveraged in other downstream tasks.

There are two major types of setups in LP tasks, i.e., transductive and inductive. In a transductive setup, all entities in the train and validation sets are required to be part of the training set. On the other hand, in an inductive setup, the validation and test sets may contain entities that are not seen during training. Even though most of the well known LP approaches [2, 13, 18, 21, 27] are proposed for transductive settings, there are also several approaches [3, 4] introduced that work in inductive settings. However, these representation-learning based inductive LP approaches do not pay attention to relations. Unlike entities for which there are textual descriptions that could be used as features for the entities, relations are usually just randomly initialized like in BLP [4]. QBLP [1] is a method that is proposed for inductive LP in hyper-relational graphs. This work could be generalized for unseen relations but since it does not provide a way to generate features for unseen relations, it could only be applied for inductive LP with relations involved during training.

The most straightforward way to find features for relations is to use the descriptions of the relations. However, the issue is that the textual description could be either too short or entirely unavailable. In such cases, it is required to generate features for relations utilizing the structural information available at hand. This indicates that there is a need for a method which generates features for relations so that inductive LP could be performed with unseen relations.

To this end, in this work, a novel approach *Relation Aware Inductive Link preDiction (RAILD)* which predicts missing links in KGs considering both unseen entities and unseen relations is introduced. To the best of our knowledge, *RAILD* is the first approach that handles unseen relations. It works by fine-tuning a pre-trained

Language Model (LM) to encode textual descriptions of entities and relations. Moreover, it generates a graph-based relation features by first applying a novel algorithm named *Weighted and Directed Network of Relations (WeiDNeR)* to build a directed relation-relation network from the triples available in the KG and then, generating embeddings for the relations in the network using Node2Vec model which leverages contextual information based on random walks. Then, these embeddings are in turn used as features for the relations for the LP task. Moreover, RAILD also utilizes the textual descriptions of the relations as features, by either combining them with the features generated by the feature generator component or separately.

Figure 1 provides an example of inductive LP settings followed in this work. Generally, inductive LP is divided into two categories: i) semi-inductive and ii) fully-inductive. In the semi-inductive setting, either the head or the tail is unseen but not both, and in the fully-inductive setting, both head and tail entities are unseen. In this definition, relations are often overlooked, i.e., they are usually assumed to be seen in the training set and hence are randomly initialized or as in MLMLM [3], they could be encoded using their labels (corresponding text descriptions) but without learning representations (embeddings) for them. Hence, this work divides the settings into three categories for clarity, i.e., semi-inductive (with seen relations), fully-inductive (with seen relations), and truly-inductive (with unseen entities and unseen relations).

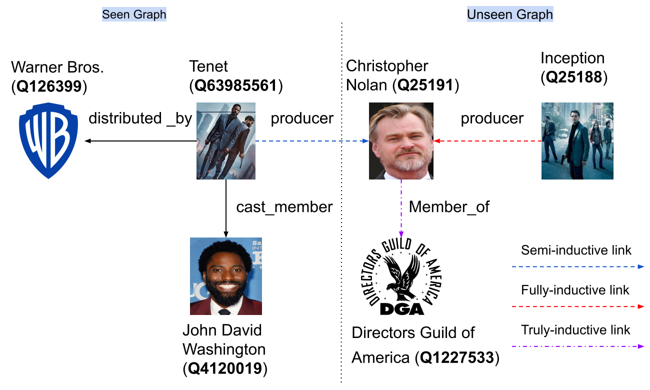

Figure 1: An example illustrating different settings of inductive LP tasks, i.e., semi-inductive (the link from *Tenet* to *Christopher Nolan*), fully-inductive (the link from *Inception* to *Christopher Nolan*), and truly-inductive (the link from *Christopher Nolan* to *Directors Guild of America*) settings

This work formulates and addresses the following research questions: **Does encoding relations by generating features result in any performance improvement over State-Of-The-Art (SOTA) inductive LP models and also enable LP with unseen relations?** Following are the main contributions that are achieved in this work while attempting to answer the question:

- A novel algorithm is introduced to build a *relation-relation network*, i.e., **WeiDNeR**, for the purpose of generating features for relations in a KG solely from the contextual information present in the graph structure.
- The results indicate that instead of randomly initializing relations in inductive LP, encoding the relations like the way entities are encoded by utilizing proper features leads to outperforming the SOTA inductive LP models on triple-based KGs.
- An experiment-supported evidence is provided showing that the algorithm introduced to generate features, WeiDNeR, enables producing competitive results as compared to using textual descriptions of relations as features.
- As part of the work, a novel LP dataset named **Wikidata68K**[1] which contains unseen relations in the validation and test sets is introduced along with an automated pipeline to generate such datasets. Creating this dataset was required as there exists no such kind of evaluation dataset due to the fact that, to the best of our knowledge, no existing LP work deals with unseen relations. The results obtained with the proposed model on this challenging dataset are provided which could be seen as a first attempt to facilitate further research in the community on the topic of LP with unseen relations.

This paper is organized as follows: Section 2 discusses the related works in the area of inductive LP. In Section 3, the proposed model is presented followed by discussions on experimental results in Section 4. Section 5 concludes the work with some future directions.

2 BACKGROUND AND RELATED WORKS

2.1 Inductive LP Settings in triple-based KGs

Existing inductive LP models operate either in `semi-inductive` setting where one of the head or tail entities is seen during training or `fully-inductive` setting where both head and tail entities are unseen during training. In both these settings, relations are usually assumed to be known during training. For the sake of clarity, in this work, predicting with unseen relations is defined separately named as `truly-inductive` LP setting. The formal definitions of these three settings are provided as follows:

Given a KG $G = (E, R)$ where E and R represent set of entities and relations respectively, let T_{tr}, T_{va}, and T_{te} be sets of training, validation, and test triples where E_{tr} & R_{tr}, E_{va} & R_{va}, and E_{te} & R_{te} are their corresponding set of entities and relations respectively.

*In **semi-inductive** setting.* For every triple $< h, r, t > \in T_{va}$ or $< h, r, t > \in T_{te}$, either or both of $h \in E_{tr}$ & $t \in T_{tr}$ holds true while $R_{va} \subseteq R_{tr}$ and $R_{te} \subseteq R_{tr}$.

*In **fully-inductive** setting.* For every triple $< h, r, t > \in T_{va}$ or $< h, r, t > \in T_{te}$, both $h \notin E_{tr}$ & $t \notin E_{tr}$ holds true while $R_{va} \subseteq R_{tr}$ and $R_{te} \subseteq R_{tr}$.

*In **truly-inductive** setting.* For every triple $< h, r, t > \in T_{va}$ or $< h, r, t > \in T_{te}$, either or both of $h \notin E_{tr}$ & $t \notin E_{tr}$ holds true while there exist a set $R_v \subseteq R_{va}$ and a set $R_t \subseteq R_{te}$ where $R_v \nsubseteq R_{tr}$ and $R_t \nsubseteq R_{tr}$.

2.2 Inductive LP Approaches

Adapting most of the existing transductive LP models such as RotatE [18], Distmult [27], ComplEx [21], and TransE [2] for inductive settings requires expensive re-training in order to learn embeddings

[1]https://doi.org/10.5281/zenodo.7066504

for unseen entities. Therefore, such models are not applicable to making predictions with unseen entities. This led to the creation of some inductive LP approaches which are presented as follows.

Rule-based methods. Statistical rule-mining approaches make use of logical formulas to learn patterns present in KGs [14]. Despite the fact that such approaches are inherently applicable to inductive settings, they are prone to limited expressiveness and scalability issues. In order to address this issue, NeuralLP [28] is proposed and it works by learning first-order logical rules in an end-to-end differentiable model. DRUM [16] is another method that applies a differentiable approach for mining first-order logical rules from KGs and provides improvement over NeuralLP.

Embedding-based methods. Training entity encoders through feed-forward and graph neural networks is one way to generate representations for unseen entities as in GraphSAGE [11]. However, such approaches require fixing a set of attributes (e.g., bag-of-words) before training in order to learn entity representations which leads to restricting their application on downstream tasks as discussed in [4]. Aggregating neighborhood information through a graph neural network is one way to generate embeddings for entities [10, 23]. The drawbacks of these approaches lie in the fact that they require the new nodes (i.e., the unseen entities) to be surrounded by known nodes and fail to handle entirely new graphs as discussed in [19]. KEPLER [25] is a unified model for knowledge embedding (KE) and pre-trained language representation by encoding textual entity descriptions with a pre-trained LM as their embeddings, and then jointly optimizing the KE and LM objectives. However, due to the additional language modeling objective, KEPLER is quite expensive to compute and requires more training data. BLP [4] utilizes a pretrained LM for learning representations of entities via a LP objective which is inspired by the work DKRL [26]. It demonstrates the power of LMs in facilitating the strong generalizability of entity embeddings on downstream tasks. QBLP [1] is a model proposed to extend BLP for hyper-relational KGs by exploiting the semantics present in qualifiers.

Other Approaches. GraIL [19] is a method that reasons over local subgraph structures to predict missing links in KGs. MLMLM [3] proposes a mean likelihood method to compare the likelihood of different text of different token lengths sampled from a Masked LM to perform LP. Even though both of these approaches could predict missing links with unseen entities, they learn embeddings neither for entities nor for relations.

Note that the models discussed so far do not consider unseen relations except MLMLM which also does not learn embeddings at all. To address both limitations, i.e., considering unseen relations and also learning embeddings while predicting missing links, a novel method RAILD is proposed in this work for inductive LP in triple-based KGs. RAILD improves the SOTA methods such as BLP by introducing a separate encoder that utilizes structured information from KGs to generate features for relations. Moreover, in addition to the textual descriptions of entities, it also leverages the semantics present in the textual descriptions of relations by using a BERT encoder. Note that, in order to perform the LP task, both the structure-based and text-based encoders are combined.

3 RAILD: RELATION AWARE INDUCTIVE LINK PREDICTION

The general architecture of the proposed approach is given in Figure 2. As mentioned before, RAILD fine-tunes BERT pre-trained model to encode entities with an LP task. Differently from BLP where relations are randomly initialized, in RAILD the same pre-trained BERT model is also applied to encode relations using their corresponding textual descriptions, as shown in Figure 2 component ②. In addition to encoding relations using BERT, a feature generator component that is based solely on graph structure is also proposed. Hence, two kinds of vectors could be generated as features for relations, i.e., text-based and graph-based. Given a triple $< h, r, t >$, the two feature vectors generated for the relation r are concatenated into a single vector. Since concatenation of the two relation vectors leads to doubling the output vector dimension, the vectors of the head/tail entities are also duplicated (concatenating the head vector with itself to match the size of the concatenated relation vector). Then, the resulting head $\mathbf{h}$, tail $\mathbf{t}$ and relation $\mathbf{r}$ vectors are passed to the LP scoring function.

Textual description encoding is performed by passing the text as input to a pre-retrained LM (specifically BERT but any other transformer based LM model could be used as well) and then passing the obtained vector from BERT through a feed-forward layer, as shown in Figure 2 component ① and ②. For the graph-based feature generation for relations, two major steps are applied, i.e., building a relation-relation network (Figure 2 component ③) and generating node embeddings for the created network where the nodes are relations (Figure 2 component ④). In the subsequent sections, the different components of the proposed model are presented in detail. First, encoding textual descriptions using pre-trained BERT is discussed followed by the description of the WeiDNeR algorithm. Then, the node embedding model applied in this work is presented. Finally, the chosen scoring functions are analyzed.

3.1 Encoding Textual Descriptions using BERT

Textual descriptions of entities contain information that would provide useful semantics while learning KG representations. In order to make use of such text data for representation learning, both static embedding models such as SkipGram [15] and contextual embedding models like BERT have been extensively applied with different machine learning and natural language processing tasks. The power of transformer networks [22] in encoding text to contextualized vectors has been well received. In particular, pre-trained embedding models such as BERT provide an advantage to fine-tune the model on other downstream tasks.

In BLP, pre-trained BERT is fine-tuned on the inductive LP task and it showed promising results as compared to other methods. Our approach follows the same step, however, the difference is that i) in BLP the encoder is used only for entities whereas in our approach it is used also to encode relations, and ii) the fine-tuning is extended by adding an additional component with structure-based features for relations as shown in Figure 2.

Let $d = (w_1, ..., w_k)$ be an entity or relation description, the BERT tokenizer first adds two special tokens [CLS] and [SEP] to the beginning and end of d, respectively $([CLS], w_1, ..., w_k, [SEP])$. BERT

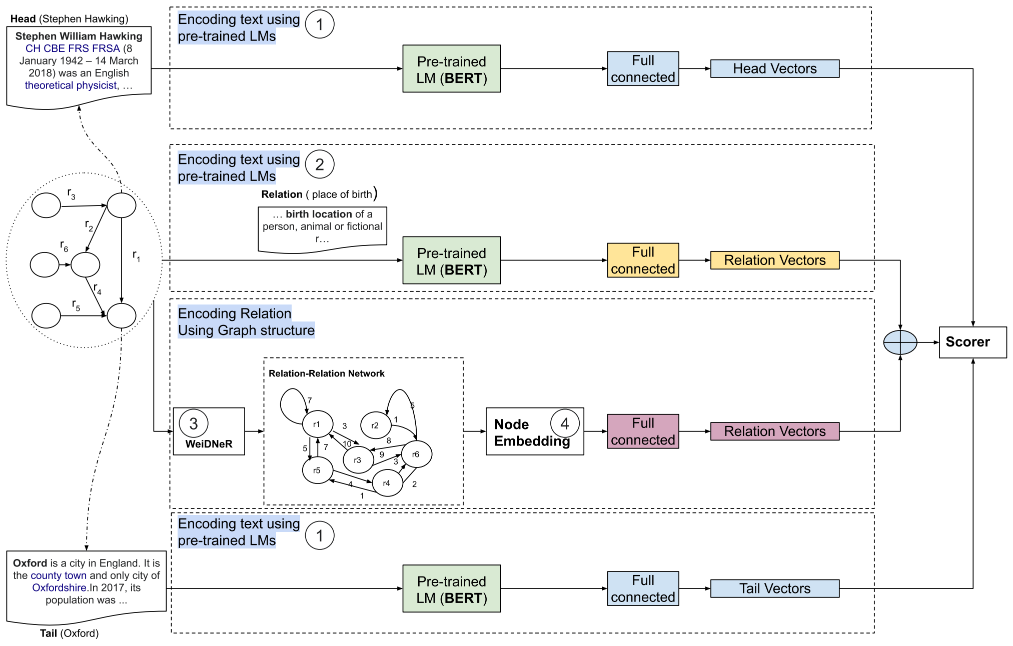
Head (Stephen Hawking)
Stephen William Hawking CH CBE FRS FRSA (8 January 1942 – 14 March 2018) was an English theoretical physicist, ...
Encoding text using pre-trained LMs 1
Pre-trained LM (BERT)
Full connected
Head Vectors
Encoding text using pre-trained LMs 2
Relation (place of birth)
... birth location of a person, animal or fictional r...
Relation Vectors
Encoding Relation Using Graph structure
3 WeiDNeR
Relation-Relation Network
Node Embedding 4
Scorer
Oxford is a city in England. It is the county town and only city of Oxfordshire.In 2017, its population was ...
Tail (Oxford)
Tail Vectors

Figure 2: RAILD framework

takes this as an input leading to a sequence of k + 2 contextualized embeddings as an output, i.e., $BERT(D) = [h_{CLS}, h_1, ..., w_k, h_{SEP}]$.

As in BLP and many other works which employ BERT for text encoding, this work also utilizes the contextualized vector $h_{CLS} \in \mathbb{R}$ where h is the hidden size in the BERT architecture. Once h_{CLS} is obtained, it will be given as an input to a linear layer that reduces the dimension of the representation, to yield the output entity or relation embedding $h = Wh_{CLS}$, where $w \in \mathbb{R}^{d \times h}$ is the weight with d being the chosen embedding dimension. Note that as is shown in Figure 2, the weights are shared with also the linear layer applied to the relation embeddings obtained with Node2Vec model.

3.2 Weighted and Directed Network of Relations (WeiDNeR)

WeiDNeR is designed based on the following assumption. Given a KG $G = (R, E, T \subseteq E \times R \times E)$, $r_1, r_2 \in R$ and $T_1 \subseteq (T \cap (E \times r_1 \times E))$, $T_2 \subseteq (T \cap (E \times r_2 \times E))$, the assumption would be that the higher the number of common entities between T_1 and T_2, the higher the probability that r_1 and r_2 could be semantically similar.

Hence, based on this assumption, an algorithm is proposed which generates a directed and weighted network graph $N_{rel} = (V, L \subseteq V \times V, w)$ where the nodes V are relations in the input KG (i.e., $V \subseteq R$), L is the set of edges connecting the nodes, and $w : L \mapsto \mathbb{R}$ assigns weight to each edge. Algorithm 1, step by step, explains the process of creating the network graph.

Following the procedure in this algorithm, if there is a direct link between two nodes $\textcircled{r_1}$ and $\textcircled{r_2}$ in the generated network N_{rel} then the following statement holds true.

$\left|head(T_1) \cap head(T_2)\right| > 0$ OR $\left|tail(T_1) \cap tail(T_2)\right| > 0$ OR $\left|tail(T_1) \cap head(T_2) > 0\right|$

where $head(T_i)$ and $tail(T_i)$ are the sets of entities occurring at the head and tail positions in the set of triples T_i respectively.

If there is no direct link, then the statement becomes false, i.e., the two relations are not associated with any common entity in the input KG.

Algorithm description. Taking a KG $G = (R, E, T \subseteq E \times R \times E)$ with $\{< h_i, r_j, t_k > | < h_i, r_j, t_k > \in T\}$ where $h_i, t_k \in E$ and $r_j \in R$ as an input and generates a relation-relation network N_{rel}. For each pair of distinct relations ($r_a, r_b \in R$) it performs the following steps:

- it counts the number of pair of triples where the relation in the first triple is r_a and in the second is r_b and the tail entity in the first triple is the same as the head entity in the second triple (i.e., refer to line 3)..
- it counts the number of pairs of triples where the relation in the first triple is r_a and in the second is r_b and the head entity in the first triple is the same as the tail entity in the second triple (i.e., refer to line 4).
- it computes the number of entities shared by the triples associated with r_a and r_b at the exact same position at the head or at the tail, (i.e., refer to line 5).
- If $\#direct + \#indirect > 0$, then an edge from node r_a to node r_b will be created with the summed result given as a weight for the edge (i.e., refer to lines 6 to 10).

On the other hand, if r_a and r_b are the same, then the following will be performed.

- it counts the number of pairs of triples where the relations in both triples is r_a and the tail entity in the first triple is the same as the head entity in the second triple (i.e., refer to line 12)

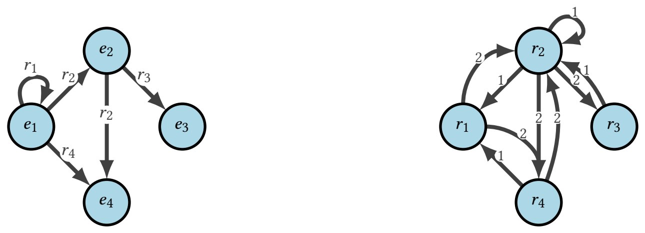

Figure 3: An example to show how Algorithm 1 works; taking the graph in the left, it produces the graph in the right.

- it computes the number of entities shared by the triples associated with r_a at the exact same position at the head or at the tail. (refer to line 13)
- If $\#direct + \#indirect > 0$, then an edge from node r_a to node r_b will be created with the summed result given as a weight for the edge (refer to lines 14 to 16).

Algorithm 1: WeiDNeR - An algorithm to generate a directed and weighted relation-relation network

```
   Data: $T \leftarrow Triples\ in\ KG$
   Result: $N_{rel}$
1  for each pair of relations $< r_a, r_b >$ do
2      if $r_a \neq r_b$ then
3          $\#direct_{<r_a,r_b>} \leftarrow \left|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, t_1 = h_2\}\right|$;
4          $\#direct_{\langle r_b,r_a\rangle} \leftarrow \left|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, h_1 = t_2\}\right|$;
5          $\#indirect \leftarrow \left|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, (h_1 = h_2 \vee t_1 = t_2)\}\right|$;
6          $Weight_{\langle r_a,r_b\rangle} = \#direct_{\langle r_a,r_b\rangle} + \#indirect$;
            $Weight_{\langle r_b,r_a\rangle} = \#direct_{\langle r_b,r_a\rangle} + \#indirect$;
7          if $Weight_{\langle r_a,r_b\rangle} > 0$ then
8              $N_{rel} \leftarrow N_{rel} \bigcup \{\langle r_a, r_b, Weight_{\langle r_a,r_b\rangle}\rangle\}$;
9          if $Weight_{\langle r_b,r_a\rangle} > 0$ then
10             $N_{rel} \leftarrow N_{rel} \bigcup \{\langle r_b, r_a, Weight_{\langle r_a,r_b\rangle}\rangle\}$;
11     else
12         $\#direct \leftarrow \left|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, t_1 = h_2, (h_1 \neq t_1 \vee h_1 \neq t_2)\}\right|$;
13         $\#indirect \leftarrow \left|\{(\langle h_1, r_a, t_1\rangle, \langle h_2, r_b, t_2\rangle) : \langle h_1, r_a, t_1\rangle \in T, \langle h_2, r_b, t_2\rangle \in T, ((h_1 = h_2 \wedge t_1 \neq t_2) \vee (t_1 = t_2 \wedge h_1 \neq h_2))\}\right|$;
14         $Weight_{\langle r_a,r_a\rangle} = \#direct + \#indirect$;
15         if $Weight_{\langle r_a,r_a\rangle} > 0$ then
16             $N_{rel} \leftarrow N_{rel} \bigcup \{r_a, r_a, Weight_{\langle r_a,r_a\rangle}\}$;
```

3.3 Node Embeddings

Features for the nodes in a given network N_{rel} could be generated leveraging the network's structural information. In order to generate these features, in this work, Node2Vec [9] which learns continuous feature representations for nodes in networks with the likelihood of preserving neighborhood information is used. It applies second-order (biased) random walks to efficiently explore diverse neighborhoods of a given node and then makes use of the SkipGram [15] word embedding method to learn embeddings by treating the generated walks as sentences. Given a network (such as N_{rel}), in order to select the next hop, Node2Vec computes second-order transition probabilities as shown in Equation 1.

$$p(u|v,t) = \frac{\alpha_{pq}(t,u)w(u,v)}{\sum_{u' \in N_v} \alpha_{pq}(t,u')w(u',v)} \tag{1}$$

where $u, v \in V$, N_v denotes the neighboring nodes of v, and $w(u, v)$ is the weight of the edge between the nodes u and v, and α is the bias factor used to reweigh the edge weights depending on the previous visited state and it is computed state as shown in Equation 2.

$$\alpha(t,v) = \begin{cases} \frac{1}{p}, & \text{if } d_{tv} = 0 \\ 1, & \text{if } d_{tv} = 1 \\ \frac{1}{q}, & \text{if } d_{tv} = 2 \end{cases} \tag{2}$$

where p is the *return parameter* which controls the likelihood of immediately revisiting a node, q is the *in-out parameter* controlling the likelihood of revisiting a node's one-hop neighborhood, and d_{tv} is the shortest distance between the nodes t and v.

These random walks are then passed to the Skip-gram model to learn the node embeddings. Since the Skip-gram model aims to learn continuous feature representations for words by optimizing a neighborhood preserving likelihood objective, in Node2Vec it could be interpreted as aiming to maximize the probability of predicting the correct context node v for a given center node u.

3.4 Training Procedure

The graph-based features for relations in training, validation, and test sets are created separately and used as inputs for when models are trained. In a truly-inductive setting, only the triples from the training set are used to generate the graph-based features for the relations appearing in the training. Similarly, for relations in the validation and test sets, only triples from the validation and test sets are used respectively. This is performed in order to avoid using information from the unseen graphs (i.e., from validation and test sets) to learn features during training. Similarly, in both semi-inductive and fully-inductive settings, only the triples from the training set are taken as input to generate the graph-based feature for the relations.

Once the features of the entities and relations are generated or encoded, then they are used to optimize the model for LP by applying stochastic gradient descent. For each positive triple $< e_i, r_j, e_k >$, a positive score S_p is computed. Then, a corrupted negative triple is created by replacing the head or the tail entity with a random entity, and its score S_n is computed.

3.5 Computational complexity

As it is discussed in the previous sections, RAILD uses text-based and graph-based encoders. The text-based encoder is used for both entities and relations whereas the graph-based encoder is used only to encode relations. Note that the graph-based features for relations are pre-computed and hence, the major part of the computational cost of training the model comes from text-based encoder. The BERT encoder used has a complexity of $O(n^2)$ for encoding a sentence of length n. This entails that for training RAILD, the time complexity would be $O(|T|\,n^2)$ where T is the set of triples. The length of sentences n is in practice fixed and assuming the n is the same for all entities and relations, the complexity would remain linear with respect to the number of triples in the KG, up to a constant factor.

During testing, the text-based encoder is applied only for unseen entities and unseen relations while the embeddings for seen entities and seen relations can be pre-computed. Hence, the LP for a given entity and relation is linear in the number of entities and relations in the graph.

4 EXPERIMENTS

In this section, the details on experimentation including the baselines, the datasets, the experimentation settings, and the results are discussed. Our implementation and the datasets are made publicly available[2].

4.1 Datasets

The three inductive LP settings discussed in Section 2 are considered for experimentation. In order to evaluate RAILD in the semi-inductive setting the datasets FB15K-237 [20] and WN18RR [5] with the splits provided in [4], are used. In a fully-inductive setting, the model is evaluated on dataset Wikidata5M [25] and compared against SOTA models. The statistics of these datasets are provided in Table 1. To the best of our knowledge, there are no benchmark datasets that contain unseen relations in their validation and test sets. To address this issue and to enable the evaluation of RAILD with unseen relations, a new dataset Wikidata68K is created taking Wikidata5M as raw data. The pipeline developed to create this dataset is inspired by [7] and is constituted of the following steps.

1. Input: Raw data T containing triples from which the dataset will be created, and a set of pairs of relations and their types RT. In Wikidata, there exists a metaclass (`Q107649491`: type of Wikidata property) with instances that are types of properties (i.e., relations). For example, `Q29546443` (Wikidata property for items about books) is an instance of `Q107649491` and the property `P123(publisher)` is an instance of `Q29546443`. Therefore, `(P123, Q29546443)` could be an entry in RT.
2. Removing relations which occur in less than N number of triples ($N = 3$, for Wikidata68K).
3. Removing inverse relations, entities and relations without a label, and duplicate relations.
4. Randomly splitting the set of relations into three R_1, R_2, and R_3 while trying to keep the same type of relations in the same set based on RN and extract their corresponding triples T_1, T_2, and T_3 from T.
5. Creating K-cores for each of T_1, T_2, and T_3. (For Wikidata68K, the value of k is set to 10, 6, and 5 for T_1, T_2, and T_3 respectively).
6. Removing relations which are skewed towards either the head or the tail at least 50% of the time, from each of T_1, T_2, and T_3.

[2]https://github.com/GenetAsefa/RAILD

Table 1: Dataset statistics

	WN18RR	**FB15K-237**	**Wikidata5M**	**WD20K(25)**
Relations	11	237	822	333
	Training			
Entities	32,755	11,633	4,579,609	17,275
Triples	69,585	215,082	20,496,514	38,023
	Validation			
Entities	4,094	1,454	7,374	3,092
Triples	11,381	42,164	6,699	4,072
	Test			
Entities	4,094	1,454	7,475	2,615
Triples	12,087	52,870	6,894	3,329

	Wikidata68K		
	Training	Validation	Test
Entities	55,488	6,559	5,813
Relations	72	37	44
Triples	667,413	67,892	45,512

4.2 Baselines

For semi-inductive and fully-inductive settings, RAILD could be compared with SOTA models like BLP and KEPLER on FB15K-237, WN18RR, and Wikidata5M datasets. Since there exists no SOTA model which handles unseen relations, four different baselines Glove-BOW$_t$, Glove-DKRL$_t$, BE-BOW$_t$, and BE-DKRL$_t$ are created by extending the baselines in BLP, i.e., Glove-BOW, Glove-DKRL, BE-BOW, and BE-DKRL respectively to also encode relations using their textual descriptions in the same way they encode entities. These models are different re-implementations of DKRL [26] where Glove-DKRL uses Glove embeddings as an input to the DKRL architecture whereas Glove-BOW is the Bag-Of-Word baseline of DKRL. Furthermore, BE-BOW, and BE-DKRL are other varieties that use context-insensitive BERT Embeddings (BE). Note that the baselines created in this work are used for evaluation in the truly-inductive setting on Wikidata68K dataset and to compare them with RAILD.

4.3 Experimentation Setting

Scoring. TransE, SimplE, DistMult, and ComplEx are some of the well known translational models with `TransE` being the simplest among all. ComplEx handles antisymmetric relations better than both TransE and DistMult [21]. However, TransE could also perform well in some cases, for example, in BLP the best performing scoring function is TransE followed by ComplEx. Hence, in this paper, the scoring functions `TransE` and `ComplEx` are selected.
Model Selection For the Node2Vec model, number of walks=1000, length=10, window size=10, epochs=100, dim=768 are used for FB15K-237 with semi-inductive split and Wikidata5M with fully-inductive split. For WN18RR with semi-inductive split, number of walks=5000, length=10, window size=5, epochs=100, dim=768 are used. For Wikidata68K, number of walks=10, length=10, window size=10, epochs=100, and dim=768.

Similar to [4], for all newly created baselines and RAILD models, a grid search is run on FB15K-237 and the hyperparameter values with the best performance on the validation set are chosen. Then, these values are reused for training with the other datasets. For the

BOW and DKRL baselines, inspired by [4], learning rate: 1e-5, 1e-4, 1e-3, L2 regularization coefficient: 0, 1e-2, 1e-3 are applied. Adam optimizer is used with no learning rate schedule, and the models are trained for 80 epochs with a batch size of 64 with WN18RR, FB15k-237, and 40 epochs with a batch size of 254 with Wikidata68K.

For the RAILD models, loss function: margin, negative log-likelihood, learning rate: 1e-5, 2e-5, 5e-5, L2 regularization coefficient: 0, 1e-2, 1e-3 are used. Adam optimizer with a learning rate decay schedule with a warm-up for 20% of the total number of iterations is used. The models are trained for 40 epochs (80 epochs for models which combine text and graph-based features for relations) with a batch size of 64 with WN18RR and FB15k-237, and 5 epochs with a batch size of 128 with Wikidata5M. In all the experiments, the negative sample size is set to 64.

4.4 Results

Two main varieties of RAILD, i.e., `RAILD-TransE` and `RAILD-ComplEx`, are created with the scoring functions TransE and ComplEx respectively. The results obtained with the different inductive LP settings are discussed in the subsequent sections.

4.4.1 ***Results in semi-inductive setting.*** The results obtained with the semi-inductive setting on WN18RR and FB15K-237 are shown in Table 2. The table compares 2 different varieties of RAILD (i.e., RAILD-TransE and RAILD-ComplEx) with the different models from [4] and our baselines. These results show that `RAILD-TransE` outperforms all the other models on FB15K-237 w.r.t. all metrics. On the contrary, on WN18RR RAILD-ComplEx provides the best result w.r.t. all metrics whereas the second best results are obtained with RAILD-TransE w.r.t. all metrics except Hits@1.

Although TransE is a less elaborate model than ComplEx, it provides better results when used with RAILD on FB15K-237 and competitive results on WN18RR. Same is the case with the results obtained in the truly-inductive setting on Wikidata68K (see Section 4.4.3). This suggests that the expressiveness of TransE could be highly improved with RAILD which has a more expressive encoder.

4.4.2 ***Results in fully-inductive setting.*** Table 4 shows the results obtained with the fully-inductive setting on the dataset Wikidata5M. Due to limited computational resources, for the experiment on Wikidata5M the distilled version of Bert, i.e., DistilBert [17] is used since it is cheaper to train as compared to Bert. However, it should be noted that since DistilBert is a slimmed-down version of BERT with fewer parameters it may lead it to be less powerful than Bert. Although MLMLM is not an embedding-based LP model, it is compared with our approach on Wikidata5M. It can be seen that even with DistilBert RAILD-TransE trained on Wikidata5M outperforms KEPLER and MLMLM w.r.t. almost all metrics.

4.4.3 ***Results in truly-inductive setting.*** Table 5 presents the results obtained on Wikidata68K (see Section 4.1 for details). The set of training, validation, and test relations are mutually exclusive. Moreover, 89% of validation entities and 74% of test entities are not seen during training. For datasets like Wikidata68K, it is not possible to just randomly initialize the relations (i.e., it is required for an LP model to have features for relations). Therefore, in order to assess the capability of the proposed model RAILD on such a challenging dataset (Wikidata68K), the baselines discussed in Section 4.2 are created. The best results on this dataset are obtained with RAILD-TransE w.r.t. all the metrics.

As compared to the other datasets, the results obtained on Wikidata68K, in general, are low. This is mostly attributed to the nature of the dataset as explained above, i.e., the relations sets in the train validation and test sets being 100% mutually exclusive. Moreover, the WeiDNeR algorithm is applied to the training set, the validation set, and the test set separately so as to avoid generating features using unseen graphs for training. As this is the first work, to the best of our knowledge, to ever make an attempt to perform LP with unseen relations, it would facilitate further research in the community to redirect the focus to unseen relations as well as entities.

4.4.4 ***Ablation studies.*** As the results given in Table 3 for the datasets FB15K-237, WN18RR, and Wikidata5M indicate, according to almost all the metrics, combining text-based and graph-based features for relations (i.e., RAILD-TransE) provides better results than using them separately. Moreover, RAILD-TransE(w/o text) model variant which uses only graph-based relation features is competitive with its counterpart text-based variant RAILD-TransE(w/o feat) and specially on Wikidata5M it even provides slightly better hits@1 and hits@10 results than RAILD-TransE(w/o feat). This indicates that the RAILD-TransE(w/o text) could be used in cases where KGs do not contain textual descriptions for their relations. Similarly, combining the two kind of features for Wikidata68K provides better results w.r.t. hits@10 and equal or competitive results w.r.t. the other metrics as compared to RAILD-TransE(w/o feat).

Note that both RAILD-TransE(w/o feat) and RAILD-TransE(w/o txt) outperform the BLP models which share the same scoring function. For instance, RAILD-TransE(w/o txt) which uses only graph-based features for relations outperforms all the baselines including BLP-TransE [4] which randomly initializes relations, on both datasets FB15K-237 and WN18RR w.r.t. almost all the metrics.

4.4.5 ***Additional Experiments.*** In addition to the experiments discussed above further experiments are also performed to compare the performance of the proposed model with QBLP [1] which is an inductive LP model developed for hyper-relational KG. The same set of optimal hyperparameter values from FB15K-237 datasets are used.RAILD is compared with QBLP on a WD20K(25) dataset [1] for hyper-relational KG. The dataset statistics considering only the triples (removing qualifiers) are given in Table 2 and the results in Table 6. The performance of our model could be negatively impacted by the fact that the size of this dataset is very small as compared to the other datasets used in this work such as FB15K-237. Moreover, since there are relations that occur only in a few triples, generating relations features using WeiDNeR could not be applied as these relations become outliers, i.e., they could not be linked to any other relation (see Algorithm 1). Therefore, only RAILD-TransE(w/o feat) could be used. Despite the argument stated above, RAILD scores the best Hits@10 as compared to QBLP which makes use of qualifiers.

5 CONCLUSION AND FUTURE WORK

In this work, a novel inductive LP model `RAILD` which handles unseen relations is introduced. It works by fine-tuning pretrained LMs with an LP objective. Textual descriptions of entities and relations are used to generate features for the corresponding entities and

Table 2: LP results on semi-inductive setting on WN18RR and FB15K-237 datasets

		FB15K-237				WN18RR			
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
S/F-inductive models	Glove-BOW*	0.172	0.099	0.188	0.316	0.170	0.055	0.215	0.405
	Glove-DKRL*	0.112	0.062	0.111	0.211	0.115	0.031	0.141	0.282
	BE-BOW*	0.173	0.103	0.184	0.316	0.180	0.045	0.244	0.450
	BE-DKRL*	0.144	0.084	0.151	0.263	0.139	0.048	0.169	0.320
	BLP-TransE*	0.195	0.113	0.213	0.363	0.285	0.135	0.361	0.580
	BLP-DistMult*	0.146	0.076	0.156	0.286	0.248	0.135	0.288	0.481
	BLP-ComplEx*	0.148	0.081	0.154	0.283	0.261	0.156	0.297	0.472
	BLP-SimplE*	0.144	0.077	0.152	0.274	0.239	0.144	0.265	0.435
Ours	Glove-BOW$_t$	0.1464	0.0813	0.1636	0.2681	0.1589	0.0465	0.2085	0.3812
	Glove-DKRL$_t$	0.1131	0.0678	0.1176	0.1990	0.1111	0.0283	0.1362	0.2749
	BE-BOW$_t$	0.1569	0.0857	0.1780	0.2923	0.1810	0.0424	0.2483	0.4529
	BE-DKRL$_t$	0.1385	0.0817	0.1473	0.2477	0.1342	0.0461	0.1636	0.3090
	RAILD-TransE	**0.2163**	**0.1268**	**0.2411**	**0.3974**	0.2909	0.1360	0.3689	0.5997
	RAILD-ComplEx	0.1971	0.1169	0.2121	0.3639	**0.3204**	**0.1772**	**0.3895**	**0.6087**

Table 3: Ablation studies with all 4 datasets using TransE scoring function.

	FB15K-237				WN18RR			
	MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
RAILD-TransE	**0.2163**	**0.1268**	**0.2411**	**0.3974**	**0.2909**	0.1360	**0.3689**	**0.5997**
RAILD-TransE(w/o feat)	0.2130	0.1267	0.2363	0.3872	0.2906	**0.1377**	0.3672	0.5944
RAILD-TransE(w/o txt)	0.2030	0.1168	0.2258	0.3777	0.2855	0.1312	0.3640	0.5945
	Wikidata68K				Wikidata5M			
RAILD-TransE	0.0285	0.0059	**0.0283**	**0.0688**	**0.4551**	0.2200	**0.6345**	**0.8489**
RAILD-TransE(w/o feat)	**0.0300**	**0.0077**	**0.0283**	0.0661	0.4529	0.2274	0.6190	0.8376
RAILD-TransE(w/o txt)	0.0137	0.0014	0.0130	0.0320	0.4522	**0.2304**	0.6163	0.8378

Table 4: LP results on Wikidata5M dataset using DistilBERT instead of BERT for RAILD models

	MRR	Hits@1	Hits@3	Hits@10
Glove-BOW*	0.343	0.092	0.531	0.756
Glove-DKRL*	0.362	0.082	0.586	0.798
BE-BOW*	0.282	0.077	0.403	0.660
BE-DKRL*	0.322	0.097	0.474	0.720
BLP-TransE*	**0.478**	**0.241**	**0.660**	**0.871**
KEPLER [25]	0.402	0.222	0.514	0.730
MLMLM [3]	0.284	0.226	0.285	0.348
RAILD-TransE (**DistilBERT**)	0.4551	0.2200	0.6345	0.8489

relations. Moreover, a novel algorithm, i.e., WeiDNeR, is proposed to generate a directed and weighted relation-relation network given a KG. The results of the extensive experiments indicate that using graph structure information to generate relations brings an improvement over models which randomly initialize relations. Moreover, for relations with textual descriptions, it is possible to use the structured information to encode relations and hence, learn embeddings for unseen relations. As a future direction, the proposed model will be adapted to hyper-relational KGs. Moreover, the WeiDNeR algorithm will be further investigated for the LP task where few-shot relations exist.

Table 5: LP results on Wikidata68K datasets

	MRR	Hits@1	Hits@3	Hits@10
Glove-BOW$_t$	0.0119	0.0005	0.0146	0.0295
Glove-DKRL$_t$	0.0031	0.0005	0.0029	0.0064
BE-BOW$_t$	0.0184	0.0005	0.0225	0.0474
BE-DKRL$_t$	0.0055	0.0006	0.0052	0.0125
RAILD-TransE	**0.0285**	**0.0059**	**0.0283**	**0.0688**
RAILD-ComplEx	0.0157	0.0027	0.0125	0.0351

Table 6: LP results with semi-inductive setting on WD20K(25) dataset. #QP denotes the number of qualifiers per statement.

	#QP	MRR	Hits@1	Hits@10
BLP-TransE*	0	0.1245	0.0598	0.2343
QBLP*	0	0.1702	0.0882	0.2950
QBLP*	2	0.2036	0.1177	0.3226
QBLP*	4	**0.2105**	**0.1232**	0.3009
QBLP*	6	0.1950	0.1114	0.3160
RAILD-TransE (w/o feat)	0	0.1586	0.0761	**0.3313**

REFERENCES

[1] Mehdi Ali, Max Berrendorf, Mikhail Galkin, Veronika Thost, Tengfei Ma, Volker Tresp, and Jens Lehmann. 2021. Improving Inductive Link Prediction Using Hyper-Relational Facts. In *SEMWEB*.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-Relational Data. In *NIPS*.

[3] Louis Clouâtre, Philippe Trempe, Amal Zouaq, and Sarath Chandar. 2021. MLMLM: Link Prediction with Mean Likelihood Masked Language Model. In *Findings of ACL/IJCNLP (Findings of ACL, Vol. ACL/IJCNLP 2021)*. Association for Computational Linguistics, 4321–4331.

[4] Daniel Daza, Michael Cochez, and Paul T. Groth. 2021. Inductive Entity Representations from Text via Link Prediction. *Proceedings of the Web Conference 2021* (2021).

[5] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[6] Xin Luna Dong, Evgeniy Gabrilovich, Geremy Heitz, Wilko Horn, Ni Lao, Kevin Murphy, Thomas Strohmann, Shaohua Sun, and Wei Zhang. 2014. Knowledge Vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. 601–610. http://www.cs.cmu.edu/~nlao/publication/2014.kdd.pdf Evgeniy Gabrilovich Wilko Horn Ni Lao Kevin Murphy Thomas Strohmann Shaohua Sun Wei Zhang Geremy Heitz.

[7] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. 2021. LiterallyWikidata - A Benchmark for Knowledge Graph Completion using Literals. In *ISWC*.

[8] Genet Asefa Gesese, Russa Biswas, Mehwish Alam, and Harald Sack. 2019. A Survey on Knowledge Graph Embeddings with Literals: Which model links better Literal-ly? *arXiv preprint arXiv:1910.12507* (2019).

[9] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable Feature Learning for Networks *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 855–864. https://doi.org/10.1145/2939672.2939754

[10] Takuo Hamaguchi, Hidekazu Oiwa, Masashi Shimbo, and Yuji Matsumoto. 2017. Knowledge Transfer for Out-of-Knowledge-Base Entities : A Graph Neural Network Approach. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 1802–1808. https://doi.org/10.24963/ijcai.2017/250

[11] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1025–1035.

[12] Shaoxiong Ji, Shirui Pan, E. Cambria, P. Marttinen, and Philip S. Yu. 2021. A Survey on Knowledge Graphs: Representation, Acquisition and Applications. *IEEE transactions on neural networks and learning systems* PP (2021).

[13] Seyed Mehran Kazemi and David Poole. 2018. SimplE Embedding for Link Prediction in Knowledge Graphs. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 4289–4300.

[14] Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. 2018. Fine-Grained Evaluation of Rule- and Embedding-Based Systems for Knowledge Graph Completion. In *SEMWEB*.

[15] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). http://arxiv.org/abs/1301.3781

[16] Ali Sadeghian, Mohammadreza Armandpour, Patrick Ding, and Daisy Zhe Wang. 2019. *DRUM: End-to-End Differentiable Rule Mining on Knowledge Graphs*. Curran Associates Inc., Red Hook, NY, USA.

[17] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR* abs/1910.01108 (2019). arXiv:1910.01108 http://arxiv.org/abs/1910.01108

[18] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *International Conference on Learning Representations*.

[19] Komal Teru, Etienne Denis, and Will Hamilton. 2020. Inductive Relation Prediction by Subgraph Reasoning. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 9448–9457. https://proceedings.mlr.press/v119/teru20a.html

[20] Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*.

[21] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction *(ICML'16)*. JMLR.org, 2071–2080.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[23] Peifeng Wang, Jialong Han, Chenliang Li, and Rong Pan. 2019. Logic Attention Based Neighborhood Aggregation for Inductive Knowledge Graph Embedding. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence* (Honolulu, Hawaii, USA) *(AAAI'19/IAAI'19/EAAI'19)*. AAAI Press, Article 878, 8 pages. https://doi.org/10.1609/aaai.v33i01.33017152

[24] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering* 29, 12 (2017), 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499

[25] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhiyuan Liu, Juan-Zi Li, and Jian Tang. 2021. KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation. *Transactions of the Association for Computational Linguistics* 9 (2021), 176–194.

[26] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation Learning of Knowledge Graphs with Entity Descriptions. *Proceedings of the AAAI Conference on Artificial Intelligence* 30, 1 (Mar. 2016). https://ojs.aaai.org/index.php/AAAI/article/view/10329

[27] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. In *International Conference on Learning Representations (ICLR)*.

[28] Fan Yang, Zhilin Yang, and William W. Cohen. 2017. Differentiable Learning of Logical Rules for Knowledge Base Reasoning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 2316–2325.