Support and Centrality: Learning Weights for Knowledge Graph Embedding Models

Gengchen Mai, Krzysztof Janowicz, and Bo Yan

STKO Lab, University of California, Santa Barbara, CA, USA

Abstract. Computing knowledge graph (KG) embeddings is a technique to learn distributional representations for components of a knowledge graph while preserving structural information. The learned embeddings can be used in multiple downstream tasks such as question answering, information extraction, query expansion, semantic similarity, and information retrieval. Over the past years, multiple embedding techniques have been proposed based on different underlying assumptions. The most actively researched models are translation-based which treat relations as translation operations in a shared (or relation-specific) space. Interestingly, almost all KG embedding models treat each triple equally, regardless of the fact that the contribution of each triple to the global information content differs substantially. Many triples can be inferred from others, while some triples are the foundational (basis) statements that constitute a knowledge graph, thereby *supporting* other triples. Hence, in order to learn a suitable embedding model, each triple should be treated differently with respect to its information content. Here, we propose a data-driven approach to measure the information content of each triple with respect to the whole knowledge graph by using rule mining and PageRank. We show how to compute triple-specific weights to improve the performance of three KG embedding models (TransE, TransR and HolE). Link prediction tasks on two standard datasets, FB15K and WN18, show the effectiveness of our weighted KG embedding model over other more complex models. In fact, for FB15K our TransE-RW embeddings model outperforms models such as TransE, TransM, TransH, and TransR by at least 12.98% for measuring the *Mean Rank* and at least 1.45% for *HIT@10*. Our HolE-RW model also outperforms HolE and ComplEx by at least 14.3% for *MRR* and about 30.4% for *HIT@1* on FB15K. Finally, TransR-RW show an improvement over TransR by 3.90% for *Mean Rank* and 0.87% for *HIT@10*.

Keywords: Knowledge Graph Embedding · Rule Mining · PageRank.

1 Introduction

A knowledge graph (KG) is a data repository that describes entities and their relationships across domains according to some schema, e.g., an ontology, and is typically organized in the form of a graph, e.g., a directed multi-relational graph [11], such that the nodes represent (real-world) entities and edges represent their relations. As argued by Paulheim [9] there is no commonly agreed upon formal definition of the term nor a common technology stack. Examples range from the Google Knowledge Graph, Microsoft's Satori, and Freebase to KGs based on W3C technologies such as DBpedia, YAGO, and Wikidata. In fact, one can consider the entire Linked Data cloud as a global, densely interlinked knowledge graph. A statement in such KG is represented in the form of a triple. In the Linked Data community, these triples are often referred to as *subject-predicate-object* triple, while the knowledge graph embeddings community has settled on the term *head-relation-tail*; which we will use throughout this work to ease comparison to previous research. To give a concrete example, such triples may encode the statement that Santa Barbara is part of California (`dbr:Santa_Barbara,_California`, `dbo:isPartOf`, `dbr:California`) or that Santa Barbara has a certain population count (`dbr:Santa_Barbara,_California`, `dbo:populationTotal`, `88410^^xsd:integer`). In the first case, the relation is a so-called object property, while the second case shows a datatype property.

Similar to word embedding which encodes each word as a dense continuous vector, knowledge graph embedding [8, 1, 12] aims at representing components of a knowledge graph including entities and relations into continuous vectors or matrices while preserving the structural information of the KG. Those learned entities and relations embeddings can be used in multiple downstream tasks such as KG completion [5], relation inference, relation extraction, knowledge fusion, question answering, query expansion, information extraction, information retrieval [6], and recommender system. Over the past years, multiple embedding techniques have been proposed based on different assumptions. The most actively researched category are translation-based models including models such as *TransE* [1], *TransH* [12], *TransM* [3] and *TransR* [5] which treat relations as translation operations in a shared (or relation specific) space. Recently, methods that measure the plausibility of triples by matching the latent semantics of entities and relations have been proposed such as *HolE* [7] and *ComplEx* [10]. [2]. Interestingly, most work on learning knowledge graph embeddings focuses entirely on object properties, and, therefore, we will restrict our examples and model to those as well[1]. Furthermore, most KG embedding models treat all triple equally, despite the fact that their information content, i.e., their contribution to the overall graph, differers substantially. Some triples act as foundational (basis) statements that cannot be reconstructed from others, while most other triples can be inferred. Put differently, the first kind of triples offer *support* for the second kind. Consequently, in order to emphasize the information content contribution of each triple to the knowledge graph and to learn a suitable embedding model, each triple should be weighted differently.

The research contributions of our work are as follows: we proposed a data-driven approach to measure the information content of each triple with respect to the entire knowledge graph. We apply rule mining and PageRank to estimate the *inference structure* of the current KG and derive the information content of each triple. Rule mining, here AMIE+ [4], enables us to measure the

[1] Counter-examples include KG embedding techniques such as RESCAL which also includes literals [8]

support between triples, while PageRank is used to determine the centrality of triples within a secondary graph created from the left-hand side and right-hand sides of the mined rules. The PageRank scores of this secondary graph are then used to compute triple weights which are then used in the loss function of the KG embedding model. In order to demonstrate the effectiveness of the proposed measure, we modify the translation-based KG embedding model TransE, TransR and the semantic matching model HolE by introducing a triple-specific weighting schema. We use two commonly used datasets, FB15K and WN18, and a link prediction task to show the effectiveness of our weighted model over other models. In fact, for FB15K our TransE-RW[2] embeddings model outperforms models such as TransE, TransM, TransH, and TransR by at least 12.98% for measuring the *Mean Rank* and at least 1.45% for *HIT@10*. Our HolE-RW model also outperforms HolE and ComplEx by at least 14.3% for *MRR* and about 30.4% for *HIT@1* on FB15K. In addition, our TransR-RW model also show an improvement over TransR by 3.90% for *Mean Rank* and 0.87% for *HIT@10*. The smaller improvement of our method over TransR may be caused by the higher complexity (larger number of parameters) of TransR.

The remainder of this paper is structured as follows. First, we discuss work related to our proposed method in Section 2. Then, in Section 3, we present the methods to measure the information content of triples and describe a weighted KG embedding model. Next, experiment results are presented and discussed in Section 4. Finally, in Section 5, we summarize our work.

2 Related Work

Here, we review existing work on knowledge graph embeddings, point out their advantages and disadvantages, and compare them with our proposed models.

KG embedding aims at learning distributional representations for components of a knowledge graph. Entities are usually represented as continuous vectors while relations, i.e., object properties, are typically represented as vectors [1, 12], matrices [5], or tensors. More complex representation methods are more expressive while at the same time suffer from their higher complexity. In order to set up a learning problem, a scoring function $f_r(h, t)$ is defined on each triple/statement (h, r, t) which measures the accuracy of translation or the probability of the correctness of the current triple.[3] Finally, a loss function is defined to set up an optimization problem. In order to learn meaningful representations of entities and relations, we aim at minimize the loss while maximize the total plausibility of the observed triples.

Most KG embedding models treat a knowledge base as a collection of triples $S^+ = \{(h, r, t)\}$ and take each triple as one training sample. According to [11], KG embedding models can be classified into two groups: 1) *translation-based models* (e.g. *TransE*, *TransH*, *TransR*, and *TransD*) and 2) *semantic matching*

[2] **R**ule-supported **W**eights.

[3] Recall that r stands for a given relation, h for head, i.e., a triple's subject, and t for tail, i.e., an entity in the object position.

models (e.g. *RESCAL*, *HolE* [7], and ComplEx [10]). We will focus on three models from these two groups: *TransE*, *TransR* and *HolE*.

Translation-based models treat relations as translation operations on the entity space or a relation specific space. The first and most well-known translation-based model is *TransE* [1]. The idea is inspired by the linguistic regularities discovered among the learned word embeddings. For example, the relationship between `Angola` and `Kwanza` is similar to the relationship between `Iran` and `Rial` which can be expressed by an equation of their corresponding word vectors: $w_{Rial} \approx w_{Kwanza} - w_{Angola} + w_{Iran}$ or $w_{Angola} - w_{Kwanza} \approx w_{Iran} - w_{Rial}$. A hidden translation vector is assumed to operate between `Angola` and `Kwanza` which represent the `currency` relation between a country and the currency it uses. As an analogy, given a triple (h, r, t), *TransE* assumes that relation r is an explicit translation operation which translates the head entity h to the tail entity t. In other words, it assumes $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when (h, r, t) holds. Scoring is defined as the distance between $\mathbf{h} + \mathbf{r}$ and $\mathbf{t}$. In Eq. 1, $\parallel . \parallel$ can be L_1- or L_2-norm.

$$f_r(h,t) = \parallel \mathbf{h} + \mathbf{r} - \mathbf{t} \parallel \tag{1}$$

Although *TransE* is effective at modeling one-to-one relations, the assumption that $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ when (h, r, t) holds is less suitable when dealing with one-to-many, many-to-one, and many-to-many relations. It also has difficulty handling reflexive and transitive relations. Based on the observation of these limitations of *TransE*, many translation-based models have been proposed to address these issues. *TransH* projects head entity h and tail entity t into the relation specified hyperplane which is defined by the norm vector $\mathbf{u_r}$ of the current relation r. Then the score function is defined as the distance between $(\mathbf{h} - \mathbf{u_r}^\top \mathbf{h} \mathbf{u_r}) + \mathbf{r}$ and $(\mathbf{t} - \mathbf{u_r}^\top \mathbf{t} \mathbf{u_r})$ in this hyperplane. In Eq. 2, $\parallel . \parallel_2^2$ represents the square of L_2-norm.

$$f_r(h,t) = \parallel (\mathbf{h} - \mathbf{u_r}^\top \mathbf{h} \mathbf{u_r}) + \mathbf{r} - (\mathbf{t} - \mathbf{u_r}^\top \mathbf{t} \mathbf{u_r}) \parallel_2^2 \tag{2}$$

TransR and *TransD* share a similar idea as *TransH*; however, rather than project entities into hyperplanes, they introduce relation-specific spaces.

Besides allowing different relation-specific embeddings for each entity, another line of research is relaxing the overly restrictive requirement of $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$. *TransM* associates each training triple (h, r, t) with a weight w_r which represents the degree of mapping of the corresponding relation r (See Eq. 3). The weight w_r is calculated by using 1) the average number of head entities per tail entity, denoted by $h_r pt_r$ (head per tail) and 2) the average number of tail entities per head entity, denoted by $t_r ph_r$ (tail per head). This means that a triple will receive lower weight if its relation r has more complex mapping properties. Our proposed method is similar to *TransM* in the sense that both of them give a weight to each triple. However, *TransM* uses the same weight for all triples with the same relation, while our method given each triple a different weight according to its information content wrt. the KG. We will show that this substantially improves over the results reported for *TransM*. It also addresses the issue that *TransM* essentially simply puts more weight on those triples that are more in line with *TransE's* underlying $\mathbf{h} + \mathbf{r} \approx \mathbf{t}$ assumption.

$$f_r(h,t) = w_r \parallel \mathbf{h} + \mathbf{r} - \mathbf{t} \parallel = \frac{1}{log(h_r pt_r + t_r ph_r)} \parallel \mathbf{h} + \mathbf{r} - \mathbf{t} \parallel \quad (3)$$

Another groups of KG embeddings models, so-called semantic matching models, measure the plausibility of triples by matching the latent semantics of entities and relations. Different models capture the interactions between latent factors of entity and relation embeddings in different ways. Here we discuss *HolE* as an example. The scoring function of Holographic Embeddings (*HolE*) is shown in Eq. 4. By using circular correlation operation $\star$ to compose entity representations, *HolE* is able to capture rich interactions between entity embeddings while maintaining its efficiency and simplicity. The non commutativeness of $\star$ also make it keep the asymmetry of the relations. σ is the logistic function and $-$ is used to align the interpretation of the scoring function with other models which implies that smaller score indicates a higher plausibility of the triple.

$$f_r(h,t) = -\sigma\big(\mathbf{r}^{\intercal}(\mathbf{h} \star \mathbf{t})\big) = -\sigma\left(\sum_{i=0}^{d-1} [\mathbf{r}]_i \sum_{k=0}^{d-1} [\mathbf{h}]_k \cdot [\mathbf{t}]_{(k+i) \bmod d}.\right) \quad (4)$$

3 Methodology

Consider the problem of measuring the information content contribution of each triple to a KG; intuitively a triple $T_i = (h_i, r_i, t_i)$ will have a higher contribution if other triples can be inferred from it. These inferred triples can be derived either purely based on T_i or based on a conjunction condition including T_i and other triples. Hence, one way to interpret the information content contribution of a triple T_i is that if T_i is excluded from the current KG, a certain number of triples cannot be inferred from it any longer. For example, as for DBpedia, if the triple T_i (`dbr:Santa_Barbara,_California`, `dbo:isPartOf`, `dbr:California`) is excluded from DBpedia, hundreds of triples which can be inferred from it based on the transitive property of parthood, e.g., that University of California, Santa Barbara is part of California given that we know that it is located in Santa Barbara, will no longer be reachable. Put differently, number of inferred triples of triple T_i is a measure of the information content contribution of T_i to the KG. However, there are some shortcomings to such measure.

First, enumerating each triple and executing inferences on the entire KG may be computationally complex given a large graph and ontology (particularly using an expressive description logic). Second, this type of reasoning also requires a formal ontology in the first place and thus only applies to Semantic Web style knowledge graphs that use ontologies that explicitly make use of language features such as subclassing. In addition, *isolated triples* become a substantial problem. Isolated triples are triples in a KG which can neither be used to infer any another triples nor can be inferred by any triples. By using the method above, these triples will have a very low information content because they cannot infer any triples and excluding them from the KG will not affect the number of inferred triples. However, information theory tells us that those *isolated triples* should have a high information content as they cannot be compressed. Consequently, we need to go beyond the intuitive notion of information content for triples introduced above. At the same time, and to appeal to the broader KG community, we want to work buttom-up first, i.e. not rely on the existence of a strong formal ontology. In order to provide an automatic and general method for measuring the information content of triples, and as will be detailed below, we will use rule mining to estimate the *support* between triples and then measure the centrality of triples within a secondary graph formed by these support relations using PageRank. The result will be individual weights per triple that we will use in the loss function to learn embeddings.

Given a KG (the training dataset) represented as a set of triples $S^+ = \{(h_i, r_i, t_i)\}$. For each triple (h_i, r_i, t_i), its head and tail entity are $h_i, t_i \in E$ (the set of entities) and its relation is $r_i \in L$ (the set of relations). Our model measures the contribution of each triple to the global information content of the KG by investigating the inference relationships among these triples and use this measure to learn a suitable KG embedding model for the current KG. Our method can be divided into four steps: 1) rule mining; 2) rule instantiation; 3) triple inference graph construction and triple weights calculation; and 4) learning a weighted KG embedding model. Fig. 1 illustrates the first three steps of our workflow and each of these four steps will be described in detail below.

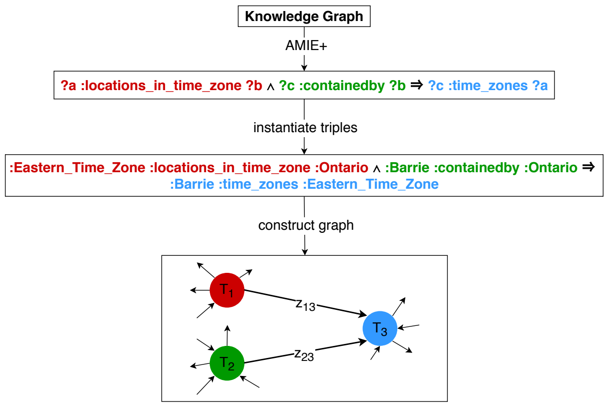
Knowledge Graph
AMIE+
?a :locations_in_time_zone ?b ∧ ?c :containedby ?b ⇒ ?c :time_zones ?a
instantiate triples
:Eastern_Time_Zone :locations_in_time_zone :Ontario ∧ :Barrie :containedby :Ontario ⇒ :Barrie :time_zones :Eastern_Time_Zone
construct graph
T1
T2
T3
z13
z23

Fig. 1. The workflow of computing the information content of each triple in a KG

3.1 Rule Mining

Generally speaking, logical rule mining is a machine learning method to find rules in a KG that describe the common correlations between triples. Modern rule mining systems like AMIE and AMIE+ [4] aim at mining logical rules efficiently from large RDF-based knowledge bases. An atom in a Horn rule R_i is a triple whose subject or/and object is replaced by variables. A Horn rule R_i is composed of a head $r(x, y)$ and a body $\{B_1, B_2, ..., B_n\}$, where the head $r(x, y)$ is a single atom and the body is the conjunction of multiple atoms or just one atom [4]. Eq. 5 shows the general form of a Horn rule R_i where both $r(x, y)$ and B_i are atoms. Note that in $r(x, y)$, r represents a specific relation where x and y are subject and object who can be either real entities or variables. R_i can be abbreviated as $\overrightarrow{B} \Rightarrow r(x, y)$. We utilize AMIE+ for rule mining. The rule in the second box from Fig. 1 is an example of a mined rules from AMIE+.

$$R_i : \; B_1 \; \wedge \; B_2 \; \wedge ... \; \wedge \; B_n \; \Rightarrow \; r(x, y) \tag{5}$$

AMIE+ requires three parameters: a threshold *minHC* of the head coverage of the mined rules, a maximum rule length *maxLen*, and a threshold *minConf* for the PCA confidence score. We will describe each of them in detail below.

Given a rule $R_i : \overrightarrow{B} \Rightarrow r(x, y)$, the *support* of R_i is the number of correct predictions of rule R_i in the current KG, or, in other words, the number of distinct pairs of head and tail entities $\#(x, y)$ in the rule head among all the instantiations of the current rule. *Rule instantiation* is the process to substitute the variables in a rule with entities (constants) in the KG such that all instantiated atoms/triples in the rule head and rule body are in the KG. The result rules are called *grounded rules*.

Based on the definition of *rule instantiation*, a naive way to define how good a mined rule is can be computed as the number of instantiation of the current rule over the size of the current KG (See Eq. 6). In Eq. 6, $\#(S^+)$ represents the number of triples in S^+. We refer to it as *frequency* in the following. Instead of using *frequency*, AMIE+ uses *head coverage* which is defined as the *support* of a rule divided by the number of statements with rule head relation r (See Eq. 7). Each rule from AMIE+ has a head coverage value. The parameter *minHC* controls the minimum head coverage value of the mined rules such that all rules with head coverage less than *minHC* will be excluded. The default is 0.01.

$$freq(R_i) = \frac{\#(instatiate(\overrightarrow{B} \Rightarrow r(x, y)))}{\#(S^+)} \tag{6}$$

$$hc(R_i) = \frac{support(\overrightarrow{B} \Rightarrow r(x, y))}{\#(r)} \tag{7}$$

Second, *maxLen* restricts the maximum length of the mined rules. The length of rules is defined as the number of atoms in the rule including head and body. For example, the rule in the second box in Fig. 1 has length 3. A longer rule length means a larger rule search space for AMIE+. The default *maxLen* is 3.

Third, *minConf* controls the minimum PCA confidence scores of mined rules. Head coverage does not take into consideration false predictions of the mined rule, while the confidence scores provide a way to obtain counterexamples for the rule mining. The mined rules are associated with two confidence scores - *standard confidence score* (Closed-World Assumption) and *PCA confidence score* (Partial Completeness Assumption) - which describe how confident AMIE+ is about the currently mined rules based on the observed triples in the KG. The higher the confidence scores is, the more likely the rule will make correct predictions. Further detail for these two confidence scores, are described in [4]. AMIE+ utilizes the PCA confidence scores and excludes rules whose confidence scores are less than *minConf*, with a default of 0.1.

Head coverage, standard confidence score, and PCA confidence score are three ways to represent the inference power of a rule. Although these three parameters can take other values than the default values, [4] does not suggest to do so. Hence, our model utilizes the default parameter setting of AMIE+.

3.2 Rule Instantiation

After rule mining is applied to get the inference relationship between triples, the mined rules are instantiated to get grounded rules. As per the definition of *rule instantiation* above, variables in each atom need to be instantiated by entities in the KG such that these entities satisfy both the rule head and rule body. As for rule R_k in the second box from Fig. 1, the instantiating process can be understood as sending a SPARQL SELECT query to the original KG in which atoms in both the rule head and rule body are the graph patterns in this query.

The third box in Fig. 1 shows one example of the grounded rule for R_k. Note that each mined rule is associated with four rule predication quality/correctness measures: *frequency*, *head coverage*, *standard confidence score*, and *PCA confidence score*. These measures can also be used in their grounded rules to indicate the likelihood of correct predication.

3.3 Triple Inference Graph Construction & Weights Calculation

Given a rule $R_h : B_1 \wedge B_2 \Rightarrow B_3$, one of its grounded rules is $R_{hj} : T_1 \wedge T_2 \Rightarrow T_3$ with *frequency* f_{freq}, *head coverage* f_{hc}, *standard confidence score* f_{cwa}, and *PCA confidence score* f_{pca}. After applying rule mining and rule instantiation, we are able to obtain the inferencing relationships between different triples. In order to provide a holistic view of these rules and the relationships between triples, we construct a triple inference graph based on these grounded rules from different rules. Each triple (statement) is represented as a node and each directed edge e_{ij} from node T_i to node T_j indicates that statement T_i infers statement T_j. As for $R_{hj} : T_1 \wedge T_2 \Rightarrow T_3$, two edges can be obtained: e_{13} from nodes T_1 to T_3 and e_{23} from nodes T_2 to T_3. The weights of each edges are derived from the four rule predication correctness measures f_{freq}, f_{hc}, f_{cwa}, and f_{pca}. Note that one triple T_j can be the rule heads of many grounded rules which may or may not instantiated from the same rules. As for those grounded rules, another triple T_i can appear in the rule bodies of some of them. Let GR_1, GR_2, ..., GR_k, ..., GR_r be all grounded rules which are instantiated from the mined rules from AMIE+. Let f_1, f_2, ..., f_k, ..., f_r be one rule predication correctness measure from those four measures. All grounded rules should use the same type of measures. Let L_1, L_2, ..., L_k, ..., L_r be the rule lengths of those grounded rules. α_{ik} is an indicator function to indicate whether triple T_i appear in the rule body of grounded rule GR_k ($\alpha_{ik} = 1$ when T_i is in the rule body of GR_k; 0 otherwise). β_{jk} is an indicator function to indicate whether triple T_j is the rule head of grounded rule GR_k ($\beta_{jk} = 1$ when T_j is the rule head of GR_k; 0

otherwise). Then the equation to calculate edge weight z_{ij} of e_{ij} from triple T_i to triple T_j is shown in Eq. 8.

$$z_{ij} = \sum_{i=1}^{r} \alpha_{ik}\beta_{jk}\frac{f_k}{L_k - 1} \tag{8}$$

Following the method above, we construct a secondary triple inference graph based on those grounded rules. The third and fourth boxes of Fig. 1 illustrate the graph construction process. In this triple inference graph, the more incoming links a triple has, the more likely this statement is able to be inferred by other statements which implies that this triple has less information, at least from an information theoretic compression perspective. Information content is calculated as the negative logarithm of the probability. The probability in this context is the probability of inferencing a triple (statement) in our triple inference graph. In order to obtain the inferencing probability of each triple in the graph, we model it as a stochastic process and more specifically a Markov Chain. Each state in the Markov Chain corresponds to a node in our graph and the transition probability between states are determined by the number of links/edges and edge weights between nodes. E.g., if there are 5 outgoing links/edges from node T_i and one of them connects to node T_j, then the transition probability from node T_i to T_j is 0.2 if those 5 edges have equal weights. The stationary distribution of this Markov Chain gives us the inferencing probability of each triple in the graph.

However, this method only works when the graph (Markov Chain) is strongly connected, meaning every node can be reach from any other node in the graph, otherwise the stationary distribution may not be unique. This requirement translated into our case would imply that every statement can be inferred by any other statement, which is less likely to be true. Disconnected components, dangling links and loops are common in our inferencing graph. To deal with these cases, we use the PageRank algorithm which solves these issues by providing a teleport probability which allows the random walker to jump to a random node in the graph with a certain probability at each time step. In this case, the stationary distribution of the Markov Chain with the teleport probability becomes unique again. We use this stationary distribution as our inferencing probability to calculate the information content. Disconnected or isolated triples will have a lower inferencing probability, thus possessing richer information content.

In this work, an edge weighted PageRank is applied to the constructed triple inference graph. The final weight of each triple is calculated based on the PageRank value PR_i of each node/triple (See Eq. 9). Here $\frac{\#(S^+)}{\sum -log_2(PR_i)}$ is a normalization factor to make the mean value of result triple weights to be 1.0.

$$w_i = -log_2(PR_i) \times \frac{\#(S^+)}{\sum -log_2(PR_i)} \tag{9}$$

3.4 Learning A Weighted Knowledge Graph Embedding Model

After obtaining the triple weights, we deploy a weighted KG embedding model based on multiple existing models (*TransE*, *TransR*, and *HolE*). The training dataset is the observed triples in S^+. The plausibility scoring function of a triple $T_i = (h_i, r_i, t_i) \in S^+$ with weight w_i can be any scoring function of any translation-based models (*TransE*, *TransH*, *TransR*, and *TransD*) or semantic matching models (*TATEC*, *DistMult*, and *HolE*) as long as these models use pairwise ranking loss functions to set up the learning task. The plausibility scoring functions of *TransE* and *HolE* are shown in Eq. 1 and 4. We will denote the weighted version of *TransE*, *TransR* and *HolE* as *TransE-RW*, *TransR-RW* and *HolE-RW*.

To learn the KG embedding, we use the pairwise ranking loss function as other models do. However, in the loss function, we multiply w_i with the subtraction value between the plausibility score of triple $T_i = (h_i, r_i, t_i)$ and the score of one of T_i's corrupted triples $T_i^{'} = (h_i^{'}, r_i, t_i^{'})$ (See Eq. 10). The intuition is that as in the margin idea in support vector machine, the pairwise ranking function aims to make the observed triples well separated from the corrupted triples in the plausibility score space and $f_r(h_i, t_i) - f_r(h_i^{'}, t_i^{'})$ is a measure of the distinction degree or distance for triple T_i. Since different triples have different contribution to the global information content of the KG, the loss function should consider T_i more if it has larger information content.

$$\mathcal{L} = \sum\nolimits_{(h_i, r_i, t_i) \in S^+} \sum\nolimits_{(h_i^{'}, r_i, t_i^{'}) \in S^-_{(h_i, r_i, t_i)}} \big[\gamma + w_i\big(f_r(h_i, t_i) - f_r(h_i^{'}, t_i^{'})\big)\big]_+ \quad (10)$$

The set of corrupted triples for triples $T_i^{'} = (h_i^{'}, r_i, t_i^{'})$ is constructed according to Eq. 11. Two negative sampling methods are used: 1) replacing either the triple's head or tail entity with a random entity (denoted as *unif.*) and 2) the negative sampling method proposed by [12] which uses head per tail h_rpt_r and tail per head t_rph_r (denoted as *bern.*). The second method will corrupt a triple by replacing the head with probability $\frac{t_rph_r}{t_rph_r + h_rpt_r}$ and corrupt a triple by replacing the tail with probability $\frac{h_rpt_r}{t_rph_r + h_rpt_r}$.

$$S^-_{(h_i, r_i, t_i)} = \big\{(h_i^{'}, r_i, t_i) \mid h_i^{'} \in E\big\} \cup \big\{(h_i, r_i, t_i^{'}) \mid t_i^{'} \in E\big\} \quad (11)$$

As for *TransE-RW*, the same constraint as *TransE* has been applied during embedding model training which restricts the L_2-norm of the embeddings of entities to be 1. It prevents the loss from being trivially minimized by enlarging the norms of the embeddings of entities. We follow the same training process of *TransE*. First, the entity embedding matrix $\mathbf{E}$ and relation embedding matrix $\mathbf{L}$ are initialized by using uniform distribution $uniform(-\frac{6}{\sqrt{K}}, \frac{6}{\sqrt{K}})$ where K is the embedding dimension. Then, the relation embeddings are normalized before the training process begins. In each iteration, L_2 normalization has been applied to entity embeddings before gradient decent. The *adam* optimizer is used for the optimization. The same process is also utilized for *HolE-RW*.

4 Experiment

Two standard datasets - FB15K and WN18 - are used to evaluate all models. FB15K and WN18 are standard datasets which have been used to evaluate KG embedding models [1, 12, 5]. WN18 is extracted from WordNet in which entities are word senses and relations correspond to the lexical relationships between word senses. FB15K is a subset extracted from Freebase in which entities have at least 100 mentions in Freebase and also appear in Wikilinks dataset. Given the richer relational structure of FB15K, we expect a larger rule set, and, thus, a more visible difference to the baseline due to the learned weights.

First, we calculate the weights for each triple in the training datasets of those two datasets as described above. As for the rule mining step, we use AMIE+ [4] as the rule mining system. Next, we construct the triple inference graphs based on these mined rules and apply edge weighted PageRank. As expected AMIE+ was able to identify substantially more rules for the Freebase dataset (41195) than for WordNet (140). The triple weights for each triple from the training datasets of FB15K and WN18 are calculated based on the PageRank values. Since there are four types of rule predication correctness measures (*frequency* f_{freq}, *head coverage* f_{hc}, *standard confidence score* f_{cwa}, and *PCA confidence score* f_{pca}), four different triple inference graphs can be constructed for each dataset based on different measures. This results in four different types of triple weights for each dataset which are indicated by $freq$, hc, cwa, and pca. For each dataset, we compute Spearman's correlation coefficients between each pair of triples weights. Table 1 and Table 2 show the Spearman's correlation coefficients matrix of triples weights on WN18 and FB15K. As can be seen from Table 1 and Table 2, the calculated triple weights from different methods are highly correlated (at least 0.704 for WN18 and 0.788 for FB15K).

Table 1. Spearman's correlation coefficients between weights calculated by different rule predication correctness measures on WN18

ρ	freq	hc	cwa	pca
freq	1	0.704	0.899	0.879
hc	-	1	0.790	0.779
cwa	-	-	1	0.889
pca	-	-	-	1

Table 2. Spearman's correlation coefficients between weights calculated by different rule predication correctness measures on FB15K

ρ	freq	hc	cwa	pca
freq	1	0.788	0.877	0.855
hc	-	1	0.805	0.848
cwa	-	-	1	0.972
pca	-	-	-	1

The computed triple weights are used in *TransE-RW*, *TransR-RW* and *HolE-RW* models as shown in Equation 10. To show the effectiveness of our weighted model, we empirically evaluate *TransE-RW*, *TransR-RW* and *HolE-RW* together with related models by using a common link prediction task on these standard datasets by following the evaluation protocol of [1]. Given a correct triple $T_k = (h_k, r_k, t_k)$ from the testing dataset of FB15K (or WN18), we replace the head entity h_k (or tail t_k) with all other entities from the dictionary of FB15K (or WN18). If there are n entities in the current dataset, this triple

[4] `https://www.mpi-inf.mpg.de/departments/databases-and-information-systems/research/yago-naga/amie/`

corruption operation will result in n triples in which $n-1$ triples are corrupted triples together with the correct triple T_k. The plausibility scores for each of those n triples can be computed based on the plausibility score functions of *TransE*, *HolE* (See Equation 1 and 4) and *TransR-RW* by using the trained entity and relation embeddings. By ranking the scores in the ascending order, we can get the rank of the original correct triple T_k. Note that some of the corrupted triples may also appear in the KG. For example, as for triple (`dbr:Santa_Barbara,_California`, `dbo:isPartOf`, `dbr:California`), if we replace the head `dbr:Santa_Barbara,_California` with `dbr:San_Francisco`, the result corrupted triple (`dbr:San_Francisco`, `dbo:isPartOf`, `dbr:California`) is still in the DBpedia KG. These false negative samples need to be filtered out. We report both the original rank and the rank after filtering out those false negatives (denote as *Raw* and *Filter*). Aggregated over all triples in the testing dataset of FB15K (or WN18), multiple metrics are reported: 1) the *Mean Rank*; 2) the mean reciprocal rank *MRR*; 3) the proportion of ranks not larger than K (denoted as *HIT@K where K can be 1, 3, 10*). A KG embedding model with lower *Mean Rank*, higher MRR, and higher *HIT@K* is better. Note that different papers report different metrics. So we use different metrics for *TransE-RW*, *TransR-RW* (*Mean Rank*, *MRR*[5], and *HIT@10*) and *HolE-RW* (*MRR*, *HIT@1*, *HIT@3*, and *HIT@10*) to make our results comparable to the related models. RDF2VecGlove [2] also utilizes PageRank to facilitate RDF graph embedding learning. However, it applies PageRank on the original KG while we apply PageRank on the triple inference graph. RDF2VecGlove does not define a plausibility scoring function for each triple which make it difficult to directly apply RDF2VecGlove to the KG completion task.

We implemented the *TransE-RW*, *TransR-RW* and *HolE-RW* models using TensorFlow. Hyperparameters are selected using grid search. The hyperparameters we use for FB15K are: the embedding dimension $K = 50$, the margin $\gamma = 1.0$, distance norm $d = L_1$, and learning rate $\alpha = 0.0001$ for *TransE-RW* ($K = 80$, $\gamma = 1.0$, $d = L_1$, and $\alpha = 0.0001$ for *TransR-RW*; $K = 200$, $\gamma = 1.0$, and $\alpha = 0.002$ for *HolE-RW*). The hyperparameters we use for WN18 are: the embedding dimension $K = 20$, the margin $\gamma = 2.0$, distance norm $d = L_1$, and learning rate $\alpha = 0.0005$ for *TransE-RW* ($K = 30$, $\gamma = 2.0$, $d = L_1$, and $\alpha = 0.001$ for *TransR-RW*; $K = 150$, $\gamma = 1.5$, and $\alpha = 0.00005$ for *HolE-RW*).

Table 3 shows the link prediction results of *TransE-RW* and *TransR*[6] on both *Raw* and *Filter* settings by comparing with other models. *TransE-RW* outperforms other translation-based models on both *Mean Rank* and *HIT@10* on both datasets, i.e., FB15K and WN18. As for FB15K, all of TransE-RW models except *TransE-RW*$_{\text{cwa}}$ (bern) outperform *TransE*, *TransM*, and *TransH* on *HIT@10* with an improvement ranging from **8.23% to 47.98%**.Even for *TransR*, Most TransE-RW models shows a imporove over *TransR* on *HIT@10* for FB15K. All *TransE-RW* models with *unif* negative sampling setting produce

[5] [7] points out that *MRR* is less sensitive to outliers than *Mean Rank*. So we also report *MRR* in *TransE-RW* and *TransR-RW*

[6] Note that we only implement *TransR-RW* on *freq* weight as an example.

Table 3. Link Prediction Result of *TransE-RW* and *TransR-RW* (*unif* indicates using random negative sampling method; *bern* indicates using the method proposed by [12])

DataSet	WN18						FB15K					
Metric	Mean Rank		MRR		HIT@10		Mean Rank		MRR		HIT@10	
	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter	Raw	Filter
TransE [1]	263	251	-	-	75.4	89.2	243	125	-	-	34.9	47.1
TransM [3]	293	281	-	-	75.7	85.4	197	94	-	-	44.6	55.2
TransH (unif.) [12]	318	303	-	-	75.4	86.7	211	84	-	-	42.5	58.5
TransH (bern.) [12]	401	388	-	-	73.0	82.3	212	87	-	-	45.7	64.4
TransR (unif.) [5]	232	**219**	-	-	78.3	91.7	226	78	-	-	43.8	65.5
TransR (bern.) [5]	238	225	-	-	**79.8**	92.0	198	77	-	-	48.2	68.7
TransE-RW$_{\text{freq}}$ (unif.)	298	286	0.361	0.487	77.8	91.4	216	69	0.225	0.422	46.8	69.4
TransE-RW$_{\text{freq}}$ (bern.)	**231**	**219**	**0.391**	**0.516**	78.1	91.0	243	144	0.252	0.424	49.4	67.8
TransE-RW$_{\text{hc}}$ (unif.)	266	253	0.371	0.496	77.1	90.7	212	**67**	0.226	0.420	46.8	68.8
TransE-RW$_{\text{hc}}$ (bern.)	272	260	0.377	0.495	77.3	89.8	235	134	**0.258**	0.444	**50.2**	69.6
TransE-RW$_{\text{cwa}}$ (unif.)	281	269	0.359	0.483	77.0	90.8	213	**67**	0.225	0.418	47.0	69.0
TransE-RW$_{\text{cwa}}$ (bern.)	277	265	0.378	0.486	75.4	86.8	245	149	0.241	0.386	47.2	63.4
TransE-RW$_{\text{pca}}$ (unif.)	292	279	0.353	0.472	76.2	89.6	217	71	0.227	0.423	47.1	**69.7**
TransE-RW$_{\text{pca}}$ (bern.)	318	305	0.375	0.484	75.4	86.9	232	132	0.256	**0.445**	50.1	**69.7**
TransR-RW$_{\text{freq}}$ (unif.)	351	336	0.319	0.448	77.8	**93.4**	230	76	0.173	0.356	44.2	67.1
TransR-RW$_{\text{freq}}$ (bren.)	320	306	0.326	0.442	78.0	92.0	**196**	74	0.230	0.426	48.3	69.3

lower *Mean Rank* than the baseline models like TransE, TransM, TransH, even TransR with improvement ranging from **7.79%** to **12.99**. As for WN18, all of our models except *TransE-RW*$_{\text{cwa}}$ (bern) out perform *TransE*, *TransM*, and *TransH* on *HIT@10* with improvement ranging from **2.46% to 11.06%** while have a slightly lower *HIT@10* than TransR. *TransE-RW*$_{\text{freq}}$ (bern) produces the lowest *Mean Rank* than all other models, while other *TransE-RW* produce comparable results on *Mean Rank* compared to TransE, TransM, and TransH. The results for WN18 are dominated by the very small set of inferred rules. In general, KG are expected to be similar to Freebase, DBpedia, Wikidata, and so forth, for which *TransE-RW* yields substantial improvements over all baselines. We report WordNet results here to stay in line with the literature. Note that TransR has much higher time complexity ($\mathcal{O}(n_eK + n_rK + n_rK^2)$) compared to TransE ($\mathcal{O}(n_eK + n_rK)$). We demonstrate that by including the weighted strategy, even the most simple model such as TransE can outperform a much more complex model such as TransR. To demonstrate the generalization of our weight method, we also implemented *TransR-RW*$_{\text{freq}}$ (unif.)/(bern.). Compared to the original TransR, *TransR-RW*$_{\text{freq}}$ (bern.) provide lower *Mean Rank* and higher *HIT@10* for FB15K and higher *HIT@10* for WN18. The only metric TransR-RW does worse than TransR is *Mean Rank* for WN18 while *TransE-RW*$_{\text{freq}}$ (bern) can beat TransR on this metric. Compared to TransE-RW, TransR-RW does not show a substantial improvement.

Similar to [1, 12], we classify the relations into 1-to-1, 1-to-n, n-to-1, and n-to-n categories according to the head per tail h_rpt_r and tail per head t_rph_r values of each relation. We classify the left side or right side to 1 or n according to the fact whether h_rpt_r (left side) and t_rph_r (right side) is less than 1.5. For example, a given relation is classified as 1-to-n if its h_rpt_r value is less than

Table 4. Link prediction results of *TransE-RW* on FB15K by relation categories

Task	Predicting head (HITS@10)				Predicting tail (HITS@10)			
Relation Category	1-to-1	1-to-n	n-to-1	n-to-n	1-to-1	1-to-n	n-to-1	n-to-n
TransE [1]	43.7	65.7	18.2	47.2	43.7	19.7	66.7	50
TransM [3]	76.8	86.3	23.1	52.3	76.3	29	85.9	56.7
TransH (unif.) [12]	66.7	81.7	30.2	57.4	63.7	30.1	83.2	60.8
TransH (bern.) [12]	66.8	87.6	28.7	64.5	65.5	39.8	83.3	67.2
TransE-RW$_{\text{freq}}$ (unif.)	75.5	88.5	37.5	**70.3**	74	41.6	86.5	72.7
TransE-RW$_{\text{freq}}$ (bern.)	81.1	92.6	27.5	68.2	78.8	34.4	92	71.3
TransE-RW$_{\text{hc}}$ (unif.)	74.1	88.9	**39.1**	69	73.7	42	86.6	71.8
TransE-RW$_{\text{hc}}$ (bern.)	**81.9**	**93.8**	30	70.1	78	36.5	**93.1**	**73.4**
TransE-RW$_{\text{cwa}}$ (unif.)	75	89.2	38.1	69.4	74.9	42	87.6	71.9
TransE-RW$_{\text{cwa}}$ (bern.)	77.7	90.2	23.9	63.2	74.8	29.3	90.5	66.6
TransE-RW$_{\text{pca}}$ (unif.)	75.2	89.2	38.5	70.2	74.5	**43.0**	87.2	72.8

1.5, while its $t_r ph_r$ value is larger than or equal to 1.5. After classifying the relations into these four categories, we aggregate the *HIT@10* by each category for head prediction and tail prediction of *TransE-RW* on FB15K. Table 4 shows the results of *TransE-RW* and compare them with the results from other models. We can see that our *TransE-RW* models outperform other models on *HIT@10* for every relation category in both head and tail prediction.

Table 5. Link prediction results of *HolE-RW*

DataSet	WN18					FB15K				
Metric	MRR		HIT			MRR		HIT		
	Filter	Raw	1	3	10	Filter	Raw	1	3	10
HolE	0.938	0.616	93	**94.5**	94.9	0.524	0.232	40.2	61.3	73.9
ComplEx	0.941	0.587	**93.6**	**94.5**	94.7	0.692	0.242	59.9	75.9	**84**
HolE-RW$_{\text{freq}}$ (unif.)	0.91	0.624	89.5	92.1	93.4	0.702	0.699	69.0	70.0	72.1
HolE-RW$_{\text{freq}}$ (bern.)	0.913	0.645	89.5	92.7	94.0	0.675	0.671	65.8	67.5	70.6
HolE-RW$_{\text{hc}}$ (unif.)	0.932	0.688	92.3	93.6	94.5	0.646	0.64	62.5	64.4	68.2
HolE-RW$_{\text{hc}}$ (bern.)	0.922	0.686	90.8	93.2	94.1	0.705	0.699	69.2	70.4	72.6
HolE-RW$_{\text{cwa}}$ (unif.)	**0.942**	**0.693**	93.5	**94.5**	**95.5**	0.695	0.692	68.3	69.3	71.6
HolE-RW$_{\text{cwa}}$ (bern.)	0.922	0.684	91.0	93.2	93.9	**0.791**	**0.788**	**78.1**	**79.0**	81.1
HolE-RW$_{\text{pca}}$ (unif.)	0.931	0.686	92.3	93.7	94.5	0.635	0.63	61.5	63.4	67.1
HolE-RW$_{\text{pca}}$ (bern.)	0.926	0.688	91.4	93.5	94.4	0.756	0.754	74.6	75.4	77.3

We also report *HolE-RW* performance on the link prediction tasks and compare it with HolE and ComplEx. HolE-RW$_{\text{cwa}}$ (bern.) outperforms HolE and ComplEx by at least 14.3% for *MRR*, about 4.1% for *HIT@3*, and about 30.4% for *HIT@1* on FB15K. HolE-RW$_{\text{cwa}}$ (bern.) and HolE-RW$_{\text{pca}}$ (bern.) can outperform HolE for *HIT@10* on FB15K while ComplEx has the best performance. As for WN18, HolE-RW$_{\text{cwa}}$ (unif.) can outperform both HolE and ComplEx on almost all the metrics while ComplEx outperform it by 0.1% on *HIT@1*. As discussed above, few rules can be derived from WN18 and HolE and ComplEx already achieve 90%+ performance. Hence, we do not see a large change on WN18.

5 Conclusion

In this work, we proposed a bottom-up method to measure the information content of each triple with respect to the whole knowledge graph and implement weighted knowledge graph embedding models based on the idea that not all triples should be weighted equally. Instead, triples that can be used to infer other triples offer support for those, thereby also decreasing their information content. We applied rule mining to derive the inference structures of a knowledge graph and to construct a secondary, directed, weighted graph based on these support relations and their confidence. Next, we apply an edge-weighted PageRank (PR) to this secondary graph to get a centrality score of each triple and then compute its information content as $-log(PR_i)$. To demonstrate the effectiveness of the weighting, we modifed three popular models from different KG embedding model groups and performed link prediction on two standard datasets. The results show that our TransE-RW models outperform other models including TransE, TransM, TransH, and TransR by at least 12.98% for *Mean Rank* and 1.45% for *HIT@10* on FB15K. HolE-RW outperforms HolE and ComplEx by at least 14.3% for *MRR* and about 30.4% for *HIT@1* on FB15K.

References

1. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Advances in neural information processing systems. pp. 2787–2795 (2013)
2. Cochez, M., Ristoski, P., Ponzetto, S.P., Paulheim, H.: Global RDF vector space embeddings. In: ISWC. pp. 190–207. Springer (2017)
3. Fan, M., Zhou, Q., Chang, E., Zheng, T.F.: Transition-based knowledge graph embedding with relational mapping properties. In: Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing (2014)
4. Galárraga, L., Teflioudi, C., Hose, K., Suchanek, F.M.: Fast rule mining in ontological knowledge bases with amie + +. The VLDB Journal **24**(6), 707–730 (2015)
5. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI. vol. 15, pp. 2181–2187 (2015)
6. Mai, G., Janowicz, K., Yan, B.: Combining text embedding and knowledge graph embedding techniques for academic search engines. In: SemDeep-4 (2018)
7. Nickel, M., Rosasco, L., Poggio, T.A., et al.: Holographic embeddings of knowledge graphs. In: AAAI. pp. 1955–1961 (2016)
8. Nickel, M., Tresp, V., Kriegel, H.P.: A three-way model for collective learning on multi-relational data. In: ICML. vol. 11, pp. 809–816 (2011)
9. Paulheim, H.: Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web **8**(3), 489–508 (2017)
10. Trouillon, T., Dance, C.R., Gaussier, É., Welbl, J., Riedel, S., Bouchard, G.: Knowledge graph completion via complex tensor factorization. The Journal of Machine Learning Research **18**(1), 4735–4772 (2017)
11. Wang, Q., Mao, Z., Wang, B., Guo, L.: Knowledge graph embedding: A survey of approaches and applications. IEEE Transactions on Knowledge and Data Engineering **29**(12), 2724–2743 (2017)
12. Wang, Z., Zhang, J., Feng, J., Chen, Z.: Knowledge graph embedding by translating on hyperplanes. In: AAAI. vol. 14, pp. 1112–1119 (2014)