Weight-aware Tasks for Evaluating Knowledge Graph Embeddings

Wei Kun Kong [a], Xin Liu [b], Teeradaj Racharak [a,*], Guanqun Sun [a], Qiang Ma [c] and Le-Minh Nguyen [a]
[a] *School of Information Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan*
E-mails: kong.diison@jaist.ac.jp, racharak@jaist.ac.jp, sun.guanqun@jaist.ac.jp, nguyenml@jaist.ac.jp
[b] *Artificial Intelligence Research Center, National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan*
E-mail: xin.liu@aist.go.jp
[c] *Department of Social Informatics, Kyoto University, Kyoto, Japan*
E-mail: qiang@i.kyoto-u.ac.jp

Abstract. Knowledge graph embeddings, representing entities and relations using vectors or matrices, widely participate in solving various problems together with deep learning, such as natural language understanding and named entity recognition. The quality of knowledge graph embeddings highly affects the performance of the models on many knowledge-involved tasks. Link prediction (LP) and triple classification (TC) are widely adopted to evaluate the performance of knowledge graph embeddings. Link prediction is to predict the missing entity that completes a triple, which represents a fact in knowledge graphs, while triple classification is to determine whether the unknown triple is true or not. Both link prediction and triple classification can intuitively reflect the performance of the knowledge graph embedding model; however, it treats every triple equally, which is not capable of evaluating the performance of the embedding models on knowledge graphs that offer the *weight* information on the triples. As a consequence, this paper originally introduces two weight-aware extended tasks for LP and TC, called *weight-aware link prediction* (WaLP) and *weight-aware triple classification* (WaTC), respectively, aiming to better evaluate the performance of the embedding models on weighed knowledge graphs. WaLP and WaTC emphasize the ability of the embeddings to predict and classify triples with high weights, respectively. Lastly, we respond to the newly introduced tasks by proposing a general method *WaExt* to extend existing knowledge graph embedding models to weight-aware extensions. We test WaExt on four knowledge graph embedding models, achieving competitive performance than the baselines. The code is available at: https://github.com/Diison/WaExt.

Keywords: weight-aware evaluation tasks, knowledge graph embeddings

1. Introduction

Knowledge graphs (KGs) store real-world knowledge in the form of graphs, facilitating the advancement of artificial intelligence. Facts encoded in knowledge graphs are mostly formalized as a set of triples (h, r, t), in which h represents the head entity, t represents the tail entity, and r represents the relation between h and t. Several large-scale knowledge graphs, such as DBpedia [1], YAGO [2], Wikidata [3], NELL [4], and KnowledgeVault [5] have been published. The vigorous development of knowledge graphs has led to their extensive application in various real-world scenarios, from information retrieval [6], question answering [7, 8], to recommender systems [9, 10], and domain-specific tasks [11, 12]. Figure 1a) is an example of the knowledge graph extracted from the Wikipedia page "Family of Donald Trump".

*Corresponding author. E-mail: racharak@jaist.ac.jp.

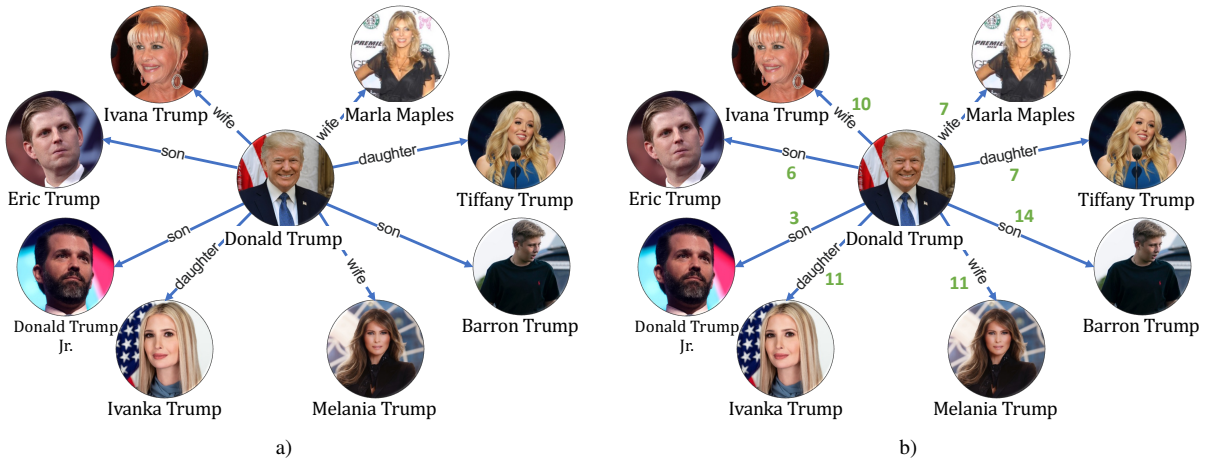


Fig. 1. An example of a knowledge graph 1a) and a weighted knowledge graph 1b). Both of them are extracted from the Wikipedia page of "Family of Donald Trump"[1]. The weight of the triple in 1b) represents the co-occurrence of the head entity and the tail entity in the Wikipedia page of "Family of Donald Trump".

Knowledge graph embedding (KGE) maps entities and relationships in a knowledge graph into continuous vector spaces to enable efficient and meaningful representation learning of structured information. Two essential tasks, link prediction (LP) and triple classification (TC), are widely utilized to assess the effectiveness of knowledge graph embeddings. In link prediction, the objective is to predict the missing entity in an incomplete triple such as (h, r, ?) or (?, r, t) [13]. By completing the triple with the missing entity, link prediction aims to infer previously unknown triples in the knowledge graph. On the other hand, triple classification is concerned with determining the truth value of an unseen triple (h, r, t). The task involves classifying whether the triple is valid and corresponds to a real-world fact or is false and does not hold in the real world. Both link prediction and triple classification serve as critical evaluation tasks for the quality of knowledge graph embeddings, as they assess the ability of the embeddings to capture facts and meaningful patterns within the graph.

In the real world, facts are not discrete items that can be simply categorized as true (1) or false (0). Instead, people assign different weights to various facts, with a heightened focus on those deemed more important. However, deterministic knowledge graphs treat all facts with equal significance, thus limiting the expressive capability of knowledge graph embedding models. For instance, consider two facts about Donald Trump, (Donald Trump, president, USA) and (Donald Trump, pseudonym, John Barron). The former, representing his role as the president of the USA, holds far greater importance in learning the embedding of "Donald Trump" compared to the latter, which pertains to a pseudonym of little relevance to most individuals. Although it is a fact that Donald Trump's pseudonym is John Barron, its significance is minimal in the context of knowledge graph representation learning.

Weighted knowledge graphs (WKGs) extend the concept of deterministic knowledge graphs by associating a weight with each triple. Figure 1b) provides an example of a weighted knowledge graph, where the weights between two entities represent statistics of their co-occurrence in the Wikipedia page "Family of Donald Trump". This formalism has proven valuable in representing uncertainty [14], confidence scores [4], degrees of relations [15], edge importance [16], and even out-of-band knowledge [17] across an increasing number of scenarios.

Weighted knowledge graphs are instrumental in modeling interactions between entities, exemplified by their application in representing protein interactions in STRING [17] and capturing the co-occurrence of concepts in Probase [15]. The utilization of weighted knowledge graphs extends its benefits to various downstream tasks, including natural language processing and knowledge discovery. For instance, they facilitate the inference of basic-level categorization for knowledge-driven applications [18] and enable the interpretation of keywords through WKGs for concept-based web searching [19].

[1] https://en.wikipedia.org/wiki/Family_of_Donald_Trump

Indeed, the traditional evaluation metrics, such as link prediction and triple classification, lack the capability to discriminate the weights of triples in weighted knowledge graphs, leading to suboptimal evaluation effectiveness. These evaluation protocols treat all triples that are to be predicted equally, regardless of their weights, which can be disadvantageous in many real-world scenarios where the importance of triples varies significantly based on their weights. For instance, studies like [20] have demonstrated that protein-protein interaction networks exhibit degree-weighted behavior. In such networks, the probability of interaction between two proteins is generally proportional to the product of their numbers of interacting partners or degrees. In such cases, it becomes crucial to distinguish between triples based on their weights to accurately assess the performance of knowledge graph embeddings.

1.0.1. Twofold Contributions

Link prediction and triple classification, being weight-agnostic, fall short in capturing the true performance of KG embeddings in scenarios like the aforementioned one, as they do not adequately consider the significance of different triples with varying weights. Hence, embeddings obtained from weighted knowledge graphs necessitate the introduction of weight-aware evaluation tasks to accurately assess their performance on triples with different weights. In this paper, we address this need by introducing **W**eight-**a**ware **L**ink **P**rediction (WaLP) and **W**eight-**a**ware **T**riple **C**lassification (WaTC) as novel evaluation tasks. According to their definitions (cf. Section 3), if two knowledge graph embedding models correctly predict the same number of triples, the one that can predict more triples with high weights is regarded as a superior one.

To fully leverage the potential of knowledge graph embedding models on weighted knowledge graphs, we propose a universal **W**eight-**a**ware **Ext**ension (WaExt) framework that extends existing knowledge graph embedding models by incorporating weight information into their scoring function. According to the experimental results, our proposed framework effectively emphasizes triples with higher weights, leading to improved performance on both weight-agnostic and weight-aware evaluation tasks. The introduction of weight-aware evaluation tasks provides a more comprehensive assessment of knowledge graph embeddings in real-world scenarios, where the importance of triples varies significantly. As a result, our approach offers a more meaningful understanding of the underlying patterns and relationships within the knowledge graph.

2. Related Works

2.1. Weight-agnostic Knowledge Graph Embedding Models

Weight-agnostic knowledge graph embedding models [21] are specifically tailored for knowledge graphs that do not include weights, with a primary focus on encoding facts within the knowledge graph. These models are designed to handle deterministic knowledge graphs, which can be further categorized into translational distance models, semantic matching models, and graph neural network-based models, based on their distinct approaches to modeling the interaction between entities and relations.

2.1.1. Translational Distance Models

Translational distance models, exemplified by TransE [22] and TransH [23], employ distance-based scoring functions. In these models, entities are represented as vectors in a representation space, and relations are represented as vector operations that are applied to entity vectors. Distance-based scoring functions play a critical role in these models for evaluating triples' plausibility in the knowledge graph. The plausibility of a triple is measured based on the distance between the tail entity and the translated head entity, which is operated on by the specific relation.

2.1.2. Semantic Matching Models

Semantic matching models, including DistMult [24] and ComplEx [25], represent entities and relations as vectors and interactions of vectors, respectively. These models adopt similarity-based scoring functions, which assess the plausibility of facts by measuring the similarity between the head entity and the tail entity under a specific interaction. By leveraging similarity-based scoring functions, semantic matching models capture the semantic relationships between entities and relations in the knowledge graph, providing a robust approach for evaluating the validity of triples.

2.1.3. Graph Neural Network-based Models

Knowledge graphs are organized by heterogeneous graphs, which is a type of graph where different types of nodes and edges coexist. In contrast to homogeneous graphs, which consist of only one type of node and one type of edge, heterogeneous graphs include multiple types of nodes and edges, each representing different entities and relationships. As graph-structured data, it is a natural fit for the application of graph neural networks [26] to learn embeddings from knowledge graphs. Graph neural networks are built on the principle that each node in a graph is characterized by its features and its relationships with other nodes. Originally designed for node-focused applications with simple undirected graphs, graph neural networks have been adapted to handle knowledge graphs, which involve multiple directed relations [27, 28].

Several works [29–31] have made significant contributions by extending graph neural networks to learn embeddings from knowledge graphs. These extensions are designed to handle the complexities of knowledge graphs, which typically involve multiple types of nodes and edges representing various entities and relations. By incorporating mechanisms to account for the plural node types and plural node types in knowledge graphs, these extensions enhance the capability of graph neural networks to effectively capture the intricate relationships and patterns present in the knowledge graph.

2.2. Weight-aware Knowledge Graph Embedding Model

FocusE [32] introduces an add-on layer for non-weight-aware knowledge graph embedding models to enable them to focus on high-weight triples. Regardless of the semantics of weights used in the literature, FocusE only considers the weight value associated with each link, under the assumption that weights intensify or mitigate the probability of the existence of a link. FocusE is adapted between the scoring and loss layers to modulate the output of the scoring layer based on the weights of the triples, to obtain weighted losses so that FocusE can learn embeddings from training triples with high weights. For a given positive weighted triple $l^+ := \langle (h,r,t), w \rangle$, its corresponding negative triple is l^-, the score of a weighted triple given by FocusE layer is

$$h(l) = \alpha \cdot ln\left(1 + e^{f(l)}\right)$$

where $f(l)$ is the scoring function of the base model and the modulating factor α is

$$\alpha = \begin{cases} \beta + (1-w)(1-\beta), & \text{if } l^+ \\ \beta + w(1-\beta), & \text{if } l^- \end{cases}$$

The hyper-parameter $\beta \in [0,1]$. The loss function is

$$L = -\sum_{t^+,t^-} \log \frac{e^{h(t^+)}}{e^{h(t^+)} + e^{h(t^-)}}$$

Nayyeri et al. [33] delves into the topic of introducing noise on weights during the collection of weighted triples and its impact on the accuracy of weight fitting. It emphasizes the requirement for the model's predicted weights to fall within a neighborhood of the real weights, as governed by the inequality equation:

$$w_{h,r,t} - \eta^{-^2}_{h,r,t} \leqslant f(h,r,t) \leqslant w_{h,r,t} + \eta^{+^2}_{h,r,t}$$

Furthermore, the paper explores the concept of weighted rule loss as a means to introduce logic rules into the model's training. This is achieved by minimizing the error between the predicted weights of the triple in the rule head and the product of the weights of the triples in the rule body. The findings and insights from this research shed light on the challenges and potential solutions in handling noise during weight collection and incorporating logic rules into the learning process.

Seo and Lee [34] focuses on the intriguing domain of whole-graph embedding, where the objective is to represent an entire graph as a single vector. Specifically, for weighted graphs, the paper proposes a methodology involving traversing the given graph to extract node-weight sequences. These sequences capture paths within the graph, comprising nodes and the weights associated with the edges between them. To encode these node-weight sequences into fixed-length vectors, the paper adopts the employment of an LSTM autoencoder. The results and implications of this research contribute to a deeper understanding of whole-graph embedding techniques, highlighting the significance of capturing weighted paths in graph representation.

ProbWalk [35] considers the weights of edges in a graph as transition probabilities. The paper introduces a novel method that outlines a strategy for sampling surrounding vertices based on their weights and generating random walks for graph embedding, guided by the transition probability. This innovative approach provides valuable insights into the utilization of transition probabilities as a means to capture the underlying structure of the graph in the embedding process. The outcomes and implications of this research contribute to the advancement of graph embedding techniques, particularly in understanding the interplay between edge weights and transition probabilities.

Mai et al. [36] revolves around the construction of a triple inference graph, aimed at improving the performance of deterministic knowledge graph embedding models in link prediction tasks. The paper proposes a methodology that computes triple-specific weights to enhance the modeling accuracy. Additionally, the paper introduces a framework called RW (Rule-supported Weights), which leverages the weight of a triple to rescale the distance between positive and corresponding negative triples. It is worth noting that RW can be considered as a special case of our proposed model where the weight of the negative sample is set to the weight of the corresponding positive sample. The outcomes of this research offer valuable insights into the development of efficient techniques for knowledge graph embedding and link prediction, showcasing the benefits of triple-specific weighting and the RW framework.

2.3. Evaluation Task for Knowledge Graph Embeddings

2.3.1. Link Prediction

Link Prediction (LP) is a crucial evaluation task in knowledge graph embedding, aimed at predicting the existence of a relation between two entities or determining the missing entity when given an entity and a relation within graph structural data. Link prediction finds versatile applications across various domains, such as predicting friend relations among users in a social network [37], forecasting co-author relations in citation networks [38], and anticipating interactions between genes and proteins in biological networks [39].

During the evaluation, each testing triple's head entity is removed and replaced with each entity from the dictionary in turn. The model computes scores for these corrupted triples, and the scores are then sorted in ascending order to determine the ranking of the correct entity. This process is repeated, but this time the tail entity is replaced instead of the head.

Mean Rank (MR)[40], Mean Reciprocal Rank (MRR)[41], and Hits@N [22] are widely used evaluation metrics by link prediction in the context of knowledge graph embedding. Mean Rank calculates the average of these predicted ranks, providing an overall measure of how well the model ranks the correct entities among all possible choices. Mean Reciprocal Rank calculates the average of the reciprocal of the ranks, emphasizing the importance of correctly ranking the top choices. Compared to mean rank, mean reciprocal rank is indeed more robust to knowledge graphs that contain outlier triples with extremely low plausibility. Hits@N calculates the proportion of correct entities that appear in the top N ranks, reflecting the model's ability to make accurate predictions among the most likely candidates. It complements MR and MRR by providing a more focused assessment of the model's performance on top-ranked predictions.

2.3.2. Triple Classification

The triple classification (TC) [42] is a binary classification task used to determine whether the given triple $l := (h, r, t)$ is true or not. The test set used for evaluation consists of triples from the knowledge graph, along with some number of randomly sampled negative triples. The test set is further divided into two groups based on the triples' scores: strong and weak/false. A testing triple l is labeled as strong if it exists in the knowledge graph and its confidence score is greater than the threshold τ, otherwise, it is labeled as weak/false.

Accuracy and F1 score are widely adopted metrics by triple classification. Accuracy measures the proportion of correctly classified strong and weak/false triples out of the total test set. The F1 score is a common metric used to

measure the models' performance on this task. It combines precision and recall, providing a balanced measure of how well the model can correctly identify both strong and weak/false triples.

2.3.3. Tail Entity Prediction

The task of tail entity prediction introduced by UKGE [43] is designed to evaluate the performance of knowledge graph embedding models on weighted knowledge graphs. In this task, the goal is to predict the correct tail entity for a given corrupted triple in the form of $(h, r, ?)$. To perform tail entity prediction, the model builds and scores all possible triples, and then ranks all the possible triples according to their scores. The evaluation metric used in this task is the normalized Discounted Cumulative Gain (nDCG) [44]. nDCG is commonly used to assess the performance of ranking models, taking the relevance of the ranked items into account. In the context of learning embeddings from weighted knowledge graphs, nDCG measures how well the model ranks the possible tail entities according to weights of possible triples.

However, one challenge in adopting nDCG is that it necessitates having weights for all possible triples, including negative triples, while weights of negative triples are typically not provided in the dataset. To overcome this limitation, UKGE employs probabilistic soft logic (PSL) to generate weights for negative triples. Nevertheless, inferring the weights of triplets from the model introduces uncertainty to the evaluation process.

3. Methodology

The aforementioned evaluation tasks were originally designed for deterministic knowledge graphs, where weights were not considered during the evaluation process. However, to address this limitation and account for the importance of weights to triples, weight-aware link prediction and weight-aware triple classification have been introduced.

Given a weighted knowledge graph $\mathcal{G}$ such that

$$\mathcal{G} \coloneqq \left\{ \left\langle \left(h_i, r_i, t_i\right), w_i \right\rangle \right\}_{i=1}^{u}$$

, where $h_i, t_i \in \mathcal{E}$, $r_i \in \mathcal{R}$, and $w_i \in \mathbb{R}_{\geqslant 0}$. $\mathcal{E}$ and $\mathcal{R}$ represent entity and relation sets, respectively. We will now describe the weight-aware link prediction and weight-aware triple classification tasks as follows:

3.1. Weight-aware Link Prediction

Weight-aware link prediction extends link prediction and takes the weights assigned to positive triples into consideration. The objective is to predict the existence of relations between entities, giving higher emphasis to triples with higher weights.

3.1.1. Task Description

Given a corrupted triple in the form of $(h, r, ?)$ (or $(?, r, t)$), the objective is to identify a proper entity $\varphi \in \mathcal{E}$ to restore the corrupted triple to a complete form, i.e., (h, r, φ) (or (φ, r, t), respectively) with awareness of the weight of $(h, r, \varphi) \in \mathcal{G}$ (or $(\varphi, r, t) \in \mathcal{G}$, respectively).

3.1.2. Evaluation Protocol and Metrics

For each testing weighted triple $\langle (h_i, r_i, t_i), w_i \rangle$ in the testing set, the weight w_i is omitted. Subsequently, the head entity h_i is removed and replaced by each of the entities in the dictionary in turn, generating a set of possible triples $\{\langle (\varphi, r_i, t_i), ? \rangle\}$ where $\langle (\varphi, r_i, t_i), * \rangle \notin \mathcal{G}$. Next, the models compute scores for the testing weighted triple and the possible triples, which are then sorted in ascending order. Following the sorting process, the ranking of the testing weighted triple r_i is recorded. In order to provide additional rewards to models that perform better on triples with higher weights, the activation function g is applied to adjust the ranking based on the weight of the testing triple, resulting in the weighted ranking of the testing triple. For the sake of uniformity, we assume that the activation function is monotonically increasing, so the weighted ranking of the testing triple should be

$$r_i^w = r_i \cdot \frac{1}{g(w_i)}$$

This entire procedure is then repeated, but this time, t_i is removed instead of h_i.

We introduce weight-aware mean rank (WaMR), weight-aware mean reciprocal rank (WaMRR), and weight-aware Hits@N (WaHits@N) as performance metrics for weight-aware link prediction. These metrics are represented by Equations 1, 2, and 3, respectively. In these equations, $c = \frac{\sum_{j=1}^{u} g(w_j)}{u}$ represents the normalization factor. WaMR calculates the average of the weighted rankings of all testing triples. WaMRR computes the average of the reciprocals of the weighted rankings. WaHits@N measures the weighted count of the top N triples, which holds significance for recommender systems. The introduced these three weight-aware metrics places emphasis on the prediction of high-weight triples more than their weight-agnostic counterparts.

$$\text{WaMR} = \frac{1}{u}\sum_{i=1}^{u} c \cdot r_i^w, \tag{1}$$

$$\text{WaMRR} = \frac{1}{u}\sum_{i=1}^{u} \frac{1}{c \cdot r_i^w}, \tag{2}$$

$$\text{WaHits@N} = \frac{1}{u}\sum_{i=1}^{u} \mathbb{I}[c \cdot r_i^w \leqslant N] \tag{3}$$

The function $g : \mathbb{R}_{\geqslant 0} \to \mathbb{R}$ is introduced as an activation function with the constraint $g(w) \neq 0$. This activation function is utilized to re-scale the weights, allowing for increased attention to more important triples. It is important to note that g is defined as a general function with the non-zero constraint to limit its possible candidates. This condition is chosen to prevent unattended triples from being overlooked. In Subsection 4.3, we provide illustrations of suitable candidates for the function g. Finally, the $\mathbb{I}[\text{expn}]$ denotes the indicator function, which outputs 1 if the expression expn is true, and 0 otherwise.

3.2. Weight-aware Triple Classification Task

Weight-aware triple classification extends triple classification, in which the gain of the model to be assessed is not solely determined by the classification result but also by the weights associated with the true triples under testing.

3.2.1. Task Description

The task involves a set of triples comprising positive triples $(h_i, r_i, t_i) \in \mathcal{G}$ and negative triples $(h_j, r_j, t_j) \notin \mathcal{G}$. The goal is to classify positive triples from negative triples while taking into account the awareness of the weights of the triples in $\mathcal{G}$. The gain for classifying a triple is calculated based on the result of classifying and the weight of the triple.

3.2.2. Evaluation Protocol and Metrics

For each testing weighted triple $\langle (h_i, r_i, t_i), w_i \rangle$, the weight w_i is omitted. The head entity h_i is then removed to create the head-corrupted testing triple (φ, r_i, t_i). Subsequently, the head entity of the head-corrupted testing triple is replaced by k entities from the dictionary, resulting in a set of head-corrupted testing triples. To create the mixture set for the head-corrupted testing triple, the testing triples are mixed with their head-corrupted testing triples:

$$mix_h := \{\langle (h_j, r_j, t_j), ? \rangle \mid \langle (h_j, r_j, t_j), w_j \rangle \in \mathcal{G}\} \cup \{\langle (\varphi, r_j, t_j), ? \rangle \mid (\varphi, r_j, t_j) \notin \mathcal{G}\} \tag{4}$$

Similarly, the tail entity t_i is removed to create the tail-corrupted testing triple (h_i, r_i, φ). The tail entity of the tail-corrupted testing triple is then replaced by k entities from the dictionary, generating a set of possible triples for

the tail-corrupted testing triple. To construct the mixture set for the tail-corrupted testing triple, the testing triples are mixed with their tail-corruption testing triples:

$$mix_t \coloneqq \left\{ \left\langle \left(h_j, r_j, t_j \right), ? \right\rangle \mid \left\langle \left(h_j, r_j, t_j \right), w_j \right\rangle \in \mathcal{G} \right\} \cup \left\{ \left\langle \left(h_i, r_j, \varphi \right), ? \right\rangle \mid \left(h_i, r_j, \varphi \right) \notin \mathcal{G} \right\} \tag{5}$$

The model is tasked with scoring the triples and dividing mix_h and mix_t into a positive set and a negative set, respectively, based on the score of the triples and a threshold τ. After classifying the triples, there are three types of triples that are of interest. The first type consists of true triples classified as positive, forming a true positive set (S_{tp}). The second type comprises false triples classified as positive, forming a false positive set (S_{fp}). The third type includes true triples classified as negative, forming a false negative set (S_{fn}). We introduce weight-aware F1 score (WaF1) as performance metrics for weight-aware triple classification, which is represented by Equation 6. WaF1 is the harmonic mean of the weight-aware precision *wa_prec* and weight-aware recall *wa_recall*. The proposed WaF1 metric provides greater rewards than F1 to models that correctly classify a higher number of triples with higher weights.

$$\text{WaF1} = 2 * \frac{\text{wa\_prec} * \text{wa\_recall}}{\text{wa\_prec} + \text{wa\_recall}} \tag{6}$$

$$\text{where} \quad \text{wa\_prec} = \frac{\sum\limits_{\langle (h_i,r_i,t_i),? \rangle \in TP} \frac{1}{c} \cdot g(w_i)}{\sum\limits_{\langle (h_i,r_i,t_i),? \rangle \in TP} \frac{1}{c} \cdot g(w_i) + \sum\limits_{\langle (h_i,r_i,t_i),? \rangle \in FP} \frac{1}{c} \cdot g(w_i)}$$

$$\text{wa\_recall} = \frac{\sum\limits_{\langle (h_i,r_i,t_i),? \rangle \in TP} \frac{1}{c} \cdot g(w_i)}{\sum\limits_{\langle (h_i,r_i,t_i),? \rangle \in TP} \frac{1}{c} \cdot g(w_i) + \sum\limits_{\langle (h_i,r_i,t_i),? \rangle \in FN} \frac{1}{c} \cdot g(w_i)}$$

3.3. Weight-aware Extension Framework

Since existing deterministic knowledge graph embedding models excel at learning interactions of entities and relations within triples, we propose a general framework called WaExt for injecting weights into these existing models. This involves combining the weights and their scoring function $f(h,r,t)$, as illustrated in Figure 2. In Figure 2, the operator $\oplus$ represents a summation operation that aggregates all the losses (in Figure 2a) and the weighted losses (in Figure 2b). The scoring function $f(h,r,t)$ is employed by the knowledge graph embedding model during the training phase to calculate the score of each triple from the training set S. After being extended by the proposed framework, the weight-aware scoring function $f_w(h,r,t,w)$ can be represented as Equation 7, calculating the score of a triple based on both its plausibility and its weight.

$$f_w(h,r,t,w) \coloneqq g(w) \cdot f(h,r,t) \tag{7}$$

We adopt the margin ranking loss [22] as the loss function:

$$\mathcal{L} = \sum_{\langle (h,r,t),w \rangle \in S} \sum_{\langle (h',r',t'),w' \rangle \in S'} \left[\gamma + f_w(h,r,t,w) - f_w\left(h',r,t',w'\right) \right]_+ \tag{8}$$

where $[x]_+$ denotes the positive part of x, $\gamma > 0$ is a margin hyperparameter, w' is the weight of the negative triples (regarded as a hyper-parameter of the model), and S' is the set of the negative triples defined as follows:

$$S' \coloneqq \left\{ \left\langle \left(h', r, t \right), w' \right\rangle \mid h'_i \in E \setminus \{h_i\} \right\}_{i=1}^{u} \cup \left\{ \left\langle \left(h, r, t' \right), w' \right\rangle \mid t'_i \in E \setminus \{t_i\} \right\}_{i=1}^{u} \tag{9}$$

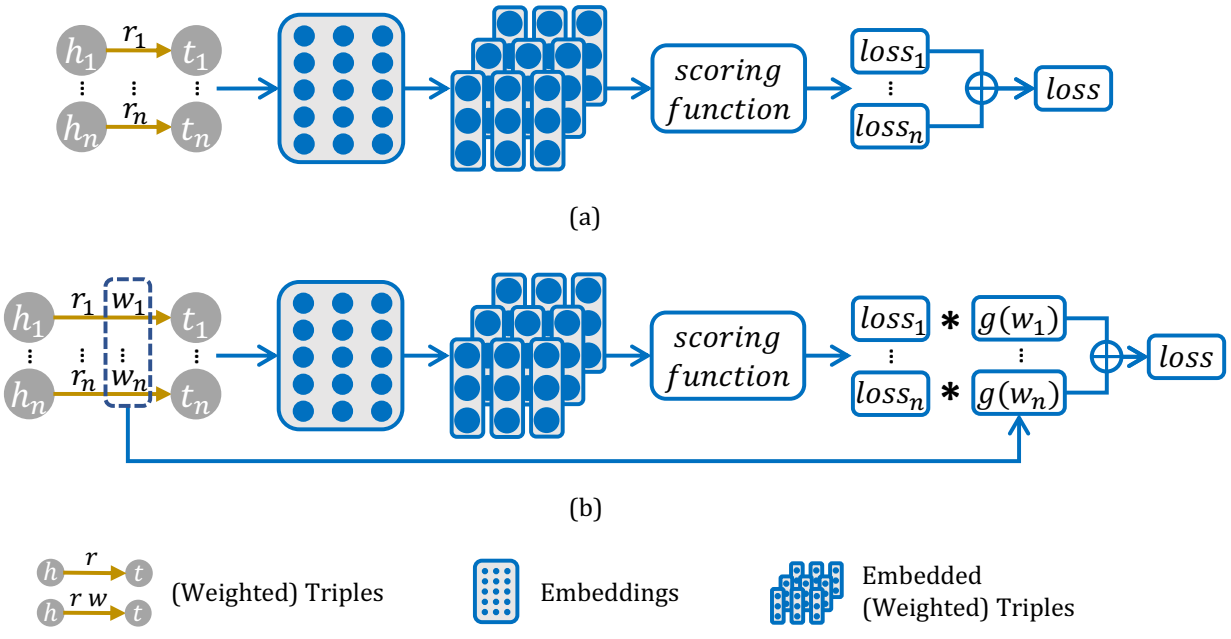


Fig. 2. An illustration of WaExt. (a) is the process of base model, while (b) is its weight-aware extension.

Note that the model is only aware of the weights of the triples during the training process. For the testing phase, the weights of the triples are omitted, and the weight-aware extensions score the triples using the same scoring functions as their base models.

4. Experiments and Results

4.1. Experiment Setting

We conducted experiments on three datasets: CN15K, NL27K, and PPI5K [45]. CN15K is a subgraph of ConceptNet [14] and comprises 15,000 entities 36 relations, and 229,235 weighted triples in English. The original scores in ConceptNet vary from 0.1 to 22, while the weights in CN15K are normalized to the range [0.1, 1.0]. NL27K is extracted from NELL [4], a weighted knowledge graph obtained from webpage reading. NL27K contains 27,221 entities, 405 relations, and 175,412 weighted triples. The weights in NL27K are normalized to the interval [0.1, 1.0]. Notably, 8 and 4 unseen relations in the training set appear in the testing and validation sets of NL27K, respectively. These out-of-distribution relations could harm the models' performance on NL27K. We also investigate this observation in our experiments. PPI5K is a subset of the protein-protein interaction knowledge graph STRING [17], consisting of 255,114 weighted triples for 4,999 proteins and 7 relations. STRING labels the interactions between proteins with the probabilities of occurrence. The weights in PPI5K fall within the interval [0.15, 1.0]. For more detailed datasets' statistics, refer to Table 1.

The correlation between the weight and the triple degree in the three datasets is illustrated in Figure 3. The weight distribution of triples in the three datasets is illustrated in Figure 4. In these three datasets, the weight of triples in CN5K shows the highest concentration around 0.7 and 0.8, while the distribution in other regions is relatively uniform. Additionally, the degree of triplets in CN5K is generally lower than 1000. Moving on to NL27K, the weights of triples are concentrated in several intervals, with a significant concentration in the high-weight interval [0.8, 1] and around 0.4. The degree of triples in NL27K is concentrated in two regions, near 6000 and lower than 2000. Regarding PPI5K, the weights of triples are most dispersed, mainly concentrated in the low-weight range. Similarly, the degree of the triplets in PPI5K is generally lower than 2000.

Table 1

Datasets Statistics. #Ent denotes the number of entities. #Rel denotes the number of the relations. #Tri denotes the number of the triples. INR denotes the interval of the weights, i.e, the biggest weight minus the smallest weight. Avg(deg) denotes the average of the degree of the entities and Med(deg) denotes the median of the degree of the entities.

		#Ent	#Rel	#Tri	INR	Avg(deg)	Med(deg)
CN15K	train	15000	36	193274	0.900	25.77	12
	test	10659	34	19166	0.900	3.60	2
	val	10158	35	16795	0.900	3.31	2
NL27K	train	27221	405	149100	0.899	10.95	4
	test	9711	287	14034	0.898	2.89	1
	val	9000	279	12278	0.899	2.73	1
PPI5K	train	4999	7	214661	0.847	85.88	21
	test	3703	7	21566	0.847	11.65	4
	val	3557	7	18887	0.847	10.62	3

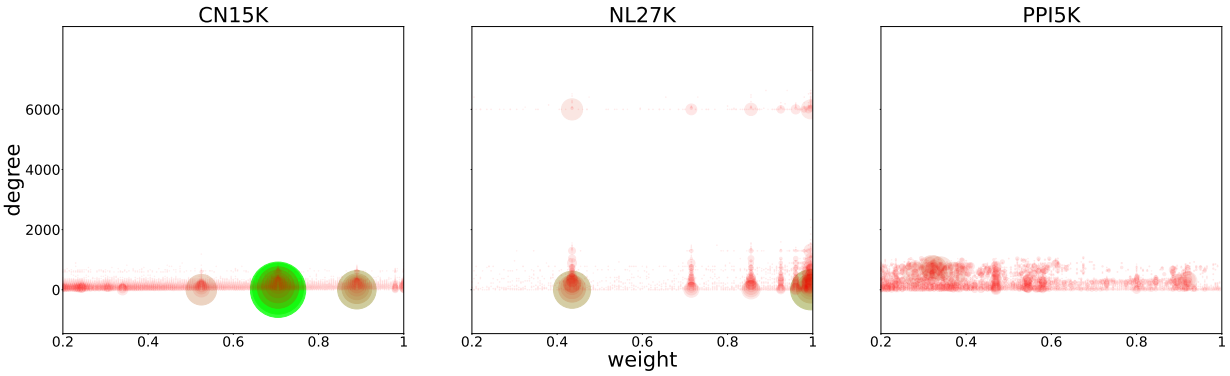
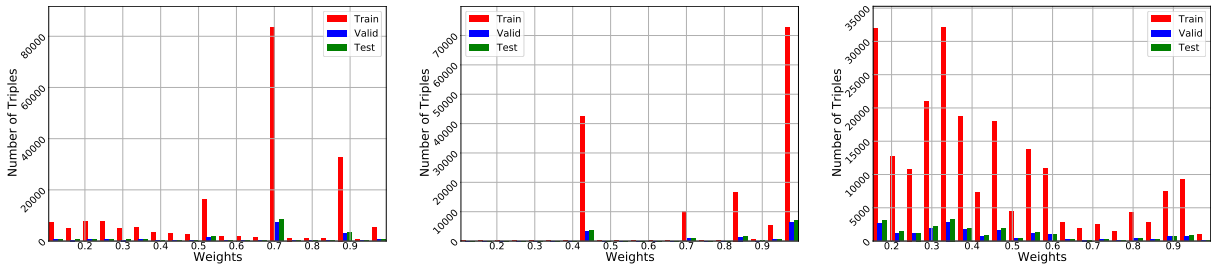


Fig. 3. The Correlation of Weight and Triple Degree in CN15K, NL27K, and PPI5K. The degree of a triple is calculated as the average of the degree of the head entity and the degree of the tail entity. The weights of the triples range from 0 to 1, while the degrees fall within the range of 0 to 7300. For analysis purposes, both the weight and degree intervals are divided into 200 subintervals. The number of triples falling within each interval is counted and recorded as $num(tri)$. In the visualization, each circle represents an interval. The center of the circle indicates the center of weight and the center of degree for that interval. The color of the circle is defined using RGB=(1-$num(tri)$/25541, $num(tri)$/25541, 0), where $num(tri)$/25541 is the normalized number of triples in that interval. The opacity and radius of each circle are determined by $num(tri)$, where higher opacity and a bigger radius signify a higher number of triples within that interval. A greener or denser color indicates a larger number of triples in the subinterval represented by the center of the circle.



a) CN15K b) NL27K c) PPI5K

Fig. 4. Weight Distribution of Triples in CN15K, NL27K, and PPI5K.

We implemented weight-aware link prediction, and weight-aware triple classification, and applied the weight-aware extension framework to the knowledge graph embedding models with the help of the PyKEEN toolkit [46]. As for the activation function, we considered the linear function with coefficient=1 and the exponential function

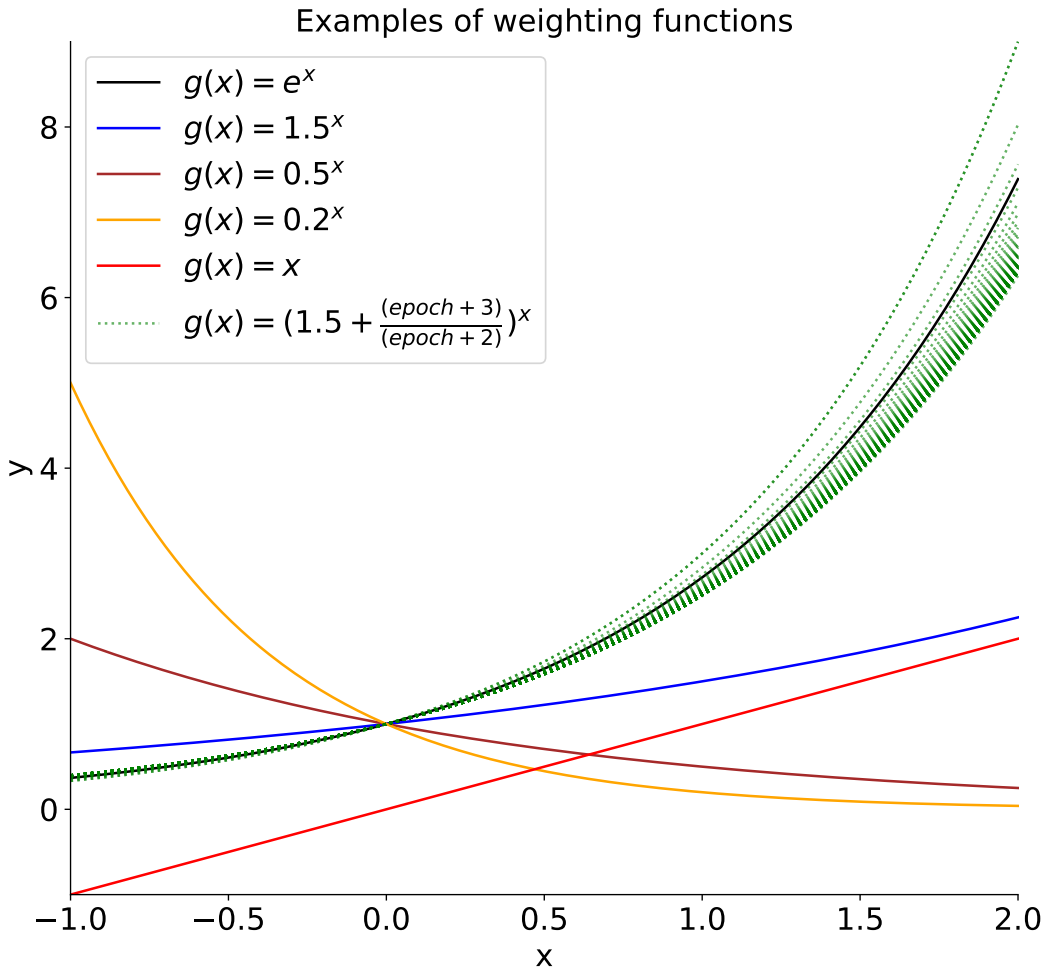


Fig. 5. The activation functions with static base and dynamic base.

with bases=[0.001, 0.01, 0.1, 0.2, ..., 0.9, 1.1, ..., 1.5, e, 3.0, 4.0]. Additionally, we explored a dynamic exponent base, adjusting the base of the exponential function in every epoch to prevent overfitting on high-weight triples. For this purpose, we choose

$$g(x) = \left(base + \frac{(epoch+3)}{(epoch+2)}\right)^x$$

as the dynamic weighting function. An illustration of the activation functions can be found in Figure 5.

The embedding dimension was set to 50. We adopted the Adam optimizer with the initial learning rate $\lambda = 0.001$, and the weight of the negative triples w' from $\{0, 0.5, 1, avg(w), 1 - avg(w)\}$, where $avg(w)$ represents the mean of the weights of triples in the training set. The margin of the loss function γ was set to 1. The number of false triples for every testing triple in triple classification and weight-aware triple classification was set to 100. We trained the models for 1000 epochs, evaluated them every 10 epochs, and saved their best results. As MRR is considered a relatively more robust and comprehensive metric, we use it as the criterion for recording the best results.

4.2. Base Knowledge Graph Embedding Models

We have implemented the proposed framework based on two representative translational distance models: TransE and TransH, as well as two representative semantic matching models: ComplEx and DistMult.

TransE [22] is one of the most representative translational distance models. It interprets entities as vectors and the relation as a translation vector of the head entity in one embedding space. The scoring function of TransE is expressed as

$$s = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_p$$

TransE is effective for 1-to-1 relations but falls short in modeling 1-to-N or N-to-N relations. In contrast, TransH [23] addresses these limitations by introducing a mechanism that projects entities onto relation-specific hyperplanes. The scoring function of TransH is defined as follows:

$$s = -\left\|\left(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r\right) + \mathbf{r} - \left(\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r\right)\right\|_2^2$$

Here, $\mathbf{w}_r$ represents the normal vector of a specific hyperplane associated with the relation r.

DistMult [24] represents each relation as a diagonal matrix, allowing it to model pairwise interactions between entities and capture latent semantics effectively. The scoring function of DistMult is given by:

$$s = \mathbf{h}^\top \operatorname{diag}(\mathbf{r})\mathbf{t}$$

where $\operatorname{diag}(\cdot)$ denotes the diagonalization function.

ComplEx [25] extends DistMult by introducing a complex vector space to embed the knowledge graph, with the aim of better modeling asymmetric relations. The scoring function of ComplEx is asymmetric, providing different scores for facts involving asymmetric relations, depending on the order of entities involved. The score function is defined as follows:

$$s = \operatorname{Re}\left(\mathbf{h}^\top \operatorname{diag}(\mathbf{r})\bar{\mathbf{t}}\right)$$

where $\bar{\mathbf{t}}$ represents the conjugate of $\mathbf{t}$, and $\operatorname{Re}(\cdot)$ takes the real part of a complex value.

4.3. Results on Link Prediction and Weight-aware Link Prediction

The results of the base models and their weight-aware extension versions on both the link prediction and weight-aware link prediction tasks are presented in Table 2 and Table 3, respectively. In terms of link prediction, our proposed weight-aware extension models outperform TransE, TransH, ComplEx, and DistMult on metrics such as mean rank, mean reciprocal rank, and most of Hits@N. We attribute this improvement to the fact that triples with higher weights tend to contain more valuable information and less noise, and the weight-aware extension models can effectively emphasize these high-weight triples.

Regarding link prediction in greater detail, WeExt employing an exponential function with a dynamic base as the activation equation, attains the optimal performance in 8 out of the total 12 model-dataset combinations. On the other hand, WeExt using an exponential function with a static base as the activation equation, achieves the best performance on the remaining 4 combinations. Additionally, WaExt adopting a linear function with coefficient=1 as the activation equation has also resulted in significant performance improvements for 7 model-dataset combinations. We believe that exploring optimal coefficients for the linear function could lead to even better performance. As part of our future work, we plan to conduct more experiments to search for the optimal coefficient.

We choose FocusE as the baseline, and its results are presented in Table 4. As our implementation and FocusE are based on different knowledge graph embedding libraries, we can not directly compare the performance of WaExt and FocusE. Instead, we focus on comparing the improvements achieved compared to the base models. While FocusE

Table 2

Results of base models (TransE, TransH, ComplEx, DistMult) and weight-aware extension models (WaTransE, WaTransH, WaComplEx, WaDistMult) on the link prediction task. "$\exp_{sta.}$" indicates the extended models adopt an exponential function with a static base, while "$\exp_{dyn.}$" refers to the extended models adopt an exponential function with a dynamic base.

Dataset	Model		MR	MRR	Hit@1	Hit@3	Hit@5	Hit@10
CN15K	TransE		1184.3	0.1042	0.0367	0.1249	0.1721	0.2369
	WaTransE	linear	1232.3	0.1094	**0.0377**	0.1349	0.1839	0.2498
		$\exp_{sta.}$	**947.4**	0.1135	0.0372	0.1404	0.1931	**0.2636**
		$\exp_{dyn.}$	1044.1	**0.114**	0.0376	**0.1435**	**0.1943**	0.2621
	TransH		1768.7	0.0773	**0.0419**	0.0856	0.1069	0.1402
	WaTransH	linear	2136.7	0.0869	0.0419	0.0998	0.1296	0.1726
		$\exp_{sta.}$	**1031.2**	0.1023	0.0403	0.1247	0.1615	0.2162
		$\exp_{dyn.}$	1533.6	**0.1025**	0.039	**0.1265**	**0.1657**	**0.2239**
	ComplEx		1872.5	0.1232	0.065	0.1393	0.1803	0.2409
	WaComplEx	linear	1866	0.1195	0.0604	0.1369	0.177	0.2362
		$\exp_{sta.}$	**1743.8**	**0.1377**	**0.0756**	**0.1606**	**0.2017**	**0.2583**
		$\exp_{dyn.}$	1864.1	0.1236	0.0655	0.1391	0.1809	0.2408
	DistMult		987.4	0.105	0.0397	0.1267	0.1678	0.2264
	WaDistMult	linear	**954.7**	0.1112	0.0394	0.1348	0.1828	0.2469
		$\exp_{sta.}$	1120.9	0.1127	**0.0406**	**0.1386**	0.1836	0.247
		$\exp_{dyn.}$	1100.7	**0.1129**	0.0405	0.1383	**0.1853**	**0.2491**
NL27K	TransE		130.1	0.3349	**0.2116**	0.398	0.4638	0.5575
	WaTransE	linear	213.4	0.3196	0.1886	0.3942	0.4613	0.5501
		$\exp_{sta.}$	92.1	**0.3457**	0.2038	0.423	0.5013	**0.6033**
		$\exp_{dyn.}$	**90.4**	0.3411	0.1933	**0.4255**	**0.503**	0.5969
	TransH		817.4	0.2654	0.1848	0.3019	0.3453	0.4054
	WaTransH	linear	205.2	0.3227	0.2163	0.3728	0.4323	0.5151
		$\exp_{sta.}$	242.1	**0.326**	**0.2231**	0.373	0.4321	0.5125
		$\exp_{dyn.}$	**185.9**	0.3241	0.2163	**0.3742**	**0.4356**	**0.5167**
	ComplEx		**214.7**	0.6468	0.5422	0.7116	0.7747	0.8467
	WaComplEx	linear	229.4	0.608	0.5025	0.6674	0.7332	0.813
		$\exp_{sta.}$	271.1	**0.6886**	**0.597**	**0.7447**	**0.8007**	**0.862**
		$\exp_{dyn.}$	215.3	0.6469	0.542	0.7119	0.7746	0.8462
	DistMult		**156.6**	0.4037	0.313	0.4461	0.4958	0.5652
	WaDistMult	linear	160.7	0.4459	0.3576	**0.4866**	**0.5353**	**0.6077**
		$\exp_{sta.}$	160.7	0.4459	0.3576	**0.4866**	**0.5353**	**0.6077**
		$\exp_{dyn.}$	161.1	**0.4461**	**0.3585**	**0.4866**	**0.5353**	0.6063
PPI5K	TransE		**27.6**	0.1398	0	0.1746	0.269	0.4181
	WaTransE	linear	105.3	0.1364	0	0.1823	0.2678	0.3951
		$\exp_{sta.}$	29.1	0.1515	**0.0001**	0.1999	0.2956	0.4403
		$\exp_{dyn.}$	30.2	**0.1528**	0	**0.2042**	**0.3**	**0.4419**
	TransH		49.9	0.1093	0	0.1303	0.1959	0.3155
	WaTransH	linear	1163.9	0.1216	**0.0217**	0.1612	0.2251	0.3207
		$\exp_{sta.}$	**41**	0.1395	0.0001	0.1836	0.2677	0.3967
		$\exp_{dyn.}$	52.7	**0.1417**	0	**0.1906**	**0.2739**	**0.4067**
	ComplEx		**7.8**	0.9247	0.8735	0.9743	0.9857	0.9915
	WaComplEx	linear	10.1	0.9482	0.915	**0.9809**	**0.9885**	**0.992**
		$\exp_{sta.}$	11.7	0.9502	**0.9197**	0.9804	0.988	0.9915
		$\exp_{dyn.}$	11.8	**0.9503**	**0.9197**	0.9804	0.9881	0.9913
	DistMult		27.1	0.4383	0.3203	0.4932	0.5599	0.6495
	WaDistMult	linear	**21.2**	0.4216	0.2996	0.4696	0.548	0.6552
		$\exp_{sta.}$	25.2	0.45	0.3401	0.4909	**0.566**	**0.667**
		$\exp_{dyn.}$	24	**0.4556**	**0.353**	**0.4943**	0.5618	0.6513

Table 3

Results of base models (TransE, TransH, ComplEx, DistMult) and weight-aware extension models (WaTransE, WaTransH, WaComplEx, WaDistMult) on the weight-aware link prediction task. "$\exp_{sta.}$" indicates the extended models adopt an exponential function with a static base, while "$\exp_{dyn.}$" refers to the extended models adopt an exponential function with a dynamic base.

Dataset	Model		MR	MRR	Hit@1	Hit@3	Hit@5	Hit@10
CN15K	TransE		1276.5	0.1068	0.0069	0.118	0.1702	0.2349
	WaTransE	linear	2076.5	**0.1187**	**0.0079**	0.136	0.184	0.2502
		$\exp_{sta.}$	**1038.8**	0.1175	0.0075	0.1341	0.1909	0.2594
		$\exp_{dyn.}$	1166.6	0.1187	0.0077	**0.1369**	**0.1936**	**0.2597**
	TransH		1880	0.078	0.0103	0.0829	0.1074	0.1393
	WaTransH	linear	2407.6	0.0909	0.0131	0.0962	0.1298	0.1722
		$\exp_{sta.}$	**1747.6**	0.1067	**0.0096**	0.1212	0.1659	0.2201
		$\exp_{dyn.}$	1754.9	**0.1071**	0.0092	**0.1219**	**0.1664**	**0.2226**
	ComplEx		2028.9	0.128	0.0344	0.132	0.1776	0.2375
	WaComplEx	linear	2027.1	0.1231	0.0285	0.1291	0.1735	0.2328
		$\exp_{sta.}$	**1896.4**	**0.1462**	**0.0468**	**0.1552**	**0.2009**	**0.256**
		$\exp_{dyn.}$	2020.2	0.1284	0.0346	0.1318	0.1793	0.2384
	DistMult		1066.5	0.1077	0.0107	0.1189	0.1646	0.2229
	WaDistMult	linear	**1042.5**	0.1142	0.0108	0.1262	0.1789	0.2437
		$\exp_{sta.}$	1255	0.1168	**0.0114**	**0.1321**	0.181	0.2456
		$\exp_{dyn.}$	1230.2	**0.1169**	0.0108	0.1317	**0.1832**	**0.2474**
NL27K	TransE		148	0.3473	0.1567	0.3776	0.445	0.5438
	WaTransE	linear	335.6	0.3549	**0.1616**	0.3534	0.4494	0.5433
		$\exp_{sta.}$	**98.3**	**0.358**	0.1516	0.4016	0.4796	**0.5849**
		$\exp_{dyn.}$	103.7	0.3573	0.1499	**0.4048**	**0.4815**	0.5833
	TransH		946.3	0.2787	0.138	0.2934	0.3341	0.4006
	WaTransH	linear	279.7	0.3448	0.1734	**0.362**	0.422	0.5077
		$\exp_{sta.}$	318.4	**0.3487**	**0.1792**	0.3607	0.4199	0.5052
		$\exp_{dyn.}$	**241.3**	0.3458	0.1722	0.3619	**0.4228**	**0.5095**
	ComplEx		**241.4**	0.6453	0.3504	0.6929	0.7553	**0.8372**
	WaComplEx	linear	270.3	0.593	0.2973	0.6516	0.7167	0.7967
		$\exp_{sta.}$	367.2	**0.7054**	**0.4488**	**0.7123**	**0.7612**	0.823
		$\exp_{dyn.}$	242.2	0.6453	0.3503	0.6932	0.7557	0.838
	DistMult		**174.7**	0.4111	0.2157	0.4318	0.4825	0.5546
	WaDistMult	linear	174.9	0.4428	0.223	**0.4725**	**0.5209**	**0.5968**
		$\exp_{sta.}$	174.9	0.4428	0.223	**0.4725**	**0.5209**	**0.5968**
		$\exp_{dyn.}$	175.5	**0.4437**	**0.225**	0.472	0.5214	0.5965
PPI5K	TransE		**30.9**	0.1583	0	0.1603	0.2529	0.3977
	WaTransE	linear	208.3	**0.19**	**0.0508**	0.1916	0.263	0.375
		$\exp_{sta.}$	36.8	0.1767	0	0.19	0.2799	0.413
		$\exp_{dyn.}$	35.1	0.178	**0**	**0.1923**	**0.2823**	**0.4214**
	TransH		55.9	0.1253	0	0.1232	0.1903	0.3024
	WaTransH	linear	1347.5	0.1505	**0.0198**	0.1573	0.2213	0.3131
		$\exp_{sta.}$	**48.6**	0.1649	0	0.1756	0.2586	0.3836
		$\exp_{dyn.}$	63	**0.1678**	0	**0.182**	**0.2624**	**0.3878**
	ComplEx		**7.9**	0.9205	0.2948	0.9651	0.9826	0.9909
	WaComplEx	linear	10.2	0.9414	0.3064	0.9768	0.9878	**0.9917**
		$\exp_{sta.}$	11.8	**0.9427**	**0.3066**	0.9771	0.9876	0.9914
		$\exp_{dyn.}$	11.8	0.9427	0.3065	**0.9776**	**0.9879**	0.9913
	DistMult		29.9	0.4306	0.0621	**0.4781**	0.5502	0.6465
	WaDistMult	linear	**22.3**	0.401	0.0499	0.4486	0.5393	0.6526
		$\exp_{sta.}$	27.6	0.4384	0.0617	0.4709	**0.5537**	**0.6643**
		$\exp_{dyn.}$	26.1	**0.4404**	**0.063**	0.4765	0.5517	0.6504

Table 4
Results of FocusE on the link prediction task.

	Model	MR	MRR	Hits@1	Hits@3	Hits@5	Hits@10
CN15K	TransE	6864.0	0.0027	0.0014	0.0020	0.0030	0.0047
	+FocusE	1288.9	0.0984	0.0433	0.1149	0.1353	0.2022
	DistMult	1829.8	0.0956	0.0577	0.1026	0.1265	0.1669
	+FocusE	2293.6	0.1059	0.0658	0.1156	0.1316	0.1826
NL27K	TransE	118.9	0.4357	0.2717	0.5421	0.6238	0.7214
	+FocusE	138.7	0.3397	0.2014	0.4182	0.4585	0.5817
	DistMult	117.7	0.6615	0.5540	0.7310	0.7976	0.8682
	+FocusE	2566.3	0.4092	0.3183	0.4535	0.4851	0.5859
PPI5K	TransE	20.0	0.1858	0	0.2569	0.3790	0.5570
	+FocusE	35.6	0.1526	0	0.2138	0.2685	0.4364
	DistMult	4.4	0.9239	0.8647	0.9823	0.9872	0.9916
	+FocusE	14.4	0.7745	0.6795	0.8402	0.8650	0.9407

improves the performance on CN15K, it leads to performance loss on NL27K and PPI5K. While WaExt brings a more stable performance improvement to the base model compared to FocusE.

Regarding weight-aware link prediction, both WaExt utilizing the exponential function with the static base as its activation function, and WaExt utilizing the exponential function with the dynamic base as its activation function, have demonstrated comparable performance and respectively achieved the best results among 5 model-dataset combinations. Additionally, WaExt adopting a linear function with coefficient=1 also exhibited commendable performance, with its extended TransE model achieving the optimal performance on CN15K and PPI5K.

The aforementioned results on link prediction and weight-aware link prediction do not entirely illustrate the superiority of our proposed weight-aware evaluation tasks. To address this, we exemplify a suboptimal model to highlight the advantages of the proposed tasks. Figure 6 illustrates the distribution of triples whose ranking is no more than 1 predicted by the TransE and WaTransE on NL27K. WaTransE adopts an exponential function with a static base of 1.5 as the activation function, and the weight of negative examples is set to 0.5. In the link prediction task, TransE successfully predicted a higher number of Hits@1 triplets, resulting in a higher Hits@1 score compared to other models. However, in the weight-dependent link prediction, WaTransE excelled in predicting a larger number of high-weight triplets, which advantage has been significantly amplified and led to a higher WaHits@1 score.

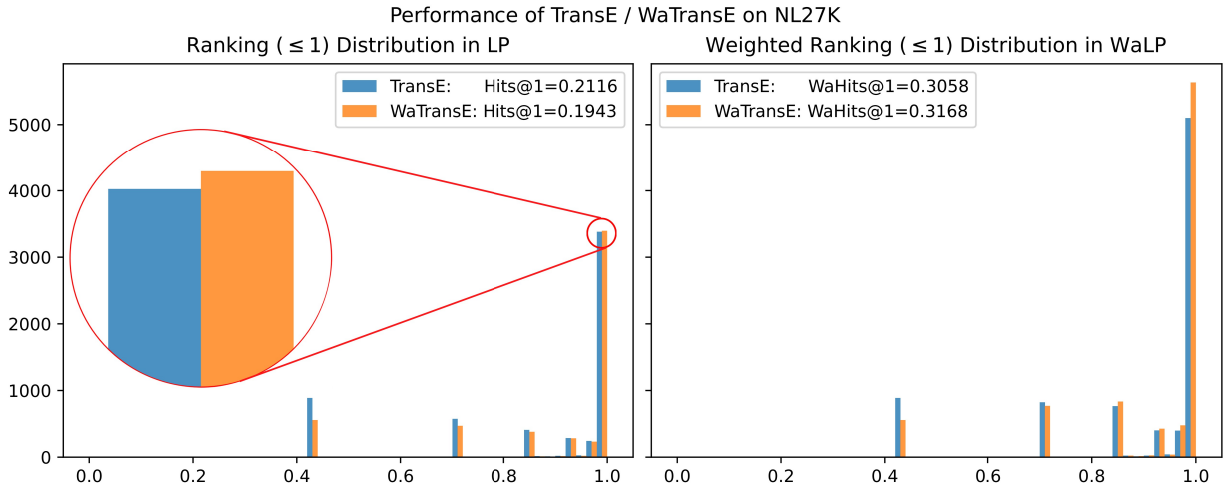


Fig. 6. Distribution of triples with the ranking/weight-aware ranking of no more than 1.

4.4. Result on Triple Classification and Weight-aware Triple Classification

The results of base models and their weight-aware extensions on the triple classification and weight-aware triple classification are presented in Table 5. From the table, it is evident that our proposed weight-aware extensions outperform TransE, TransH, ComplEx, and DistMult on both triple classification and weight-aware triple classification across all three datasets.

Table 5

Results of base models and their weight-aware extension models on the triple classification task and weight-aware triple classification task. "$\exp_{sta.}$" indicates the extended models adopt an exponential function with a static base, while "$\exp_{dyn.}$" refers to the extended models adopt an exponential function with a dynamic base.

	Model		F1	WaF1	Model		F1	WaF1	Model		F1	WaF1
CN15K	TransE		0.2554	0.2826	TransE		0.3486	0.3605	TransE		0.603	0.6094
	WaTransE	linear	0.2015	0.2663	WaTransE	linear	0.3097	0.32	WaTransE	linear	0.5444	0.5806
		$\exp_{sta.}$	**0.3179**	**0.3868**		$\exp_{sta.}$	0.3622	**0.3748**		$\exp_{sta.}$	0.6183	0.6233
		$\exp_{dyn.}$	0.2552	0.2833		$\exp_{dyn.}$	**0.3672**	0.3739		$\exp_{dyn.}$	**0.6327**	**0.6423**
	TransH		0.0937	0.1042	TransH		0.1446	0.152	TransH		0.4872	0.51
	WaTransH	linear	0.1302	0.1326	WaTransH	linear	0.2898	0.2894	WaTransH	linear	0.438	**0.5794**
		$\exp_{sta.}$	0.1825	0.187		$\exp_{sta.}$	0.3078	0.3078		$\exp_{sta.}$	**0.5664**	0.5569
		$\exp_{dyn.}$	**0.1837**	**0.1886**		$\exp_{dyn.}$	**0.3176**	**0.3153**		$\exp_{dyn.}$	0.5548	0.563
	ComplEx		0.435	0.4856	ComplEx		0.3743	0.398	ComplEx		0.9762	1.0171
	WaComplEx	linear	0.4015	**0.7508**	WaComplEx	linear	0.3486	0.4484	WaComplEx	linear	0.9475	**1.2222**
		$\exp_{sta.}$	0.4238	0.4987		$\exp_{sta.}$	**0.5641**	**0.6126**		$\exp_{sta.}$	**0.9777**	1.0216
		$\exp_{dyn.}$	**0.4387**	0.5057		$\exp_{dyn.}$	0.3808	0.405		$\exp_{dyn.}$	0.9772	1.021
	DistMult		0.165	0.1851	DistMult		0.1873	0.2004	DistMult		0.6943	0.713
	WaDistMult	linear	0.198	0.2363	WaDistMult	linear	0.2604	0.3	WaDistMult	linear	0.6862	0.7341
		$\exp_{sta.}$	**0.2088**	0.2474		$\exp_{sta.}$	0.2773	0.3202		$\exp_{sta.}$	0.7011	0.7317
		$\exp_{dyn.}$	0.207	**0.2494**		$\exp_{dyn.}$	**0.318**	**0.3576**		$\exp_{dyn.}$	**0.7108**	**0.7495**

5. Conclusion

In this paper, we originally explore the weight-aware link prediction task and propose three evaluation metrics for weight-aware link prediction (Section 3.1). We also originally explore the weight-aware triple classification task and propose weight-aware F1 score as the three evaluation metrics (Section 3.2). With respect to the novel tasks, we propose a method to extend deterministic knowledge graph embedding models to their weight-aware version, and provide the weight-aware extensions for the base models (Section 3.3).

The weight-aware tasks emphasize the ability of knowledge graph embedding models to correctly predict and classify triples according to the weights of the triples, which is critical for applications in some scenarios that involve non-deterministic knowledge, such as text understanding and protein-protein interaction.

We propose a general framework WaExt for extending the deterministic knowledge graph embedding models to learn weight-aware embeddings from weighted knowledge graphs. To illustrate its usage, we apply WaExt to TransE, TransH, ComplEx, and DistMult, and get the weight-aware extensions for them (i.e., WaTransE, WaTransH, WaComplEx, and WaDisMult, respectively). The weight-aware extensions of the base models can learn embeddings better from triples with high weights and outperform baseline models in link prediction, triple classification, and weight-aware tasks. Our extensive experiments reveal that the exponential activation function and the linear activation function is effective for WaExt. We will explore more suitable activation functions in the future.

Acknowledgements

This work was supported by JST SPRING, Grant Number JPMJSP2102. This work was also partially supported by JSPS Grant-in-Aid for Early-Career Scientists (Grant Number 22K18004), JSPS Grant-in-Aid for Scientific Research (Grant Number 21K12042), and the New Energy and Industrial Technology Development Organization (Grant Number JPNP20006).

References

[1] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P.N. Mendes, S. Hellmann, M. Morsey, P. van Kleef, S. Auer and C. Bizer, DBpedia - A large-scale, multilingual knowledge base extracted from Wikipedia, *Semantic Web* **6** (2015), 167–195.
[2] T.P. Tanon, G. Weikum and F.M. Suchanek, YAGO 4: A Reason-able Knowledge Base, *The Semantic Web* **12123** (2020), 583–596.
[3] D. Vrandeić and M. Krötzsch, Wikidata: a free collaborative knowledgebase, *Commun. ACM* **57** (2014), 78–85.
[4] T.M. Mitchell, W.W. Cohen, E.R. Hruschka, P.P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, J. Krishnamurthy, N. Lao, K. Mazaitis, T. Mohamed, N. Nakashole, E.A. Platanios, A. Ritter, M. Samadi, B. Settles, R.C. Wang, D. Wijaya, A.K. Gupta, X. Chen, A. Saparov, M. Greaves and J. Welling, Never-Ending Learning, *Communications of the ACM* **61** (2015), 103–115.
[5] X. Dong, E. Gabrilovich, G. Heitz, W. Horn, N. Lao, K.P. Murphy, T. Strohmann, S. Sun and W. Zhang, Knowledge vault: a web-scale approach to probabilistic knowledge fusion, *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining* (2014).
[6] L. Dietz, A. Kotov and E. Meij, Utilizing Knowledge Graphs for Text-Centric Information Retrieval, *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018).
[7] S. Hu, L. Zou, J.X. Yu, H. Wang and D. Zhao, Answering Natural Language Questions by Subgraph Matching over Knowledge Graphs, *IEEE Transactions on Knowledge and Data Engineering* **30** (2018), 824–837.
[8] X. Huang, J. Zhang, D. Li and P. Li, Knowledge Graph Embedding Based Question Answering, *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (2019).
[9] S. Zhou, X. Dai, H. Chen, W. Zhang, K. Ren, R. Tang, X. He and Y. Yu, Interactive Recommender System via Knowledge Graph-enhanced Reinforcement Learning, *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).
[10] Q. Guo, F. Zhuang, C. Qin, H. Zhu, X. Xie, H. Xiong and Q. He, A Survey on Knowledge Graph-Based Recommender Systems, *IEEE Transactions on Knowledge and Data Engineering* **34** (2022), 3549–3568.
[11] C. Rudnik, T. Ehrhart, O. Ferret, D. Teyssou, R. Troncy and X. Tannier, Searching News Articles Using an Event Knowledge Graph Leveraged by Wikidata, *Companion Proceedings of The 2019 World Wide Web Conference* (2019).
[12] P. Ernst, C. Meng, A. Siu and G. Weikum, KnowLife: A knowledge graph for health and life sciences, *2014 IEEE 30th International Conference on Data Engineering* (2014), 1254–1257.
[13] B. Taskar, M.F. Wong, P. Abbeel and D. Koller, Link Prediction in Relational Data, in: *NIPS*, 2003.
[14] R. Speer, J. Chin and C. Havasi, ConceptNet 5.5: An Open Multilingual Graph of General Knowledge, in: *AAAI*, 2017.
[15] W. Wu, H. Li, H. Wang and K.Q. Zhu, Probase: a probabilistic taxonomy for text understanding, *SIGMOD* (2012).
[16] O.D. la Cruz Cabrera, M. Matar and L. Reichel, Edge importance in a network via line graphs and the matrix exponential, *Numerical Algorithms* **83** (2019), 807–832.
[17] D. Szklarczyk, A. Franceschini, S. Wyder, K. Forslund, D. Heller, J. Huerta-Cepas, M. Simonovic, A.C.J. Roth, A. Santos, K. Tsafou, M. Kuhn, P. Bork, L.J. Jensen and C. von Mering, STRING v10: protein–protein interaction networks, integrated over the tree of life, *Nucleic Acids Research* **43** (2015), D447–D452.
[18] Z. Wang, H. Wang, J.-R. Wen and Y. Xiao, An Inference Approach to Basic Level of Categorization, *CIKM* (2015).
[19] Y. Wang, H. Li, H. Wang and K.Q. Zhu, Concept-Based Web Search, in: *ER*, 2012.
[20] J. Ivanic, A. Wallqvist and J. Reifman, Evidence of probabilistic behaviour in protein interaction networks, *BMC Systems Biology* **2** (2007), 11–11.
[21] Q. Wang, Z. Mao, B. Wang and L. Guo, Knowledge Graph Embedding: A Survey of Approaches and Applications, *IEEE Transactions on Knowledge and Data Engineering* **29** (2017), 2724–2743.
[22] A. Bordes, N. Usunier, A. García-Durán, J. Weston and O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, in: *NIPS*, 2013.
[23] Z. Wang, J. Zhang, J. Feng and Z. Chen, Knowledge Graph Embedding by Translating on Hyperplanes, in: *AAAI*, 2014.
[24] B. Yang, S.W.-t. Yih, X. He, J. Gao and L. Deng, Embedding Entities and Relations for Learning and Inference in Knowledge Bases, in: *Proceedings of the International Conference on Learning Representations*, 2015.
[25] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier and G. Bouchard, Complex embeddings for simple link prediction, in: *International conference on machine learning*, PMLR, 2016, pp. 2071–2080.
[26] F. Scarselli, M. Gori, A.C. Tsoi, M. Hagenbuchner and G. Monfardini, The Graph Neural Network Model, *IEEE Transactions on Neural Networks* **20** (2009), 61–80.
[27] Y. Hu, K. Janowicz, P. Hitzler and K. Sengupta, The Semantic Web Journal as Linked Data., in: *ISWC (Posters & Demos)*, 2015.

[28] D. Van Assche, T. Delva, G. Haesendonck, P. Heyvaert, B. De Meester and A. Dimou, Declarative RDF graph generation from heterogeneous (semi-) structured data: A systematic literature review, *Journal of Web Semantics* (2022), 100753.
[29] M. Schlichtkrull, T. Kipf, P. Bloem, R. van den Berg, I. Titov and M. Welling, Modeling Relational Data with Graph Convolutional Networks, *ArXiv* **abs/1703.06103** (2018).
[30] S. Vashishth, S. Sanyal, V. Nitin and P.P. Talukdar, Composition-based Multi-Relational Graph Convolutional Networks, *ArXiv* **abs/1911.03082** (2020).
[31] D. Nathani, J. Chauhan, C. Sharma and M. Kaul, Learning Attention-based Embeddings for Relation Prediction in Knowledge Graphs, in: *ACL*, 2019.
[32] S. Pai and L. Costabello, Learning Embeddings from Knowledge Graphs With Numeric Edge Attributes, in: *IJCAI*, 2021.
[33] M. Nayyeri, G.M. Çil, S. Vahdati, F. Osborne, A. Kravchenko, S. Angioni, A. Salatino, D.R. Recupero, E. Motta and J. Lehmann, Link Prediction of Weighted Triples for Knowledge Graph Completion Within the Scholarly Domain, *IEEE Access* **9** (2021), 116002–116014.
[34] M. Seo and K.Y. Lee, A Graph Embedding Technique for Weighted Graphs Based on LSTM Autoencoders, *J. Inf. Process. Syst.* **16** (2020), 1407–1423.
[35] X. Wu, H. Pang, Y. Fan, L. Yang and Y. Luo, ProbWalk: A random walk approach in weighted graph embedding, *Procedia Computer Science* **183** (2021), 683–689.
[36] G. Mai, K. Janowicz and B. Yan, Support and Centrality: Learning Weights for Knowledge Graph Embedding Models, in: *International Conference Knowledge Engineering and Knowledge Management*, 2018.
[37] L.A. Adamic and E. Adar, Friends and neighbors on the Web, *Soc. Networks* **25** (2003), 211–230.
[38] H. Cho and Y. Yu, Link prediction for interdisciplinary collaboration via co-authorship network, *Social Network Analysis and Mining* **8** (2018), 1–12.
[39] E.M. Airoldi, D.M. Blei, S.E. Fienberg and E.P. Xing, Mixed Membership Stochastic Block Models for Relational Data with Application to Protein-Protein Interactions, 2006.
[40] A. Bordes, J. Weston, R. Collobert and Y. Bengio, Learning Structured Embeddings of Knowledge Bases, in: *AAAI*, 2011.
[41] E.M. Voorhees, The TREC-8 Question Answering Track, *Natural Language Engineering* **7** (2000), 361–378.
[42] R. Socher, D. Chen, C.D. Manning and A. Ng, Reasoning With Neural Tensor Networks for Knowledge Base Completion, in: *NIPS*, 2013.
[43] X. Chen, M. Chen, W. Shi, Y. Sun and C. Zaniolo, Embedding Uncertain Knowledge Graphs, in: *AAAI*, 2019.
[44] T.-Y. Liu, Learning to rank for information retrieval, *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (2009).
[45] T. Dettmers, P. Minervini, P. Stenetorp and S. Riedel, Convolutional 2D Knowledge Graph Embeddings, in: *AAAI*, 2018.
[46] M. Ali, M. Berrendorf, C.T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp and J. Lehmann, PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings, *J. Mach. Learn. Res.* **22** (2021), 82:1–82:6.
[47] D. Liben-Nowell and J.M. Kleinberg, The link-prediction problem for social networks, *Journal of the Association for Information Science and Technology* **58** (2007), 1019–1031.
[48] W.-t. Yih, M.-W. Chang, C. Meek and A. Pastusiak, Question Answering Using Enhanced Lexical Semantic Models, in: *ACL*, 2013.
[49] M. Chen, Y. Tian, X. Chen, Z. Xue and C. Zaniolo, On2Vec: Embedding-based Relation Prediction for Ontology Population, in: *SDM*, 2018.
[50] G. Ji, S. He, L. Xu, K. Liu and J. Zhao, Knowledge Graph Embedding via Dynamic Mapping Matrix, in: *ACL*, 2015.
[51] Y. Lin, Z. Liu, M. Sun, Y. Liu and X. Zhu, Learning Entity and Relation Embeddings for Knowledge Graph Completion, in: *AAAI*, 2015.
[52] M.A. Hasan, V. Chaoji, S. Salem and M.J. Zaki, Link prediction using supervised learning, 2006.
[53] A. Kimmig, S.H. Bach, M. Broecheler, B. Huang and L. Getoor, A short introduction to probabilistic soft logic, in: *NIPS 2012*, 2012.
[54] Z. Ye, Y.J. Kumar, G.O. Sing, F. Song and J. Wang, A Comprehensive Survey of Graph Neural Networks for Knowledge Graphs, *IEEE Access* **10** (2022), 75729–75741. doi:10.1109/ACCESS.2022.3191784.
[55] J. Ivanic, A. Wallqvist and J. Reifman, Probing the Extent of Randomness in Protein Interaction Networks, *PLoS Computational Biology* **4** (2008).
[56] Z. Chen, M.-Y. Yeh and T.-W. Kuo, PASSLEAF: A Pool-bAsed Semi-Supervised LEArning Framework for Uncertain Knowledge Graph Embedding, in: *AAAI*, 2021.
[57] J. Hu, R. Cheng, Z. Huang, Y. Fang and S. Luo, On Embedding Uncertain Graphs, *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017).
[58] M. Nickel, L. Rosasco and T.A. Poggio, Holographic Embeddings of Knowledge Graphs (2016).