Doctoral Dissertation

Knowledge Graph Representation Learning: Variants and Applications

KONG Wei Kun

Supervisor	NGUYEN Le-Minh
Main Examiner	NGUYEN Le-Minh
Examiners	TOJO Satoshi
	SHIRAI Kiyoaki
	HASEGAWA Shinobu
	MA Qiang

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
[Information Science]

(March, 2023)

Abstract

In the 1960s, a prototype of the knowledge graph was proposed to enable formal reasoning and code representations of knowledge. Decades of development followed, with particularly significant progress in the past ten years. The release of large-scale knowledge graphs and the springing up of powerful embedding models have sparked the enthusiasm of researchers. Today, knowledge graphs are being applied in many fields, including natural language processing, autonomous driving, biology, and finance.

Knowledge graph representation learning, also known as knowledge graph embedding, aims to represent a knowledge graph using a set of vectors and matrices in a low-dimensional vector space. This is essential for utilizing knowledge graphs in deep learning models. Currently, most available knowledge graph embedding models only embed entities and relations using the triples provided by the knowledge graphs. This approach does not make full use of available resources. However, some knowledge graph representation models have been proposed to learn knowledge graph embeddings using not only the facts in the knowledge graph but also additional useful information, such as entity type, entity description, and logic rules.

In many scenarios, the interactions between entities are inherently associated with different uncertainties, frequencies, or intensities. For example, the interaction possibility between two proteins and the importance of friends in a social network can vary greatly. Weighted knowledge graphs extend deterministic knowledge graphs by associating a weight with the triples to formalize the weighted interactions between entities. Many weighted knowledge graphs have been published, which has led to an increased focus on the weighted knowledge graph and its embeddings.

Our research focuses on two main areas: learning better representations from weighted knowledge graphs, and utilizing these representations in downstream tasks. To improve representations from weighted knowledge graphs, we explore *weight-aware knowledge graph embedding* and *weighted knowledge graph embedding*. Weight-aware knowledge graph embedding involves learning embeddings for a deterministic knowledge graph with the aid of weight information from triples. However, the learned embedding cannot deduce the weight of the triple. On the other hand, weighted knowledge graph embedding is used to embed the weighted knowledge graph, with the ability to deduce not only the triples but also their weights.

To extend the existing embedding models for deterministic knowledge graphs to learn weight-aware embeddings and weighted embeddings from

weighted knowledge graphs, we propose two general frameworks, *WaExt* and *WeExt*, respectively. To evaluate the learned embeddings from weighted knowledge graphs, we introduce two evaluation tasks, *weight-aware link prediction* and *weight-aware triple classification* for weight-aware knowledge graph embedding, and *weighted link prediction* for weighted knowledge graph embedding. For utilizing the representations in downstream tasks, we propose a framework *KGWE* to fine-tune word embeddings using knowledge graph embeddings.

The three proposed frameworks outperform the baselines on the target tasks, indicating their effectiveness in improving the performance of knowledge graph embeddings. Furthermore, the evaluation tasks introduced in this study provide a more comprehensive evaluation of these embeddings.

Keywords: Artificial Intelligence, Weighted Knowledge Graph, Knowledge Graph Embedding, Evaluation Tasks, Word Embedding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

1.1.1 Knowledge Graph and Knowledge Graph Embedding

The knowledge graph (KG) is a set of knowledge organized in labeled simple graphs, where the vertices represent entities, and the edges represent relations between the entities. Numerous large-scale knowledge graphs have been released, such as DBpedia [1], NELL [2], ConceptNet [3], YAGO [4], Freebase [5], STRING [6], Probase [7], involving general knowledge, knowledge for natural language understanding, biology knowledge, etc. Knowledge graphs are widely used in various applications and fields [8, 9, 10]. Typical applications include question-answering systems [11, 12, 13, 14, 15], recommendation systems [16, 17, 18, 19, 20], and information retrieval [21, 22, 23, 24], etc. The fields involved include medical science [25, 26], cybersecurity [27, 28, 29, 30], finance [31, 32, 29, 30], and education [33, 34], etc.

The knowledge graph not only attracts researchers' attention but also contributes significant economic value. An increasing number of companies are constructing and utilizing knowledge graphs in their business. Google, for example, leverages the knowledge graph to improve the accuracy of web searches, gain a better understanding of user queries, and recommend relevant things to the query to users [35]. Alibaba utilizes knowledge graphs to better understand consumers' needs [36], while LinkedIn uses them to optimize advertisements and recommend jobs and people to members [37].

In order to improve the quality of knowledge graphs and utilize them in deep learning models, a series of problems have been studied, such as knowledge graph embedding (KGE) [38, 39], knowledge graph completion

(KGC) [40], knowledge graph alignment (KGA) [41], etc. The critical core problem of leveraging knowledge graphs in deep learning models is how to represent them with a set of dense vectors in a low-dimensional vector space, which is the objective of knowledge graph embedding. KGE models encode entities as low-dimensional vectors and relations as operations on these entities to preserve the structure of knowledge graphs in the embedding space. In recent years, KGE models have flourished and facilitated multiple knowledge-driven tasks [13, 14, 16, 17, 19, 22].

1.1.2 Fact-alone Embedding Models

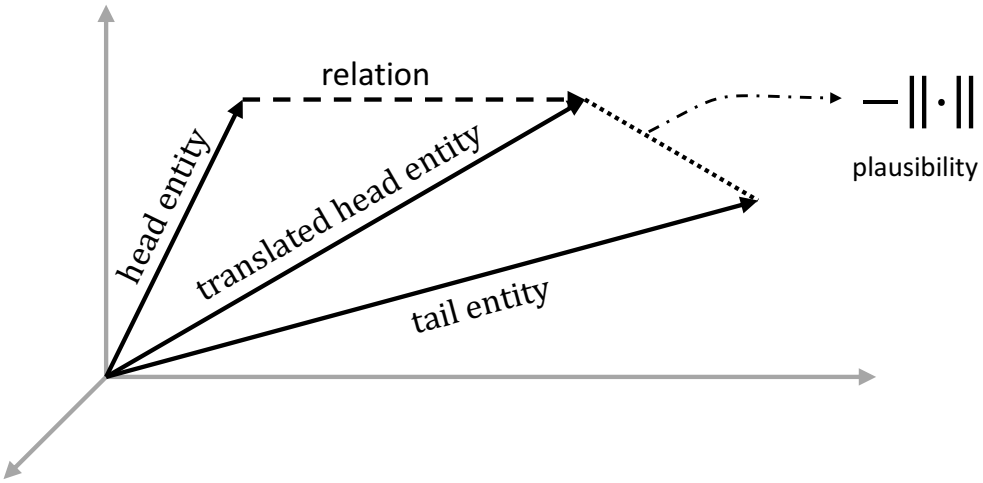


Figure 1.1: An illustration of TransE.

Most currently available knowledge graph embedding models only use the triples provided by the knowledge graphs to embed entities and relations. These models assign high plausibility to positive triples and low plausibility to negative triples. We refer to these models as fact-alone knowledge graph embedding models, following [9]. Knowledge graph embedding models can be further divided into translational distance models and bilinear models based on the different interaction modes of entities and relationships in one triple.

The translational distance model, including TransE [42], TransH [43], and TransR [44], regards the relation as a translation operation from the head entity to the tail entity, and utilizes a distance-based scoring function to measure the plausibility of triples. It is worth noting that a vector with a fixed magnitude and direction but an unfixed point of application is called a free vector, while a vector with a unique magnitude, direction, and point of

application is called a fixed vector. The translational distance model represents entities using fixed vectors and relations using free vectors, which carry distinct implications. The free vectors are utilized to represent translation operations towards the fixed vectors in the vector space, highlighting the distinct meanings attributed to entities and relations within the model.

The bilinear models, such as RESCAL [45], DistMult [46], and HolE [47], are based on tensor factorization and model the interaction of entities and relations by vector-matrix product. These models obtain high expressive power due to the use of a full-rank matrix for each relation in the scoring functions, which are in the form of $h^\top W_r t$.

Figure 1.1 illustrates TransE, the originator of translational distance models. In TransE, entities are represented as vectors in the target embedding space. The relation is modeled as a translation operation from the head entity to obtain the translated head entity. The plausibility of the triple is then determined as the negative distance between the translated head entity and the tail entity.

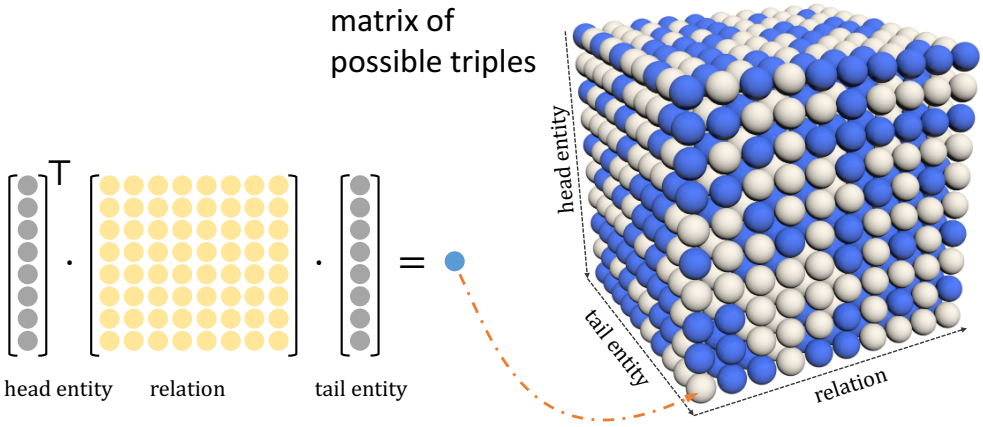


Figure 1.2: An illustration of RESCAL.

Figure 1.2 illustrates RESCAL, the originator of bilinear models. In RESCAL, plausibilities of all possible triples $(h, r, t) \equiv E * R * E$ are represented as elements of a 3-dimensional plausibility matrix, where 1 indicates positive triples and 0 indicates negative triples. Entities are represented as vectors and relations are represented as 2-dimensional matrices in the target embedding space. RESCAL learns the entity embeddings and relation matrix by decomposing the 3-dimensional plausibility matrix.

1.1.3 Embedding Models with Additional Information

Besides the triples provided by knowledge graphs, there is a wide range of additional information that can be utilized to enhance the performance of fact-alone embedding models. The available additional information includes entity types, textual descriptions, as well as logical rules.

Type-aided KGE

In heterogeneous graphs [48] or knowledge graphs that contain relation "is-a", the type of majority entities are known. For example, for a triple

$$(\text{Isaac Newton}, \textit{works-written}, \text{Optics})$$

its related type information is also given:

$$(\text{Isaac Newton}, \textit{is-a}, \text{physicist})$$

$$(\text{Optics}, \textit{is-a}, \text{book})$$

The fact-alone models regard the type of entities as an ordinary relation and the corresponding triples as standard training examples, which can not make full use of the type information of the entities.

SSE [49] introduces type information to knowledge graph embeddings by requiring entities of the same type to be close to each other in the embedding space. TKRL [50] incorporates hierarchical entity categories and multiple category labels to knowledge graph embeddings through type-specific entity projections. [51] utilizes entity type as constraints in the training phase to exclude negative examples with incorrect entity types.

Description-aided KGE

In some knowledge graphs, entities have concise descriptions that contain rich semantic information about them. In addition to entity descriptions stored in knowledge graphs, textual descriptions, such as Wikipedia articles, can also be useful for training knowledge graph embeddings. For example, in Freebase, the entity "Dalian" has the following description:

> Dalian is a major city and seaport in the south of Liaoning province. It is the southernmost city of Northeast China and China's northernmost warm water port, at the tip of the Liaodong peninsula. Dalian is the province's second-largest city and has sub-provincial administrative status; only the provincial capital is larger. The Shandong peninsula lies southwest across the Bohai

> Sea; Korea lies across the Yellow Sea to the east. Today a financial, shipping, and logistics center for Northeast Asia, Dalian has a significant history of being used by foreign powers for its ports: Dalian proper was previously known as both Dalny and Dairen but it was better known as both Port Arthur and Ryojun from its Lüshunkou district. In 2006, Dalian was named China's most livable city by China Daily.

NTN [52] learns word vectors from an auxiliary news corpus and initializes the entity embeddings by averaging the word embeddings contained in the name of the entity. [53] proposed a joint model that aligns the given KG with an auxiliary text corpus and conducts KG embedding and word embedding jointly. The alignment mechanisms include alignment by entity names, Wikipedia anchors, and entity descriptions. By aligning these two types of information, jointly embedding enables the prediction of out-of-KG entities. DKRL [54] associates each entity with a structure-based embedding and a description-based embedding, capturing structural information conveyed in KG facts and textual information expressed in the entity description, respectively. TEKE [55] annotates entities in a given text corpus and constructs a co-occurrence network composed of entities and words. TEKE defines an entity's textual context as its neighbors in the co-occurrence network and textual context for a relation as the common neighbors of its head entity and tail entity. The weighted average of the word embeddings in the textual context is incorporated into fact-alone embeddings to learn more expressive entity and relation representations.

Logical rules-aided KGE

Logical rules contain rich background information and have been widely studied in knowledge representation and reasoning [56]. The logic rules are extremely useful information for reasoning over knowledge graphs. For example, if the triple

$$(\text{Bei Jing}, \mathit{Capital\text{-}Of}, \text{China})$$

and the rule

$$\forall x, y : (x, \mathit{Capital\text{-}Of}, y) \rightarrow (x, \mathit{Located\text{-}In}, y)$$

are known, we can know that $(\text{Bei Jing}, \mathit{Located\text{-}In}, \text{China})$ is also true.

KALE [57] and [58] represent facts and rules in a unified framework, as atomic and complex formulae respectively. Each triple is assigned a truth value according to its plausibility. Logical rules are first instantiated into

ground rules, and then ground rules are then interpreted as complex formulae constructed by combining triples with logical connectives, and modeled by t-norm fuzzy logic [59]. The truth value of a ground rule is a composition of the truth values of the constituent triples, indicating to what degree the ground rule is satisfied.

1.2 Motivation

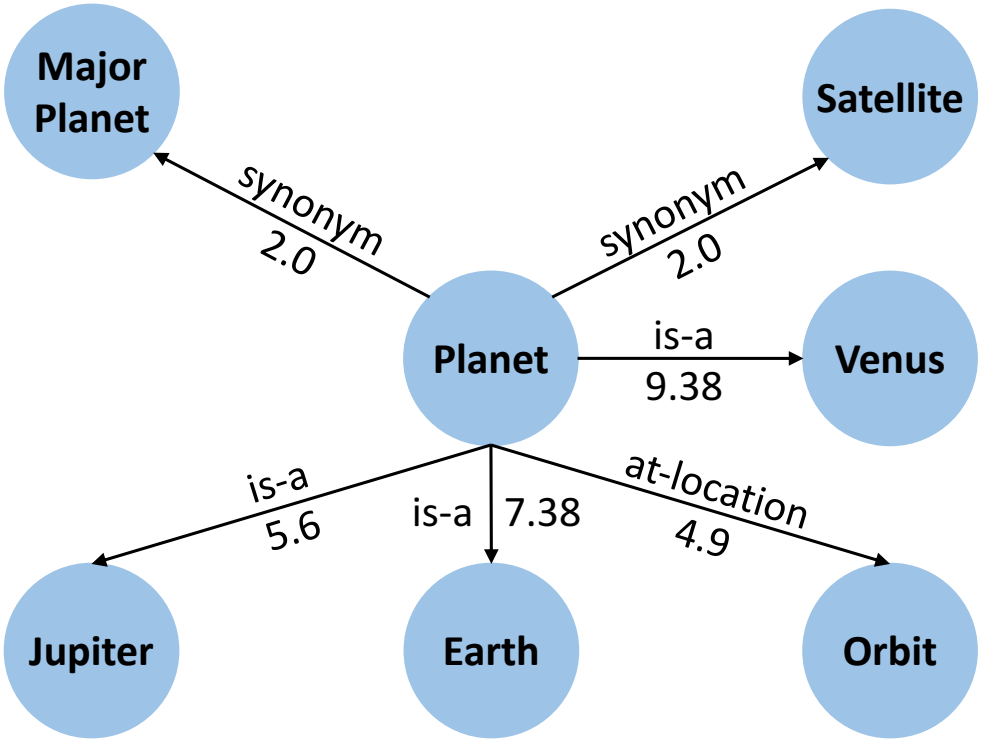


Figure 1.3: An example of the weighted knowledge graph taken from ConceptNet.

In addition to the aforementioned additional information, the weight information of triples has also attracted increasing attention from researchers. In many scenarios, the interactions between entities inherently are associated with different uncertainties, frequencies, or intensities. Without considering the specific semantics, we call all numbers representing link strength, uncertainty, and frequency as weights of triples. For example, in a social network, people interact with all their friends, but especially frequently interact with

their close friends. In biology, the possibility of interactions between proteins is generally proportional to the product of their numbers of interacting partners or degrees. There is an illustration of the weighted knowledge graph in Figure 1.3.

Many weighted knowledge graphs, such as ConceptNet [3], NELL [2], and STRING [6], have been published, making triple-weight information receive increasing focus. Follow [60], to distinguish the knowledge graph from the weighted knowledge graph, we refer to the knowledge graph without weight information as the deterministic knowledge graph in the following. Some embedding models, Such as UKGE [60], PASSLEAF [61], learn weight information from the weighted knowledge graphs. FocusE [62] utilizes the weight information and a nonlinear function to rescale the contribution by each triple to the total loss to train better knowledge graph embeddings.

However, UKGE and PASSLEAF only adopt low-expressive nonlinear functions to fit the weights of triples, which limits the model's ability in weight prediction. UKGE and PASSLEAF are evaluated in link prediction and weight prediction respectively, which can not show the performance of the model in these two tasks simultaneously. FocusE rescales the losses on triples based on their weights using a nonlinear function under the assumption that the importance of the triple varies according to the weight of the triple. However, FocusE is evaluated only on the non-weight-aware link prediction task, which fails to reflect the weight distribution of the correctly predicted triples.

1.3 Main Contribution

In order to learn more effective knowledge graph embeddings with the aid of weight information and utilize knowledge graph embeddings to downstream tasks more effectively, we have done the following works:

- We propose WaExt, a general framework for extending non-weight-aware knowledge graph embedding models to their weight-aware version. We explore more effective nonlinear functions for rescaling the losses on triples with different weights.
- To evaluate the models under the assumption that the importance of the triple for learning knowledge graph embeddings varies according to the weight of the triple, we introduce weight-aware link prediction and weight-aware triple classification tasks. Weight-aware link prediction and weight-aware triple classification are weight-aware extensions for

link prediction and triple classification, aiming for more comprehensively evaluating knowledge graph embeddings learned from weighted knowledge graphs.

- We propose WeExt, a general framework that extends the deterministic knowledge graph embedding models to enable them to embed weighted knowledge graphs. The models extended by WeExt can learn not only the plausibility of the triple but also the weight of the triple.
- To better evaluate the weighted knowledge embeddings, we introduce the weighted link prediction task. Unlike the existing works evaluating the models in link prediction and weight prediction asynchronously, weighted link prediction evaluates the models' performance in predicting the link attached to a weight. Weighted link prediction synchronously reflects the performance of the model in predicting links and predicting weights.
- We propose KGWE, a general framework that can fine-tune word embeddings using knowledge graph embeddings. The word embeddings finetuned by KGWE achieves better performance on the word similarity task and several downstream tasks.

1.4 Structure of the Dissertation

The remaining parts of the dissertation are organized as follows:

- Chapter 2 gives the definitions of some essential terminologies including knowledge, graphs, knowledge graphs, and knowledge graph embedding. We first discuss the classification of knowledge from the perspectives of philosophy and computer science and clarify the type of knowledge involved in knowledge graphs. To enable researchers with different knowledge backgrounds to better understand and utilize knowledge graphs, we give a compendious definition of the knowledge graph which contains the most essential factors. We also briefly review the development of knowledge graphs in the past 40 years.
- Chapter 3 introduces the proposed weight-aware extending framework WaExt and the evaluation tasks including the weight-aware link prediction task and weight-aware triple classification task. To demonstrate the effectiveness of the proposed framework, we conduct experiments on existing representative deterministic knowledge graph embedding models.

- Chapter 4 introduces WeExt, a general framework to encode not only the plausibility of the triple but also the weight information of the triple into knowledge graph embeddings. We describe the weighted link prediction task from the evaluation protocol to metrics. We extend two representative translational distance models and two representative bilinear models using WeExt. The extensions achieve competitive performance to the baselines in link prediction, weight prediction, and weighted link prediction.

- Chapter 5 explores a possible approach to utilize the knowledge graph embeddings in downstream tasks. We propose KGWE, a general framework to fine-tune the word embeddings under the guide of knowledge graph embeddings.

- Chapter 6 reviews the conducted works and their significance and discusses further directions.

Figure 1.4 shows an illustration of Chapter 3, Chapter 4, and Chapter 5.

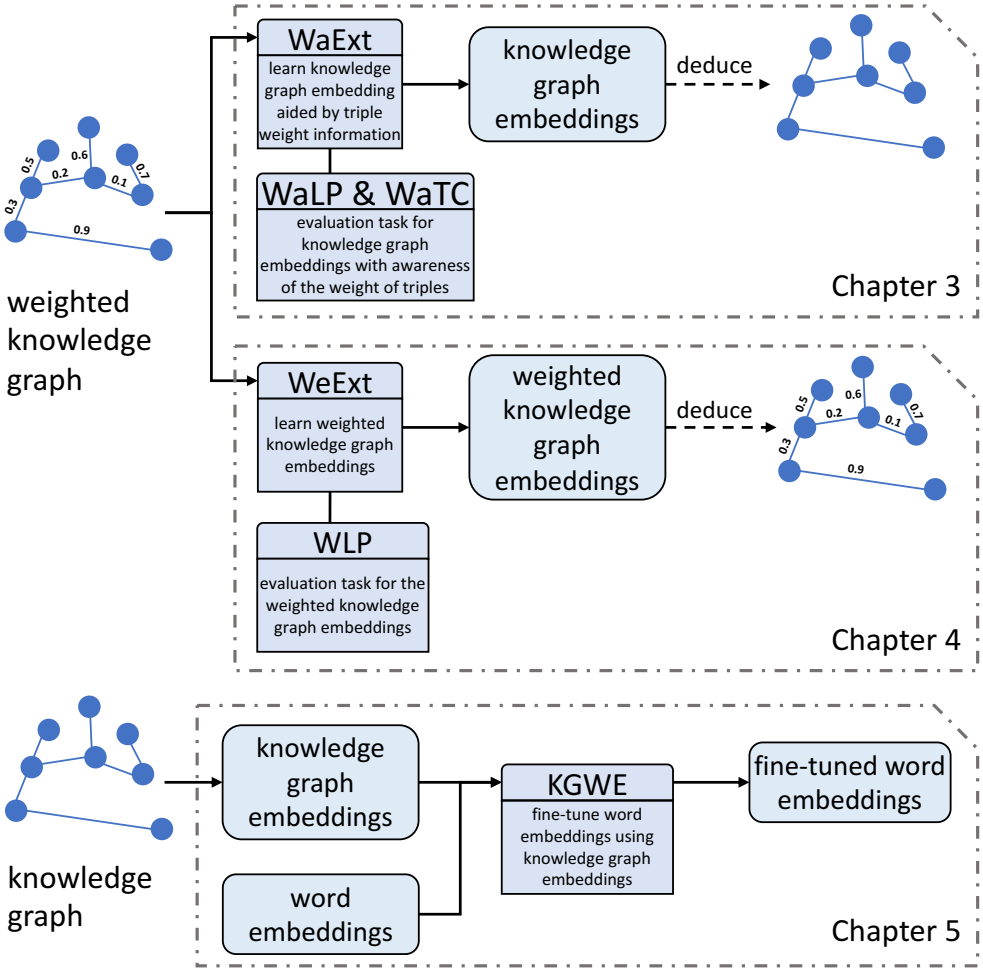


Figure 1.4: An illustration of Chapter 3, Chapter 4, and Chapter 5.

Chapter 2

Preliminary

2.1 Knowledge Graph

2.1.1 Knowledge

Knowledge is familiarity with objects in the real world, understanding facts in the real world, or practical skills. Knowledge has been widely discussed by philosophers and computer scientists [63, 64, 65, 66, 67, 68]. We sort out the various existing definitions of knowledge and try to give a definition of knowledge from a computer science researcher's perspective.

Philosophical Perspective

From a philosophical perspective, there are three different kinds of knowledge: acquaintance knowledge, propositional knowledge (knowledge-that), and knowledge-how.

Acquaintance knowledge [63, 65] can be defined by Definition 1:

Definition 1 (Acquaintance Knowledge)**.** *We have acquaintance with anything of which we are directly aware, without the intermediary of any process of inference or any knowledge of truths.*

For example, we know our mothers, our friends, our pets, etc., by being acquainted with them.

Propositional knowledge [64] can be defined Definition 2:

Definition 2 (Propositional Knowledge)**.** *Let S be a knowing subject. p is a piece of propositional knowledge if and only if:*

- *p is true.*

- *S is justified in believing that p.*
- *S believes that p.*

Propositional knowledge is knowledge of facts. We acquire propositional knowledge when we learn that, for example, we know Bei Jing is the capital of China.

Knowledge-how [69] can be defined by Definition 3:

Definition 3 (Knowledge-how)**.** *Knowledge-how is the specific knowledge one possesses when one can truly be described as knowing how to do something.*

Knowledge-how is knowledge of ability and experience. We acquire knowledge-how about playing ping-pong (table tennis) when we truly know how to play ping-pong.

Computer Science Perspective

From the knowledge level [68] perspective, an intelligent system is described as an agent that processes its knowledge to determine the actions based on the principle of rationality to take to achieve its goals. The principle of rationality is that if an agent has knowledge that one of its actions will lead to one of its goals, then the agent will select that action. Knowledge, as the medium at the knowledge level, is defined as whatever can be ascribed to an agent, such that its behavior can be computed according to the principle of rationality.

Compared with the more general definition at the knowledge level, the definition of knowledge in knowledge engineering [66, 67, 70] is more detailed:

Definition 4 (Knowledge)**.** *Knowledge is a human understanding of a subject matter that has been acquired through proper study or practice. Knowledge can be divided into explicit knowledge and tacit knowledge:*

- *Explicit knowledge: the knowledge that is easy to articulate, capture, and codify.*
- *Tacit knowledge is generally personal, subjective, resided in people's heads or muscle memory and is hard to express and capture.*

Explicit knowledge is derived from information, which is derived from data. Explicit knowledge is usually people's cognition of objects, while implicit knowledge is mainly human practical skills and experience.

Explicit knowledge in computer science corresponds to propositional knowledge in philosophy, which is the province of knowledge graphs. Tacit knowledge is more like knowledge-how, which is an area that deep learning models are good at. For example, the medical imaging models based on deep learning are comparable to experienced doctors [71].

2.1.2 Graph

The graph is a concept in mathematics used to model pairwise relations between objects. The graph data structure in computer science is an implementation of the graph in mathematics. Adopting the graph as the data structure of knowledge brings a number of benefits when compared with relational models. For example, graphs provide a concise and intuitive abstraction for a variety of domains [72]. Graphs also allow for postponing the definition of a schema and organizing the data in a more flexible manner [73].

Let $\mathcal{P}_k(V)$ be the set of all k-element subsets of the set V, the graph can be defined as [74]:

Definition 5 (Graph)**.** *A graph is a triple $G = (V, E, \phi)$ where*

- *V is a finite set of vertices,*
- *E is a finite set of edges,*
- *ϕ is a function with domain E and codomain $\mathcal{P}_2(V)$.*

According to Definition 5, a graph allows more than one edge between two vertices. For example, in a traffic network graph in a region, there may be multiple roads between two cities. An illustration of a graph is shown in Figure 2.1.

A graph allows multiple edges between two vertices. For example, there may be multiple roads between two cities in a traffic graph. A simple graph removes multiple edges between two vertices in a graph, leaving only one of the edges:

Definition 6 (Simple Graph)**.** *A simple graph G is a pair $G = (V, E)$ where*

- *V is a finite set, called the vertices of G,*
- *E is a subset of $\mathcal{P}_2(V)$ (i.e., is a set E of two-element subsets of V), called the edges of G.*

Definition 6 describes the essential factors of a simple graph: a set of vertices and a set of non-repetitive edges connecting the vertices. An illustration of a simple graph is shown in Figure 2.2.

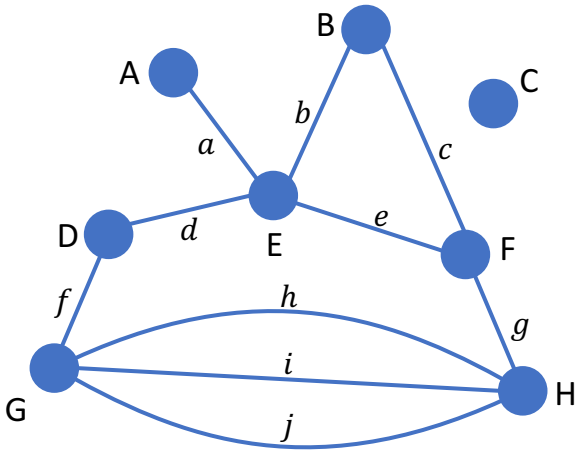


Figure 2.1: An illustration of the graph.

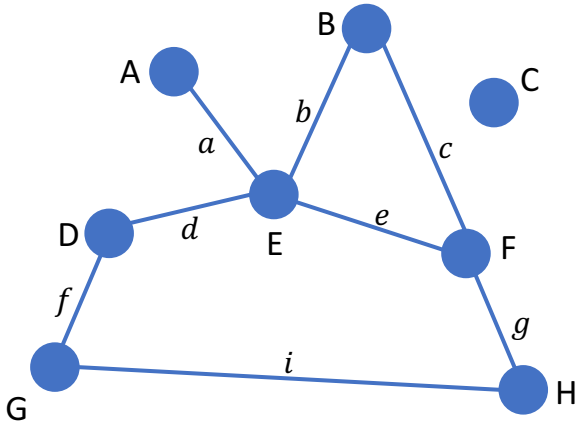


Figure 2.2: An illustration of the simple graph.

2.1.3 Knowledge Graph

The knowledge graph is a simple graph representing knowledge, in which the vertices represent entities and the edges represent relations between entities. The term knowledge graph is loosely used by researchers from various research backgrounds without a unified definition. We try to give a compendious definition of the knowledge graph as below without considering the characteristics of the research domain.

Definition 7 (Knowledge Graph)**.** *A knowledge graph is a set of triples* $\mathcal{KG} = \{(h, r, t)\}$ *where*

- *the triple* (h, r, t) *represents a piece of knowledge that indicates the relation between* h *and* t *is* r*,*
- $h \in \mathcal{E}$ *and* $t \in \mathcal{E}$ *represent head entity and tail entity, respectively,*
- $r \in \mathcal{R}$ *represents the relation between head entity and tail entity,*
- $\mathcal{E}$ *is a finite set of entities,*
- $\mathcal{R}$ *is a finite set of relations.*

The facts in the real world vary over time. The *temporal knowledge graph* utilizes timestamps to identify the validity of triples.

Definition 8 (Temporal Knowledge Graph)**.** *A temporal knowledge graph is a set of temporal triples* $\mathcal{TKG} = \{\langle (h, r, t), s\rangle\}$ *where*

- *the temporal triple* $\langle (h, r, t), s\rangle$ *represents a piece of temporal knowledge that indicates the relation between* h *and* t *is* r *and the timestamp* s *of the knowledge,*
- $h \in \mathcal{E}$ *and* $t \in \mathcal{E}$ *represent head entity and tail entity, respectively,*
- $r \in \mathcal{R}$ *represents the relation between head entity and tail entity,*
- $s = \tau$ *or* $(\tau_s, \tau_e) \in \mathcal{T}$ *represents the time or the time interval of the knowledge,*
- $\mathcal{E}$ *is a finite set of entities,*
- $\mathcal{R}$ *is a finite set of relations.*
- $\mathcal{T}$ *is a finite set of timestamps.*

The edges of the knowledge graph can also be assigned different weights to identify the uncertainty of different triples [75], confidence score [76], degree of relations [7], edge importance [77], called *weighted knowledge graph*.

Definition 9 (Weighted Knowledge Graph)**.** *A weighted knowledge graph is a set of triples* $\mathcal{WKG} = \{\langle (h, r, t), w\rangle\}$ *where*

- *the weighted triple* $\langle (h, r, t), w\rangle$ *represents a piece of weighted knowledge that indicates the relation between* h *and* t *is* r *and the weight of the knowledge,*
- $h \in \mathcal{E}$ *and* $t \in \mathcal{E}$ *represent head entity and tail entity, respectively,*

- $r \in \mathcal{R}$ *represents the relation between head entity and tail entity,*
- $w \in R_{\geq 0}$ *represents the weight of the knowledge,*
- $\mathcal{E}$ *is a finite set of entities,*
- $\mathcal{R}$ *is a finite set of relations.*

Note that the knowledge graph mentioned in this thesis without additional explanation is knowledge graphs including vanilla knowledge graphs, temporal knowledge graphs, and weighted knowledge graphs. We refer to a vanilla knowledge graph as a *deterministic knowledge graph* to distinguish it from temporal knowledge graphs and weighted knowledge graphs.

2.2 Knowledge Graph Representation Learning

2.2.1 Knowledge Representation

Knowledge Representation [78, 79] can be defined by Definition 10:

Definition 10 (Knowledge Representation)**.** *Knowledge representation is to represent knowledge in a formal approach that can be utilized by the computer.*

There are four main knowledge representation approaches: logical representation [80, 81], frame representation [82], and production rules [83], knowledge graphs.

The logical representation is to represent knowledge using logic, such as first-order predicate logic, description logic, modal logic, and non-monotonic logic.

A frame is an information structure with a frame name and a number of slots. The slot is the holder of information concerning a particular item called the slot filler. The slot fillers map the objects in the domain of discourse. A frame expresses a concept involving the various entities that appear as slot-fillers of its slots.

The production rules are also known as situation-action rules, widely used in expert systems. The rules are the IF-THEN-ACTION rules: if the clause is true, then the action will be performed. Production rules are clear, simple, and easy to incorporate additional knowledge, modify knowledge, and delete knowledge as the rules are independent. But as the knowledge base grows, it becomes difficult to keep track of the rules. It is hard to handle knowledge about cause and effect using production rules.

2.2.2 Knowledge Graph Representation Learning

Representation learning replaces manual feature engineering and allows a machine to both learn the features and use them to perform a specific task. In machine learning, representation learning [84] can be defined by Definition 11.

Definition 11 (Representation Learning)**.** *Representation learning is to automatically discover the necessary representations from raw data for a specific task.*

Knowledge graph representation learning is loosely used by a mount of knowledge graph embedding models [85, 86, 87, 54, 51] to represent representation learning on knowledge graphs.

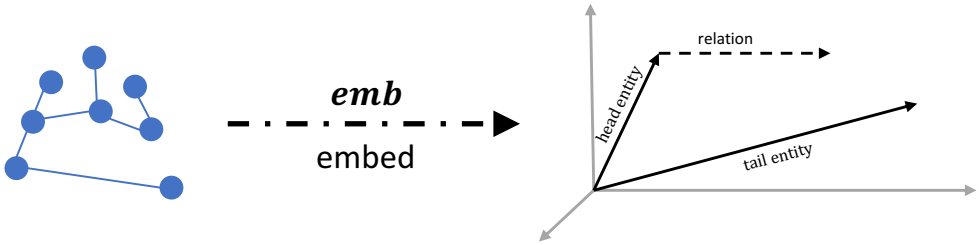
Knowledge graph embedding is to represent a knowledge graph using a set of vectors in low-dimensional space. This set of vectors, called knowledge graph embeddings, should contain the essential structural information of the knowledge graph. We give a compendious definition in Definition 12.

Definition 12 (Knowledge Graph Embedding)**.** *Given a knowledge graph $\mathcal{KG}$, knowledge graph embedding is to find a set of embeddings $\mathcal{V} = (\mathcal{V}_e, \mathcal{V}_r)$, a function $\boldsymbol{emb}$, and a function $\boldsymbol{emb}^{-1}$ that satisfy $\mathcal{V} = \boldsymbol{emb}(\mathcal{KG})$ and $\mathcal{KG} = \boldsymbol{emb}^{-1}(\mathcal{V})$ where*

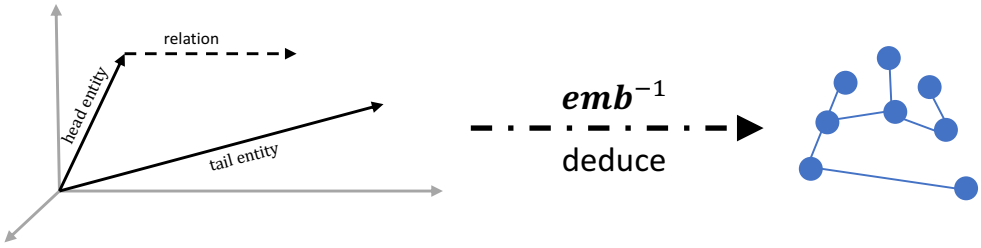
- $\boldsymbol{emb}$ *is the embedding function which can be used to embed a knowledge graph to get the embeddings,*
- $\boldsymbol{emb}^{-1}$ *is the deduction function that can deduce the knowledge graph according to its embeddings,*
- $\mathcal{V}$ *is the vectors and matrices used for representing the discrete items in $\mathcal{KG}$, where $\mathcal{V}_e \in \mathcal{R}^n$ is vectors for representing entities and $\mathcal{V}_r \in \mathcal{R}^n$ (or $\mathcal{V}_r \in \mathcal{R}^n \times \mathcal{R}^n$) is vectors (matrices) for representing relations.*

For deterministic knowledge graphs, deducing the knowledge graph is deducing the structure of the knowledge graph, i.e., deducing the triples in the knowledge graph. But for weighted knowledge graphs, deducing the weighted knowledge graph means not only deducing the triples but also deducing the weight of the triples. Figure 2.3(a) shows an illustration of learning embedding from the knowledge graph, and Figure 2.3(b) is deducing the knowledge graph from its embeddings.

The target tasks of knowledge graph embedding, such as link prediction, node classification, and triple classification, highly overlap with the target tasks of knowledge graph representation learning, requiring similar features



(a) learn embeddings from the knowledge graph



(b) deduce the knowledge graph from its embeddings

Figure 2.3: An illustration of the knowledge graph embedding.

learned from knowledge graphs. In this thesis, we regard knowledge graph representation learning and knowledge graph embedding as one task.

Knowledge representation is the representation of the knowledge stored in people's heads in a way that computers can understand. The knowledge graph is one of the results of knowledge representation. Knowledge graph representation learning is the representation of the knowledge graph in a way that deep learning models can utilize. From this perspective, the knowledge graph representation learning is a secondary representation of knowledge. Figure 2.4 shows an illustration of the relations between information, knowledge, knowledge graphs, and knowledge graph embeddings.

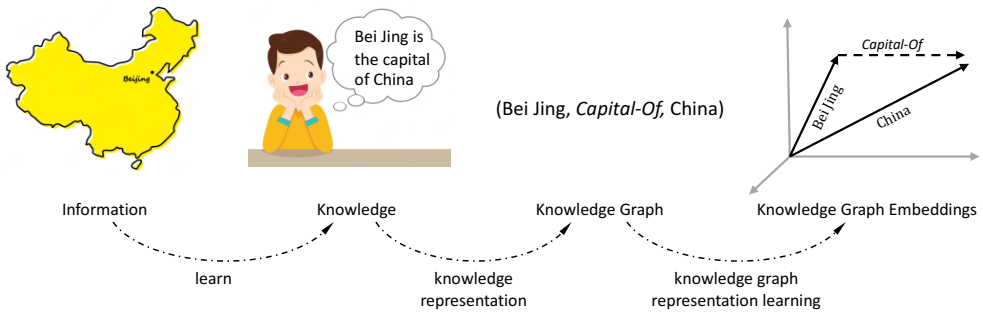


Figure 2.4: An illustration of the relations between information, knowledge, knowledge graphs, and knowledge graph embeddings.

Chapter 3

Weight-aware Knowledge Graph Embedding & Evaluation Tasks

3.1 Problem Statement

Knowledge graphs (KG) store real-world knowledge in the form of graphs, promoting the development of artificial intelligence. Many large-scale knowledge graphs, such as DBpedia [1], YAGO [88], Wikidata [89], NELL [76], and KnowledgeVault [90] have been published. With the vigorous development of KGs, they have been widely used in several real-world applications, from information retrieval [91], question answering [92, 93], to recommender systems [94, 95], and domain-specific tasks [96, 97]. Figure 3.1(b) is an illustration of a knowledge graph.

Facts encoded in knowledge graphs (KG) are mostly formalized as a set of triples (h, r, t), in which h denotes the head entity, t denotes the tail entity, and r denotes the relation between h and t. As two of the prominent tasks, link prediction (LP) and triple classification (TC) are widely adopted for evaluating the quality of knowledge graph embeddings (KGE). Informally, LP is defined as: given an incomplete triple like (h, r, ?) or (?, r, t), the task is to predict the missing entity to complete the triple [98]. In addition, TC is defined as: given an unseen triple (h, r, t), this task is to discriminate the truth value of the triple.

However, facts in the real world are not discrete items that are either true (1) or false (0). People weigh different facts and pay more attention to important facts. However, deterministic knowledge graphs treat different facts equally, which limits the expressive ability of knowledge graph embed-

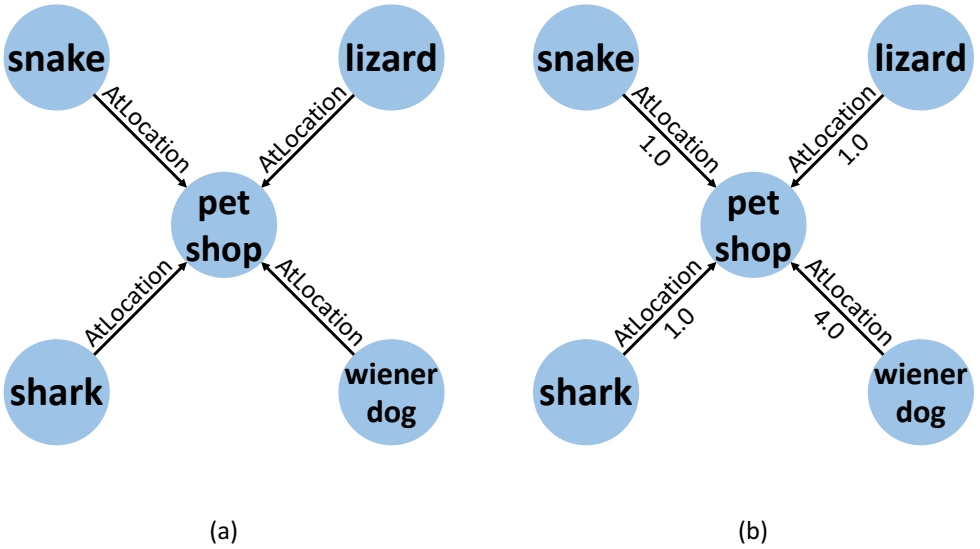


Figure 3.1: An illustration of a knowledge graph (a) and a weighted knowledge graph (b) taken from ConceptNet. The weight value in (b) indicates how believable the information is. A typical weight is 1.0, and the number is higher when the information comes from more sources or more reliable sources.

ding (KGE) models. For example, the fact `(Donald Trump, president, USA)` is much more important for learning the embedding of `"Donald Trump"` than the fact `(Donald Trump, pseudonym, John Barron)`. Few people care about a presidential pseudonym. `(Donald Trump, pseudonym, John Barron)` is, of course, a fact, but it is insignificant.

Figure 3.1 illustrates how weights collaborate with triples. Figure 3.1(a) shows a deterministic knowledge graph, which suggests that snakes, lizards, sharks, and wiener dogs are likely to appear in pet shops. However, this creates an illusion that lizards and sharks are common pets in pet stores like wiener dogs, which is apparently caused by missing weight information. In contrast, Figure 3.1(b) shows a weighted knowledge graph where the weight of the triple `(wiener dog, AtLocation, pet shop)` is significantly higher than the weight of the other three triples, indicating that wiener dogs are more likely to appear in a pet shop than the other animals. The weights are obviously useful for learning embeddings for the entities.

Weighted knowledge graphs (WKG) generalize deterministic KGs by associating a weight to each triple. Figure 3.1 (b) is an illustration of a WKG,

where the weights between two entities are statistics of the co-occurrence of these two entities in the Wikipedia page of Donald Trump. This formalism have been used to represent uncertainty [75], confidence score [76], degree of relations [7], edge importance [77], and even out-of-band knowledge [99] in a growing number of scenarios. One prominent application of weighted triples is to model the interactions between entities, such as, the interactions of proteins in STRING [99] and the co-occurrence of concepts in Probase [7]. The weighted knowledge graphs could also enhance many natural language processing models in several downstream tasks, such as inferring basic-level of categorization for knowledge-driven applications [100] and interpreting keywords using WKG for concept-based web searching [101].

Nevertheless, LP and TC do not discriminate the weights of triples and equally deal with the predicted triples with high weights and those with low weights, to the detriment of evaluation effectiveness. However, in many real-world scenarios of using weighted knowledge graphs, it is important to distinguish the triples based on their weights. For example, [102] demonstrates that protein-protein interaction networks have degree-weighted behavior, whereby the probability of interaction between two proteins is generally proportional to the product of their numbers of interacting partners or degrees. In this case, LP and TC cannot evaluate the real performance of the KG embeddings.

Twofold contributions

Firstly, to fill the aforementioned gap, this section introduces a generalized formalization of LP called the *weight-aware link prediction* task (WaLP) and a generalized formalization of TC called the *weight-aware triple classification* task (WaTC). According to their definitions (cf. Section 3.3), these two tasks can evaluate the performance of the knowledge graph embedding model that involves the weights of the triples. In WaLP and WaTC, correctly predicting a triple and classifying a triple do not equally contribute to the result but they contribute based on the weight of each triple. 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.

Secondly, to demonstrate the effectiveness of the proposed method *WaExt*, we extend four base knowledge graph embedding models, namely TransE [42], TransH [43], ComplEx [47] and DistMult [103] by generalizing the based models with weights for WaLP and WaTC. Our extensive experiments reveal that the weight-aware extension of their based model outperforms the baselines on LP, TC, WaLP, and WaTC tasks, showing that emphasizing high-weight triples can lead to better performance of knowledge graph embeddings.

3.2 Related Work

3.2.1 Non-weight-aware Knowledge Graph Embedding Model

Nonweight-aware knowledge graph embedding models [9] are designed for knowledge graphs without weights, focusing on encoding facts in knowledge graphs. According to different modeling of the interaction between entities and relations, deterministic knowledge graph embedding models can be divided into translational distance models and bilinear models.

Translational distance models

Translational distance models, such as TransE [42] and TransH [43], adopt distance-based scoring functions. Translational distance models treat the entities and relations as vectors and operations to vectors in the representation space, respectively. Distance-based scoring functions measure the plausibility based on the distance between the head entity and the tail entity that has been operated by the specific relation.

Semantic matching models

Semantic matching models, such as DistMult [103] and ComplEx [104], treat entities and relations as vectors and interactions of vectors, respectively. They adopt similarity-based scoring functions, which measure plausibility of facts based on the similarity of the head entity and the tail entity under a specific interaction.

3.2.2 Weight-aware Knowledge Graph Embedding Model

FocusE [62] 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 $l^+ \coloneqq \langle (h, r, t), w \rangle$, its corresponding negative triple is l^-, the score of a weighted triple given by FocusE layer is

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

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

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

3.2.3 Evaluation Task for Knowledge Graph Embeddings

Link prediction

Link Prediction (LP) is the task of predicting the existence of a relation between two entities or predicting the missing another entity given an entity and a relation in graph structural data. LP can be adopted to predict friend relation among users in a social network [105], predict co-author relation in a citation network [106], and predict interactions between genes and proteins in a biological network [107].

Mean rank (MR) [108], mean reciprocal rank (MRR) [109] and Hits@N [42] are widely used for evaluation. For each testing triple, the head is removed and replaced by each of the entities of the dictionary in turn. The scores of those corrupted triples are first computed by the models and then sorted by ascending order; the ranking of the correct entity is finally stored. This whole procedure is repeated while removing the tail instead of the head. MR calculates the mean of those predicted ranks, MRR calculates the mean of reciprocal of the ranks, and the Hits@N calculates the proportion of correct entities ranked in the top N.

Triple classification

The triple classification task (TC) [52] is defined as binary classification that decides whether the weight of a given triple $l \coloneqq (h, r, t)$ is strong or not. A triple l is considered as strong if its confidence score is above a specified threshold τ. The representation models need to distinguish triples in a KG from negative links and high-confidence triples from low-confidence ones. The test set consists of triples from a KG and randomly sampled negative links equally, and is divided into two groups: strong and weak/false, by their ground truth confidence scores. A testing triple l is strong if l is in the KG

and the weight of l is bigger than τ, otherwise weak/false. F1 score and accuracy are widely adopted to measure the models' performance on this task.

Tail entity prediction

This task is introduced by UKGE [60] to evaluate the performance of KGE models on WKGs. For a corrupted triple in form of $(h, r, ?)$, tail entity prediction is to predict a proper tail entity to make the corrupted triple as a true one. This task puts all the entities in the vocabulary to the corrupted triple to form the set of all possible triples. Then, let the KGE model score all the possible triples and rank all possible triples according their score. The normalized Discounted Cumulative Gain (nDCG) [110] is adopted to evaluate the performance of the KGE model. But, nDCG requires the weight of all possible triples, even for negative triples. UKGE adopts the probabilistic soft logic to generate weight for negative triples so that both positive and negative relational facts in WKG can be utilized.

3.3 Methodology

The above evaluation tasks have been originally designed for deterministic knowledge graphs, which omit to use weights on the evaluation. Weight-aware link prediction (WaLP) and weight-aware triple classification (WaTC), which emphasize triples with high weights, aim to fill this gap.

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

$$\mathcal{G} \coloneqq \left\{ \langle (h_i, r_i, t_i), w_i \rangle \right\}_{i=1}^{u} \tag{3.1}$$

where $h_i, t_i \in \mathcal{E}$, $r_i \in \mathcal{R}$ and $w_i \in R_{\geq 0}$. Both $\mathcal{E}$ and $\mathcal{R}$ are entity and relation sets, respectively. We describe WaLP and WaTC tasks as follows.

3.3.1 Weight-aware Link Prediction Task

Task description

Given a corrupted triple in the form of $(h, r, ?)$ (or $(?, r, t)$), the task is to find a proper entity $\varphi \in \mathcal{E}$ to complete the corrupted triple to a complete triple in the form of (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). The reward for completing a corrupted triple is calculated according to the weight of the complete triple.

Algorithm 1: Weight-aware Link Prediction

Input: $\mathcal{WKG} = \{\langle (h, r, t), w\rangle\}$, KGE model $\mathcal{M}$, weighting function g
Result: {WaMR, WaMRR, WaHits@N}

ranks, r_ranks, hits = 0, 0, {k:0}
while *i in range(|$\mathcal{WKG}$|)* **do**
  mix_t, mix_h, score$_t$, $score_h$ = $[(h_i, r_i, t_i)], [(h_i, r_i, t_i)], [\ ], [\ ]$
  for $j = 0;\ j < |\mathcal{E}|;\ j = j + 1$ **do**
    if $(h_i, r_i, e_j) \notin \mathcal{WKG}$ **then**
      mix_t.append((h_i, r_i, e_j))
    if $(e_j, r_i, t_i) \notin \mathcal{WKG}$ **then**
      mix_h.append((e_j, r_i, t_i))
  end
  for $k = 0;\ k < |mix_t|;\ k = k + 1$ **do**
    score$_t$.*append*($\mathcal{M}$.*score*($mix_t[k]$))
  end
  $rank_t$ = rank(score$_t$).*get*((h_i, r_i, t_i))
  ranks += $rank_t \cdot g(w_i)$
  r_ranks += $1/(rank_t \cdot g(w_i))$
  for *k in hits.keys()* **do**
    if $rank_t <= k$ **then**
      hits[k] += $g(w_i)$
  end
  for $k = 0;\ k < |mix_h|;\ k = k + 1$ **do**
    score$_h$.*append*($\mathcal{M}$.*score*($mix_h[k]$))
  end
  $rank_h$ = rank(score$_h$).*get*((h_i, r_i, t_i))
  ranks += $rank_h \cdot g(w_i)$
  r_ranks += $1/(rank_h \cdot g(w_i))$
  for *k in hits.keys()* **do**
    if $rank_h <= k$ **then**
      hits[k] += $g(w_i)$
  end
end
WaMR, WaMRR, WaHits@N = ranks/$(2 \cdot |\mathcal{WKG}|)$, r_ranks/$(2 \cdot |\mathcal{WKG}|)$, hits[N]/$(2 \cdot |\mathcal{WKG}|)$

Evaluation protocol and metrics

For each testing weighted triple $\langle (h_i, r_i, t_i), w_i\rangle$ in the testing set, w_i is omitted. The head entity h_i is removed and replaced by each of the entities in

the dictionary in turn, resulting in a set of possible triples $\{\langle(\varphi, r_i, t_i), ?\rangle\}$ where $\langle(\varphi, r_i, t_i), ?\rangle \notin \mathcal{G}$. The scores of the testing weighted triple and those possible triples are first computed by the models and then sorted in ascending order. After sorting, the ranking of the testing weighted triple rk_i is recorded. This whole procedure is repeated while removing t_i instead of h_i.

We introduce weight-aware mean rank (WaMR), weight-aware mean reciprocal rank (WaMRR), and weight-aware Hits@N (WaHits@N) to measure the performance of the models on WaLP, shown in Equations 3.2, 3.3, and 3.4, respectively. WaMR is the weighted average of the rankings of all testing triples. WaMRR is the sum of reciprocal weighted rankings. Compared with WaMR, WaMRR is less susceptible to the interference of abnormally poor rankings. WaHits@N is the weighted count of the top N triples, which are important for recommender systems. The introduced three metrics emphasize the prediction of high-weight triples more than their unweighted versions.

$$\mathrm{WaMR} = \frac{1}{\sum_{i=1}^{u} g(w_i)} \sum_{i=1}^{u} g(w_i) \cdot \mathrm{rk}_i, \tag{3.2}$$

$$\mathrm{WaMRR} = \frac{1}{\sum_{i=1}^{u} g(w_i)} \sum_{i=1}^{u} \frac{1}{g(w_i) \cdot \mathrm{rk}_i}, \tag{3.3}$$

$$\mathrm{WaHits@N} = \frac{1}{\sum_{i=1}^{u} g(w_i)} \sum_{i=1}^{u} g(w_i) \cdot I[\mathrm{rk}_i \leq N] \tag{3.4}$$

The function $g : R_{\geq 0} \to R$ with $g(w) \neq 0$ is introduced as an activation function that re-scales the weights to put more attention to more important triples. As you see, g is defined as a general function with the non-zero constraint to restrict its candidates. This condition is chosen to avoid unattended triples. We illustrate good candidates g in Subsection 3.4.3. Finally, the $I[expn]$ is the indicator function, which outputs 1 if $expn$ is true, and 0 otherwise. We present the complete evaluation procedure in Algorithm 1.

3.3.2 Weight-aware Triple Classification Task

Task description

Given a set of triples containing positive triples $(h_i, r_i, t_i) \in \mathcal{G}$ and negative triples $(h_j, r_j, t_j) \notin \mathcal{G}$, the task is to classify positive triples from negatives triples according to awareness on the weights of the triples in $\mathcal{G}$. The reward for correct classification of a triple and the punishment for wrong classification of a triple are calculated according to the weight of the triples.

Evaluation protocol and metrics

For any testing weighted triple $\langle (h_i, r_i, t_i), w_i \rangle$, w_i is omitted. The head entity h_i is removed to form the head-corrupted testing triple (φ, r_i, t_i). The head entity of the head-corrupted testing triple is replaced by k entities in the dictionary in turn, resulting in a set of head-corrupted testing triples. Mix the testing triples with the possible triples by corrupting heads, we get the mixture set for the head-corrupted testing triple:

$$\begin{aligned} mix_{head} \coloneqq & \left\{ \langle (h_j, r_j, t_j), ? \rangle \mid \langle (h_j, r_j, t_j), w_j \rangle \in \mathcal{G} \right\} \\ & \cup \left\{ \langle (\varphi, r_j, t_j), ? \rangle \mid (\varphi, r_j, t_j) \notin \mathcal{G} \right\} \end{aligned} \tag{3.5}$$

Similarly, the tail entity t_i is removed to form the tail-corrupted testing triple (h_i, r_i, φ). The tail entity of the tail-corrupted testing triple is replaced by k entities in the dictionary in turn, resulting in a set of possible triples for the tail-corrupted testing triple. Mix the testing triples with the possible triples from those of the tail-corruption, we get the mixture set for the tail-corrupted testing triple as follows:

$$\begin{aligned} mix_{tail} \coloneqq & \left\{ \langle (h_j, r_j, t_j), ? \rangle \mid \langle (h_j, r_j, t_j), w_j \rangle \in \mathcal{G} \right\} \\ & \cup \left\{ \langle (h_i, r_j, \varphi), ? \rangle \mid (h_i, r_j, \varphi) \notin \mathcal{G} \right\} \end{aligned} \tag{3.6}$$

The model is required to divide mix_{head} and mix_{tail} into a positive set and a negative set, respectively. We introduce weight-aware F1 (WaF1) score

$$\text{WaF1} = 2 * \frac{\text{wa\_prec} * \text{wa\_recall}}{\text{wa\_prec} + \text{wa\_recall}} \tag{3.7}$$

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

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

to measure the performance of the models on WaTC, where *TP* means the true positive triples, *FP* means the false positive triples, and *FN* means the false negative triples. Compared to macro F1 score, WaF1 emphasizes more on the model's judgment over correctly classified high-weight triples. We present the complete evaluation procedure in Algorithm 2.

Algorithm 2: Weight-aware Triple Classification

Input: $\mathcal{WKG} = \{\langle (h, r, t), w\rangle\}$, KGE model $\mathcal{M}$, weighting function g, number of the negative samples k

Output: {WaF1}

tp, fp, fn = 0, 0, 0 **while** *i in range(*$|\mathcal{WKG}|$*)* **do**
  mix_t, *value*=[(h_i, r_i, t_i)], [1]
  for $j = 0;\ j < k;$ **do**
    idx = random(0, $|\mathcal{E}|$)
    if $(h_i, r_i, e_{idx}) \notin \mathcal{WKG}$ *and* $(h_i, r_i, e_{idx}) \notin mix_t$ **then**
      mix_t.append((h_i, r_i, e_{idx}))
      value.append(0)
      j = j + 1
  end
  $pvt = \mathcal{M}$.predict(mix_t)
  tp += $g(w_i) \cdot (value * pvt).count(1)$
  fp += $(pvt - value).count(1)$
  fn += $g(w_i) \cdot (value - pvt).count(1)$
  mix_h, *value*=[(h_i, r_i, t_i)], [1]
  for $j = 0;\ j < k;$ **do**
    idx = random(0, $|\mathcal{E}|$)
    if $(e_{idx}, r_i, t_i) \notin \mathcal{WKG}$ *and* $(e_{idx}, r_i, t_i) \notin mix_h$ **then**
      mix_h.append((e_{idx}, r_i, t_i))
      value.append(0)
      j = j + 1
  end
  $pvh = \mathcal{M}$.predict(mix_h)
  tp += $g(w_i) \cdot (value * pvh).count(1)$
  fp += $(pvh - value).count(1)$
  fn += $g(w_i) \cdot (value - pvh).count(1)$
end
WaF1 = 2·tp/(2·tp + fp + fn)

3.3.3 Weight-aware Extensions of the Base Models

Since existing deterministic knowledge graph embedding models learn interactions of entities and relations within triples well, we propose a general method WaExt for injecting weights into the existing knowledge graph embedding models that combine the weights and their scoring function $f(h, r, t)$, as shown in Figure 3.2. The operator $\oplus$ in the figure denotes a summation operation that aggregates all the losses (Figure 3.2a) and the weighted losses

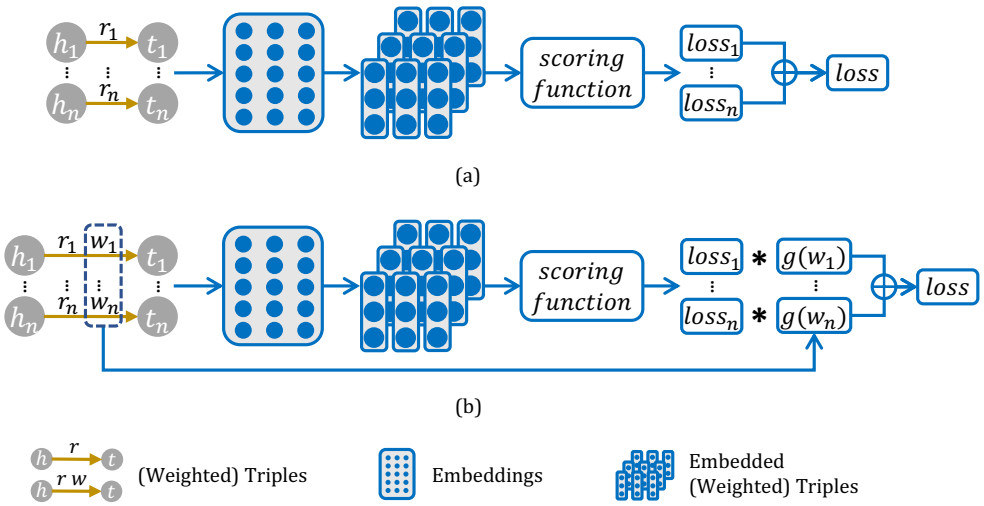


Figure 3.2: An illustration of WaExt. (a) is the process of base model, while (b) is its weight-aware extension.

(Figure 3.2b). The scoring function is adopted by the knowledge graph embedding model during the training phase to calculate the score of each triple from the training set S. In our WKG representation model, the weighted scoring function

$$f_w(h, r, t, w) := g(w) \cdot f(h, r, t) \tag{3.8}$$

calculates the score of a triple based on its weight.

We adopt the margin ranking loss [42] as the loss function for the proposed models:

$$\mathcal{L} = \sum_{\langle (h,r,t),w\rangle \in S} \sum_{\langle (h',r',t'),w'\rangle \in S'} [\gamma + f_w(h, r, t, w) - f_w(h', r, t', w')]_+ \tag{3.9}$$

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:

$$S' := \left\{ \langle (h', r, t), w' \rangle \mid h'_i \in E \setminus \{h_i\} \right\}_{i=1}^{u} \cup \left\{ \langle (h, r, t'), w' \rangle \mid t'_i \in E \setminus \{t_i\} \right\}_{i=1}^{u} \tag{3.10}$$

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

3.4 Experiment and Result

3.4.1 Experiment Setting

Consistent with prior studies[60, 62], we selected CN15K, NL27K, and PPI5K [111] as our datasets. CN15K is a subgraph of ConceptNet [75], containing 15,000 entities 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 [0.1, 1.0]. NL27K is extracted from NELL [76], a weighted KG 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]. PPI5K is a subset of the protein-protein interaction knowledge base STRING [99] that contains 255,114 weighted triples for 4,999 proteins and 7 interactions. STRING labels the interactions between proteins with the probabilities of occurrence. The weights in PPI5K fall in the interval [0.15, 1.0]. The Statistics of the datasets is shown in Table 3.1. The correlation of the weight and the triple degree in the three datasets is shown in Figure 3.3.

Table 3.1: Statistics of the datasets. #Ent denotes the number of the 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

Figure 3.4 shows the coverage of the entities and relations that appear in the test and validation sets by the triples in the training set of each dataset. 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.

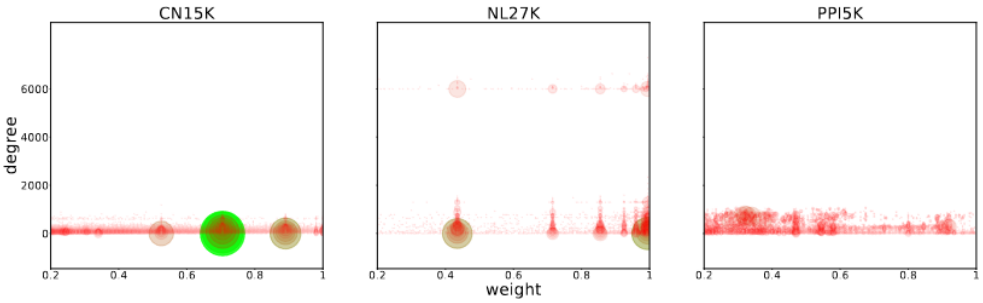


Figure 3.3: The correlation of the weight and the triple degree in CN15K, NL27K, and PPI5K. The degree of a triple is the average of the degree of the head entity and the degree of the tail entity. The weights of the triples lay in [0,1], and the degrees lay in [0, 7300]. We divide the intervals of weight and degree into 200 subintervals, respectively. Count the number of triples falling in this interval, and record it as $num(tri)$. The center of the circle represents the center of weight and center of degree of the interval. The color is defined by RGB=(1-$num(tri)$/25541, $num(tri)$/25541, 0), where $num(tri)$/25541 is the normalized number of triples. The opaque and the radius of each circle represent $num(tri)$, where higher opaque and bigger radius mean more triples here. If the color of an area is greener or denser, there are more triples in the subinterval of the centers of the circles.

We implemented WaLP, WaTC, and the weight-aware extension for the base models based on the PyKEEN toolkit [112]. We selected the exponential function as our candidate activation function, and searched a static exponent base among {0.2, 0.5, 1.5, e}. We also explore a dynamic exponent base, e.g., adjusting the base in every epoch to avoid over-fitting on high-weight triples. We choose $\frac{epoch+3}{epoch+2}$ as the dynamic base, which varies from 1.5 to 1 during training. An illustration of the activation functions is shown in Figure 3.5. We selected learning rate λ for the stochastic gradient descent among {0.001, 0.01, 0.1}, and the weight of the negative triples w' among {0, 0.5, 1, $avg(w)$, $1 - avg(w)$, 2}, where $avg(w)$ is the mean of the weights of the triples in the training set. Only if the base is less than 1, the weight of the negative triples is set to 2. The margin of the loss function γ was set to 1. The number of the false triples in WaTC was set to 100. The dimension of embeddings and the number of training epochs were set to 50. We trained the models for 1000 epochs, evaluated the models per 10 epoch, and save their best results. We evaluated the proposed models and the base models on both of LP and WaLP tasks. For convenience of our implementation, we replaced $\frac{1}{\sum_{i=1}^{u} g(w_i)}$ in WaMR (Equation 3.2), WaMRR (Equation 3.3), and WaHits@N

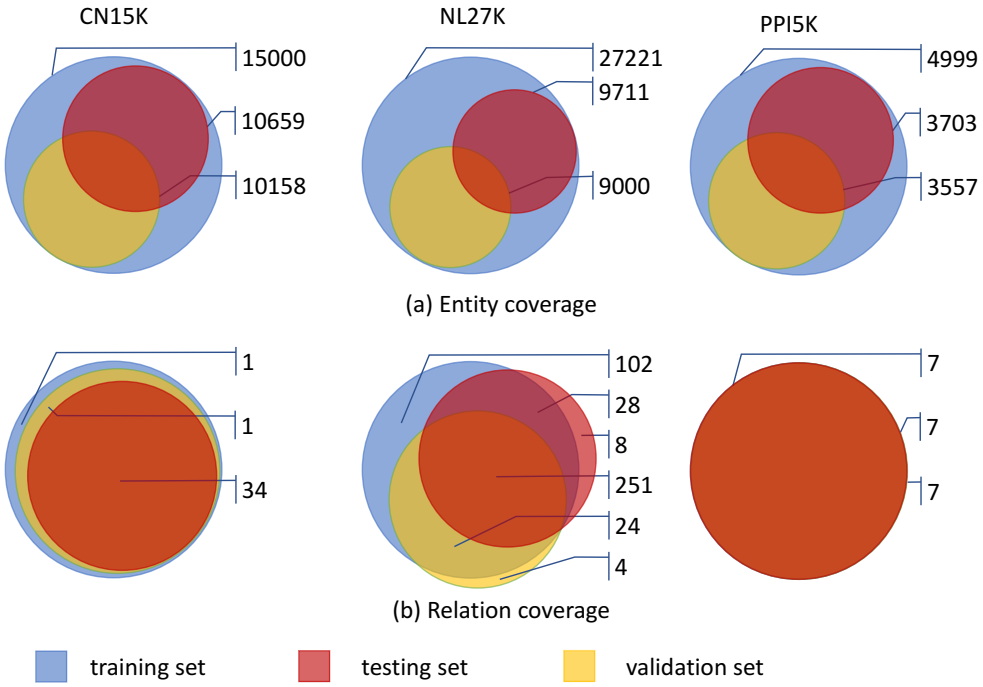


Figure 3.4: (a) Entity coverage and (b) relation coverage of the training set of the dataset. There are eight and four out-of-distribution relations in the NL27K testing set and validation set, respectively.

(Equation 3.4) with $\frac{1}{u}$.

3.4.2 Base Models

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

TransE

TransE [42] 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 of TransE is $s = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_p$.

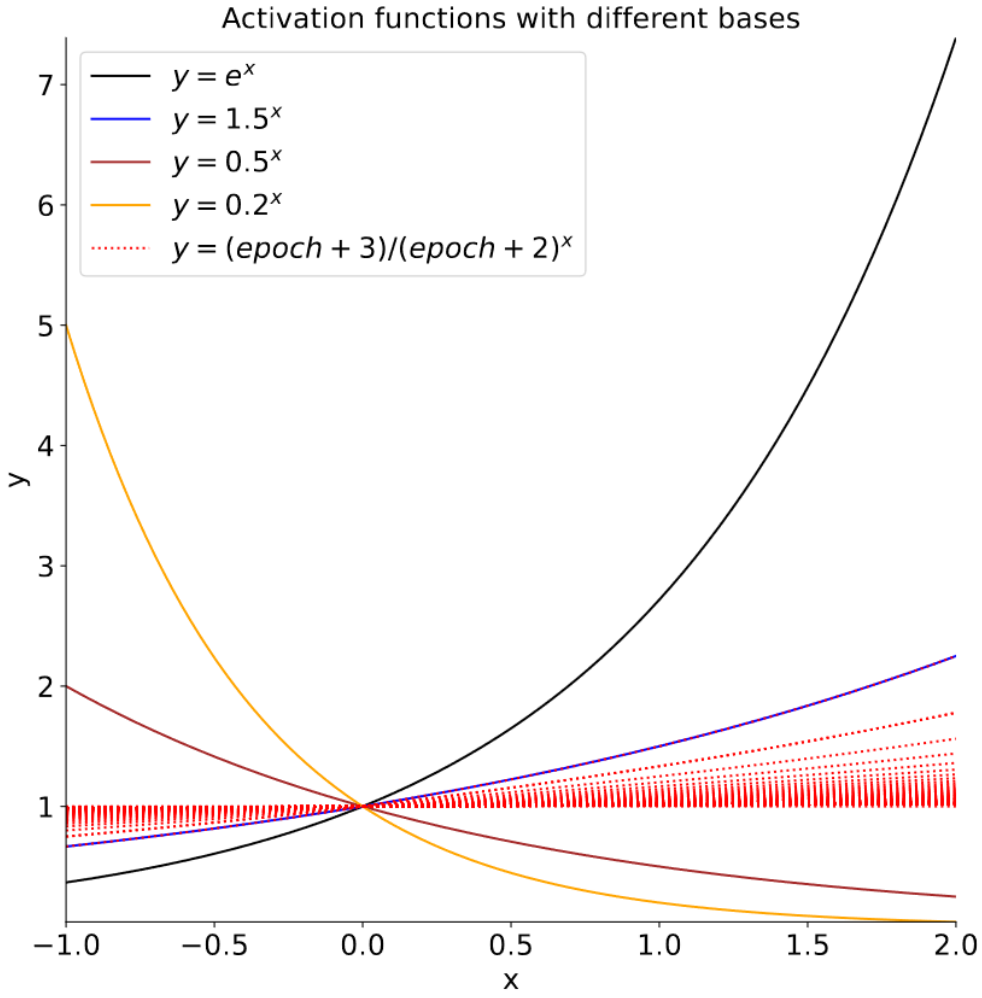


Figure 3.5: The activation functions with static base and dynamic base.

TransH

TransE is effective for 1-to-1 relations, but cannot model the 1-to-N or N-to-N relation well. TransH [43] models 1-to-N and N-to-N relations by introducing the mechanism of projecting to relation-specific hyperplanes. The scoring of TransH is $s = -\left\|\left(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r\right) + \mathbf{r} - \left(\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r\right)\right\|_2^2$, where $\mathbf{w}_r$ stands for the normal vector of a specific hyperplane.

DistMult

DistMult [103] represents each relation as a diagonal matrix that models pairwise interactions between entities to capture the latent semantics. The scoring of DistMult is $s = \mathbf{h}^\top \operatorname{diag}(\mathbf{r})\mathbf{t}$, where $\operatorname{diag}(\cdot)$ is a diagonalization function.

ComplEx

ComplEx [104] extends DistMult by introducing complex vector space to embed the knowledge graph, aiming for better modeling asymmetric relations. This scoring function of ComplEx is asymmetric, and facts with asymmetric relations can receive different scores according to the order of entities involved. The score function is defined as: $s = \operatorname{Re}\left(\mathbf{h}^\top \operatorname{diag}(\mathbf{r})\bar{\mathbf{t}}\right)$, where $\bar{\mathbf{t}}$ is the conjugate of $\mathbf{t}$ and $\operatorname{Re}(\cdot)$ means taking the real part of a complex value.

3.4.3 Result on Link Prediction and Weight-aware Link Prediction

The results of base models and their weight-aware extension versions on the link prediction and the weight-aware link prediction tasks are shown in Table 3.2 and Table 3.4, respectively. For link prediction, our proposed weight-aware extension models outperform TransE, TransH, ComplEx, and DistMult on the mean rank, mean reciprocal rank, and most of Hits@N. In particular, WaTransH has a nearly 50% improvement compared to TransH. We assume that this is because the triples with high weights may contain more useful information and less noise and the weight-aware extension models can emphasize the triples with high weights.

Note that WaComplEx achieves the best performance with the activation function $g(w) = 0.2^w$ on CN15K and NL27K, while WaTransE, WaTransH and WaDistMult achieve the best performance with $g(w) = 1.5^w$ and $g(w) = e^w$. ComplEx assigns less attention to high-weight triples, which is different from the other three models. This phenomenon might be caused by the fact that TransE, TransH, and DisMult employ the pairwise Hinge loss, while ComplEx uses the pointwise logistic loss. A further theoretical analysis is required to understand the phenomenon and we aim to achieve it in the future.

We choose FocusE as the baseline, whose result is shown in Table 3.3. FocusE's application scope is relatively limited and it is unable to consistently enhance the performance of knowledge graph representation models on link prediction tasks: it achieves better performance than base models on CN15K

but shows a decline in performance on NL27K and PPI5K. Since FocusE is implemented based on AmpliGraph [113] and WaExt is implemented using PyKEEN [112], the training strategies and details of the two libraries may differ, resulting in differences in the performance of the base models. Therefore, the performance of the extended model alone cannot directly indicate the performance difference between WaExt and FocusE. Currently, we are comparing the performance of WaExt and FocusE based on the improvement in the performance of the extended model over the base model. From this perspective, WaExt outperforms FocusE in link prediction tasks across all three datasets. To directly compare the performance of WaExt and FocusE, we will implement WaExt using AmpliGraph [113] in future work.

We show the performance curve of WaExt in Figure 3.6 and Figure 3.7. WaExt's improvement to the base model is mild and stable. We show the weight distributions of correctly predicted Hits@100 triples in Figure 3.8. WaExt is able to correctly predict more triples with high weights. We show the rank distributions of all triples in Figure 3.9. WaExt is able to make all positive triples get higher ranking.

A comparison of the correctly predicted Hits@100 triple distributions for WaTransE and TransE on CN15K, NL27K, and PPI5K is shown in Figure 3.8. We find that WaTransE outperforms TransE in predicting triples with high weights on all three datasets, which makes WaTransE achieve better performance on the WLP task than TransE.

3.4.4 Result on Triple Classification and Weight-aware Triple Classification

The results of base models and their weight-aware extensions on the TC and the WaTC task are shown in Table 3.5. From the table, it is clear that our proposed weight-aware extensions outperform TransE, TransH, ComplEx, and DistMult on both TC and WaTC tasks over the three datasets.

The results suggest that assigning higher weights to significant triples during the training phase improves the quality of embeddings learned from the knowledge graph. However, it is important to note that although WaComplEx$_{sta}$ performs better on the weighted triple classification task, it shows a decline in performance on the triple classification task. This is attributed to the fact that the base of the effective weighting function used for WaComplEx$_{sta}$ is less than 1. We will further investigate the mechanism for the weighting function selection in future work.

Table 3.2: Results of base models and their weight-aware extension models on the link prediction task. "sta." means the extended models with static base. "dyn." means the extended models with dynamic base.

	Model		MR	MRR	Hits@1	Hits@3	Hits@5	Hits@10
CN15K	TransE		1191.0	0.1048	**0.0370**	0.1273	0.1719	0.2381
	WaTransE	sta.	**1076.9**	**0.1130**	0.0370	**0.1419**	**0.1928**	**0.2604**
		dyn.	1178.2	0.1053	0.0370	0.1276	0.1740	0.2384
	TransH		1734.5	0.0777	0.0414	0.0869	0.1073	0.1422
	WaTransH	sta.	**1036.6**	**0.1020**	**0.0423**	**0.1210**	**0.1589**	**0.2123**
		dyn.	1933.0	0.0781	0.0404	0.0906	0.1124	0.1419
	ComplEx		1923.3	0.1239	0.0687	0.1386	0.1762	0.2349
	WaComplEx	sta.	**1774.4**	**0.1331**	**0.0746**	**0.1506**	**0.1928**	**0.2503**
		dyn.	1840.2	0.1223	0.0655	0.1380	0.1782	0.2354
	DistMult		991.2	0.1049	0.0399	0.1261	0.1678	0.2263
	WaDistMult	sta.	**966.6**	**0.1064**	0.0381	**0.1294**	**0.1710**	**0.2340**
		dyn.	989.4	0.1049	**0.0401**	0.1261	0.1670	0.2251
NL27K	TransE		132.1	0.3336	**0.2100**	0.3955	0.4624	0.5548
	WaTransE	sta.	**118.3**	**0.3376**	0.2023	**0.4109**	**0.4825**	**0.5804**
		dyn.	121.6	0.3351	0.2095	0.4004	0.4688	0.5618
	TransH		811.4	0.2698	0.1896	0.3059	0.3505	0.4108
	WaTransH	sta.	**337.4**	**0.2997**	**0.1986**	**0.3482**	**0.4044**	**0.4761**
		dyn.	609.8	0.2720	0.1874	0.3112	0.3547	0.4187
	ComplEx		**203.2**	0.6453	0.5391	0.7096	0.7786	0.8477
	WaComplEx	sta.	266.2	**0.6852**	**0.5970**	**0.7368**	**0.7932**	**0.8560**
		dyn.	222.5	0.6401	0.5333	0.7066	0.7705	0.8435
	DistMult		190.2	0.4092	0.3208	0.4494	0.4992	0.5731
	WaDistMult	sta.	**159.1**	**0.4343**	**0.3435**	**0.4783**	**0.5292**	**0.5968**
		dyn.	182.1	0.4158	0.3306	0.4497	0.5023	0.5773
PP15K	TransE		25.0	0.1467	0	0.1852	0.2836	0.4363
	WaTransE	sta.	31.6	**0.1513**	**0.0001**	**0.2014**	**0.2961**	0.4372
		dyn.	**24.9**	0.1493	0	0.1906	0.2908	**0.4440**
	TransH		49.7	0.1104	**0.0030**	0.1278	0.1933	0.3132
	WaTransH	sta.	**32.0**	**0.1339**	0	**0.1700**	**0.2551**	**0.3855**
		dyn.	50.4	0.1113	0.0026	0.1295	0.1951	0.3174
	ComplEx		**7.4**	0.9273	0.8785	0.9759	0.9862	**0.9915**
	WaComplEx	sta.	9.8	**0.9491**	**0.9180**	**0.9792**	**0.9877**	0.9909
		dyn.	7.7	0.9263	0.8771	0.9746	0.9858	0.9913
	DistMult		25.7	0.4518	0.3409	0.4997	0.5643	0.6550
	WaDistMult	sta.	**22.8**	**0.4682**	**0.3621**	0.5080	0.5793	**0.6845**
		dyn.	30.1	0.4678	0.3583	**0.5201**	**0.5827**	0.6718

Table 3.3: 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

3.5 Summary

In this section, we originally explore the weight-aware link prediction task and propose three evaluation metrics for weight-aware link prediction (Section 3.3.1). We also originally explore the weight-aware triple classification task and propose weight-aware F1 score as the three evaluation metrics (Section 3.3.2). 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 3.3.3).

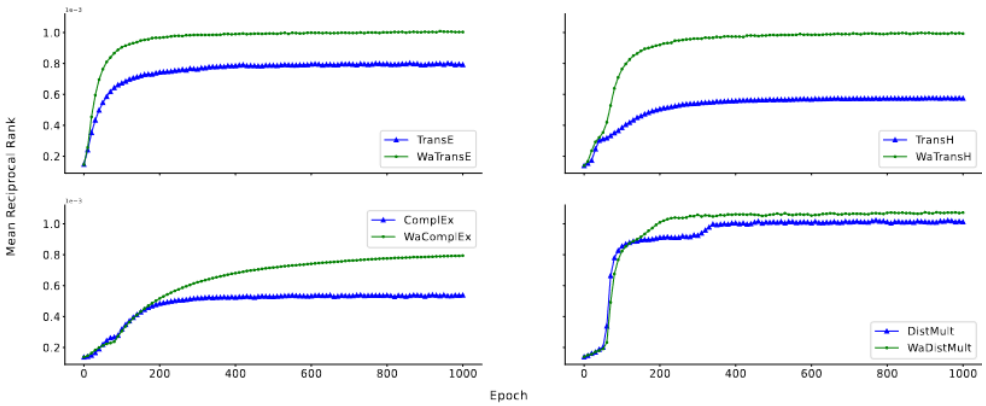
The weight-aware tasks emphasize the ability of knowledge graph embedding models to correctly predict and classify triples according to the weights of the triples, which is critical for applications in some scenarios that involve non-deterministic knowledge, such as text understanding and protein-protein 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 both link prediction, triple classification and the weight-aware tasks. Our extensive experiments reveal that the exponential activation function is effective for WaExt. We will explore more suitable activation functions in

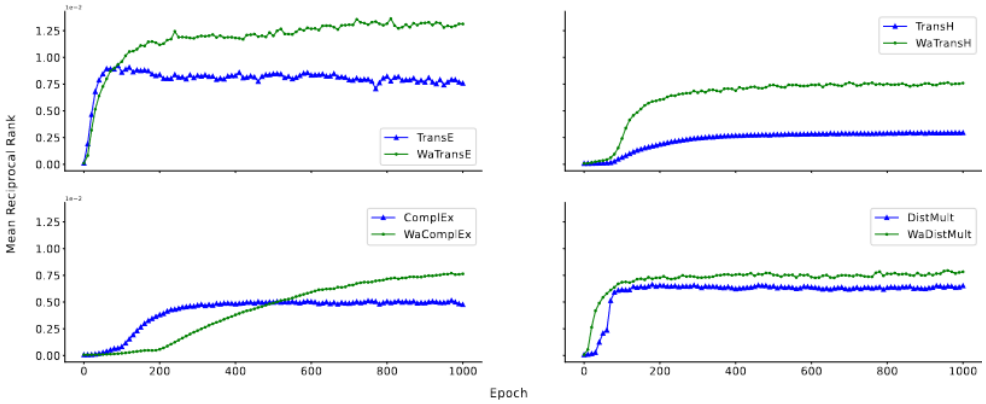
Table 3.4: Results of base models and their weight-aware extension models on the weight-aware link prediction task. "sta." means the extended models with static base. "dyn." means the extended models with dynamic base.

	Model		WaMR	WaMRR	WaH@1	WaH@3	WaH@5	WaH@10
CN15K	TransE		669.2	0.2064	0.0817	0.1861	0.2353	0.3015
	WaTransE	sta.	**657.7**	**0.2250**	**0.0928**	**0.2063**	**0.2546**	**0.3176**
		dyn.	661.7	0.2075	0.0834	0.1880	0.2351	0.2991
	TransH		969.9	0.1505	0.0655	0.1141	0.1402	0.1790
	WaTransH	sta.	**605.5**	**0.2030**	**0.0859**	**0.1699**	**0.2103**	**0.2694**
		dyn.	1074.5	0.1517	0.0680	0.1176	0.1396	0.1770
	ComplEx		1083.7	0.2470	0.1025	0.1872	0.2296	0.2900
	WaComplEx	sta.	**994.4**	**0.2688**	**0.1117**	**0.2042**	**0.2459**	**0.3038**
		dyn.	1117.3	0.2439	0.1015	0.1804	0.2194	0.2732
	DistMult		558.2	0.2067	0.0833	0.1777	0.2217	0.2837
	WaDistMult	sta.	**542.3**	**0.2086**	0.0831	**0.1826**	**0.2291**	**0.2965**
		dyn.	554.6	0.2069	**0.0838**	0.1781	0.2208	0.2827
NL27K	TransE		61.6	0.7967	0.3060	0.4973	0.5647	**0.6649**
	WaTransE	sta.	103.8	0.7854	0.3029	0.4928	0.5544	0.6420
		dyn.	**60.2**	**0.7973**	**0.3075**	**0.5026**	**0.5695**	0.6648
	TransH		407.1	0.6508	0.2524	0.3736	0.4189	0.4905
	WaTransH	sta.	**77.9**	**0.7312**	**0.2787**	**0.4481**	**0.5134**	**0.5999**
		dyn.	306.6	0.6554	0.2557	0.3781	0.4282	0.5004
	ComplEx		**101.9**	1.4837	0.6221	0.8009	**0.8508**	**0.9042**
	WaComplEx	sta.	135.7	**1.6184**	**0.6643**	**0.8052**	0.8498	0.9002
		dyn.	105.8	1.4689	0.6142	0.8005	0.8508	0.9028
	DistMult		92.1	0.9583	0.3845	0.5255	0.5798	0.6604
	WaDistMult	sta.	**73.7**	**0.9913**	**0.4116**	**0.5509**	**0.6039**	**0.6835**
		dyn.	88.6	0.9735	0.3894	0.5290	0.5824	0.6661
PPI5K	TransE		18.0	0.2568	0.0586	0.2496	0.3558	0.5136
	WaTransE	sta.	26.2	0.2582	**0.0724**	0.2530	0.3469	0.4848
		dyn.	**17.9**	**0.2613**	0.0605	**0.2560**	**0.3630**	**0.5197**
	TransH		35.9	0.1946	0.0428	0.1788	0.2554	0.3853
	WaTransH	sta.	**23.8**	**0.2434**	**0.0667**	**0.2324**	**0.3237**	**0.4556**
		dyn.	36.4	0.1962	0.0427	0.1809	0.2596	0.3905
	ComplEx		**4.7**	1.4287	0.8976	0.9825	0.9895	**0.9928**
	WaComplEx	sta.	6.3	**1.4592**	**0.9348**	**0.9853**	**0.9897**	0.9918
		dyn.	4.9	1.4276	0.8953	0.9814	0.9888	0.9928
	DistMult		18.2	0.6888	0.3675	0.5460	0.6157	0.7138
	WaDistMult	sta.	**15.8**	0.6863	0.3771	0.5641	**0.6441**	**0.7485**
		dyn.	21.3	**0.7145**	**0.3812**	**0.5650**	0.6304	0.7292

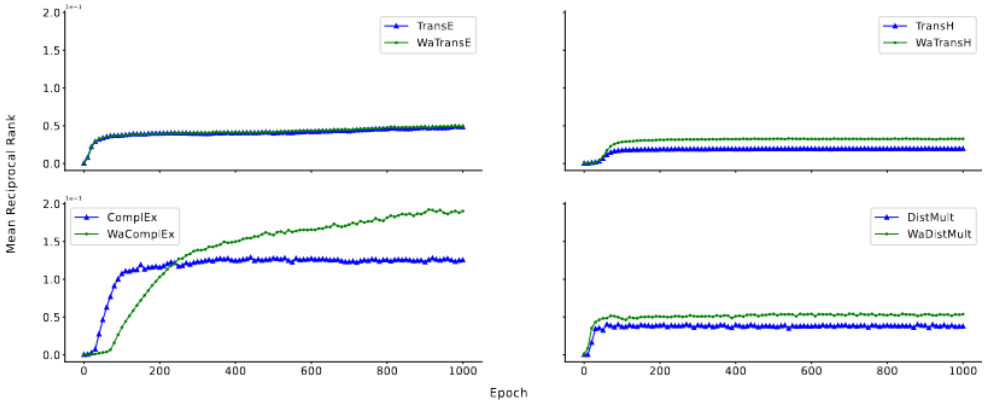
the future.



(a) CN15K

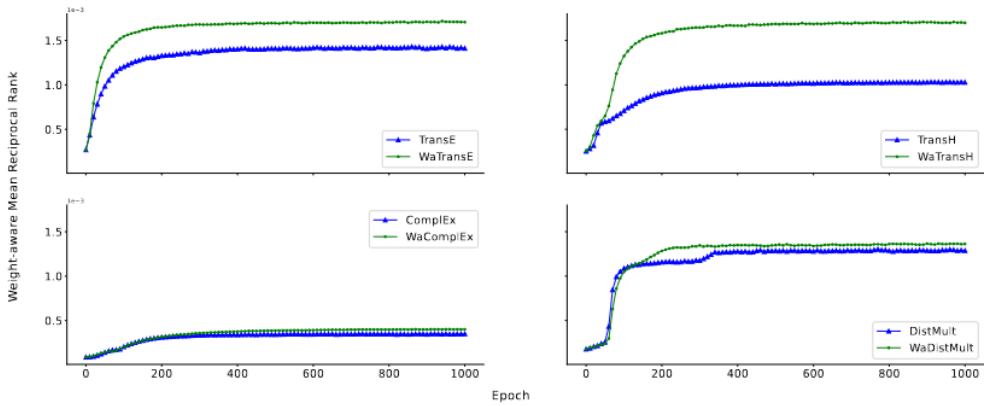


(b) NL27K

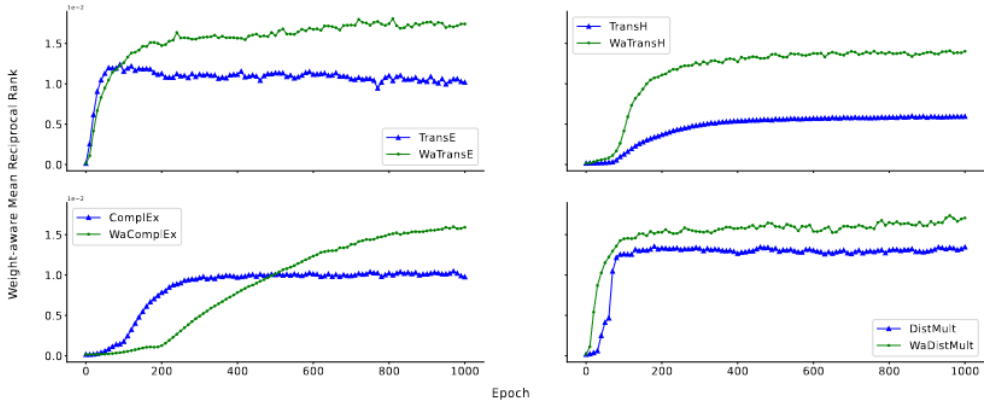


(c) PPI5K

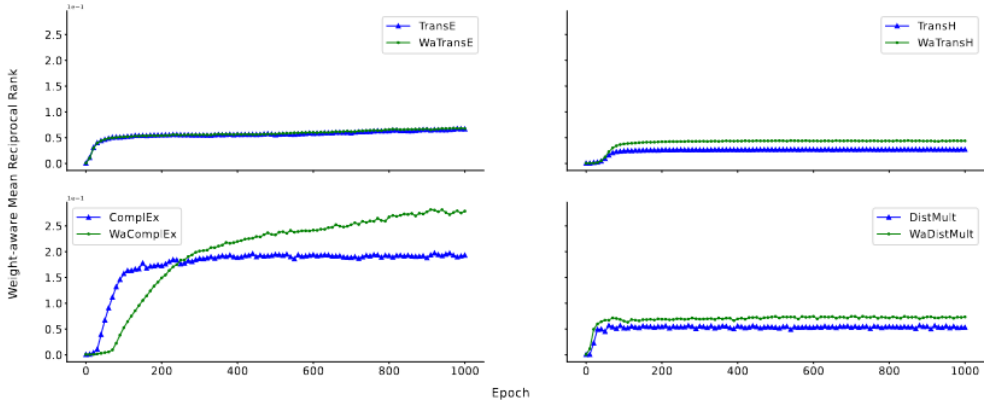
Figure 3.6: Performance of link prediction (MRR) on CN15K, NL27K and PPI5K.



(a) CN15K

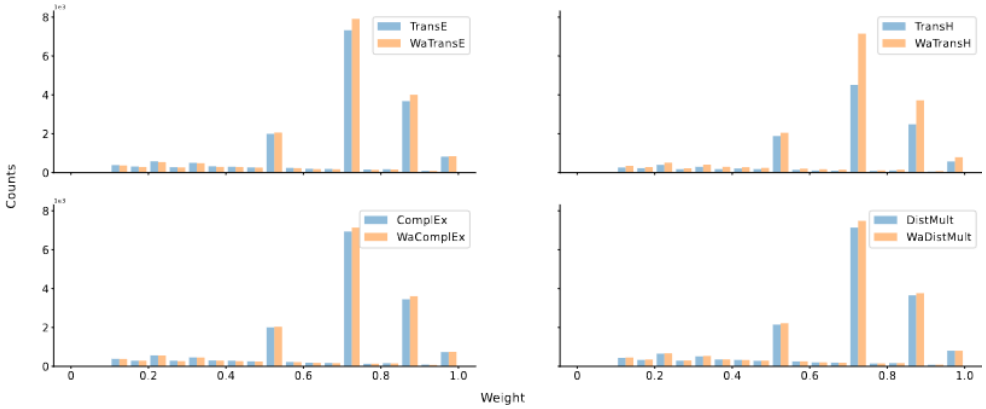


(b) NL27K

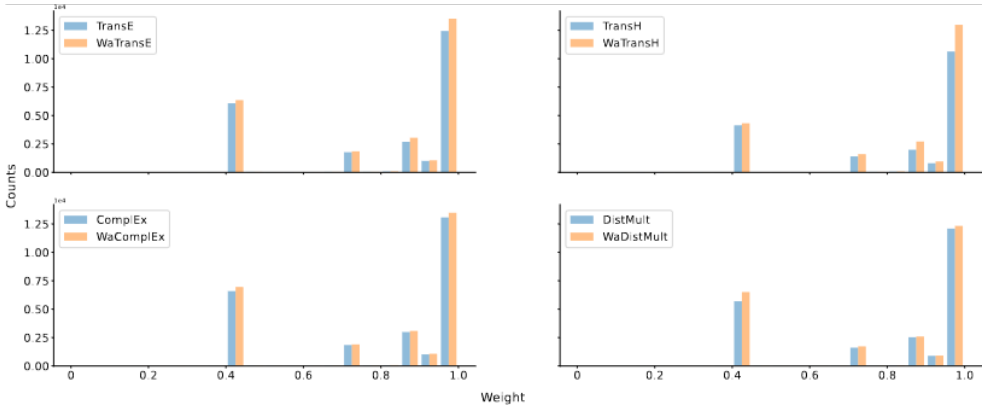


(c) PPI5K

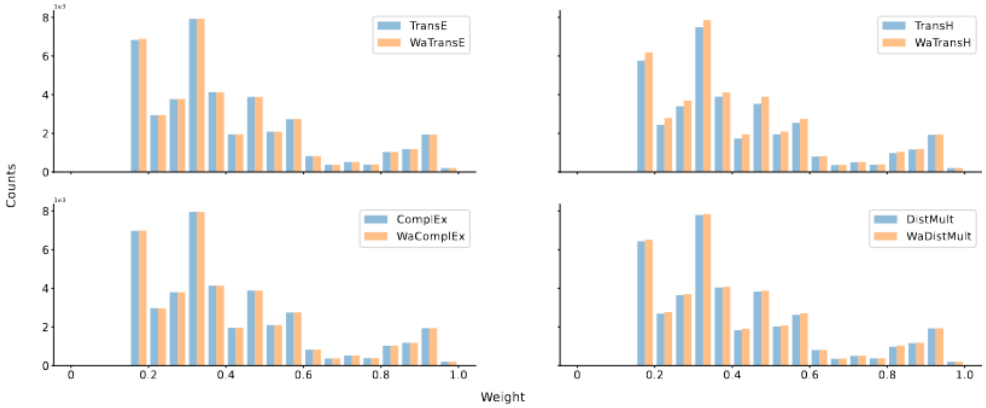
Figure 3.7: Performance of weight-aware link prediction (WaMRR) on CN15K, NL27K and PPI5K.



(a) CN15K

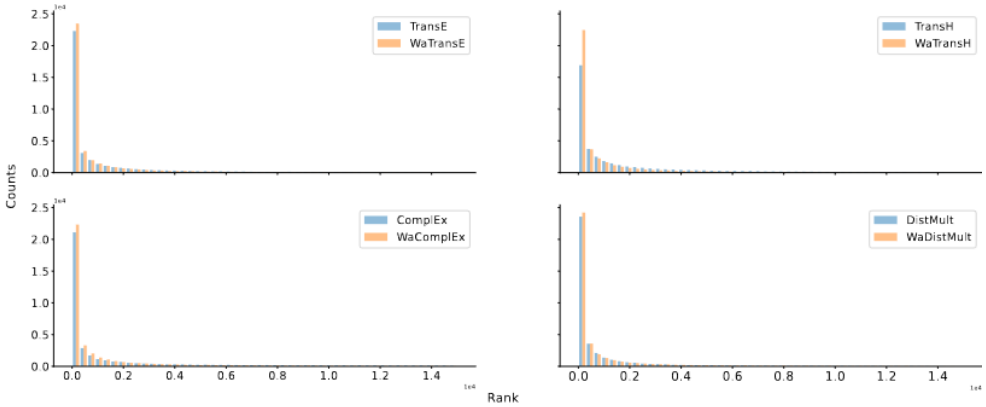


(b) NL27K

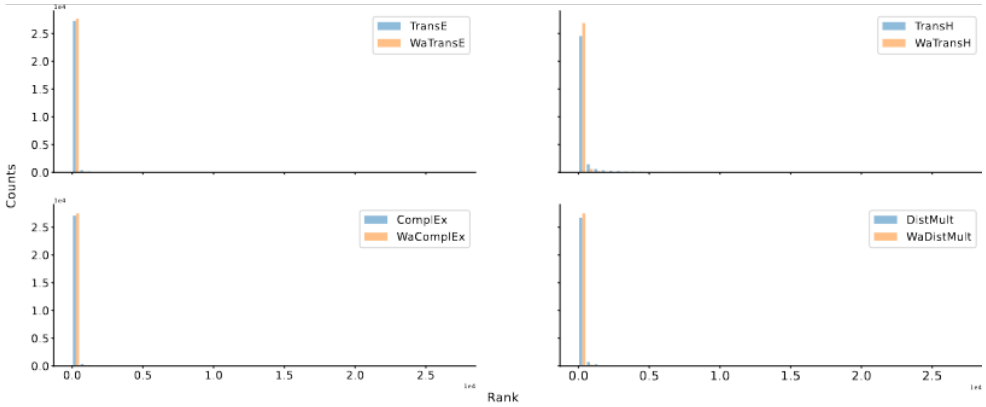


(c) PPI5K

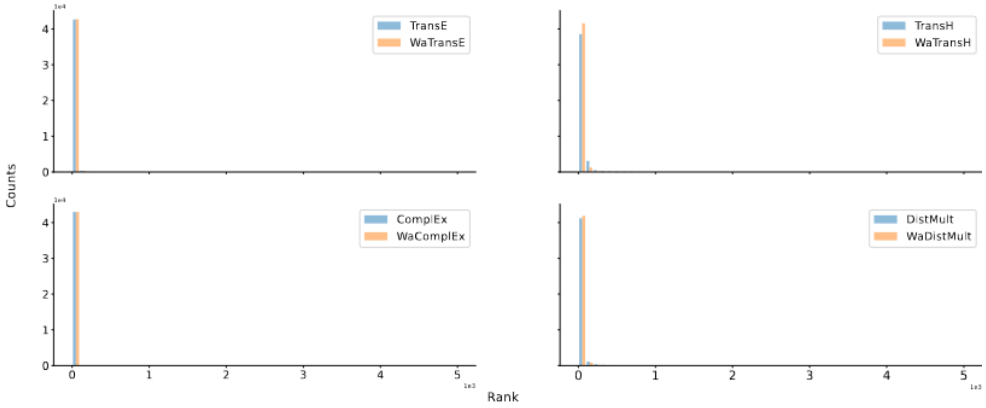
Figure 3.8: Distributions of correctly predicted Hits@100 triples on CN15K, NL27K and PPI5K.



(a) CN15K



(b) NL27K



(c) PPI5K

Figure 3.9: Rank distributions of all testing triples in CN15K, NL27K and PPI5K.

Table 3.5: Results of base models and their weight-aware extension models on the triple classification task and weight-aware triple classification task. "sta." means the extended models with static base. "dyn." means the extended models with dynamic base.

	Model		F1	WaF1
CN15K	TransE		0.255421	0.471335
	WaTransE	sta.	**0.318690**	0.456810
		dyn.	**0.257189**	**0.473832**
	TransH		0.093696	0.171650
	WaTransH	sta.	**0.150479**	**0.290128**
		dyn.	**0.094256**	**0.172605**
	ComplEx		0.434951	0.797724
	WaComplEx	sta.	**0.437375**	0.743155
		dyn.	**0.437654**	**0.801842**
	DistMult		0.164980	0.301339
	WaDistMult	sta.	**0.202502**	**0.353637**
		dyn.	**0.165068**	0.300674
NL27	TransE		0.348590	0.816554
	WaTransE	sta.	**0.365673**	**0.849980**
		dyn.	**0.348676**	0.816331
	TransH		0.144636	0.333883
	WaTransH	sta.	**0.289826**	**0.702594**
		dyn.	0.142370	0.329619
	ComplEx		0.374295	0.863087
	WaComplEx	sta.	**0.570674**	0.466630
		dyn.	**0.380700**	**0.884543**
	DistMult		0.187338	0.425659
	WaDistMult	sta.	**0.260377**	**0.554761**
		dyn.	**0.203572**	**0.464910**
PPI5K	TransE		0.602971	0.973062
	WaTransE	sta.	**0.607554**	0.940403
		dyn.	**0.635340**	**1.019078**
	TransH		0.487225	0.754997
	WaTransH	sta.	**0.566378**	**0.940029**
		dyn.	**0.487783**	**0.757455**
	ComplEx		0.976221	1.519832
	WaComplEx	sta.	**0.978150**	1.515479
		dyn.	0.976357	**1.520123**
	DistMult		0.694333	1.099331
	WaDistMult	sta.	**0.701792**	**1.114662**
		dyn.	**0.695856**	**1.100713**

Chapter 4

Weighted Knowledge Graph Embedding

4.1 Problem Statement

Knowledge graphs (KG) are thriving and promoting many downstream tasks, such as academic search [114], social relationship recognition [115], and drug discovery [116]. Facts encoded in KG are mostly formalized as triples (h, r, t), in which h denotes the head entity, t denotes the tail entity, and r denotes the relation between h and t. This formalism is sometimes referred to as *deterministic knowledge graph* [117, 118, 119] since triples are employed to represent facts.

Much recent attention has been paid to weighted knowledge graphs (WKG) such as Probase [7], NELL [76], ConceptNet [120], and the Protein-Protein Interaction Knowledge Base STRING [99, 121], which generalize deterministic knowledge graphs by associating a weight $w \in R$ to each triple. Facts encoded in WKG are mostly formalized as weighted triples $\langle (h, r, t), w \rangle$, though the semantics of weight can be various. For example, it has been used to represent uncertainty [75], confidence score [76], degree of relations [7], edge importance [77], and even out-of-band knowledge [99] in a growing number of scenarios. In real-world usages, it is obvious that the weighted triples model more precise knowledge. For example, while both (*Honda*, *competeswith*, *Toyota*) and (*Honda*, *competeswith*, *Chrysler*) look somewhat correct, the former fact should have a higher confidence than the latter one, since *Honda* and *Toyota* are both Japanese car manufacturers and have highly overlapping customer bases. This modelling can be done if one supposes the semantics of weights based on the confidence score and associates the former triple with a higher value.

As for basic elements in deterministic knowledge graphs, entities and relations are discrete symbols, which are not easy to be utilized by machine learning and deep learning models. To address this issue, knowledge graph embedding (KGE) [9] has been investigated to represent the discrete symbols in knowledge graphs as a set of vectors in a specific low-dimensional vector space, and it requires that the representation should enable to deduce the knowledge graphs from this set of vectors. Link prediction (LP) is widely adopted as a task to evaluate the performance of embedding in deducing the structure of any KG. An illustration of knowledge graph embedding is shown in Figure 4.1.

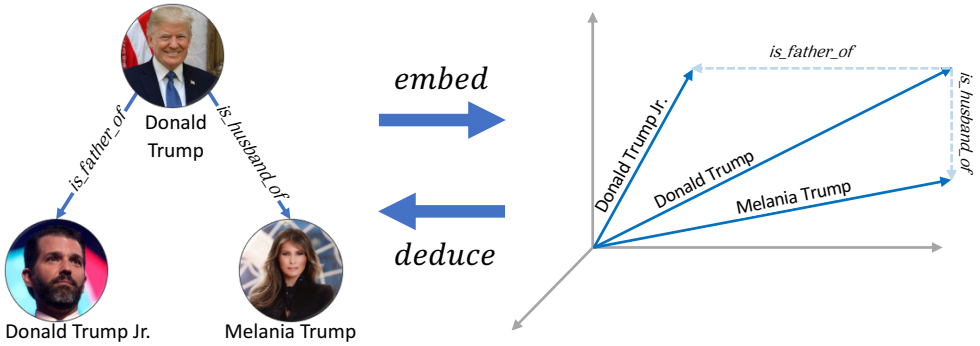


Figure 4.1: Knowledge graph embedding

While KGE algorithms focus on representing deterministic knowledge graphs, they cannot work well when the semantics of triples are imposed by weights. This problem leads to an extended study on weighted knowledge graph embedding (WKGE) aimed to embed entities and relations in a WKG into a set of vectors in a specific low-dimensional vector space. The embeddings of WKG are required to be able to not only deduce the triples in a WKG but also deduce the weight of the triples, which requires the embeddings of entities and relations have encoded the weight information.

To determine the embedding of weighted triples in WKG, some WKGE algorithms [60, 61] have been proposed to decompose this main task into two sub-tasks, namely link prediction, and weight prediction. An illustration of this decomposition is shown in Figure 4.2. However, we observe that this embedding scheme does not achieve the best performance on the link prediction task and on the weight prediction task synchronously. An illustration of the performance of UKGE [60] in link prediction and weight prediction on NL27K is shown in Figure 4.3.

To encode weight information in knowledge graph embeddings, UKGE [60] and PASSLEAF [61] adopt a non-linear function to convert the triple plausi-

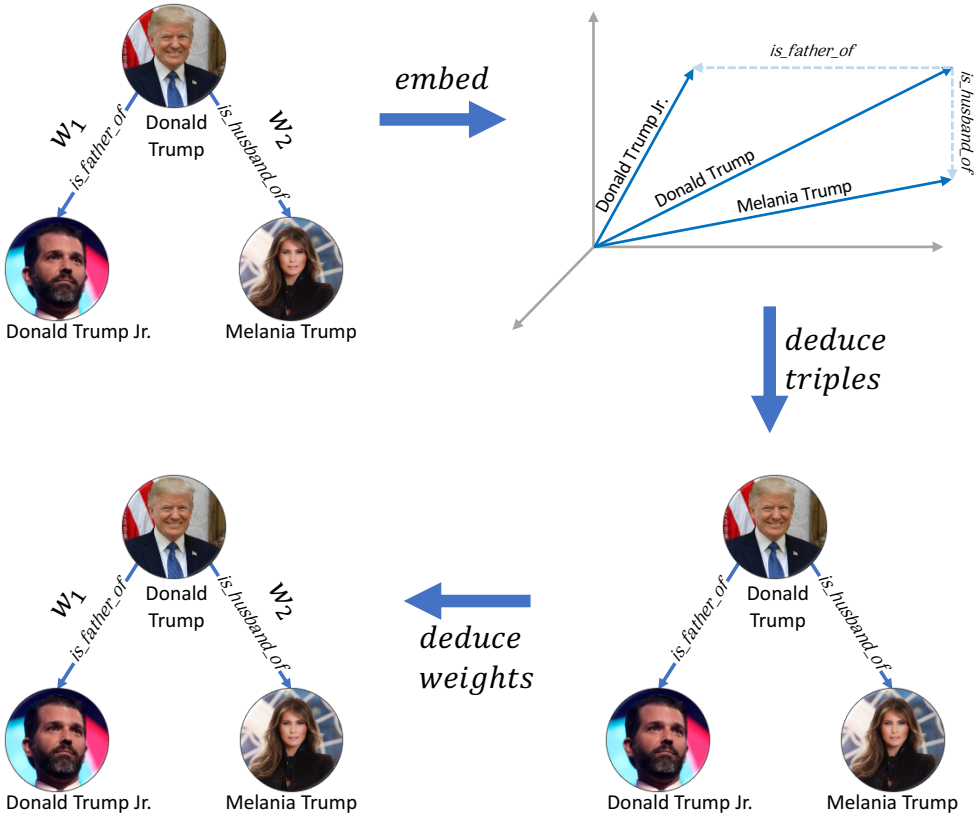


Figure 4.2: Knowledge graph embedding

bility score to the weight of the triple, equating the plausibility of the triple and the weight of the triple. But even though the positive triples may have been attached to different weights conveying different meanings, the plausibility of all positive triples should be the same.

To fill this gap, we introduce the WeExt framework that includes an independent weight prediction module to existing deterministic knowledge graph embedding models, enabling them to encode the weight information of the triples. The introduced weight prediction module takes the embeddings of the head entity, the relation, and the tail entity as input, which contain richer information than the plausibility of the triple. During training, we jointly optimize the model's performance in encoding facts and weights.

We also fill a gap of the WKGE model evaluation task. We design the *weighted link prediction* (WLP) task to comprehensively evaluate the model's ability in deducing weighted triples. WLP adjusts the ranking of positive triples in all possible triples according to the accuracy of weight prediction to

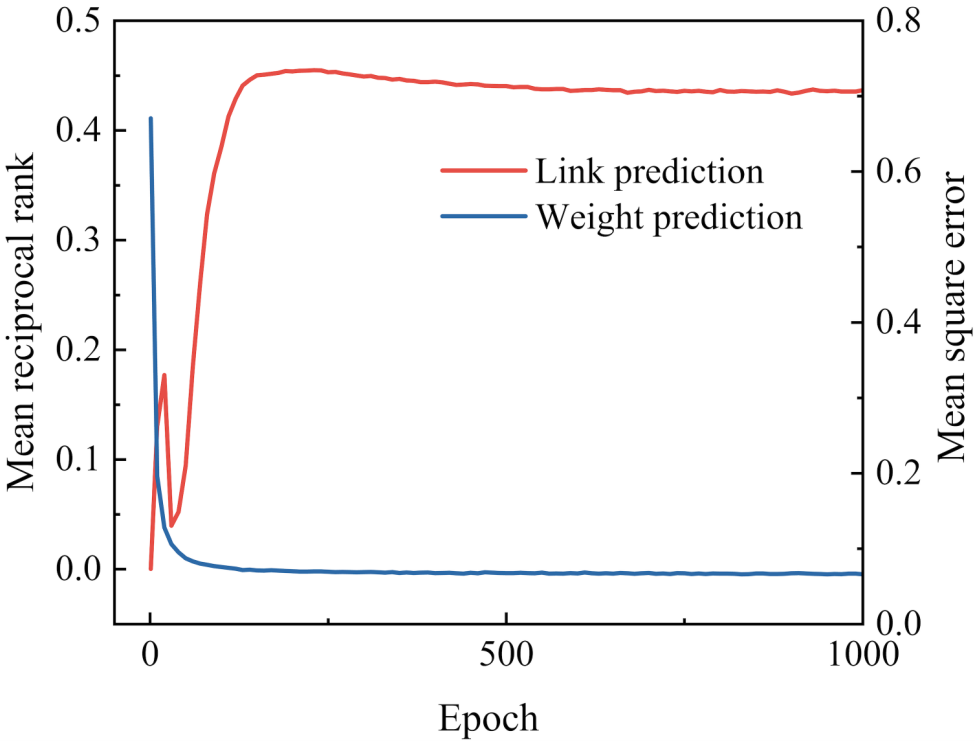


Figure 4.3: The performance of UKGE on NL27K. The red line is the mean reciprocal rank in link prediction. The blue line is the mean square error in weight prediction.

simultaneously demonstrate the performance of the model on link prediction and weight prediction.

We conduct experiments with two representative KGE translational distance models TransE, TransH, and two KGE representative bilinear models DistMult, HolE. The results show that the proposed framework WeExt achieves competitive performance over the baseline models on link prediction, weight prediction, and weighted link prediction.

4.2 Related Work

4.2.1 Deterministic Knowledge Graph Embedding Models

Deterministic knowledge graph embedding models [9] are designed for deterministic knowledge graphs, focusing on encoding facts in knowledge graphs. According to different modeling of the interaction between entities and relations, deterministic knowledge graph embedding models can be divided into translational distance models and bilinear models.

Translational Distance Models

The translational distance model, such as TransE [42] and TransH [122], regards the relation as a translation operation from the head entity to the tail entity and utilizes a distance-based scoring function to measure the plausibility of triples.

Bilinear Models

The bilinear models, such as RESCAL [45], DistMult [103], and HolE [47], are based on the tensor factorization and model the interaction of entities and relations by vector-matrix product, obtaining high expressive power due to the use of a full rank matrix for each relation in the score functions which are in the form of $h^\top W_r t$.

4.2.2 Weighted Knowledge Graph Embedding Models

UKGE

UKGE [60] is an embedding model for uncertain knowledge graphs which associate each triple to a confidence score. The model requires logical rules as additional inputs to help enforce the global consistency of predicted facts. UKGE learns the weight of a given triple by squashing the plausibility score of the triple calculated by DistMult using a non-linear function, such as

$$\phi\left(s\left(l\right)\right) = \frac{1}{1 + e^{-(w \cdot s(l) + b)}} \tag{4.1}$$

or

$$\phi\left(s\left(l\right)\right) = \min\left(\max\left(w \cdot s\left(l\right) + b, 0\right), 1\right). \tag{4.2}$$

where w is a weight, b is a bias and $s\left(l\right)$ is the score of the triple l given by DistMult. UKGE adopts mean square error (MSE) to measure the loss on

learning the weights. Given a set of positive relation facts, the loss on the positive triples is

$$\mathcal{L}_{pos} = \sum_{l \in \mathcal{L}^+} \left| \phi\left(s\left(l\right)\right) - w \right|^2 \tag{4.3}$$

UKGE estimates the weight of negative triples using probabilistic soft logic [56] and measures the loss on negative triples by the square of the distance [123]

$$\mathcal{L}_{neg} = \sum_{l \in \mathcal{L}^-} \sum_{\gamma \in \Gamma} \left| \psi_\gamma\left(\phi(s\left(l\right))\right) \right|^2 \tag{4.4}$$

where $\mathcal{L}^-$ be a set of negative relaions and Γ be a set of grounded rules. $\psi_\gamma\left(\phi(l)\right)$ denotes the distance to satisfaction of the rule γ in PSL. For any rule $\gamma \equiv \gamma_{body} \rightarrow \gamma_{head}$, the distance d_γ describing the satisfaction of the rule is

$$d_\gamma = \max\left\{0, I\left(\gamma_{body}\right) - I\left(\gamma_{head}\right)\right\} \tag{4.5}$$

where $I\left(l\right)$ is the soft truth value of the triple l

$$I\left(l\right) = \begin{cases} w, & l \text{ is positive} \\ \phi(s\left(l\right)), & l \text{ is negative} \end{cases} \tag{4.6}$$

But for negative triples not covered by the rule, the loss is

$$\mathcal{L}_{neg} = \sum_{l \in \mathcal{L}^-} \left| \phi\left(s\left(l\right)\right) - 0 \right|^2 \tag{4.7}$$

which treats the weight of triples not covered by the rules as 0. Thus, the total loss is

$$\mathcal{L} = \mathcal{L}_{pos} + \mathcal{L}_{neg} \tag{4.8}$$

UKGE only optimizes its ability on weight prediction but does not put attention to its performance on link prediction.

PASSLEAF

PASSLEAF [61] extends UKGE to be able to utilize the scoring functions of RotatE and ComplEx to calculate triple weights. PASSLEAF automatically generates weights for negative samples by the model expecting to mitigate the false-negative problem and introduces a sample pool to keep negative samples to improve training efficiency.

TransHExt

TransHExt [124] adopts a 3-layer feed-forward neural network to TransH for predicting the weight w_p of any triple (h, r, t)

$$w_p = \mathcal{N}\left(\left(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r\right) + \mathbf{r} - \left(\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r\right)\right)$$

where $\mathcal{N}$ is the 3-layer feed-forward neural network and $\mathbf{w}_r$ is the normal vector of the relation-specific hyperplane. $\mathbf{h}, \mathbf{r}, \mathbf{t}$ are corresponding vectors of the triple (h, r, t), respectively. The accuracy of weight prediction is measured by

$$acc(w_p, w) = \begin{cases} \frac{w-|w-w_p|}{w}, & w_p \in [0, 2w] \\ 0, & otherwise \end{cases}$$

TransHExt simultaneously optimizes its performance on link prediction and weight prediction through a joint loss

$$\mathcal{L} = \sum_{l \in \mathcal{L}^+} \sum_{l^- \in \mathcal{L}^-} \gamma + \left[f\left(l\right) + acc\left(w_p, w\right)\right] - f\left(l^-\right)$$

4.2.3 Evaluation Tasks for Knowledge Graph Embedding Models

Link Prediction

Link prediction (LP) is the task of predicting the existence of a relation between two entities. LP can be adopted to predict friend relation among users in a social network [105], predict co-author relation in a citation network [106], and predict interactions between genes and proteins in a biological network [107]. Mean rank (MR) [108], mean reciprocal rank (MRR) [109] and Hits@N [42] are widely used for evaluation of the models. For each test triple, the head is removed and replaced by each of the entities of the dictionary in turn. The scores of those corrupted triples are first computed by the models and then sorted by ascending order; the rank of the correct entity is finally stored. This whole procedure is repeated while removing the tail instead of the head. MR calculates the mean of those predicted ranks, MRR calculates the mean of the reciprocal of the ranks, and the Hits@N calculates the proportion of correct entities ranked in the top N.

Weight Prediction

Weight prediction task (WP) [60] is to predict weights of unseen triples. For each weighted triple $\langle (h, r, t), ? \rangle$ in the test set, the task is to predict the

missing weight w. The mean squared error (MSE) and the mean absolute error (MAE) between the predicted values and the ground truth are adopted as the evaluation metrics.

4.3 Methodology

4.3.1 WeExt

To embed the WKGs, we introduce a weight prediction module consisting of preprocessing and a neural weight predictor (nwp) to predict the weight for a given triple. The architecture of the proposed framework WeExt is shown in Figure 4.4.

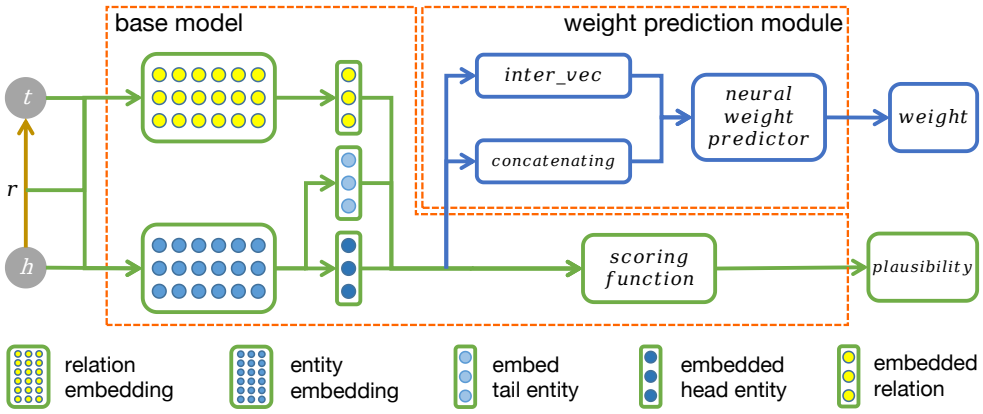


Figure 4.4: The framework of WeExt. The green components are the components of the base KGC model.

For any deterministic KGE model, a head entity, a relation, and a tail entity interact according to a preset paradigm to produce an interaction vector. This interaction can be divided into two steps, the first step is to preprocess the entities and relations, and the second step is to perform addition operation (translation distance model) or multiplication operation (two-line sex model) to get the interaction vector. The plausibility of the triple is obtained by modulo the interaction vector. UKGE computes the weight of the triple by squashing the plausibility of the triple by a non-linear function, while we argue that the interaction vector of the triple maintains richer information than the plausibility of the triple, thus it may be possible to predict the weights with higher accuracy using the interaction vector.

Based on the above assumption, we design the *inter_vec* (*ivec*) preprocessing. *ivec* processes the entities and the relations following the base model

but removes the modulo operation and outputs the interaction vectors directly.

Since translational distance models measure the distance between the head and the translated tail entity, the well-trained translational distance models produce interaction vectors close to the zero vector. To mitigate the above problem, we design another preprocessing which concatenates the processed head entity, the relation, and the tail entity after the first step of the interaction of the base model. We call this preprocessing as *concatenating* (*cat*).

We implement WeExt on the basis of four deterministic knowledge graph embedding models, including two representative translational distance models TransE and TransH, and two representative bilinear models DistMult and HolE.

Next, we illustrate how *ivec* and *cat* work through our implementation. The scoring functions of the base models are:

- TransE: $s = -\|\mathbf{h} + \mathbf{r} - \mathbf{t}\|_p$
- TransH: $s = -\left\|\left(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r\right) + \mathbf{r} - \left(\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r\right)\right\|_2^2$
- DistMult: $s = \|\mathbf{r} \circ \mathbf{h} \circ \mathbf{t}\|$
- HolE: $s = \mathbf{r}(\mathbf{h} \star \mathbf{t})$

where p is the norm, $\mathbf{w}_r$ is the relation-specific hyperplane, $\circ$ is the element-wise product, and $\star$ is the circular correlation.

The *inter_vec* preprocessing for the base models are:

- TransE: $\mathbf{p} = \mathbf{h} + \mathbf{r} - \mathbf{t}$
- TransH: $\mathbf{p} = \left(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r\right) + \mathbf{r} - \left(\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r\right)$
- DistMult: $\mathbf{p} = \mathbf{r} \circ (\mathbf{h} \circ \mathbf{t})$
- HolE: $\mathbf{p} = \mathbf{r} \circ (\mathbf{h} \star \mathbf{t})$

The *concatenating* preprocessings for the base models are:

- TransE: $\mathbf{p} = cat\,(\mathbf{h},\ \mathbf{r},\ -\mathbf{t})$
- TransH: $\mathbf{p} = cat\left(\mathbf{h} - \mathbf{w}_r^\top \mathbf{h}\mathbf{w}_r,\ \mathbf{r},\ -(\mathbf{t} - \mathbf{w}_r^\top \mathbf{t}\mathbf{w}_r)\right)$
- DistMult: $\mathbf{p} = cat\,(\mathbf{h},\ \mathbf{r},\ \mathbf{t})$
- HolE: $\mathbf{p} = cat\,(\mathbf{r},\ \mathbf{h} \star \mathbf{t})$

We implement the neural weight predictor using a four-layer feed-forward neural network. The neural weight predictor predicts the weight based on the output of the preprocessing component:

$$w_p = nwp(\mathbf{p})$$

To better illustrate the workflow of WeExt, we take TransH as an example and explain how WeExt is used to extend the base model (TransH) in Figure 4.5.

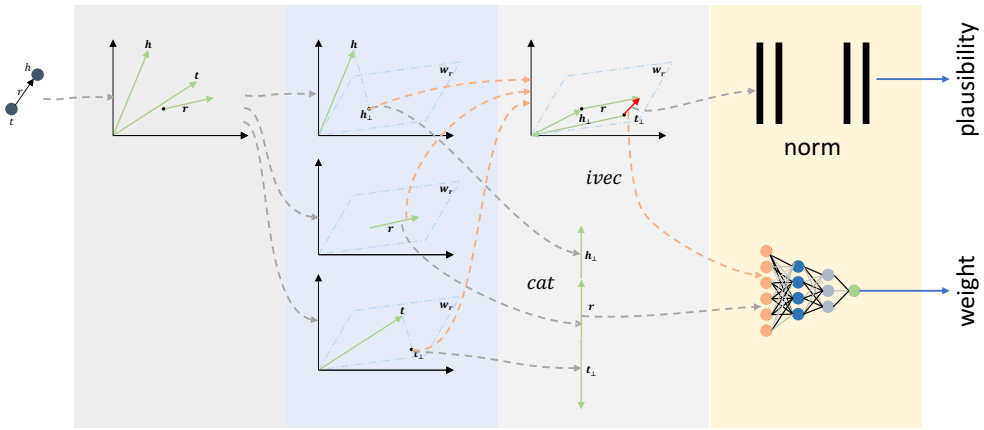


Figure 4.5: An example of the proposed framework based on TransH.

4.3.2 Training Protocol

For a given positive training set

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

we generate a corresponding negative set by replacing the head entity using all other entities and replacing the tail entity using all other entities:

$$\begin{aligned} S' &= \{(h', r, t')\} \\ &= \{(h'_i, r_i, t_i) \mid h'_i \in E \setminus \{h_i\}\}_{i=1}^{u} \\ &\quad \cup \{(h_i, r_i, t'_i) \mid t'_i \in E \setminus \{t_i\}\}_{i=1}^{u}. \end{aligned} \tag{4.9}$$

We adopt margin ranking loss [42] to measure the loss on learning the facts:

$$\mathcal{L}_{link} = \sum_{(h,\ell,t)\in S} \sum_{(h',\ell,t')\in S'} [\gamma + f(h,r,t) - f(h',r,t')]_+ \tag{4.10}$$

We measure the loss of the weight prediction module on learning the weight of the positive triple using

$$\mathcal{L}_{weight} = \frac{|w - w_p|}{w} \tag{4.11}$$

The total loss of the model is

$$\mathcal{L} = (1 - \alpha) \cdot \mathcal{L}_{link} + \alpha \cdot \mathcal{L}_{weight} \tag{4.12}$$

where the combination coefficient α is a hyper-parameter that balances the model between learning facts and learning weights.

4.3.3 Weighted Link Prediction Task

Task Description

Weighted link prediction (WLP) aims to simultaneously add missing relations and the corresponding missing weights to the incompleted WKGs. We describe WLP as follows:

Given a weighted knowledge graph

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

where $h_i, t_i \in E$, $r_i \in R$ and $w_i \in (0, 1]$, the E and R are entity and relation sets, respectively. A corrupted weighted triple is defined as a weighted triple without the relation and the weight, i.e., $\langle (h, ?, t), ? \rangle$. WLP is to complete the missing relations and the weights of the corrupted weighted triples in WKG, making them to completed weighted triples of the form $\langle (h, r, t), w \rangle$.

Evaluation Protocol

For a test weighted triple $\langle (h_i, r_i, t_i), w_i \rangle$, w_i is omitted. The head entity h_i is replaced by each of the entities of the dictionary in turn to form all possible triples $\langle (h_j, r_i, t_i), ? \rangle |_{j=1}^{j=|u|}$. Triple scores are calculated by the scoring function of the base model and then sorted in ascending order. After sorting, the ranking of the testing weighted triple rk_i is recorded. This whole procedure is repeated while removing t_i instead of h_i.

We measure the accuracy of predicting the weight by

$$acc(h, r, t, w) = \frac{w - |w - w_p|}{w} \tag{4.13}$$

We adjust the ranking of the positive triple rk_i using the accuracy of the weight prediction $acc(h_i, r_i, t_i, w_i)$ and a threshold τ:

$$\mathrm{rk}_i^{'} = \mathrm{rk}_i \cdot exp\left(\tau - acc(h_i, r_i, t_i, w_i)\right) \tag{4.14}$$

We adopt mean rank (MR), mean reciprocal rank (MRR), and Hits@N (Hits@N) to measure the performance of the models on WLP, shown in Equations 4.15, 4.16, and 4.17, respectively.

$$\mathrm{MR} = \sum_{i=1}^{u} \mathrm{rk}_i^{'}, \tag{4.15}$$

$$\mathrm{MRR} = \sum_{i=1}^{u} \frac{1}{\mathrm{rk}_i^{'}}, \tag{4.16}$$

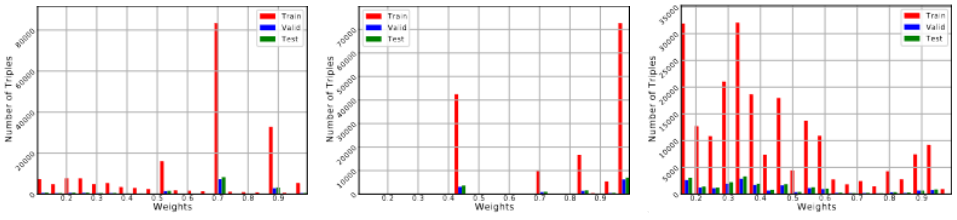
$$\mathrm{Hits@N} = \sum_{i=1}^{u} I[\mathrm{rk}_i^{'} \leq N] \tag{4.17}$$

where the $I[expn]$ is the indicator function, which outputs 1 if $expn$ is true, and 0 otherwise. We present the complete evaluation procedure in Algorithm 3.

4.4 Experiments and Results

To measure the performance of the proposed WeExt framework, we evaluate the weighted extensions of the base models on link prediction, weight prediction, and weighted link prediction.

4.4.1 Experiment Setting



(a) CN15K. (b) NL27K. (c) PPI5K.

Figure 4.6: The weight distribution in the datasets.

We conducted experiments on CN15K, NL27K, and PPI5K [111] datasets. CN15K is a subgraph of ConceptNet [75], containing 15,000 entities and

Algorithm 3: Weighted Link Prediction

Input: $\mathcal{WKG} = \{\langle (h, r, t), w\rangle\}$, KGE model $\mathcal{M}$, threshold of accuracy th

Result: {MR, MRR, Hits@N}

ranks, r_ranks = 0, 0
hits = {k:0}
while *i in range(|$\mathcal{WKG}$|)* **do**
  mix_t, mix_h, score$_t$, $score_h = [(h_i, r_i, t_i)], [(h_i, r_i, t_i)], [\ ], [\ ]$
  for $j = 0;\ j < |\mathcal{E}|;\ j = j + 1$ **do**
    if $(h_i, r_i, e_j) \notin \mathcal{WKG}$ **then**
      mix_t.append((h_i, r_i, e_j))
    if $(e_j, r_i, t_i) \notin \mathcal{WKG}$ **then**
      mix_h.append((e_j, r_i, t_i))
  end
  for $k = 0;\ k < |mix_t|;\ k = k + 1$ **do**
    score$_t$.*append*($\mathcal{M}$.*score*($mix_t[k]$))
  end
  $err_w = exp(th - max\,(abs(w_i - \mathcal{M}.predict_w((h_i, r_i, t_i)))/w_i, 0))$
  $rank_t$ = rank(score$_t$).*get*((h_i, r_i, t_i))
  ranks += $rank_t \cdot err_w$
  r_ranks += $1/(rank_t \cdot err_w)$
  for *k in hits.keys()* **do**
    if $rank_t <= k$ **then**
      hits[k] += err_w
  end
  for $k = 0;\ k < |mix_h|;\ k = k + 1$ **do**
    score$_h$.*append*($\mathcal{M}$.*score*($mix_h[k]$))
  end
  $rank_h$ = rank(score$_h$).*get*((h_i, r_i, t_i))
  ranks += $rank_h \cdot err_w$
  r_ranks += $1/(rank_h \cdot err_w)$
  for *k in hits.keys()* **do**
    if $rank_h <= k$ **then**
      hits[k] += err_w ;
  end
end
MR, MRR, Hits@N = ranks/$(2 \cdot |\mathcal{WKG}|)$, r_ranks/$(2 \cdot |\mathcal{WKG}|)$, hits[N]/$(2 \cdot |\mathcal{WKG}|)$

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 [0.1, 1.0]. NL27k is extracted from NELL [76], an weighted KG 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]. PPI5k is a subset of the protein-protein interaction knowledge base STRING [99] that contains 255,114 weighted triples for 4,999 proteins and 7 interactions. STRING labels the interactions between proteins with the probabilities of occurrence. The weights in PPI5k fall in the interval [0.15, 1.0]. We drop out duplicated quadruplets in CN15K and PPI5K. The statistics of the WKGs are shown in Table 4.1.

We implemented the proposed framework and the weighted link prediction task based on the PyKEEN toolkit [112]. We choose 0.01 as the learning rate λ for the stochastic gradient descent among and searched the combination coefficient α for loss function among $\{0.1, 0.2, 0.01, 0.001, 0.0001\}$. The margin of the loss function γ was set to 1. The dimension of embeddings was set to 50. We trained the models for 3000 epochs, evaluated the models per 10 epochs, and save their best results.

Table 4.1: Statistics of weighted knowledge graphs. #Ent denotes the number of the entities, #Rel denotes the number of the relations, #Tri denotes the number of the triples, INR denotes the interval of the weights, Avg(d) denotes the average of the degree of the entities, and Med(d) denotes the median of the degree of the entities.

		#Ent	#Rel	#Tri	INR	Avg(d)	Med(d)
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

4.4.2 Base Models

We implement the proposed framework based on two representative translational distance models: TransE and TransH, and two representative semantic

matching models: DistMult and HolE.

TransE

TransE [42] 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 of TransE is

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

TransH

TransE is effective for 1-to-1 relations, but cannot model the 1-to-N or N-to-N relation well. TransH [122] models 1-to-N and N-to-N relations by introducing the mechanism of projecting to relation-specific hyperplanes. The scoring of TransH is

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

where $\mathbf{w}_r$ stands for the normal vector of the relation-specific hyperplane.

DistMult

DistMult [103] represents each relation as a diagonal matrix that models pairwise interactions between entities to capture the latent semantics. The scoring of DistMult is

$$s = \|\mathbf{r} \circ \mathbf{h} \circ \mathbf{t}\|$$

where $\circ$ is the element-wise product.

HolE

For a given triple, HolE [47] first composes the head entity and tail entity using the circular correlation operation [125], then matches the relational with the compositional vector of the head entity and tail entity to score the given triple. Since circular correlation is not commutative, HolE is able to model asymmetric relations. The scoring of HolE is

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

where $\star$ is the circular correlation.

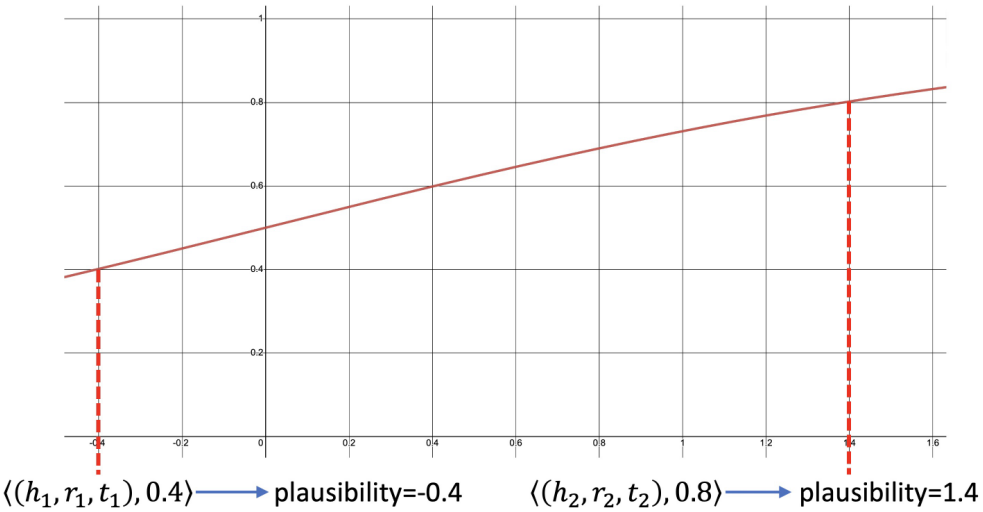


Figure 4.7: An illustration of how UKGE infers weights from the plausibility of triples. Given two triples A=$\langle (h_1, r_1, t_1), 0.4 \rangle$ and B=$\langle (h_2, r_2, t_2), 0.8 \rangle$, the non-linear function is sigmoid function: $s(t) = \frac{1}{1+e^{-1}}$. Let a well-trained UKGE model predict the plausibility of the given triples, the plausibility of triple-A will be -0.4 and plausibility of triple-B will be 1.4.

4.4.3 Results on Link Prediction

For a test triple (h_i, r_i, t_i), h_i is removed and replaced by each of the entities in the dictionary in turn to form all possible triples $(h_j, r_i, t_i) \,|_{j=0}^{j=|u|}$. Triple scores are calculated by the scoring function of the base model and sorted by ascending order. The rank of the positive triple is recorded as rk_i. This whole procedure is repeated while removing the tail entity instead of the head entity. We adopt MR, MRR, and Hits@N to measure the performance of the models on link prediction, as shown in Equation 4.15, Equation 4.16 and Equation 4.17, respectively. We choose UKGE as the baseline.

The results on Link prediction are shown in Table 4.2. Because MRR is not sensitive to extremely poor rankings, we mainly focus on the MRR score. After introducing a weight prediction module for base models, the extended models have to optimize their performance on both link prediction and weight prediction simultaneously. The results show that the introduced additional optimization term does not cause the model's performance to decrease on the link prediction task, there may be a slight improvement for some models instead, for example, TransE, TransH, and HolE. DistMultExt achieves a worse performance than DistMult on all the datasets. DistMultExt outperforms UKGE on CN15K and PPI5K, but due to the improved

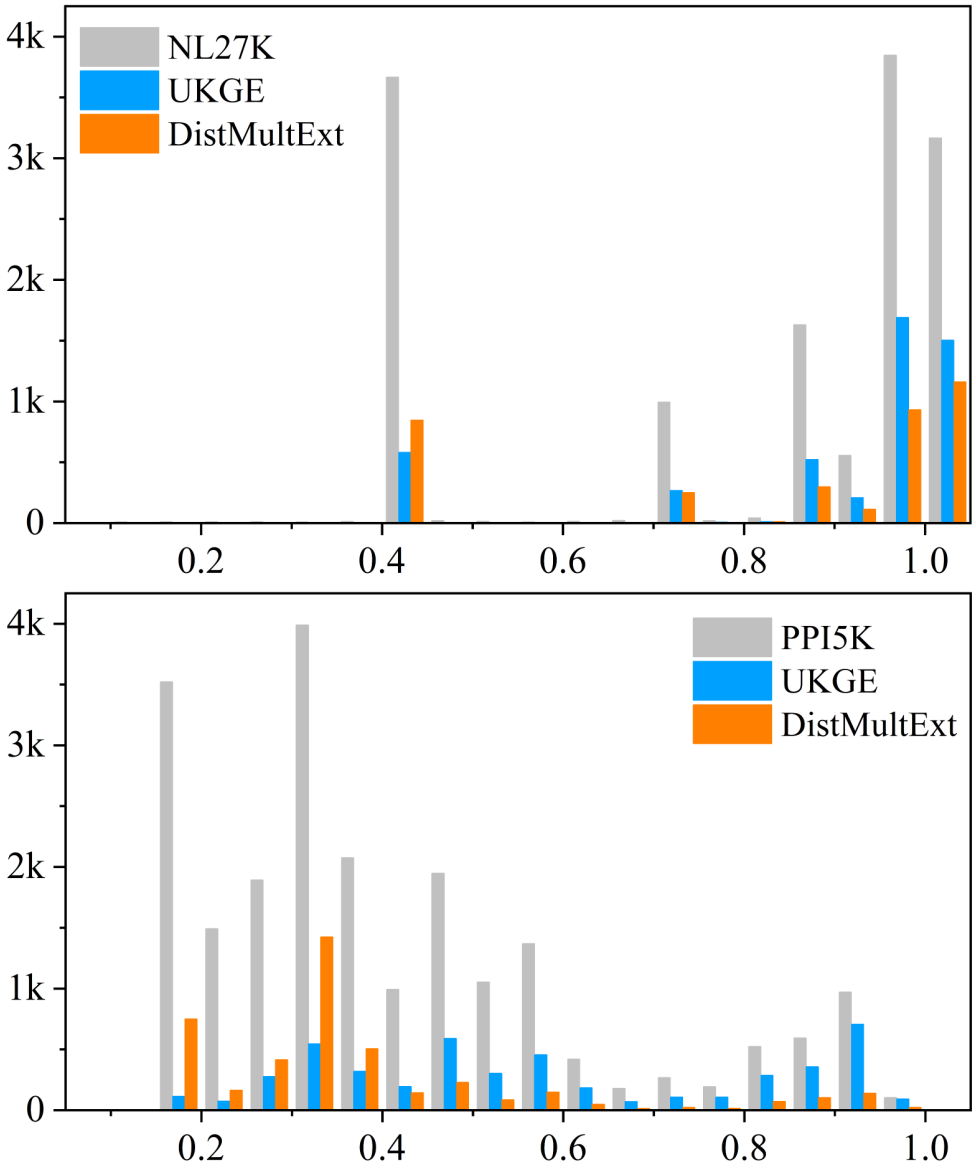


Figure 4.8: An illustration of the weight distribution of triples correctly predicted by UKGE and DistMultExt on NL27K and PPI5K.

performance of UKGE on NL27K compared to DistMult, the performance of UKGE on NL27K is much better than DistMultExt.

From another perspective, all the extended models outperform UKGE on CN15K. DistMultExt and HolEExt can surpass UKGE on PPI5K. But as for

NL27K, only HolEExt achieves better performance than UKGE. We assume the performance difference on the different datasets is caused by the weight distribution of the respective datasets, shown in Figure 4.6, and the way how UKGE learns the weights. Triples in NL27K are centralized in high-weight regions, triples in PPI5K are centralized in low-weight regions, and weights in CN15K are not so polarized as NL27K and PPI5K.

UKGE utilizes a non-linear function to squeeze the plausibility of triples to obtain the weights of the triples, which makes the triples with a small weight gap to gain a larger plausibility gap. An illustration is shown in Figure 4.7. Thus, UKGE tends to assign stronger plausibility for high-weight triples and weaker plausibility for low-weight triples, which is why UKGE's performance on PPI5K is not as good as half of the best-performing model. In contrast, on NL27K, where high-weight triples account for a large proportion, the gap between the performance of UKGE and the best-performing model is much smaller.

Figure 4.8 shows the weight distribution of triples correctly predicted by UKGE and DistMultExt on NL27K and PPI5K, we can see that UKGE predicts more triples with high weights, while DistMultExt performs more balanced on different intervals.

Moreover, WeExt with the *cat* preprocessing outperforms WeExt with the *ivec* preprocessing on PPI5K, and they achieve similar performance on CN15K and NL27K.

4.4.4 Results on Weight Prediction

For each weighted triple $\langle (h_i, r_i, t_i), w_i \rangle$ in the test set, we predict the weight based on the triple (h_i, r_i, t_i) and report the mean squared error (MSE) and mean absolute error (MAE).

The results on weight prediction are shown in Table 4.3. Except for TransHExt that performs worse than UKGE on NL27K, all the weighted extensions outperform UKGE in the weight prediction task for all three datasets. The result shows that adopting neural networks to learn weights from processed embeddings is superior to utilizing nonlinear functions to learn weights by squeezing the plausibility of the triple.

Moreover, WeExt with the *cat* preprocessing outperforms WeExt with the *ivec* preprocessing, not only for the translational distance models but also for the bilinear models, indicating that after the model is well-trained, the cascade of entities and relations retains richer information than the interaction vector.

4.4.5 Result on Weighted Link Prediction

The results on weighted link prediction are shown in Table 4.4. All weighted extensions outperform UKGE in weighted link prediction on CN15K. DistMultExt and HolEExt outperform UKGE on both NL27K and PPI5K. Although the link prediction performance of DistMultExt on NL27K is worse than UKGE, DistMultExt achieves better performance in the weighted link prediction on NL27K thanks to the better weight prediction performance.

Moreover, WeExt with the *cat* preprocessing outperforms WeExt with the *ivec* preprocessing on PPI5K, but they achieve similar performance on CN15K and NL27K. The performance of the weighted extensions on weighted link prediction is consistent with the performance trend on link prediction, but not consistent with the performance of the model's weighted prediction. This indicates that under the current evaluation protocol for weighted link prediction, the performance of the model on the link prediction task is dominant, while the performance of the model on the weight prediction task has been taken into consideration, but only in a subordinate position.

4.5 Summary

In this section, we propose a framework called WeExt for extending deterministic knowledge graphs to be capable of embedding weighted knowledge graphs. To facilitate the performance evaluation of our extended WKGE models, we propose the novel weighted link prediction task. Compared with the widely-used asynchronous link prediction and weight prediction tasks, weighted link prediction can synchronously evaluate the performance of weighted knowledge graph embedding in link prediction and weight prediction.

In the next work, we plan to design a new evaluation protocol to alleviate the impact of extreme data on the score, so that the score can better reflect the model performance.

Table 4.2: Results on link prediction

	Model		MR	MRR	Hits@1	Hits@3	Hits@5	Hits@10
CN15K	UKGE		1760.4	0.0800	0.0372	0.0840	0.1161	0.1634
	TransE		1240.2	0.1078	0.0371	0.1327	0.1818	0.2457
	TE_{Ext}	ivec	1206.4	**0.1091**	0.037	**0.1357**	**0.1821**	**0.2498**
		cat	1233.3	0.1084	0.0372	0.1324	0.1817	0.2466
	TransH		1716.3	0.0789	0.0419	0.0879	0.1107	0.1455
	TH_{Ext}	ivec	1747.5	0.079	0.0415	0.0898	0.1118	0.1442
		cat	1745.1	0.0804	**0.0425**	0.0899	0.1132	0.1486
	DistMult		**966.1**	0.1072	0.041	0.13	0.1711	0.2315
	DM_{Ext}	ivec	1105.7	0.0907	0.0408	0.1012	0.1339	0.1831
		cat	1208.5	0.0773	0.0373	0.0815	0.108	0.152
	HolE		1319.7	0.0935	0.0477	0.0994	0.1335	0.1832
	HE_{Ext}	ivec	1327.7	0.0946	0.0496	0.1014	0.1329	0.1813
		cat	1330.9	0.0945	0.0506	0.1007	0.1318	0.1794
NL27	UKGE		236.6	0.4550	0.3697	0.4879	0.5483	0.6268
	TransE		111.4	0.3736	0.2298	0.4554	0.5325	0.6344
	TE_{Ext}	ivec	114.8	0.3794	0.2366	0.4593	0.5364	0.6387
		cat	**109.2**	0.3818	0.2344	0.4662	0.5449	0.6465
	TransH		686.1	0.2721	0.1874	0.3125	0.3593	0.42
	TH_{Ext}	ivec	671.2	0.2798	0.1978	0.3168	0.3623	0.4226
		cat	677.5	0.2769	0.194	0.3156	0.3579	0.4193
	DistMult		179.1	0.3993	0.3065	0.4421	0.4939	0.57
	DM_{Ext}	ivec	264.8	0.3703	0.2822	0.4069	0.4573	0.5321
		cat	186.3	0.3729	0.2823	0.4118	0.4634	0.5354
	HolE		135	0.5223	0.4198	0.5716	0.639	0.7241
	HE_{Ext}	ivec	144.5	**0.5382**	**0.4377**	**0.5873**	**0.6507**	**0.7333**
		cat	134.8	0.5291	0.4277	0.579	0.641	0.7267
PPI5K	UKGE		29.3	0.3759	0.2405	0.4320	0.5092	0.6435
	TransE		19.6	0.1819	0.0001	0.2432	0.3704	0.5636
	TE_{Ext}	ivec	18.4	0.1815	0	0.2397	0.3682	0.5635
		cat	18.7	0.1818	0	0.2425	0.3679	0.5637
	TransH		49.5	0.1106	0.0029	0.1274	0.1906	0.3155
	TH_{Ext}	ivec	50.1	0.1196	0	0.1523	0.2193	0.3334
		cat	48.7	0.1199	0	0.1532	0.217	0.3283
	DistMult		24.1	0.459	0.3427	0.5175	0.5763	0.6662
	DM_{Ext}	ivec	34.6	0.4302	0.3266	0.473	0.5298	0.6186
		cat	32.6	0.4585	0.3549	0.4994	0.5584	0.6495
	HolE		6.6	0.8426	0.7589	0.9149	0.9473	0.9719
	HE_{Ext}	ivec	**6.1**	0.845	0.7628	0.9168	0.9479	0.9722
		cat	6.6	**0.8542**	**0.7762**	**0.9218**	**0.951**	**0.9743**

Table 4.3: Results on weight prediction

	Model		MSE	MAE
CN15K	UKGE		8.61	19.90
	TransE_{Ext}	ivec	4.60	14.66
		cat	3.83	13.08
	TransH_{Ext}	ivec	4.26	13.89
		cat	3.91	12.89
	DistMult_{Ext}	ivec	5.10	15.62
		cat	**3.67**	**11.93**
	HolE_{Ext}	ivec	5.86	17.09
		cat	5.79	16.31
NL27K	UKGE		2.36	6.90
	TransE_{Ext}	ivec	1.45	5.98
		cat	1.23	5.24
	TransH_{Ext}	ivec	2.12	8.12
		cat	3.05	10.46
	DistMult_{Ext}	ivec	1.39	6.13
		cat	**1.22**	5.09
	HolE_{Ext}	ivec	1.31	**5.01**
		cat	1.26	5.38
PPI5K	UKGE		0.95	3.79
	TransE_{Ext}	ivec	0.24	2.77
		cat	0.24	2.58
	TransH_{Ext}	ivec	0.49	3.7
		cat	0.42	3.32
	DistMult_{Ext}	ivec	0.34	2.89
		cat	0.28	2.70
	HolE_{Ext}	ivec	0.16	**1.83**
		cat	**0.15**	1.85

Table 4.4: Results on weighted link prediction

	Model		MR	MRR	Hits@1	Hits@3	Hits@5	Hits@10
CN15K	UKGE		4031.2	0.0795	0.0266	0.0736	0.1066	0.1556
	TE_{Ext}	ivec	3602	0.1129	0.035	0.1153	0.1639	0.2295
		cat	4467.6	**0.1154**	0.0342	**0.1174**	**0.1694**	**0.2361**
	TH_{Ext}	ivec	4907.8	0.0865	0.0393	0.0821	0.1059	0.1398
		cat	4551.3	0.0868	0.0381	0.0829	0.1085	0.1438
	DM_{Ext}	ivec	3071.9	0.0961	0.0365	0.0898	0.1242	0.1749
		cat	3224.2	0.083	0.033	0.0737	0.1025	0.147
	HE_{Ext}	ivec	**2761.2**	0.0984	0.039	0.0924	0.1263	0.1777
		cat	2786.5	0.0992	**0.0433**	0.0886	0.1201	0.1696
NL27K	UKGE		483.5	0.5269	0.3566	0.4844	0.5521	0.637
	TE_{Ext}	ivec	**137.1**	0.4198	0.1967	0.4481	0.5356	0.6491
		cat	149.6	0.4093	0.1801	0.4429	0.5378	0.6462
	TH_{Ext}	ivec	621.5	0.2996	0.156	0.3044	0.3547	0.4258
		cat	633.8	0.2942	0.1455	0.3039	0.3544	0.4221
	DM_{Ext}	ivec	280.7	0.4148	0.2518	0.4012	0.456	0.536
		cat	382	0.419	0.2558	0.3996	0.4581	0.5438
	HE_{Ext}	ivec	272	**0.624**	**0.4128**	**0.58**	**0.6563**	**0.746**
		cat	339.1	0.6038	0.3941	0.5579	0.636	0.7236
PPI5K	UKGE		34.3	0.4212	0.2131	0.4193	0.5073	0.6518
	TE_{Ext}	ivec	18.6	0.1996	0	0.2227	0.3574	0.5631
		cat	18.3	0.2018	0	0.225	0.3697	0.5772
	TH_{Ext}	ivec	47.4	0.1336	0	0.1459	0.2174	0.3441
		cat	45.3	0.1363	0	0.147	0.217	0.345
	DM_{Ext}	ivec	34.5	0.4782	0.2835	0.4641	0.5263	0.6235
		cat	33.7	0.5124	0.309	0.4906	0.5563	0.6547
	HE_{Ext}	ivec	**7.7**	0.9769	0.7241	0.9096	0.9457	**0.9722**
		cat	22.6	**0.9948**	**0.7448**	**0.914**	**0.9473**	0.9717

Chapter 5

Knowledge-guided Word Embedding Fine-tuning Model

5.1 Problem Statement

Word embedding models represent words as real-valued vectors in a semantic vector space, promoting the development of many natural language processing tasks. The most intuitive way to represent words is using one-hot vectors, which contain one at the words' corresponding index and zeros in all other indices. This method can assign an independent vector for each word; however, the angle between any two one-hot vectors in the semantic space is a right angle, thus it is hard to measure the similarity between two words. Apart from this, the one-hot vectors face the curse of dimensionality problem while the vocabulary expands because the dimension of any one-hot vector equals to the vocabulary's size.

To address the above deficiencies, some neural-based architectures (cf. [126, 127, 128]) have heen proposed to learn word embeddings based on the distributional hypothesis: words appearing in a similar context must have similar meanings and are thus represented similarly. The neural-based architectures can learn word representations that are superior to one-hot vectors on the word similarity task. However, there is a lack of supervision between a target word and its context words when training with such architectures. Specifically, it is possible that words appearing in a context window may not be related and the related words do not appear in the same context window. This shows why the size of a context window can significantly impact the performance of the word embeddings on the word similarity task [129]. In other words, neural-based models merely learn to detect which words appear with a target word in a given context window regardless of their semantical

relationship with the target word.

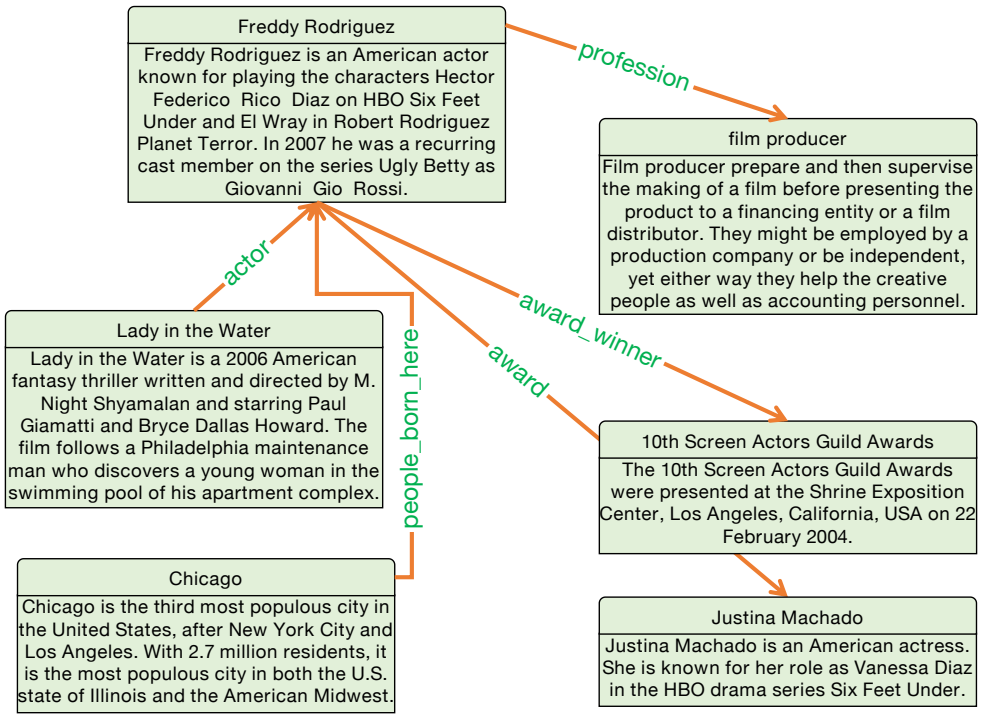


Figure 5.1: An example of entity-description pairs extracted from Freebase.

Many models are proposed to enhance the quality of word embeddings with knowledge graphs [130, 131]. For instance, Xu *et al.* [132] defined regularization functions to incorporate knowledge into a skip-gram model for strengthening the learnt vectors. Roy *et al.* [133] extended the original objective function of distributed representation learning to embed different knowledge sources during training. Wang *et al.* [53] combined a knowledge graph model, a text model, and an alignment model into the single objective function for jointly training entities and words within the same vector space. Tissier *et al.* [134] augmented an input text with extra knowledge and used a skip-gram model for training the embeddings. Faruqui *et al.* [135] developed a retrofitting method to fine-tune word embeddings so that they satisfy additional constraints defined in semantic lexicons. While these methods utilize knowledge sources for improving the quality of word embeddings, there is no study that enhances the word embeddings with knowledge graph embeddings (cf. the review in [136]). Thus, we aim to fill this gap in this work.

It is worth mentioning that some knowledge graphs (e.g. [117, 1]) contains not only rich knowledge information in form of triples, but also rich semantic

information in form of concise descriptions. Figure 5.1 illustrates an example of the related entity-description pairs sampled from Freebase [117]. Since each entity and its description refer to the same object in the real world, we assume that words within the entity description are highly related to that entity. In addition, we assume that words in the descriptions of the related entities are also highly related to each other. Exceptional models that follow this principle are [54, 137]. On the one hand, these works utilize embeddings of the descriptions to improve the quality of entity embeddings. On the other hand, this research considers the opposite direction of [54, 137]; that is, our research question is to investigate how entity embeddings can be leveraged to enhance the quality of word embeddings.

For this purpose, this work introduces a novel **K**nowledge-**G**uided **W**ord **E**mbedding fine-tuning model (KGWE). Informally, KGWE adopts entity-description pairs taken from knowledge graphs and entity embeddings trained by knowledge graph embedding models as training inputs. For each entity-description pair, it embeds the entity and its description using entity embeddings and word embeddings, respectively, where the entity embeddings are fixed. Then, KGWE encodes the embedded description and trains to maximize the likelihood of predicting each embedded entity in order to obtain the higher quality of word embeddings for any words within the entity descriptions. To ensure the effectiveness of KGWE, we conduct experiments with two different word embedding models (i.e., GloVe [128] and Word2Vec [127]) and five different entity embedding models (i.e., ANALOGY [138], DistMult [46], TransD [139], TransE [42] and TransH [122]) with four different dimensions (50, 100, 200 and 300). We also implement two encoders for encoding entity descriptions: a bag-of-word (BoW)-based encoder and an RNN-based encoder. Our experiments show that both Word2Vec and GloVe processed by KGWE not only gain significant performance improvement on word similarity task, but also outperform the baselines on the relation classification task, and the sentence polarity classification task. Also, the BoW-based encoder with a simpler architecture yields the better performance than the RNN-based encoder.

In sum, the main contribution of this section is a knowledge-guided fine-tuning technique for utilizing knowledge graph resources to obtain higher-quality semantic word embeddings. In contrast to the existing works, KGWE extracts the related words of the entities from the entity-description pairs that can lead to higher-quality training data. KGWE utilizes powerful entity embeddings learned by the knowledge graph embedding models, which have been shown to be more effective at integrating the knowledge into word embeddings.

In our previous study [140], our experiments have verified the effective-

ness of KGWE on word similarity task. In this section, we evaluate the performance of KGWE with RNN-based encoder and BoW-based encoder using extrinsic evaluation, to prove the effectiveness of KGWE on natural language processing downstream tasks. The results demonstrate that KGWE can also improve the performance of word embeddings on downstream natural language processing tasks.

5.2 Related Works

The existing word embeddings' learning approaches can be categorized into two main streams [141, 130]: (1) static word embedding models (cf. [127, 128]) and (2) dynamic word embedding models (cf. [142, 143]). Informally, static word embedding models learn context-independent vectors for words in a vocabulary regardless of any context, while dynamic word embedding models learn a language model which can generate different word embeddings for a given word based on different context.

To integrate knowledge into dynamic word embeddings, ERNIE [144] randomly masks some word-entity alignments and then predicts all corresponding entities based on aligned word embeddings. KnowBert [145] utilizes a word-to-entity attention mechanism to align word embeddings and entity embeddings. KEPLER [146] is a unified model which encodes textual entity descriptions with a pre-trained language model, and then jointly optimizes the knowledge graph embedding and language modeling objectives.

It is worth noting that static word embeddings have several superiorities, although dynamic word embeddings can achieve better performance [147]. For instance, training dynamic (contextual) word embeddings is often resource-consuming, even if ignoring the training phase. The computational cost of using static word embeddings is typically tens of millions times lower than using dynamic embedding models. Many NLP tasks inherently rely on static word embeddings; for example, for the purpose of interpretability [148], bias detection [149] and removal [150], analyzing word vector spaces[151] or other metrics which are non-contextual by choice. Furthermore, static word embedding can complement dynamic word embedding in different situations; for instance, for separating static from contextual semantics, or for improving joint embedding performance on downstream tasks. Our KGWE focuses on enhancing the static word embedding. Thus, we specifically make further reviews as follows.

The development of static word representations can also further divided into two approaches: (1) global context-based models [152] and (2) local context-based models [127, 128]. The global context-based models firstly

summarize the statistical information of the context into a high dimensional sparse co-occurrence matrix, and then apply dimensionality reduction techniques on the co-occurrence matrix in order to produce lower dimensional vectors. On the other hand, local context-based models learn word embeddings which appear in every sliding window based on a neural network model. Apropos to these approaches, the training objective functions are defined to predict: (1) the next word given a sequence of words, (2) a target word given its context words, or (3) all contexts given a target word.

Moreover, the models of utilizing knowledge for enhancing static word embeddings can be categorized into two approaches: (1) co-training and (2) fine-tuning. The co-training models train word embeddings using both text and external knowledge simultaneously, whereas the fine-tuning models fine-tune the pre-trained word embeddings based on the external knowledge. Examples of the co-training approach are WAE [133], Dict2Vec [134], RC-NET [132], and MPME [153]. An example of the fine-tuning approach is Retrofitting [135]. We briefly discuss each of them in the following.

Firstly, WAE lets the target word predict both of its context words and its labels to encode diverse information from heterogeneous knowledge sources (e.g. human annotations, raw texts, malware attribute enumeration, and characterization specifications) into word embeddings. Secondly, Dict2Vec adopts new word co-occurrence pairs as a kind of weak supervision to Word2Vec [127]. The word co-occurrence pairs used by Dict2Vec are extracted from the word definitions in natural language dictionaries, which are supposed to contain latent word similarity and relatedness information. Thirdly, RC-NET attaches the relation restriction and the category restriction to Word2Vec. The relation restriction is extracted by a knowledge graph embedding model. The word categories are formed by the head words or the tail words of the same relation. The category restriction aims to make words in the same category closer to each other than before. Fourthly, [53] trains a Word2Vec model and a knowledge graph embedding model separately, and align the two models using the anchors extracted from Wikipedia or the entity names. Fifthly, MPME [153] jointly learns the representations of words, entities, and mentions, to bridge text and knowledge representations at the sense level. The representations are learned by separate models and aligned by a unified optimization objective. Different mention senses are distinguished by taking advantage of both textual context information and knowledge of reference entities. Lastly, Retrofitting is to make the embeddings of the related words closer to each other than before in the semantic vector space, under the premise of minor changes to the word embeddings. The related words are generated from the word sets in semantic lexicons.

KGWE is conceptually similar to Retrofitting in a sense that both of

them utilize the external knowledge to fine-tune the static word embeddings. However, KGWE differs from Retrofitting in the aspect that KGWE fine-tunes the pre-trained word embeddings using the entity embeddings learned by knowledge graph embedding models, while Retrofitting adopts constraints extracted from semantic lexicons.

5.3 KGWE: Knowledge-Guided Word Embedding Fine-tuning Model

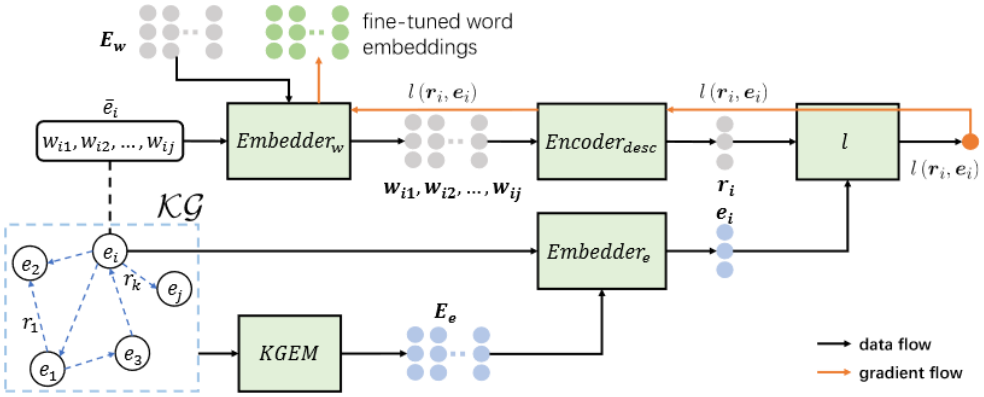


Figure 5.2: The proposed model architecture of KGWE

To fine-tune word embeddings under the guidance of knowledge, we design our KGWE model as shown in Figure 5.2. Formally, let $\mathcal{KG} := \{(h_i, r_i, t_i)\}_{i=1}^{N}$ be a knowledge graph containing N triples of head, relation, and tail entities. Each head/tail entity in $\mathcal{KG}$ is also associated with a sequence of words as its description, denoted by . For instance, $\overline{h}$ refers to the description of head entity h. In addition, we also denote the set of all head/tail entities within $\mathcal{KG}$ by $\mathcal{E}$. Assume that one can find well-quality entity embeddings of a knowledge graph; for instance, via an adoption of ANALOGY, DistMult, TransD, TransE, and TransH. The main idea of KGWE is to fine-tune an embedding of each word w in description $\overline{e}$ with $e \in \mathcal{E}$ so that the vector representation of description $\overline{e}$ is aligned closely to the adopted (and fixed) entity embedding.

To intuitively explain the mechanism of KGWE, we consider an arbitrary e and its description $\overline{e}$ in $\mathcal{KG}$ as an example. Here, $e_i \in \mathcal{E}$ and $\overline{e}_i := (w_{i1}, w_{i2}, \ldots, w_{ij})$ be a sequence of j-words. KGWE consists of sequential operations as follows:

Firstly, a given knowledge graph $\mathcal{KG}$ is fed into a knowledge graph embedding model (KGEM), which can be implemented by ANALOGY, DistMult, TransD, TransE, or TransH, to learn the entity embeddings E_e and the relation embeddings E_r:

$$E_e, E_r \coloneqq KGEM(\mathcal{KG}) \tag{5.1}$$

Secondly, given E_e and an arbitrary word embedding E_w, the target of KGWE here is to fine-tune E_w in a way that the representation of $\overline{e}_i$ becomes closely to the same one in E_e. Let $Embedder_e : \mathcal{E} \times R^{|\mathcal{E}| \times n} \rightarrow R^n$ be a function that returns an entity embedding and $Embedder_w : \mathcal{S} \times R^{|\mathcal{E}| \times n} \rightarrow R^{j \times n}$ be a function that returns a sequence of word embeddings for the description $\overline{e}_i \in \mathcal{S}$. The next operations are follows:

$$e_i \coloneqq Embedder_e(e_i, E_e) \tag{5.2}$$

$$(w_{i1}, w_{i2}, \ldots, w_{ij}) \coloneqq Embedder_w(\overline{e}_i, E_w) \tag{5.3}$$

Thirdly, the embedded description $(w_{i1}, w_{i2}, \ldots, w_{ij})$ is fed into the description encoder $Encoder_{desc} : R^{j \times n} \rightarrow R^n$ to obtain a representation of the description, denoted by r_i, as follows:

$$r_i \coloneqq Encoder_{desc}(w_{i1}, w_{i2}, \ldots, w_{ij}) \tag{5.4}$$

The final operation is to train for supervising the vector representation r_i closely aligned with the embedded entity e_i by optimizing the following loss function l:

$$l(r_i, e_i) \coloneqq (r_i - e_i)^2 \tag{5.5}$$

Intuitively, KGWE treats the embedded entity e_i as the label of description representation r_i. After training, KGWE considers the fine-tuned E_w as its desired output, i.e., the knowledge-guided word embedding. Note that Equation 5.5 spells out that KGWE computes the squared error for any entity i. Thus, our implementation defines the total loss $\mathcal{L}$ for the whole dataset as the mean squared error over N triples as follows:

$$\mathcal{L} \coloneqq \frac{1}{N} \sum_{i=1}^{N} l(r_i, e_i) \tag{5.6}$$

Finally, we discuss our two implementations of the description encoder $Encoder_{desc}$ proposed in this work for KGWE: (1) a RNN-based encoder and a BoW (Bag of Word)-based encoder for self-containment. Formally, we utilize the Gated Recurrent Unit (GRU) [154] to define our RNN-based encoder, as in Equation 5.7. For the BoW-based encoder, it computes the summation

Table 5.1: Word similarity results of KGWE with the Bow-based encoder on GloVe.6B.50d

		WordSim353			MEN			RMT	Rare	Sim	Sim	SE17T2		Baker	YP
		all	simi	rel	full	dev	test		Words	Lex	Verb	trial	test	Verb	Verb
	GloVe	0.414	0.552	0.348	0.652	0.644	0.669	0.619	0.266	0.265	0.154	0.414	0.362	0.250	0.386
Retrofitting	FrameNet	0.394	0.557	0.322	0.622	0.611	0.645	0.600	0.279	0.289	0.217	0.420	0.364	0.233	0.461
	PPDB	0.433	0.572	0.358	**0.686**	**0.675**	**0.709**	**0.647**	0.277	**0.399**	**0.244**	0.472	0.374	**0.308**	0.459
	WordNet_{syn}	0.407	0.561	0.324	0.649	0.641	0.667	0.621	0.267	0.338	0.239	0.465	0.388	0.191	**0.508**
	WordNet_{all}	0.405	0.577	0.296	0.674	0.666	0.690	0.611	0.222	0.371	0.238	0.414	0.390	0.208	0.444
KGWE	ANALOGY	0.447	**0.583**	0.388	0.652	0.644	0.669	0.619	0.273	0.265	0.154	0.649	0.450	0.250	0.386
	DistMult	0.446	0.583	0.386	0.652	0.644	0.669	0.619	0.285	0.265	0.154	0.649	0.447	0.250	0.386
	TransD	0.447	0.583	0.387	0.652	0.644	0.669	0.619	**0.293**	0.265	0.154	0.649	0.447	0.250	0.386
	TransE	0.445	0.583	0.384	0.652	0.644	0.669	0.619	0.278	0.265	0.154	0.647	0.446	0.250	0.386
	TransH	**0.447**	0.582	**0.389**	0.652	0.644	0.669	0.619	0.268	0.265	0.154	**0.649**	**0.457**	0.250	0.386

of each word embedding in the embedded entity description with an equal weight (**1**) to obtain the description representation (cf. Equation 5.8). The structure of each encoder is illustrated in Figure 5.3.

$$r_i \coloneqq GRU(w_{i1}, w_{i2}, \ldots, w_{ij}) \tag{5.7}$$

$$r_i \coloneqq \sum_{o=1}^{j} \mathbf{1} \odot w_{io} \tag{5.8}$$

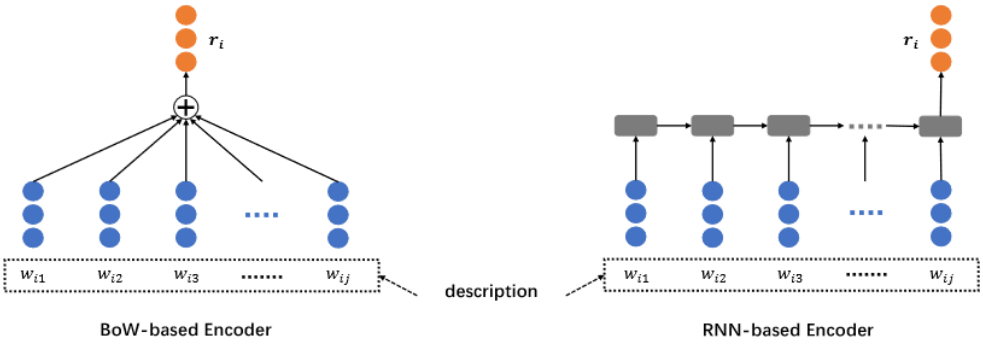


Figure 5.3: The structures of BoW-based encoder and RNN-based encoder.

5.4 Experiments and Results

5.4.1 Training Data and Experimental Setting

To evaluate the proposed model, we conducted experiments to fine-tune Word2Vec[127] and GloVe[128]. We adopted 300-dimensional Word2Vec

Table 5.2: Word similarity results of KGWE with the Bow-based encoder on GloVe.6B.100d

		WordSim353			MEN			RMT	Rare	Sim	Sim	SE17T2		Baker	YP
		all	simi	rel	full	dev	test		Words	Lex	Verb	trial	test	Verb	Verb
GloVe.6B.100d		0.443	0.580	0.376	0.681	0.676	0.689	0.619	0.288	0.298	0.180	0.484	0.367	0.302	0.453
Retrofitting	FrameNet	0.411	0.566	0.343	0.639	0.631	0.657	0.607	0.298	0.305	0.240	0.474	0.365	0.273	0.511
	PPDB	0.467	0.606	0.396	**0.710**	**0.702**	**0.727**	**0.653**	0.302	**0.430**	**0.266**	0.480	0.382	**0.364**	0.516
	WordNet$_{syn}$	0.434	0.582	0.351	0.676	0.668	0.692	0.622	0.292	0.365	0.263	0.443	0.387	0.230	**0.585**
	WordNet$_{all}$	0.439	0.607	0.336	0.698	0.692	0.709	0.625	0.235	0.400	0.264	0.401	0.399	0.254	0.509
KGWE	ANALOGY	0.474	0.612	0.414	0.681	0.676	0.689	0.619	0.289	0.298	0.180	0.736	0.453	0.302	0.453
	DistMult	0.477	0.615	**0.418**	0.681	0.676	0.689	0.619	**0.307**	0.298	0.180	**0.738**	**0.458**	0.302	0.453
	TransD	0.475	**0.613**	0.414	0.681	0.676	0.689	0.619	0.299	0.298	0.180	0.719	0.457	0.302	0.453
	TransE	**0.477**	0.613	0.417	0.681	0.676	0.689	0.619	0.288	0.298	0.180	0.719	0.461	0.302	0.453
	TransH	0.475	0.612	0.413	0.681	0.676	0.689	0.619	0.295	0.298	0.180	0.719	0.458	0.302	0.453

Table 5.3: Word similarity results of KGWE with the Bow-based encoder on GloVe.6B.200d

		WordSim353			MEN			RMT	Rare	Sim	Sim	SE17T2		Baker	YP
		all	simi	rel	full	dev	test		Words	Lex	Verb	trial	test	Verb	Verb
GloVe.6B.200d		0.482	0.608	0.416	0.710	0.707	0.717	0.620	0.314	0.340	0.198	0.563	0.393	0.284	0.515
Retrofitting	FrameNet	0.439	0.587	0.374	0.667	0.660	0.679	0.604	**0.323**	0.333	0.250	0.525	0.388	0.249	0.539
	PPDB	0.498	0.628	0.426	**0.738**	**0.732**	**0.750**	**0.655**	0.319	**0.468**	0.281	0.540	0.403	**0.333**	0.557
	WordNet$_{syn}$	0.474	0.610	0.388	0.705	0.699	0.717	0.617	0.316	0.398	**0.281**	0.492	0.418	0.214	**0.624**
	WordNet$_{all}$	0.474	0.632	0.368	0.729	0.725	0.736	0.627	0.252	0.437	0.279	0.498	0.428	0.228	0.570
KGWE	ANALOGY	**0.518**	0.640	**0.459**	0.710	0.707	0.717	0.620	0.314	0.340	0.198	0.816	0.489	0.284	0.515
	DistMult	0.515	0.639	0.453	0.710	0.707	0.717	0.620	0.318	0.340	0.198	0.800	0.487	0.284	0.515
	TransD	0.516	0.640	0.454	0.710	0.707	0.717	0.620	0.316	0.340	0.198	**0.816**	**0.493**	0.284	0.515
	TransE	0.516	**0.641**	0.455	0.710	0.707	0.717	0.620	0.319	0.340	0.198	0.814	0.485	0.284	0.515
	TransH	0.515	0.640	0.452	0.710	0.707	0.717	0.620	0.314	0.340	0.198	0.814	0.488	0.284	0.515

Table 5.4: Word similarity results of KGWE with the Bow-based encoder on GloVe.6B.300d

		WordSim353			MEN			RMT	Rare	Sim	Sim	SE17T2		Baker	YP
		all	simi	rel	full	dev	test		Words	Lex	Verb	trial	test	Verb	Verb
GloVe.6B.300d		0.736	0.803	0.688	0.802	0.803	0.798	0.665	0.297	0.408	0.283	0.779	0.586	0.341	0.571
Retrofitting	FrameNet	0.474	0.625	0.403	0.703	0.698	0.711	0.629	0.347	0.363	0.279	0.542	0.415	0.273	0.581
	PPDB	0.543	0.674	0.467	0.765	0.760	0.775	0.675	0.352	0.496	0.312	0.562	0.432	0.344	0.588
	WordNet$_{syn}$	0.518	0.655	0.433	0.738	0.734	0.747	0.646	0.342	0.432	0.314	0.492	0.44	0.233	0.645
	WordNet$_{all}$	0.52	0.68	0.413	0.761	0.759	0.766	0.648	0.277	0.468	0.312	0.507	0.454	0.25	0.597
KGWE	ANALOGY	0.736	0.803	0.688	**0.802**	**0.803**	0.798	0.669	0.320	0.408	0.283	0.779	0.587	0.341	0.571
	DistMult	0.736	0.803	0.688	0.802	0.803	0.798	0.666	0.308	0.408	0.283	0.779	**0.588**	0.341	0.571
	TransD	0.736	0.803	0.688	0.802	0.803	**0.798**	**0.669**	0.322	0.408	0.283	0.779	0.588	0.341	0.571
	TransE	0.736	0.803	0.688	0.802	0.803	0.798	0.666	0.315	0.408	0.283	0.779	0.586	0.341	0.571
	TransH	0.736	0.803	0.688	0.802	0.803	0.798	0.666	**0.325**	**0.408**	0.283	0.779	0.587	0.341	0.571

trained on the Google News Corpus.We adopted four different dimensions (50, 100, 200, 300) for GloVe trained on Wikipedia and Gigaword corpus. We adopted five types of entity embeddings trained on FB15K[42], i.e., ANALOGY[138], DistMult[46], TransD[139], TransE[42] and TransH[122]. We adopt the OpenKE[1][155] toolkit to train the entity embeddings with en-

[1]https://github.com/thunlp/OpenKE

Table 5.5: Word similarity results of KGWE with the Bow-based encoder on Word2Vec 300d

		WordSim353			MEN			RMT	Rare	Sim	Sim	SE17T2		Baker	YP
		all	simi	rel	full	dev	test		Words	Lex	Verb	trial	test	Verb	Verb
	Word2Vec	0.698	0.772	0.635	0.732	0.733	0.731	0.490	0.268	0.443	0.364	0.657	0.481	0.450	0.553
Retrofitting	FrameNet	0.104	0.165	0.036	0.239	0.239	0.243	0.320	0.178	0.160	0.127	0.168	0.125	0.227	0.214
	PPDB	0.192	0.245	0.129	0.270	0.269	0.273	0.336	0.165	0.309	0.174	0.333	0.174	0.269	0.298
	WordNet$_{syn}$	0.174	0.232	0.093	0.214	0.222	0.198	0.315	0.181	0.214	0.165	0.290	0.135	0.232	0.253
	WordNet$_{all}$	0.205	0.274	0.126	0.252	0.260	0.234	0.351	0.123	0.276	0.148	0.092	0.171	0.246	0.197
KGWE	ANALOGY	0.698	0.772	0.635	0.734	0.736	0.732	0.522	**0.282**	0.443	0.364	**0.684**	0.482	0.454	0.553
	DistMult	0.698	0.772	0.635	0.733	0.733	0.732	0.522	0.269	0.443	0.364	0.657	0.482	**0.464**	0.553
	TransD	0.698	0.772	0.635	0.732	0.733	0.731	0.516	0.268	0.443	0.364	0.657	0.481	0.450	0.553
	TransE	0.698	0.772	0.635	0.733	0.733	0.732	0.529	0.280	0.443	0.364	0.657	0.483	0.451	0.553
	TransH	0.698	0.772	0.635	**0.736**	**0.738**	**0.734**	**0.543**	0.282	0.443	0.364	0.657	**0.489**	0.453	0.553

Table 5.6: Statistical information of the dataset used in our experiment

		Total Pairs	Non OOV		Half OOV		Both OOV	
WS353	all	352	182	51.7%	134	38.1%	36	10.2%
	sim	203	95	46.8%	82	40.4%	26	12.8%
	rel	252	136	54%	98	39%	18	7.1%
MEN	full	3000	764	25.5%	1324	44.1%	912	30.4%
	dev	2000	508	25.4%	882	44.1%	610	30.5%
	test	1000	256	25.6%	442	44.2%	302	30.2%
RMTurk		287	71	24.7%	137	47.7%	79	27.5%
RareWords		2034	113	5.6%	896	44.1%	1025	50.4%
SimLex999		999	330	33%	413	41.3%	256	25.6%
SimVerb3500		3500	550	15.7%	1491	42.6%	1459	41.7%
SE17T2	trial	18	12	66.7%	5	27.8%	1	5.6%
	test	500	187	37.4%	212	42.4%	101	20.2%
BakerVerb143		144	75	52.1%	62	43.1%	7	4.9%
YPVerb130		130	17	13.1%	48	36.9%	65	50%

tity/relation dimension in {50, 100, 200, 300}. We adopted entity-description pairs extracted from FB15K [42], a dataset extracted from a typical large-scale KG Freebase [117], as the training data of KGWE.

To guarantee that every entity in the training data relates to a description, 47 entities whose descriptions contain less than three words were removed from FB15K [54], resulting in the training data used in our experiment (see the statistics of the training data in Table 5.9). Since the longest description in the training data contains 343 words, we restricted the length of the descriptions to be 20. For any description containing more than 20 words, we intercepted only the first 20 ones from it. After preprocessing, there were 14,904 entities and 25,985 words in our training set.

Table 5.7: Statistical information of the lexicons used by Retrofitting

		Total	FrameNet					PPDB						WN_{syn}/WN_{all}					
		Pairs	Non OOV		Half OOV		Both OOV		Non OOV		Half OOV		Both OOV		Non OOV		Half OOV		Both OOV	
WS353	all	352	139	39.5%	134	38.1%	79	22.4%	330	93.8%	16	4.5%	6	1.7%	345	98%	7	2%	0	0%
	sim	203	87	42.9%	70	34.5%	46	22.7%	193	95.1%	6	3%	4	2%	199	98%	4	2%	0	0%
	rel	252	98	38.9%	104	41.3%	50	19.8%	237	94%	13	5.2%	2	0.8%	246	97.6%	6	2.4%	0	0%
MEN	full	3000	1173	39.1%	1270	42.3%	557	18.6%	2868	95.6%	126	4.2%	6	0.2%	2985	99.5%	15	0.5%	0	0%
	dev	2000	790	39.5%	831	41.6%	379	19%	1909	95.5%	87	4.4%	4	0.2%	1988	99.4%	12	0.6%	0	0%
	test	1000	383	38.3%	439	43.9%	178	17.8%	959	95.9%	39	3.9%	2	0.2%	997	99.7%	3	0.3%	0	0%
RMTurk		287	43	15%	131	45.6%	113	39.4%	266	92.7%	20	7%	1	0.3%	169	58.9%	96	33.4%	22	7.7%
RareWords		2034	112	5.5%	1066	52.4%	856	42.1%	965	47.4%	947	46.6%	122	6%	1376	67.6%	646	31.8%	12	0.6%
SimLex999		999	571	57.2%	318	31.8%	110	11%	993	99.4%	6	0.6%	0	0%	999	100%	0	0%	0	0%
SimVerb3500		3500	2743	78.4%	697	19.9%	60	1.7%	3337	95.3%	157	4.5%	6	0.2%	3500	100%	0	0%	0	0%
SE17T2	trial	18	8	44.4%	5	27.8%	5	27.8%	16	88.9%	1	5.6%	1	5.6%	18	100%	0	0%	0	0%
	test	500	108	21.6%	193	38.6%	199	39.8%	361	72.2%	73	14.6%	66	13.2%	416	83.2%	59	11.8%	25	5%
BakerVerb143		144	20	13.9%	64	44.4%	60	41.7%	144	100%	0	0%	0	0%	57	39.6%	58	40.3%	29	20.1%
YPVerb130		130	99	76.2%	25	19.2%	6	4.6%	122	93.8%	8	6.2%	0	0%	130	100%	0	0%	0	0%

Table 5.8: Experiment results on relation classification

		GloVe.6B.50d		GloVe.6B.100d		GloVe.6B.200d		GloVe.6B.300d		Word2Vec.300d	
		Accuracy	M-A F1	Accuracy	M-A F1	Accuracy	M-A F1	Accuracy	M-A F1	Accuracy	M-A F1
	vanilla	0.6732	0.6313	0.7265	0.6882	0.7276	0.6989	0.7276	0.6926	0.7619	0.7319
Retrofitting	FrameNet	0.6743	0.6459	0.7081	0.6766	0.7221	0.6973	0.6827	0.653	0.6827	0.653
	PPDB-xl	0.6827	0.6525	0.7232	0.6798	0.7368	0.6992	0.7033	0.6641	0.7033	0.6641
	$WordNet_{syn}$	0.671	0.6445	0.7229	0.679	0.7122	0.6864	0.6805	0.6509	0.6805	0.6509
	$WordNet_{all}$	0.6676	0.6316	0.7247	0.6944	0.7317	0.6913	0.6879	0.6624	0.6879	0.6624
KGWE	ANALOGY	0.7111	0.6832	**0.7494**	0.7182	0.7471	0.717	0.7501	0.7186	0.781	0.751
	DistMult	**0.7137**	0.6811	0.7494	0.7191	0.7449	0.7166	0.7512	0.7211	0.7803	0.7491
	TransD	0.707	0.6817	0.7483	0.715	0.7446	0.715	0.7449	0.7213	0.7795	0.7518
	TransE	0.7103	0.6901	0.7471	0.7137	0.7453	**0.7258**	0.7512	0.7218	0.7781	**0.7576**
	TransH	0.7074	**0.6913**	0.7483	**0.7197**	**0.7486**	0.7218	**0.7519**	**0.7261**	**0.7869**	0.7552

Table 5.9: Statistics of the training data

Dataset	Relation	Entity	Train	Valid	Test
FB15K	1,341	14,904	472,860	48,991	57,803

Regarding the training, we adopted the mean squared error as the loss function as discussed in Section 5.3 and optimized using the stochastic gradient descent. The learning rate of the optimizer was set to 0.0001. For the RNN-based encoder (cf. Equation 5.7), we adopted GRU with one recurrent layer whose weights were initialized randomly via the uniform distribution. For the BoW-based encoder (cf. Equation 5.8), we assigned an equal weight (**1**) to all words. Our model with a BoW-based encoder can achieve the best performance within 500 training epochs for most combinations of word embeddings and entity embeddings. We also compared the performance with Retrofitting[2][135] using four different semantic lexicons: FrameNet[156], PPDB[157], $WordNet_{syn}$ [158] and $WordNet_{all}$[158], with the

[2]https://github.com/mfaruqui/retrofitting

default configuration reported in [135].

5.4.2 Knowledge Graph Embedding Models

This subsection explains each knowledge graph embedding model (KGEM) used in our model for its self-containment.

ANALOGY is a bilinear model which represents each entity as a vector and each relation as a matrix to capture their latent semantics. The relation matrix models pairwise interactions between latent factors. The score of a triple (h, r, t) is defined by a bilinear function: $f_r(h, t) = \mathbf{h}^\top \mathbf{M}_r \mathbf{t}$, where $\mathbf{h}, \mathbf{t} \in R^d$ are embeddings of head entity and tail entity, and $\mathbf{M}_r \in R^{d \times d}$ is a linear map associated with the relation.

DistMult is a simplified bilinear model by restricting $\mathbf{M}_r$ to diagonal matrices. For each relation r, it introduces a vector embedding $\mathbf{r} \in R^d$ and requires $\mathbf{M}_r = \text{diag}(\mathbf{r})$. The scoring function is hence defined as:

$$f_r(h, t) \coloneqq \mathbf{h}^\top \text{diag}(\mathbf{r})\mathbf{t} = \sum_{i=0}^{d-1} [\mathbf{r}]_i \cdot [\mathbf{h}]_i \cdot [\mathbf{t}]_i. \tag{5.9}$$

This score captures pairwise interactions between only the components of $\mathbf{h}$ and $\mathbf{t}$ along the same dimension. However, DistMult can only deal with symmetric relations which is not powerful enough for general knowledge graphs.

TransE is the most representative translational distance model. It learns both entity embeddings and relation embeddings in the same space. Given a triple (h, r, t), the relation is interpreted as a translation vector r so that the embedded head entity h and tail entity t can be connected by r with low error, i.e., $h + r \approx t$. TransE is simple and efficient, but weak in dealing with 1-to-N, N-to-1, and N-to-N relations.

TransH introduces relation-specific hyperplanes to overcome the disadvantages of TransE in dealing with 1-to-N, N-to-1, and N-to-N relations. TransH models each relation r as a vector $\mathbf{r}$ on a hyperplane with $\mathbf{w}_r$ as the normal vector. For a triple (h, r, t), the entity representations h and $\mathbf{t}$ are first projected onto the hyperplane, resulting in

$$\mathbf{h}_\perp = \mathbf{h} - \mathbf{w}_r^\top \mathbf{h} \mathbf{w}_r, \quad \mathbf{t}_\perp = \mathbf{t} - \mathbf{w}_r^\top \mathbf{t} \mathbf{w}_r \tag{5.10}$$

Each projection is assumed to be connected by $\mathbf{r}$ on the hyperplane with a low error, i.e., $\mathbf{h}_\perp + \mathbf{r} \approx \mathbf{t}_\perp$. By introducing such projections to relation-specific hyperplanes, TransH can enable different roles of an entity in different relations.

TransD introduces relation-specific spaces to deal with 1-to-N, N-to-1, and N-to-N relations. For a triple (h, r, t), TransD introduces additional

mapping vectors $\mathbf{w}_h, \mathbf{w}_t \in R^d$ and $\mathbf{w}_r \in R^k$ for $\mathbf{h}, \mathbf{t} \in R^d$ and $\mathbf{r} \in R^k$. The projections of $\mathbf{h}$ and $\mathbf{t}$ are

$$\mathbf{h}_\perp = \left(\mathbf{w}_r \mathbf{w}_h^\top + \mathbf{I}\right) \mathbf{h}, \quad \mathbf{t}_\perp = \left(\mathbf{w}_r \mathbf{w}_t^\top + \mathbf{I}\right) \mathbf{t} \tag{5.11}$$

The scoring function is defined as

$$f_r(h,t) \coloneqq - \left\|\mathbf{h}_\perp + \mathbf{r} - \mathbf{t}_\perp\right\|_2^2 \tag{5.12}$$

TransD is powerful in modeling complex relations, but loses the simplicity and efficiency of TransE and TransH.

5.4.3 Word Similarity Task

We evaluated the embeddings fine-tuned word by KGWE with the BoW-based encoder on the word similarity task. We adopted the Spearman's rank correlation coefficient between the cosine similarity of the fine-tuned word embedding pairs and human judgments as the evaluation metric. Suppose x be the cosine similarity scores of the fine-tuned word embeddings and y be the human judgments, the Spearman's rank correlation coefficient $\rho_s(x,y)$ is:

$$\rho_s(x,y) = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \tag{5.13}$$

The closer the score is to 1, the closer the embeddings' performance is to human judgment. We performed our evaluations on WordSim353 [159, 160], MEN [161], RadinskyMTurk [162], RareWords [163], SimLex999 [164], SimVerb3500 [164], SemEval17Task2 [165], BakerVerb143 [166] and Yang-Powers130 [167]. We adopted the GluonNLP[3] [168] toolkit to evaluate the fine-tuned word embeddings.

Note that WordSim353 is a widely used word similarity benchmark, containing 353 word pairs, each associated with an average of 13 to 16 human judgments. MEN contains 3,000 pairs of randomly selected words that occur as ESP tags (pairs sampled to ensure a balanced range of relatedness levels according to a text-based semantic score), each pair is scored on a [0, 1]-normalized semantic relatedness scale. RadinskyMTurk contains 287 word pairs extracted from New York Times news articles, each pair's similarity score is obtained by using the Amazon's Mechanical Turk workers. RareWords contains 2034 pairs of low-frequency words. SimLex-999 is a benchmark for evaluating the meaning of words and concepts. SimLex-999 focuses on measuring how well models capture similarity, rather than

[3]https://nlp.gluon.ai/index.html

relatedness or association. SemEval17Task2 provides a reliable framework for evaluating both monolingual and multilingual semantic representations, and similarity techniques. In our experiment, we only adopted the English monolingual word pairs. SimVerb-3500 provides human ratings for the similarity of 3,500 verb pairs, covering all normed verb types from the USF free-association database, providing at least three examples for every VerbNet class. BakerVerb consists of 143 verb pairs, constructed from 122 unique verb lemma types. The participating verbs appear ≥ 10 times in the concatenation of the labour legislation and the environment datasets. Only pairs of verbs that were considered at least remotely similar by human judges (independent of those that provided the similarity scores) were included. YangPowersVerb contains 130 verb pairs extracted from TOEFL (Test of English as a Foreign Language) questions, and ESL (English as a second language) questions.

5.4.4 Results on Word Similarity Task

The results of GloVe (dimension = 50, 100, 200, 300) and Word2Vec (dimension = 300) fine-tuned by the KGWE model with a BoW-based description encoder on the word similarity task are shown in Tables 5.1, 5.2, 5.3, 5.4 and 5.5, respectively. The boldface numbers represent the best results on the tasks. For 50-dimensional, 100-dimensional and 200-dimensional GloVe, KGWE outperforms GloVe and Retrofitting on WordSim353, RareWords and SemEval17Task2, but gains no improvement on other benchmarks than GloVe. For 300-dimensional GloVe, KGWE outperforms Retrofitting on all benchmarks and gains improvement on MEN, RadinskyMTurk, RareWords, and SemEval17Task2 than GloVe. For 300-dimensional Word2Vec, KGWE outperforms Retrofitting on all benchmarks and gains improvement on MEN, RadinskyMTurk, RareWords, SemEval17Task2, and BakerVerb143 than Word2Vec.

In addition, KGWE achieves very little performance improvement on the three verb similarity benchmarks: SimVerb3500, BakerVerb143, and YangPowersVerb130. Based on our analysis, we hypothesize that this is because we only adopt entity embeddings but drop the relation embeddings. Hence, we lose the information contained in relational embedding. Figure 5.4 shows the Spearman Rank Correlation curve during training, demonstrating that the performance of KGWE is stable. Indeed, it shows to gradually improve the word embedding. Even if it cannot improve the performance on word embedding, it does not destroy the word representation like existing fine-tuning methods such as Retrofitting.

As limited by the scale of our training data, the word embeddings we fine-tuned are relatively less, which has a great impact on the performance

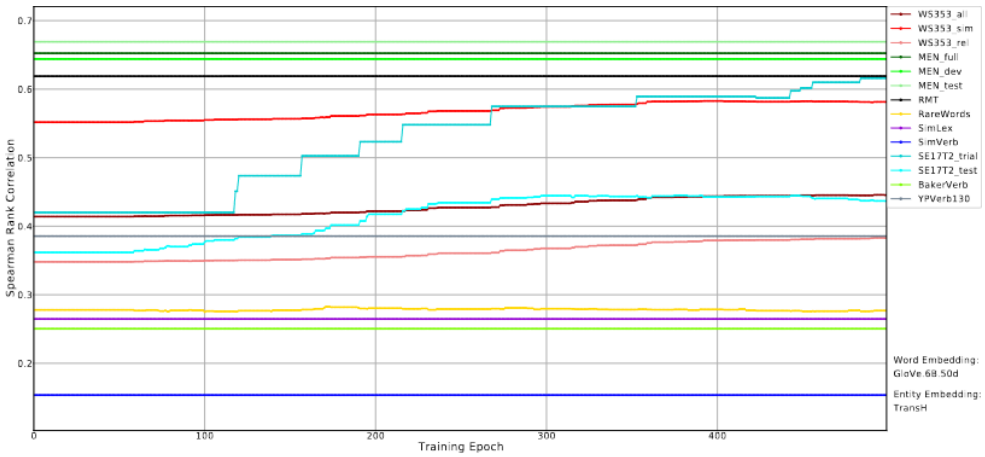


Figure 5.4: Performance on each word similarity task during the training

Table 5.10: Word similarity result of KGWE with a RNN-based description encoder on GloVe.6B.50d

	MEN			Rare	SE17T2		YP
	full	dev	test	Words	trial	test	Verb
GloVe.6B.50d	0.652	0.644	0.669	0.266	0.414	0.362	0.386
KGWE$_{RNN}$ ANALOGY	**0.653**	0.644	0.669	**0.278**	**0.42**	0.362	**0.388**

of KGWE. We divide the word pairs into three categories according to the number of the OOV (out of vocabulary) words involved in the word pair: Non OOV, Half OOV, Both OOV. We count the number and proportion of these three types of word pairs in the training data of KGWE and Retrofitting, as shown in Tables 5.6 and 5.7, respectively. The tables reveal that FrameNet covers the least word pairs, while the performance of Retrofitting with FrameNet is also the worst. Although PPDB and WN$_{syn}$/WN$_{all}$ have much better coverage of word pairs on all benchmarks than KGWE, KGWE can achieve similar performance to Retrofitting with PPDB and WN$_{syn}$/WN$_{all}$ on GloVe (50d, 100d, 200d), and better performance than Retrofitting on 300-dimensional GloVe and Word2Vec.

As for KGWE with the RNN-based description encoder, the fine-tuned word embeddings cannot gain any performance improvement on the word similarity tasks. We hypothesize that this is because the RNN-based description encoder is more complex, and the optimizer can make the deviation between entity embedding and entity description representation sufficiently low by optimizing the weights of the RNN-based encoder. We also trained the RNN-based model for 20,000, 30,000, 40,000 epochs and found that the

most significant changes were on the embeddings of the unknown words, padding words, and beginning/ending labels of sentences; all were initialized with zero vectors. The remaining words' embeddings have not been obviously fine-tuned. We did experiment on KGWE with RNN-based encoder whose weight is freezed. The result is shown in Table 5.10, KGWE with RNN-encoder achieves better performance on four word similarity benchmarks, which can prove our hypothesis.

5.4.5 Results on Relation Classification Task

The task of relation classification [169] is to predict a plausible semantic relation between a pair of nouns which can be informally defined as follows: given a sentence S with the annotated pairs of nouns n_1 and n_2, the task is to identify a relation between n_1 and n_2. We use the SemEval-2010 Task 8 dataset to evaluate our proposed model[169], which contains 10,717 annotated examples, including 8,000 training instances and 2,717 test instances. There are 9 relationships (with two directions) and an undirected "Other" class. Note that we did not take the "Other" class into account in our experiment. Rather, we adopt the evaluation implementation in [170][4]. The results are shown in Table 5.8. For all kinds of embeddings, our proposed model can outperform the naive word embeddings and the retrofitting model.

5.4.6 Results on Sentence Level Polarity Classification

The sentence level sentiment polarity classification task is evaluated with the MR dataset of short movie reviews [171]. For this task, only one sentence per review is classified into positive/negative. Binary classification is performed by a simplified version of the model proposed by [172], which is implemented by [170]. The results are shown in Table 5.11, demonstrating that for all kinds of embeddings, our proposed model can outperform the naive word embeddings and retrofitting model.

5.4.7 Case Analysis

We randomly selected an example from the relation classification task and the sentence polarity classification task, and applied KGWE (BoW-based encoder, epoch=250, TransE) on these two examples using glove.6B.50d for their embeddings. The results from each task are shown in Table 5.12 and

[4]https://github.com/shashwath94/Extrinsic-Evaluation-tasks

Table 5.11: Experiment results on sentence level polarity classification

		GloVe.6B				Word2Vec
		50d	100d	200d	300d	300d
vanilla		88.36	88.28	89.92	90.88	90.84
Retrofitting	FrameNet	88.56	88.36	90.36	91.00	86.76
	PPDB-xl	87.00	88.04	90.32	90.72	86.84
	WordNet$_{syn}$	88.20	87.80	90.08	90.44	86.92
	WordNet$_{all}$	87.80	87.56	90.44	90.60	87.08
KGWE	ANALOGY	**89.28**	89.64	91.00	91.16	91.52
	DistMult	89.20	89.56	90.96	91.24	91.60
	TransD	89.12	**89.68**	91.04	91.08	91.40
	TransE	89.16	89.60	**91.16**	**91.28**	**91.64**
	TransH	89.16	89.52	90.88	91.12	91.56

Table 5.12: An example on relation classification

input	context:	Inside of it, the first details about the game were revealed through an interview with series director Masahiro Yasuma.
	entity:	details, interview
label	Message-Topic (e2,e1)	
glove.6B.50d	Other	
KGWE	Message-Topic (e2,e1)	

Table 5.13: An example on sentence polarity classification

input	the documentary will combine an interview with mr. mcnamara discussing some of the tragedies and glories of the 20th century, archival footage, documents, and an original score by philip glass.
label	positive
glove.6B.50d	negative
KGWE	positive

Table 5.13, respectively. We found that KGWE helps to improve the representation of the embedding, yielding the correct classification as desired. Table 5.14 shows the entity-description pair that involves the entity "interview" and the word "interview". We believe that it is the knowledge introduced by KGWE, who has significantly improved the performance of word embeddings.

Table 5.14: The entity-description pair involved with "interview"

entity	interview
id	/m/01jdpf
description	"An interview is a conversation between two or more people where questions are asked by the interviewer to elicit facts or statements from the interviewee. Interviews are a standard part of journalism and media reporting, but are also employed in many other situations, including qualitative research."@en

5.5 Conclusion and Future Works

this section proposes KGWE that utilizes the knowledge from entity embeddings learned from any knowledge graph embedding model and shows to aid in fine-tuning the static word embeddings. In addition, we provide two implementation methods of the entity description encoder in KGWE: the RNN-Based encoder and the BoW-based encoder. We evaluate the proposed model on both of intrinsic evaluation task and extrinsic evaluation, including the word similarity task with various benchmarks, the relation classification task, and the sentence polarity classification task. The results demonstrate that the word embeddings fine-tuned by KGWE with a BoW-based encoder can significantly outperform the baseline word embeddings. In the future, we will explore how to utilize relational embeddings to guide fine-tuning verbs.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has investigated several interesting topics belonging to the category of knowledge graph embedding.

Initially, we analyze the imbalance between learning embeddings on weighted knowledge graphs and evaluating the embeddings with non-weight-aware tasks. We introduce two weight-aware evaluation tasks for evaluating the embeddings learned from weighted knowledge graphs. We propose a framework WaExt to extend the non-weight-aware knowledge graph embedding models to their weight-aware version, achieving better performance on both weight-aware tasks and non-weight-aware tasks.

Secondly, we notice that the weight information is critically important to the weighted knowledge graphs and explore how to encode the weight information of the triples to the knowledge graph embeddings. We propose a framework WeExt to extend the unweighted knowledge graph embedding models to their weighted version. WeExt can well take into account the performance of knowledge graph embeddings on the link prediction task and weight prediction task, resulting in better performance on the weighted link prediction task.

Lastly, we notice the mismatch between the word vector space where the word embedding is located and the knowledge graph vector space where the knowledge graph embedding is located, which brings difficulty to the utilization of knowledge graph embeddings on the downstream tasks. We propose a model that uses knowledge graph embeddings to guide fine-tuning of static word embeddings, improving the quality of the word embeddings.

6.2 Future Work

In the future, we will explore ways to efficiently encode timestamps and time intervals of triples in temporal knowledge graphs. Most existing models embed time as discrete items, which results in significant information loss, such as the inability to intuitively compare the order of events. To address this issue, we plan to propose a sequence-to-sequence model.

Publications

1. **Kong W. K.**, Racharak T. and Nguyen M. L., "Can Knowledge Enhance Reading Comprehension? An Integrated Approach with Semantic Lexicon," 12th International Conference on Knowledge and Systems Engineering (KSE), 2020, pp. 7-12, doi: 10.1109/KSE50997.2020.9287218.

2. **Kong W. K.**, Racharak T., Cao. Y., Peng C. and Nguyen M. L., "KGWE: A Knowledge-guided Word Embedding Fine-tuning Model," 2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI), 2021, pp. 1221-1225, doi: 10.1109/ICTAI52525.2021.00193.

3. **Kong, W.K.**, Zheng, S., Nguyen M.L., Ma, Q., "A Multi-agent Reinforcement Learning Approach Towards Congestion-aware Route Recommendation for Tourists," 14th Data Engineering and Information Management Forum (DEIM), 2022, D24-5.

4. **Kong, W.K.**, Zheng, S., Nguyen M.L., Ma, Q., "Diversity-Oriented Route Planning for Tourists," 33rd International Conference on Database and Expert Systems Applications (DEXA), 2022, doi: 10.1007/978-3-031-12426-6_20

5. **Kong W. K.**, Liu X., Racharak T. and Nguyen M. L., "TransHExt: a Weighted Extension for TransH on Weighted Knowledge Graph Embedding," The 21st International Semantic Web Conference (ISWC), 2022.

6. **Kong W. K.**, Liu X., Racharak T., Sun. G., Ma Q. and Nguyen M. L., "Weight-aware Tasks for Evaluating Knowledge Graph Embeddings," The Semantic Web journal, Under Review.

7. **Kong W. K.**, Liu X., Racharak T., Sun. G., Ma Q. and Nguyen M. L., "WeExt: Extending Deterministic Knowledge Graph Embedding Models for Embedding Weighted Knowledge Graphs," IEEE ACCESS, Revising.

Bibliography

[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*, vol. 6, pp. 167–195, 2015.

[2] T. M. Mitchell, W. W. Cohen, E. 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*, vol. 61, pp. 103 – 115, 2015.

[3] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," *ArXiv*, vol. abs/1612.03975, 2017.

[4] F. M. Suchanek, G. Kasneci, and G. Weikum, "Yago: a core of semantic knowledge," in *WWW '07*, 2007.

[5] K. D. Bollacker, C. Evans, P. K. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *SIGMOD Conference*, ser. SIGMOD '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 1247–1250.

[6] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, and C. von Mering, "String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Research*, vol. 47, pp. D607 – D613, 2019.

[7] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: a probabilistic taxonomy for text understanding," *SIGMOD*, 2012.

[8] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition, and applications," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, pp. 494–514, 2020.

[9] 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*, vol. 29, pp. 2724–2743, 2017.

[10] X. Zou, "A survey on application of knowledge graph," in *Journal of Physics: Conference Series*, vol. 1487, no. 1. IOP Publishing, 2020, p. 012016.

[11] A. Fader, L. Zettlemoyer, and O. Etzioni, "Open question answering over curated and extracted knowledge bases," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 1156–1165.

[12] X. Yao and B. Van Durme, "Information extraction over structured data: Question answering with freebase," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 956–966.

[13] A. Bordes, S. Chopra, and J. Weston, "Question answering with subgraph embeddings," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 615–620.

[14] A. Bordes, J. Weston, and N. Usunier, "Open question answering with weakly supervised embedding models," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2014, pp. 165–180.

[15] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 221–231.

[16] H. Wang, F. Zhang, X. Xie, and M. Guo, "Dkn: Deep knowledge-aware network for news recommendation," in *Proceedings of the 2018 world wide web conference*, 2018, pp. 1835–1844.

[17] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 353–362.

[18] V. Bellini, V. W. Anelli, T. Di Noia, and E. Di Sciascio, "Auto-encoding user ratings via knowledge graphs in recommendation scenarios," in *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, 2017, pp. 60–66.

[19] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *The world wide web conference*, 2019, pp. 2000–2010.

[20] H. Zhao, Q. Yao, J. Li, Y. Song, and D. L. Lee, "Meta-graph based recommendation fusion over heterogeneous information networks," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 635–644.

[21] J. Dalton, L. Dietz, and J. Allan, "Entity query feature expansion using knowledge base links," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, 2014, pp. 365–374.

[22] H. Raviv, O. Kurland, and D. Carmel, "Document retrieval using entity-based language models," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 65–74.

[23] F. Ensan and E. Bagheri, "Document retrieval model through semantic linking," in *Proceedings of the tenth ACM international conference on web search and data mining*, 2017, pp. 181–190.

[24] X. Liu and H. Fang, "Latent entity space: a novel retrieval approach for entity-bearing queries," *Information Retrieval Journal*, vol. 18, no. 6, pp. 473–503, 2015.

[25] P. Ernst, C. Meng, A. Siu, and G. Weikum, "Knowlife: a knowledge graph for health and life sciences," in *2014 IEEE 30th International Conference on Data Engineering*. IEEE, 2014, pp. 1254–1257.

[26] L. Shi, S. Li, X. Yang, J. Qi, G. Pan, and B. Zhou, "Semantic health knowledge graph: semantic integration of heterogeneous medical knowledge and services," *BioMed research international*, vol. 2017, 2017.

[27] Y. Jia, Y. Qi, H. Shang, R. Jiang, and A. Li, "A practical approach to constructing a knowledge graph for cybersecurity," *Engineering*, vol. 4, no. 1, pp. 53–60, 2018.

[28] Y. Qi, R. Jiang, Y. Jia, R. Li, and A. Li, "Association analysis algorithm based on knowledge graph for space-ground integrated network," in *2018 IEEE 18th International Conference on Communication Technology (ICCT)*. IEEE, 2018, pp. 222–226.

[29] S. A. Elnagdy, M. Qiu, and K. Gai, "Understanding taxonomy of cyber risks for cybersecurity insurance of financial industry in cloud computing," in *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2016, pp. 295–300.

[30] ——, "Cyber incident classifications using ontology-based knowledge representation for cybersecurity insurance in financial industry," in *2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2016, pp. 301–306.

[31] J. Liu, Z. Lu, and W. DU, "Combining enterprise knowledge graph and news sentiment analysis for stock price prediction," in *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 2019.

[32] B. Ulicny, "Constructing knowledge graphs with trust," in *4th International Workshop on Methods for Establishing Trust of (Open) Data, Bentlehem, USA*, 2015.

[33] P. Chen, Y. Lu, V. W. Zheng, X. Chen, and B. Yang, "Knowedu: A system to construct knowledge graph for education," *Ieee Access*, vol. 6, pp. 31 553–31 563, 2018.

[34] C. Grévisse, R. Manrique, O. Mariño, and S. Rothkugel, "Knowledge graph-based teacher support for learning material authoring," in *Colombian Conference on Computing*. Springer, 2018, pp. 177–191.

[35] A. Singhal, "Introducing the knowledge graph: Things, not strings," May 2012. [Online]. Available: https://blog.google/products/search/introducing-knowledge-graph-things-not/

[36] X. Luo, L. Liu, Y. Yang, L. Bo, Y. Cao, J. Wu, Q. Li, K. Yang, and K. Q. Zhu, "Alicoco: Alibaba e-commerce cognitive concept net," in *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, 2020, pp. 313–327.

[37] H. Qi, "Building the linkedin knowledge graph," October 2016. [Online]. Available: https://engineering.linkedin.com/blog/2016/10/building-the-linkedin-knowledge-graph

[38] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in neural information processing systems*, vol. 26, 2013.

[39] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. v. d. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European semantic web conference*. Springer, 2018, pp. 593–607.

[40] J. Li, H. Shomer, J. Ding, Y. Wang, Y. Ma, N. Shah, J. Tang, and D. Yin, "Are graph neural networks really helpful for knowledge graph completion?" *arXiv preprint arXiv:2205.10652*, 2022.

[41] Z. Sun, W. Hu, Q. Zhang, and Y. Qu, "Bootstrapping entity alignment with knowledge graph embedding." in *IJCAI*, vol. 18, 2018, pp. 4396–4402.

[42] A. Bordes, N. Usunier, A. García-Durán, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *NIPS*, 2013.

[43] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, 2014.

[44] Y. Lin, Z. Liu, M. Sun, Y. Liu, and X. Zhu, "Learning entity and relation embeddings for knowledge graph completion," in *AAAI*, 2015.

[45] M. Nickel, V. Tresp, and H.-P. Kriegel, "A three-way model for collective learning on multi-relational data," in *ICML*, 2011.

[46] B. Yang, W. tau Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *CoRR*, vol. abs/1412.6575, 2015.

[47] M. Nickel, L. Rosasco, and T. A. Poggio, "Holographic embeddings of knowledge graphs," in *AAAI*, 2016.

[48] R. Hussein, D. Yang, and P. Cudré-Mauroux, "Are meta-paths necessary? revisiting heterogeneous graph embeddings," in *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp. 437–446.

[49] S. Guo, Q. Wang, B. Wang, L. Wang, and L. Guo, "Semantically smooth knowledge graph embedding," in *Annual Meeting of the Association for Computational Linguistics*, 2015.

[50] R. Xie, Z. Liu, and M. Sun, "Representation learning of knowledge graphs with hierarchical types," in *International Joint Conference on Artificial Intelligence*, vol. 2016, 2016, pp. 2965–2971.

[51] D. Krompass, S. Baier, and V. Tresp, "Type-constrained representation learning in knowledge graphs," in *International Workshop on the Semantic Web*. Springer, 2015, pp. 640–655.

[52] R. Socher, D. Chen, C. D. Manning, and A. Ng, "Reasoning with neural tensor networks for knowledge base completion," in *NIPS*, 2013.

[53] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph and text jointly embedding," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

[54] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *AAAI*, 2016.

[55] Z. Wang and J.-Z. Li, "Text-enhanced representation learning for knowledge graph," in *International Joint Conference on Artificial Intelligence*, 2016.

[56] A. Kimmig, S. H. Bach, M. Broecheler, B. Huang, and L. Getoor, "A short introduction to probabilistic soft logic," in *NIPS*, 2012.

[57] S. Guo, Q. Wang, L. Wang, B. Wang, and L. Guo, "Jointly embedding knowledge graphs and logical rules," in *Conference on Empirical Methods in Natural Language Processing*, 2016.

[58] T. Rocktäschel, S. Singh, and S. Riedel, "Injecting logical background knowledge into embeddings for relation extraction," in *North American Chapter of the Association for Computational Linguistics*, 2015.

[59] P. Hájek, *Metamathematics of fuzzy logic*. Springer Science & Business Media, 2013, vol. 4.

[60] X. Chen, M. Chen, W. Shi, Y. Sun, and C. Zaniolo, "Embedding uncertain knowledge graphs," in *AAAI*, 2019.

[61] 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.

[62] S. Pai and L. Costabello, "Learning embeddings from knowledge graphs with numeric edge attributes," in *IJCAI*, 2021.

[63] B. Russell, "Knowledge by acquaintance and knowledge by description," in *Proceedings of the Aristotelian society*, vol. 11. JSTOR, 1910, pp. 108–128.

[64] J. J. Ichikawa and M. Steup, "The Analysis of Knowledge," in *The Stanford Encyclopedia of Philosophy*, Summer 2018 ed., E. N. Zalta, Ed. Metaphysics Research Lab, Stanford University, 2018.

[65] B. Russell, *The problems of philosophy*. OUP Oxford, 2001.

[66] E. H. Shortliffe, B. G. Buchanan, and E. A. Feigenbaum, "Knowledge engineering for medical decision making: A review of computer-based clinical decision aids," *Proceedings of the IEEE*, vol. 67, no. 9, pp. 1207–1224, 1979.

[67] L. M. Markus, "Toward a theory of knowledge reuse: Types of knowledge reuse situations and factors in reuse success," *Journal of management information systems*, vol. 18, no. 1, pp. 57–93, 2001.

[68] A. Newell, "The knowledge level," *Artificial intelligence*, vol. 18, no. 1, pp. 87–127, 1982.

[69] C. Pavese, "Knowledge How," in *The Stanford Encyclopedia of Philosophy*, Fall 2022 ed., E. N. Zalta and U. Nodelman, Eds. Metaphysics Research Lab, Stanford University, 2022.

[70] R. Studer, V. R. Benjamins, and D. Fensel, "Knowledge engineering: principles and methods," *Data & knowledge engineering*, vol. 25, no. 1-2, pp. 161–197, 1998.

[71] D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," *Annual review of biomedical engineering*, vol. 19, p. 221, 2017.

[72] R. Angles and C. Gutierrez, "Survey of graph database models," *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1–39, 2008.

[73] R. Angles, M. Arenas, P. Barceló, A. Hogan, J. Reutter, and D. Vrgoč, "Foundations of modern query languages for graph databases," *ACM Computing Surveys (CSUR)*, vol. 50, no. 5, pp. 1–40, 2017.

[74] E. A. Bender and S. G. Williamson, *Lists, decisions and graphs.* S. Gill Williamson, 2010.

[75] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *AAAI*, 2017.

[76] 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*, vol. 61, pp. 103 – 115, 2015.

[77] O. D. la Cruz Cabrera, M. Matar, and L. Reichel, "Edge importance in a network via line graphs and the matrix exponential," *Numerical Algorithms*, vol. 83, pp. 807–832, 2019.

[78] "Introduction to knowledge-based systems," in *Proceedings Electronic Technology Directions to the Year 2000*, 1995, pp. 18–27.

[79] R. Davis, H. Shrobe, and P. Szolovits, "What is a knowledge representation?" *AI magazine*, vol. 14, no. 1, pp. 17–17, 1993.

[80] F. Baader, "Logic-based knowledge representation," in *Artificial intelligence today.* Springer, 1999, pp. 13–41.

[81] L. A. Zadeh, "Knowledge representation in fuzzy logic," in *An introduction to fuzzy logic applications in intelligent systems.* Springer, 1992, pp. 1–25.

[82] L. Steels, "Frame-based knowledge representation," 1978.

[83] R. Davis, B. Buchanan, and E. Shortliffe, "Production rules as a representation for a knowledge-based consultation program," *Artificial intelligence*, vol. 8, no. 1, pp. 15–45, 1977.

[84] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[85] B. An, B. Chen, X. Han, and L. Sun, "Accurate text-enhanced knowledge graph representation learning," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 745–755.

[86] R. Xie, Z. Liu, J. Jia, H. Luan, and M. Sun, "Representation learning of knowledge graphs with entity descriptions," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[87] Z. Wang, J. Li, Z. Liu, and J. Tang, "Text-enhanced representation learning for knowledge graph," in *Proceedings of International joint conference on artificial intelligence (IJCAI)*, 2016, pp. 4–17.

[88] T. P. Tanon, G. Weikum, and F. M. Suchanek, "Yago 4: A reason-able knowledge base," *The Semantic Web*, vol. 12123, pp. 583 – 596, 2020.

[89] D. Vrandeić and M. Krötzsch, "Wikidata: a free collaborative knowledgebase," *Commun. ACM*, vol. 57, pp. 78–85, 2014.

[90] 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.

[91] 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.

[92] 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*, vol. 30, pp. 824–837, 2018.

[93] 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.

[94] 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.

[95] 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*, vol. 34, pp. 3549–3568, 2022.

[96] 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.

[97] 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*, pp. 1254–1257, 2014.

[98] B. Taskar, M. F. Wong, P. Abbeel, and D. Koller, "Link prediction in relational data," in *NIPS*, 2003.

[99] 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*, vol. 43, pp. D447 – D452, 2015.

[100] Z. Wang, H. Wang, J.-R. Wen, and Y. Xiao, "An inference approach to basic level of categorization," *CIKM*, 2015.

[101] Y. Wang, H. Li, H. Wang, and K. Q. Zhu, "Concept-based web search," in *ER*, 2012.

[102] J. Ivanic, A. Wallqvist, and J. Reifman, "Evidence of probabilistic behaviour in protein interaction networks," *BMC Systems Biology*, vol. 2, pp. 11 – 11, 2007.

[103] 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 *ICLR*, 2015.

[104] 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.

[105] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Soc. Networks*, vol. 25, pp. 211–230, 2003.

[106] H. Cho and Y. Yu, "Link prediction for interdisciplinary collaboration via co-authorship network," *Social Network Analysis and Mining*, vol. 8, pp. 1–12, 2018.

[107] 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.

[108] A. Bordes, J. Weston, R. Collobert, and Y. Bengio, "Learning structured embeddings of knowledge bases," in *AAAI*, 2011.

[109] E. M. Voorhees, "The trec-8 question answering track," *Natural Language Engineering*, vol. 7, pp. 361 – 378, 2000.

[110] 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.

[111] T. Dettmers, P. Minervini, P. Stenetorp, and S. Riedel, "Convolutional 2d knowledge graph embeddings," in *AAAI*, 2018.

[112] 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.*, vol. 22, pp. 82:1–82:6, 2021.

[113] L. Costabello, A. Bernardi, A. Janik, S. Pai, C. L. Van, R. McGrath, N. McCarthy, and P. Tabacof, "AmpliGraph: a Library for Representation Learning on Knowledge Graphs," Mar. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.2595043

[114] C. Xiong, R. Power, and J. Callan, "Explicit semantic ranking for academic search via knowledge graph embedding," in *WWW*, 2017.

[115] Z. Wang, T. Chen, J. S. J. Ren, W. Yu, H. Cheng, and L. Lin, "Deep reasoning with knowledge graph for social relationship understanding," in *IJCAI*, 2018.

[116] X. Lin, Z. Quan, Z.-J. Wang, T. Ma, and X. Zeng, "Kgnn: Knowledge graph neural network for drug-drug interaction prediction," in *IJCAI*, 2020.

[117] W. Wu, H. Li, H. Wang, and K. Q. Zhu, "Probase: a probabilistic taxonomy for text understanding," 2012.

[118] J. Lehmann, R. Isele, M. Jakob, A. Jentzsch, D. Kontokostas, P. N. Mendes, S. Hellmann, M. Morsey, P. Van Kleef, S. Auer *et al.*, "Dbpedia – a large-scale, multilingual knowledge base extracted from wikipedia," *Semantic web*, vol. 6, no. 2, pp. 167–195, 2015.

[119] F. Mahdisoltani, J. Biega, and F. Suchanek, "Yago3: A knowledge base from multilingual wikipedias," in *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

[120] R. Speer, J. Chin, and C. Havasi, "Conceptnet 5.5: An open multilingual graph of general knowledge," in *Thirty-first AAAI conference on artificial intelligence*, 2017.

[121] D. Szklarczyk, J. H. Morris, H. Cook, M. Kuhn, S. Wyder, M. Simonovic, A. Santos, N. T. Doncheva, A. Roth, P. Bork *et al.*, "The string database in 2017: quality-controlled protein–protein association networks, made broadly accessible," *Nucleic acids research*, p. gkw937, 2016.

[122] Z. Wang, J. Zhang, J. Feng, and Z. Chen, "Knowledge graph embedding by translating on hyperplanes," in *AAAI*, 2014.

[123] S. Bach, B. Huang, B. London, and L. Getoor, "Hinge-loss markov random fields: Convex inference for structured prediction," in *UAI*, 2013.

[124] W. K. Kong, X. Liu, T. Racharak, and L.-M. Nguyen, "Transhext: a weighted extension for transh on weighted knowledge graph embedding," in *ISWC*, 2022.

[125] T. A. Plate, "Holographic reduced representations," *IEEE Transactions on Neural networks*, vol. 6, no. 3, pp. 623–641, 1995.

[126] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," in *J. Mach. Learn. Res.*, 2000.

[127] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *ICLR*, 2013.

[128] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *EMNLP*, 2014.

[129] X. Yang and K. Mao, "Task independent fine tuning for word embeddings," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 885–894, 2017.

[130] A. Roy and S. Pan, "Incorporating extra knowledge to enhance word embedding," in *IJCAI*, 2020.

[131] F. Lu, P. Cong, and X. Huang, "Utilizing textual information in knowledge graph embedding: A survey of methods and applications," *IEEE Access*, vol. 8, pp. 92 072–92 088, 2020.

[132] C. Xu, Y. Bai, J. Bian, B. Gao, G. Wang, X. Liu, and T. Liu, "Rc-net: A general framework for incorporating knowledge into word representations," *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, 2014.

[133] A. Roy, Y. Park, and S. Pan, "Predicting malware attributes from cybersecurity texts," in *NAACL*, 2019.

[134] J. Tissier, C. Gravier, and A. Habrard, "Dict2Vec: Learning word embeddings using lexical dictionaries," in *EMNLP*, 2017.

[135] M. Faruqui, J. Dodge, S. K. Jauhar, C. Dyer, E. Hovy, and N. A. Smith, "Retrofitting word vectors to semantic lexicons," in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, May–Jun. 2015, pp. 1606–1615. [Online]. Available: https://aclanthology.org/N15-1184

[136] 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*, vol. 29, no. 12, pp. 2724–2743, 2017.

[137] W. Zhou, S. Wang, and C. Jiang, "Knowledge graph embedding with interactive guidance from entity descriptions," *IEEE Access*, vol. 7, pp. 156 686–156 693, 2019.

[138] H. Liu, Y. Wu, and Y. Yang, "Analogical inference for multi-relational embeddings," in *ICML*, 2017.

[139] G. Ji, S. He, L. Xu, K. Liu, and J. Zhao, "Knowledge graph embedding via dynamic mapping matrix," in *ACL*, 2015.

[140] K. W. Kun, T. Racharak, C. Yiming, P. Cheng, and M. Le Nguyen, "Kgwe: A knowledge-guided word embedding fine-tuning model," in *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2021, pp. 1221–1225.

[141] Y. Wang, Y. Hou, W. Che, and T. Liu, "From static to dynamic word representations: a survey," *International Journal of Machine Learning and Cybernetics*, pp. 1–20, 2020.

[142] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," in *NAACL-HLT*, 2018.

[143] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019.

[144] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, and Q. Liu, "Ernie: Enhanced language representation with informative entities," in *ACL*, 2019.

[145] M. E. Peters, M. Neumann, R. Logan, R. Schwartz, V. Joshi, S. Singh, and N. A. Smith, "Knowledge enhanced contextual word representations," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 43–54. [Online]. Available: https://aclanthology.org/D19-1005

[146] X. Wang, T. Gao, Z. Zhu, Z. Zhang, Z. Liu, J. Li, and J. Tang, "Kepler: A unified model for knowledge embedding and pre-trained language representation," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 176–194, 2021.

[147] P. Gupta and M. Jaggi, "Obtaining better static word embeddings using contextual embedding models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, Aug. 2021, pp. 5241–5253. [Online]. Available: https://aclanthology.org/2021.acl-long.408

[148] M. Kaneko and D. Bollegala, "Gender-preserving debiasing for pre-trained word embeddings," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 1641–1650. [Online]. Available: https://aclanthology.org/P19-1160

[149] H. Gonen and Y. Goldberg, "Lipstick on a pig: Debiasing methods cover up systematic gender biases in word embeddings but do not remove them," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 609–614. [Online]. Available: https://aclanthology.org/N19-1061

[150] T. Manzini, L. Yao Chong, A. W. Black, and Y. Tsvetkov, "Black is to criminal as caucasian is to police: Detecting and removing multiclass bias in word embeddings," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 615–621. [Online]. Available: https://aclanthology.org/N19-1062

[151] I. Vulić, S. Ruder, and A. Søgaard, "Are all good word vector spaces isomorphic?" in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3178–3192. [Online]. Available: https://aclanthology.org/2020.emnlp-main.257

[152] P. D. Turney and P. Pantel, "From frequency to meaning: Vector space models of semantics," *ArXiv*, vol. abs/1003.1141, 2010.

[153] Y. Cao, L. Huang, H. Ji, X. Chen, and J.-Z. Li, "Bridge text and knowledge by learning multi-prototype entity mention embedding," in *ACL*, 2017.

[154] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[155] X. Han, S. Cao, L. Xin, Y. Lin, Z. Liu, M. Sun, and J. Li, "OpenKE: An open toolkit for knowledge embedding," in *Proceedings of EMNLP*, 2018.

[156] C. F. Baker, C. Fillmore, and J. Lowe, "The berkeley framenet project," in *COLING-ACL*, 1998.

[157] J. Ganitkevitch, B. V. Durme, and C. Callison-Burch, "PPDB: The paraphrase database," in *HLT-NAACL*, 2013.

[158] G. Miller, "WordNet: a lexical database for english," *Commun. ACM*, vol. 38, pp. 39–41, 1995.

[159] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 406–414.

[160] E. Agirre, E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa, "A study on similarity and relatedness using distributional and WordNet-based approaches," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Boulder, Colorado: Association for Computational Linguistics, Jun. 2009, pp. 19–27.

[161] E. Bruni, G. Boleda, M. Baroni, and N.-K. Tran, "Distributional semantics in technicolor," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 136–145.

[162] K. Radinsky, E. Agichtein, E. Gabrilovich, and S. Markovitch, "A word at a time: Computing word relatedness using temporal semantic analysis," in *Proceedings of the 20th International Conference on World Wide Web*, ser. WWW '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 337–346.

[163] T. Luong, R. Socher, and C. Manning, "Better word representations with recursive neural networks for morphology," in *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*. Sofia, Bulgaria: Association for Computational Linguistics, Aug. 2013, pp. 104–113.

[164] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation," *Computational Linguistics*, vol. 41, pp. 665–695, 2015.

[165] J. Camacho-Collados, M. T. Pilehvar, N. Collier, and R. Navigli, "Semeval-2017 task 2: Multilingual and cross-lingual semantic word similarity," in *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, August 2017, pp. 15–26.

[166] S. Baker, R. Reichart, and A. Korhonen, "An unsupervised model for instance level subcategorization acquisition," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 278–289.

[167] D. Yang and D. Powers, "Verb similarity on the taxonomy of wordnet - dataset," 10 2006.

[168] J. Guo, H. He, T. He, L. Lausen, M. Li, H. Lin, X. Shi, C. Wang, J. Xie, S. Zha, A. Zhang, H. Zhang, Z. Zhang, Z. Zhang, S. Zheng, and Y. Zhu, "GluonCV and GluonNLP: Deep learning in computer vision and natural language processing," *J. Mach. Learn. Res.*, vol. 21, pp. 23:1–23:7, 2020.

[169] I. Hendrickx, S. N. Kim, Z. Kozareva, P. Nakov, D. Ó Séaghdha, S. Padó, M. Pennacchiotti, L. Romano, and S. Szpakowicz, "SemEval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals," in *Proceedings of the 5th International Workshop on Semantic Evaluation*. Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 33–38. [Online]. Available: https://aclanthology.org/S10-1006

[170] A. Rogers, S. Hosur Ananthakrishna, and A. Rumshisky, "What's in your embedding, and how it predicts task performance," in *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2690–2703. [Online]. Available: https://aclanthology.org/C18-1228

[171] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 115–124. [Online]. Available: https://aclanthology.org/P05-1015

[172] Y. Kim, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1746–1751. [Online]. Available: https://aclanthology.org/D14-1181