24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

A Hash Learning Framework for Search-Oriented Knowledge Graph Embedding

Meng Wang[1] and **Tongtong Wu**[1] and **Guilin Qi**[1,2]

Abstract. Knowledge graph representation learning, also called knowledge graph embedding, is the task of mapping entities and relations into a low-dimensional, continuous vector space, and, as a result, can support various machine learning models to perform knowledge completion tasks with good performance and robustness. However, most of existing embedding models focus on improving the link prediction accuracy while ignoring the time-efficiency in search-intensive applications over large-scale knowledge graphs. To tackle this problem, in this paper, we encode knowledge graph into Hamming space and introduce a novel HAsh Learning Framework (HALF) for search-oriented knowledge graph embedding. The proposed method can be applied to recent various knowledge graph embedding models for accelerating the computation of searching embeddings by utilizing the bitwise operations (XNOR and Bitcount). Experimental results on benchmark datasets demonstrate the effectiveness of our proposed method, which gets a bonus of speed-up in the searching embeddings while the accuracy and scalability of the original model are basically maintained.

1 Introduction

Knowledge graph (KG) representation learning, also known as knowledge graph embedding (KGE), is the task of learning the latent representations of entities and relations for knowledge completion, reasoning, and inferencing over a graph-structured knowledge base (e.g., , Freebase [3], Wikidata [26] and DBpedia [2]). This research area has gained massive attention due to the broad range of KG related applications, such as knowledge acquisition, recommender systems and natural language processing.

Existing KGE approaches [30] without extra information fusion can be roughly categorized into three categories: translation-based models, matrix factorization models and neural network models. Translation-based model TransE [4] and its variants (e.g., TransR [20]) represent both entities and relations as vectors in the same space, and consider a relation as the distance vector from a head entity to a tail entity. Matrix factorization models (e.g, RESCAL [24], HolE [23], Analogy [21]) take a KG as a sparse three-dimensional tensor and reduce KGE to a matrix factorization task, and the eigenvectors gained from a factorization are named as the representation of entities and relations respectively. Neural network models (e.g., ConvE [8]) conduct semantic matching using neural network architectures to maximize the probability of true facts.

[1] School of Computer Science and Engineering, Southeast University, Nanjing, China, {meng.wang, wutong8023, gqi}@seu.edu.cn
[2] MOE Key Laboratory of Computer Network and Information Integration, Nanjing, China

Generally, existing KGE models are designed for linking prediction. In other words, if we wonder what is the tail of $< h,\ r,\ ? >$, each entity in the set of candidates which generally are total entities in the KG will be substituted into the scoring function with h, r, ranking to find out which triple gained the highest score. In search-intensive scenarios (e.g., question answering and semantic search over large-scale knowledge graphs), the existing embedding models are enduring high time complexity to search the target entity. In this paper, we propose a HAsh Learning Framework (HALF) for KGE to tackle this problem. As shown in Figure 1, the basic idea of this framework is three-fold: 1) The representation of each entity is a binary vector, which is the so-called hash code or binarized embedding, rather than a real-value embedding in typical models. 2) In predicting period, a model following HALF should generate a binary vector t' with h and r to index the target entity. 3) HALF utilizes the Hamming distance between the index t' and each candidate c for ranking, rather than test every candidate by the scoring function.

Our proposed method HALF can reduce the computational complexity of testing each candidate entity, and accelerate the computation by utilizing the bitwise operations (XNOR and Bitcount) on the learned binary vectors. Specifically, when design HALF, we mainly address the follow two difficulties: 1) Since the scoring functions of different models are various, we propose a universal methodology to modify the calculation of the scoring function and transform most models into HALF way. 2) Since obtaining optimal hash codes for entities is NP-hard due to the binary constraints, we utilize a weighted $SoftSign$ layer to relax the constraints and apply a multi-objective optimization mechanism to the framework training. We conduct experiments on several benchmark datasets and the results show that the proposed HALF can be well applied to the improvement of various models in embedding search scenarios.

The main contributions of this paper can be summarized as follows:

- We study the inefficiency of KGE in search-intensive scenarios and propose a novel hash learning framework, i.e., HALF, to address this problem.
- We analyze the scoring functions of various KGE methods and propose a universal methodology to modify different functions to our HALF way.
- We introduce the idea of Hash learning into KGE and solve the problem of discrete vector learning by a multi-objective optimization mechanism.
- We extensively evaluate HALF by modifying five baseline models and test them in terms of accuracy, scalability and time-efficiency. The results show that HALF get a bonus of speed-up in searching embeddings while the accuracy and scalability of the original model are basically maintained.

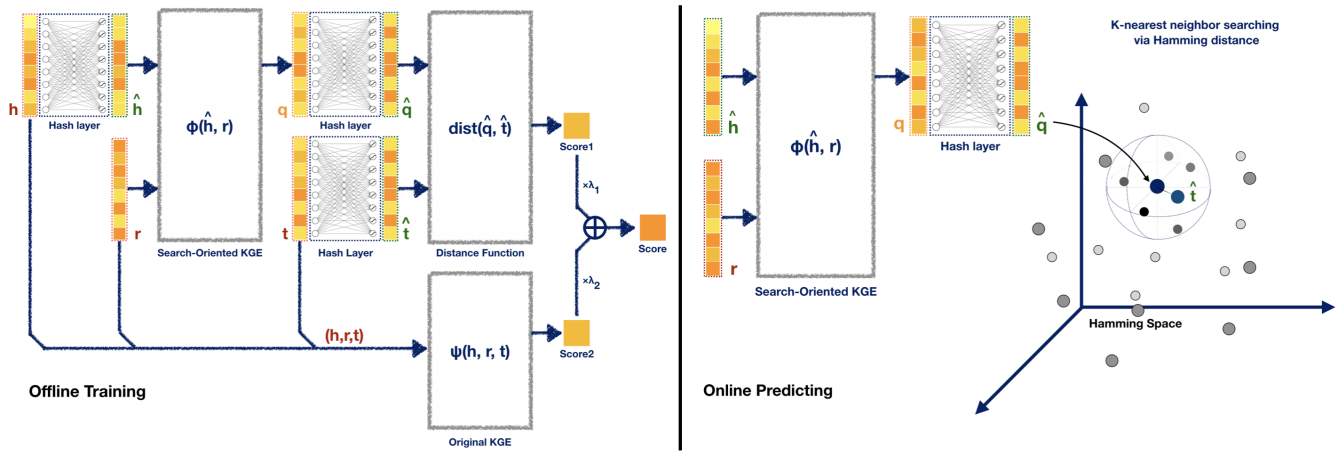

Figure 1. The framework of HALF, where $\mathbf{h}$, $\mathbf{r}$, $\mathbf{t}$ are the embedding of a triple (h, r, t) respectively, and $\hat{\mathbf{h}}$, $\hat{\mathbf{t}}$ represents the binary hash code of entity h and t. In search-oriented KGE model, a query embedding $\mathbf{q}$ is generated by $\phi(\hat{\mathbf{h}}, \mathbf{r})$. Furthermore, we combine the distance between binarized $\hat{\mathbf{q}}$ and $\hat{\mathbf{t}}$, and the score generated by the original KGE model $\psi(\mathbf{h}, \mathbf{r}, \mathbf{t})$ as the overall scoring function. Note that, in order to prevent the zero-gradient problem, we utilize a hash layer with the softsign activation function to approximate during off-line training (as shown in the left part) and utilize sign function during on-line predicting (as shown in the right part). With gained the hashed representation of each entity, thus the link predicting problem is reduced into k-nearest neighbor searching in Hamming space.

2 Preliminary

Let $\mathcal{E}$ denote the set of all entities of size $n = |\mathcal{E}|$ and $\mathcal{R}$ the set of all relations of size $m = |\mathcal{R}|$ present in a KG $\mathcal{G}$. A triple is represented as (h, r, t), with $h, t \in \mathcal{E}$ denoting head and tail entities respectively and $r \in \mathcal{R}$ is the relation between them.

A typical KGE approach generally formalizes the link prediction problem as a point-wise learning to rank problem, where the objective is to learn a scoring function $\psi : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \mapsto \mathbb{R}$. Namely, the score of a triple (h, r, t) is defined as $\psi(h, r, t) \in \mathbb{R}$, which is proportional to the likelihood that the fact encoded by (h, r, t) is true. In Table 1 we summarize the scoring function of several recent KGE models. With this notations, we then define the search-oriented KGE problem.

Definition 1 *(Search-Oriented KGE). Search-oriented KGE is a KGE paradigm based on the concept of "search" which typically consists of queries and candidates. The scoring function of search-oriented KGE is formalized as* $\psi(c, q) = -dist(\mathbf{c} - \phi(q))$*, where* $q \in \mathcal{Q}$ *is the query,* $\mathbf{c} \in \mathbb{R}^d$ *is the embedding of a candidate* $c \in \mathcal{C}$*,* $\phi : \mathcal{Q} \mapsto \mathbb{R}^d$ *is a mapping function to embed* q *in same space with* c*,* $dist : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ *is a distance function.*

According to the definition, TransE [4] is a well-defined model satisfying the setting of Search-Oriented KGE, in which the scoring function can be rewrote as $\psi(h, r, t) = ||\phi(\mathbf{h}, \mathbf{r}) - \mathbf{t}||_2 = Euclid(\phi(\mathbf{h}, \mathbf{r}), \mathbf{t})$, where $Euclid(.)$ means the Euclidean distance function and $\phi(h, r) = \mathbf{h} + \mathbf{r}$.

Note that, in this paper we focus on the meta query which is defined as a corrupted triple $(h, r, ?)$, concerning the following three reasons: 1) Simple questions which involve only one triple (e.g., who is e's tutor?) is the most frequent queries in search engines or knowledge question answering systems. 2) The methodology for training $(h, r, ?)$ can be applied to train $(?, r, t)$ as well. 3) Multiple meta queries can be combined to obtain complex queries, which can be formalized as $\psi(c, q_1, q_2, ..., q_k) = dist(\mathbf{c} - \phi_1(\phi_2(...\phi_k(q))))$.

Furthermore, in order to accelerate the computation by utilizing the bitwise operations, we introduce the concept of KG hashing.

Definition 2 *(Knowledge Graph Hashing). Given a KG* $\mathcal{G} = \mathcal{E} \times \mathcal{R} \times \mathcal{E}$*, the problem of KG hashing aims to represent each entity* $e \in \mathcal{E}$ *into a low-dimensional Hamming space* $\{\pm 1\}^d$*, i.e., learning a hash function* $h_{\mathcal{G}} : \mathcal{E} \mapsto \{\pm 1\}^d$*, where* $d \ll n$ *and* $n = |\mathcal{E}|$ *is the size of* $\mathcal{E}$*. In the space* $\{\pm 1\}^d$*, the bit uncorrelation and balance conditions should be satisfied as much as possible.*

Bit balance means that the binary value on each dimension has equal chance to be 1 or -1, maximizing the entropy of each dimension. Bit uncorrelation means that different dimensions of a vector are uncorrelated, being usually implemented by orthogonal constraints.

Next, we will introduce our proposed method HALF in detail. In HALF, we mainly design a practical methodology to modify various types of KGE models to a search-friendly way, by which the evaluation time of each candidate entity will be reduced. The architecture is summarized in Figure 1.

2.1 Search-Oriented Knowledge Graph Embedding

In this section, we study how to rewrite the scoring function of KGE models into the search-oriented form:

$$\psi(c, q) = -dist(\mathbf{c} - \phi(q)) \tag{1}$$

By deconstructing the final operation of the KGE scoring functions, we take the following three basic operators into consideration, which are dot product, Frobenius norm and linear transformation.

Table 1. A summary of KGE scoring functions and the respective search-oriented version. $\star$ and $*$ represent correlation operator and convolution operator respectively [23], and f_c represents a convolution neural network.

Method	Embedding	Scoring function $\psi(h,r,t)$	Hash code	Scoring function $dist(\psi(h,r),t)$
TransE [4]	$\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^{d_1}$	$-\|\|\mathbf{h}+\mathbf{r}-\mathbf{t}\|\|_\mathbf{2}$	$\mathbf{h},\mathbf{t} \in \pm 1^{d_2}$	$-Hamm(\mathbf{h}+\mathbf{r},\mathbf{t})$
TransR [20]	$\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^{d_1};\ \mathbf{M_r} \in \mathbb{R}^{d_1 \times d_1}$	$-\|\|\mathbf{M_r h}+\mathbf{r}-\mathbf{M_r t}\|\|_\mathbf{2}$	$\mathbf{h},\mathbf{t} \in \pm 1^{d_2}$	$-Hamm(\mathbf{M_r^{-1}}(\mathbf{M_r h}+\mathbf{r}),\mathbf{t})$
RESCAL [24]	$\mathbf{h},\mathbf{t} \in \mathbb{R}^{d_1};\ \mathbf{M_r} \in \mathbb{R}^{d_1 \times d_1}$	$\mathbf{h}^\mathsf{T}\mathbf{M_r t}$	$\mathbf{h},\mathbf{t} \in \pm 1^{d_2}$	$-Hamm(\mathbf{h}^\mathsf{T}\mathbf{M_r},\mathbf{t})$
HolE [23]	$\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^{d_1}$	$\mathbf{r}^\mathsf{T}(\mathbf{h}\star\mathbf{t})$	$\mathbf{h},\mathbf{t} \in \pm 1^{d_2}$	$-Hamm(\mathbf{h}*\mathbf{r},\mathbf{t})$
ANALOGY [21]	$\mathbf{h},\mathbf{t} \in \mathbb{R}^{d_1};\ \mathbf{M_r} \in \mathbb{R}^{d_1 \times d_1}$	$\mathbf{h}^\mathsf{T}\mathbf{M_r t}$	$\mathbf{h},\mathbf{t} \in \pm 1^{d_2}$	$-Hamm(\mathbf{h}^\mathsf{T}\mathbf{M_r},\mathbf{t})$
ConvE [8]	$\mathbf{h},\mathbf{r},\mathbf{t} \in \mathbb{R}^{d_1}$	$f_c(\mathbf{h},\mathbf{r})\mathbf{t}$	$\mathbf{h},\mathbf{t} \in \pm 1^{d_2}$	$-Hamm(f_c(\mathbf{h},\mathbf{r}),\mathbf{t})$

2.1.1 Dot Product

Dot product is a binary operation which is wildly utilized in matrix factorization-based and neural network KGE models. Dot-product is formalized as $dot : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}$ and defined as:

$$dot(\mathbf{a},\mathbf{b}) = \sum_{i=1}^{d} a_i b_i \tag{2}$$

With supposing $\mathbf{a},\mathbf{b} \in \{\pm 1\}^d$, then

$$\begin{aligned} dot(\mathbf{a},\mathbf{b}) &= \sum_{i \in \{a_i = b_i\}}^{d} a_i b_i + \sum_{i \in \{a_i \neq b_i\}}^{d} a_i b_i \\ &= d - 2Hamm(\mathbf{a},\mathbf{b}), \end{aligned} \tag{3}$$

where $Hamm(.)$ means the Hamming distance between two vectors. Hence, the following objective function

$$\max_{\mathbf{a},\mathbf{b}} dot(\mathbf{a},\mathbf{b}) \quad s.t.\ \mathbf{a},\mathbf{b} \in \{\pm 1\}^d \tag{4}$$

is equivalent to

$$\min_{\mathbf{a},\mathbf{b}} Hamm(\mathbf{a},\mathbf{b}) \quad s.t.\ \mathbf{a},\mathbf{b} \in \{\pm 1\}^d, \tag{5}$$

which can be applied to modify the scoring function of RESCAL [24], HolE [23] and ConvE [8], etc.

2.1.2 Frobenius Norm

Frobenius norm, or the so-called l_2-norm, is widely used in the scoring functions of translation-based KGE models, which is defined as:

$$||\mathbf{a}||_2 = \sqrt{\sum_{i=1}^{d} a_i^2}, \tag{6}$$

With supposing $\mathbf{a},\mathbf{b} \in \{\pm 1\}^d$, then:

$$\begin{aligned} ||\mathbf{a}-\mathbf{b}||_2 &= \sqrt{\sum_{i=1}^{d} (\mathbf{a_i}-\mathbf{b_i})^\mathbf{2}} \\ &= \sqrt{\sum_{i \in \{a_i = b_i\}}^{d} (\mathbf{a_i}-\mathbf{b_i})^\mathbf{2} + \sum_{i \in \{a_i \neq b_i\}}^{d} (\mathbf{a_i}-\mathbf{b_i})^\mathbf{2}} \\ &= 2\sqrt{Hamm(\mathbf{a},\mathbf{b})} \end{aligned} \tag{7}$$

2.1.3 Linear Transformation

Linear transformation is a mapping function $\mathcal{M} : \mathbb{R}^{d_1} \mapsto \mathbb{R}^{d_2}$, which is equivalent with a matrix $\mathbf{M} \in \mathbb{R}^{d_1 \times d_2}$. Linear transformation is commonly seen in variants of TransE [4] (e.g., TransR [20]) to evaluate triples by utilizing the entity projection in a relation-associated space. Hence, the problem of separate the t from such linear transformation-based scoring functions could be formalized to learn an inverse function $\mathcal{A} : \mathbb{R}^{d_2} \mapsto \mathbb{R}^{d_1}$. With the constraint of $\mathbf{M}$ is a non-singular matrix, the optimal matrix $\mathbf{A}$ is supposed to satisfy

$$\mathbf{AM} = \mathbf{MA} = \mathbf{I}; \quad \mathbf{A} = \mathbf{M^{-1}}. \tag{8}$$

According to the above strategies and the setting of Search-Oriented KGE, we rewrite the scoring functions of several classical models, which are summarized in Table 1.

2.2 Hash Learning

In this section, we introduce the approximation method to obtain the optimal binary vectors of entities, with maintaining the bit balance and uncorrelation.

2.2.1 Hash Layer

HALF aims to learn a d_b-dimensional binary vector $\mathbf{e_b}$ by converting the d-dimensional embedding $\mathbf{e}$, which is a real-value embedding and continuous in nature. The binarization process can be simplified as a dense layer with a sign activation function, or the so-called hash layer $g = \mathrm{sign}(\mathbf{W}^\mathsf{T}\mathbf{e}+\mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{d \times d_b}$ and $\mathbf{b} \in \mathbb{R}^{d_b}$ are parameters.

However, the gradient of sign function is zero for all nonzero inputs and ill-defined at zero, which is not qualified in the standard back-propagation for training neural networks. Inspired by HashNet [6], we introduce the weighted softsign as the approximation activation function:

$$\mathrm{softsign}(\beta, z) = \frac{\beta z}{1+\beta|z|}, \tag{9}$$

where z is the input and $\beta > 0$ is a scaling weight and $|.|$ is the absolute value of the input. With increasing β, the weighted softsign will converge to the original sign function,

$$\lim_{\beta \to \infty} \mathrm{softsign}(\beta, x) = \mathrm{sign}(x) \tag{10}$$

The softsign is similar to the hyperbolic tangent tanh utilized in HashNet [6], but its tails are quadratic polynomials rather than exponentials, i.e., it approaches its asymptotes much slower.

Algorithm 1: Multi-Objective Optimization

Input: The initial representation $\mathbf{r} \in \mathbb{R}^{\mathbf{d_1}}$ of relations $r \in \mathcal{R}$. The initial representation $\mathbf{e} \in \mathbb{R}^{\mathbf{d_1}}$ of entities $e \in \mathcal{E}$, which is wrote as $\mathbf{h}$ and $\mathbf{t}$ to differentiate the head and tail entity in a triple $t \in \mathcal{T}$. A sequence $1 = \beta_0 < \beta_1 < ... < \beta_S = \infty$ and current stage s. The rate p for self-balanced dropout. The initial parameter $\mathbf{\Theta_0}$.

Output: The binarized hash code $\hat{\mathbf{e}} \in \{\pm\mathbf{1}\}^{\mathbf{d_2}}$ of each entity $e \in \mathcal{E}$, which is wrote as $\hat{\mathbf{h}}$ and $\hat{\mathbf{t}}$ to differentiate the head and tail entity. The trained parameter $\mathbf{\Theta}$.

Initialize the structure of the original KGE model and the search-oriented model by Eq. (3), (7), and (8);
Initialize the structure of hash layer g by Eq. (9), with setting $\mathrm{softsign}(\beta_s z)$ as current activation function;
Initialize $\lambda_1 = 0$, $\lambda_2 = 1 - \beta_1$, and $\lambda_3 = 0.001$;
Initialize optimizer O;
for $epoch \leftarrow 0$ to $|Epoch|$ **do**
  for $batch \leftarrow 0$ to $|Batch|$ **do**
    Calculate $\mathcal{L}$ by Eq. (11), (14), and (15); // sample mini-batch from train set
    Update $\mathbf{\Theta}$ and the embedding $\mathbf{h},\ \mathbf{r},\ \mathbf{t}$ by O according to $\mathcal{L}$;
  Update $\lambda_1 \leftarrow \lambda_1 + \frac{1}{|Epoch|}$; // update the objective according to the training process
  Update $\lambda_2 \leftarrow 1 - \lambda_1$;
  if $epoch\ \%\ 10$ **is** 0 **then**
    // update the weight of softsign to approximate sign
    Update $\mathrm{softsign}(\beta_s z) \leftarrow \mathrm{softsign}(\beta_{s+1} z)$ as the activation function of g;
return $\hat{\mathbf{e}} \leftarrow g(\mathbf{e})$ where the activation is sign;

2.2.2 Bit Balance and Uncorrelation

The representation $\mathbf{E} \in \{\pm 1\}^{d_b \times n}$ of the entity set $\mathcal{E}$ is a binary matrix, in which each column is the d_b-dimensional hash code of an entity e and n is the total number of entities. In order to maximize the entropy of E and maintain the semantic similarity between the representation of similar entities, we introduce bit balance $\mathbf{E}\mathbf{1}_n = \mathbf{0}$ and uncorrelation $\mathbf{E}\mathbf{E}^{\mathsf{T}} = \mathbf{I}_d$ as constraints, where $\mathbf{1}_n$ is a vector of length n with all ones. In other words, bit balance requires each bit to fair 50% of the time, and bit uncorrelation requires each row in $\mathbf{E}$ is uncorrelated with any other rows and the uncorrelation of columns is ignored, which indicates alias may share the same hash code. Here, we approximate the above two constraints by utilizing batch normalization and self-balanced dropout during the training of HALF.

Batch Normalization (BN) is generally utilized to accelerates the training and reduces the overall impact of the weights scale [15]. With normalized by BN, the output $\hat{z}$ of each neuron of the hash layer has the expected value of 0 and the variance of 1, as long as the elements of each mini-batch are sampled from the same distribution neglecting noise. The normalization process is formulated as follows:

$$\hat{z} = \frac{z - \mu}{\delta + \epsilon}, \tag{11}$$

where μ and δ are statistics of mean and standard deviation respectively, and ϵ is the added minor noise. With maintaining the bit balance of the ith dimension of jth hash code $\mathbf{e}_b^j$, the mean of the ith dimension is

$$\mu^i = \frac{1}{n}\sum_{j=1}^{n} \mathbf{E}_{ij} = 0, \tag{12}$$

and the standard deviation is

$$\delta^i = \frac{1}{n}\sum_{j=1}^{n} \sqrt{(\mathbf{E}_{ij} - \mu^i)^2} = 1. \tag{13}$$

Namely, the optimal balanced hash code obey a special case of distribution $\mathcal{N}(0, 1)$, and BN is the approximated balance constraint for mini-batch training.

Self-Balanced Dropout (SBD) is a novel variant of dropout algorithm to prevent over-fitting by randomly perturbing the features of the inputs [18]. For every modified input $\hat{\mathbf{e}}_i \in \mathbb{R}^d$ for the hash layer, the SBD maintains each $\mathbf{e}_{ij}$ with the keep probability p, or set to a trainable variable $\mathbf{e}_{mask}$, not 0 in original dropout. Formally, $\mathbf{e}_{ij}$ modified by SBD is expressed as:

$$\hat{\mathbf{e}}_{ij} = \begin{cases} \mathbf{e}_{ij}, & with\ probability\ p \\ \mathbf{e}_{mask}, & with\ probability\ q = 1 - p \end{cases} \tag{14}$$

Compared with the original dropout, SBD prevents the similar optimization direction of parameters when some features in $\mathbf{e}$ are highly related. Refer to [18] for detailed proof. Here, the random perturbation of SBD is employed in HALF, which aims to guarantee the bit uncorrelation between each dimension of hash code $\mathbf{e}_b$.

2.3 Multi-Objective Optimization

The loss function of HALF is as follows:

$$\begin{aligned} \mathcal{L} =\ & \mathcal{L}_{HALF} + \mathcal{L}_{KGE} + \mathcal{L}_{REG} \\ =\ & \lambda_1 \Big(\sum_{(h,r,t)\in\mathcal{G}} ||\phi(\hat{\mathbf{h}}, \mathbf{r}) - \hat{\mathbf{t}}||_2 - \sum_{(h',r,t')\notin\mathcal{G}} ||\phi(\hat{\mathbf{h}}', \mathbf{r}) - \hat{\mathbf{t}}'||_2 \Big) \\ & + \lambda_2 \Big(\sum_{(h,r,t)\in\mathcal{G}} \psi(\mathbf{h}, \mathbf{r}, \mathbf{t}) - \sum_{(h',r,t')\notin\mathcal{G}} \psi(\mathbf{h}', \mathbf{r}, \mathbf{t}') \Big) \\ & + \lambda_3 ||\mathbf{\Theta}||_2 \end{aligned} \tag{15}$$

In Eq. (15), the first term $\mathcal{L}_{HALF}$ measures loss in the modified search-oriented KGE method, where $\hat{\mathbf{h}}$ or $\hat{\mathbf{t}}$ mean the hash codes of entities which are generated from the embedding $\mathbf{h}$ or $\mathbf{t}$ by a trained hash layer. The second term $\mathcal{L}_{KGE}$ measures loss in the original KGE method, which directs the model to learn the real-value embedding of entities and relations. The last is the regularization term for preventing over-fitting, with λ_1 λ_2, and λ_3 are the weight parameters.

In predicting period, with given the query $(h, r, ?)$, we utilize the trained function $\phi(\mathbf{h}, \mathbf{r})$ to generate a index hash code of the target

entity t, or the so-called tail entity, and search the nearest entity from the candidate set with the hamming distance to the index. The training procedure is demonstrated in Algorithm 1.

3 Experiment

3.1 Datasets and Evaluation Protocol

For evaluating the performance of the proposed framework for link prediction on knowledge graphs, we conduct experiments on four commonly used benchmark datasets (i.e., WN18, FB15K, WN18RR, and FB15K237). Table 2 shows the data statics.

Table 2. Benchmark datasets.

	WN18	FB15K	WN18RR	FB15K237
$\lvert\mathcal{E}\rvert$	40,943	14,951	40,559	14,505
$\lvert\mathcal{R}\rvert$	18	1,345	11	237
$\lvert\mathcal{T}_{train}\rvert$	141,442	483,142	86,835	272,115
$\lvert\mathcal{T}_{valid}\rvert$	5,000	5,000	3,034	17,535
$\lvert\mathcal{T}_{test}\rvert$	5,000	59,071	3,134	20,466

We followed the standard evaluation protocol, link prediction, which aims to predict an entity that has a specific relation r with another entity, i.e., predicting t with given $(h, r, ?)$ or predicting h with given $(?, r, t)$. Note that, in this paper we focus on consuming the total computation time in the former setting $(h, r, ?)$, while predicting $(?, r, t)$ we can take $dist(\phi(?, r), t)$ as a scoring function and test each candidate h as well.

3.2 Baselines

We compare the performance of HALF against a variety types of KGE models developed in recent years. They are:

- Translation based models, i.e., TransE [4] and TransR [20], which model relations as translation operators between entities in the embedding space.
- Semantic matching based models, i.e., RESCAL [24], HolE [23], and Analogy [21], which deconstruct knowledge graph via collective matrix factorization.
- Semantic matching based model, i.e., ConvE [8], which classifies a triple into a fact or wrong via neural networks.

We apply the proposed HALF to modify the above five models and we try to clarify: 1) whether our proposed strategies for rewriting scoring function is effective and 2) whether the hash learning method is work. To evaluate the performance of trained hash codes of KG, we list the following methods as baselines as well:

- Truncating the entity embedding trained by KGE models with a sign function.
- Binarized CANDECOMP/PARAFAC (B-CP) [16], which is a tensor decomposition based model to learn the binarized KG embeddings.

3.3 Experiment Setup

In order to validate the performance of variant KGE models, we mainly conduct experiments based on OpenKE [13], containing the reimplementation of several typical models (e.g., TransE, ANALOGY, HolE, etc.)[3] and further incorporate the implementation of ConvE[4] and B-CP[5] into OpenKE. We selected the hyper-parameters of the HALF framework via grid search according to the mean reciprocal rank (MRR) on the validation set. The search range of hyper-parameter are as follows - self-balanced dropout in hash layer $\{0.0, 0.1, 0.2, 0.3\}$, embedding dropout $\{0.0, 0.1, 0.2, 0.3\}$, batch size $\{16, 32, 64, 128, 256\}$, learning rate $\{0.0001, 0.001, 0.003, 0.01\}$. In order to fairly evaluate the performance of each model, we will set the embedding size and the hash code length to 200.

3.4 Results

We applied HALF to variant KGE methods and then compared the HALF-versions with original models and other hashing approaches (i.e, B-CP) on WN18 and FB15K. Moreover, since HALF-ConvE and HALF-HolE showed a competitive performance, we additionally tested them on WN18RR and FB15K237. The main results are shown in Table 3.

3.4.1 Signed KGE V.S. Original KGE:

According to the performance of most KGE models and their signed variants, quantizing the trained entity embedding causes serious loss of information. For example, the original ConvE achieved the best performance with the trained real-value embedding while the signed version even worse than Sgn-TransE. It seems that the more trained parameters kept inside models, the worse information loss caused by the quantization of embeddings.

Moreover, the performance of Sgn-HolE is totally different from other variants, as it achieves a high accuracy on most benchmarks, However, the rate of Hit@10 is equal to Hit@1, which indicates that no less than 10 binarized hash codes cannot be distinguished as the target due to the equal Hamming distance. In a other words, the good news is that the target entity do generally appears in the nearest area (i.e., the first-order set) to a query, e.g., (h, r, ?), but the specific capacity of the first-order entities still needs to be explored.

3.4.2 HALF-Models V.S. Signed-KGE:

With the proposed hash learning framework, the accuracy of each model has been improved in varying degrees. There are three interesting information hidden in the comparison:

1) HALF combined with neural network-based models generalize a higher performance improvement compared to its signed-version than matrix factorization-based or translation-based models. For example, although the performance of original-ANALOGY is similar to original-ConvE, after hashing, the performance of the later is 10 percentages higher than that of the former.
2) HolE shows competitive performance in both signed-version and HALF-version. Moreover, HALF-HolE even achieves much higher performance than Original-HolE on most benchmarks.
3) Hit@10 may not work in the context of hash-based searching, nor does MRR. Actually, in addition to HolE, other signed models can still distinguish Hit@1 from Hit@10 by quantizing directly, because the intermediate results are not strictly discretized, for example, when $\mathbf{h} \in \{\pm\mathbf{1}\}^{\mathbf{d}}$ and $\mathbf{r} \in \mathbb{R}^{\mathbf{d}}$, then sum $(\mathbf{h} + \mathbf{r}) \notin \{\pm\mathbf{1}\}^{\mathbf{d}}$.

[3] https://github.com/thunlp/OpenKE.git
[4] https://github.com/TimDettmers/ConvE.git
[5] https://github.com/KokiKishimoto/cp decomposition.git

Table 3. Link prediction results on four benchmarks. We

	Models	WN18				FB15K				WN18RR				FB15K237			
			Hit@				Hit@				Hit@				Hit@		
		MRR	10	3	1	MRR	10	3	1	MRR	10	3	1	MRR	10	3	1
Original-KGE	TransE	.388	.741	.609	.139	.380	.641	.472	.231	-	-	-	-	-	-	-	-
	TransR	.897	.920	.906	.886	.477	.731	.563	.336	-	-	-	-	-	-	-	-
	RESCAL	.548	.726	.597	.456	.280	.470	.314	.184	-	-	-	-	-	-	-	-
	ANALOGY	.942	.947	.944	**.939**	.725	.854	.785	.646	-	-	-	-	-	-	-	-
	HolE	.938	.947	.945	.930	.524	.739	.613	.402	.365	.379	.369	.356	.162	.297	.180	.095
	ConvE	**.942**	**.955**	**.947**	.935	**.745**	**.873**	**.801**	**.670**	**.460**	**.480**	**.430**	**.390**	**.316**	**.491**	**.350**	**.239**
Signed-KGE	Sgn-TransE	.024	.067	.026	.000	.099	.278	.134	.004	-	-	-	-	-	-	-	-
	Sgn-TransR	.001	.001	.001	.000	.001	.001	.000	.000	-	-	-	-	-	-	-	-
	Sgn-RESCAL	.092	.158	.096	.058	.118	.197	.124	.077	-	-	-	-	-	-	-	-
	Sgn-ANALOGY	.442	.581	.503	.382	.121	.185	.148	.097	-	-	-	-	-	-	-	-
	Sgn-ConvE	.304	.349	.316	.241	.214	.284	.224	.157	0.153	.209	.136	.124	.091	.118	.084	.029
	Sgn-HolE	**.731**	**.731**	**.731**	**.731**	**.689**	**.689**	**.689**	**.689**	**.283**	**.283**	**.283**	**.283**	**.147**	**.147**	**.147**	**.147**
d=200	B-CP	**.901**	**.933**	**.918**	**.881**	**.695**	**.835**	**.760**	**.611**	**.450**	**.500**	**.460**	**.420**	**.278**	**.446**	**.304**	**.194**
HALF-(d=200) Models	Half-TransE	.354	.755	.477	.154	.262	.468	.288	.164	-	-	-	-	-	-	-	-
	Half-TransR	.114	.196	.123	.006	.115	.222	.119	.006	-	-	-	-	-	-	-	-
	Half-RESCAL	.124	.212	.132	.007	.125	.243	.131	.006	-	-	-	-	-	-	-	-
	Half-ANALOGY	.578	.782	.649	.467	.173	.328	.188	.009	-	-	-	-	-	-	-	-
	Half-ConvE	**.903**	**.923**	**.911**	**.897**	.732	.759	.727	.703	**.422**	**.471**	**.439**	.401	.359	.393	.350	.306
	Half-HolE	.875	.875	.875	.875	**.826**	**.838**	**.830**	**.817**	.413	.413	.412	**.412**	**.584**	**.599**	**.586**	**.572**

3.4.3 HALF-Models V.S. B-CP:

As shown in Table 3, B-CP is a binarized KGE model which generalizes good performance in link prediction tasks. However, B-CP is essential a KGE model than a KG hash learning model, where the representation of each entity is quantized into $\{\pm\frac{1}{3}\}^d$ in practice. Moreover, B-CP has not utilized the advances of Hamming distance to accelerate the target searching, but is designed as a discriminant model to determine the probability of each triple. Compared with the B-CP, the proposed Half-HolE/ConvE achieved the state of the art performance with competitive efficiency and accuracy.

3.5 Analysis

3.5.1 Bit-Balance and Uncorrelation

In this section, we compare the performance of the HALF without the bit-balance and uncorrelation constraints and the unconstrained HALF. We conduct experiments on four benchmarks and list the MRR score in Table 4.

Table 4. The effectiveness of bit-balance and uncorrelation constraints.

	MRR			
	FB15K	WN18	FB15K237	WN18RR
Sign-ConvE	.304	.214	.153	.091
HALF-ConvE (unconstrained)	.857	.691	.375	.260
HALF-ConvE	.903	.732	.422	.359

Since hash codes of the same dimension contain less information than real-value embeddings, considering to maximize the entropy of a hash code, we formalize the objective function as follows:

$$\text{argmax } H = -\sum_{i=1}^{d'}[p_i \log \frac{1}{p_i} + (1-p_i)\log\frac{1}{1-p_i}] \quad (16)$$
$$s.t.\ 1 \leq d' \leq d, 0 \leq p_i \leq 1,$$

where H measures the quantity of information in a d-dimensional binary vector. $d' \leq d$ is the number of uncorrelation dimensions and p_i is the probability of ith dimension to be 1, otherwise it will be -1. Obviously, the bit-balance constraint keeps each $p_i = 0.5$ and the bit-uncorrelation constraint keeps $d' = d$, which is the optimal solution for Eq. 16.

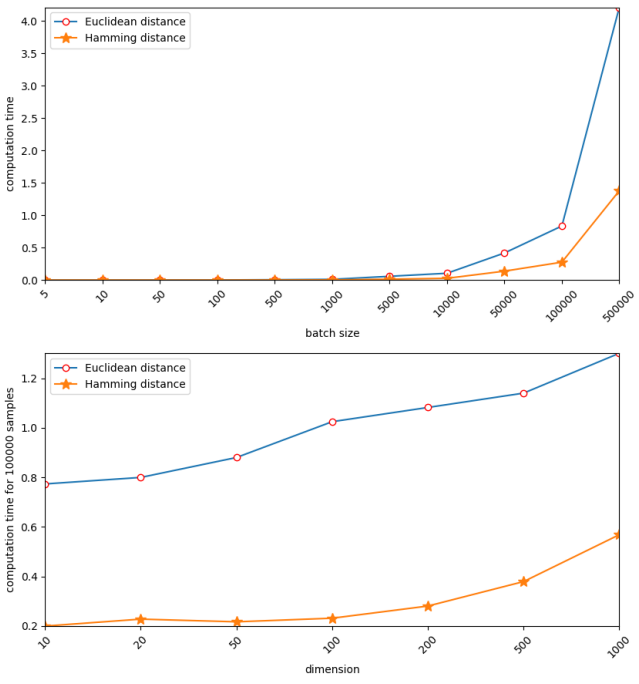

Figure 2. The comparison of computation time (seconds) between Euclidean distance and Hamming distance, varying with the batch size or the dimension size.

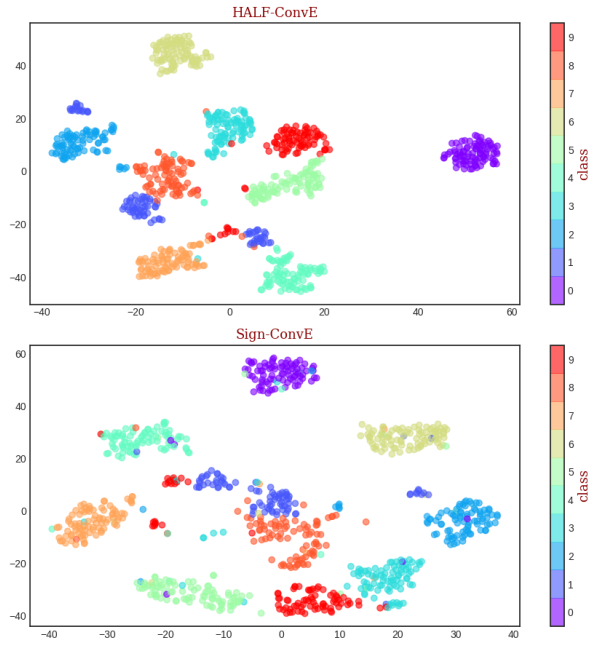

Figure 3. Binarized KG embedding visualization.

3.5.2 *Time Efficiency*

As described in the above section, HALF can accelerate the computation of predicting period by using the bitwise operations (XNOR and Bitcount). To compare the computation speed between Euclidean distance and Hamming distance, we conduct two experiments as follows: (a) fix the embedding dimension $d = 200$ and calculate the execution time varying the batch size n from 5 to 500,000, and (b) fix the batch size $n = 100,000$ and calculate the execution time varying the embedding dimension d from 10 to 1,000. Figure 2 clearly shows that bitwise operations can provide speed-up compared to standard float operation in Euclidean distance computation.

3.5.3 *Embedding Visualization*

In Figure 3, we visualize the binarized KG embedding distribution using t-SNE [9, 22]. The binarized embeddings are trained on FB15K, but only 10 classes of entity embedding are displayed for easier comparison. Model (a) the signed ConvE achieves 21.4% MRR and our proposed version, Model (b) HALF-ConvE achieves 73.2% MRR.

We sampled 1000 entities from FB15K randomly , which are evenly distributed in 10 categories. Obviously, HALF modified model has more compact and separable clusters, which indicates that binarized embeddings keep more semantic similarity via the training process. This descends from the design of the multi-objective training framework. Without HALF, the signed embeddings overlap with each other, which may make the original model difficult to compare.

4 Related Work

4.1 Knowledge Graph Embedding

Several KGE models have been introduced in the literature, such as translation-based models (e.g., TransE [4] and TransR[20]), matrix factorization-based models (e.g., RESCAL [24], HoloE [23], and Analogy [21]), and neural network-based models (e.g., ConvE [8]). We refer to [30] for a recent survey. Generally, the proposed models embed entities and relations into a Euclidean space or Complex space and evaluate the performance of model according to the accuracy of link prediction or triple classification. However, when leveraging the learning KG embeddings in search-intensive applications such as semantic search [28, 12, 11] and question answering [14, 31], the KGE methods still need to be modified to specific real-world problems, e.g., approximate search [28, 14], relation reasoning [11].

Basically, the search-oriented KGE is a type of generative representation learning approach, such as TransG [32], GAKE [10], KGgan [5]. Contrasted with the proposed methods, we utilize the idea of multi-task learning [7] and propose a uniform framework to transform discriminative models (i.e, TransE, RESCAL, HoloE, ConvE etc) into generative models.

4.2 Information Network Hashing

Information network hashing, or binary network embedding has gradually [19, 25, 33, 29] been gaining attention for fast queries in large-scale homogeneous networks [1]. Apart from decreasing storage by representing each node with a few bits, information network hashing can also accelerate algorithm training by replacing inner products and distances in the Euclidean space with efficient bit-wise operations and Hamming distances. The later can be further applied to accelerate k-nearest neighbor query using recent advances in hash-code indexing [19].

4.3 Knowledge Graph Hashing

In contrast to information networks, a KG is generally formalized as a heterogeneous network, where the type of edges between each pair of nodes may be different. Binarized CANDECOMP/PARAFAC (B-CP) [17] proposed a CP-decomposition based knowledge graph hashing method, while only utilizing bit-wise operations to accelerate algorithm training ignoring the potential of hash codes in efficient searching. Wang et al. [27] also introduce a framework for encoding several specific incomplete KGs and graph queries in Hamming space. However, they cannot be applied to general KGE models for learning the binary embeddings.

5 Conclusion

In this paper, we proposed a novel hash learning framework HALF for KG representation learning. In HALF, each entity is represented by a binary vector, in turn, it will generate a candidate t' with h and r to index the target entity for Hamming distance-based quick search. We conducted comparative experiments on benchmark datasets to evaluate the performance of our framework. Results show that the framework significantly speeds up the searching process and provides good results. In the future, we will explore the real-world applications (e.g., large-scale knowledge graph link prediction) based on the learning binarized KG embeddings.

Acknowledgment

This work was supported by National Key Research and Development Program of China with Grant Nos. 2018YFC0830201 and 2017YFB1002801; National Science Foundation of China with Grant Nos. 61906037, 61902063 and U1736204; the Judicial Big Data Research Centre, School of Law at Southeast University with Grant No.4313059291; the Opening Project of State Key Laboratory of Digital Publishing Technology with Grant No. cndplab-2020-M001; and the Legendary NBA player Kobe Bryant, his spirit encourages me to keep pushing boundaries of my research work.

REFERENCES

[1] Dimitrios Bermperidis, *Active and Adaptive Techniques for Learning over Large-Scale Graphs*, Ph.D. dissertation, 2019.
[2] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann, 'Dbpedia-a crystallization point for the web of data', *Journal of Web Semantics*, **7**(3), 154–165, (2009).
[3] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor, 'Freebase: a collaboratively created graph database for structuring human knowledge', in *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 1247–1250, (2008).
[4] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko, 'Translating embeddings for modeling multi-relational data', in *Proceedings of the 27th Annual Conference on Neural Information Processing Systems*, pp. 2787–2795, (2013).
[5] Liwei Cai and William Yang Wang, 'KBGAN: adversarial learning for knowledge graph embeddings', in *Proceedings of the 16th North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 1470–1480, (2018).
[6] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu, 'Hashnet: Deep learning to hash by continuation', in *Proceedings of the 16th IEEE International Conference on Computer Vision*, pp. 5609–5618, (2017).
[7] Rich Caruana, 'Multitask learning', *Machine Learning*, **28**(1), 41–75, (1997).
[8] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel, 'Convolutional 2d knowledge graph embeddings', in *Proceedings of the 32th Conference on Artificial Intelligence*, pp. 1811–1818, (2018).
[9] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell, 'Decaf: A deep convolutional activation feature for generic visual recognition', in *Proceedings of the 31st International Conference on Machine Learning*, pp. 647–655, (2014).
[10] Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu, 'GAKE: graph aware knowledge embedding', in *Proceedings of the 26th International Conference on Computational Linguistics*, pp. 641–651, (2016).
[11] William L. Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec, 'Embedding logical queries on knowledge graphs', in *Proceedings of the 31st Annual Conference on Neural Information Processing Systems*, pp. 2030–2041, (2018).
[12] Shuo Han, Lei Zou, Jeffrey Xu Yu, and Dongyan Zhao, 'Keyword search on RDF graphs - A query graph assembly approach', in *Proceedings of the 26th Conference on Information and Knowledge Management*, pp. 227–236, (2017).
[13] Xu Han, Shulin Cao, Xin Lv, Yankai Lin, Zhiyuan Liu, Maosong Sun, and Juanzi Li, 'Openke: An open toolkit for knowledge embedding', in *Proceedings of the 22nd Conference on Empirical Methods in Natural Language Processing*, pp. 139–144, (2018).
[14] Xiao Huang, Jingyuan Zhang, Dingcheng Li, and Ping Li, 'Knowledge graph embedding based question answering', in *Proceedings of the 12th International Conference on Web Search and Data Mining*, pp. 105–113, (2019).
[15] Sergey Ioffe and Christian Szegedy, 'Batch normalization: Accelerating deep network training by reducing internal covariate shift', in *Proceedings of the 32nd International Conference on Machine Learning*, pp. 448–456, (2015).
[16] Koki Kishimoto, Katsuhiko Hayashi, Genki Akai, Masashi Shimbo, and Kazunori Komatani, 'Binarized knowledge graph embeddings', in *Proceedings of the 41st European Conference on Information Retrieval*, pp. 181–196, (2019).
[17] Koki Kishimoto, Katsuhiko Hayashi, Genki Akai, Masashi Shimbo, and Kazunori Komatani, 'Binarized knowledge graph embeddings', in *Proceedings of the 41st European Conference on Information Retrieval*, pp. 181–196, (2019).
[18] Shen Li, Chenhao Su, Renfen Hu, and Zhengdong Lu, 'Self-balanced dropout', *CoRR*, **abs/1908.01968**, (2019).
[19] Defu Lian, Kai Zheng, Vincent W. Zheng, Yong Ge, Longbing Cao, Ivor W. Tsang, and Xing Xie, 'High-order proximity preserving information network hashing', in *Proceedings of the 24th International Conference on Knowledge Discovery & Data Mining*, pp. 1744–1753, (2018).
[20] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu, 'Learning entity and relation embeddings for knowledge graph completion', in *Proceedings of the 29th Conference on Artificial Intelligence*, pp. 2181–2187, (2015).
[21] Hanxiao Liu, Yuexin Wu, and Yiming Yang, 'Analogical inference for multi-relational embeddings', in *Proceedings of the 34th International Conference on Machine Learning*, pp. 2168–2178, (2017).
[22] Laurens van der Maaten and Geoffrey Hinton, 'Visualizing data using t-sne', *Journal of Machine Learning Research*, **9**, 2579–2605, (2008).
[23] Maximilian Nickel, Lorenzo Rosasco, and Tomaso A. Poggio, 'Holographic embeddings of knowledge graphs', in *Proceedings of the 30th Conference on Artificial Intelligence*, pp. 1955–1961, (2016).
[24] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel, 'A three-way model for collective learning on multi-relational data', in *Proceedings of the 28th International Conference on Machine Learning*, pp. 809–816, (2011).
[25] Xiaobo Shen, Shirui Pan, Weiwei Liu, Yew-Soon Ong, and Quan-Sen Sun, 'Discrete network embedding', in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 3549–3555, (2018).
[26] Denny Vrandečić and Markus Krötzsch, 'Wikidata: a free collaborative knowledgebase', *Communications of the ACM*, **57**(10), 78–85, (2014).
[27] Meng Wang, Haomin Shen, Sen Wang, Lina Yao, Yinlin Jiang, Guilin Qi, and Yang Chen, 'Learning to hash for efficient search over incomplete knowledge graphs', in *Proceedings of the 19th IEEE International Conference on Data Mining*, pp. 1360–1365, (2019).
[28] Meng Wang, Ruijie Wang, Jun Liu, Yihe Chen, Lei Zhang, and Guilin Qi, 'Towards empty answers in SPARQL: approximating querying with RDF embedding', in *Proceedings of the 17th International Semantic Web Conference*, pp. 513–529, (2018).
[29] Qixiang Wang, Shanfeng Wang, Maoguo Gong, and Yue Wu, 'Feature hashing for network representation learning', in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 2812–2818, (2018).
[30] Quan Wang, Zhendong Mao, Bin Wang, and Li Guo, 'Knowledge graph embedding: A survey of approaches and applications', *IEEE Trans. Knowl. Data Eng.*, **29**(12), 2724–2743, (2017).
[31] Ruijie Wang, Meng Wang, Jun Liu, Weitong Chen, Michael Cochez, and Stefan Decker, 'Leveraging knowledge graph embeddings for natural language question answering', in *Proceedings of the 24th International Conference on Database Systems for Advanced Applications*, pp. 659–675, (2019).
[32] Han Xiao, Minlie Huang, and Xiaoyan Zhu, 'Transg : A generative model for knowledge graph embedding', in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pp. 227–236, (2016).
[33] Hong Yang, Shirui Pan, Peng Zhang, Ling Chen, Defu Lian, and Chengqi Zhang, 'Binarized attributed network embedding', in *Proceedings of the 18th IEEE International Conference on Data Mining*, pp. 1476–1481, (2018).