Journal of Intelligent Information Systems (2023) 61:739–763
https://doi.org/10.1007/s10844-023-00796-y

RESEARCH

TuckerDNCaching: high-quality negative sampling with tucker decomposition

Tiroshan Madushanka[1,2] · Ryutaro Ichise[2,3]

Received: 19 January 2023 / Revised: 8 May 2023 / Accepted: 9 May 2023 /
Published online: 20 June 2023

Abstract
Knowledge Graph Embedding (KGE) translates entities and relations of knowledge graphs (KGs) into a low-dimensional vector space, enabling an efficient way of predicting missing facts. Generally, KGE models are trained with positive and negative examples, discriminating positives against negatives. Nevertheless, KGs contain only positive facts; KGE training requires generating negatives from non-observed ones in KGs, referred to as negative sampling. Since KGE models are sensitive to inputs, negative sampling becomes crucial, and the quality of the negatives becomes critical in KGE training. Generative adversarial networks (GAN) and self-adversarial methods have recently been utilized in negative sampling to address the vanishing gradients observed with early negative sampling methods. However, they introduce the problem of false negatives with high probability. In this paper, we extend the idea of reducing false negatives by adopting a Tucker decomposition representation, i.e., TuckerDNCaching, to enhance the semantic soundness of latent relations among entities by introducing a relation feature space. TuckerDNCaching ensures the quality of generated negative samples, and the experimental results reflect that our proposed negative sampling method outperforms the existing state-of-the-art negative sampling methods.

Keywords Negative sampling · Knowledge graph embedding · Tucker decomposition

1 Introduction

Knowledge Graphs (KGs), such as Freebase, DBpedia, WordNet, and YAGO, provide structured representations of facts(knowledge). These textual data are in the form of $(head, relation, tail)$, known as triplets, e.g., (*DaVinci, painted, MonaLisa*). KGs have

✉ Ryutaro Ichise
ichise@iee.e.titech.ac.jp

Tiroshan Madushanka
tiroshan@nii.ac.jp ; tiroshanm@kln.ac.lk

1 SOKENDAI (The Graduate University for Advanced Studies), Tokyo, Japan

2 National Institute of Informatics, Tokyo, Japan

3 Tokyo Institute of Technology, Tokyo, Japan

been utilized in many real-world applications, such as question-answering, recommender systems, and information retrieval systems. Although KGs contain an extensive volume of information, they are often incomplete and sparse as the knowledge is constructed on the basis of available facts or ground truth, which are subject to often changes. Therefore, it is essential to have methods to complete KGs automatically by adding missing/modified knowledge or facts that are changed with time. Recent research has shown the potential of utilizing machine learning (ML) methods such as Knowledge graph embedding (KGE) effectively for knowledge graph completion.

KGE approaches provide an efficient solution to find and complete missing facts in incomplete knowledge graphs. In fact, KGE methods provide better inference capability. KGE maps $entities$ and $relations$ in KGs into a low-dimensional vector space while preserving their semantic meaning. Recent KGE techniques have shown promising results in knowledge acquisition tasks such as link prediction and triplet classification. In conventional KGE approaches, observed instances (positives) rank higher than unobserved instances (negatives) and hence, accelerate the training process of ML algorithms. However, KGs contain only positive examples, so negative examples must be generated artificially. Hence, exploring strategies to generate quality negatives that support learning better knowledge representations is critical in KGE. For instance, considering the positive fact (*DaVinci, painted, MonaLisa*), the negative fact (*DaVinci, painted, CreationOfAdam*) is considered as a quality negative as it enables the KGE model to optimize the knowledge representation unlike a typical negative fact (*DaVinci, painted, France*). Therefore, negative sampling becomes indispensable in knowledge representation learning as the KGE model's performance relies on negative selection.

Most negative sampling methods involve randomly corrupting positives on the basis of a closed world assumption (Bordes et al., 2013; Wang et al., 2014) or exploiting the KG structure when generating negatives (Zhang et al., 2019; Ahrabian et al., 2020). Regardless, methods that randomly corrupt positives suffer from vanishing gradients as they generate triplets with zero gradients during training. As a solution to the vanishing gradient problem, a new direction for negative sampling has been introduced, adopting the changes in the negative sampling distribution and generating negatives with large gradients dynamically (Cai et al., 2018). However, to the best of our knowledge, most of the state-of-the-art negative sampling methods suffer from false negatives as they do not guarantee that the generated ones are always relevant negatives, i.e., in the case of generating true or latent positives as negatives. As KGE models are sensitive to inputs, false negatives usually fool the models, losing the semantics of $entities$ and $relations$. Therefore, generating quality negatives that enhance KGE representation learning is still an open and challenging task in negative sampling.

In this paper, we propose a negative sampling method by extending the previous work, MDNCaching (Madushanka et al., 2022) by introducing a relation feature space instead of a relationship matrix between the head and tail. The proposed method models latent relations using available positive KG elements and utilizes relation predictions to remove latent positives (false negatives) from the negative candidate space. We use the Tucker decomposition technique (Tucker, 1966) with our latent relation model representation to update entity and relation feature spaces effectively. The introduction of the relation feature space further extends the ability to represent multiple relations between entity pairs. First, we train a latent relation model from positive facts utilizing Tucker decomposition. Then, we predict the latent relations and eliminate false negatives from the candidate negative sample space. We use the caching technique to effectively manage negative triplets with large gradients and update the cache considering the changes to the embedding space to overcome the vanishing gradient problem.

In summary, the major contributions of this paper are three-fold. (1) A negative sampling method is introduced that eliminates the false negatives suffered in previous dynamic distribution-based negative sampling methods. (2) The Tucker decomposition technique is used with our novel latent relation model representation for modeling latent relations (3) Experiments on benchmark datasets reflect the effectiveness of TuckerDNCaching using standard metrics. The remainder of this paper is organized as follows. Section 2 discusses existing research on knowledge graph embedding and negative sampling. Section 3 presents the new negative sampling method for generating quality negatives with large gradients considering the dynamic distribution of the embedding space while eliminating false negatives referring to a latent relation model trained with the Tucker decomposition technique. Section 4 presents an experimental study in which we compare our proposed negative sampling method with baseline results of benchmark datasets and analyze results with the state-of-the-art. In Section 5, we conclude this paper.

2 Related work

Knowledge graph completion remains a challenging research field, and many different approaches have been introduced to make KGs machine-readable, utilizing reasoning techniques. Knowledge graph embedding, also called knowledge representation learning, projects KG elements, i.e., entities and relations, into a low-dimensional continuous vector space and utilizes the numerical representation of embeddings to perform knowledge acquisition tasks. Typically, using a scoring function, the KGE model captures the similarities between two entities on the basis of a relation. Depending on the properties of the scoring function, two main KGE approaches are found: translational distance-based models and semantic matching-based models. Translational distance-based models interpret relations as geometric transformations in the latent space, where models evaluate the distance of projected KG elements using a scoring function (Bordes et al., 2013; Ji et al., 2015; Wang et al., 2014). Recent studies have demonstrated improvements in translational distance-based models, such as TorusE (Ebisu et al., 2018), by addressing the regularization problem of TransE and incorporating techniques such as adaptive margins based on density distribution (Chenchen et al., 2019) and the use of self-attention and position-aware embeddings (Siheng et al., 2020). In contrast, the semantic matching-based approaches model the latent semantics represented in vectorized entities and relations through matrix decomposition (Nickel et al., 2011; Trouillon et al., 2016; Yang et al., 2015). The embeddings of both approaches are learned by solving an optimization problem that maximizes the scoring function for observed triplets (positives) while minimizing it for unobserved triplets (negatives). Thus, negative sampling is necessary for training a KGE model because negative and positive triplets must be provided during the training.

2.1 Negative sampling in KGE

KGE models learn knowledge representations by discriminating positives from negatives. Thus, the quality of the negatives affects the training of the models, and the performance of knowledge representation downstream tasks. The existing negative sampling approaches in KGE can be divided into two main categories: fixed distribution-based and dynamic distribution-based sampling.

2.1.1 Fixed distribution-based sampling

Fixed distribution-based sampling is typically utilized with knowledge representation learning due to its simplicity and efficiency. For example, Uniform negative sampling (Bordes et al., 2013) which constructs negative triplets by replacing (corrupting) either the head or tail entity of a positive triplet from a randomly sampled entity from an entity set. However, randomly corrupted negatives can be easily distinguished because they are low-quality negatives. In addition, Uniform sampling generates false negatives due to its random selection, such as replacing head entity $DaVinci$ with $Michelangelo$. It generates a false negative fact $(Michelangelo, Gender, Male)$.

Bernoulli negative sampling (Wang et al., 2014) is introduced with the idea of corrupting either the head entity or tail entity by considering the statistical information of entities and relations that enhance the chance of replacing the head entity in one-to-many relations and tail entity in many-to-one relations. In addition, other novel approaches (Kanojia et al., 2017; Krompaß et al., 2015; Xie et al., 2017) have analyzed statistical features of knowledge graphs to generate negatives rather than randomly corrupting positives. However, fixed distribution-based methods are sampled from fixed distributions. This approach does not consider the dynamics of the distributions and hence, this approach suffers from vanishing gradient problems (Zhang et al., 2019).

2.1.2 Dynamic distribution-based sampling

To address the problem of vanishing gradient existing in fixed distribution-based sampling methods, dynamic distribution-based sampling methods were introduced. With a successful adaptation of the Generative Adversarial Network (GAN) (Goodfellow et al., 2014) for modeling dynamic distributions, IGAN (Wang et al., 2018) and KBGAN (Cai et al., 2018) were introduced as negative sampling methods to generate negatives with large gradients. The GAN-based approach generates negatives by dynamically approximating a negative sampling distribution (generator). At the same time, training continues between the generator and the discriminator to optimize the knowledge representation. KBGAN is the first attempt to adapt GAN to negative sampling in KGE. In KBGAN, the generator produces a probability distribution over a candidate set of negatives, selects the one with the highest probability from candidates, and then feeds to the discriminator that minimizes the marginal loss between positive and negative samples to improve the final embedding. The generator is selected from one of two semantic matching-based KGE models (DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016)). The discriminator is selected from two translational distance-based KGE models (TransE (Bordes et al., 2013), TransD (Ji et al., 2015)). By replacing the probability-based log-loss KGE generator in KBGAN, IGAN utilizes a two-layer fully connected neural network as the generator while keeping the embedding model as the discriminator. KSGAN (Hu et al., 2019) is an extension of KBGAN, which is an adversarial learning approach with a new component for knowledge selection. The knowledge selection filters false triplets and selects a semantic negative triplet for a given positive triplet. However, GAN-based methods require pre-training which has an impact on efficiency. Later, RotatE (Sun et al., 2019) introduced a self-adversarial *Self-Adv* sampling approach based on a self-scoring function. However, Self-Adv does not perform consistently on other KGE models. By selecting negatives using a random walk approach that ignores non-semantic similar neighbors, Structure Aware Negative Sampling (SANS) (Ahrabian et al., 2020) shows better performance over the Self-Adv. With the aim of generating hard negatives, NSCaching (Zhang et al., 2019) introduces an approach to maintain a cache of high-quality negatives. After evaluating the gradients, NSCaching stores negative triplets with large gradients in head/tail caches. Entity Similarity-based Negative Sampling (ESNS) (Yao et al., 2022) considers semantic similarities among entities when selecting negatives and utilizes a shift-based logistic loss function. Despite addressing the problem of vanishing gradients, dynamic distribution-based sampling methods produce false negatives with a high probability whereas the latent positives reflect large gradients.

3 TuckerDNCaching

This section describes the proposed negative sampling method in detail. First, we experimentally analyze the challenges in generating quality negatives in KGE. Then, we introduce the proposed negative sampling method to address the existing challenges in negative sampling, generating quality negatives.

3.1 Problem definition

A negative sample $(\bar{h}, r, t)$ is difficult to discriminate against a positive sample (h, r, t) when a corrupted entity $\bar{h}$ is semantically equivalent to the original entity h according to the semantics in a knowledge graph. For instance, given the positive $(DaVinci, painted, MonaLisa)$, $(DaVinci, painted, CreationOfAdam)$ is a quality negative candidate as it is semantically correct but factually incorrect. For knowledge representation learning, a such negative is harder to discriminate than discriminating facts $(DaVinci, painted, France)$ and $(DaVinci, painted, Louvre)$ when learning the important semantics of the KG. Next, we define the concept of quality negative based on the above explanation and the explanation provided by Liu et al. (2020).

Definition 1 *(**Quality Negative.**) A quality negative is a semantically meaningful but factually incorrect triplet that is hard to distinguish without referring to the ground truth.*

When capturing semantically meaningful negatives, it is necessary to eliminate frequently appearing false negatives. For instance, when evaluating the importance probability of candidate negatives $(DaVinci, painted, CreationOfAdam)$, $(DaVinci, painted, LadyWithAnErmine)$, and $(DaVinci, painted, TheLastSupper)$, it is important to eliminate true facts such as $TheLastSupper$ and $LadyWithAnErmine$ resulting a semantically meaningful but factually incorrect negative, i.e., $(DaVinci, painted, CreationOfAdam)$, for the positive triplet $(DaVinci, painted, MonaLisa)$.

3.2 Notation

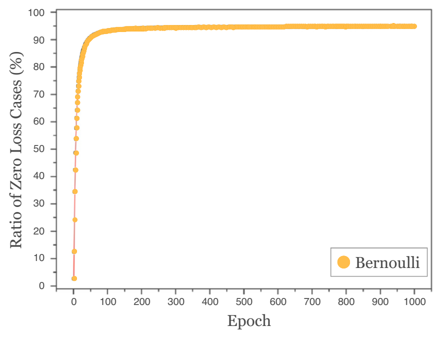
We denote the sets of entities and relations as $\mathcal{E}$ and $\mathcal{R}$, respectively. A fact (edge) in a knowledge graph is represented by a triplet (h, r, t), where h is a head entity in $\mathcal{E}$, t is a tail entity in $\mathcal{E}$, and r is a relationship in $\mathcal{R}$. The set of observed facts in KG is denoted by $\mathcal{G}$, which is equivalent to $\{(h, r, t)\}$. The corrupted head triplet is denoted by $(\bar{h}, r, t)$, and the corrupted tail triplet is denoted by $(h, r, \bar{t})$. Furthermore, we use boldface characters to represent the embedding vectors of h, r, and t, i.e., $\mathbf{h}$, $\mathbf{r}$, and $\mathbf{t}$, respectively.

3.3 Analysis of challenges with generating quality negatives

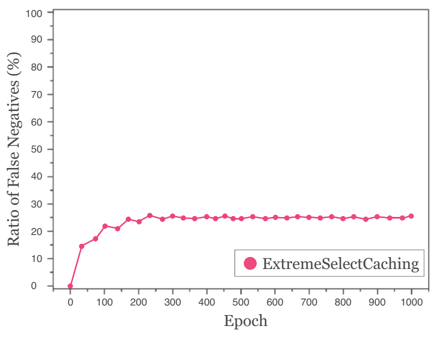
Negatives generated by a fixed distribution-based negative sampling method are effective at the beginning of stochastic training of KGE. However, randomly generated negative triplets are likely to be out of the margin with a high probability after training a few epochs and do not further contribute to KGE learning. This is termed as the **vanishing gradient problem** or **zero loss problem**. We analyzed the zero loss problem by selecting Bernoulli negative sampling as the candidate.

Figure 1(a) shows that the ratio of zero loss cases increases dramatically as the training continues with TransE for the FB15K237 dataset. This result shows that the fixed distribution-based negative sampling methods quickly lead to a vanishing gradient problem. The negative samples effectively contribute only at the first epochs. Therefore, these methods yield a slow convergence and may deviate from optimal embedding learning. Hence, generating negatives with large gradients is important to provide continuous learning of the semantics in KGs.

Even though the dynamic distribution of negatives attempts to solve the problem of vanishing gradients, it introduces the problem of false negatives with high probability compared to the fixed distribution-based negative sampling methods. We analyzed this problem by disabling the false negative filtration step in our proposed TuckerDNCaching negative sampling method. We called it “ExtremeSelectCaching" as it evaluates the importance of all candidates for selecting negatives with large gradients and then caching negatives with large gradients. We evaluated the ratio of false negatives in the negative samples during the training of the “ExtremeSelectCaching" negative sampling method. Figure 1(b) illustrates that dynamic selection introduces the problem of false negatives with high probability as the ratio of false negative samples increases dramatically within a few training epochs. Typically, dynamic distribution-based negative sampling methods evaluate the gradient of a negative using an underlying KGE scoring function. When the distribution of embeddings changes while learning the knowledge representation, the appearance of false negatives increases since they have large gradients. The state-of-the-art dynamic distribution-based negative sampling methods attempt to manage the ratio of false negatives by selecting candidates from a small pool sampled from all the candidates (Zhang et al., 2019; Cai et al., 2018), introducing a trade-off between the false negatives ratio and the quality of the negative candidates. However, as the knowledge graph embedding models learn to discriminate positive triplets against nega-



(a) Ratio of Zero Loss Cases



(b) Ratio of False Negatives

Fig. 1 Experimental analysis: zero loss problem and false negative problem. (a) The ratio of zero loss cases in training FB15K237 by TransE with Bernoulli negative sampling. (b) The ratio of false negatives in ExtremeSelectCaching for FB15K237

tive triplets, false negatives fool the models into losing the actual semantics of entities and relations.

The analysis further suggests that the proposed negative sampling method should generate a negative triplet with a large gradient that is not an apparent false negative when producing a quality negative candidate.

3.4 TuckerDNCaching

Recall the stated challenges in negative sampling, (a) adopting a dynamic distribution of negatives to avoid vanishing gradients, and (b) avoiding false negatives that lead to losing the semantics of the KG. A negative sampling method must be carefully designed to overcome these challenges.

The proposed TuckerDNCaching adopts a dynamic distribution of negative samples when selecting candidate negatives as it is required to avoid the problem of vanishing gradients and enable the underlying KGE model to learn the semantics of the KG continuously. Since quality negative samples are less, we need to ensure that all possible quality negative samples are explored. Hence, TuckerDNCaching models the distribution of all candidates and selects quality negatives. When selecting candidates with large gradients, we refer to the underlying scoring function of the KGE model. However, modeling the distribution for all candidates introduces complexity in executing the steps in TuckerDNCaching. To manage this, we utilize a caching technique that maintains negatives with large gradients for each positive fact and introduce a lazy update procedure for revamping caches that refresh after $\mathcal{N}$ number of epochs later rather than immediately. We aim to maintain two separate caches, i.e., head-cache $\mathcal{H}(t, r)$ that maintains candidates for *head* corruption and tail-cache $\mathcal{T}(h, r)$ that maintains candidates for *tail* corruption. We uniformly sample negatives from the cache efficiently without introducing any bias.

The proposed negative sampling method introduces a Tucker decomposition-based latent relation model to predict and eliminate false negatives from the negative sample space. Modeling the distribution of all candidates and eliminating false negatives ensures that the proposed method explores all quality negatives. The projections of latent relations between entities decrease the possible discrimination for latent positives when the KGE model is learning the KG semantics.

Next, we describe how Tucker decomposition is used to model the KG's latent relation. Then, we provide details on our negative sampling method, TuckerDNCaching.

3.4.1 Tucker decomposition for latent relation modeling

A tensor is a multidimensional array. An Nth-order tensor is an element of the tensor product of N vector spaces, each of which has its own coordinate system. A first-order tensor is a vector, a second-order tensor is a matrix, and tensors of order three or higher are considered higher-order tensors. Interestingly, tensors can be represented compactly in decomposed forms. Several decomposition techniques are available; among them, CP (Hitchcock, 1927) and Tucker (Tucker, 1966) are popular. CP expresses the tensor as a sum of rank one tensors, i.e., a sum of the outer product of vectors. Tucker decomposition is a generalization of CP decomposition. It decomposes the tensor into a small core tensor and factor matrices. For example, the Tucker decomposition of a third-order data tensor $X \in \mathbb{R}^{I\times J\times K}$ can be represented as $X \approx \mathcal{G} \times_1 A \times_2 B \times_3 C$, where $\mathcal{G} \in \mathbb{R}^{X\times Y\times Z}$ is a third-order core tensor, $A \in \mathbb{R}^{I\times X}$, $B \in \mathbb{R}^{J\times Y}$, and $C \in \mathbb{R}^{K\times Z}$ are factor matrices, and $\times_n$ is an n-mode tensor product with a matrix. Knowledge graph data can be represented as a $\{0, 1\}$ valued third order tensor $Y \in \{0, 1\}^{E \times R \times E}$, where E is the total number of entities and R is the number of relations, with $Y_{i,j,k} = 1$ if the relation (i, j, k) is available.

The previous study, i.e., MDNCaching (Madushanka et al., 2022), introduced the idea of eliminating false negatives from the candidate negatives while sampling the candidates from a dynamic distribution of negatives. MDNCaching utilizes the matrix decomposition technique to model a relationship between *head* and *tail* entities, i.e., let h be a set of heads, t be a set of tails, and R be a relation matrix between h and t as $R = R_{|h| \times |t|}$. MDNCaching represents latent relations such that $R \approx H \times T^{\top}$, where H represents the head features, and T represents the tail features. However, the latent relation model lacks semantic soundness as it utilizes a relation matrix that reflects the most probable relationship between an entity pair concerning two feature spaces for head and tail entities. In addition, matrix representation is weak in representing many-to-many relations as KG facts are interpreted in a two-dimensional tensor.

To improve the semantic soundness of the relation representations, we introduce a relation feature space in the latent relation model apart from the entity feature space, modeling KG facts in a three-dimensional tensor with the proposed method. We utilized the Tucker decomposition tensor representation, which is more general and flexible. Given the KG facts $X \in \mathbb{R}^{I,J,K}$, Tucker decomposition outputs a weight tensor $W \in \mathbb{R}^{X,Y,Z}$ and three matrices $E_h \in \mathbb{R}^{I,X}$, $R \in \mathbb{R}^{J,Y}$, $E_t \in \mathbb{R}^{K,Z}$ such that it interprets $X \approx W \times_1 E_h \times_2 R \times_3 E_t$ ($\times_n$ indicates the tensor product along the n^{th} mode).

3.4.2 Proposed framework

Our framework for the proposed negative sampling method is illustrated in Fig. 2 including critical steps for a tail corruption scenario.

The proposed TuckerDNCaching negative sampling method follows six critical steps in generating a quality negative for a given positive triplet and ensures the exploration of all

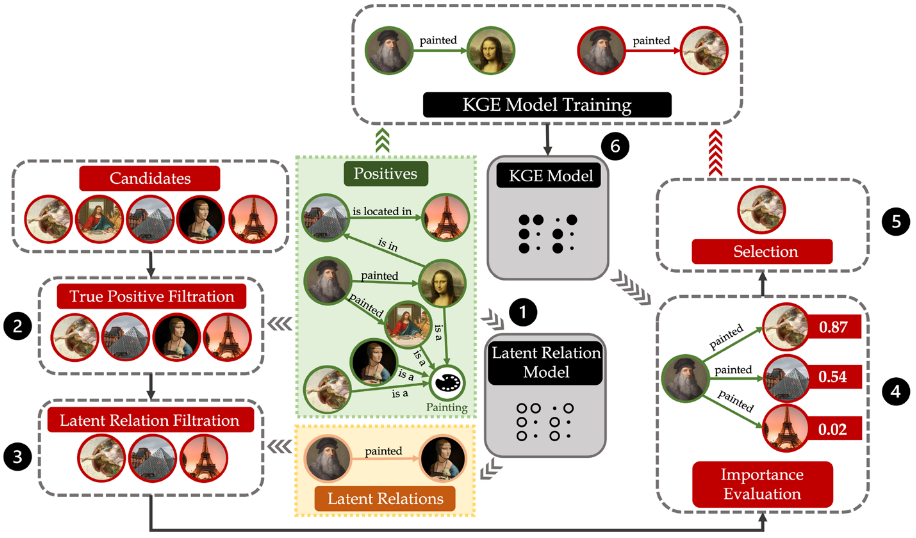


Fig. 2 Critical steps of proposed TuckerDNCaching negative sampling method

possible important and effective negatives to the best. In step 1, the negative sampling method, performs latent relation model training, referring to positives. The significant contribution of the proposed method is eradicating false negatives from the negative sample space modeling latent relations in a three-dimensional tensor. Since the idea of TuckerDNCaching is to model the dynamic distribution of all candidate negatives, the candidate space for a negative sample is initiated with all entities except for the given positive elements. For example, given the positive *(DaVinci, painted, MonaLisa)*, the proposed method initializes the candidate negatives as *(TheCreationOfAdam, TheLastSupper, Louvre, LadyWithAnErmine, Paris)*, where $\mathcal{E}$ = *(DaVinci, MonaLisa, TheCreationOfAdam, TheLastSupper, Louvre, LadyWithAnErmine, Paris)*. In step 2, TuckerDNCaching drops true positives to eliminate all the observed positive facts from the candidate sample space. Since KG consists of one-to-many, many-to-many, and many-to-one relations, it is essential to drop true positives from the candidate negatives. For instance, given the positive *(DaVinci, painted, TheLastSupper)*, candidate negative *TheLastSupper* is removed from the negative sample space. In step 3, TuckerDNCaching drops false negatives, i.e., latent positives, from the candidate negatives predicting possible latent relations from the latent relation model which is trained in step 1. Eliminating false negatives before the importance evaluation for selecting negatives with large gradients is essential since latent positives comprise large gradients. For instance, given the tail corrupted triplet *(DaVinci, painted, ∗)*, the latent relation model predicts *(DaVinci, painted, LadyWithAnErmine)* as a latent fact and removes that from the candidate negatives. To avoid vanishing gradients while exploring quality negatives, TuckerDNCaching introduces an importance probability $p_{imp}(x)$:

$$p_{imp}(x) = \begin{cases} head\ corruption;\ p_{imp}(\bar{h}) = p(\bar{h} \mid (t,r)) = \frac{exp(f(\bar{h},r,t))}{\sum_{h_i \epsilon H_{(t,r)}} exp(f(\bar{h_i},r,t))} \\ tail\ corruption;\ \ p_{imp}(\bar{t}) = p(\bar{t} \mid (h,r)) = \frac{exp(f(h,r,\bar{t}))}{\sum_{t_i \epsilon T_{(h,r)}} exp(f(h,r,\bar{t_i}))} \end{cases} \tag{1}$$

where $H_{(t,r)}$ is head candidate negatives, and $T_{(h,r)}$ is tail candidate negatives. The importance probability $p_{imp}(x)$ samples essential and effective negatives from candidate negatives considering their gradients referring to the underlying scoring function $f(h,r,t)$. A higher $p_{imp}(x)$ reflects that the candidate negative is more effective and important with KGE model learning. In step 4, following the probability $p_{imp}(x)$, the proposed method evaluates the importance probability for all candidate negatives. In step 5, the quality negatives are screened, considering the probability values, and the method then directs the screened negatives, i.e., *(DaVinci, painted, TheCreationOfAdam)*, to KGE model training. In step 6, the typical KGE model training is performed, discriminating positive *(DaVinci, painted, MonaLisa)* against the generated negative, i.e., *(DaVinci, painted, TheCreationOfAdam)*.

However, modeling the distribution of all candidate negatives and selecting quality negatives introduce complexity with execution. Therefore, a caching technique is adopted to handle the execution efficiency. Also, a lazy update procedure is used to evaluate the importance of candidate negatives and update the caches. The integration of latent relation model training, negative cache initialization, and cache update procedure with the existing KGE model training framework is described in Section 3.4.3 while carefully referring to critical steps in the proposed TuckerDNCaching.

3.4.3 Implementation of TuckerDNCaching with KGE training framework

Referring to the typical KGE model training framework, the proposed TuckerDNCaching introduces latent relation model training, negative cache initialization, and cache update steps, which are described in Algorithm 1.

Algorithm 1: TuckerDNCaching-based KG embedding.

```
   Input: Knowledge graph $\mathcal{G} = \{(h, r, t)\}$, and latent relation model $f_r(h, t)$ of $d_r$ dimension with
          parameters $\theta_R$, and score function $f(h, r, t)$ of embedding dimension $d$ with parameters $\theta_E$
   Output: Embedding model with parameters $\theta_E$
1  Initialize latent relation model $f_r(h, t)$ with parameters $\theta_R$, embedding model with parameters $\theta_E$
   randomly, and initialize caches, i.e., head-cache $\mathcal{H}$ and tail-cache $\mathcal{T}$.
2  Train latent relation model for $N_{iterations}$ number of epochs.
3  Loop
4      foreach $(h, r, t) \in \mathcal{G}$ do
5          Index $\mathcal{H}$ by $(t, r)$, i.e.,$\mathcal{H}_{(t,r)}$ and $\mathcal{T}$ by $(h, r)$, i.e.,$\mathcal{T}_{(h,r)}$.
6          Uniformly sample negative candidates from $\bar{h} \in \mathcal{H}_{(t,r)}$ and $\bar{t} \in \mathcal{T}_{(h,r)}$.
7          Select negatives $(\bar{h}, r, \bar{t})$ either as $(\bar{h}, r, t)$ or $(h, r, \bar{t})$ considering the relation cardinality of
           $r$.
8          Update knowledge graph embeddings discriminating $(h, r, t)$ against $(\bar{h}, r, \bar{t})$.
9      end foreach
10     Update cache $\mathcal{H}$ and $\mathcal{T}$ using Algorithm 2;
11 end
```

The parameters of latent relation model (θ_R) and KGE model, (θ_E) are initialized in line-1. Reflecting the steps in Section 3.4.2, the latent relation model training is performed in line-2 following step 1 in Fig. 2. KGE model trains iteratively for a certain number of epochs. Typically, we forge a triplet as a candidate for negatives by replacing either *head* or *tail*. Generated negatives are stored in two separate caches, i.e., head-cache $\mathcal{H}$ (indexed by (t, r)) and tail-cache $\mathcal{T}$ (indexed by (h, r)). When a positive triplet is considered, the corresponding caches that contain candidate negatives, i.e., $\mathcal{H}(r, t)$ and $\mathcal{T}(h, r)$, are indexed in line-5. To carefully select negatives from caches without introducing bias, candidate negatives are uniformly selected from the head-cache $\mathcal{H}_{(\mathrm{t,r})}$ and the tail-cache $\mathcal{T}_{(h,r)}$ in line-6. The final selection of negatives for a respective positive is selected from the negative head candidates or tail candidates considering the relation cardinality in line-7. Referring to the generated negatives, typical KGE model training is performed (step 6) using the underlying scoring function f in line-8. To model the dynamic distribution of candidate negatives and adopt the changes in embeddings, caches are updated using Algorithm 2 in line-10.

The critical steps from 2 to 5 in the proposed framework are implemented in Algorithm 2 as it describes steps in generating quality candidate negatives and storing them in respective caches. When caches are updating, TuckerDNCaching initializes candidates for the head-cache $\mathcal{H}_{(\mathrm{t,r})}$ and the tail-cache $\mathcal{T}_{(h,r)}$ with all entities $\mathcal{E}$ except for the given positive elements in line-2 in Algorithm 2. Reflecting on step 2 in the proposed framework, true positive filtration from candidate negatives is performed in line-3. Latent positive filtration from candidate negatives refers to the predicted latent relations (step 3) is performed in line-4. The importance and the effectiveness of the candidate negatives of true negatives are evaluated on the basis of their gradients using the importance probability $p_{imp}(x)$ (step 4) in line-5. On the basis of the probability values of $p_{imp}(x)$, N_c candidates are selected and stored in respective caches (step 5) in line-6.

Algorithm 2: Cache update.

```
  Input: Knowledge graph $\mathcal{G} = \{(h, r, t)\}$, and latent relation model $f_r(h, t)$ of $d_r$ dimension with
         parameters $\theta_R$, score function $f(h, r, t)$ of embedding dimension $d$ with parameters $\theta_E$, and
         cache size $N_c$
  Output: head-cache $\mathcal{H}$ and tail-cache $\mathcal{T}$
1 foreach $(h,r,t) \in \mathcal{G}$ do
2     Initialize negative candidates for $\mathcal{H}_{(t,r)}$ and $\mathcal{T}_{(h,r)}$.
3     Remove true positives from $\mathcal{H}_{(t,r)}$ and $\mathcal{T}_{(h,r)}$ referring to $\mathcal{G}$.
4     Predict latent relations from $f_r(h, t)$, $\forall \bar{h} \in \mathcal{H}_{(t,r)}$, $\forall \bar{t} \in \mathcal{T}_{(h,r)}$ and drop false negatives from
      $\mathcal{H}_{(t,r)}$ and $\mathcal{T}_{(h,r)}$.
5     Evaluate importance probability $p_{imp}(x)$ considering (1) $\forall \bar{h} \in \mathcal{H}_{(t,r)}$ and $\forall \bar{t} \in \mathcal{T}_{(h,r)}$.
6     Select $N_c$ number of candidates with high $p_{imp}(x)$ from $\mathcal{H}_{(t,r)}$ and $\mathcal{T}_{(h,r)}$.
7 end foreach
```

TuckerDNCaching differentiates itself from the state-of-the-art negative sampling methods from three perspectives. First, by modeling the distribution of all possible candidates, it effectively discovers rare and quality negatives. Second, the latent relation predictions and false negative filtering avoid false negatives in the candidate space. Third, the probability $p_{imp}(x)$ evaluates the importance and the effectiveness of candidate negatives on the basis of a scoring function to avoid vanishing gradients. Since TuckerDNCaching eliminates false negatives, final candidate negatives comprise only quality negatives. In contrast to the previous MDNCaching, the proposed TuckerDNCaching improves the semantic soundness of the latent relations utilizing Tucker decomposition representation introducing an additional relation feature space with a three-dimensional tensor. The proposed method extends the caching technique originally proposed for NSCaching to manage generated negative candidates effectively but addresses the problem of false negatives successfully eradicating the false negatives in negative sample space using a latent relation model. As the proposed method does not depend on the selection of the scoring function, it improves robustness in training models from scratch with fewer parameters than previous dynamic negative sampling methods IGAN, KBGAN, and KSGAN. Even though some negative sample methods introduce loss functions to achieve the best performance, e.g., ESNS introduces a shift-based point-wise logistic loss function, TuckerDNCaching is not biased toward any loss function and provides flexibility.

3.5 Space and time complexity analysis

TuckerDNCaching introduces additional costs with the Tucker decomposition-based latent relation model in Algorithm 1 and negative caches introduced in Algorithm 2 compared to the typical KGE framework. In Algorithm 2, the time complexity of predicting latent relations $f_r(h, \bar{t}))$ and $f_r(\bar{h}, t))$ at step 4 is $O(| \mathcal{E} | d_r)$. Besides, computing the score of candidate entities at step 5 is $O(| \mathcal{E} | d)$. In step 6 Algorithm introduces a time complexity of $O(| \mathcal{E} |)$, which is negligible compared to costs in steps 4 and 5. Thus total time complexity introduces for one triplet is $O(| \mathcal{E} | (d+d_r))$. With the utilization of the lazy update approach in updating cache that refreshes the cache after N number of epochs later rather than immediate, we can optimize the time complexity to $O((| \mathcal{E} | (d + d_r))/(n + 1))$. When considering the space complexity, Algorithm 2 requires $O(| \mathcal{E} | d_r)$ space to predict latent relations while $O(| \mathcal{E} | d)$ to evaluate scores of $| \mathcal{E} |$. Since the algorithm manages negative candidates in different *head* and *tail* caches, Algorithm 2 requires an additional space of $O(| \mathcal{G} | N_c)$ where N_c is cache

Table 1 Comparison with state-of-the-art negative sampling strategies

	Model Parameters	Minibatch Computation	
		Time	Space
Bernoulli (baseline)	$(\mid \mathcal{E} \mid + \mid \mathcal{R} \mid)d$	$O(md)$	$O(md)$
KBGAN	$2(\mid \mathcal{E} \mid + \mid \mathcal{R} \mid)d$	$O(mNd)$	$O(mNd)$
KSGAN	$2(\mid \mathcal{E} \mid + \mid \mathcal{R} \mid)d$	$O(m(N+s)d)$	$O(m(N+s)d)$
NSCaching	$(\mid \mathcal{E} \mid + \mid \mathcal{R} \mid)d$	$O(\frac{m}{n+1}(N+e)d)$	$O(m(N+e)d)$
MDNCaching	$(\mid \mathcal{E} \mid + \mid \mathcal{R} \mid)d + 2(\mid \mathcal{E} \mid)d_r$	$O(\frac{m}{n+1} \mid \mathcal{E} \mid (d+d_r))$	$O(m \mid \mathcal{E} \mid (d+d_r))$
TuckerDNCaching	$(\mid \mathcal{E} \mid + \mid \mathcal{R} \mid)(d+d_r)$	$O(\frac{m}{n+1} \mid \mathcal{E} \mid (d+d_r))$	$O(m \mid \mathcal{E} \mid (d+d_r))$

Model parameters are based on TransE, m is the size of mini-batch, n is the epochs of lazy-update, embedding dimension, latent relation model dimension d_r, cache size N, cache extension size e, selection size s

size, but practically it is smaller than the actual as facts contain many N-1, 1-N, and N-N in $\mathcal{G}$. Comparisons with state-of-the-art are summarized in Table 1.

4 Experiments

We evaluated TuckerDNCaching from three perspectives: (a) effectiveness in addressing the stated challenges in negative sampling, (b) accuracy in latent relation modeling, and (c) accuracy of the link prediction task.

4.1 Experimental setup

4.1.1 Experimental design

We evaluated the performance of our proposed negative sampling method on the link prediction task. Link prediction aims to indicate the occurrence of links in graphs, i.e., predict the missing $head$ (h) or $tail$ (t) entity for a positive triplet (h, r, t) and evaluate the rank of the $head$ and $tail$ entities among all predicted entities. To experimentally evaluate the effectiveness of the proposed method in addressing the stated challenges, i.e., (a) adopting a dynamic distribution of negatives to avoid vanishing gradients and (b) avoiding false negatives among the candidates, we conducted two experiments on each aspect. The proposed TuckerDNCaching was compared with MDNCaching for different relation types to compare the improvements in latent relation modeling. We measured the accuracy of latent relation predictions, evaluating the percentage of correct relation predictions between $head$ and $tail$ in the test dataset. To test the quality of selected negatives, we store candidate negatives for a particular positive triplet and manually compare the negatives' semantics. We compare the link prediction results with the state-of-the-art negative sampling methods to compare the link prediction accuracy. We evaluate the performance of the link prediction task referring to two translational distance-based models (TransE (Bordes et al., 2013), TransD (Ji et al., 2015)) and two semantic matching models (DistMult (Yang et al., 2015), ComplEx (Trouillon et al., 2016)). Details on baseline KGE models are given in Table 2.

4.1.2 Datasets

The experiments were conducted on four standard benchmark datasets, i.e., WN18 (Miller, 1995), WN18RR (Wang et al., 2019), FB15K (Bollacker et al., 2008) and FB15K237 (Toutanova and Chen, 2015), which are widely tested with knowledge graph embedding

Table 2 Scoring functions for triplet (h, r, t) and parameters

	Model	Scoring Function	Parameters
Translational	TransE	$\|h + r - t\|_{1/2}$	$h, r, t \in \mathbb{R}^n$
distance-based	TransD	$\left\|h + w_r w_h^\top h + r - (t + w_r w_t^\top t)\right\|_2^2$	$h, t, w_h, w_t, \in \mathbb{R}^n, r, w_r \in \mathbb{R}^k$
Semantic	DistMult	$h^\top \cdot diag(r) \cdot t$	$h, r, t \in \mathbb{R}^n$
matching-based	ComplEx	$Re(h^\top \cdot diag(r) \cdot \bar{t})$	$h, r, t \in \mathbb{C}^n$

$diag(r)$ constructs diagonal matrix with r

Table 3 Statistics of datasets used in KGE experiments

Dataset	#entity	#relation	#train	#valid	#test
WN18	40,943	18	141,442	5,000	5,000
WN18RR	40,943	11	86,835	3,034	3,134
FB15K	14,951	1,345	483,142	50,000	59,071
FB15K237	14,541	237	272,115	17,535	20,466

related work (Hu et al., 2019; Cai et al., 2018; Zhang et al., 2019; Madushanka et al., 2022).The WN18RR and FB15K237 datasets are challenging and realistic, as they have been constructed by removing inverse-duplicate relations from their origins, i.e., WN18 and FB15K, respectively. The statistics of the datasets are shown in Table 3.

4.1.3 Performance measurement

We considered the "filtered scenario" for our performance evaluation on the link prediction task. Hence, valid entities outscoring the target ones have not considered mistakes and were thus skipped when computing the rank. Furthermore, we evaluated the results on the basis of the following metrics.

- *Mean Rank (MR)* is the average of the obtained ranks, $MR = \frac{1}{\|Q\|}\sum_{q\in Q} q$. A smaller value of MR tends to infer better results. However, since MR is susceptible to outliers, the Mean Reciprocal Rank is widely used.
- *Mean Reciprocal Rank (MRR)* is the average of the inverse of the obtained ranks, $MRR = \frac{1}{\|Q\|}\sum_{q\in Q}\frac{1}{q}$. A higher value of MRR tends to infer better results.
- *Hit@K* is the ratio of predictions for which the rank is equal to or lesser than a threshold k, $Hits@K = \frac{\|\{q\in Q : q \leq K\}\|}{\|Q\|}$. A higher value of Hits@K tends to infer better results.

4.1.4 Optimization and implementation

A knowledge graph embedding model was tuned by minimizing the objective function with the Adam optimizer. First, we adjusted hyper-parameters referring to the Bernoulli sampling method on the basis of MRR. We executed each evaluation up to 1000 epochs and present the best result for MRR. We started our experiments within the following ranges for hyper-parameters: embedding dimension $d \in \{50, 100, 250, 1000\}$, learning rate $\eta \in \{0.0005, 0.005, 0.05, 0.5\}$, and margin value $\gamma \in \{1, 2, 3, 4, 5\}$, which were optimized for the best performance. In addition to the typical hyper-parameters related to KGE model training, the proposed method introduces three new parameters: latent model training epochs ($N_{iterations} = 200\ epochs$), latent model dimension ($d_r = 50$), and cache size N_c. After every 20 epochs, caches were updated. We selected cache size N_c to be 250. We implemented the proposed negative sampling method, i.e., TuckerDNCaching, on top of the TorchKGE (Boschin, 2020) Python module with the PyTorch framework and executed it on an NVIDIA RTX A6000 GPU.

4.2 Results

4.2.1 Impact of negatives with large gradients

We compared the distributions of negative triplets for TuckerDNCaching and the baseline Bernoulli negative sampling method to evaluate the impact of negatives with large gradients. We measured the complementary cumulative distribution function (CCDF) $F_D(x) = P(D \geq x)$ to show the proportion of negative triplets that satisfy $D \geq x$ where $D_{(h^*,r,t^*)} = f(h^*, r, t^*) - f(h, r, t)$, and Fig. 3 shows the results.

When the minus distance between the scores of a positive sample and its negative sample, i.e., $D_{(h^*,r,t^*)}$, was smaller than a minus margin $-\gamma(\gamma > 0)$, the distance between the positive and negative samples was large enough. Hence, the negative sample contributed a zero gradient to the loss function. Quality negative samples are those with $D > -\gamma$. Figure 3(a) shows that the distribution of negative triplets with large gradients for Bernoulli tended to reduce as the learning continued, leading to a vanishing gradient problem. However, Fig. 3(b) shows that the dynamic selection approach in TuckerDNCaching managed to overcome the vanishing gradient problem as it generated negatives with large gradients to support the KGE learning. We can see that TuckerDNCaching tended to reduce negatives that satisfied $D > -\gamma$ until epoch 50. After that, it kept producing negatives with large gradients, supporting the KGE model in continuously learning the semantics of KG.

4.2.2 Impact of false negative elimination

Our primary research contribution in TuckerDNCaching is to model latent relations precisely, drop false negatives from the negative candidate space, and enhance the quality of candidates. The analysis in Section 3.3 describes the challenge with the existence of false negatives when a negative sampling method tends to model the dynamic distribution of all possible negatives. Therefore, it is essential to filter false negatives from candidate negatives. To compare the impacts of false negatives' existence and nonexistence, we compared the link prediction results of the proposed TuckerDNCaching, previous MDNCaching, and ExtremeSelectCaching (introduced in Section 3.3).

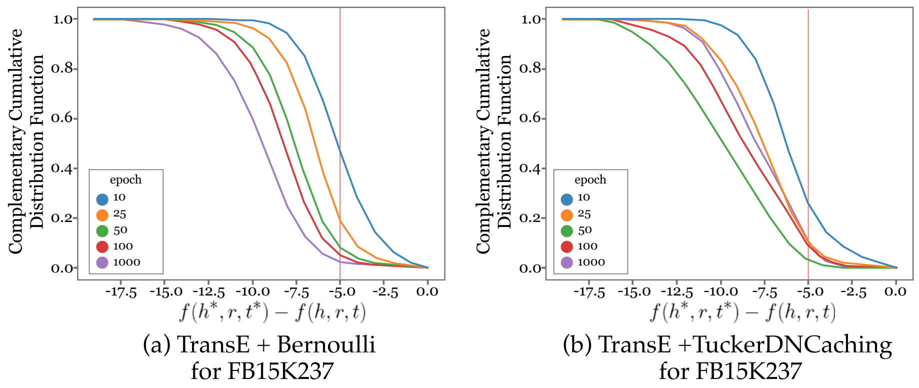
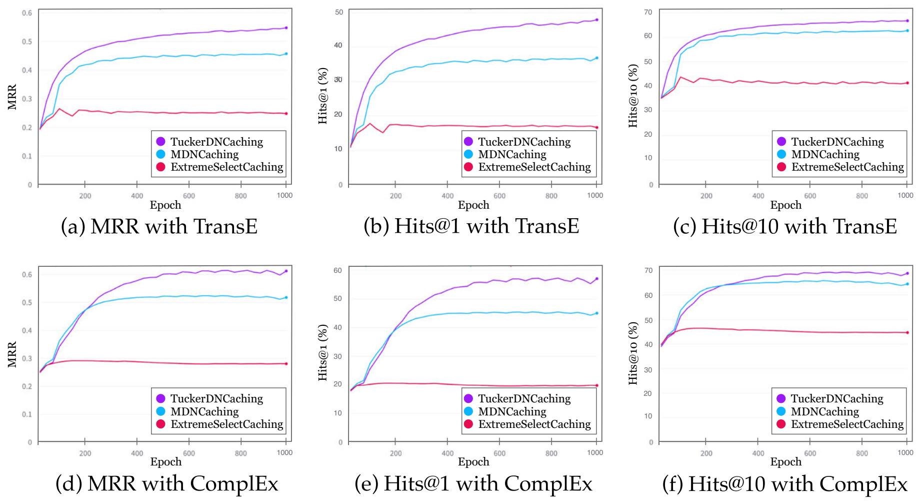


Fig. 3 Comparison of distributions of negative triplets on FB15K237 for TransE with Bernoulli and TuckerDNCaching negative sampling methods. The Red dashed line shows where margin γ (=5) lies in (a) and (b). (a) Distribution of negative triplets with Bernoulli negative sampling method for different epochs. (b) Distribution of negative triplets with TuckerDNCaching negative sampling method for different epochs



(a) MRR with TransE (b) Hits@1 with TransE (c) Hits@10 with TransE

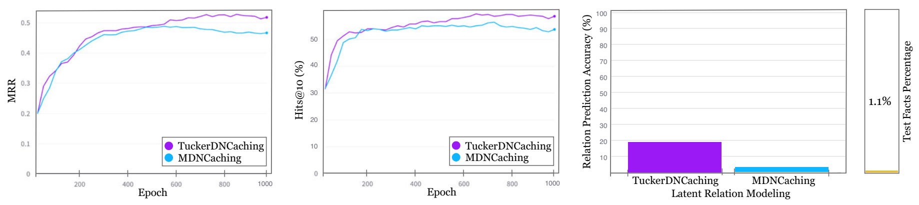
(d) MRR with ComplEx (e) Hits@1 with ComplEx (f) Hits@10 with ComplEx

Fig. 4 Performance comparison on link prediction for TuckerDNCaching, MDNCaching, and ExtremeSelectCaching on FB15K237

Figure 4 shows the results of the link prediction task describing the impact of false negative filtration. Results on FB15K237 data show that the false negative filtration effectively enhanced link predictions for TransE and ComplEx. The enhancement to the latent relation modeling by utilizing the Tucker decomposition is reflected in Fig. 4, as it substantially enhanced the link prediction results for MRR and Hits@1 compared with MDNCaching. It is important to note that dynamic selection does not continuously support learning as the negative space contains positives, and the results reflect the importance of identifying false negatives.

4.2.3 Visualization of complements for relation types

To evaluate the capability in modeling many-to-many relations in the proposed TuckerDNCaching, we compared the link prediction results for many-to-many relations on the FB15K237 dataset. We compared the link prediction results for TuckerDNCaching with MDNCaching, and the results are described in Fig. 5. The improvement in accuracy for predicting many-to-many latent relations reflects that TuckerDNCaching overcomes the representation problem with MDNCaching by introducing a relation feature space with the



(a) MRR for Many-to-Many Relations (b) Hits@10 for Many-to-Many Relations (c) Many-to-Many Relations Accuracy

Fig. 5 Performance comparison on link prediction for TuckerDNCaching and MDNCaching considering many-to-many relations in FB15K237 for ComplEx w.r.t latent relation model accuracy

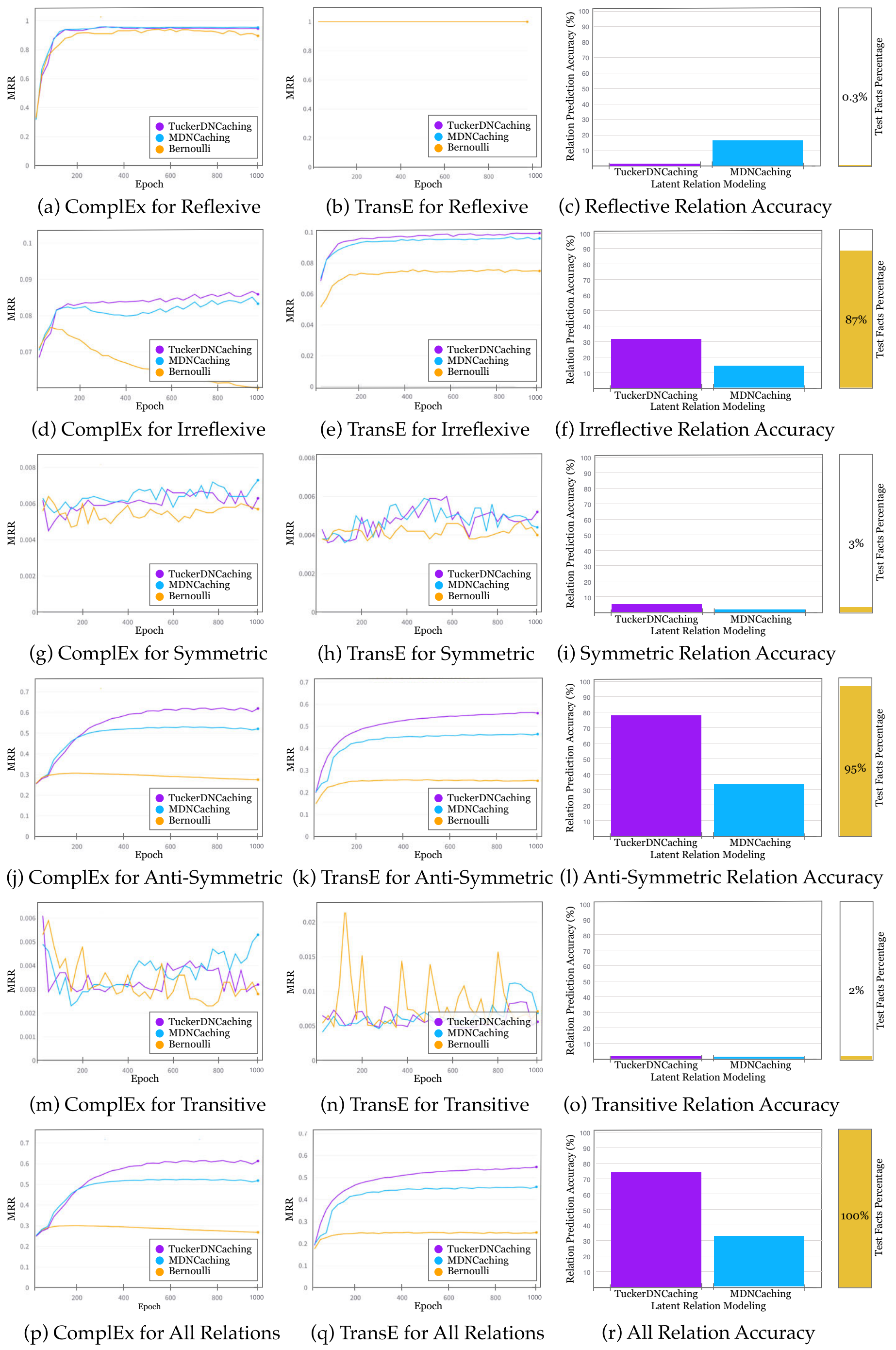


Fig. 6 MRR performance comparison on link prediction for TuckerDNCaching, MDNCaching, and Bernoulli negative sampling methods with relation properties for FB15K237 with latent relation model accuracy

Tucker decomposition representation technique. As TuckerDNCaching improves the latent relation modeling for many-to-many relations, it performs better filtration of latent facts and enhances the link prediction.

To further evaluate the improvements in latent relation modeling regarding MDNCaching, we compared the link prediction performance for different relation types on FB15K237. We separated test facts into pails on the basis of the properties of their relations. When a relation retains multiple relation properties, the corresponding test facts are moved to all the related pails. For reference, we consider complete test facts as “All relations" as they contain facts of all relation types. We evaluated the accuracy of the latent relation model for the test dataset of FB15K237 and analyzed the MRR performance. We considered the following relation properties:

- Reflexivity: $r \in R$ is reflexivity if $\forall (h, r, t) \in \mathcal{G}_{train}$, $(h, r, h) \in \mathcal{G}_{train}$ too.
- Irreflexivity: $r \in R$ is irreflexive if $\forall e \in E$ $(e, r, e) \notin \mathcal{G}_{train}$.
- Symmetry: $r \in R$ is symmetric if $\forall (h, r, t) \in \mathcal{G}$, $(t, r, h) \in \mathcal{G}$ too.
- Anti-symmetry: $r \in R$ is anti-symmetric if $\forall (h, r, t) \in \mathcal{G}$, $(t, r, h) \notin \mathcal{G}$.
- Transitivity: $r \in R$ is transitive if $\forall$ pair of facts $(h, r, x) \in \mathcal{G}$ and $(x, r, t) \in \mathcal{G}$, $(h, r, t) \in \mathcal{G}$ as well.

Figure 6 illustrates the comparison results on the link prediction task for ComplEx and TransE with the FB15K237 dataset with latent relation model prediction accuracies. When comparing the latent relation model accuracy for different relation types, one can see that TuckerDNCaching performed better in modeling latent relations except for the reflective relation type. The results evidence that the improvements in latent relation modeling lead to better performance on the link prediction task. For instance, when considering the accuracy of the latent relation model for irreflective and anti-symmetric relations, respective graphs reflect noticeable enhancements to the corresponding link prediction performance, referring to the baseline Bernoulli negative sampling method. Interestingly, irrespective of the negative sampling method, the underlying KGE models tended to successfully learn semantics for reflective relation. For the symmetric relation, we observed slight improvements in link predictions for ComplEx and TransE compared with the baseline Bernoulli negative sampling method. On the other hand, the results reflect no evident performance improvement compared with the baseline Bernoulli sampling method when the latent relation modeling for the transitive relation was inadequate. When considering the representation of relations, introducing the relation feature space with Tucker decomposition representation enhanced the latent relation modeling compared with the matrix decomposition-based MDNCaching. The results reflect that TuckerDNCaching enhances link predictions by eliminating possible false negatives.

4.2.4 Examples of negative triplets

To test the quality of the negative samples generated by the proposed TuckerDNCaching, we visualized the changes to the entities in the cache. Following IGAN (Wang et al., 2018), we analyzed the negative candidates sampled for FB13 as the triplets are more intuitive than WN18RR and FB15K237. We compared the negative candidates for the positive $(panorama, profession, actor)$ following NSCaching (Zhang et al., 2019). We considered tail corruption, i.e., $(panorama, profession, ?)$, and Table 4 describes the first ten entries in different epochs. The results show that the initial negative candidates were meaningless, e.g., $marc\_mitscher$ and $david\_farrar$. However, as the learning continued, TuckerDNCaching gradually changed negative candidate entities to human jobs, e.g., $bookmaker$,

Table 4 Example of negative entities in cache on FB13. Each line reflects first 10 sampled entities from *tail-cache* of a positive fact ($manorama$, $profession$, $actor$) in different epochs

epoch	First 10 candidates in cache
0	marc_mitscher, ivan_ii_of_russia, david_farrar, kim_philby, luigi_galleani, laval_quebec, patriarch_photios_i_of_constantinople, josias_hoffman, cynthiana, james_anthony_murphy
100	**naval_officer**, **roman_emperor**, **cardinal**, **pastor**, **logician**, jean_joseph_vaudechamp, john_winthrop_the_younger, joseph_schrembs, howard_culver, ken_barrington
250	**bookmaker**, **talent_agent**, lord_francis_douglas, **war_correspondent**, **archbishop**, **mechanic**, **sales_manager**, **jazz_composer**, **art_historian**, **pitcher**
500	**nobleman**, **game_show_host**, **editor_in_chief**, **televangelist**, **ballerina**, **prime_minister**, **hip_hop_production**, **saint**, **jazz_composer**, **university_president**

Bold entities are correct type matches for relation *profession* in FB13 in each epoch

$talent\_agent$, that are semantically meaningful with ($panorama$, $profession$, ?). This reflects the capability of our negative sampling method to generate negative triplets that are semantically meaningful and effective in discriminating with positives.

4.2.5 Link prediction

To carefully compare TuckerDNCaching with MDNCaching, we evaluated the results on the link prediction task considering MRR, Hits@1, Hits@3, and Hits@10 for the WN18RR and FB15K237 datasets since they are more challenging, and the results are summarized in Table 5. The results on link prediction reflect that the TuckerDNCaching negative sampling method overall enhanced the link prediction and, most interestingly, it enhanced Hits@1 and MRR substantially, referring to the previous MDNCaching. When considering the MRR results for the WN18RR dataset, TuckerDNCaching outperformed MDNCaching for all KGE models. TuckerDNCaching outperformed MDNCaching except for TransD with the FB15K237 dataset. Interestingly TuckerDNCaching achieved a 19.72% improvement for TransE, while it achieved an 18.08% improvement for ComplEx compared with MDNCaching. This improvement follows with Hits@1 as it is intuitive with MRR. Hits@3 reflects the best results for all baseline KGE models with WN18RR and FB15K237 except for TransD with WN18RR. When considering the link prediction task requirement, improvements to Hits@1 and MRR reflect how well the embedding model learns the semantics of the KG. The results evidence that TuckerDNCaching enhances link prediction by enhancing the semantic soundness of the latent relation model and improving the quality of the generated negatives compared with the previous MDNCaching, overcoming the representation issues.

Following the approach in related work (Zhang et al., 2019; Hu et al., 2019; Madushanka et al., 2022), we compared the link prediction results with the state-of-the-art negative sampling methods and compared the results considering MR, MRR, and Hits@10. Table 6 summarizes the performance comparison for link prediction. We compared results with state-of-the-art negative sampling methods regarding the reported performance comparison in NSCaching (Zhang et al., 2019) for Bernoulli, KBGAN, and NSCaching concerning train-

Table 5 Comparison of MRR, Hits@1, Hits@3, Hits@10 on WN18RR and FB15K237 datasets between MDNCaching and TuckerDNCaching

Score function	Negative Sampling Strategy	WN18RR				FB15k237			
		MRR	Hits@1	Hits@3	Hits@10	MRR	Hits@1	Hits@3	Hits@10
TransE	MDNCaching	0.2390	4.40	40.10	53.20	0.4575	36.90	49.83	62.77
	TuckerDNCaching	**0.2563**	**5.71**	**43.14**	**53.83**	**0.5477**	**47.99**	**58.40**	**66.77**
	Δ	7.24%	29.77%	7.58%	1.18%	19.72%	30.05%	17.20%	6.37%
TransD	MDNCaching	0.1737	**0.37**	**29.48**	48.36	**0.2683**	**15.03**	29.95	46.43
	TuckerDNCaching	**0.1873**	0.29	29.23	**48.76**	0.2559	14.05	**31.01**	**47.34**
	Δ	7.83%	-21.62%	-0.85%	0.83%	-4.63%	-6.52%	3.54%	1.96%
DistMult	MDNCaching	0.3921	32.39	42.65	**51.37**	0.2726	18.75	30.05	44.40
	TuckerDNCaching	**0.4181**	**37.80**	**43.49**	49.22	**0.2881**	**20.11**	**31.85**	**46.25**
	Δ	6.63%	16.70%	1.97%	-4.19%	5.69%	7.25%	5.99%	4.17%
ComplEx	MDNCaching	0.4729	44.13	48.80	**54.05**	0.5243	45.39	56.10	65.82
	TuckerDNCaching	**0.4961**	**48.17**	**50.34**	52.27	**0.6191**	**57.89**	**63.91**	**69.25**
	Δ	4.91%	9.15%	3.16%	-3.29%	18.08%	27.54%	13.92%	5.21%

Table 6 Comparison of state-of-the-art negative sampling methods on WN18, WN18RR, FB15K and FB15K237 datasets

Score function	Negative Sampling method	WN18			WN18RR			FB15K			FB15K237		
		MRR	MR	Hits@10	MRR	MR	Hits@10	MMR	MR	Hits@10	MMR	MR	Hits@10
TransE	Bernoulli	0.5001	**249**	94.13	0.1784	3924	45.09	0.4951	65	77.37	0.2556	197	41.89
	KBGAN	0.6606	301	94.80	0.1808	5356	43.24	0.3771	335	72.67	0.2926		46.59
	KSGAN	0.7910	—	**95.40**	0.2120	—	48.70	—	—	—	0.2790	—	46.20
	NSCaching	0.7818	**249**	94.63	0.2002	4472	47.83	0.6391	62	80.95	0.2993	**186**	47.64
	MDNCaching	0.8036	430	94.91	0.2390	**3054**	53.20	0.8404	**51**	92.25	0.4575	226	62.77
	TuckerDNCaching	**0.8084**	454	95.06	**0.2563**	3062	**53.83**	**0.8477**	**51**	**92.45**	**0.5477**	223	**66.77**
TransD	Bernoulli	0.5093	**256**	94.61	0.1901	3555	46.41	0.4529	63	76.55	0.2451	**188**	42.89
	KBGAN	0.5950	332	94.68	0.1875	4083	46.41	0.3151	184	69.77	0.2465	825	44.40
	KSGAN	**0.8140**	—	**95.20**	**0.2200**	—	47.90	—	—	—	0.2800	—	46.50
	NSCaching	0.7994	286	95.16	0.2013	**3104**	48.39	**0.6415**	**58**	81.32	**0.2863**	189	**47.85**
	MDNCaching	0.6535	349	94.66	0.1737	4477	48.36	0.5387	86	80.59	0.2683	354	46.43
	TuckerDNCaching	0.6893	328	94.70	0.1873	4225	**48.76**	0.6192	75	**83.20**	0.2559	280	47.34
DistMult	Bernoulli	0.7918	862	93.38	0.3964	7420	45.25	0.5376	102	78.69	0.2491	280	42.03
	KBGAN	0.7275	794	93.08	0.2039	11351	29.52	0.4227	321	64.35	0.2272	276	39.91
	KSGAN	—	—	—	—	—	—	—	—	—	—	—	—
	NSCaching	**0.8306**	827	93.74	0.4128	7708	45.45	**0.7448**	**81**	**83.91**	0.2834	**273**	45.56
	MDNCaching	0.7928	**368**	95.24	0.3921	**2946**	**51.37**	0.6097	126	81.29	0.2726	283	44.40
	TuckerDNCaching	0.8001	380	**95.32**	**0.4181**	3910	49.22	0.6094	118	81.76	**0.2881**	276	**46.25**

Table 6 continued

Score function	Negative Sampling method	WN18			WN18RR			FB15K			FB15K237		
		MRR	MR	Hits@10	MRR	MR	Hits@10	MMR	MR	Hits@10	MMR	MR	Hits@10
ComplEx	Bernoulli	0.9115	808	94.39	0.4431	**4693**	51.77	0.6253	138	80.72	0.2596	238	43.54
	KBGAN	0.7233	966	85.81	0.3180	7528	35.51	0.5002	294	76.10	0.1910	881	32.07
	KSGAN	0.9280	—	95.00	0.4060	—	45.60	—	—	—	0.2680	—	44.00
	NSCaching	0.9355	1072	93.98	0.4463	5365	50.89	0.7995	94	86.28	0.3021	**221**	48.05
	MDNCaching	0.9494	**744**	95.68	0.4729	5312	**54.05**	0.9201	**80**	94.94	0.5243	816	65.82
	TuckerDNCaching	**0.9551**	790	**95.75**	**0.4961**	8792	52.27	**0.9363**	121	**95.05**	**0.6191**	757	**69.25**

Note that results for the MR metric and all metric results for the DistMult KGE model for all datasets are not available for KSGAN. Metric results for all KGE models for the FB15K dataset are also not available for KSGAN as the original did not include them ing from scratch. KSGAN (Hu et al., 2019) was compared with the reported performances. We evaluated the link prediction results for MDNCaching as it is the reference model for the proposed TuckerDNCaching.

When considering the results for translational distance-based models, it is evident that the proposed negative sampling method achieved a substantial improvement for all datasets; achieving the best values for MRR or Hits@10 which provides meaningful insights into the improvements. TransE reflects improvement for all the datasets and TransD reflects the best results for WN18RR and FB15K while achieving competitive results with WN18 and FB15K237. When evaluating results for semantic matching-based KGE models, we observed that the proposed method outperformed the state-of-the-art negative sampling methods except with DistMult for FB15K dataset. Interestingly, comparing the link prediction results for the datasets which do not have any inverse relation test leakage, i.e. WN18RR, and FB15K237, the proposed TuckerDNCaching improves the link prediction results substantially. On the other hand, even though TuckerDNCaching obtains the best results against the state-of-the-art negative sampling strategy for FB15K and WN18 datasets, MRR results reflect that the data leakage in these datasets impacts latent relation modeling and quality negative generation.

The link prediction results with the benchmark datasets show that the proposed negative sampling methods enhanced the KG embeddings by generating quality negatives. The substantial improvements in MRR and Hits@10 reflect that TuckerDNCaching successfully overcomes the stated challenges with negative generation and enhances latent relation modeling with false negative filtration while modeling the dynamic distribution of all negative candidates.

5 Conclusion

In this paper, we proposed TuckerDNCaching, an extension of MDNCaching, improving latent relation modeling. The TuckerDNCaching negative sampling method addresses the problem of false negatives by reducing latent positives predicted through the Tucker decomposition approach, replacing the previous matrix decomposition approach in MDNCaching. The proposed method effectively manages separate caches for *head* and *tail* candidates containing quality negatives, thus addressing the challenges in negative sampling. We experimentally evaluated TuckerDNCaching on four widely used datasets and four scoring functions covering translational distance and semantic matching models. The results of the link prediction task reflect a substantial enhancement with TransE, Dismult, and ComplEx KGE models. Notably, the ComplEx and TransE KGE models with TuckerDNCaching improved the link prediction for all four datasets. We extended our experiment by separately analyzing the effect of dynamic sampling and false negative filtration. In addition, we experimented with complements of the latent relation model for different relation properties and compared the link prediction results for the properties. The empirical results reflect that TuckerDNCaching effectively enhanced knowledge graph embedding generating quality negatives that support the KGE models to learn important semantics of the KG. Also, experimental results evidence that TuckerDNCaching gains faster and more effective model training which achieves better performance with fewer epochs. Possible enhancements to latent relation modeling and effective execution will continue as our future work[1].

[1] https://github.com/ichise-laboratory/tuckerdncaching

Author Contributions Conceptualization, all authors; methodology, T.M.; experimental designs, all authors; model development and writing - original draft preparation, T.M.; writing - review, and editing, all authors.

Funding Not Applicable.

Data Availability Data transparency. The source code is available from GitHub

Declarations

Conflicts of interest The authors declare that they have no competing interests.

Ethical approval and consent to participate Not Applicable.

Consent for publication Not Applicable.

References

Ahrabian, K., Feizi, A., Salehi, Y., et al (2020) Structure aware negative sampling in knowledge graphs. In: Proceedings of Conference on Empirical Methods in Natural Language Processing. ACL. pp 6093–6101. https://doi.org/10.18653/v1/2020.emnlp-main.492

Bollacker, K., Evans, C., Paritosh, P., et al (2008) Freebase: A collaboratively created graph database for structuring human knowledge. In: Proceedings of ACM SIGMOD International Conference on Management of Data. ACM SIGMOD '08. pp 1247–1250. https://doi.org/10.1145/1376616.1376746

Bordes A., Usunier, N., Garcia-Durán, A., et al (2013) Translating embeddings for modeling multi-relational data. In: Proceedings of International Conference on Neural Information Processing Systems. Curran Associates Inc., NIPS' 13. pp 2787–2795

Boschin, A., (2020) TorchKGE: Knowledge graph embedding in python and pytorch. CoRR abs/2009.02963. 2009.02963

Cai, L., Wang, W.Y., (2018) KBGAN: Adversarial learning for knowledge graph embeddings. In: Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics. ACL. pp 1470–1480. https://doi.org/10.18653/v1/N18-1133

Chenchen, G., Chunhong, Z., Han, X., et al. (2019). AWML: adaptive weighted margin learning for knowledge graph embedding. *Journal of Intelligent Information Systems, 53*, 167–197. https://doi.org/10.1007/s10844-018-0535-2

Ebisu, T., Ichise, R., (2018) TorusE: Knowledge graph embedding on a lie group. In: Proceedings of AAAI Conference on Artificial Intelligence. pp 1819–1826

Goodfellow, I., Pouget-Abadie, J., Mirza, M., et al (2014) Generative adversarial nets. In: Advances in Neural Information Processing Systems, vol 27. Curran Associates Inc. pp 2672–2680

Hitchcock, F. L. (1927). The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics, 6*, 164–189.

Hu, K., Liu, H., Hao, T., (2019) A knowledge selective adversarial network for link prediction in knowledge graph. In: Proceedings of Conference on Natural Language Processing and Chinese Computing. Springer. pp 171–183. https://doi.org/10.1007/978-3-030-32233-5_14

Ji, G., He, S., Xu, L., et al (2015) Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of International Joint Conference on Natural Language Processing. ACL. pp 687–696. https://doi.org/10.3115/v1/P15-1067

Kanojia, V., Maeda, H., Togashi, R., et al (2017) Enhancing knowledge graph embedding with probabilistic negative sampling. In: Proceedings of International Conference on World Wide Web Companion. WWW, WWW '17 Companion. pp 801–802. https://doi.org/10.1145/3041021.3054238

Krompaß, D., Baier, S., Tresp, V., (2015) Type-constrained representation learning in knowledge graphs. In: TProceedings of International Workshop on the Semantic Web. Springer. pp 640–655

Liu, H., Hu, K., Wang, F. L., et al. (2020). Aggregating neighborhood information for negative sampling for knowledge graph embedding. *Neural Computing and Applications, 32*, 17637–17653. https://doi.org/10.1007/s00521-020-04940-5

Madushanka, T., Ichise, R., (2022) MDNCaching: A strategy to generate quality negatives for knowledge graph embedding. In: Proceedings of International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer. pp 877–888. https://doi.org/10.1007/978-3-031-08530-7_74

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communication of the ACM, 38*(11), 39–41. https://doi.org/10.1145/219717.219748

Nickel, M., Tresp, V., Kriegel, H.P., (2011) A three-way model for collective learning on multi-relational data. In: Proceedings of International Conference on Machine Learning. Omnipress, ICML'11. pp 809–816

Siheng, Z., Zhengya, S., & Wensheng, Z. (2020). Improve the translational distance models for knowledge graph embedding. *Journal of Intelligent Information Systems, 55*, 445–467. https://doi.org/10.1007/s10844-019-00592-7

Sun, Z., Deng, Z., Nie, J., et al (2019) RotatE: Knowledge graph embedding by relational rotation in complex space. In: Proceedings of International Conference on Learning Representations

Toutanova, K., Chen, D., (2015) Observed versus latent features for knowledge base and text inference. In: Proceedings of Workshop on Continuous Vector Space Models and their Compositionality. ACL. pp 57–66. https://doi.org/10.18653/v1/W15-4007

Trouillon, T., Welbl J, Riedel S, et al (2016) Complex embeddings for simple link prediction. In: Proceedings of International Conference on Machine Learning. JMLR.org, ICML'16. pp 2071–2080

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika, 31*, 279–311. https://doi.org/10.1007/BF02289464

Wang, P., Li, S., Pan, R., (2018) Incorporating GAN for negative sampling in knowledge representation learning. In: Proceedings of AAAI Conference on Artificial Intelligence. AAAI Press, AAAI'18/IAAI'18/EAAI'18. pp 2005–2012

Wang, Y., Ruffinelli, D., Gemulla, R., et al (2019) On evaluating embedding models for knowledge base completion. In: Proceedings of Workshop on Representation Learning for NLP. ACL. pp 104–112. https://doi.org/10.18653/v1/W19-4313

Wang, Z., Zhang, J., Feng, J., et al (2014) Knowledge graph embedding by translating on hyperplanes. In: Proceedings of AAAI Conference on Artificial Intelligence. AAAI Press, AAAI'14. pp 1112–1119

Xie, Q., Ma, X., Dai, Z., et al (2017) An interpretable knowledge transfer model for knowledge base completion. In: Proceedings of Annual Meeting of the Association for Computational Linguistics). ACL. pp 950–962. https://doi.org/10.18653/v1/P17-1088

Yang, B., Yih, W., He, X., et al (2015) Embedding entities and relations for learning and inference in knowledge bases. In: Proceedings of International Conference on Learning Representations

Yao, N., Liu, Q., Li, X., et al (2022) Entity similarity-based negative sampling for knowledge graph embedding. In: Proceedings of Pacific Rim International Conference on Artificial Intelligence. Springer. pp 73–87

Zhang, Y., Yao, Q., Shao, Y., et al (2019) NSCaching: Simple and efficient negative sampling for knowledge graph embedding. In: Proceedings of IEEE International Conference on Data Engineering. IEEE. pp 614–625