Briefings in Bioinformatics, 00(0), 2020, 1–16

doi: 10.1093/bib/bbaa256
Problem solving protocol

Drug–drug interaction prediction with Wasserstein Adversarial Autoencoder-based knowledge graph embeddings

Yuanfei Dai, Chenhao Guo, Wenzhong Guo and Carsten Eickhoff

Corresponding authors: Carsten Eickhoff, Center for Biomedical Informatics, Brown University, Providence, RI, USA. Tel: +86 13110525162; Fax: +86 0591-22865158; E-mail: carsten@brown.edu; Wenzhong Guo, College of Mathematics and Computer Sciences, Fuzhou University, Fujian, China. Tel: +86 13110525162; Fax: +86 0591-22865158; E-mail: guowenzhong@fzu.edu.cn

Abstract

An interaction between pharmacological agents can trigger unexpected adverse events. Capturing richer and more comprehensive information about drug–drug interactions (DDIs) is one of the key tasks in public health and drug development. Recently, several knowledge graph (KG) embedding approaches have received increasing attention in the DDI domain due to their capability of projecting drugs and interactions into a low-dimensional feature space for predicting links and classifying triplets. However, existing methods only apply a uniformly random mode to construct negative samples. As a consequence, these samples are often too simplistic to train an effective model. In this paper, we propose a new KG embedding framework by introducing adversarial autoencoders (AAEs) based on Wasserstein distances and Gumbel-Softmax relaxation for DDI tasks. In our framework, the autoencoder is employed to generate high-quality negative samples and the hidden vector of the autoencoder is regarded as a plausible drug candidate. Afterwards, the discriminator learns the embeddings of drugs and interactions based on both positive and negative triplets. Meanwhile, in order to solve vanishing gradient problems on the discrete representation—an inherent flaw in traditional generative models—we utilize the Gumbel-Softmax relaxation and the Wasserstein distance to train the embedding model steadily. We empirically evaluate our method on two tasks: link prediction and DDI classification. The experimental results show that our framework can attain significant improvements and noticeably outperform competitive baselines.
Supplementary information: Supplementary data and code are available at https://github.com/dyf0631/AAE_FOR_KG.

Key words: drug–drug interaction; knowledge graph embedding; adversarial learning; Wasserstein distance.

Introduction

For optimal therapeutic effect, it is often necessary to take advantage of drug combinations. However, the intended efficacy of a drug may be changed substantially when co-administered alongside another agent. Formally, drug–drug interactions (DDI) are pharmacological interactions between drug ingredients that can alter the function of drugs, cause adverse drug reactions (ADR) and even medical malpractice [1]. While ideally we would like to discover all possible interactions between drugs during clinical trial, some unrecognized interactions may only be revealed after the drugs are approved for clinical use. ADRs cause roughly 100 000 fatalities [2] and 74 000 emergency room visits in

Yuanfei Dai is a PhD student of the College of Mathematics and Computer Sciences, Fuzhou University, China. He is now a visiting scholar at Brown Center for Biomedical Informatics, Brown University, USA. His research interests involve knowledge graph, bioinformatics and machine learning.
Chenhao Guo received his B.S. degree from Fuzhou University. His research interests involve knowledge graph and machine learning.
Wenzhong Guo is a professor of the College of Mathematics and Computer Sciences, Fuzhou University, China. His research interests involve natural language processing, machine learning and complex network algorithm.
Carsten Eickhoff is an assistant professor of Brown Center for Biomedical Informatics, Brown University, USA. His research interests involve bioinformatics, machine learning, clinical text mining and patient-centric information retrieval.
Submitted: 9 March 2019; **Received (in revised form):** 25 February 2020

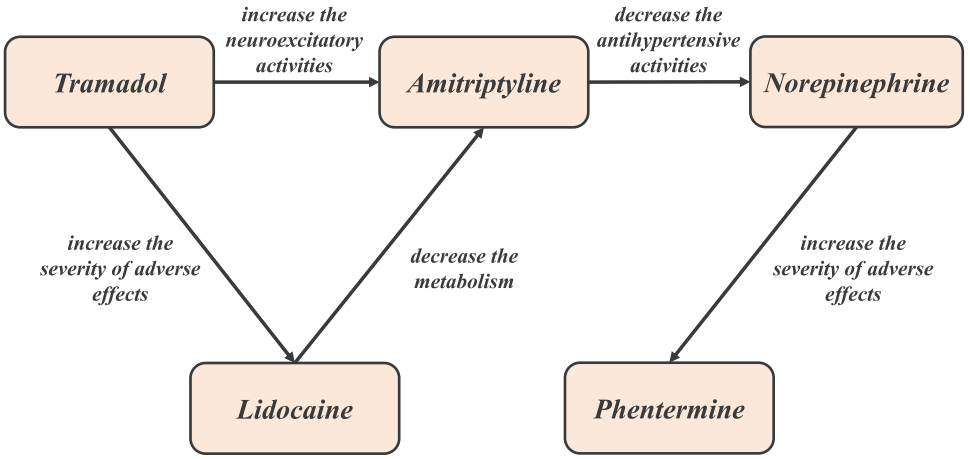

Fig. 1. A simple instance of a DDI KG.

the United States, annually [3]. For instance, acetylsalicylic acid (ASA), also known as aspirin, is a commonly used drug for the treatment of fever and pain, which has both anti-inflammatory and antipyretic effects. However, when ASA is combined with 1-benzylimidazole, the risk or severity of hypertension can be increased. To alleviate these risks and improve quality of care, large-scale and reliable DDI prediction becomes a key task in clinical practice.

To date, various DDI prediction approaches have been proposed to solve this issue. There are some examples from the fields of pharmacogenomics and pharmacology including [4, 5]. However, these methods can only handle a limited range of DDI cases because of their dependency on clinical and laboratory data. Besides, this kind of method requires many characteristics such as molecular structures, pharmacology, indications, etc. of each drug. For this reason, knowledge graph (KG)-based computational prediction approaches that do not rely on these expensive, labor-intensive features have received ever-increasing attention due to their capability of enabling automatic, fast assessments of possible DDIs.

DDI data can be represented as a KG in which nodes indicate entities and edges denote relations. A typical DDI KG is constructed with a series of triplet facts (h, r, t) in which h and t represent head and tail drugs, respectively, and r indicates the interaction between h and t. Accordingly, the DDI prediction problem can be posed as a link prediction task via KG embedding, which aims to embed each entity and relation to a low-dimensional feature space for knowledge fusion and more efficient computing. Figure 1 shows an example of a DDI KG.

Over the past years, several machine learning and deep learning approaches have been proposed to embed DDI KGs for predicting unknown DDIs [6–8]. Firstly, training a KG embedding model requires negative samples and there are no confirmed negatives in the original DDI datasets. In order to conveniently generate enough negative samples to train the model, Bordes et al. [9] introduced a local closed-world assumption (LCWA) into the KG. Under the LCWA, all statements that exist in the KG are assumed correct. Conversely, any statements that do not exist are false. This assumption is conducive to the generation of negative samples, as we only need to construct triplets that are not contained in the original KG to be considered as fake samples. Most of the existing embedding models have been generating negative triplets via a uniform negative sampling strategy [9–11]. This sampling randomly replaces the head or tail entity in a positive triplet with a different one from the entity collection, where all entities share the same sampling weights. Trouillon et al. [12] evaluated the performance impact of the number of negative triplets constructed per positive training sample. The results revealed that generating more negatives can, up to a saturation threshold, yield better performance. However, this general sampling method usually adds only limited benefit to the robustness and effectiveness of the derived embedding model and may even delay model convergence as noted by Schroff et al. [13] and Hermans et al. [14]. Thus, we utilize adversarial learning to generate more plausible negative triplets for improving the performance of KG representation learning. For instance, if we want to replace the head drug in an observed triplet (Tramadol, increase neuroexcitatory activities, Amitriptyline) to construct a negative triplet between the two candidates 'Ibuprofen' and 'Nexium', 'Ibuprofen' makes for a more deceptive replacement due to its pharmacological similarity to 'Tamadol'. Afterwards, this more plausible triplet can force the KG embedding model to improve performance to distinguish its authenticity. Nevertheless, it is also likely to choose other irrelevant drugs that would make it easy for the embedding model to distinguish and does not encourage it to improve (e.g. 'Nexium') if the head drug is replaced by the above random sampling mode. Our proposed method represents all drugs in a unified feature space, and then selects a suitable drug as a substitute to generate a deceptive negative sample according to the spatial position and distance between the drugs, thereby further improving the performance of the model.

Unfortunately, adversarial learning methods such as generative adversarial networks (GANs) have not yielded satisfying results for natural language processing tasks as the standard GAN is limited to the continuous real number space, i.e. continuous data, but cannot directly operate on discrete data such as words. To overcome this deficiency, recent research provides a number of feasible approaches by applying policy gradients, a class of policy-based reinforcement learning (RL) algorithms, to replace the traditional back propagation [15, 16].

Although these RL approaches have been proven effective, high-variance gradient estimates make models require vast amounts of computational resources while their complex hyperparameters increase instability of the already difficult-to-train GANs. In this work, we propose a new method that introduces Gumbel-Softmax relaxation [17, 18] and adversarial autoencoders (AAEs) based on Wasserstein distances for training DDI embedding models steadily on discrete data. In contrast to complicated RL mechanisms, the Gumbel-Softmax relaxation

can efficiently simplify our model and allow for a fast iterative adversarial learning framework without intensive RL heuristics for accelerating the convergence of the entire model. Compared to GANs, AAEs can control the manner in which the generator constructs negative samples, making their outputs resemble real data more closely. Furthermore, we use the Wasserstein distance as an advanced metric to replace the original Jensen-Shannon (JS) divergence in the traditional adversarial learning framework.

To this end, we first construct an autoencoder where the latent code vector **z** (i.e. its hidden units) is trained to generate more plausible entities (drugs) as negative samples. Since the entity we intend to generate is a one-hot vector and this type of discrete data is not differentiable in the training process, a Gumbel-Softmax relaxation and the Wasserstein distance are employed to handle the issue of vanishing gradients on discrete data without policy gradient mechanisms. Then, negative and positive triplets are jointly fed into the discriminator to obtain the embeddings, which are regarded as the final representation of the KG. Our novel contributions in this paper are summarized as follows:

- We present a new approach to solve the prediction of DDIs and their side effects. Compared with clinical trials or traditional machine learning-based methods, our approach does not require numerous manual features to yield better performance.
- Technically, to the best of our knowledge, we are the first to introduce AAEs to KG representation learning. The latent vector of the autoencoder is capable of generating more reasonable negative samples, and the discriminator utilizes these negative and positive triplets to train the KG embedding model.
- Different from traditional adversarial learning for KG embedding that requires intensive RL heuristics, we apply Gumbel-Softmax relaxation and Wasserstein distance to resolve the problem of vanishing gradients on discrete data and accelerate the convergence of the KG embedding model.
- We evaluate the performance of the proposed model on link prediction and triple classification tasks. The experimental results show that our model outperforms existing KG embedding models.

The remainder of this article is organized as follows. Section 2 introduces related work on DDI detection and prediction and several representative KG embedding models. In Section 3, we illustrate the overall framework and training procedure of the proposed adversarial learning model in detail. Section 4 delineates benchmark datasets, parameter initialization settings and experimental details. Section 5 provides a side-by-side qualitative comparison and discussion between our results and those obtained via existing methods. Finally, concluding remarks are discussed in Section 6.

Related work

In this section, we introduce current research on DDI detection and prediction. Additionally, we give a brief overview of several prominent existing KG embedding methods.

DDI detection and prediction

DDI prediction is a key task in pharmacology. Many existing studies that obtain results on specific types of interactions depend on *in vivo* and *in vitro* experiments. Krishna et al. [19] constructed a crossover study to evaluate the effects of gastric pH values on the absorption of posaconazole. The results displayed that both dissolution and absorption of posaconazole would be decreased under increased gastric pH conditions (e.g. induced via co-administration with proton pump inhibitor drugs, such as esomeprazole or omeprazole). To reveal the clinical effect of omeprazole on the inhibition of platelet clopidogrel, Ho et al. [20] conducted a retrospective cohort study of 8,205 patients, finding that the risk of adverse outcomes would be increased when clopidogrel is accompanied by proton pump inhibitors. Menon et al. [21] evaluated DDIs between the 3D regimen of direct-acting antiviral agents for the treatment of chronic hepatitis C virus infection (such as ombitasvir, Paritaprevir and dasabuvir) and various common medications via 13 experiments. Although the above works yield detailed comparative results, they do not scale well due to laboratory requirements. With the advancement of computing methods and resources, researchers moved their attention towards large-scale structured databases and machine learning based approaches to solve this problem.

Several studies have proposed automatic DDI discovery schemes. For instance, Cheng and Zhao [1] introduced phenotypic, therapeutic, genomic and chemical structural similarity as drug features and employed five machine learning approaches, including k-nearest neighbors, naïve Bayes, logistic regression, decision trees and support vector machines, to predict the authenticity of DDIs. Li et al. [22] constructed a Bayesian network for forecasting the combined effect of drugs by incorporating drug molecular and pharmacological phenotypes. Lately, to further enhance the performance of the DDI prediction model, many semantic and topological measures between drugs are utilized as input features for discovering potential DDIs [23]. Muñoz et al. [24] utilized KGs as a convenient uniform representation to integrate multiple forms of heterogeneous data, so that the data can be represented by a unified feature description. However, these feature-based approaches not only rely heavily on the quality of hand-crafted features, but also suffer from issues of data incompleteness and sparsity.

KG embedding has received increasing attention due to its strong capability of overcoming data incompleteness and sparsity problems. KG embedding methods have been demonstrated to offer competitive performance in DDI prediction tasks. Among them, Abdelaziz et al. [25] proposed a large-scale framework for DDI prediction, called Tiresias. It first integrated various drug-related variables as a DDI KG, then leveraged this KG to compute several similarity measures between all the drugs and predicted potential DDIs via a logistic regression classifier. Celebi et al. [26] applied several classic KG embedding algorithms such as TransE and TransD to extract feature vectors in order to predict potential interactions between drugs. Ma et al. [27] used multi-view graph autoencoders to integrate multiple types of drug-related information, and added an attention mechanism to calculate the corresponding weights of each view for better interpretability. Zitnik et al.[8] developed a graph convolutional neural network in which an end-to-end model was built for multi-relational link prediction on a multi-modal graph. Karim et al. [28] combined ComplEx (a traditional KG embedding method) with a convolutional-LSTM network to further refine model performance.

In the following, we will discuss a representative range of KG embedding techniques in greater detail.

Existing KG embedding approaches

There has been an increasing amount of literature on KG embedding to represent both entities and relations in a low-dimensional continuous feature space [29, 30]. We have broadly

categorised these existing embedding methods into three categories: *translation-based methods*, *tensor factorization-based methods* and *neural network-based methods*.

Translation-based embedding methods

Mikolov et al. [31] proposed translation invariance in the word embedding algorithm *word2vec* that allows words with similar connotation to have similar representations. Following this principle, Bordes et al. [9] proposed the **TransE** KG embedding model. **TransE** interprets relations as translation vectors between head and tail entities on the low-dimensional embedding vector space, namely $\boldsymbol{h} + \boldsymbol{r} \approx \mathbf{t}$. A score function is defined to measure the plausibility of each triplet fact (h, r, t). The score indicates the distance between $\boldsymbol{h} + \boldsymbol{r}$ and $\mathbf{t}$, and the function is formulated as follows:

$$f_r(h,t) = \left\| \boldsymbol{h} + \boldsymbol{r} - \mathbf{t} \right\|_{\ell_1/\ell_2}, \tag{1}$$

where ℓ_1, ℓ_2 are L_1-norm and L_2-norm respectively. It is worth noting that the embedding model yields a low score if a triplet (h, r, t) is valid and a high score otherwise.

Although **TransE** generally delivers solid performance, it struggles to solve complex relations, such as $1-N$, $N-1$, and $N-N$. **TransH** [32] was proposed to overcome this drawback by introducing relation-specific hyperplanes. **TransR** [33] expanded relation-specific hyperplanes to relation-specific spaces. Since then, a large number of embedding models investigated different ways to improve performance. **TransA** [10] abandoned traditional Euclidean distances and adopted adaptive Mahalanobis distances as a better metric on account of their flexibility and adaptability. **TransG** [34] proposed to modify the model by introducing multidimensional Gaussian distributions to replace the original conclusive numerical space and constructed a probabilistic embedding model to represent entities and relations.

Tensor factorization-based embedding methods

Tensor factorization is another effective approach to KG embedding. **RESCAL** [35] is the representative approach in this direction. Under **RESCAL**, all triplet facts in the KG are projected into a 3D binary tensor $\mathcal{X}$ to express the inherent structure, $\mathcal{X}_{ijk} = 1$ indicates that the observed triplet (i-th entity, k-th relation, j-th entity) exists in the graph; otherwise, $\mathcal{X}_{ijk} = 0$ refers to an unknown or non-existent triplet. Afterwards, rank-d factorization is applied to obtain latent semantics in the KG. The principle that this model follows is formulated as:

$$\mathcal{X}_k \approx A R_k A^T, \text{ for } k = 1, 2, \cdots, m, \tag{2}$$

where $A \in \mathbb{R}^{n \times d}$ is a matrix which has the capability of capturing the latent semantic structure of entities and $R_k \in \mathbb{R}^{d \times d}$ is a matrix that models the pairwise interactions in the k-th relation. According to this principle, the score function $f_r(h, t)$ is defined as:

$$f_r(h,t) = \boldsymbol{h}^\top \mathbf{M}_r \mathbf{t}, \tag{3}$$

where $\boldsymbol{h}, \mathbf{t} \in \mathbb{R}^d$ represent embedding vectors of entities like in the above models, the matrix $\mathbf{M}_r \in \mathbb{R}^{d \times d}$ denotes the latent semantic meanings in relation r. To simplify the computational complexity of **RESCAL**, **DistMult** [36] restricted $\mathbf{M}_r$ to diagonal matrices, i.e. $\mathbf{M}_r = \text{diag}(\boldsymbol{r}), \boldsymbol{r} \in \mathbb{R}^d$. The score function is transformed as follows:

$$f_r(h,t) = \boldsymbol{h}^\top \text{diag}(\boldsymbol{r}) \mathbf{t}. \tag{4}$$

The original **DistMult** model is symmetric in head and tail entities for every relation; **ComplEx** [12] leveraged complex-valued embeddings to extend **DistMult** to asymmetric relations. The embeddings of entities and relations exist in the complex space $\mathcal{C}^d$, instead of the real space $\mathbb{R}^d$ in which **DistMult** embedded. The score function is modified to:

$$f_r(h,t) = \text{Re}\left(\boldsymbol{h}^\top \text{diag}(\boldsymbol{r}) \bar{\mathbf{t}} \right), \tag{5}$$

where $\text{Re}(\cdot)$ denotes the real part of a complex value, and $\bar{\mathbf{t}}$ represents the complex conjugate of $\mathbf{t}$. By using this score function, triplets that have asymmetric relations can obtain different scores depending on the sequence of entities.

SimplE [11] proposed the inverse embedding of relations and leveraged it to calculate the average Canonical Polyadic score of (h, r, t) and (t, r^{-1}, h). The score function is formulated as follows:

$$f_r(h,t) = \frac{1}{2} \left(\boldsymbol{h} \circ \boldsymbol{r}\mathbf{t} + \mathbf{t} \circ \boldsymbol{r}'\boldsymbol{h} \right), \tag{6}$$

where $\boldsymbol{r}'$ denotes the embedding of inversion relation and $\circ$ indicates the element-wise Hadamard product. **RotatE** [37] proposed a rotational model in which each relation is regarded as a rotation from source entity to target entity in complex space, as $\mathbf{t} = \boldsymbol{h} \circ \boldsymbol{r}$. In addition, **RotatE** also provided a self-adversarial negative sampling mode, which selects negative triplets according to the scores calculated by the current embedding model.

Neural network-based embedding methods

Deep neural networks have become popular in a multitude of fields due to their strong generalization and representation abilities. They have been widely used for KG embedding.

ConvKB [38] was proposed to capture semantic information contained in entities and relations by incorporating convolutional neural networks (CNN). In **ConvKB**, the embedding vectors $\boldsymbol{h}, \boldsymbol{r}$ and $\mathbf{t}$ are concatenated to a matrix as an input layer, and after a convolution operation, the final output is obtained.

Inspired by adversarial learning, Minervini et al. [39] proposed an adversarial set regularization method for regularizing traditional embedding models, where an adversary samples the most plausible set of input representations. De Cao and Kipf [40] adopted GANs to generate molecules with specific desired chemical properties. Moreover, Wang et al. [15] and Cai et al. [16] applied GANs to sample plausible negative training examples for KG embedding via policy gradients. They employed the generator $G(z; \theta)$ to construct negative triplets and utilized the discriminator $D(x; \phi)$ as an embedding model to distinguish artificial from real triplets.

In summary, most previous methods used random sampling or GANs to generate negative training triplets. While GANs improved model performance, they also drastically increased computational complexity and brought instability to the training process. In this paper, we describe a new framework based on AAEs for improving the representation ability of models by generating high quality plausible negative samples to train the discriminator. Compared with the above methods, our framework

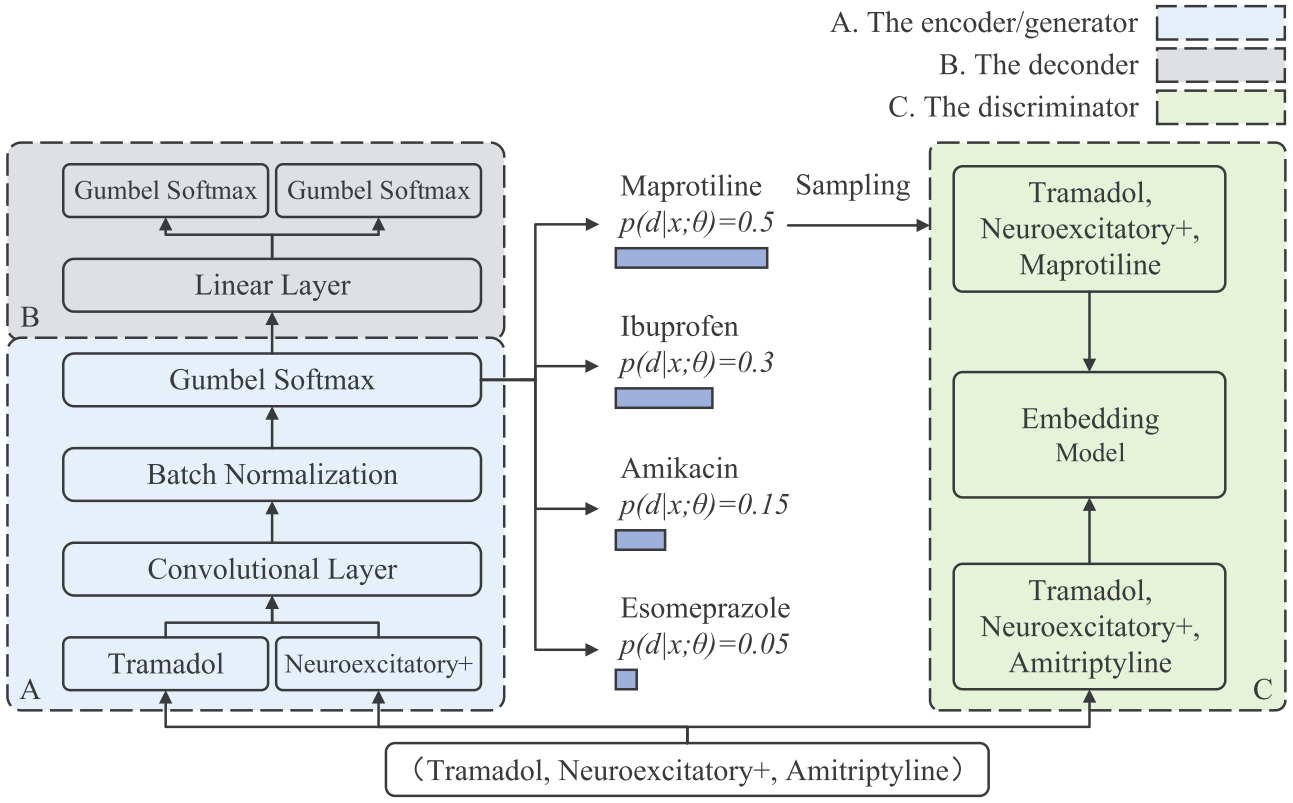

Fig. 2. The architecture of the proposed AAE for KG embedding. (a) The encoder of the autoencoder learns to generate plausible negative triplets for the discriminator. (b) The decoder of the autoencoder is applied to further refine the performance of the encoder by minimizing reconstruction errors. (c) The generated negative triplet and the original positive triplet are both fed into the discriminator, as illustrated in the right part. (d) The discriminator is trained to yield a robust and effective KG representation model. Neuroexcitatory+ indicates the interaction 'increase neuroexcitatory activities'.

can generate more reasonable negative samples in less time, thereby improving the performance and practical usefulness of the embedding model.

Method

A KG is a directed graph in which the nodes correspond to entities and the edges represent various types of relations between pairs of entities. Given a KG composed of a collection of triplet facts $\Omega = \{(h, r, t)\}$, and a pre-defined embedding dimension d, KG embedding aims to represent each entity $h \in E$ and relation $r \in R$ in a d-dimensional continuous vector space, where E and R are the sets of entities and relations, respectively. To simplify the problem, we transform entities and relations into uniformly sized embedding spaces, i.e. $d = k$. In other words, the embedding process projects textual triplets (h, r, t) into a dense numerical vector space, where each entity or relation is transformed to a d-dimensional vector. With this vector representation, we can facilitate link prediction, DDI classification and other downstream applications. In DDI KGs, drugs are entities and interactions between drugs are represented as relations. It is worth to note that all DDI KGs we introduced and utilized only contain the names of various drugs and interactions, and their direct relationships. Apart from that, there are no drug properties or other additional information available.

Figure 2 illustrates the proposed adversarial learning framework. At the beginning, a head or tail drug is discarded randomly from an authentic DDI, and the resulting fragmentary triplet (Tramadol, increase neuroexcitatory activities, ?) is picked up as the input of the encoder. The encoder receives it and generates a one-hot vector which indicates another drug that has a similar effect or structure to Amitriptyline (such as Maprotiline which obtains the highest probability) from the collection of candidate drugs, this one-hot vector needs to be fed to both the decoder and the discriminator. For the decoder direction, the final outputs are two new vectors corresponding to the inputs of the encoder. The decoder restricts them to be as close to the inputs of the encoder as possible. As a consequence, we can not only guarantee that the model can generate different drugs, but also ensure that the generated drugs are proximal to the original ones in feature space. For the discriminator direction, the drug 'Maprotiline' is selected to construct the final negative triplet (Tramadol, increase neuroexcitatory activities, Maprotiline). Finally, the negative and positive triplets are jointly fed into the KG embedding model for learning embedding vectors.

Autoencoder for sampling negative triplet facts

The goal of the autoencoder is to provide more plausible negative triplets for the discriminator than what can be obtained via traditional random negative sampling.

Shortcomings of traditional negative sampling

Since Bordes et al. [9] proposed to obtain corrupted triplets via uniform negative sampling, many researchers have applied this strategy to sample negative triples in the training process. This

sampling strategy randomly selects a candidate entity from the entity set E to replace the head or tail entity from the original positive triplet. It is worth to note that all candidate entities in entity set E share the same probability of being drawn.

Obviously, this sampling method cannot contribute much to training an effective embedding model in most cases. As an example, given a valid triplet (*Tramadol*, *increase neuroexcitatory activities*, *Amitriptyline*), our goal is to replace the tail drug with another acceptable drug to reassociate a plausible triplet. Given the word '*neuroexcitatory*' in the relation and the drug type of '*Amitriptyline*', it is intuitive to the domain expert that the tail drug should be a kind of pain reliever. If we choose the candidate drug in a random manner, many constructed negative triplets such as (*Tramadol, increase neuroexcitatory activities, Esomeprazole*) or (*Tramadol, increase the neuroexcitatory activities, Minoxidil*) can be trivially picked up as false by the discriminator, resulting in only infrequent parameter updates. By comparison, another generated triplet such as (*Tramadol, increase neuroexcitatory activities, Acetaminophen*) seems to be a more reasonable DDI, because '*Acetaminophen*' is more pharmacologically similar to '*Tramadol*' than '*Minoxidil*'. It is worth noting that, although this traditional sampling method does not perform as well as the proposed method in most cases, when a KG is particularly sparse (in other words, there are extremely few triplets corresponding to each entity in the KG) the random sampling method has the opportunity to select any entity as a negative sampling option to train the model, so as to obtain a well-trained embedding model. This article does not consider such extreme cases.

As a consequence, we introduce an autoencoder to construct more plausible negative triplets instead of traditional uniform sampling. Here, the encoder aims to generate drugs as the generator in an adversarial learning framework, while the decoder restricts the manner and type of the generated drug, forcing it closer to the input drug and interaction. However, there is still a 'non-differentiability' problem in discrete data generation.

Gumbel-Softmax relaxation with discrete data

In this section, we first illustrate why training adversarial learning models with discrete data is a vital issue. From a mathematical perspective, assuming the total number of drugs (entities) is $|E|$, the next generated one-hot index vector $\mathrm{y} \in \mathbb{R}^{|E|}$ can be obtained by sampling:

$$\mathrm{y} \sim \sigma(\mathrm{o}), \tag{7}$$

where $\mathrm{o} \in \mathbb{R}^{|E|}$ denotes the output logits of the last layer in the generator, $\sigma(\cdot)$ indicates the Softmax function. The sampling operation in Equation (7) implies a step function that is not differentiable at the end of the generator output. Because the differential coefficient of a step function is 0 almost everywhere, we have $\frac{\partial \mathrm{y}}{\partial \theta_G} = 0$, *a.e.*, where θ_G are the parameters of the generator. According to the chain rule, the gradients of the generator loss l_G with respect to θ_G are calculated as:

$$\frac{\partial \mathrm{l}_G}{\partial \theta_G} = \frac{\partial \mathrm{y}}{\partial \theta_G}\frac{\partial \mathrm{l}_G}{\partial \mathrm{y}} = 0 \quad a.e. \tag{8}$$

As a result, $\partial \mathrm{l}_G / \partial \theta_G = 0$ means that the gradients of the generator loss cannot be propagated back to the generator via the discriminator. In other words, the generator cannot update its own parameters based on the feedback provided by the discriminator. This phenomenon is called the 'vanishing gradient' or 'non-differentiability' issue of adversarial learning models in discrete data domains.

From an instance point of view, even though a Softmax output vector of the generator $\alpha = [0.25, 0.35, 0.25, 0.15]$ can improve the performance of the generator to optimize α to $\beta = [0.05, 0.70, 0.15, 0.10]$ allowing localizing a specific entity, the final sampling result has not changed, i.e. $Onehot(\alpha) = Onehot(\beta) = [0, 1, 0, 0]$. The identical sampling one-hot vectors are repeatedly fed to the discriminator, so that the gradients obtained by the discriminator are inoperative, and the convergence direction of the generator is indistinct, no matter how powerful the discriminator may be.

In order to solve the 'non-differentiablity' issue, this paper leverages a Gumbel-Softmax relaxation technique which can approximate patterns sampled from a categorical distribution by defining a continuous distribution on the simplex. There are two parts in the Gumbel-Softmax relaxation: (1) The Gumbel-Max trick. Following previous studies proposed by Jang et al. [17] and Maddison et al. [18], the sampling in Equation (7) should be reparametrized as follows:

$$\mathrm{y} = \mathrm{one\_hot}\left(\arg\max_{1 \le \mathrm{i} \le |E|} \left(\mathrm{o_i} + g_\mathrm{i}\right)\right), \tag{9}$$

where $\mathrm{o_i}$ is the i-th element of o and $g_1, \cdots, g_{|E|}$ are i.i.d. samples drawn from a standard Gumbel distribution, i.e. $g_\mathrm{i} = \log(-\log U_\mathrm{i})$ with $U_\mathrm{i} \sim Uniform(0, 1)$. (2) Relaxing the discreteness. So far the 'arg max' operation in Equation (9) is still non-differentiable. We employ the Softmax function as a differentiable, continuous approximation to further approximate it, and calculate a $|E|$-dimensional sample vector $\hat{\mathrm{y}}$. Each entry in $\hat{\mathrm{y}}$ is acquired by:

$$\hat{\mathrm{y}}_\mathrm{i} = \frac{\exp\left(\left(\left(\mathrm{o_i}\right) + g_\mathrm{i}\right)/\tau\right)}{\sum_{a=1}^{|E|} \exp\left(\left(\left(o_a\right) + g_a\right)/\tau\right)}, \tag{10}$$

where $\tau > 0$ is an adjustable parameter referred to as the *inverse temperature*. When the temperature τ approaches 0, the sampled vectors from the Gumbel-Softmax distribution are equal to one-hot vectors and the Gumbel-Softmax distribution becomes identical to the categorical distribution. It is worth noting that, in this way, $\hat{\mathrm{y}}$ can be differentiated with respect to o, we can utilize $\hat{\mathrm{y}}$ to replace one-hot vector y as the final output of the generator. Consequently, the 'non-differentiability' issue is solved by taking advantage of the Gumbel-Softmax relaxation. The generator (the encoder part of our autoencoder) can smoothly generate one-hot vectors that indicate plausible drugs.

Autoencoder architecture

In the generator, each drug and interaction are initially transformed from a one-hot index vector to a specific embedding feature space associated with two embedding matrices, one for drugs, indicated by $E^{|E| \times d}$, and one for interactions, indicated by $R^{|R| \times k}$, $|E|$ and $|R|$ are the total numbers of drugs and interactions, respectively. In this paper, the embedding dimensionality of drugs is identical to that of interactions, i.e. $d = k$. Because of this setting, we can concatenate the embedded vectors of head drug $\boldsymbol{h}$ and interaction $\boldsymbol{r}$, and reshape it as an input $\boldsymbol{A} = Reshape([\boldsymbol{h}, \boldsymbol{r}])$ to the 2D convolutional network layer which has been shown to extract available features with filters ω. A feature

map tensor $\mathcal{T} \in \mathbb{R}^{b\times m\times n}$ is calculated through this layer, where b is the number of feature maps with dimensions m and n. After that, we reshape the tensor $\mathcal{T}$ into a single vector $\mathbf{t} \in \mathbb{R}^{bmn}$, and then transform it into an $|E|$-dimensional feature vector by using the projection matrix $\mathbf{W} = \mathbb{R}^{bmn\times|E|}$. Finally, the Gumble-Softmax relaxation described above is applied to generate a plausible tail drug. Mathematically, the one-hot vector of drug $\boldsymbol{v}$ is calculated as:

$$\boldsymbol{v} = g\left(Re(Re([\boldsymbol{h};\boldsymbol{r}]*\omega)\mathbf{W})\right), \tag{11}$$

where $Re(\cdot)$ represents the reshape operation, and $g(\cdot)$ is the Gumble-Softmax relaxation. The output of the generator is a one-hot index vector that refers to a specific drug. This drug, when associated with the inputs of the generator including head drug and interaction forms the corrupted triplet.

The one-hot vector $\boldsymbol{v}$ acts as the input, given by two linear network layers. In order to invoke the restriction of the autoencoder forcing the neural network to capture only significant features of the data, there are two outputs in the decoder. These two output dimensions are $|E|$ and $|R|$, corresponding to the dimensions of the two generator inputs.

KG embedding discriminator

The discriminator used in our framework is constructed following previous research. As described in Section 2, the individual models have different structures and score functions. The embeddings of drugs and interactions are obtained by minimizing the ranking loss associated with positive and negative triplets. Different from previous models in which negative samples are generated via random sampling from whole drug set, we apply an autoencoder to construct more plausible triplets to refine the performance of the model.

Training strategy

The training procedure is comprised of three main parts: **i**) the parameter update of the autoencoder in which G and A indicate the generator (the encoder) and the decoder, respectively; **ii**) the parameter update of the discriminator D; **iii**) the parameter update of the generator G.

The autoencoder is designed to learn an effective representation of data. In this paper, we employ the encoder network $G(\mathbf{z};\theta)$ to generate high-quality negative drugs and apply the decoder network $A(\mathbf{x};\eta)$ to restrict the sampling direction to obtain more plausible samples. To update parameters θ and η, we train the autoencoder by minimizing the reconstruction error $L_{G,A}$:

$$\min L_{G,A} = \min_{(\mu=\theta,\eta)} \|x - A\left(G\left(z;\theta\right);\eta\right)\|^2. \tag{12}$$

The goal of the discriminator network $D(\mathbf{x};\phi)$ is to distinguish a sample $\boldsymbol{x}$ as originating either from the real distribution $p_r(\boldsymbol{x})$ or the generator $p_\theta(\boldsymbol{x})$. Given an original training sample $(\boldsymbol{x}, y)$, $y \in \{1, -1\}$ signals whether it is a true sample from $p_r(\boldsymbol{x})$ or a generated sample from $p_\theta(\boldsymbol{x})$, the optimization objective of the discriminator L_D is to minimize cross-entropy:

$$\min L_D = \min_{\phi} -\left(y\log p(y=1|\boldsymbol{x}) + (1-y)\log p(y=0|\boldsymbol{x})\right). \tag{13}$$

If the distribution $p(\boldsymbol{x})$ is a mixture of distributions $p_r(\boldsymbol{x})$ and $p_\theta(\boldsymbol{x})$ in equal proportions, i.e. $p(\boldsymbol{x}) = \frac{1}{2}(p_r(\boldsymbol{x}) + p_\theta(\boldsymbol{x}))$, then Equation (13) can be rewritten as:

$$\min L_D = \min_{\phi} -\left(\log D(\mathbf{x};\phi) + \log(1 - D(G(\mathbf{z}^{(i)};\theta);\phi))\right). \tag{14}$$

The goal of the generator is the opposite of the discriminator, the generator tries to construct negative samples which can fool the discriminator into confusing a negative sample for a real one. Its objective function L_G is formulated as:

$$\min L_G = \min_{\theta} \left(\log(1 - D(G(\mathbf{z};\theta),\phi))\right). \tag{15}$$

Compared with a traditional single-objective optimization task, the optimization goals of these two networks in the adversarial game are extremely challenging. There are many potential issues in the traditional adversarial network training process such as training instability and difficulty, uninformative loss functions of generator and discriminator, and a lack of diversity in the generated samples.

These problems are caused by the attempt to minimize the JS divergence between the real distribution $p_r(\boldsymbol{x})$ and the generated distribution $p_\theta(\boldsymbol{x})$. The JS divergence can only be computed when two distributions P, Q have overlapping parts. When these two distributions do not overlap or the overlapping parts are negligible in size, their JS divergence is identically equal to $\log 2$. That means that when the real distribution $p_r(\boldsymbol{x})$ and the generated distribution $p_\theta(\boldsymbol{x})$ have no overlap, the outputs of the discriminator are 0 for all generated data, i.e. $D(G(\mathbf{z},\theta)) = 0, \forall \mathbf{z}$. As a result, the gradients of the generator vanish.

Inspired by Wasserstein GANs [41] in which Wasserstein distance (also known as Earth-Mover distance) is introduced as a more robust metric to replace the JS divergence, we use this distance measure to improve the performance of our KG embedding framework in this article. Given a real distribution $p_r(\boldsymbol{x})$ and a generated distribution $p_\theta(\boldsymbol{x})$, the 1st-Wasserstein distance between them is formalized as:

$$\mathrm{W}\left(p_r, p_\theta\right) = \inf_{\gamma\sim\Pi(P_r,P_\theta)} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\gamma}[\|\mathbf{x}-\mathbf{y}\|], \tag{16}$$

where $\Pi\left(P_r, P_\theta\right)$ is the set of all possible joint distributions with marginal distribution $\gamma(\mathbf{x},\mathbf{y})$. When there are no overlapping or slightly overlapping parts between two distributions, the JS divergence becomes constant. In contrast, the 1st-Wasserstein distance can measure distances between two non-overlapping distributions.

Equation (16) is difficult to calculate directly, and needs to be transformed into a solvable form via the Kantorovich-Rubinstein duality theorem [42]. According to this theorem, the Wasserstein distance between two distributions can be converted into an upper bound on the expected difference between distributions p_r and p_θ for a function that satisfies the K-Lipschitz continuum. We rewrite the 1st Wasserstein distance:

$$\mathbf{W}\left(p_r, p_\theta\right) = \frac{1}{K}\sup_{\|f\|_L\le K}\left(\mathbb{E}_{x\sim p_r}[f(\mathbf{x})] - \mathbb{E}_{x\sim p_\theta}[f(\mathbf{x})]\right), \tag{17}$$

where $f(\cdot)$ is the K-Lipschitz function, that satisfies the following:

$$\|f\|_L \triangleq \sup_{x\neq y}\frac{|f(\mathbf{x}) - f(\mathbf{y})|}{|\mathbf{x}-\mathbf{y}|} \le K. \tag{18}$$

If a function is differentiable and its derivatives are bounded, then this function is a Lipschitz continuous function. Because the discriminator neural network $D(x;\phi)$ satisfies the above conditions, it is also a Lipschitz continuous function, allowing us to

Table 1. Statistics of the data sources

Datasets	#Drugs	#DDI Types	#Train	# Valid	#Test
DeepDDI	1,710	86	153,828	19,228	19,228
Decagon	637	200	897,446	112,181	112,181

approximate the upper bound in Equation (17) as:

$$\min_{\phi} -\left(\mathbb{E}_{x\sim p_r}[D(\mathbf{x};\phi)] - \mathbb{E}_{x\sim p_\theta}[D(\mathbf{x};\phi)]\right). \tag{19}$$

Different from standard discriminator networks in which the final layer is a sigmoid function over an output range of $[0, 1]$, at this point, we only need to find a network $D(\mathbf{x};\phi)$ that maximizes the difference in expectations between the two distributions p_r and p_θ. As a consequence, the final layer in our discriminator $D(\mathbf{x};\phi)$ is a linear layer, and its range is not limited. That means that, for real samples, the score of $D(\mathbf{x};\phi)$ should be high, and for samples generated by the model, low scores are expected.

Moreover, to make $D(\mathbf{x};\phi)$ satisfy the K-Lipschitz continuity condition, we can approximate it by limiting the range of the parameter ϕ, such that $\phi \in [-c, c]$, c is a relatively small positive number.

The goal of the generator is to minimize the Wasserstein distance, making the real distribution p_r and the generated distribution p_θ coincide as much as possible, i.e.

$$\min_{\theta} -\mathbb{E}_{z\sim p(z)}[D(G(z;\theta);\phi)]. \tag{20}$$

Because $D(\mathbf{x};\phi)$ is an unsaturated function, the gradients of the generator parameters θ will not disappear. This solves the problem of instability in original adversarial framework. In addition, by replacing the JS divergence by the Wasserstein distance, the objective function of the generator in this framework can alleviate the model collapse problem to a certain extent and make the generated samples more diverse. The detailed procedure of this adversarial framework for KG embedding is described in Algorithm 1.

Algorithm 1: Adversarial training of the autoencoder with Wasserstein distance for learning interactions in knowledge graph embeddings.

Input : The set of positive DDIs $\Omega = \{(h, r, t)\}$, the number of training iterations e, the number of discriminator iterations per generator iteration n_{dis}, mini-batch size m, the learning rate of the generator α, the learning rate of the discriminator β, the clipping parameter c.

Output: Drugs and interactions embeddings learned by D.

Initialize the generator G with parameters θ_0, the decoder A with parameters η_0, the discriminator D with parameters ϕ_0;
for $k = 1, \cdots, e$ **do**
  for $i = 1, \cdots, m$ **do**
    $L_{G,A}^{(i)} = \left\|x^{(i)} - A\left(G\left(z^{(i)};\theta\right);\eta\right)\right\|^2$; // Update the autoencoder by minimizing $L_{G,A}^{(i)}$
  end
  $g_\mu \leftarrow \nabla_\mu \frac{1}{m}\sum_{i=1}^{m} L_{G,A}^{(i)}, \quad (\mu = \theta, \eta)$;
  $\theta \leftarrow \theta - \alpha \cdot \text{Adagrad}\,(\theta, g_\mu)$;
  $\eta \leftarrow \eta - \alpha \cdot \text{Adagrad}\,(\eta, g_\mu)$;
  for $t = 1, \cdots, n_{dis}$ **do**
    for $i = 1, \cdots, m$ **do**
      $L_D^{(i)} = -\left[D\left(x^{(i)};\phi\right) - D\left(G\left(z^{(i)};\theta\right);\phi\right)\right]$; // Update the discriminator by minimizing $L_D^{(i)}$
    end
    $f_\phi \leftarrow \nabla_\phi \frac{1}{m}\sum_{i=1}^{m} L_D^{(i)}$;
    $\phi \leftarrow \phi - \beta \cdot \text{Adagrad}\left(\phi, f_\phi\right)$;
    $\phi \leftarrow \text{clip}(\phi, -c, c)$;
  end
  for $i = 1, \cdots, m$ **do**
    $L_G^{(i)} = -D\left(G\left(z^{(i)};\theta\right);\phi\right)$; // Update the generator by minimizing $L_G^{(i)}$
  end
  $h_\theta \leftarrow \nabla_\theta \frac{1}{m}\sum_{i=1}^{m} L_G^{(i)}$;
  $\theta \leftarrow \theta - \alpha \cdot \text{Adagrad}\,(\theta, h_\theta)$;
end
Return Drug and interaction embeddings.

Experiments

In this section, we first describe the experimental datasets, then introduce important hyper-parameter settings and comparison methods for our experiment. Afterwards, link prediction and DDI classification experiments are constructed for comparing performance of the proposed methods with benchmark and the state-of-the-art models. Finally, we project the high-dimensional embedding feature space to two-dimensions for visual inspection of qualitative example outputs.

Datasets

We conduct our link prediction and DDI classification experiments on two widely used public datasets: **DeepDDI** and **Decagon**. For both datasets, we randomly sample 80% of drug-drug pairs as training data, 10% as validation data and the remaining 10% as test data. Statistics of these two datasets are collected in Table 1.

DeepDDI [7] is composed of 1,710 drugs and 86 different interaction types from DrugBank [43] capturing 192,284 drug-drug pairs as samples. 99.87% of drug-drug pairs only have one type of DDI.

Decagon [8] is composed of 637 drugs and 200 different interaction types from the TWOSIDES dataset [44] capturing 1,121,808 drug-drug pairs as samples. We follow common practice by sampling 200 medium frequency DDI types ranging from Top-600 to Top-800, ensuring that every DDI type has at least 90 drug combinations. 73.27% of drug-drug pairs have more than one type of DDI.

Comparison methods

To comprehensively evaluate the performance of our proposed model, we select several representative methods from the three categories of KG embedding as baselines to be compared with our approach. These baselines are described as follows:

- **TransE** [9] represents both entities and relations in a low-dimensional feature space, and interprets relations as translation operations to concatenate the entities.
- **DistMult** [36] proposes a multi-relational learning method in which the bilinear objective is effective at capturing relational semantics.
- **ComplEx** [12] describes a simple tensor factorization method using embedding vectors with complex values to handle symmetric and asymmetric relations.
- **KBGAN** [16] introduces an adversarial learning framework for KG embedding in which a generator is applied to sample negative triplets for refining the performance of the discriminator.
- **SimplE** [11] develops an embedding method based on Canonical Polyadic decomposition that extends the model to learn the two embedding vectors of each entity independently.
- **RotatE** [37] embeds entities as complex-value vectors and defines relations as rotations from the head entity to the tail in a complex vector space. In addition, it utilizes a new self-adversarial negative sampling method to train the embedding model.

Link prediction

Link prediction is a characteristic task which aims to infer the missing drug when given an existing drug and interaction query. Concretely, the target of link prediction is to predict the missing drug t if given (h, r) or predict h given (r, t). Results are obtained via ranking by discriminator scores.

Metrics

For each DDI (h, r, t) in the test set, the real head drug (or tail drug) is circularly replaced by all drugs in the drug set E. Then, the scores corresponding to all triplets are computed, all scores are ranked in descending order.

However, some reconstructed DDIs might coincidentally be authentic in the DDI KG. In this case, the reconstructed DDI which is a true fact might yield a high ranking, resulting in an inaccurate assessment. To avoid this situation, following Bordes et al. [9], we employ the '**Filtered**' setting to eliminate all reconstructed DDIs which appear either in the training, validation, or test datasets. Finally, model performance is measured in terms of:

- **MR**: the average rank of the real entities.
- **MRR**: the average reciprocal rank of the real entities.
- **HITS@N%**: the proportion of real entities that ranked in the top N. Here, we specially choose $N = 1, 3, 10$ to validate the performance of compared models.

It should be noted that good performance is indicated by low **MR** and high **MRR** and **HITS@N%** scores.

Training protocol

We utilize the Adagrad self-adaptive optimizer for training, and perform parameter optimization via limited grid search: the learning rate of the generator $\alpha \in \{0.01, 0.005, 0.001\}$, the learning rate of the discriminator $\beta \in \{0.5, 0.1, 0.05, 0.01\}$, the size of drug and interaction embedding vectors $d \in \{50, 100, 200\}$, the mini-batch size $m \in \{256, 512, 1024\}$, the number of discriminator training iterations per generator iteration $n_{dis} \in \{1, 2, 5\}$ and the number of overall training iterations $e \in \{300, 500, 700, 1000\}$. The final parameter settings are determined on the validation set.

On the DeepDDI dataset, the best configurations are $\{\alpha = 0.001, \beta = 0.05, d = 200, m = 512, n_{dis} = 1, e = 300\}$ for our model with ComplEx, $\{\alpha = 0.001, \beta = 0.1, d = 200, m = 512, n_{dis} = 1, e = 300\}$ for our model with SimplE and $\{\alpha = 0.001, \beta = 0.5, d = 200, m = 512, n_{dis} = 2, e = 500\}$ for our model with RotatE. On the Decagon dataset, the best configurations are $\{\alpha = 0.005, \beta = 0.5, d = 200, m = 1024, n_{dis} = 1, e = 1000\}$ for our model with ComplEx, $\{\alpha = 0.005, \beta = 0.5, d = 200, m = 512, n_{dis} = 2, e = 1000\}$ for our model with SimplE and $\{\alpha = 0.005, \beta = 0.5, d = 200, m = 512, n_{dis} = 5, e = 1000\}$ for our model with RotatE. More details about parameter settings are illustrated in Appendix A.

Comparison with state-of-the-art models

We adopt the above configurations to train our model and compare our results to state-of-the-art methods. Table 2 shows a detailed comparison of the proposed approach and comparative methods on the two standard benchmark datasets. We can observe that:

- On both datasets, the KG embedding models trained via our proposed adversarial framework obtain a better performance on all metrics compared with other state-of-the-art methods. Especially ComplEx model we trained, it achieved the best performance on both datasets.
- As early models in KG embedding, TransE and DistMult have their inherent limitations in expressiveness compared with current methods. These issues are unlikely to be completely compensated by advanced training approaches. That is the reason against training them via adversarial learning in the experiment. And ComplEx and SimplE, which can be regarded as extension version of DistMult, achieve an improvement on these metrics by introducing complex-valued embeddings.
- Compared with these models with uniform negative sampling, KBGAN and RotatE also yield good performances on the two benchmark datasets. Especially, KBGAN achieves the 2nd best result in MR and RotatE achieves the 2nd best result in HITS@1% on the Decagon dataset. The pivotal factor to their good performance is that they also utilize adversarial learning to train the model. More detailed analyses are provided in Section 5.1.
- On the DeepDDI datasets, the proposed framework can improve the performance by an average of 3 points of HITS@10% beyond the original methods. Even under the increased complexity of the Decagon dataset that includes more interaction types and 73.27% drug-drug pairs having more than one type of interaction, we observe an average improvement of 1 percentage point.

DDI classification

DDI classification is an important pharmacological task that aims to determine the authenticity of a DDI triplet. As some existing articles [6, 7, 28] investigate DDI prediction, we follow them in casting DDI classification as a multi-label interaction prediction problem.

Given a pair of drugs, we first construct DDI triplets by repeatedly adding each interaction in the interaction set R into the pair of drugs, then estimate the confidence in every generated triplet. Those interactions corresponding to high-score triplets are the ones we want to obtain.

Table 2. Evaluation results on link prediction. The best two results are highlighted in rgb 1,0,0red and rgb 0,0,1blue and are formatted in boldface

Methods	DeepDDI					Decagon				
	MR	MRR	HITS@1%	HITS@3%	HITS@10%	MR	MRR	HITS@%1	HITS@3%	HITS@10%
TransE[9]	33.1338	0.2355	0.0069	0.3858	0.6216	44.7126	0.1005	0.0001	0.1106	0.2923
DistMult[36]	42.9008	0.1474	0.0532	0.2063	0.3765	57.9994	0.1212	0.0415	0.1149	0.2506
ComplEx[12]	15.8439	0.5196	0.3731	0.5725	0.7889	20.6787	0.2133	0.0514	0.2018	0.4249
KBGAN[16]	15.6894	0.5209	0.3824	0.5791	0.7957	rgb 0,0,1**20.2376**	0.2068	0.0492	0.1893	0.4081
SimplE[11]	16.0407	0.5037	0.3650	0.5738	0.7895	21.3255	0.2034	0.0484	0.1906	0.4188
RotatE[37]	15.5169	0.4925	0.3432	0.5624	0.7683	43.0815	0.1705	rgb 0,0,1**0.0928**	0.1686	0.3203
Our model (ComplEx)	rgb 1,0,0**11.9863**	rgb 1,0,0**0.5455**	rgb 1,0,0**0.4076**	rgb 1,0,0**0.6224**	rgb 0,0,1**0.8046**	rgb 1,0,0**19.9381**	rgb 1,0,0**0.2280**	0.0613	rgb 1,0,0**0.2201**	rgb 1,0,0**0.4372**
Our model (SimplE)	14.5913	rgb 0,0,1**0.5308**	rgb 0,0,1**0.3979**	0.6019	0.7996	20.6682	rgb 0,0,1**0.2176**	0.0539	rgb 0,0,1**0.2047**	rgb 0,0,1**0.4287**
Our model (RotatE)	rgb 0,0,1**13.8462**	0.5284	0.3904	rgb 0,0,1**0.6044**	rgb 1,0,0**0.8071**	41.3432	0.1772	rgb 1,0,0**0.0974**	0.1763	0.3314

Table 3. Evaluation results on DDI classification. The best two results are highlighted in rgb 1,0,0red and rgb 0,0,1blue and are formatted in boldface.

Methods	DeepDDI					Decagon				
	PR-AUC	ROC-AUC	P@1	P@3	P@5	PR-AUC	ROC-AUC	P@1	P@3	P@5
ComplEx[12]	0.7419	0.9355	0.6866	0.2537	0.1597	0.3568	0.9391	0.2212	0.1852	0.1618
KBGAN[16]	0.7562	0.9436	0.6919	0.2643	0.1612	0.3605	0.9414	0.2270	0.1869	0.1637
SimplE[11]	0.7499	0.9310	0.6927	0.2553	0.1625	0.3588	0.9396	0.2241	0.1857	0.1623
RotatE[37]	0.7676	0.9348	0.7084	0.2678	rgb 0,0,1**0.1668**	0.2138	0.8390	0.1449	0.1168	0.1020
Our model (ComplEx)	0.7615	rgb 1,0,0**0.9527**	0.7030	0.2631	0.1644	rgb 0,0,1**0.3623**	rgb 0,0,1**0.9469**	rgb 1,0,0**0.2306**	rgb 0,0,1**0.1934**	rgb 0,0,1**0.1685**
Our model (SimplE)	rgb 0,0,1**0.7693**	0.9431	rgb 0,0,1**0.7177**	rgb 0,0,1**0.2685**	0.1661	rgb 1,0,0**0.3642**	rgb 1,0,0**0.9480**	rgb 0,0,1**0.2291**	rgb 1,0,0**0.1935**	rgb 1,0,0**0.1703**
Our model (RotatE)	rgb 1,0,0**0.7899**	rgb 0,0,1**0.9480**	rgb 1,0,0**0.7296**	rgb 1,0,0**0.2762**	rgb 1,0,0**0.1722**	0.2215	0.8485	0.1501	0.1216	0.1059

Metrics

- **ROC-AUC**: the area under the receiver operating characteristic curve.
- **PR-AUC**: the area under the precision-recall curve.
- **P@K**: the mean percentage of true predicted labels among TOP-K over all samples. In this paper, $K = 1, 3, 5$ are selected as evaluation indicators to estimate the performance of models.

Training protocol

In this task, we use the models trained for link prediction. Thus, all settings and hyper-parameter configurations are retained from above.

Comparison with state-of-the-art models

The DDI classification results are displayed in Table 3. Since TransE and DistMult are comparably primitive models, their performance is not expected to be convincing for this task and the corresponding models are not included in this experiment. As can be seen from Table 3:

- Similar to the link prediction task, our proposed method achieves consistent improvements in this scenario. On both datasets, the proposed framework refines the performance of all baseline KG embedding models.
- Where the improved ComplEx method outperformed other models on link prediction, for the classification task, the improved RotatE method yields the best results on the DeepDDI dataset while the improved SimplE method obtains the best performance on Decagon.
- The performance of RotatE is highly variable across datasets. The improved model yields the best performance in four of the five metrics on the DeepDDI dataset (PR-AUC, P@1, P@3 and P@5). Even the P@5 results rank a close 2nd. However, on the Decagon dataset, RotatE and its adversarial training version show the worst results among all methods. We will discuss the specific reasons for these results in Section 5.2.
- On the DeepDDI datasets, the proposed framework can improve the performance by an average of 2 points of PR-AUC beyond the original methods. Even under the increased complexity of the Decagon dataset, we observe an average improvement of 0.6 percentage point.

Discussion

In the following we analyze and discuss the results observed in the above experiments and in order to demonstrate the effect of our model more graphically, include a visual representation of the model.

Link prediction

On both standard benchmark datasets, the KG embedding models using our adversarial framework gain better performance than existing methods on all metrics. KBGAN and RotatE also obtain good performance by introducing adversarial mechanisms to generate negative samples. The results demonstrate that the adversarial learning has the capability to construct more plausible triplets than random sampling does, and that these samples are conducive to improving the performance of the embedding models. The concrete adversarial mechanism proposed in this work differs from what is described in existing literature.

RotatE proposed a self-adversarial negative sampling mode, which selects negative triplets in accordance with scores calculated by the current KG embedding model. More specifically, in the traditional model, the weight of each negative sample in the loss function is equally large:

$$L = -\log\sigma\left(\gamma - f_r(\mathbf{h},\mathbf{t})\right) - \sum_{i=1}^{n}\frac{1}{k}\log\sigma\left(f_r\left(\mathbf{h}_i',\mathbf{t}_i'\right) - \gamma\right), \quad (21)$$

where σ denotes the sigmoid function, γ indicates a fixed margin, and $f_r(\mathbf{h}_i',\mathbf{t}_i')$ is the score of the i-th negative sample. RotatE first calculates the score of each negative sample according to the current embedding model, and then utilizes a Softmax operation to convert these scores into the weight corresponding to the negative sample. Finally, these weights are employed to calculate the total loss. Its mathematical definition is given by:

$$p\left(h_j', r, t_j' \middle| \left\{\left(h_i, r_i, t_i\right)\right\}\right) = \frac{\exp\alpha f_r\left(\mathbf{h}_j',\mathbf{t}_j'\right)}{\sum_i \exp\alpha f_r\left(\mathbf{h}_i',\mathbf{t}_i'\right)}, \quad (22)$$

$$L = -\log\sigma\left(\gamma - f_r(\mathbf{h},\mathbf{t})\right) - \sum_{i=1}^{n} p\left(h_i', r, t_i'\right)\log\sigma\left(f_r\left(\mathbf{h}_i',\mathbf{t}_i'\right) - \gamma\right), \quad (23)$$

where α indicates the temperature of sampling. Although the training time of this method is shorter than the general generative adversarial framework on account of removing the generator, the performance of the original RotatE is not as convincing as ours. As this original formulation lacks the game-style training process, the discriminator cannot compete with the generator and improve one another.

Our negative sampling strategy introduces an integral generative adversarial framework for better training of the discriminator that is similar to KBGAN. Compared with KBGAN, our proposed method has two advantages: 1) On top of the generator, we add a decoder module that turns the original negative sampling into an autoencoder framework. This scheme ensures the fake drug generated by the generator to not deviate too far from the real one, thereby improving the plausibility of the negative sampling; 2) As it is not possible to use discrete data directly in the original GANs, where the discrete sampling step prevents gradients from propagating back to the generator, KBGAN relies on RL to achieve its goal. However, increased computational cost and unstable training are inherent problems in RL. Our method can solve both problems, as shown in Figure 4.

To compare the negative samples generated by different sampling strategy more intuitively, we also visualize the corrupted triplets which are constructed by the generator and random sampling, respectively, to further stress this point in Table 4.

Finally, it should be noted that this article focuses on comparing the impact of different negative sampling methods on model performance under the same conditions. As a consequence, we only utilize ranking-based loss functions which have a single output to construct our experiments. Instead, to obtain better end-to-end performance, KG embedding could also have been cast as a multi-class evaluation problem by employing multi-class based loss formulations that may lead to better downstream performance.

DDI Classification

The same algorithm has different performance in link prediction and DDI classification tasks. This indicates that the two tasks measure different performance aspects of the KG embedding

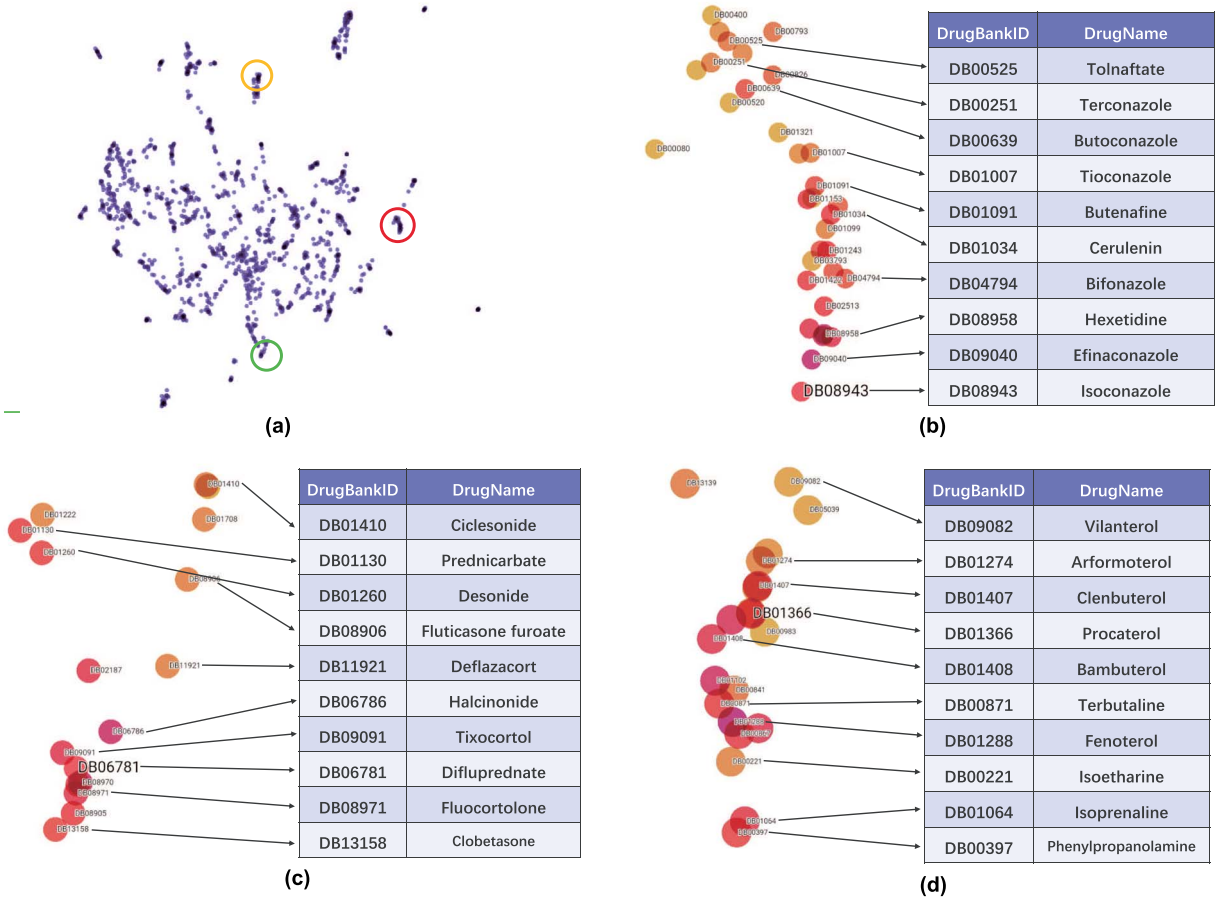

Fig. 3. Illustrations of KG embedding vectors. (a) An overview of embedding vectors after dimensionality reduction. (b) The enlarged selection in the rgb 1,0,0red circle (the right one in (a)) contains mostly anti-fungal drugs. (c) The enlarged selection in the rgb 0,1,0green circle (the bottom one in (a)) contains mostly corticosteroids. (d) The enlarged selection in the yellowyellow circle (the top one in (a)) contains asthma medication. As we can see, the drugs in each of the above circles have similar effects or categories. This illustration proves that our model follows the translation invariance criterion.

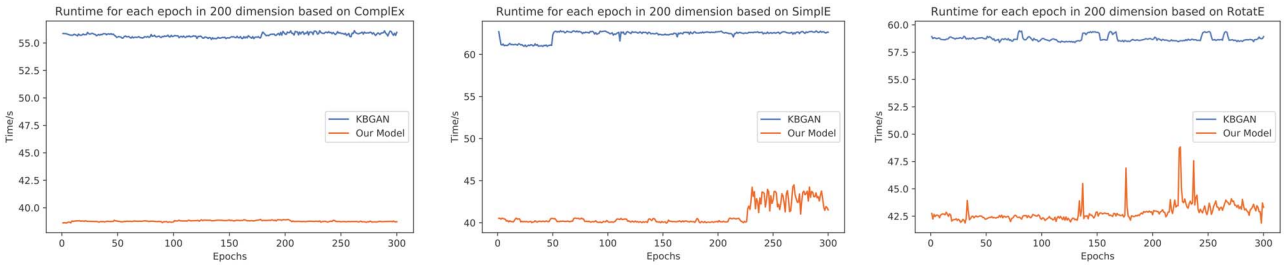

Fig. 4. Run times for each epoch on DeepDDI dataset. We plot the training times of KBGAN and our model in which ComplEx, SimplE and RotatE are as their discriminator in 200 dimensions, respectively.

model. The result stresses the flexible adaptability and extensibility of our framework to different tasks.

The performance of RotatE varies greatly between the two datasets. A likely explanation lies in RotatE's fixed composition method [37], utilizing the element-wise Hadamard product ($r_1 \circ r_2$). For instance, given data on three persons (a, b, c), where b is the elder brother (marked as r_1) of a and c is the older sister (marked as r_2) of b, we can easily infer that c is the older sister of a. The relation between c and a is r_2 rather than $r_1 \circ r_2$. Perhaps this composition method can infer additional information when the number of available relations is small, but once the number of observed relations increases, this capability will no longer yield added value.

Analysis of learned embeddings

To highlight the capabilities of our proposed framework in a qualitative manner we include two visualization experiments: negative samples constructed by random mode versus the generator, and an illustration of KG embedding vectors. It is worth to note that our model has access to only the structure of the graph in the form of triplets and is agnostic of any other features and

Table 4. Some instances of negative samples constructed by random and generator sampling from the DeepDDI dataset. In this table, all drugs are shown in boldface and all interactions are denoted by star (★). Additionally, there is a parenthesis below each drug, which contains the function or character of the drug. The triplets in the 1st column are positive, and the underlined drug signifies that it would be replaced by other drugs in the next two columns. The replacement drugs sampled randomly are listed in the 2nd column, and the 3rd column displays the drugs generated by our method.

Positive triplets	Random sampling	Generator sampling
Midazolam (hypnotic sedative)	**Lumefantrine** (antimalarial)	**Methadyl acetate** (narcotic analgesic)
★ increases the risk of adverse effects	**Diltiazem** (antihypertensive)	**Levacetylmethadol** (narcotic analgesic)
Dezocine (partial opiate)	**Pefloxacin** (antibacterial)	**Nalbuphine** (narcotic)
Treprostinil (treatment of pulmonary hypertension)	**Carmustine** (treatment of brain tumors)	**Ridogrel** (prevention of thrombo-embolism)
★ increases the antiplatelet activities	**Plicamycin** (antineoplastic antibiotic)	**Milrinone** (vasodilator)
Tirofiban (prevention of blood clotting)	**Pipazethate** (antitussive)	**Trapidil** (vasodilator and anti-platelet agent)
Indapamide (thiazide-like diuretic)	**Mefenamic acid** (anti-inflammatory)	**Hydrochlorothiazide** (thiazide diuretic)
★ decreases the metabolism	**Rolapitant** (Neurokinin-1 receptor antagonist	**Chlorthalidone** (thiazide-like diuretic)
Saquinavir (HIV protease inhibitor)	**Kappadione** (Vitamin K derivative)	**Chlorothiazide** (thiazide diuretic)

properties. Therefore, we cannot easily utilize visualization via attention mechanisms, as our input does not represent specific features or characteristics.

Table 4, compares traditional random negative samples with generated ones. We can note that the generator is able to select more semantically relevant drugs as negative samples. For instance, given a real triplet (*Midazolam, increases the risk of adverse effects, Dezocine*), the generator adopts three tail drugs, i.e. *Methadyl acetate*, *Levacetylmethadol* and *Nalbuphine*. All three drugs, similar to *Dezocine*, have narcotic effects. Thus, the negative triplets constructed by these drugs are more plausible and potentially deceptive.

Given such high-quality negative triplets, we can train better KG embedding models which have strengthened representation and generalization capabilities. Similar to word embedding, KG embedding also follows a basic principle that entities with similar connotation should have similar representations. We demonstrate this by projecting the trained drug vectors into a two-dimensional space to validate whether they satisfy this principle. Figure 3 shows an illustration of KG embedding vectors after dimensionality reduction.

Embedding vectors are projected to two-dimensional space by applying UMAP dimensionality reduction [45] (Figure ??). We select three regions of the embedding space and zoom in on them to observe if the drugs in those areas are related in indication or category. Figure ?? lists 10 drugs in the red circle with consistently anti-fungal effect. Similarly, the vast majority of drugs in the yellow region are corticosteroids, and the green region contains asthma medication. This illustration also intuitively supports the effectiveness of our model from a qualitative perspective.

It should be noted that this article focuses on improving the performance of KG embedding models, while ignoring the comparison with clinical trials or traditional machine learning based methods. These older methods often do not provide datasets or the datasets are too small to train modern KGE models such as ours. We hope to remedy this limitation in future work.

Complexity and training time analysis

The training time complexity of all models introduced in this experiment, including TransE, DistMult, ComplEx, SimplE and RotatE, is $\mathcal{O}(d)$ where d denotes the dimensionality of the embedding space. Since KBGAN and our proposed framework both consist of two parts, a generator (an autoencoder for our framework) with $\mathcal{O}(d)$ time complexity and a discriminator with $\mathcal{O}(d)$ time complexity, the time complexity of these two frameworks remains linear in d. However, as we mentioned above, policy gradient based methods can be unstable in the training process, while our proposed method can complete optimization more efficiently.

Figure 4 plots the runtime of 300 training epochs of KBGAN and our model on DeepDDI. Both models use ComplEx, SimplE and RotatE as their discriminator. The result indicates that our method effectively shortens the training time compared with policy gradient based GANs. Moreover, because our model and KBGAN both can be divided into two major modules: generator and discriminator, the number of parameters will be greater than that of a single embedding model. Thus, although their formal complexity is the same as that of single embedding models, these two adversarial learning frameworks will require more time to complete the training process.

In addition to the complexity of the model, the size of the KG, including the total number of entities and relations and the resulting number of triplets, are also significant indicators that affect training time. Models on the DeepDDI dataset require more time to train than those on Decagon, because the number of entities in DeepDDI is significantly greater than in Decagon.

Conclusions

The goal of this study is to find a new approach to negative sampling that improves the performance of DDI KG embedding models. In this paper, we propose an adversarial learning framework based on Wasserstein distances for this task. We evaluate the proposed method on link prediction and DDI classification tasks. Our experiments on two standard collections confirm that the performance of all baseline models can be significantly improved using our adversarial learning framework.

The approach has several major advantages over existing KG embedding models. First, we introduce an AAE framework to represent DDI KGs. The autoencoder is employed to generate more plausible drugs as negative samples, and these negative triplets are fed to the discriminator along with authentic positive ones for improving the performance of embedding models. Our approach also utilizes a Gumbel-Softmax relaxation and Wasserstein distance to handle vanishing gradient issues on discrete data. Compared with traditional policy gradients in RL, the proposed method can complete optimization tasks more efficiently. Most notably, the work presented here can be applied to refine the performance of most existing models without the need for major modifications.

Our method is not limited to the DDI domain. Going beyond the application and scope of this immediate work, future work will include evaluating the benefits the model presented here holds for other graph embedding tasks such as recommendation, classification and retrieval settings on hierarchical data.

Supplementary data

Supplementary data and code are available at https://github.com/dyf0631/AAE_FOR_KG.

Funding

This work is supported by the China Scholarship Council (grant no. 201906650004) and the National Natural Science Foundation of China (nos. U1705262 and 61672159).

References

1. Cheng F, Zhao Z. Machine learning-based prediction of drug–drug interactions by integrating drug phenotypic, therapeutic, chemical, and genomic properties. *J Am Med Informat Assoc* 2014; **21**(e2): e278–e286.
2. Giacomini KM, Krauss RM, Roden DM, *et al*. When good drugs go bad. *Nature* 2007; **446**(7139): 975–977.
3. Percha B, Altman RB. Informatics confronts drug–drug interactions. *Trends Pharmacol Sci* 2013;34(3): 178–184.
4. Jia J, Zhu F, Ma X, *et al*. Mechanisms of drug combinations: interaction and network perspectives. *Nat Rev Drug Discov* 2009; **8**(2): 111–128.
5. Palleria C, Di Paolo A, Giofrè C, *et al*. Pharmacokinetic drug–drug interaction and their implication in clinical management. *J Res Med Sci* 2013; **18**(7): 601.
6. Ma T, Shang J, Xiao C, Sun J. Genn: Predicting correlated drug–drug interactions with graph energy neural networks. In: *Proceedings of the 2020 International Conference on Learning Representations*, 2020.
7. Ryu JY, Kim HU, Lee SY. Deep learning improves prediction of drug–drug and drug–food interactions. *Proc Natl Acad Sci* 2018; **115** (18): 4304–4311.
8. Zitnik M, Agrawal M, Leskovec J. Modeling polypharmacy side effects with graph convolutional networks. *Bioinformatics* 2018; **34**(13): i457–i466.
9. Bordes A, Usunier N, Garcia-Duran A, *et al*. Translating embeddings for modeling multi-relational data. In: *Advances in Neural Information Processing Systems*. New York: Curran Associates, Inc., 2013, 2787–2795.
10. Jia Y, Wang Y, Lin H, *et al*. Locally adaptive translation for knowledge graph embedding. In: *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, Palo Alto: AAAI Press, 2016, 992–998.
11. Kazemi SM, Poole D. Simple embedding for link prediction in knowledge graphs. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, New York: Curran Associates, Inc., 2018, 4289–4300.
12. Trouillon T, Welbl J, Riedel S, *et al*. Complex embeddings for simple link prediction. In: *International Conference on Machine Learning*, Brookline: JMLR, Inc., 2016, 2071–2080.
13. Schroff F, Kalenichenko D, Philbin J. Facenet: A unified embedding for face recognition and clustering. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Palo Alto: AAAI Press, 2015, 815–823.
14. Hermans A, Beyer L, Leibe B. In defense of the triplet loss for person re-identification. *Preprint, arXiv: 1703.07737*, 2017.
15. Wang P, Li S, Pan R. Incorporating gan for negative sampling in knowledge representation learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, 2018.
16. Cai L, Wang WY. Kbgan: Adversarial learning for knowledge graph embeddings. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, vol. 1, Massachusetts: Association for Computational Linguistics, 2018, 1470–1480.
17. Jang E, S. Gu, Poole B. Categorical reparameterization with gumbel-softmax. In: *Proceedings of the 7th International Conference on Learning Representations*, Massachusetts: openreview.net, 2017.
18. Maddison CJ, Mnih A, Teh YW. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the 7th International Conference on Learning Representations*, 2017.
19. Krishna G, Moton A, Ma L, *et al*. Pharmacokinetics and absorption of posaconazole oral suspension under various gastric conditions in healthy volunteers. *Antimicrob Agents Chemother* 2009; **53** (3): 958–966.
20. Ho PM, Maddox TM, Wang L, *et al*. Risk of adverse outcomes associated with concomitant use of clopidogrel and proton pump inhibitors following acute coronary syndrome. *JAMA* 2009; **301**(9): 937–944.
21. Menon RM, Badri PS, Wang T, *et al*. Drug–drug interaction profile of the all-oral anti-hepatitis c virus regimen of paritaprevir/ritonavir, ombitasvir, and dasabuvir. *J Hepatol* 2015; **63**(1): 20–29.
22. Li P, Huang C, Fu Y, *et al*. Large-scale exploration and analysis of drug combinations. *Bioinformatics* 2015; **31**(12): 2007–2016.
23. Kastrin A, Ferk P, Leskošek B. Predicting potential drug-drug interactions on topological and semantic similarity features using statistical learning. *PLoS One*, **13**(5), 2018.
24. Muñoz E, Nováček V, Vandenbussche P.-Y. Facilitating prediction of adverse drug reactions by using knowledge graphs and multi-label learning models. *Brief Bioinform* 2019; **20**(1): 190–202.
25. Abdelaziz I, Fokoue A, Hassanzadeh O, *et al*. Large-scale structural and textual similarity-based mining of knowledge graph to predict drug–drug interactions. *J Web Semant* 2017; **44**:104–117.
26. Celebi R, Yasar E, Uyar H, *et al*. *Evaluation of knowledge graph embedding approaches for drug-drug interaction prediction using linked open data*. In: *Proceedings of the 11th International Conference Semantic Web Applications and Tools for Life Sciences*, Aachen: CEUR-WS.org, 2018.
27. Ma T, Xiao C, Zhou J, *et al*. Drug similarity integration through attentive multi-view graph auto-encoders. In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, 3477–3483.
28. Karim MR, Cochez M, Jares JB, *et al*. Drug–drug interaction prediction based on knowledge graph embeddings and convolutional-lstm network. In: *Proceedings of the 10th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, New York: Association for Computing Machinery, 2019, 113–123.
29. Wang Q, Mao Z, Wang B, *et al*. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, **29**(12): 2724–2743, 2017.
30. Ji S, Pan S, Cambria E, *et al*. A survey on knowledge graphs: Representation, acquisition and applications. Preprint arXiv: 2002.00388, 2020.
31. Mikolov T, Sutskever I, Chen K, *et al*. Distributed representations of words and phrases and their compositionality. In

Advances in Neural Information Processing Systems, New York: Curran Associates, Inc., 2013, 3111–3119.
32. Wang Z, Zhang J, Feng J, Chen Z. Knowledge graph embedding by translating on hyperplanes. In: *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014, 1112–1119.
33. Lin Y, Liu Z, Sun M, Y. Liu, Zhu X. Learning entity and relation embeddings for knowledge graph completion. In: *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, 2015, 2181–2187.
34. Xiao H, Huang M, Zhu X. Transg: A generative model for knowledge graph embedding. In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume **1**, 2016, 2316–2325.
35. Nickel M, Tresp V, Kriegel H.-P. A three-way model for collective learning on multi-relational data. In: *Proceedings of the 28th International Conference on International Conference on Machine Learning*, Brookline: JMLR, Inc., 2011, 809–816.
36. Yang B, Yih SW, He X, *et al*. Embedding entities and relations for learning and inference in knowledge bases. In: *Proceedings of the 2015 International Conference on Learning Representations*, Massachusetts: openreview.net, 2015.
37. Sun Z, Deng Z.-H, Nie J.-Y, Tang J. Rotate: Knowledge graph embedding by relational rotation in complex space. In: *International Conference on Learning Representations*, Massachusetts: openreview.net, 2019.
38. Nguyen DQ, Nguyen TD, Nguyen DQ, Phung D. A novel embedding model for knowledge base completion based on convolutional neural network. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, vol. 2, 2018, 327–333.
39. Minervini P, Demeester T, Rocktäschel T, Riedel S. Adversarial sets for regularising neural link predictors. In: *Uncertainty in Artificial Intelligence-Proceedings of the 33rd Conference, UAI 2017*. Curran Associates Inc., 2017.
40. De Cao N Kipf T. MolGAN: An implicit generative model for small molecular graphs. In: *ICML 2018 workshop on Theoretical Foundations and Applications of Deep Generative Models*, Brookline: JMLR, Inc., 2018.
41. Arjovsky M, Chintala S, Bottou L. Wasserstein generative adversarial networks. In: *International Conference on Machine Learning*, 2017, 214–223.
42. Villani C. *Optimal Transport: Old and New*, vol. 338. Berlin: Springer Science & Business Media, 2008.
43. Wishart DS, Knox C, Guo AC, Cheng D, Shrivastava S, Tzur D, Gautam B, Hassanali M. Drugbank: a knowledgebase for drugs, drug actions and drug targets. *Nucleic Acids Res* 2008; **36**(suppl_1): D901–D906.
44. Tatonetti NP, Patrick PY, Daneshjou R, Altman RB. Data-driven prediction of drug effects and interactions. *Sci Transl Med* 2012; **4**(125): 125ra31–125ra31.
45. McInnes L, Healy J, Melville J. Umap: Uniform manifold approximation and projection for dimension reduction. *Journal of Open Source Software* 2018; **3**(29): 861.

Appendix

A Parameter settings

On the DeepDDI dataset, the training and testing time of the our proposed KG embedding model is $23h : 58m : 02s$ for ComplEx; $18h : 28m : 48s$ for SimplE and $19h : 04m : 18s$ for Rotate in 200 dimensions. On the Decagon dataset, the training times are $1d : 19h : 37m : 56s$ for ComplEx; $2d : 3h : 23m : 15s$ for SimplE and $1d : 22h : 19m : 22s$ for Rotate in 200 dimensions. The parameter settings of the original embedding models are listed as follows: $\{\alpha = 0.5, d = 200, m = 512, e = 1000\}$ for TransE; $\{\alpha = 0.5, d = 200, m = 1024, e = 700\}$ for DistMult; $\{\alpha = 0.5, d = 200, m = 1024, e = 500\}$ for ComplEx; $\{\alpha = 0.5, d = 200, m = 1024, e = 1000\}$ for KBGAN; $\{\alpha = 0.5, d = 200, m = 512, e = 500\}$ for SimplE; $\{\alpha = 0.5, d = 200, m = 1024, e = 1000\}$ for RotatE. In the actual experiment, we did not conduct a grid search for the inverse temperature value. Instead, we directly called the function *gumbel_softmax* in Pytorch (https://pytorch.org/docs/stable/nn.functional.html?highlight=gumbel_softmax∖#torch.nn.functional.gumbel_softmax), which can well ensure the output of the encoder to be one-hot encoded when its parameter *hard* is set to True. In addition, the clipping parameter c is a constant, which is equal to 0.99. Note that we used the original author's code of KBGAN (https://github.com/cai-lw/KBGAN) to construct experiments for KBGAN. Finally, the ratio of positive and negative samples in the training process is 1: 1.

B Graphical representation of model performance

A visual representation of the results is shown in Figure B5.

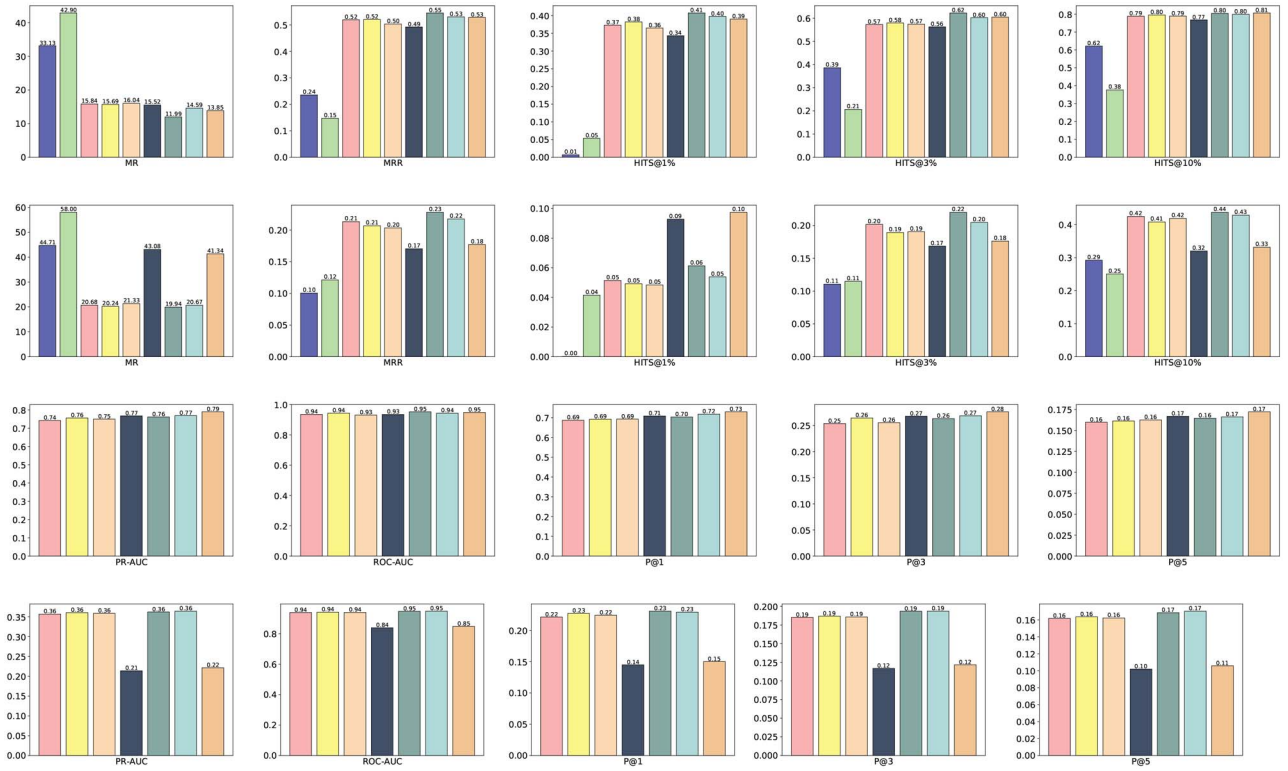

Fig. B5. A visual representation of the results. The first two lines represent the results for link prediction task on Deepddi and Decagon dataset respectively; The last two lines represent the results for triplets classification task on Deepddi and Decagon dataset respectively. The ordering of all algorithms in the figure is consistent with that in Table 2 and 3.