On Regularizing Neural Link Predictors

Abstract

Since neural link predictors are thought to have poor generalization in some particular circumstances, regularization is commonly required during the training. In this paper, we attempt to introduce various regularization methods to the factorization-based neural link predictors and evaluate how they impact the task of knowledge graph completion (KGC). We propose three novel regularizers, including gradient penalty, multi-task learning and manifold regularization. Extensive and comprehensive experiments are conducted to analyze how the proposed methods could potentially help KGC tasks. Based on the experiment results on benchmark datasets FB15k-237 and WN18RR, we find both gradient penalty and multi-task learning methods could bring significant generalization improvement to neural link predictors.

1 Introduction

As the result of constructing large-scale knowledge graphs (KG) such as Freebase [Bollacker *et al.*, 2008], DBpedia [Bizer *et al.*, 2009] and YAGO [Suchanek *et al.*, 2007], more entities with few or zero relations were added to the KG. These missing entries resulted in an incomplete structure of the knowledge graph. Therefore, it was crucial to investigate the implicit relationships among entities or relations to recover the missing facts and construct a complete KG for real-world applications. The research on Knowledge Graph Completion (KGC) was proposed to tackle such a problem.

During the past decade, the neural link predictor, a kind of Knowledge Graph Embedding (KGE) Model, has become more and more popular in the research of KGC tasks. This method focuses on learning low-dimensional representation (embeddings) for entities and relations based on existing triples, and then uses the learned embeddings to evaluate the plausibility of new facts through a scoring function.

However, neural link predictors are thought to have poor generalization [Trouillon *et al.*, 2016] in some particular circumstances. To get out of such a predicament, regularization is commonly required during the training of KGE models.

In this paper, inspired by the regularizers in latent space representation models [Hoffman *et al.*, 2019; Rifai *et al.*, 2011; Thanh-Tung *et al.*, 2019; Ghosh *et al.*, 2020], multi-task learning [Moradi *et al.*, 2020; Kim *et al.*, 2020], and matrix factorization [Cai *et al.*, 2010], we propose 3 new regularizers for KGC tasks with the hope of improving the model generalization performance. Specifically, the methods include *gradient penalty*, *multi-task learning*, and *manifold regularization*.

2 Background and related work

A knowledge graph $\mathcal{G}$ is defined by a set of entities nodes $\mathcal{E}$ and a set of relations $\mathcal{R}$. The data stored in the knowledge graph is formed as factual triples $\langle s, r, o \rangle$, where each triple represents a connection between subject entity s and object entity o with relation type r. It is noticeable that the subjects and the objects are from the same set of entities, so the knowledge graph lies in a 3-order space, with $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$.

Knowledge Graph Completion The general Knowledge Graph Completion (KGC) tasks include complementing the missing entities, relations, or even attributes. In this paper, we are going to focus on a specific type of KGC problem called neural link prediction.

Neural link prediction intends to predict the missing entries by mining the facts in the knowledge graph and learning the representation for each entity and relation. The completion task forms its dataset as follows. The training set consists of a series of triples that are known to hold true in a knowledge graph, denoted as $\mathcal{S} = \left\{ (s_1, r_1, o_1), \ldots, \left(s_{|\mathcal{S}|}, r_{|\mathcal{S}|}, o_{|\mathcal{S}|}\right) \right\} \subseteq \mathcal{G}$. While the queries in the validation and test sets come with

the form (subject, relation, ?) or (?, relation, object), the model is required to find out the index of the missing entities.

We would like to answer the query $\langle s, r, ?\rangle$ (similarly to $\langle ?, r, o\rangle$) by finding the object entity o^* that has the highest conditional probability $P_\theta(o^* \mid s, r)$, where θ is the trainable parameters in the model. An intuitive way to solve this problem is by calculating $P_\theta(o' \mid s, r)$ for all $o' \in \mathcal{E}$, which include $|\mathcal{E}| = N_e$ possible entities, and finding the one with the highest possibility, noted as o^*. The likelihood of $P_\theta(o \mid s, r)$ can be estimated by normalizing a parametric score function $\phi_\theta(s, r, o)$, which is,

$$P_\theta(o \mid s, r) = \frac{\exp(\phi_\theta(s, r, o))}{\sum_{o'} \exp(\phi_\theta(s, r, o'))}$$

Score function: Neural link predictors can be characterized by their scoring function ϕ_θ. Formally, we will use $\mathbf{e}_s$, $\mathbf{e}_o$ and $\mathbf{w}_r$ to denote the embedding of a subject s, object o and relation r, and in this paper we are particularly interested in the factorization-based models. For instance, DistMult[Yang *et al.*, 2015] defined the score function as $\phi_\theta(s, r, o) = \sum_k e_s^k w_r^k e_o^k := \langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o\rangle$, where $\langle \cdot, \cdot, \cdot\rangle$ denotes the tensor inner product. Canonical Tensor (CP) Decomposition[Hitchcock, 1927] uses 2 distinguished representation for an entity when it is used for subject or object, and use a same tensor inner product to calculate the score function. ComplEx[Yuan and Zhang, 2016] extends DistMult to solve the problem of symmetry and anti-symmetry by introducing complex numbers to the embeddings, which has the score function $\phi(s, r, o) = \mathrm{Re}(\langle \mathbf{e}_s, \mathbf{w}_r, \bar{\mathbf{e}}_o\rangle)$, where $\mathrm{Re}(x)$ is the real part of x. Tucker[Balazevic *et al.*, 2019] introduces a core matrix to the model, and this core tensor works as an information compression for the original tensor. It has the score function $\phi(s, r, o) = \mathcal{Z} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o$.

2.1 Training objective

Neural link predictors could be trained by a large range of loss functions, e.g ranking losses, binary logistic regression or sampled multi-class log-loss. In this thesis, we will follow the convention in [Lacroix *et al.*, 2018] to use multi-class log loss during the training stage. The loss function can be interpreted as the negative summation of subject and object log-likelihood, which is

$$\mathcal{L} = -\sum_{\langle s,r,o\rangle \in \mathcal{S}} [\log P_\theta(s \mid r, o) + \log P_\theta(o \mid s, r)] \tag{1}$$

We further introduce loss term for each training triple $\langle s, r, o\rangle$ to simplify the expression, which is $\ell_{s,r,o} = -\log P_\theta(s \mid r, o) - \log P_\theta(o \mid s, r)$.

2.2 Regularizers

Previous studies suggest that, without a regularizer, the factorization-based models could trivially minimize the loss $\mathcal{L}$ by increasing the embeddings norm[Bordes *et al.*, 2013], which would impair the testing performance. So it is necessary to introduce regularizations to the models. Thus, the loss function with a regularizer can be written as

$$\begin{aligned}\mathcal{L} &= \sum_{s,r,o\in S} \ell_{s,r,o} + \lambda \mathbf{R}(\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r) \\ &= \sum_{s,r,o\in\mathcal{S}} (\ell_{s,r,o} + \sum_{\mathbf{z}\in\{\mathbf{e}_s,\mathbf{e}_o,\mathbf{w}_r\}} \lambda \mathbf{R}(\mathbf{z}))\end{aligned}$$

For simplicity, we use $\mathbf{z} \in \mathbb{C}^K$ to denote the embedding vector instead of the conventional notation $\mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o$ in the following section.

To the best of our knowledge, all prior researches on regularizing neural link prediction were about norm-based methods, which aimed at preventing large values of the embedding and minimizing the rank in tensor decomposition. Among them, l_1 and l_2 norm of embeddings were most frequently used. For example, the regularizers used by [Bordes *et al.*, 2013] and [Trouillon *et al.*, 2016] are simply $\mathbf{R}(\mathbf{z}) = \|\mathbf{z}\|_1$ and $\mathbf{R}(\mathbf{z}) = \|\mathbf{z}\|_2$

More recent work [Lacroix *et al.*, 2018; Yuan and Zhang, 2016] started to consider using tensor norm as a regularizer instead of simple embeddings norms.[Yuan and Zhang, 2016] first proposed to use tensor trace (nuclear) norm as a regularizer, and [Lacroix *et al.*, 2018] used tensor nuclear-3 norm as the regularization. Their work suggested that the nuclear norm can work as an approximation of the tensor rank. While in the factorization models that we consider, the nuclear-3 (n_3) norm works exactly the same as a l_3 norm of each embedding. This is a simple yet efficient method to minimizing the rank in tensor factorization and is proved to be significantly beneficial for training factorization-based neural link predictors.

In this paper, we will also look into how our methods perform when combined with these norm-based regularizers and try to find the intuition behind them.

3 Our regularizers

3.1 Norm-based regularization

Inspired by the huge success of elastic net [Zou and Hastie, 2005], our first attempt is to combine l_1, l_2 and n_3 norms and define a new regularizer:

$$\mathbf{R}(\mathbf{z}) = \lambda_1 \|\mathbf{z}\|_1 + \lambda_2 \|\mathbf{z}\|_2 + \lambda_3 \|\mathbf{z}\|_3$$

3.2 Gradient Penalty

Gradient penalty has been widely applied to latent space models, e.g. Generative Adversarial Neural Networks (GANs) [Thanh-Tung *et al.*, 2019]

and Variational Auto-Encoders (VAEs) [Rifai *et al.*, 2011]. Neural link predictors are also latent space models with the encoder being embedding lookup functions and the decoder being score functions [Hamilton, 2020]. Thus, we are curious whether the gradient penalty methods in generative models could also work on the scheme of neural link predictors. In neural link predictors, we only consider applying gradient penalty to the decoder part (score function) since the encoder part is simply a lookup function and not differentiable.

The input vector would be the embedding vectors $\mathbf{z} \in \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o$ and the output would the score function $\phi_\theta(s, r, o)$. Formally, denoting the decoder function as $y = f(\mathbf{z})$, a small perturbation applied to the input $\mathbf{z}$ can be expressed as $\mathbf{z} + \boldsymbol{\epsilon}$. According to Taylor expansion, the corresponding function output will be changed to

$$f(\mathbf{z} + \boldsymbol{\epsilon}) = f(\mathbf{z}) + \sum_{i=1}^{K} \epsilon_i \cdot \frac{\partial f}{\partial z_i}(\mathbf{z}) + O\left(\boldsymbol{\epsilon}^2\right)$$

If we wish to minimize the output change, that is $f(\mathbf{z} + \boldsymbol{\epsilon}) - f(\mathbf{z})$, it is equivalent to minimizing the term $\sum_{i=1}^{K} \epsilon_i \cdot \frac{\partial f}{\partial z_i}(\mathbf{z})$ by neglecting the second order infinitesimal $O\left(\epsilon^2\right)$. That is to say, the model output change caused by input perturbation $\boldsymbol{\epsilon}$ is governed by the so-called Jacobian function

$$J_i(\mathbf{z}) \equiv \frac{\partial f}{\partial z_i}(\mathbf{z}), \qquad i \in \{1, \ldots, K\}$$

.

Thus, minimizing the norm of Jacobian function $\|\mathbf{J}(\mathbf{z})\|_p$ would work as a regularizer to make the model insensitive to input noise. The inputs of the model are the embeddings of the entities and the relations, $\mathbf{e}_s$, $\mathbf{w}_r$ and $\mathbf{e}_o$. By introducing the l_2 gradient penalty to our model, we can now form the new loss function as

$$\mathcal{L} = \sum_{s,r,o \in \mathcal{S}} (\ell_{s,r,o} + \sum_{\mathbf{z} \in \{\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r\}} \lambda \|\mathbf{J}(\mathbf{z})\|_2)$$

the output function $f(\mathbf{z}) = \phi_\theta(s, r, o)$, so

$$\mathbf{J}_i(\mathbf{z}) \equiv \frac{\partial \phi_\theta(s, r, o)}{\partial \mathbf{z}} \mid_i, \qquad i \in \{1, \ldots, K\} \quad (2)$$

To clarify, another method to calculate the gradient penalty is by taking the gradient of the multi-class output of the models with respect to the embeddings. In the multi-class classification case when there are in total C classes, the output would also be a vector, denoted as $\vec{f} = \{f_1, f_2, \ldots, f_C\}$. Then a Jacobian Matrix $\mathbf{J}$ would be constructed as

$$\mathbf{J}_{c,i}(\mathbf{z}) \equiv \frac{\partial f_c}{\partial z_i}(\mathbf{z})$$

with $c \in \{1, \ldots, C\}$ and $i \in \{1, \ldots, K\}$

We first attempted to calculate the gradient of the multi-class output of the models with respect to the embeddings, but it turned out to be computationally inefficient. The embedding size K is usually chosen to be 500-4000, and the multi-class output is a vector of size N_e (N_e is the number of entities in a dataset, in FB15k-237 there are 14504 entities.). So in this situation, the Jacobian Matrix would be $N_e \times K$ and the tensor to be computed in the GPU space will have the size $b \times N_e \times K$ (b is the batch size), which would be computationally intractable.

So we instead calculate the Jacobian matrix w.r.t score function as in Equation (2), reducing the tensor size to be $b \times K$. In this way the computational resources are saved and we can still obtain the gradient penalty.

3.3 Multi-task Learning as a Regularizer

Graph representation learning algorithms like Node2Vec, Struc2Vec, use embedding to encode the graph structure. Inspired by this, we consider predicting the graph features to be helpful in the training of neural link predictors. In this part, we will manually construct graph features based on factual triples and their multi-hop relationships. And the model will be asked to predict features during training as an auxiliary tasks, which can be viewed as a regularizer.

Former studies[Dobrowolska *et al.*, 2021; Galkin *et al.*, 2021] have suggested several ways to design node representation features. Based on their work, we develop three types of feature representations, respectively called in-range and in-domain (IRID) feature representation, random paths representation, and NodePiece representation.

IRID Feature Representation In-range and in-domain (IRID) feature representation utilizes the relation types to construct the features. Considering that the type of the relationship could be highly correlated to its subject or object, we can aggregate all the relation types that are directly connected to an entity to construct a feature. Specifically, given a triple $\langle s, r, o \rangle$, we say relation r is in the range of subject s and in the domain of object o. Thus two clauses can be defined, namely in-range and in-domain:

$$\text{in-range}(e, r) = \begin{cases} 1 & \forall o, \text{if } \exists (e, r, o) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{in-domain}(e, r) = \begin{cases} 1 & \forall s, \text{if } \exists (s, r, e) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

If we use both in-range and in-domain features for all relations to represent a node entity, we can easily get a binary vector, $\vec{h} \in \mathbb{R}^{2N_r}$. An example of IRID representation can be found in Figure 1a.

Random Paths Feature Representation The idea of Random Paths feature representation was first proposed from the work of [Das *et al.*, 2020]. The algorithm works as follows: First, for each node entity, n random paths[1] starting from this node are sampled, and each path can be expressed as a sequence of entities and relations, e.g. $p = (e_1, r_1, e_2, \dots e_{K-1}, r_{K-1}, e_K)$, in which e_i's are the nodes this path walks through and r_i's are the relations that connect these nodes.

This path sampling method aims to build a subgraph around an entity and find out the entities and relations that are closely linked to the target node. Although it is intuitive to store all the information (both the node indices and the relation types) in the path to constructing the feature, the feature complexity would result in a giant number ($N_e \times (N_e + 2N_r)$) and the accompanying heavy computation do not allow us to do so. A more efficient way is to only store a set of anchor nodes in the feature vector, which will be discussed in the next section. But for now, the feature vector is constructed only by the types of relations a path travels along. The direction of the relation (forward and inverse) is considered in our work, which leads to a vector of size $N_e \times 2N_r$. For simplicity, we still construct a binary feature, where 1 represents the case when a relation appears in the sampled path p, and 0 otherwise.

Here we give a simple example of this method in Figure 1b. Assuming there are 5 relations types, r_1, r_2, r_3, r_4, r_5 and we sampled two length-3 paths starting from e_1, one is $(e_1, r_1, e_2, r_2, e_3, r_5, e_2)$ and the other is $(e_1, r_1, e_4, r_2, e_5, r_3', e_6)$, where r_3' is the inverse of r_3. Since we do not consider the sequence of nodes and relations in a path, the constructed feature is $[1, 1, 0, 0, 1, 0, 0, 1, 0, 0]$.

Using random paths representation as a training objective is more tricky since the existence of an entity on a path is completely new information to the model, and it requires the algorithms to encode more knowledge into the embeddings.

NodePiece Feature Representation Aside from the relation types, the entities that are relevant to the target node are also useful for constructing the node feature. However, since the number of entities in a knowledge graph is usually giant, it is computationally not feasible to construct a feature vector to identify all the entities. Thus, we refer to the idea in [Galkin *et al.*, 2021] and consider constructing the features based on a small subset of the graph entities, which are called anchor nodes.

Specifically, given a knowledge graph $\mathcal{G} = (\mathcal{E}, \mathcal{R})$ with N_e entities and N_r relation types, the task is to use fixed-size of anchor nodes and all the relation types to form a vocabulary set and represent all the entities. Then when we construct the features, the feature dimension would be largely reduced. For instance, assuming that we successfully select $|A|$ nodes as anchors, each anchor node a_i has the shortest path p directing to the target node v, which records $|A|$ paths. We keep the k nearest anchors by measuring the distance between a_i's and v. Then the index of these anchor nodes and the relation types along the paths could be used to represent the target nodes. An example can be found in Figure 1c.

The anchor nodes can be either selected randomly or based on importance measurement. For our purpose, degree, centrality or Personalized PageRank (PPR) [Hamilton, 2020] on nodes would be combined to determine the anchor nodes. And we continue to use the convention in [Galkin *et al.*, 2021] by sampling 40% of the anchor nodes by Node Centrality, 50% by Personalized PageRank and 10% by random sampling. Similar to the IRID feature construction, we used a binary identity function to construct the feature. For each entity node v, the feature is decomposed into 2 parts, the anchor node representation part $\vec{h}^a$, and the relation representation part. The first part could be expressed as,

$$\vec{h}_i^a = \begin{cases} 1 & \text{if } a_i \text{ is one of the } k\text{-nearest anchors} \\ 0 & \text{otherwise} \end{cases}$$

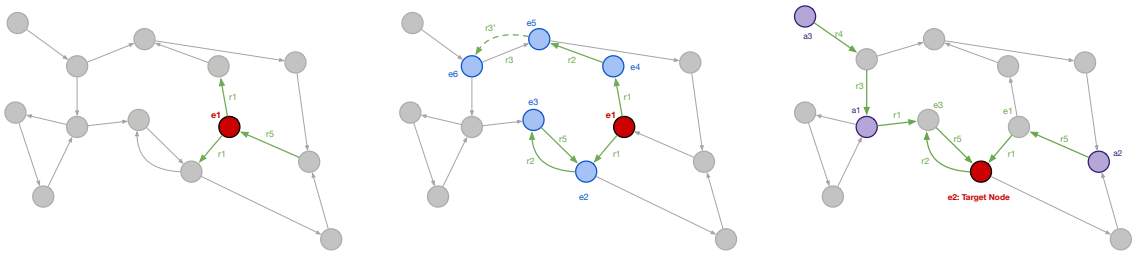
If node a_i is selected to be an anchor node and appear in the k-nearest anchors of node v, we then find k paths between all selected a_i's and v, which is $p_i = (e_1, r_1, e_2, \dots e_{K-1}, r_{K-1}, e_K)$. We again record the relation that appears in the path to form a relation feature.

$$\vec{h}_i^r = \begin{cases} 1 & \text{if relation } r \text{ appers in } p_i \\ 0 & \text{otherwise} \end{cases}$$

Then the feature vectors $\vec{h}_i^a$ and $\vec{h}_i^r$ are concatenated and finally forming the $\vec{h} \in \mathbb{R}^{|A|+N_r}$. The feature matrix then is $\mathbf{H} \in \mathbb{R}^{N_e \times (|A|+N_r)}$. In the experiment, we test a range of the number of anchors $|A| \in [200, 500, 1000]$ and number of neighbors $k \in [5, 20, 50, 100]$.

The auxiliary task design Currently, we only consider constructing features for entities, which can be denoted as $\mathbf{H} = \{\vec{h}_1, \dots \vec{h}_i, \dots, \vec{h}_{N_e}\}^T \in \mathbb{R}^{N_e \times F}$, where N_e is the number of entities and F is the length of the feature vectors. As all the features constructed are binary, the model design for the feature prediction tasks is very simple. We feed the entities embeddings to a dense layer $g(\cdot)$ with sigmoid activation to reconstruct the features and use binary cross-entropy loss to train the model. The overall loss now becomes:

[1] n is a hyper-parameter to be tuned, in our experiment we test a range of $n \in [5, 10, 100, 1000]$.



(a) An example of IRID representation

(b) An example of Random Paths representation

(c) An example of NodePiece representation

Figure 1: Examples of three feature construction methods

$$\mathcal{L} = \sum_{s,r,o \in \mathcal{S}} \ell_{s,r,o} + \lambda L'\left(g\left(\mathbf{E}\right), \mathbf{H}\right)$$

3.4 Manifold Regularization

Manifold regularization was first used in matrix factorization [Cai *et al.*, 2010]. For tensor factorization models in neural link predictors, the intuition is that if two data entities e_i and e_j lies closely in the intrinsic geometry space, their embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ should also be close. If the similarity between embeddings is measured by the l_2 distance, we can get the regularization as

$$\mathbf{R}_k = \sum_{i,j=1}^{N} \|\mathbf{e}_i - \mathbf{e}_j\|_2^2 \mathbf{W}_{ij}$$

The distance between the embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ is minimized according to the amplitude of a penalization weight $\mathbf{W}_{ij}$, which is determined by the similarity between entities e_i and e_j. By this definition we formalize the manifold regularization in neural link predictors, and the loss function now becomes

$$\mathcal{L} = \sum_{s,r,o \in S} \ell_{s,r,o} + \lambda \sum_{i=1}^{K} \mathbf{R}_k$$

Similarity Matrix Construction Manifold regularization needs to access a distance matrix $\mathbf{D}$ to retrieve the similar entities in the knowledge base. To construct a similarity matrix and calculate the weight W_{ij}, we would adopt the feature construction methods in chapter 4, and use the feature vectors as a representation for entities.

To improve the representative power, we will use integer (frequency) features instead of binary features. The methods of NodePiece representation are reused, but now we count the times that all paths travel through a specific relation r. Similar to binary NodePiece representation, this would finally form a feature vector for each entity $e_i \in \mathcal{E}$, with $\vec{h_i} \in \mathbb{R}^{N_e+2N_r}$, but now each entry of the vector is the number of times we have traveled through relation r. And the Euclidean distance $\mathbf{D}_{ij} = \|\vec{h_i} - \vec{h_j}\|_2^2$ is used to measure the similarity between two nodes.

Regularization Weights Determination The value of $\mathbf{W}_{ij}$ could be calculated according to 2 methods, respectively the Gaussian kernel and K-Nearest Neighbors.

K-Nearest Neighbors method The k-nearest neighbor based weight construction only penalize the distance between a target node e_i and its neighbor nodes.

$$\mathbf{W}_{ij} = \begin{cases} 1, & \text{if } e_i \in N_p\left(e_j\right) \text{ or } e_j \in N_p\left(e_i\right) \\ 0, & \text{otherwise} \end{cases}$$

The neighbors are found based on a distance matrix $\mathbf{D} = \mathbf{H} \cdot \mathbf{H}^T \in \mathbb{R}^{N_e \times N_e}$. The i-th row of $\mathbf{D}$ measures the distance between entity e_i and other nodes. And the k-nearest neighbors are found by sorting the distance vector $\mathbf{D}_i$ of entity e_i.

Gaussian Kernel Method In the Gaussian kernel method, the weight $\mathbf{W}_{ij}$ is calculated based on the distance between each entity in the feature matrix $\mathbf{H}$.

$$\mathbf{W}_{ij} = \exp(-(\vec{h_i} - \vec{h_j})^2/\sigma).$$

On both of the schemes, the hyper-parameters is empirically selected by grid search.

4 Empirical Study

To verify the effectiveness of our proposed methods, we come up with the following research questions (RQs) to advise our experiments:

1. How do the extra regularizers impact the KGC model performance?
2. Where does the model performance change come from? Will the new tasks bring extra information to the model?
3. Will the model performance be influenced by the datasets? Does dataset with massive relation types benefit more from relation-based

	Model	FB15k-237				WN18RR			
		MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@10
Std†	ComplEx	33.68	24.72	36.95	51.29	45.78	43.01	46.64	51.24
	ComplEx-IRID*	**34.27**	**25.33**	**37.73**	**52.26**	**46.44**	43.23	**47.65**	52.60
	ComplEx-PATH*	33.98	25.10	37.44	51.66	46.34	43.06	47.57	52.50
	ComplEx-NodePiece	34.05	25.15	37.50	51.94	46.42	**43.44**	47.55	**52.96**
FRO†	ComplEx-FRO	33.53	24.60	36.97	51.30	46.05	43.12	47.08	51.56
	ComplEx-FRO-IRID	**34.42**	**25.46**	**37.85**	**52.26**	**46.48**	43.27	**47.70**	**52.89**
	ComplEx-FRO-PATH	33.84	25.02	37.21	51.63	46.19	42.94	47.14	52.47
	ComplEx-FRO-NodePiece	34.17	25.32	37.42	51.98	46.42	**43.36**	47.61	52.84
N3†	ComplEx-N3	36.53	27.15	40.02	55.45	47.60	43.30	48.99	56.12
	ComplEx-N3-IRID	**36.79**	27.37	40.39	55.71	47.71	43.31	49.33	56.48
	ComplEx-N3-PATH	/	/	/	/	47.43	43.01	48.89	56.59
	ComplEx-N3-NodePiece	36.77	**27.48**	**40.51**	**55.80**	**47.93**	**43.60**	**49.73**	**56.77**

*IRID - In-range and in-domain representation; PATH - Random Paths Representation;
†Std means training without norm, FRO with the Frobenius norm and N3 with the nuclear-3 norm
grid search n3 $\in \{0.001, 0.01, 0.05, 0.1\}$, feature weight $\in \{0.1, 1, 10, 50, 100\}$

Table 1: General comparison for feature prediction with $k = 500$ and $b = 100$

tasks (relation prediction and predicting IRID feature) compared to datasets with fewer relation types?

Datasets Two benchmark datasets, FB15k-237 [Bollacker *et al.*, 2008] and WN18RR [Dettmers *et al.*, 2018] are selected in the paper. Since we wish to investigate how the number of relation types influence the performance. In FB15k-237, the number of predicates is relatively high compared the one in WN18RR. In WN18RR, on the other hand, most entities are sparsely represented in the training set.

Metrics Since in the knowledge graph completion task, the model is required to answering queries like $\langle s, r, ? \rangle$ and $\langle ?, r, o \rangle$ by ranking all the potential candidates, the evaluation metrics are thus also based on computing the ranks. In this paper we use Hits@k, $k \in \{1, 3, 10\}$ and filtered mean reciprocal rank (MRR)[Akrami *et al.*, 2020].

Models All the experiments are based on factorization-based models, including CP, DistMult and ComplEx. We used the nuclear N3 norm [Lacroix *et al.*, 2018] as a regularizer.

5 Conclusion

References

[Akrami *et al.*, 2020] Farahnaz Akrami, Mohammed Samiul Saeef, Qingheng Zhang, Wei Hu, and Chengkai Li. Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1995–2010, 2020.

[Balazevic *et al.*, 2019] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. Tucker: Tensor factorization for knowledge graph completion. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5184–5193. Association for Computational Linguistics, 2019.

[Bizer *et al.*, 2009] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - A crystallization point for the web of data. *J. Web Semant.*, 7(3):154–165, 2009.

[Bollacker *et al.*, 2008] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250, 2008.

[Bordes *et al.*, 2013] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26, 2013.

[Cai *et al.*, 2010] Deng Cai, Xiaofei He, Jiawei Han, and Thomas S Huang. Graph regularized nonnegative matrix factorization for data representation. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1548–1560, 2010.

[Das *et al.*, 2020] Rajarshi Das, Ameya Godbole, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. A simple approach to case-based reasoning in knowledge bases. *arXiv preprint arXiv:2006.14198*, 2020.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 1811–1818. AAAI Press, 2018.

[Dobrowolska *et al.*, 2021] Agnieszka Dobrowolska, antonio vergari, and Pasquale Minervini. Neural concept formation in knowledge graphs. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[Galkin *et al.*, 2021] Mikhail Galkin, Jiapeng Wu, Etienne Denis, and William L. Hamilton. Nodepiece: Compositional and parameter-efficient representations of large knowledge graphs. *CoRR*, abs/2106.12144, 2021.

[Ghosh *et al.*, 2020] Partha Ghosh, Mehdi S. M. Sajjadi, Antonio Vergari, Michael J. Black, and Bernhard Schölkopf. From variational to deterministic autoencoders. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[Hamilton, 2020] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020.

[Hitchcock, 1927] Frank L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Journal of Mathematics and Physics*, 6(1-4):164–189, 1927.

[Hoffman *et al.*, 2019] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *CoRR*, abs/1908.02729, 2019.

[Kim *et al.*, 2020] Bosung Kim, Taesuk Hong, Youngjoong Ko, and Jungyun Seo. Multi-task learning for knowledge graph completion with pre-trained language models. In Donia Scott, Núria Bel, and Chengqing Zong, editors, *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 1737–1743. International Committee on Computational Linguistics, 2020.

[Lacroix *et al.*, 2018] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR, 2018.

[Moradi *et al.*, 2020] Reza Moradi, Reza Berangi, and Behrouz Minaei. A survey of regularization strategies for deep models. *Artif. Intell. Rev.*, 53(6):3947–3986, 2020.

[Rifai *et al.*, 2011] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 833–840. Omnipress, 2011.

[Suchanek *et al.*, 2007] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706, 2007.

[Thanh-Tung *et al.*, 2019] Hoang Thanh-Tung, Truyen Tran, and Svetha Venkatesh. Improving generalization and stability of generative adversarial networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[Trouillon *et al.*, 2016] Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In Maria-Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR*

Workshop and Conference Proceedings, pages 2071–2080. JMLR.org, 2016.

[Yang *et al.*, 2015] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Yuan and Zhang, 2016] Ming Yuan and Cun-Hui Zhang. On tensor completion via nuclear norm minimization. *Found. Comput. Math.*, 16(4):1031–1068, 2016.

[Zou and Hastie, 2005] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.