A Systematic Analysis of Regularisation Terms for Neural Link Prediction Models

Abstract

Regularisers are instrumental in improving the generalisation accuracy of neural link prediction models. In this paper, we systematically analyse several regularisation methods for factorisation-based neural link predictors and evaluate how they impact the downstream link prediction accuracy. We consider multiple methods for regularising neural link predictors, including norm-based regularisers, gradient penalties, auxiliary training objectives, and manifold regularisation. We conduct extensive experiments on three datasets, namely WN18RR, FB15k-237 and YAGO3-10. In our analysis, we find both gradient penalty and auxiliary training objectives can improve the generalisation properties of neural link predictors when using L2 regularisation, yielding up to a 4.6% increase in MRR, 4.9% in Hits@1, and 8.3% in Hits@10 when using ComplEx. On the other hand, we only observe marginal improvements when using the nuclear-3 norm.

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. Those 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.

Regularisation is commonly required during the training of neural link predictors. Without a proper regulariser, the training process could trivially minimise the loss $\mathcal{L}$ by increasing the norm of the embeddings [Bordes *et al.*, 2013]. In this paper, inspired by the regularisers in latent space representation models [Hoffman *et al.*, 2019; Thanh-Tung *et al.*, 2019], multi-task learning [Chen *et al.*, 2021], and matrix factorisation [Cai *et al.*, 2010], we propose to utilise 4 new regularisers for KGC tasks with the hope of improving the model generalisation. Specifically, the methods include *norm-based regularisers*, *gradient penalties*, *multi-task learning*, and *manifold regularisation*.

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 s and object 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 $\mathcal{S} = \{(s_1, r_1, o_1), \ldots, (s_{|\mathcal{S}|}, r_{|\mathcal{S}|}, o_{|\mathcal{S}|})\} \subseteq \mathcal{G}$. While the queries in the validation and test sets come with the form $\langle s, r, ? \rangle$ or $\langle ?, r, o \rangle$, 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 $o' \in \mathcal{E}$, and finding the one with the highest probability, noted as o^*. The likelihood of $P_\theta(o \mid s, r)$ can be estimated by normalising a parametric score function $\phi_\theta(s, r, o)$:

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

Score function: Neural link predictors can be characterised 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 factorisation-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 [Trouillon *et al.*, 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 $\phi(s, r, o) = \mathcal{Z} \times_1 \mathbf{e}_s \times_2 \mathbf{w}_r \times_3 \mathbf{e}_o$.

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 paper, 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,

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

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

Regularisers Previous studies suggest that regularisation prevents the training process to trivially minimise the loss $\mathcal{L}$ by increasing the norm of the embeddings [Bordes *et al.*, 2013]. Recent work also shows that regularisers have a potential to improve the generalisation of neural link predictors [Chen *et al.*, 2021; Lacroix *et al.*, 2018]. The loss function with a regulariser can be written as:

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

For simplicity, we use $\mathbf{z} \in \mathbb{C}^K$ to denote the embedding vector instead of the conventional notation $\mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o$. Proved in [Lacroix *et al.*, 2018], it is possible to only regularise the embeddings in a batch to obtain weighted regulariser.

To the best of our knowledge, L1 and L2 norm of embeddings were the most frequently used regularisers in previous research. For example, the regularisers 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 started to consider using tensor norm as a regulariser instead of simple embeddings norms. [Lacroix *et al.*, 2018] suggested that the nuclear norm can work as an approximation of the tensor rank and proposed to use nuclear norm as a regulariser. While in the factorisation models that we consider, the nuclear-3 (N3) norm works exactly the same as a L3 norm of each embedding. This is a simple yet efficient method to minimising the rank in tensor factorisation and is proved to be significantly beneficial for training factorisation-based neural link predictors.

However, current methods to regularise neural link predictors are only limited to norm-based regularisers. In this paper, we will consider multiple regularisation methods from different machine learning fields, and make a systematic analysis on their impact on the neural link prediction models compared to existing regularisers.

3 Regularisation term

In this paper, we will propose four novel regularisers, and investigate their impact on the neural link prediction models.

3.1 Norm-based regularisation

L1, L2 and N3 norms are the regularisers that are generally favoured by the community. Inspired by the huge success of elastic net [Zou and Hastie, 2005], our first attempt is to combine different orders of norms and define a new regulariser:

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

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 regularisation could also work on the scheme of neural link predictors. Specifically, we consider applying gradient penalty to the decoder part (score function) since the encoder part is simply a lookup function and not differentiable.

Let the embedding vectors $\mathbf{z} \in \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o$ (with size K) be the input of the score function $y = f(\mathbf{z}) = \phi_\theta(s, r, o)$ (output). A small perturbation applied to the embedding $\mathbf{z}$ can be expressed as $\mathbf{z} + \boldsymbol{\epsilon}$ and we are interested in minimising the effect of such perturbation on the model output. According to Taylor expansion, the corresponding function output will be approximated as:

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

If we neglect the second order infinitesimal $O\left(\epsilon^2\right)$, minimising the output drift due to input perturbations, namely $f(\mathbf{z}+\boldsymbol{\epsilon}) - f(\mathbf{z})$, is equivalent to minimising the term $\sum_{i=1}^{K} \epsilon_i \cdot \frac{\partial f}{\partial \mathbf{z}_i}(\mathbf{z})$. In other words, the impact of the input perturbation $\boldsymbol{\epsilon}$ on the model output is governed by the so-called Jacobian function $\mathbf{J}(\mathbf{z}) = (\mathbf{J}_1(\mathbf{z}), \ldots \mathbf{J}_K(\mathbf{z}))$, with

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

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

Thus, minimising $\|\mathbf{J}(\mathbf{z})\|_p$ would work as a regulariser to make the model robust to input noise. By introducing the l_2 gradient penalty to our model, we can now form the new loss function as

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

We calculate the Jacobian matrix w.r.t score function as in Equation (2) instead of multi-class output of the models to reduce 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

Graph representation learning algorithms [Hamilton, 2020], e.g. 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 entity embeddings by both improving the generalisation and encoding geometrical information of the KG to the embeddings. 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 auxiliary tasks, which can be viewed as a regulariser.

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 relation types could be highly correlated to its subject or object, we can aggregate all the relation types that are directly connected to a target node to construct a feature for it. 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:

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

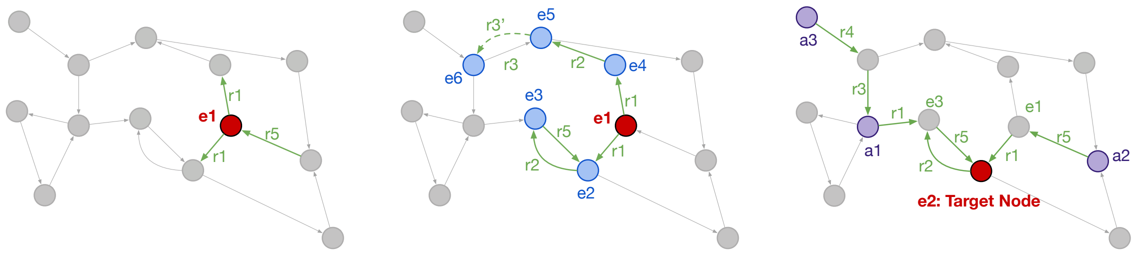
$$\text{in-domain}(e, r) = \begin{cases} 1 & \forall s, \text{ if } \exists (s, r, e) \in \mathcal{S} \\ 0 & \text{otherwise} \end{cases}$$

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

Random Paths Feature Representation The idea of Random Paths feature representation is inspired by the path sampling method in [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, i.e. $p = (e_1, r_1, e_2, \ldots e_{K-1}, r_{K-1}, e_K)$, 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 the target node and find out the entities and relations that are closely linked to it. In NodePiece representation we will consider the entities, 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 $|\mathcal{E}| \times 2|\mathcal{R}|$. 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. An example is in Figure 1b.

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



(a) Considering the target node e_1 and assuming 5 relations types exist, $r_1, \ldots, r_5$, r_1 is in the range of e_1 and r_5 is in the domain of e_1. The feature then is $[1, 0, 0, 0, 0, 0, 0, 0, 0, 1]$.

(b) Assuming 5 relations types exist, $r_1, \ldots, r_5$ and considering the inverse relation r_i', we sampled 2 length-3 paths from e_1, $p_1 = (e_1, r_1, e_2, r_2, e_3, r_5, e_2)$ and $p_2 = (e_1, r_1, e_4, r_2, e_5, r_3', e_6)$. The feature is $[1, 1, 0, 0, 1, 0, 0, 1, 0, 0]$.

(c) e_2 is the target node. If 3 anchor nodes a_1, a_2, a_3 are selected, and 2-nearest anchors are a_1 and a_2, the path from a_1, a_2 to e_2 are $p_1 = (a_1, r_1, e_3, r_5, e_2)$ and $p_2 = (a_2, r_5, e_1, r_1, e_2)$. The constructured feature is $[1, 1, 0, 1, 0, 0, 0, 1]$

Figure 1: Examples of three feature construction methods

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 $\mathcal{G} = (\mathcal{E}, \mathcal{R})$, 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. For instance, assuming that we successfully select $|A|$ nodes as anchors, each anchor node a_j has the shortest path p directing to the target node v_i, which records $|A|$ paths. We keep the k nearest anchors by measuring the distance between a_j's and v_i. 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 selected either randomly or by importance measurement. For our purpose, centrality and 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 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_i, 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,

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

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

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

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

The auxiliary task design In this paper, we only consider constructing features for entities, which can be denoted as $\mathbf{H} = (\vec{h}_1, \ldots \vec{h}_i, \ldots, \vec{h}_{|\mathcal{E}|})^T \in \mathbb{R}^{|\mathcal{E}| \times F}$, where $|\mathcal{E}|$ is the number of entities and F is the size 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. With a new training loss L' for the auxiliary training objective, the overall loss now becomes:

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

3.4 Manifold Regularisation

Manifold regularisation, a method first used in matrix factorisation [Cai *et al.*, 2010], utilises the geometric shape of a dataset to constrain the embeddings learned. For factorisation-based neural link predictors, the intuition is that if two data entities e_i and e_j are similar by some kind of measure, their embeddings $\mathbf{e}_i$ and $\mathbf{e}_j$ should also be

Dataset	regularisers	CP			DistMult			ComplEx		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
FB15k-237	Nulcear-3 norm	34.83	25.77	52.88	35.84	26.53	54.78	36.67	27.28	55.78
	Norm-based	34.85	25.78	52.97	36.06	26.79	54.83	36.81	27.46	55.90
	GP+N3	35.01	**26.05**	52.94	36.09	26.81	**54.92**	**36.90**	**27.54**	**55.90**
	IRID+N3	35.00	25.97	53.03	**36.14**	26.86	54.91	36.79	27.36	55.80
	Path+N3	35.01	26.03	53.01	36.00	26.71	54.90	36.75	27.27	55.74
	NodePiece+N3	**35.11**	26.04	**53.18**	36.12	**26.88**	54.91	36.84	27.51	55.83
	Manifold+N3	34.86	25.79	52.87	36.08	26.70	54.81	36.16	26.76	55.27
WN18RR	Nulcear-3 norm	11.40	7.43	19.25	45.07	41.02	53.11	48.60	44.14	**57.48**
	Norm-based	11.58	7.71	19.51	45.08	40.94	53.44	**48.70**	**44.54**	56.77
	GP+N3	11.69	7.58	20.33	**45.27**	41.23	53.60	48.35	44.13	56.78
	IRID+N3	11.89	**7.84**	20.35	45.22	**41.48**	53.20	48.71	44.73	57.08
	Path+N3	11.52	7.53	19.82	45.19	41.18	53.53	48.60	44.50	57.07
	NodePiece+N3	**12.12**	7.83	**21.03**	45.20	40.84	**53.74**	48.62	44.40	56.77

* GP - Gradient Penalty; IRID - In-range and in-domain feature; Path - Random Paths features; Embedding size = 2000

Table 1: Experiments results when the regularisers are applied together with N3 norm

close. The method aims to construct a manifold to If the distance between embeddings is measured by Euclidean norm, we can get the regularisation as $\mathbf{R} = \sum_{i,j=1}^{|\mathcal{E}|} \|\mathbf{e}_i - \mathbf{e}_j\|_2^2 \mathbf{W}_{ij}$.

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

$$\mathcal{L} = \sum_{s,r,o \in S} \ell_{s,r,o} + \lambda \sum_{i,j=1}^{|\mathcal{E}|} \|\mathbf{e}_i - \mathbf{e}_j\|_2^2 \mathbf{W}_{ij}$$

Similarity Matrix Construction Manifold regularisation 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 Section 3.3, and use the feature vectors as a representation for entities. Specifically, the distance matrix is calculated by measuring the distance between feature vectors, where $D_{ij} = (\vec{h}_i - \vec{h}_j)^2$.

Regularisation Weights Determination The value of $\mathbf{W}_{ij}$ could be calculated according to 2 methods, respectively the k-Nearest Neighbors (KNN) and the Gaussian kernel. The KNN weight construction only penalises the distance between a target node e_i and its neighbor nodes, where $\mathbf{W}_{ij} = 1$ if e_j is one of the k-nearest neighborhoods of e_i. The neighbors are found based on the distance matrix $\mathbf{D}$. In the Gaussian kernel method, the weight $\mathbf{W}_{ij}$ is calculated based on the distance between each entity in the feature matrix $\mathbf{H}$. $\mathbf{W}_{ij} = \exp(-(\vec{h}_i - \vec{h}_j)^2/\sigma)$, with σ being a shape parameter.

4 Empirical Study

To verify the effectiveness of our methods, the experiments are designed with the following settings:

Datasets Two benchmark datasets, FB15k-237 [Bollacker *et al.*, 2008] and WN18RR [Dettmers *et al.*, 2018] are selected in the paper.

Metrics We use Hits@k, $k \in \{1, 10\}$ and filtered Mean Reciprocal Rank (MRR) as the evaluation metrics to compare the model performance.

Models Experiments are conducted with models based on tensor factorisation, including CP, DistMult and ComplEx. We used the nuclear N3 and the L2 norm [Lacroix *et al.*, 2018] as a regulariser. The models are trained only with standard triples without reciprocal reformalisation.

4.1 The impact of regularisers

As shown in Table 1, we find nuclear norm dominates the performance in the experiments of norm-based regularisers. The best performance is observed when nuclear norm is applied solely and l_1, l_2 norms can only bring a small change.

Except for the norm-based regularisers, we compare the performance of training with and without the regularisers to answer how do the extra regularisers impact the KGC models. The extra regularisers are trained respectively with N3 or L2 norm. To find the best hyperparameter combinations, grid search was done with N3 $\in \{0, 10^{-3}, 10^{-2}, 0.05, 0.1, 0.5\}$, L2 $\in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$, regulariser weight $\in \{0, 0.01, 0.1, 1, 10, 50, 100, 1000\}$.

Experiments with N3 norm Also as illustrated in Table 1, we observe that our regularisers can only bring a marginal improvement to the models when trained together with N3 norm. Regularisers

Dataset	regularisers	CP			DistMult			ComplEx		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
FB15k-237	$L2$-norm	33.60	25.09	50.37	34.71	25.80	52.55	34.95	25.97	53.03
	GP+L2	**34.44**	25.53	**52.11**	**35.78**	26.54	**54.56**	**36.49**	**27.23**	**55.13**
	IRID+L2	33.98	25.29	51.27	35.58	**26.61**	53.66	36.01	26.95	54.41
	NodePiece+L2	34.40	**25.63**	51.57	35.29	26.37	53.37	36.11	27.14	54.18
WN18RR	$L2$-norm	8.39	6.06	12.98	44.32	41.30	50.29	45.49	42.53	51.08
	GP+L2	10.10	7.31	15.00	44.34	41.35	50.77	**47.27**	**43.34**	**55.34**
	IRID+L2	10.31	7.68	14.91	**44.48**	41.50	50.14	45.99	42.75	51.83
	NodePiece+L2	**11.27**	**7.70**	**18.00**	44.46	**41.50**	**50.77**	45.99	42.66	52.44

* GP - Gradient Penalty; IRID - In-range and in-domain feature; Embedding size = 2000

Table 2: Experiments results when the regularisers are applied together with L2 norm

like multi-task learning with random paths representation and manifold regularisation cannot even outperform baseline models. The results are disappointing at the first glimpse. But as we further look into the cases when the models are trained with smaller N3 weights, they do benefit from some of our designed regularisers.

Experiments with L2 norm We further investigate in the scenarios when the regularisers are applied to the models with L2 norm. We only conduct the experiments on regularisers found effective in previous steps and the results are illustrated in Table 2. The experiment results suggest that gradient penalty can bring a comparative generalisation improvement as N3 norm. While a consistent improvement is found on all the factorisation-based models, ComplEx benefits most from gradient penatly, with increases up to a 4.6% in MRR, 4.9% in Hits@1, and 8.3% in Hits@10. Also the models gain improvements from the auxiliary training objectives with IRID features and NodePiece feature, even though the values are not as significant as gradient penalty.

4.2 Data Efficiency Analysis

While the mechanism of gradient penalty is clear, we also hope to provide an explanation for the improvement brought by auxiliary tasks. We manage to test the impact of the feature prediction tasks on neural link predictors when data points are insufficient. We test in the scenarios when 5%, 10%, 20% and 50% data points from the whole dataset are accessed for training by uniformly random sampling from the original dataset. We construct the features respectively with the subset or with the whole dataset and train the model. Figure 2 gives an example when NodePiece feature is used.

For all the experiments with insufficient data, training with auxiliary tasks shows a significant improvement on the model performance, and using the feature constructed from the whole dataset brings even better improvement. This result proves that the feature vector contains extra information

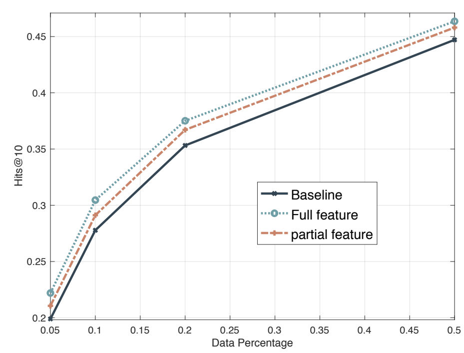


Figure 2: Data efficiency analysis for NodePiece feature evaluated on FB15k-237, with $k = 2000$ and N3 $= 0.1$

extracted from the knowledge triples, and the prediction task would potentially encode the information into the embeddings while training.

The Hits@10 increases are respectively 5.8%, 4.8%, 3.9% and 2.4% for the experiments on the subset of 5%, 10%, 20% and 50% data points. It is noted that the generalisation improvement is more remarkable when fewer data points are accessed. Our assumption is that the auxiliary task could bring the models out of the predicament of overfitting. Because if the data points are not sufficient, the model would easily overfit to the data.

5 Conclusion

Our work suggests that nuclear-3 norm is the most effective regulariser so far and the proposed regularisers cannot further boost the performance of the factorisation-based models when nuclear-3 norm is applied. However, we find that gradient penalty regularisation could bring a similar improvement to the models as nuclear norm when trained together with L2 regularisation. Multi-task learning regularisers would also benefit the training of neural link predictors, especially when the data points are insufficient. Our future work will attempt to give an explanation about the mechanism of N3 norm and why other regularisers do not interact well with it.

References

[Balazevic *et al.*, 2019] Ivana Balazevic, Carl Allen, and Timothy M. Hospedales. Tucker: Tensor factorization for knowledge graph completion. In *EMNLP/IJCNLP (1)*, 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.

[Chen *et al.*, 2021] Yihong Chen, Pasquale Minervini, Sebastian Riedel, and Pontus Stenetorp. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

[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. In *2rd Conference on Automated Knowledge Base Construction*, 2020.

[Dettmers *et al.*, 2018] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, 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.

[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.

[Lacroix *et al.*, 2018] Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. Canonical tensor decomposition for knowledge base completion. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 2869–2878. PMLR, 2018.

[Rifai *et al.*, 2011] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 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 *ICML*, 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 *ICLR (Poster)*, 2015.

[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.