A Systematic Analysis of Regularisation Terms for Neural Link Prediction Models

Anonymous Author(s)
Anonymous Affiliation
Anonymous Email

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 trained together with 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. The auxiliary training objectives are effective when the data is insufficient or the model is 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 [1], DBpedia [2] and YAGO [3], more entities with few or zero relations were added to the KG. Those entities resulted in a highly incomplete structure of the KG. Therefore, the research on Knowledge Graph Completion (KGC) was proposed to explore the implicit relationships among entities or relations and construct a complete KG for large-scale real-world applications by recovering the missing facts.

Among all the research directions of KGC, the neural link predictor, a kind of Knowledge Graph Embedding (KGE) Model, has become more and more popular recently. This method focuses on learning 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 to improve the model generalisation. In this paper, inspired by the regularisers in latent space representation models [4], multi-task learning [5], and matrix factorisation [6], we consider 4 new regularisers for KGC tasks with the hope of improving the model generalisation, including *norm-based regularisers*, *gradient penalty*, *multi-task learning*, and *manifold regularisation*. We find both gradient penalty and auxiliary training objectives can improve the generalisation properties when using L2 regularisation, and training with auxiliary tasks is especially helpful when the model has high complexity and data is insufficient. Based on our experimental results, we then conclude nuclear norm and gradient penalty are the most effective regularisers.

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

Submitted to the First Learning on Graphs Conference (LoG 2022). Do not distribute.

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 $\mathcal{S} = \left\{(s_1, r_1, o_1), \ldots, \left(s_{|\mathcal{S}|}, r_{|\mathcal{S}|}, o_{|\mathcal{S}|}\right)\right\} \subseteq \mathcal{G}$. While the queries in the validation and test sets come with the form $\langle s, r, ?\rangle$ 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 $\phi_\theta(s, r, o)$:

$$P_\theta(o \mid s, r) = \frac{\exp(\phi_\theta(s, r, o))}{\sum_{o'} \exp(\phi_\theta(s, r, o'))} \tag{1}$$

Scoring 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 [7] 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 [8] uses two distinct representations for an entity depending on whether it is used as a subject or object, and uses the same tensor inner product to calculate the score function. ComplEx [9] 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 [10] 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. Motivated by the solid results in [11], we will follow the convention 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,

$$\mathcal{L} = -\sum_{\langle s,r,o \rangle \in \mathcal{S}} \left[\log P_\theta(s \mid r, o) + \log P_\theta(o \mid s, r)\right] \tag{2}$$

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 [12]. Recent work also shows that regularisers have the potential to improve the generalisation of neural link predictors [5, 11]. The loss function with a regulariser can be written as:

$$\begin{aligned} \mathcal{L} &= \sum_{s,r,o \in S} \ell_{s,r,o} + \lambda \mathbf{R}(\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r) \\ &= \sum_{s,r,o \in \mathcal{S}} \left(\ell_{s,r,o} + \sum_{\mathbf{z} \in \{\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r\}} \lambda \mathbf{R}(\mathbf{z}) \right) \end{aligned} \tag{3}$$

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 [11], it is possible to only regularise the embeddings in a batch to obtain a weighted regulariser. But this trick is only practical for factorisation-based models, so in this work, we will mainly focus on factorisation-based neural link prediction models.

To the best of our knowledge, previous works mostly rely on L1 and L2 regularisers. For example, the regularisers used by [12] and [9] 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 embedding norms. [11] suggested that the nuclear norm can work as an approximation of the tensor rank and proposed to use the nuclear norm as a regulariser. While in the factorisation models that we consider, the nuclear 3-(N3) norm works exactly the same as the L3 norm of each embedding. Beyond that, designing auxiliary tasks for KGC models is also considered a useful regulariser. Among them, [5] adds relation prediction as a new training objective, DURA [13] constructs a dual distance-based KGC model for each tensor factorisation model and jointly trains them to get better embeddings. These regularisers have been proven to be beneficial and efficient for training factorisation-based neural link predictors.

In this paper, we will consider multiple regularisation methods from different machine learning fields that have never been tested in KGC tasks, and make a systematic analysis of their impact on the neural link prediction models compared to existing regularisers.

3 Regularisation terms

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 elastic net[14], our first attempt is to combine different orders of norms and define a new regulariser:

$$\mathbf{R}(\mathbf{z}) = \lambda_1 \|\mathbf{z}\|_1 + \lambda_2 \|\mathbf{z}\|_2 + \lambda_3 \|\mathbf{z}\|_3 \tag{4}$$

3.2 Gradient Penalty

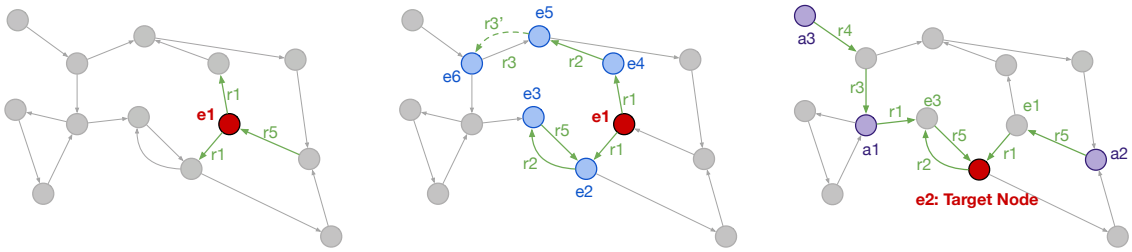
Gradient penalty has been widely applied to latent space models, e.g. Generative Adversarial Neural Networks (GANs) [15] and Variational Auto-Encoders (VAEs) [16]. Neural link predictors are also latent space models with the encoder being embedding lookup functions and the decoder being score functions [17]. 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:

$$f(\mathbf{z} + \boldsymbol{\epsilon}) = f(\mathbf{z}) + \sum_{i=1}^{K} \epsilon_i \cdot \frac{\partial f}{\partial \mathbf{z}_i}(\mathbf{z}) + O\left(\boldsymbol{\epsilon}^2\right). \tag{5}$$

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 $\mathbf{J}(\mathbf{z}) = (\mathbf{J}_1(\mathbf{z}), \dots \mathbf{J}_K(\mathbf{z}))$, with the output function being $f(\mathbf{z}) = \phi_\theta(s, r, o)$, we get

$$\mathbf{J}_i(\mathbf{z}) \equiv \frac{\partial f}{\partial \mathbf{z}_i}(\mathbf{z}) = \frac{\partial \phi_\theta(s, r, o)}{\partial \mathbf{z}_i}, \quad i \in \{1, \dots, K\}, \tag{6}$$



(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 constructed feature is $[1, 1, 0, 1, 0, 0, 0, 1]$

Figure 1: Examples of three feature construction methods

Thus, minimising $\|\mathbf{J}(\mathbf{z})\|_p$ would work as a regulariser to make the model robust to input noise. By introducing the L2 gradient penalty to our model, we can now form the new loss function as

$$\mathcal{L} = \sum_{s,r,o \in \mathcal{S}} \left(\ell_{s,r,o} + \sum_{\mathbf{z} \in \{\mathbf{e}_s, \mathbf{e}_o, \mathbf{w}_r\}} \lambda \|\mathbf{J}(\mathbf{z})\|_2 \right) \tag{7}$$

We calculate the Jacobian matrix w.r.t score function as in Equation (6) instead of multi-class output of the models to reduce the tensor size to be $b \times K$. This way, the computational resources are saved and a similar effect is obtained. To better understand the mechanism, we derive the analytical form of Gradient Penalty on factorisation models in Appendix A.5 and make a comparison with norm-based methods.

3.3 Multi-task Learning

Graph representation learning algorithms [17], 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 structure 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 [18, 19] 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 utilises the relation types to construct the features. Considering that the relation types could be highly correlated to their 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 using Random Paths feature representations is inspired by the path sampling method in [20]. 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. $p = (e_1, r_1, e_2, \dots e_{K-1}, r_{K-1}, e_K)$, 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 sub-graph 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 and the selection of hyperparameters is discussed in Appendix A.1.

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 [19] 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 the anchor nodes a_j 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 Personalised PageRank (PPR) [17] on nodes would be combined to determine the anchor nodes. 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 $\vec{h}^r$. 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 $p_j = (e_1, r_1, e_2, \dots e_{K-1}, r_{K-1}, e_K)$. We again record the relation that appears in the path to form a relation feature.

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

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 $\mathbf{H} \in \mathbb{R}^{|\mathcal{E}| \times (|A|+|\mathcal{R}|)}$. Further discussion of hyperparameter setting can be found in Appendix A.1

The auxiliary task design. In this paper, we only consider constructing features for entities, which can be denoted as $\mathbf{H} = (\vec{h}_1, \dots \vec{h}_i, \dots, \vec{h}_{|\mathcal{E}|})^T \in \mathbb{R}^{|\mathcal{E}| \times F}$, 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

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

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:

$$\mathcal{L} = \sum_{s,r,o \in \mathcal{S}} \ell_{s,r,o} + \lambda L' \left(g \left(\mathbf{E} \right), \mathbf{H} \right) \tag{8}$$

3.4 Manifold Regularisation

Manifold regularisation, a method first used in matrix factorisation [6], 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 close. 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:

$$\mathcal{L} = \sum_{s,r,o \in S} \ell_{s,r,o} + \lambda \sum_{i,j=1}^{|\mathcal{E}|} \|\mathbf{e}_i - \mathbf{e}_j\|_2^2 \mathbf{W}_{ij} \tag{9}$$

Similarity Matrix Construction. Manifold regularisation needs to access a distance matrix $\mathbf{D}$ to retrieve similar entities in the KG. To construct the $\mathbf{D}$ and calculate the weight $\mathbf{W}_{ij}$, we 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 similarity 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 Neighbours (KNN) and the Gaussian kernel. The KNN weight construction only penalises the distance between a target node e_i and its neighbour nodes, where $\mathbf{W}_{ij} = 1$ if e_j is one of the k-nearest neighbourhoods of e_i. The neighbours 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 proposed methods, we come up with the following research questions (RQs) to advise our experiments:

RQ1: How do the extra regularisers impact the KGC model performance across different datasets?

RQ2: How are the proposed regularisers compared with existing regularisers?

RQ3: Where does the model performance change come from? Will the new tasks bring extra information to the model? or does it simply improve the model generalisation?

To answer **RQ1, 2**, extensive experiments on various datasets and models are conducted. The settings are:

Datasets: Three benchmark datasets, FB15k-237 [1], WN18RR [21], and Yago3-10 [22] are selected in the paper. Experiments on smaller datasets like Kinship, Nations and UMLS are discussed in Appendix A.3.

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

Benchmark regularisers: To compare our regularisers with other existing methods, we implemented norm-based regularisers[11], Relation Prediction[5] and DURA[13] as comparative methods.

Models: Experiments are conducted with models based on tensor factorisation, including CP, DistMult and ComplEx (though DURA does not support DistMult). We used the nuclear N3 and the L2 norm [11] as

Dataset	Regularisers	CP			DistMult			ComplEx		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
FB15k-237	N3	34.83	25.77	52.88	35.84	26.53	54.78	36.67	27.28	55.78
	L1+L2+N3	34.85	25.78	52.97	36.06	26.79	54.83	36.81	27.46	55.90
	RP+N3	35.32	26.30	53.27	36.48	27.09	55.54	**37.17**	27.70	**56.17**
	GP+N3	35.01	**26.05**	52.94	36.09	26.81	**54.92**	37.02	**27.76**	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
	DURA+N3	35.05	26.11	53.03	/	/	/	36.39	27.01	55.27
WN18RR	N3	11.40	7.43	19.25	45.07	41.02	53.11	48.60	44.14	**57.48**
	L1+L2+N3	11.58	7.71	19.51	45.08	40.94	53.44	48.70	44.54	56.77
	RP+N3	12.04	7.83	19.77	45.32	41.33	53.70	**48.91**	**44.99**	56.82
	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
	Manifold+N3	12.19	8.59	19.00	**45.32**	41.30	53.70	38.10	35.12	43.34
	DURA+N3	10.66	7.48	17.89	/	/	/	48.61	44.72	56.72
Yago3-10	N3	56.17	49.45	68.50	57.02	50.27	70.04	58.12	50.79	70.65
	L1+L2+N3	56.33	49.63	**69.03**	57.27	50.30	70.40	58.16	50.97	71.45
	RP+N3	56.16	49.68	67.89	**57.41**	50.26	70.44	58.37	51.28	71.43
	GP+N3	56.56	49.75	68.90	57.36	**50.40**	70.04	58.50	51.39	**71.69**
	IRID+N3	**56.61**	49.82	68.92	57.38	50.23	**70.56**	58.21	51.30	71.03
	Path+N3	56.60	**50.07**	68.82	57.38	50.27	70.56	58.13	51.12	71.00
	NodePiece+N3	56.56	49.86	68.92	57.34	50.33	70.44	**58.62**	**51.61**	71.53
	Manifold+N3	/	/		/	/		/	/	
	DURA+N3	55.44	48.36	68.16	/	/	/	58.03	50.65	71.47

* 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

a regulariser to prevent the training process to trivially minimise the loss $\mathcal{L}$ by increasing the norm of the embeddings [12]. We also conduct experiments when the regularisers are applied in a standalone mode, and the results can be found in Appendix A.2. The models are trained with standard triples without reciprocals.

About **RQ3**, we are especially interested in providing an explanation for the improvement brought by auxiliary tasks, as the mechanism of gradient penalty and manifold regularisation is theoretically proved in the previous works[6, 15]. Thus, we designed two follow-up experiments, specifically called embedding size analysis and data efficiency analysis to solve **RQ3**. Embedding size analysis helps us to understand the impact of auxiliary tasks on models with different complexity. And data efficiency analysis trains models with a different number of data points to test the impact of extra information.

4.1 The impact of regularisers

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. The hyperparameter configuration is reported in Appendix A.1

General Observations. Our experiments suggest that all regularisers can only bring marginal improvements when combined with the nuclear norm. This is observed both in our proposed regularisers and in the

Dataset	Regularisers	CP			DistMult			ComplEx		
		MRR	H@1	H@10	MRR	H@1	H@10	MRR	H@1	H@10
FB15k-237	L2	33.60	25.09	50.37	34.71	25.80	52.55	34.95	25.97	53.03
	RP+L2	33.76	52.32	50.70	35.05	26.20	52.75	35.71	26.82	53.76
	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
	DURA+L2	32.74	24.15	49.71	/	/	/	34.37	26.50	52.60
WN18RR	L2	8.39	6.06	12.98	44.32	41.30	50.29	45.49	42.53	51.08
	RP+L2	10.06	7.61	15.47	44.29	41.24	50.34	45.99	43.08	51.01
	GP+L2	10.10	7.31	15.00	44.34	41.35	50.77	**47.27**	**43.34**	**55.34**
	IRID+L2	10.61	8.04	15.98	**44.62**	41.55	50.14	46.13	42.75	52.71
	NodePiece+L2	**11.27**	**7.70**	**18.00**	44.46	**41.50**	**50.77**	45.99	42.66	52.44
	DURA+L2	10.26	7.50	15.54	/	/	/	46.14	42.86	52.26
Yago3-10	L2	55.64	49.14	67.43	56.83	49.74	69.94	57.85	50.80	70.79
	RP+L2	56.16	**49.69**	67.89	56.86	49.93	70.20	58.09	51.24	70.71
	GP+L2	**56.24**	49.64	**68.17**	**57.17**	**50.23**	**70.17**	58.20	51.28	**71.15**
	IRID+L2	56.10	49.65	68.13	57.00	50.10	70.12	**58.21**	**51.30**	71.03
	NodePiece+L2	55.59	49.37	68.06	56.88	49.94	69.78	58.13	51.12	71.00
	DURA+L2	55.53	49.15	67.04	/	/	/	58.19	50.99	70.99

* 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

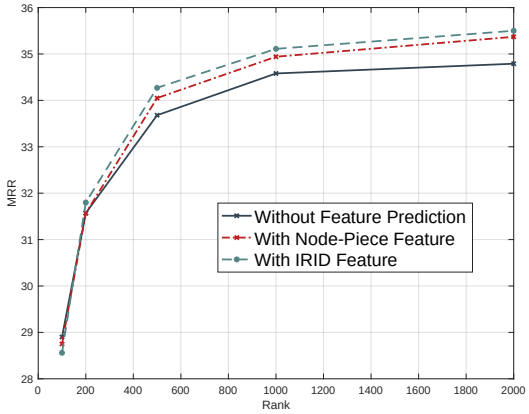
regularisers that are claimed helpful in previous research, e.g. DURA and relation prediction. However, when the regularisers are trained in the standalone mode or with L2 norms, a significant improvement can be observed in all the regularisers except the manifold regularisation and auxiliary training objective designed with random paths representation. The conclusion is supported not only by the accuracy test in Tables 1, 2 and 5 but also by the regulariser weights reported in Table 3.

Experiments with N3 norm. From our experiments, the standard deviation is around 0.3. Thus, given the small difference between model performance with different regularisers, we can only claim that most of the regularisers fail to boost the model performance when the nuclear norm is applied. The hyperparameters configuration in Table 3 can also help us to make the same conclusion. Usually, the smaller the regulariser weights are, the less influential the regularisers are to the model. In Table 3, regularisers, weights with the nuclear norm is much smaller than the others, and in some extreme scenario, the weights decay to 0, which means auxiliary regularisers do not even bring an improvement.

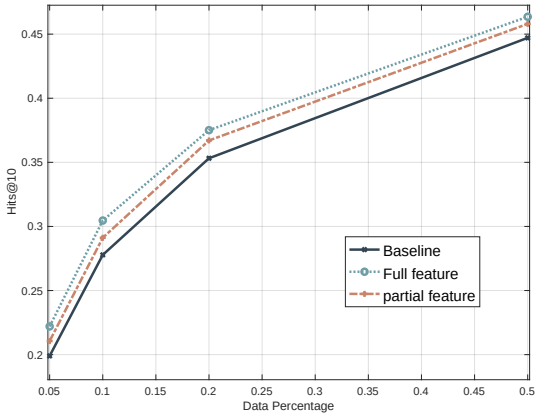
Experiments with L2 norm. However, the impact of regularisers becomes significant in the L2-norm experiments. The experiment results suggest that gradient penalty can improve comparative generalisation as the N3 norm. While a consistent improvement is found in all the factorisation-based models, ComplEx benefits most from gradient penalty, with increases up to 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 the gradient penalty.

4.2 Embedding Size analysis

The tensor factorisation-based neural link predictors are known to suffer from the problem of scalability when using larger embedding sizes [5] and the choice of embedding sizes would largely impact the generalisation of these models. Thus, in order to test the performance of our regularisers on models with different complexity, we consider the embedding sizes $K \in \{100, 200, 500, 1000, 2000\}$ and compared the MRR



(a) Embedding size analysis for IRID and NodePiece features, with $k = 2000$ and L2 $= 0.001$



(b) Data efficiency analysis for NodePiece feature evaluated on FB15k-237, with $k = 2000$ and N3 $= 0.1$

among the scenarios with/without auxiliary tasks. The results on FB15k-237 are illustrated in Figure 2a, and the numerical results can be found in Appendix A.4.

Interestingly, we clearly find that auxiliary tasks grant a better performance on the more complex models with larger embedding sizes. This result suggests that multi-task learning methods truly improve the generalisation, and help the models with higher complexity to reach their performance ceiling.

4.3 Data Efficiency Analysis

We then manage to test the impact of the auxiliary 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 original dataset are accessed for training by uniformly random sampling. To test whether the auxiliary training objective can encode extra information into the model, we construct the features respectively with the subset or with the whole dataset and train the model, as the feature vector constructed from the whole dataset contains the information that the training data points do not have.

Figure 2b gives an example of when NodePiece features are used. For all the experiments with insufficient data, training with auxiliary tasks shows a significant improvement in the model performance, and using the feature constructed from the whole dataset brings even better improvement. 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. This result proves that the auxiliary task could bring the models out of the predicament of overfitting. In the meantime, the experiments also suggest that the feature vector contains extra information extracted from the knowledge triples, and the prediction task would potentially encode the information into the embeddings while training. This is valuable as we can utilise this feature as a method to avoid over-complicated training processes, or apply it in low-resource training tasks.

5 Conclusion

Our work suggests that the nuclear 3-norm is a very effective regulariser, and the other considered regularisers did not significantly improve the accuracy of the factorisation-based models when N3 norm is applied. However, we find that gradient penalty regularisation could bring a similar improvement to the models as the 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 and the model has high complexity. Our future work will attempt to give an explanation of the mechanism of N3 norm and further investigate how to utilise the auxiliary training objectives in low-resource training tasks.

References

[1] 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. 1, 6

[2] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - A crystallization point for the web of data. *J. Web Semant.*, 7(3): 154–165, 2009. 1

[3] 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. 1

[4] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust learning with jacobian regularization. *CoRR*, abs/1908.02729, 2019. URL http://arxiv.org/abs/1908.02729. 1

[5] 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. URL https://openreview.net/forum?id=Qa3uS3H7-Le. 1, 2, 3, 6, 8, 13

[6] 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. 1, 6, 7, 13

[7] 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. 2

[8] 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. doi: https://doi.org/10.1002/sapm192761164. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/sapm192761164. 2

[9] 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. 2, 3

[10] 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. 2

[11] 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. 2, 3, 6

[12] 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. 2, 3, 7

[13] Zhanqiu Zhang, Jianyu Cai, and Jie Wang. Duality-induced regularizer for tensor factorization based knowledge graph completion. In *NeurIPS*, 2020. 3, 6

[14] 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. doi: https://doi.org/10.1111/j.1467-9868.2005.00503.x. URL https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9868.2005.00503.x. 3

[15] 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. URL https://openreview.net/forum?id=ByxPYjC5KQ. 3, 7

[16] 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. 3

[17] William L Hamilton. Graph representation learning. *Synthesis Lectures on Artifical Intelligence and Machine Learning*, 14(3):1–159, 2020. 3, 4, 5

[18] Agnieszka Dobrowolska, Antonio Vergari, and Pasquale Minervini. Neural concept formation in knowledge graphs. In *3rd Conference on Automated Knowledge Base Construction*, 2021. URL https://openreview.net/forum?id=V61-62OS4mZ. 4

[19] 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. 4, 5, 12

[20] 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. 5

[21] Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *AAAI*, pages 1811–1818. AAAI Press, 2018. 6

[22] Farzaneh Mahdisoltani, Joanna Biega, and Fabian M. Suchanek. YAGO3: A knowledge base from multilingual wikipedias. In *CIDR*. www.cidrdb.org, 2015. 6

Dataset	Regularisers	CP			DistMult			ComplEx		
		W/O	F2	N3	W/O	F2	N3	W/O	F2	N3
FB15k-237	RP	0.01	0.1	0.5	0.5	0.1	0.5	0.5	0.5	0.5
	GP	10^3	10^2	10	10^3	10^3	10	10^3	10^3	10^2
	IRID	10	10	0.1	10	10	10	50	10	10
	PATH	/	/	0.01	/	/	0.1	/	/	0.01
	NodePiece	10	10	1	10	10	1	5	10	0.1
	DURA	0.005	0.01	0	/	/	/	0	0.005	0
WN18RR	RP	0.05	0	0.1	0	0.05	0.05	0.1	0.005	0.05
	GP	10^3	10^3	10^3	10^2	10^2	0.01	10^3	10^3	500
	IRID	10	1	0.1	0.1	1	1	0	1	1
	Path	/	/	0.01	/	/	0.01	/	/	0
	NodePiece	10^2	10	10	1	1	0.1	10	10	0.1
	DURA	0.01	0.05	0	/	/	/	0.01	0.01	0
Yago3-10	RP	1	1	0	0.5	0.5	0	0.5	1	0.1
	GP	10	10^2	10	10	10^2	10^2	10	10	1
	IRID	10	10	1	0.1	10	10	10	1	10
	Path	10	10	0	1	1	0.01	1	1	0
	NodePiece	1	10	1	1	1	1	10	0.1	1
	DURA	0	0.01	0	/	/	/	0.005	0.005	/

* GP - Gradient Penalty; IRID - In-range and in-domain feature;

Table 3: Best hyperparameter configuration: the regulariser weights

A Appendix

A.1 Best hyper-parameter configuration

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 $\in \{0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$, regulariser weight $\in \{0, 0.01, 0.1, 0.5, 1, 10, 50, 100, 1000\}$. The regulariser weights that we found most benefit the model are reported in Table 3. For the regularisers that we do not find having a positive impact on the model, "0" is put in the cell.

Besides the common regularisers weight λ, for Multi-task learning regularisers and manifold regularisation, we have the following hyperparameter setting.

Hyperparameter for Random Path feature prediction. The number of random paths sampled n is a hyper-parameter to be tuned, in our experiment we test a range of $n \in [5, 10, 50, 100, 1000]$, and find the performance becomes stable with $n \geq 50$. In our empirical study, the random paths feature representation does not bring significant improvement to the KGC models. This is reflected by either the marginal performance improvement brought by this regulariser described in Tables 1, 2 and 5, or the small regulariser weights after grid search in Table 3.

Hyperparameter for NodePiece feature prediction. The hyperparameter setting for NodePiece requires a giant grid search. To simplify the process, we continue to use the convention in [19] by sampling 40% of the anchor nodes by centrality, 50% by Personalised PageRank and 10% by random sampling. In the experiment, we test a range of the number of anchors $|A| \in \{200, 500, 1000\}$ and the number of neighbourhoods $k \in \{5, 20, 50, 100\}$.Table 4 demonstrates the hyper-parameter settings for NodePiece feature prediction. We do not find a clear pattern about how to choose the number of anchor nodes and neighbours.

Dataset	Regularisers	CP			DistMult			ComplEx		
		W/O	F2	N3	W/O	F2	N3	W/O	F2	N3
FB15k-237	regulariser weight λ	10	10	1	10	10	1	5	10	0.1
	#Anchor nodes	1k	1k	500	500	200	500	500	500	500
	#Neighbours	100	100	50	10	10	10	50	100	5
WN18RR	regulariser weight λ	100	10	10	1	1	0.1	10	10	0.1
	#Anchor nodes	1k	200	500	500	200	200	200	500	1000
	#Neighbours	100	50	10	50	10	50	10	50	10
Yago3-10	regulariser weight λ	1	10	1	1	1	1	10	0.1	1
	#Anchor nodes	1k	500	500	500	500	500	200	500	200
	#Neighbours	100	50	100	50	100	100	50	100	100

* GP - Gradient Penalty; IRID - In-range and in-domain feature;

Table 4: Best hyperparameter configuration: the NodePiece regularisers

Hyperparameter for Manifold regularisation. Hyperparameter selection for weight construction based on the Gaussian and k-nearest neighbours (KNN) methods are different. For KNN, the number of neighbours k needs to be selected manually, and we set the grid to be $[5, 10, 20]$ according to the experiments of manifold regularization for Matrix Factorization[6]. For the Gaussian kernel, the grid of precision parameter σ is determined to be $[1, 3, 5, 10]$ according to a preliminary calculation. The construction of the feature vector for similarity measure is similar to the IRID and NodePiece features so we do not further discuss it here. We omit the experiment of Manifold regularisation on Yago3-10 since it causes out-of-memory issues.

Observations from the hyperparameter configuration. Regarding the performance of norm-based regularisers (L1+L2+N3), we find nuclear norm dominates the performance in the experiments of norm-based regularisers. The best performance is observed when the Nuclear norm is applied solely and l_1, l_2 norms can only bring a small change. From the experiment results, gradient penalty can help F2, or without regularisers, but when combined with n3, it does not work quite well. The hyperparameter configuration is consistent with the model performance. It is obvious that when all the regularisers are combined with N3, the regulariser weights decay significantly. That somehow suggests those regularisers do not have a dominant impact when interacting with N3 regularisers.

A.2 Standalone-mode Regularisation Experiment results

Table 5 illustrates the experiment results when the regularisers are applied in a standalone mode. From the results, some regularisers that are claimed helpful in previous literature, e.g. relation prediction and DURA regularisers do not bring a comparative boost to our proposed methods, and this impact is related to the datasets. Specifically, the most useful regulariser is the gradient penalty, which consistently improves the model performance. And auxiliary training objective benefits FB15k-237 the most. While in other datasets there is no significant improvement found. We suspect that the auxiliary training objective would also be influenced by the number of relation types. As suggested in [5], WN18RR and Yago3-10 have much smaller size of relation types than FB15k-237, and our training objectives are all constructed by relation types. So predicting the feature vectors in those datasets might be not be a complicated task, thus cannot boost the model performance a lot.

Even though the results from standalone-mode regularisers somehow suggest the impact of regularisers, we cannot give a conclusion just based on these. The KGC models without a regulariser penalising the parameter norms might converge to a sub-optimal scenario since the previous study suggests that the training process can trivially minimise the loss $\mathcal{L}$ by increasing the norm of the embeddings. Thus, further experiments with norm-based regularisers are required, as illustrated in the main body.

Dataset	Regularisers	CP			DistMult			ComplEx		
		MRR	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**
FB15k-237	W/O Reg	33.44	24.93	50.47	34.78	25.86	52.62	34.68	25.88	52.95
	RP	33.68	25.16	50.84	35.05	26.20	52.75	35.71	26.82	53.75
	GP	**34.44**	25.53	**52.10**	**35.77**	26.54	**54.55**	**36.34**	**27.22**	**54.67**
	IRID	33.98	25.22	51.42	35.58	**26.61**	53.65	35.50	26.36	53.99
	NodePiece	34.40	**25.63**	51.60	35.28	26.40	53.37	35.38	26.30	53.77
	DURA	32.61	24.04	49.59	/	/	/	34.52	25.45	52.61
WN18RR	W/O Reg	8.56	6.28	12.98	44.28	41.62	49.80	45.73	43.01	51.22
	RP	9.16	6.57	13.35	44.22	41.33	49.81	45.51	42.65	51.00
	GP	10.10	7.43	14.99	44.43	41.34	50.18	**46.84**	**43.15**	**54.15**
	IRID	10.31	7.67	14.91	**44.48**	**41.63**	50.14	45.99	42.74	51.83
	NodePiece	**11.56**	**8.12**	**18.67**	44.46	41.56	**50.77**	45.99	42.91	52.43
	DURA	10.16	7.43	15.36	/	/	/	45.86	42.77	52.37
Yago3-10	W/O Reg	55.31	48.79	67.24	56.58	49.51	69.64	57.99	50.93	70.87
	RP	**55.93**	49.34	**67.85**	56.74	49.48	**70.20**	57.71	50.68	70.61
	GP	55.54	49.01	67.42	56.83	49.66	69.82	**58.20**	**51.28**	**71.15**
	IRID	55.89	**49.55**	67.74	**56.84**	**49.88**	69.87	58.17	51.15	70.88
	NodePiece	55.50	48.90	67.42	56.84	49.83	69.76	57.95	51.15	70.71
	DURA	54.76	48.23	67.04	/	/	/	57.99	50.93	71.02

* GP - Gradient Penalty; IRID - In-range and in-domain feature; Embedding size = 1000 for yago3-10 and 2000 for the others

Table 5: Experiments results when the regularisers are applied in the standalone mode

A.3 The Experiment Results with Smaller Datasets

We did a few preliminary experiments on smaller datasets like Kinship, UMLS and Nations, and the experiments can be found in Table 6 and Table 7.

With the experiment results with the nuclear-3 norm, it is clear that the auxiliary training tasks (relation predictions, NodePiece feature prediction) can bring significant improvement to the KGC models. However, the impact of regularisers depends on the models and datasets. Specifically, relation prediction boosts experiments on Kinship most, and experiments on Nations benefit most from the NodePiece feature. However, the improvements are not always consistent - on the UMLS dataset, the regularisers do not have a good performance most of the time.

When the L2 norm is used, it can be shown that the gradient penalty can consistently benefit the models in Kinship and UMLS. And the model performance on Nations is still boosted by NodePiece features.

Even though the performance on small datasets can clearly show the power of new regularisers, they are rather unstable and heavily relied on hyper-parameter tuning. Thus, experiments on larger datasets are further required.

A.4 The numerical results for embedding sizes analysis

In Table 8 we illustrated the experiment results with/without auxiliary tasks among models with different embedding sizes as supplementary material to the visualisation figures Figure 2a.

From the results, the improvement brought by regularisers is most clear when larger embedding sizes are used. When model complexity is low, sometimes the models even do not benefit from the auxiliary tasks.

A.5 Derivation of the Analytical form of Gradient Penalty

The experiment results suggest that gradient penalty can bring an improvement to the model performance, but why does it work terribly when combined with the nuclear norm? With this question, we manage to find the explicit form of gradient penalty on CP, DistMult and ComplEx models. Given the score function of the DistMult in Section 3.2 (the derivation for CP and ComplEx works similarly),

$$\phi(s, r, o) = \mathbf{X}_{s,r,o} = \sum_k \mathbf{e}_s^k \mathbf{w}_r^k \mathbf{e}_o^k := \langle \mathbf{e}_s, \mathbf{w}_r, \mathbf{e}_o \rangle$$

Since that $\mathbf{e}_s^k$,$\mathbf{w}_r^k$ and $\mathbf{e}_o^k$ are all scalars, we can take derivative of ϕ with respective of $\mathbf{e}_s^k$

$$\frac{\partial \phi}{\partial \mathbf{e}_s^k} = \mathbf{w}_r^k \times \mathbf{e}_o^k$$

So in the vector form, the derivative is

$$\frac{\partial \phi}{\partial \mathbf{e}_s} = \mathbf{w}_r \odot \mathbf{e}_o$$

Where $\odot$ is the element-wise product. And the regularizer can be formed as

$$\begin{aligned} \mathbf{R}(\mathbf{X}) &= \|\mathbf{J}(\mathbf{e}_s)\|_2 + \|\mathbf{J}(\mathbf{w}_r)\|_2 + \|\mathbf{J}(\mathbf{e}_o)\|_2 \\ &= \|\mathbf{w}_r \odot \mathbf{e}_o\|_2 + \|\mathbf{w}_r \odot \mathbf{e}_s\|_2 + \|\mathbf{e}_s \odot \mathbf{e}_o\|_2 \end{aligned}$$

This expression works similarly to the l_2 norm but with a minor difference. Specifically, the l_2 norm only penalizes the embedding norm, which can be written as $\|\mathbf{e}_s\|$. But for gradient penalty, it penalises the product of embedding $\mathbf{e}_s \odot \mathbf{w}_r$. Overall, they all penalise the amplitude of embeddings.

In the calculation, we show that the gradient penalty works similarly to the l_2 norm. Because the l_2 norm and the nuclear norm actually work on the same thing - penalising the norm amplitude, it is not surprising that we cannot find a good performance when they interact with each other.

We finally conclude that gradient penalty could significantly improve the generalisation power of the TF-based neural link predictors, especially when applied to the model individually. It would increase the model's robustness against input perturbations, and apply a constraint to the norm at the same time.

Dataset	Regularisers	CP			DistMult			ComplEx		
		MRR	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**
Kinship	N3	82.78	73.22	97.65	51.77	36.03	86.21	86.43	78.77	98.23
	RP+N3	82.90	**74.39**	97.23	**56.49**	**41.99**	88.99	**88.54**	**82.26**	97.94
	GP+N3	83.01	74.20	97.42	52.23	36.87	86.64	87.54	80.49	**98.55**
	IRID+N3	82.64	73.22	97.66	53.70	38.71	87.73	86.35	78.77	97.65
	PATH+N3	82.37	72.94	97.83	55.82	41.29	**89.01**	86.35	78.77	97.65
	NodePiece+N3	**83.26**	74.15	**98.03**	52.95	37.96	87.40	85.89	78.02	98.46
UMLS	N3	89.33	81.36	98.38	74.59	64.98	**91.60**	96.08	93.41	**99.84**
	RP+N3	90.16	83.35	97.85	**78.38**	**70.71**	90.64	96.44	94.32	99.53
	GP+N3	89.51	81.75	98.46	74.74	64.97	91.45	**96.74**	**94.55**	99.70
	IRID+N3	90.00	82.20	**99.07**	77.22	69.55	91.02	95.89	92.96	99.69
	NodePiece+N3	**90.42**	**83.81**	99.00	77.52	69.63	91.56	95.97	93.42	99.70
Nations	N3	73.15	57.79	99.75	74.84	62.43	99.50	79.24	65.07	99.75
	RP+N3	75.99	61.81	99.74	80.15	69.60	99.74	80.80	69.60	**100.0**
	GP+N3	73.92	59.55	**100.0**	76.20	64.67	**100.0**	76.64	63.18	99.75
	IRID+N3	73.53	58.04	99.75	77.85	67.33	99.49	78.40	65.17	99.75
	NodePiece+N3	**76.52**	**63.07**	**100.0**	**81.26**	**72.61**	99.50	**82.48**	**72.89**	99.75

* GP - Gradient Penalty; IRID - In-range and in-domain feature; Embedding size = 200

Table 6: Experiments results on small datasets with N3 norm

Dataset	Regularisers	CP			DistMult			ComplEx		
		MRR	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**	**MRR**	**H@1**	**H@10**
Kinship	L2	77.59	66.25	96.68	48.31	33.47	**84.46**	8363	75.14	97.09
	RP+L2	80.50	71.30	96.58	48.00	33.71	83.33	86.77	80.06	97.56
	GP+L2	**83.92**	**75.23**	**97.42**	**48.55**	34.03	84.08	**88.17**	**81.93**	**97.89**
	IRID+L2	77.74	66.39	96.72	48.50	**34.32**	83.05	82.49	73.08	97.47
	NodePiece+L2	79.09	68.77	96.58	48.55	34.22	83.85	83.77	74.72	97.89
UMLS	L2	89.94	83.51	89.39	72.32	65.03	87.96	95.61	93.02	99.15
	RP+L2	90.16	83.82	97.85	71.97	64.19	87.42	95.95	93.71	99.15
	GP+L2	89.92	82.75	98.15	**72.94**	**65.18**	**88.88**	**96.16**	**94.01**	**99.38**
	IRID+L2	**91.07**	**85.11**	**98.54**	72.26	64.26	88.65	94.61	91.33	99.30
	NodePiece+L2	90.59	84.43	98.08	73.54	66.18	88.88	95.37	92.06	99.70
Nations	L2	75.42	60.30	100.0	79.36	68.59	100.0	80.04	67.59	99.49
	RP+L2	74.96	61.31	99.50	80.15	69.60	99.74	80.80	69.60	**100.0**
	GP+L2	76.01	61.56	9949	79.46	69.34	99.49	81.59	70.60	100.0
	IRID+L2	75.03	59.55	99.75	79.15	68.59	99.49	82.48	72.36	99.75
	NodePiece+L2	**77.28**	**64.32**	**100.0**	**81.32**	**71.10**	**100.0**	**83.48**	**74.12**	99.75

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

Table 7: Experiments results on small datasets with L2 norm

Table 8: The detailed results of embedding size analysis for feature prediction

	Model	Baseline*				With Feature Prediction			
		MRR	H@1	H@3	H@10	MRR	H@1	H@3	H@ 10
IRID	100	**28.90**	**20.96**	**31.40**	45.01	28.56	20.70	30.91	44.43
	200	31.57	23.10	34.26	48.55	**31.80**	**23.41**	**34.63**	**48.79**
	500	33.68	24.72	36.95	51.29	**34.27**	**25.33**	**37.73**	**52.26**
	1000	34.58	25.51	37.86	52.50	**35.11**	25.99	**38.69**	**53.33**
	2000	34.79	25.74	38.04	52.91	**35.50**	**26.36**	38.91	**53.99**
NP	100	-	-	-	-	28.75	20.75	31.28	**45.08**
	200	-	-	-	-	31.57	23.14	34.48	48.57
	500	-	-	-	-	34.05	25.15	37.50	51.94
	1000	-	-	-	-	34.94	**26.00**	38.33	52.89
	2000	-	-	-	-	35.37	26.30	**38.96**	53.77

* The baselines for IRID and NodePiece representation are the same, which we leave '-' here.