BEYOND LINK PREDICTION: ON PRE-TRAINING KNOWLEDGE GRAPH EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge graph embeddings (KGE) models provide low-dimensional representations of the entities and relations in a knowledge graph (KG). Most prior work focuses on training and evaluating KGE models for the task of link prediction; the question of whether or not KGE models provide useful representations more generally remains largely open. In this work, we explore the suitability of KGE models (i) for more general graph-structure prediction tasks and (ii) for downstream tasks such as entity classification. For (i), we found that commonly trained KGE models often perform poorly at structural tasks other than link prediction. Based on this observation, we propose a more general multi-task training approach, which includes additional self-supervised tasks such as neighborhood prediction or domain prediction. In our experiments, these multi-task KGE models showed significantly better overall performance for structural prediction tasks. For (ii), we investigate whether KGE models provide useful features for a variety of downstream tasks. Here we view KGE models as a form of self-supervised pre-training and study the impact of both model training and model selection on downstream task performance. We found that multi-task pre-training can (but does not always) significantly improve performance and that KGE models can (but do not always) compete with or even outperform task-specific GNNs trained in a supervised fashion. Our work suggests that more research is needed on how to pre-train KGE models and on their suitability for downstream applications.

1 INTRODUCTION

Knowledge graph embeddings (KGE) provide low-dimension representations of the entities and relations of a knowledge graph (KG). Although a large number of KGE models have been proposed in the literature—see for example the surveys of Nickel et al. (2015),Wang et al. (2017) and Ji et al. (2021)—, most prior work focuses on the task of link prediction, i.e., answering questions such as *(Austin, capitalOf, ?)* by reasoning over an incomplete KB. In addition to link prediction, it is often argued that KGEs can provide representations that capture semantic properties of the entities and, indeed, KGE models have been used to inject structured knowledge into language models (He et al., 2020; Zhang et al., 2019), visual models (Baier et al., 2017), recommender systems (Wang et al., 2018), and other types of downstream models (Wang et al., 2017).

The question of whether KGE models provide useful representations more generally remains largely open, however. Likewise, it is not well-understood how choices taken in model training and model selection affect these representations. In this work, we shed light onto these questions from multiple directions.

First, we study the suitability of out-of-the-box KGE models for basic graph-structure prediction tasks beyond link prediction. In particular, we consider the tasks of predicting the relation of a triple as suggested by Chang et al. (2020) (e.g., the relationship between *Austin* and *Texas*), the domain and range of a relation (e.g., whether *Austin* is a capital), as well as entity and relation neighborhood of each entity (e.g., which other entities are related to *Austin*). Perhaps surprisingly, we found that commonly trained KGE models often performed poorly on such tasks, challenging the intuition that KGE models capture graph structure well.

Second, we investigate whether KGE models are suitable pre-trained representations for node-level downstream tasks such as entity classification (e.g., the profession of a person) or regression (e.g.,

the average rating of a movie). To do so, we conducted an empirical study using 27 downstream tasks on two different KGs. We found that out-of-the-box KGE models often perform decent on these tasks and, in fact, the best KGE models can (but do not always) exceed the performance of recent graph neural networks such as KE-GCN (Yu et al., 2021). The KGE models with best downstream task performance were often not the best-performing models for link prediction, however. For example, we found that the basic TransE model (Bordes et al., 2013) may be superior to KGE models more suited to link prediction such as ComplEx (Trouillon et al., 2016) or RotatE (Sun et al., 2019).

Both of these findings suggest that the focus on link prediction tasks is too narrow. We thus explore whether the performance of KGE models for both graph-structure prediction and downstream tasks can be improved by better pre-training and model selection. Inspired by multi-task approaches in other areas—such as natural language processing (Aribandi et al., 2022; Sanh et al., 2022) or computer vision (Doersch & Zisserman, 2017)—, we included the graph-structure prediction tasks discussed above as additional training objectives and as evaluation measures during model selection. In particular, we propose a multi-task training (MTT) and a multi-task ranking (MTR) approach that both can be used along with an arbitrary KGE model class and without a substantial increase in computational cost. In our experimental study, the resulting multi-task KGE models had significantly better overall performance for graph-structure prediction tasks and often (but not always) also led to better downstream task performance.

In summary, the contributions of this paper are as follows: (i) We show empirically that commonly trained KGE models fail at basic graph-structure prediction tasks beyond link prediction. (ii) We propose novel multi-task training and ranking approaches that address this shortcoming. (iii) We explore the impact of standard and multi-task training as well as different approaches for model selection on downstream task performance. (iv) We contextualize KGE model performance with results obtained from recent graph neural networks, which—in contrast to KGE models—are trained directly on each downstream task. Although our work makes a step toward improved pre-training of KGE models, it also suggests that more research is needed on how to pre-train KGE models as well as on their general suitability for downstream applications.

2 Preliminaries and Related Work

We briefly describe KGE models, training and evaluation methods for link prediction, as well as prior work on other tasks. A more comprehensive discussion can be found in surveys such as (Nickel et al., 2015; Wang et al., 2017; Ji et al., 2021).

Link prediction. A *knowledge graph* $\mathcal{G} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a collection of *(subject, predicate, object)*-triples over a set $\mathcal{E}$ of entities and a set $\mathcal{R}$ of relations. Triples represent known facts such as *(Austin, capitalOf, Texas)*. The *link prediction* task considered in the context of the KGE literature is the task of inferring the subject or object to questions of form *(?, capitalOf, Texas)* and *(Austin, capitalOf, ?)*, respectively.

KGE models. KGE models (Sun et al., 2019; Trouillon et al., 2016; Yang et al., 2015; Bordes et al., 2013) represent each entity and each relation of a KG with a a low-dimensional embedding, commonly a real or complex vector. KGE models have an associated *scoring function* $\mathrm{s} : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \to \mathbb{R}$ that associates each triple with a real-valued score. Intuitively, high scores indicate plausible triples, low scores implausible triples. Commonly, the scoring function depends on the input triple only through the embeddings of its arguments. For example, TransE (Bordes et al., 2013) is a translation-based model with $\mathrm{s}(i, k, j) = -\|\boldsymbol{e}_i + \boldsymbol{r}_k - \boldsymbol{e}_j\|$, where $\boldsymbol{e}_i \in \mathbb{R}^d$ and $\boldsymbol{r}_k \in \mathbb{R}^d$ denote entity and relation embeddings of dimensionality $d > 0$, respectively. Scoring functions can be more involved, e.g., based on convolutional neural networks (Dettmers et al., 2018) or transformers (Chen et al., 2021a).

Standard training. KGE models are commonly trained on the link prediction task. We only give a high-level description here. For each triple (s, p, o) in the training data $\mathcal{G}_{\text{train}}$, KGE models are trained such that the score $\mathrm{s}(s, p, o)$ is high (a positive) but, for certain choices of $o' \in \mathcal{E}$ such that $(s, p, o') \notin \mathcal{G}_{\text{train}}$, the score of $\mathrm{s}(s, p, o')$ is low (a negative); similarly for subjects $s' \in \mathcal{E}$ with $(s', p, o) \notin \mathcal{G}_{\text{train}}$. The actual cost function varies across training types (e.g., sampled negatives or all negatives), loss

function (e.g, cross entropy), and more generally the choice of hyperparameters; see (Ali et al., 2021; Ruffinelli et al., 2020) for a more detailed discussion and experimental comparison.

Standard evaluation. The most commonly used evaluation protocol for KGE models is *entity ranking* (ER), which is also based on link prediction. Given a test triple $(s, p, o) \notin \mathcal{G}_{\text{train}}$, the model is used to answer the link prediction queries $(s, p, ?)$ and $(?, p, o)$. In particular, the scores of all possible answers that do not already occur in the training data are computed. The model is evaluated based on the rank of the test answers s and o, respectively. Common metrics are the mean reciprocal rank (MRR) and Hits@K. The reliability of entity ranking in assessing model performance was studied and questioned, e.g., in (Safavi & Koutra, 2020; Tiwari et al., 2021; Zhou et al., 2022). In contrast, our focus is mostly on other evaluation tasks.

Other training approaches. RESCAL (Nickel et al., 2011), one of the earliest KGE models, was inspired by tensor factorization methods and trained on the *reconstruction task*. Such tasks aim to to construct the entire training data using cost functions such as $\sum_{s,p,o} \| I[(s, p, o) \in \mathcal{G}_{\text{train}}] - \mathrm{s}(s, p, o) \|_2^2$, where $I[\cdot]$ is a 0/1 indicator. A similar approach was explored by Wang et al. (2019) and Li et al. (2021). We do not consider such methods further because training costs are excessive (at least unless squared error is used) and the empirical performance reported in (Wang et al., 2019; Li et al., 2021) is generally far behind KGE models trained with link prediction. Chen et al. (2021b) proposed to augment the link prediction task with *relation prediction* during training (but not evaluation). We expand upon this work by considering additional pre-training tasks and by focusing on graph-structure prediction and downstream task performance instead.

Other evaluation approaches. In early (and rarely in recent) work, KGE models were evaluated using *triple classification* (Socher et al., 2013; Wang et al., 2014; Lin et al., 2015; Wang et al., 2022). We do not consider this task in this work because performance estimates are typically overly optimistic and misleading unless hard negatives are used Safavi & Koutra (2020); such hard negatives are generally not available. Chang et al. (2020) evaluated KGE models on the relation prediction task, which we also consider as one of the evaluation tasks in this work. There is also work on explaining or interpreting KGE models (Meilicke et al., 2018; Allen et al., 2021; Rim et al., 2021), whereas our focus is on studying whether such models provide useful representations in the first place. As mentioned in the introduction, pre-trained KGE models have been used as a components in language models (He et al., 2020; Zhang et al., 2019), visual models (Baier et al., 2017), or recommender systems (Wang et al., 2018). Likewise, (Pezeshkpour et al., 2018; Jain et al., 2021) evaluated pre-trained KGE models for entity classification or regression tasks, as we do. We expand on this line of work by using a larger set of tasks (graph-structure prediction and more downstream tasks), by proposing improved pre-training methods, and by studying the impact of pre-training on downstream task performance.

3 GRAPH-STRUCTURE PREDICTION

In addition to link prediction, we explore the suitability of KGE models for other basic graph-structure prediction tasks. An example and summary is given in Table 1. We describe the form of the *queries* for each task as a triple such as $(s, ?, *)$, where s or o denote input entities, p denotes an input relation, $?$ denotes the prediction target, and $*$ acts as a wildcard. Using this notation, we consider the following tasks and queries:

- **Link prediction** (LP): Given a relation and a subject, predict the object (denoted $(s, p, ?)$). Likewise, given a relation and an object, predict the subject (denoted $(?, p, o)$).
- **Relation prediction** (REL, Chang et al. (2020); Chen et al. (2021b): Given two entities s and p, predict the relation between them (denoted $(s, ?, o)$).
- **Domain prediction** (DOM): Given a relation, predict its domain (denoted $(?, p, *)$) or its range (denoted $(*, p, ?)$).
- **Entity neighborhood prediction** (NBE): Given a subject entity, predict related objects (denoted $(s, *, ?)$). Likewise, given an object, predict related subjects (denoted $(?, *, o)$).
- **Relation neighborhood prediction** (NBR): Given a entity, predict the relations where it occurs as subject (denoted $(s, ?, *)$). Likewise, predict the relations where it occurs as object (denoted $(*, ?, o)$).

Knowledge graph	Task	Example query	Some answers
(Dallas, locatedIn, Texas)	Link (LP)	*(Austin, locatedIn, ?)*	*Texas, USA*
(Texas, locatedIn, USA)		*(?, locatedIn, Texas)*	*Austin, Dallas*
(Austin, capitalOf, Texas)	Relation (REL)	*(Austin, ?, Texas)*	*locatedIn, capitalOf*
(Austin, locatedIn, Texas)	Domain (DOM)	*(*, locatedIn, ?)*	*Texas, USA, North A.*
(Arkansas, borders, Texas)		*(?, locatedIn, *)*	*Dallas, Texas, USA*
(USA, locatedIn, North A.)	Entity neighb. (NBE)	*(Austin, *, ?)*	*Texas, USA*
(Austin, locatedIn, USA)		*(?, *, Texas)*	*Dallas, Arkansas*
	Relation neighb. (NBR)	*(Austin, ?, *)*	*capitalOf, locatedIn*
		(, ?, Texas)*	*borders, capitalOf*

Table 1: Graph-structure prediction tasks used for self-supervised pre-training and evaluation along with example queries. Here $?$ denotes the prediction target and $*$ acts as a wildcard.

Note that we use the wildcard to denote existential quantification. For example, given a ground-truth KG $\mathcal{G}$ and domain prediction query $(?, p, *)$, an entity $s \in \mathcal{E}$ is a correct answer if there exists an entity $o \in \mathcal{E}$ such that $(s, p, o) \in \mathcal{G}$.

We chose this particular set of tasks because they are simple, they capture basic information about the graph structure beyond link prediction, and they only have one prediction target (an entity or a relation). The latter property allows efficient pre-training and evaluation, as discussed below. For this reason, we exclude tasks such as entity-pair prediction Wang et al. (2019) (denoted $(?, p, ?)$ in our notation) or reconstruction Nickel et al. (2011) (denoted $(?, ?, ?)$). We leave the exploration of these or other pre-training tasks to future work.

Multi-task ranking (MTR). To evaluate the performance of KGE models on the graph-structure prediction tasks, we generalize the entity ranking (ER) protocol for link prediction. Intuitively, for each of the nine tasks (REL as well as LP/DOM/NBE/NBR for both subject targets and for object targets), we construct a query from each test triple,[1] obtain a ranking of the prediction targets (entity or relation) that do not already occur in the training data, and then use metrics such as MRR or Hits@K. The final MTR metric is given by the macro-average over all nine tasks.

We now describe how to obtain task-specific rankings. First, for a REL query of form $(s, ?, o)$, we proceed as in (Chang et al., 2020) and rank all $r' \in \mathcal{R}$ such that $(s, r', o) \notin \mathcal{G}_{\text{train}}$ in descending order of their scores $\mathrm{s}(s, r', o)$. For the other tasks, which involve wildcards, it is not immediately clear how to perform prediction using a KGE model. We first discuss scoring and ranking, then filtering of training data. Consider for example the NBR query $(s, ?, *)$, where our goal is to rank relations. The perhaps simplest approach to obtain a relation ranking is to first rank all triples of form (s, r', o'), where $r' \in \mathcal{R}$ and $o' \in \mathcal{E}$, and then rank relations by their first appearance (e.g., the relation of the highest-scoring triple is ranked at the top). More generally, we make use of an *extended score function* that accepts wildcards. The approach just described corresponds to using $\mathrm{s}(s, r', *) = \max_{o' \in \mathcal{E}} \mathrm{s}(s, r', o')$, i.e, the score of a relation r' is the score of its most plausible triple. Although other aggregation functions are feasible, we only consider max-aggregation because it does not make any additional assumptions on the scoring function. To filter training data during model evaluation, we remove all relations r' such that $(s, r', o') \in \mathcal{G}_{\text{train}}$ for some $o' \in \mathcal{E}$; i.e., we remove all prediction targets that are already implied by the training data. We proceed similarly for all other tasks involving wildcards.

Note that the number of score computations needed to predict entity targets for queries without wildcards is $O(|\mathcal{E}|)$, whereas the one for queries with wildcards is $O(|\mathcal{E}||\mathcal{R}|)$. We discuss below how the latter cost can be reduced to $O(|\mathcal{E}|)$ when the KGE model is trained with MTT.

Multi-task training (MTT). We now generalize standard KGE model training to all of the graph-structure prediction tasks. Our goal is to be able to improve KGE model performance at these tasks, while at the same time keeping training and prediction cost low. We do this by constructing a task-specific cost function for each individual task first; the final cost function is then given as a weighted

[1] The nine queries for test triple (s, p, o) are precisely the ones given in the task descriptions.

Dataset	Entities	Relations	Train	Validation	Test
FB15K-237	14 505	237	272 115	17 535	20 466
YAGO3-10	123 182	37	1 079 040	5 000	5 000
WNRR	40 559	11	86 835	3 034	3 134

Table 2: Statistics of benchmark datasets for pre-training KGEs

linear combination of the task-specific costs (and additional regularization terms), where the weights are hyperparameters.

The task-specific cost functions for link prediction and relation prediction are obtained as in standard training (Sec. 2): For each positive triple $(s, p, o) \in \mathcal{G}$, we construct a set of negatives according to the query (i.e., by perturbing the position of the prediction target) and then apply the loss function (e.g., cross entropy). For the other tasks, which involve wildcards, we proceed differently. Instead of performing some form of (costly) score aggregation during training, we "convert" tasks with wildcards into tasks without wildcards. To do so, we make use of three virtual *wildcard entities*—one for subjects (any_S), one for relations (any_R), and one for objects (any_O)—and learn embeddings for these entities. During training, we conceptually replace wildcards by their the corresponding wildcard entity and proceed as before. For example, for training triple (s, p, o) and NBR query $(s, ?, *)$, we consider the virtual triple (s, p, any_O) along with query $(s, ?, any_O)$. By doing so, we converted the NBR task into a REL task.

We also use the so-obtained wildcard embeddings for prediction in the same fashion; e.g., we set $\mathrm{s}(s, r', *) = \mathrm{s}(s, r', any_O)$. The advantage of this approach is that (i) training and prediction costs are as low as for the LP or REL tasks, and (ii) instead of performing score aggregation, the model directly learns extended scores.

Note that the wildcard embeddings are not useful for entity-level downstream tasks, where they are not used. Nevertheless, using wildcard entities during training affects all other entities as well. This is because the embedding of each entity occurs in all graph-structure prediction tasks. The entity embeddings of a low-cost KGE model thus needs to be suitable for all these tasks, not just for link prediction.

4 Experimental Study

We conducted a large experimental study. Our goals were (i) to assess KGE model performance for graph-structure prediction, (ii) to assess performance of pre-trained KGE models on downstream tasks, (iii) to assess the effect of multi-task training on both graph-structure prediction and downstream task performance, and (iv) to contextualize these results by comparing them to results obtained by recent graph neural networks (GNNs).

4.1 Experimental Setup

Knowledge graphs. We used three commonly used benchmark datasets for evaluating KGE models: FB15K-237 (Toutanova & Chen, 2015), WNRR (Dettmers et al., 2018), and YAGO3-10 (Mahdisoltani et al., 2014). Each dataset is associated with a training, a validation and a test split. FB15K-237 and WNRR are designed to be harder benchmarks for link prediction. YAGO3-10 is not, but it is considerably larger. Dataset statistics are summarized in Table 2.

KGE models. We considered four popular, representative KGE models: TransE (Bordes et al., 2013) and DistMult (Yang et al., 2015) (basic translational and factorization models, resp.) as well as RotatE (Sun et al., 2019) and ComplEx (Trouillon et al., 2016) (SOTA translational and factorization models, resp.).

KGE training. We implemented STD and MTT/MTR model training/evaluation in LibKGE Ruffinelli et al. (2020). All KGE models were trained for a maximum of 200 epochs with early stopping on validation MRR checked every 10 epochs. We used cross-entropy as loss function, as it systematically outperformed other losses in most prior studies. We used *1vsAll* train-

ing with FB15K-237 and WNRR (to achieve good results) and *NegSamp* with YAGO3-10 (to scale to this larger dataset). Since we are interested in pre-trained KGE models, no information from downstream tasks is used for KGE model training and selection; e.g., the same KGE model is used for all downstream tasks in each experiment. In particular, models were selected w.r.t. performance (MRR) on the validation data. Unless stated otherwise, models trained with STD training use the LP task, models trained with MTT training use MTR. Further improvements may be made by leveraging downstream tasks during training (Aribandi et al., 2022); we leave such exploration to future work.

KGE evaluation. We evaluate KGE models with respect each of the five graph-structure prediction tasks of Sec. 3 (LP, REL, DOM, NBE, NBR) using filtered MRR on test data. We also aggregate these metrics into the multi-task ranking MRR (MTR).

KGE hyperparameters. We closely follow the approach of the experimental study of Ruffinelli et al. (2020) to perform hyperparameter selection. We performed 30 random trials using SOBOL sampling (Bergstra & Bengio, 2012) over a large search space to tune several hyperparameters, e.g. regularization, embedding size, batch size, dropout, initialization, and task weights (each in $[0.1, 10.0]$, log scale). To keep our study feasible, we reduced the maximum batch and embedding size for larger datasets and expensive models. The full search space can be found in Table 8.

Downstream tasks. We collected or created data for 27 downstream tasks on FB15K-237 or YAGO3-10. This includes the datasets of Jain et al. (2021) for entity classification on FB15K-237 and YAGO3-10, which aims to predict the types of entities at different granularities. For regression, we use the datasets of Pezeshkpour et al. (2018) for YAGO3-10, which consist of temporal prediction tasks (e.g., the year an event took place), and the dataset of Huang et al. (2021) for node importance prediction. We also created several regression tasks for FB15K-237 from the multi-modal data of García-Durán et al. (2018) by predicting literals associated to entities (e.g., a date, a person's height, the rating of a movie). All datasets and statistics are summarized in Tables 6 and 7 in the appendix.

Downstream models. We use scikit-learn (Pedregosa et al., 2011) using only the node embedding of the pre-trained KG model as input. For classification, we consider a multi-layer perceptrons (MLP), logistic regression, KNN, and random forests. For regression, we consider MLP and linear regression.

Downstream training. Each model was trained using 5-fold cross validation and selected based on mean validation performance across folds (see below). We then retrained the selected model on the union of the training and validation split (if present). To tune hyperparameters, we use 10 trials of random search with SOBOL sampling for each downstream model. The search space is given in Table 9. Note that we treat the choice of downstream model as a hyperparameter as well.

Downstream evaluation. For entity classification, we report *weighted F1* as in (Jain et al., 2021), i.e., we aggregated across different classification tasks (denote EC). For regression, we chose relative squared error (RSE) because it is interpretable and allows meaningful averaging across the different regression tasks (denoted REG). An RSE value of 1 is equivalent to the performance of a model that always predicts the average of the dependent variable in the training data; lower values are better. For each metric, we report the mean and standard deviation over 3 training runs of the downstream model.

Downstream baselines. We consider multiple baseline models to contextualize the results obtained by pre-trained KGE models. Note that in contrast to KGEs, the baselines are directly trained on the respective downstream task (i.e., no pre-training) and they need to access the KG to perform predictions. We include KE-GCN of Yu et al. (2021) as a recent GNN, which achieved state-of-the-art results for graph alignment and entity classification in their study. For regression tasks, we use a linear layer after the final convolutional layer of KE-GCN.[2] We tune hyperparameters using 30 SOBAL trials (as for KGE models); the search space is shown in Table 9. For training, evaluation, and model selection, we otherwise follow the approach used for four downstream models (e.g,. 5-fold CV). We also consider selected SOTA results of other downstream models; see Sec. 4.5.

[2]In our experiments, this led to better performance than using a single dimensional output in the final convolution layer as done by Huang et al. (2021).

			Graph-structure prediction (↑)						*Downstream tasks*	
			LP	REL	DOM	NBE	NBR	MTR	EC (↑)	REG (↓)
FB15K-237	ComplEx	STD	**.347**	.805	.098	.011	.041	.200	.844±.008	.447±.051
		MTT	.331	**.976**	.159	.048	**.672**	**.378**	.838±.002	**.441±.050**
	DistMult	STD	.342	.388	.032	.007	.033	.135	.873±.009	.551±.062
		MTT	.327	.957	**.160**	.034	.670	.371	.862±.001	.490±.052
	RotatE	STD	.287	.919	.114	.024	.106	.225	.864±.005	.618±.037
		MTT	.295	.965	.158	.047	.658	.370	**.886±.003**	.546±.081
	TransE	STD	.300	.900	.111	.018	.049	.213	.881±.003	.628±.045
		MTT	.291	.963	.155	**.062**	.641	.368	.855±.003	.991±.147
	KE-GCN†		–	–	–	–	–	–	.829±.526	.501±.001
YAGO3-10	ComplEx	STD	**.550**	.890	.001	.050	.386	.318	.712±.008	.589±.023
		MTT	.510	.943	.045	.071	.713	.403	.729±.003	.539±.037
	DistMult	STD	.539	.877	.004	.069	.434	.330	.738±.003	.519±.019
		MTT	.538	.941	**.046**	.061	**.740**	**.412**	.745±.007	.476±.059
	RotatE*	STD	.429	–	–	–	–	–	.693±.002	.745±.030
		MTT	.324	.931	.032	.079	.638	.343	.727±.004	.593±.074
	TransE*	STD	.490	–	–	–	–	–	.741±.001	.484±.053
		MTT	.263	**.959**	.037	**.080**	.617	.330	**.755±.003**	**.326±.022**
	KE-GCN†		–	–	–	–	–	–	.700±.223	.398±.008
WNRR	ComplEx	STD	**.474**	.782	.001	.150	.578	.354	–	–
		MTT	.460	.833	.024	.181	**.791**	.416	–	–
	DistMult	STD	.447	.767	.001	.169	.605	.356	–	–
		MTT	.435	**.897**	**.025**	.190	.781	**.417**	–	–
	RotatE	STD	.460	.794	.001	**.210**	.713	.398	–	–
		MTT	.422	.874	.024	.152	.790	.408	–	–
	TransE	STD	.174	.707	.001	.119	.307	.212	–	–
		MTT	.175	.837	.021	.109	.747	.327	–	–

* Not evaluated on new graph-structure prediction tasks due to high cost.
† GCN-based model by Yu et al. (2021) trained directly on downstream tasks.

Table 3: Performance on graph-structure prediction and downstream tasks of STD and multi-task KGE models as well as KE-GCN on test data. For graph-structure prediction, we report MRR (higher is better), for entity classification (EC) we report weighted F1 (higher is better), and for regression (REG) we show relative squared error (lower is better). Bold entries show best performance per task. Underlined entries show best performance between STD and MTT training.

			Selection Method					
			EC - Weighted F1 (↑)			*REG - RSE (↓)*		
			LP	MTR	Weighted F1	LP	MTR	RSE
FB15K-237	ComplEx	STD	.844	.830	.850	.447	.654	.437
		MTT	.858	.838	.827	.394	.441	.393
	DistMult	STD	.873	.825	.846	.550	.677	.539
		MTT	.865	.861	.864	.471	.489	.476

Table 4: Performance on FB15k-237 downstream tasks for different KGE model training (STD/MTT) and selection approaches (LP/MTR/weighted F1/RSE). Note that weighted F1/RSE use downstream tasks data for model selection, wheres LP/MTR does not.

Publication	Dataset	Task	Metric	Model	Value
Bloem et al. (2021)	MDGENRE	EC	Accuracy	Features	0.66
				ComplEx (MTR)	0.66±0.01
				ComplEx (Acc)	0.68±0.00
	DMGFULL	EC	Accuracy	MR-GCN	**0.76**
				ComplEx (MTR)	0.51±0.00
				ComplEx (Acc)	0.67±0.02
Huang et al. (2021)	FB15K	NIE	NDCG@100	RGTN	**0.95**
				ComplEx (MTR)	0.41±0.00
				ComplEx (NDCG)	0.42±0.00
			Spearman	RGTN	**0.82**
				ComplEx (MTR)	0.748±0.00
				ComplEx (Spr)	0.76±0.00
Vashishth et al. (2020)	MUTAG	EC	Accuracy	CompGCN	**0.85**
				ComplEx (MTR)	0.75±0.024
				ComplEx (Acc)	0.75±0.012
Schlichtkrull et al. (2018)	AIFB	EC	Accuracy	R-GCN	**0.95**
				ComplEx (MTR)	0.88±0.00
				ComplEx (Acc)	0.88±0.00

Table 5: Comparison of downstream task entity classification (EC) and node importance estimation (NIE) between best previously published models (trained on task) pre-trained KGEs trained (MTT, selection with MTR or downstream task metric).

4.2 Graph-structure prediction

In Table 3, we report test MRR of all graph-structure prediction tasks from Table 1 for KGE models using standard training and link prediction for model selection (STD) and our proposed multi-task training and model selection (MTT). Bold entries show best performance per metric and evaluation method. For easier comparison between STD and MTT training, underlined entries highlight the best performance compared to the entry with the same corresponding KGE model on the same dataset, but that uses the other training method. The columns labeled *Downstream Tasks* are discussed in Sec. 4.3.

The results show that across all datasets and KGE models, STD training approach performs poorly on all graph-structure tasks, except LP and (often) REL. The performance for these tasks is significantly increased using MTT training. This is because these tasks have been introduced as auxiliary training objectives. Also note that MTT models have slightly lower performance on LP, but the decrease is often small and outweighted by significantly improved performance over the other tasks (often 2x–4x, up to 10x, depending on model, task and dataset). We conclude that MTT training is beneficial for graph-structure prediction tasks, where STD training is not.

4.3 Downstream Tasks

Table 3 includes mean performance across all downstream tasks for each benchmark dataset. We report Weighted F1 (higher is better) for entity classification tasks and relative squared error (lower is better) for regression tasks. As before, bold entries show best performance per metric and evaluation method, and underlined entries facilitate performance comparisons across the different training approaches. Performance on individual downstream tasks are reported in the appendix.

The results show that, perhaps surprisingly, models with weaker performance during pre-training often perform competitively in downstream tasks and sometimes even outperform models with stronger performance in graph-structure tasks. In addition, we see that MTT training achieves the top performance in all cases, but often not by a large margin, and STD models do perform better at times when comparing training methods on the same model. Compared to KE-GCN, which trains directly on each task, pre-trained KGEs perform competitively and are even superior at times. Taken

together, these results suggest that KGE models do provide suitable representations for downstream tasks.

4.4 COMPARISON TO SOTA RESULTS

In Table 5, we compare downstream task performance of a pre-trained ComplEx model using the MTT approach with state-of-the-art task-specific models from the literature. Note that we did not train these models ourselves and experimental setups used in prior work may be very different from ours. In these experiment, we did not reach state-of-the-art performance with KGE models. Aside from training directly for the task, this may be because (i) some of the datasets (AIFB, MUTAG) are very small so that KGE models do not learn much before overfitting, ann (ii) some knowledge graphs are multi-model (MDGENRE, DMGFULL), but we do not leverage this information[3] On MDGENRE, however, the additional modalities do not play a significant role, however, as not only does our pre-trained KGE achieve comparative performance, but so do the non-multimodal baselines from Bloem et al. (2021). This is not the case with the DMGFULL datasets, however, which contains a significantly higher number of triples from different modalities.

The results of this and the previous section suggests that more research is needed on the impact of different training approaches on downstream task performance of pre-trained KGE models.

4.5 IMPACT OF MODEL SELECTION

Table 4 shows performance on FB15K-237 of some KGE models using both training approaches across different types of model selection methods: selecting on LP (the standard approach), selecting on MTR and selecting directly on the metric used in to evaluate the downstream task (weighted F1 for entity classification and RSE for regression). We note that model selection with MTT training works well, as selecting directly on the downstream task metric (which we often cannot do in practice) often provides only marginal benefits. Note that model selection using the downstream task metric may also be detrimental to performance; e.g., due to potential overfitting on validation data.

5 CONCLUSION

In this work, we showed empirically that commonly trained KGE models fail at basic graph-structure prediction tasks beyond link prediction. We proposed a novel multi-task training and ranking approaches to pretrain KGE models, which leds to substantially better performance: multi-task KGE models thus capture more information about the graph structure. We explored the impact of standard and multi-task training as well as different approaches for model selection on downstream task performance and found that although standard training approach for KGE models led to decent performance, performance was often improved using multi-task training. Finally, we contextualized KGE model performance with results obtained from recent graph neural networks, which—in contrast to KGE models—are trained directly on each downstream task. Here KGE models were sometimes competetive, sometimes superior, and sometimes inferior. Although our work makes a step toward improved pre-training of KGE models, it also suggests that more research is needed on how to pre-train KGE models as well as on their general suitability for downstream applications.

[3]There are multi-model KGE models, however, e.g. (Pezeshkpour et al., 2018).

REFERENCES

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. Pykeen 1.0: A python library for training and evaluating knowledge graph embeddings. *J. Mach. Learn. Res.*, 22(82):1–6, 2021.

Carl Allen, Ivana Balazevic, and Timothy Hospedales. Interpreting knowlege graph relation representation from word embeddings. In *Ninth International Conference on Learning Representations 2021*, 2021.

Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q Tran, Dara Bahri, Jianmo Ni, et al. Ext5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*, 2022.

Stephan Baier, Yunpu Ma, and Volker Tresp. Improving visual relationship detection using semantic modeling of scene descriptions. In Claudia d'Amato, Miriam Fernandez, Valentina Tamma, Freddy Lecue, Philippe Cudré-Mauroux, Juan Sequeda, Christoph Lange, and Jeff Heflin (eds.), *ISWC*, pp. 53–68, 2017.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

Peter Bloem, Xander Wilcke, Lucas van Berkel, and Victor de Boer. kgbench: A collection of knowledge graph datasets for evaluating relational and multimodal machine learning. In *European Semantic Web Conference*, pp. 614–630. Springer, 2021.

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

David Chang, Ivana Balažević, Carl Allen, Daniel Chawla, Cynthia Brandt, and Richard Andrew Taylor. Benchmark and best practices for biomedical knowledge graph embeddings. In *Proceedings of the conference. Association for Computational Linguistics. Meeting*, volume 2020, pp. 167. NIH Public Access, 2020.

Sanxing Chen, Xiaodong Liu, Jianfeng Gao, Jian Jiao, Ruofei Zhang, and Yangfeng Ji. Hitter: Hierarchical transformers for knowledge graph embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 10395–10407, 2021a.

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*, 2021b.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. Convolutional 2d knowledge graph embeddings. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Carl Doersch and Andrew Zisserman. Multi-task self-supervised visual learning. In *Proceedings of the IEEE international conference on computer vision*, pp. 2051–2060, 2017.

Alberto García-Durán, Sebastijan Dumancic, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *EMNLP*, 2018.

Bin He, Di Zhou, Jinghui Xiao, Xin Jiang, Qun Liu, Nicholas Jing Yuan, and Tong Xu. Bert-mk: Integrating graph contextualized knowledge into pre-trained language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 2281–2290, 2020.

Han Huang, Leilei Sun, Bowen Du, Chuanren Liu, Weifeng Lv, and Hui Xiong. Representation learning on knowledge graphs for node importance estimation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

Nitisha Jain, Jan-Christoph Kalo, Wolf-Tilo Balke, and Ralf Krestel. Do embeddings actually capture knowledge graph semantics? In *European Semantic Web Conference*, pp. 143–159. Springer, 2021.

Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Zelong Li, Jianchao Ji, Zuohui Fu, Yingqiang Ge, Shuyuan Xu, Chong Chen, and Yongfeng Zhang. Efficient non-sampling knowledge graph embedding. In *Proceedings of the Web Conference 2021*, pp. 1727–1736, 2021.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. Learning entity and relation embeddings for knowledge graph completion. In *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

Farzaneh Mahdisoltani, Joanna Biega, and Fabian Suchanek. Yago3: A knowledge base from multilingual wikipedias. In *7th biennial conference on innovative data systems research*. CIDR Conference, 2014.

Christian Meilicke, Manuel Fink, Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, and Heiner Stuckenschmidt. Fine-grained evaluation of rule-and embedding-based systems for knowledge graph completion. In *International semantic web conference*, pp. 3–20. Springer, 2018.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. A three-way model for collective learning on multi-relational data. In *Icml*, 2011.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 2015.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 2011.

Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. Embedding multimodal relational data for knowledge base completion. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Wiem Ben Rim, Carolin Lawrence, Kiril Gashteovski, Mathias Niepert, and Naoaki Okazaki. Behavioral testing of knowledge graph embedding models for link prediction. In *3rd Conference on Automated Knowledge Base Construction*, 2021.

Daniel Ruffinelli, Samuel Broscheit, and Rainer Gemulla. You can teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*, 2020.

Tara Safavi and Danai Koutra. CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8328–8350, 2020.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations*, 2022.

Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European semantic web conference*, pp. 593–607. Springer, 2018.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. Reasoning with neural tensor networks for knowledge base completion. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger (eds.), *NIPS*, volume 26. Curran Associates, Inc., 2013.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. Rotate: Knowledge graph embedding by relational rotation in complex space. In *International Conference on Learning Representations*, 2019.

Sudhanshu Tiwari, Iti Bansal, and Carlos R Rivero. Revisiting the evaluation protocol of knowledge graph completion methods for link prediction. In *Proceedings of the Web Conference 2021*, pp. 809–820, 2021.

Kristina Toutanova and Danqi Chen. Observed versus latent features for knowledge base and text inference. In *Proceedings of the 3rd workshop on continuous vector space models and their compositionality*, pp. 57–66, 2015.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. Complex embeddings for simple link prediction. In *International conference on machine learning*, pp. 2071–2080. PMLR, 2016.

Shikhar Vashishth, Soumya Sanyal, Vikram Nitin, and Partha Talukdar. Composition-based multi-relational graph convolutional networks. In *International Conference on Learning Representations*, 2020.

Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Dkn: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*, pp. 1835–1844, 2018.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 2017.

Xintao Wang, Qianyu He, Jiaqing Liang, and Yanghua Xiao. Language models as knowledge embeddings. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2022)*, 2022.

Yanjie Wang, Daniel Ruffinelli, Rainer Gemulla, Samuel Broscheit, and Christian Meilicke. On evaluating embedding models for knowledge base completion. In *RepL4NLP@ ACL*, 2019.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

Bishan Yang, Scott Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. Embedding entities and relations for learning and inference in knowledge bases. In *Proceedings of the International Conference on Learning Representations (ICLR) 2015*, 2015.

Donghan Yu, Yiming Yang, Ruohong Zhang, and Yuexin Wu. Knowledge embedding based graph convolutional network. In *Proceedings of the Web Conference 2021*, pp. 1619–1628, 2021.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. Ernie: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 1441–1451, 2019.

Ying Zhou, Xuanang Chen, Ben He, Zheng Ye, and Le Sun. Re-thinking knowledge graph completion evaluation from an information retrieval perspective. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022.

A APPENDIX

A.1 DOWNSTREAM TASK STATISTICS

Benchmark	Name	Num. Classes	Train	Validation	Test
FB15K-237	Entity Type	3 (person, org, ...)	6 719	–	1 680
	Profession	5 (artist, writer, ...)	2 537	–	635
	Organization Type	4 (NGO, musical_org, ...)	342	–	86
	Writer Type	2 (journalist, poet, ...)	136	–	34
YAGO3-10	Entity Type	4 (person, org, ...)	69 592	–	17 398
	Player Type	5 (soccer, hockey, ...)	33 928	–	8 483
	Profession	5 (artists, politician, ...)	14 480	–	3 621
	Writer Type	7 (poet, novelist, ...)	4 870	–	1 218
	Scientist Type	10 (physicist, biologist, ...)	2 041	–	511
	Organization Type	4 (institution, party, ...)	1 248	–	312
	Artists Type	5 (painter, sculptor, ...)	520	–	130
	Waterbody Type	5 (lake, ocean, ...)	195	–	49

Table 6: Statistics of datasets for entity classification downstream tasks used to evaluate pre-trained KGEs. All datasets were created by Jain et al. (2021), they are split into trainining and test only and each consists of predicting entity types at different levels of the entity hierarchy.

Benchmark	Name	Task	Train	Validation	Test
FB15K-237	Node Importance	(Entity, Wikipedia page views)	9 877	1 380	2 823
	Birth Year	(People, year of birth)	3 538	442	444
	Latitude	(Location, latitude)	2 568	321	322
	Longitude	(Location, longitude)	2 560	320	322
	Person Height	(Person, height in meters)	2 295	287	288
	Size Area	(Location, area)	1 731	216	218
	Population	(Location, population)	1 543	193	193
	Film Release Year	(Film, release year)	1 493	186	188
	Org Year Founded	(Organization, year founded)	985	123	124
	Film Rating	(User, rated film 1 to 100)	591	73	75
YAGO3-10	Born on Year	(Person, year of birth)	60 409	–	6 730
	Created on Year	(Entity, year of creation)	23 896	–	2 638
	Died on Year	(Person, year of death)	13 582	–	1 513
	Destroyed on Year	(Entity, year of end/death)	1 630	–	186
	Happened on Year	(Event, year it took place)	749	–	73

Table 7: Statistics of datasets for regression downstream tasks used to evaluate pre-trained KGEs. YAGO3-10 datasets were created by Pezeshkpour et al. (2018). All FB15K-237 datasets were created by us, except the node importance task, which was created by Huang et al. (2021).

A.2 EXPERIMENTAL SETTINGS

Hyperparameter	Values
Embedding size	$\{128, 256, 512\}$
Training type	{NegSamp (YAGO3-10), 1vsAll (FB15K, WNRR)}
Task Weights (MTT)	$[0.1, 10]$, log scale
No. subject samples (NegSamp)	$[1, 10000]$, log scale
No. object samples (NegSamp)	$[1, 10000]$, log scale
Optimizer	{Adam, Adagrad}
Batch size	$\{128, 256, 512, 1024$(except on YAGO3-10)$\}$
Learning rate	$[10^{-4}, 1]$, log scale
LR scheduler patience	$[0, 10]$
L_p regularization	{L1, L2, L3, None}
Entity emb. weight	$[10^{-20}, 10^{-5}]$
Relation emb. weight	$[10^{-20}, 10^{-5}]$
Frequency weighting	{True, False}
Embedding normalization (TransE)	
Entity	{True, False}
Relation	{True, False}
Dropout	
Entity embedding	$[0.0, 0.5]$
Relation embedding	$[0.0, 0.5]$
Feature map (ConvE)	$[0.0, 0.5]$
Projection (ConvE)	$[0.0, 0.5]$
Embedding initialization	{Normal, Unif, XvNorm, XvUnif}
Std. deviation (Normal)	$[10^{-5}, 1.0]$
Interval (Unif)	$[-1.0, 1.0]$
Gain (XvNorm)	1.0
Gain (XvUnif)	1.0

Table 8: Hyperparameter search space for pre-training KGE models.

Model	Hyperparameter	Values
MLP	Hidden Layer Sizes	$\{(100,), (10,), (100, 100), (10, 10)\}$
	Alpha	$[0.00001, 0.001]$
	Learning Rate Init	$[0.001, 0.01]$
	Solver	$[Adam, LBFGS]$
Logistic Regression	C	$[100, 100000]$
KNN	n_neighbors	$[1, 10]$
Random Forest	num_estimators	$[10, 50, 100, 200]$
Linear Regression	Alpha	$[0.00001, 0.001]$
KE-GCN	Dimension	$\{16, 32, 64\}$
	Additional Layers	{0, 1, 2}
	Learning Rate	{0.001, 0.005, 0.01, 0.05, 0.1}
	Alpha	{0.3, 0.5}

Table 9: Hyperparameter search space for training downstream models. All hyperparameters except those of KE-GCN follow the semantics by scikit-learn.

A.3 EXPERIMENTAL RESULTS

		Average training epoch time in seconds		
		FB15K-237	YAGO3-10	WNRR
ComplEx	STD	04.92	097.88	2.32
	MTT	10.83	137.13	8.13
DistMult	STD	04.29	095.82	02.10
	MTT	09.27	222.37	10.98

Table 10: Average training epoch time in seconds over first 5 epochs of models with STD and MTT training. All tests were done with an 11th gen. Intel Core i7-11700K, 64GB of RAM and an NVIDIA GeForce RTX 3090.

			FB15K-237 *Entity Classification (Weighted F1 - higher is better)*			
			Type	Profession	Organization	Writer
MLP	ComplEx	STD	98.34	80.96	91.67	77.61
		MTT	98.64	78.23	92.98	66.50
	DistMult	STD	98.33	**82.29**	92.97	80.00
		MTT	98.69	80.04	94.36	82.06
Log. Reg.	ComplEx	STD	97.75	79.58	92.23	73.18
		MTT	98.57	80.83	92.98	66.81
	DistMult	STD	97.86	79.04	90.79	79.49
		MTT	98.34	77.98	91.93	67.50
KNN	ComplEx	STD	97.59	76.41	88.85	66.12
		MTT	98.61	80.27	90.01	65.60
	DistMult	STD	97.11	78.90	88.63	66.58
		MTT	98.58	80.22	89.01	72.85
Rand. For.	ComplEx	STD	98.35	73.28	84.44	76.17
		MTT	98.46	78.87	90.06	66.50
	DistMult	STD	98.39	74.51	88.62	**82.26**
		MTT	98.55	78.06	90.06	74.28
	KE-GCN		97.70	70.40	89.41	80.21

Table 11: Weighted F1 on test data of downstream classifiers (MLP, Logistic Regression, KNN and Random Forest) that use pre-trained KGE embeddings as input to solve entity classification tasks about entities in FB15K-237; and KE-GCN (Yu et al., 2021), a GCN that trains directly on the downstream data. Bold entries show best performance per task. Underlined entries show best performance (more than 1% difference) per task compared to same KGE model and same downstream model. Datasets are sorted by decreasing size of the training set from left to right.

			YAGO3-10							
			Entity Classification (Weighted F1 - higher is better)							
			Type	Player	Profession	Writer	Scientist	Organization	Artist	Waterbody
MLP	ComplEx	STD	99.35	91.94	75.47	61.30	53.09	80.74	45.18	68.92
		MTT	**99.62**	91.91	76.47	61.25	52.56	85.75	49.58	69.06
	DistMult	STD	99.35	**92.12**	77.45	59.14	52.05	83.04	54.16	64.67
		MTT	99.60	92.05	**79.08**	**64.24**	54.33	88.12	49.94	64.75
Log. Reg.	ComplEx	STD	70.49	90.52	72.97	59.13	50.97	73.33	48.26	69.79
		MTT	72.85	90.97	75.91	60.68	54.49	80.83	48.05	72.32
	Distmult	STD	71.02	90.76	74.63	57.75	52.45	77.11	52.31	64.20
		MTT	72.97	90.57	77.52	63.02	54.15	84.00	51.21	63.76
KNN	ComplEx	STD	87.34	89.37	47.23	41.00	31.58	60.05	24.13	61.37
		MTT	97.83	90.46	59.72	50.24	46.64	73.42	36.47	66.38
	DistMult	STD	88.69	89.27	48.40	38.44	32.99	51.45	25.13	**73.80**
		MTT	98.85	90.00	70.49	52.54	47.30	81.29	42.44	71.26
Rand. For.	ComplEx	STD	95.70	83.80	56.13	39.20	11.52	66.42	33.34	41.47
		MTT	99.47	86.33	72.83	45.91	26.54	83.87	38.27	54.30
	DistMult	STD	97.12	84.12	65.23	41.77	13.46	66.11	35.30	47.09
		MTT	99.13	86.39	76.12	48.12	23.68	85.37	38.05	56.55
	KE-GCN		99.29	89.32	70.97	57.11	**59.91**	**89.48**	45.27	46.93

Table 12: Weighted F1 on test data of downstream classifiers (MLP, Logistic Regression, KNN and Random Forest) that use pre-trained KGE embeddings as input to solve entity classification tasks about entities in YAGO3-10; and KE-GCN (Yu et al., 2021), a GCN that trains directly on the downstream data. Bold entries show best performance per task. Underlined entries show best performance (more than 1% difference) per task compared to same KGE model and same downstream model. Datasets are sorted by decreasing size of the training set from left to right.

			FB15K-237 *Regression (RSE - lower is better)*									
			Node Imp.	Birth Year	Latitude	Longitude	Person Height	Size Area	Population	Film Rel. Year	Date Founded	Film Rating
MLP	ComplEx	STD	0.85	0.95	**0.13**	**0.08**	0.69	0.15	0.83	0.17	0.48	**0.68**
		MTT	0.80	0.68	0.24	0.12	0.67	**0.11**	0.53	**0.13**	**0.46**	0.85
	Distmult	STD	**0.79**	**0.51**	0.16	0.10	**0.66**	0.25	0.86	0.15	0.55	0.76
		MTT	0.85	0.79	0.24	0.13	**0.66**	0.77	**0.51**	**0.13**	0.51	0.69
Lin. Reg.	ComplEx	STD	0.88	0.92	0.34	0.20	0.73	1.37	7.20	0.20	0.73	1.17
		MTT	0.84	0.78	0.44	0.21	0.67	1.31	10.84	0.17	0.69	1.13
	Distmult	STD	0.89	0.84	0.41	0.20	0.70	1.30	11.10	0.18	0.73	1.12
		MTT	0.85	0.89	0.46	0.21	0.69	1.17	11.29	0.17	0.69	1.06
KE-GCN			0.83	0.80	0.16	0.09	0.81	0.78	0.99	0.21	0.52	0.73

Table 13: Relative squared error (RSE) on test data of downstream models (MLP and Linear Regression) that use pre-trained KGE embeddings as input to solve regression tasks about entities in FB15K-237; and KE-GCN (Yu et al., 2021), a GCN that trains directly on the downstream data. Bold entries show best performance per task. Underlined entries show best performance (more than 0.01 difference) per task compared to same KGE model and same downstream model. Models with RSE above 1 are considered unsatisfactory. Datasets are sorted by decreasing size of the training set from left to right.

			YAGO3-10 *Regression (RSE - lower is better)*				
			Born on Date	Created on Date	Died on Date	Destroyed on Date	Happened on Date
MLP	ComplEx	STD	0.82	0.68	0.60	1.10	0.77
		MTT	0.52	0.68	0.66	1.53	0.50
	Distmult	STD	0.56	0.71	0.58	0.97	0.77
		MTT	0.43	0.68	0.45	1.32	0.64
Lin. Reg.	ComplEx	STD	0.82	0.72	0.75	0.97	0.64
		MTT	0.71	0.66	0.66	0.93	0.47
	Distmult	STD	0.81	0.72	0.72	0.88	0.57
		MTT	0.63	0.68	0.57	0.82	0.58
KE-GCN			**0.36**	**0.62**	**0.33**	**0.60**	**0.15**

Table 14: Relative squared error (RSE) on test data of downstream models (MLP and Linear Regression) that use pre-trained KGE embeddings as input to solve regression tasks about entities in YAGO3-10; and KE-GCN (Yu et al., 2021), a GCN that trains directly on the downstream data. Bold entries show best performance per task. Underlined entries show best performance (more than 0.01 difference) per task compared to same KGE model and same downstream model. Models with RSE above 1 are considered unsatisfactory. Datasets are sorted by decreasing size of the training set from left to right.