Beyond Link Prediction: On Pre-Training Knowledge Graph Embeddings

Daniel Ruffinelli and **Rainer Gemulla**
University of Mannheim
Germany
{druffinelli, rgemulla}@uni-mannheim.de

Abstract

Knowledge graph embeddings (KGEs) provide low-dimensional representations of the entities and relations in a knowledge graph (KG) in order to reason about the KG and to inject structured knowledge into various downstream applications. Most prior work, however, focuses almost exclusively on training and evaluating KGE models for the task of link prediction. In this work, we explore KGE models as general-purpose representations of KGs and study their suitability (i) for more generally capturing properties of the KG and (ii) for downstream tasks such as entity classification and regression. For (i), we designed a new set of graph-structure prediction tasks to assess whether models capture different structures in the graph. For (ii), we investigate whether models provide useful features for a variety of downstream tasks. We found that strong link prediction performance was neither an indication that models generally capture patterns in the graph, nor that they were more useful in downstream tasks. As a result, we included our proposed graph-structure prediction tasks as additional training objectives and found that models trained with this multi-task approach generally, but not always, performed better at both graph-structure prediction and downstream tasks. However, the most suitable choice of pre-training tasks varies across KGE models and types of downstream tasks, suggesting opportunities for more research into the relation between pre-training KGE models and their usability on downstream applications.

1 Introduction

Knowledge graph embeddings (KGE) provide representations of the entities and relations in a knowledge graph (KG). Although a large number of KGE models have been proposed, e.g. Ge et al. (2023); Xiao et al. (2022); Bai et al. (2022), most prior work focuses on the task of link prediction, i.e., answering questions such as *(Austin, capitalOf, ?)* by reasoning over an incomplete KG. In addition to link prediction, it is often argued that KGEs can provide representations that capture semantic properties of the entities (Wang et al., 2022a; Ji et al., 2021; Wang et al., 2017; Nickel et al., 2015; Bordes et al., 2013, 2011) and, indeed, pre-trained KGE models have been used to inject structured knowledge into recommender systems (El-Kishky et al., 2022; Wang et al., 2018), question answering systems (Ilyas et al., 2022) and other types of downstream applications (Ji et al., 2021).

Despite their use as KG representations in downstream applications, the question of whether pre-trained KGE models are generally useful representations of KGs—i.e. representations that are useful beyond the link prediction task—remains largely unexplored. Specifically, it is not well-understood how different pre-training settings affect these representations. This stands in contrast with representation learning of natural language, where representations are intrinsically tested for known linguistic properties (Mikolov et al., 2013) and extrinsically on their usability in downstream applications (Devlin et al., 2019; Radford et al., 2018), and where different pre-training settings are known to improve performance (Raffel et al., 2020; Liu et al., 2019).

In this work, we study the suitability of KGE models as general-purpose KG representations. First, we intrinsically assess whether KGE models capture known properties of the graph, by evaluating their performance on basic graph-structure prediction tasks. We focus on new tasks that are similar to link prediction, but that test different forms of structural knowledge, such as predicting the relation of a triple (e.g., the relationship between *Austin* and *Texas*), the domain and range of a relation (e.g., whether *Austin* is a capital), and the entity and relation neighborhood of an entity (e.g., which entities are related to *Austin*). We found that commonly trained KGE models often performed poorly on such tasks, challenging the intuition that KGE models preserve the structure of a KG.

Proceedings of the 9th Workshop on Representation Learning for NLP (RepL4NLP-2024), pages 136–162
August 15, 2024 ©2024 Association for Computational Linguistics

Second, we extrinsically evaluate whether KGE models are useful pre-trained representations for node-level downstream tasks such as entity classification (e.g., the profession of a person) or regression (e.g., the rating of a movie). We conduct an empirical study using 35 downstream tasks on three different KGs. We found that KGE models often perform decent on these tasks, almost always exceeding the performance of graph neural networks that train directly on the downstream task, such as KE-GCN (Yu et al., 2021). However, the KGE models with best downstream task performance were often not the best-performing models for link prediction. For example, the basic TransE model (Bordes et al., 2013) can be superior to KGE models with stronger performance on link prediction, such as ComplEx (Trouillon et al., 2016) or RotatE (Sun et al., 2019). This suggests that good link prediction performance is not necessarily indicative of good downstream task performance.

Both of these findings suggest that the focus on link prediction tasks is too narrow for pre-training KGE models, i.e., to provide generally useful representations of a KG. We thus included the graph-structure prediction tasks discussed above as additional training objectives. The resulting multi-task KGE models had significantly better overall performance for graph-structure prediction tasks, suggesting that the learned representations capture more information about the graph, at the cost of a small drop in link prediction performance.

Perhaps more importantly, when using pre-trained KGEs in downstream tasks, we found that multi-task training often (but not always) improved downstream performance, especially as data becomes scarce. In fact, excluding the link prediction task during pre-training resulted in better downstream performance more often than not. However, capturing more information about the graph did not directly translate to better downstream performance, as the best performing models in downstream applications were often those that were not pre-trained using all possible tasks. In general, the best choice of pre-training tasks depends on the dataset, KGE model, and type of downstream task, suggesting opportunities for more research to better understand how to pre-train KGE models so they provide generally useful KG representations. We provide all of our resources[1] to promote future work in this direction.

[1]Available at https://github.com/uma-pi1/kge-pretraining.

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. For a more comprehensive discussion, please see surveys from 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)*. In the KGE literature, the *link prediction* task is defined as predicting the subject or object to questions of the form *(?, capitalOf, Texas)* and *(Austin, capitalOf, ?)*, resp.

KGE models. KGE models represent each entity and each relation of a KG with a low-dimensional embedding. Each model has a *scoring function* $s : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$ that maps each possible triple to a real-valued score. Intuitively, high scores indicate plausible triples, low scores implausible triples. For example, TransE (Bordes et al., 2013) is a translation-based model with $s(i, k, j) = -\|\boldsymbol{e}_i + \boldsymbol{r}_k - \boldsymbol{e}_j\|$, where $\boldsymbol{e}_i \in \mathbb{R}^d$ and $\boldsymbol{r}_k \in \mathbb{R}^d$ are entity and relation embeddings, resp. Scoring functions can be more involved, e.g., based on transformers (Chen et al., 2021a).

Standard training. KGE models are commonly trained on the link prediction task. For each training triple (s, p, o), models are trained such that score $s(s, p, o)$ is high (a known positive) but score $s(s, p, o')$ is low for (pseudo-)negative triples (s, p, o'), where $o' \neq o \in \mathcal{E}$; similarly for subjects $s' \in \mathcal{E}$ with negative triple (s', p, o). Different training objectives exist, all of which follow this approach, but otherwise differ in other hyperparameters; for details, see Ali et al. (2021).

Standard evaluation. The most common evaluation protocol is *entity ranking* (ER), and it is also based on link prediction. Given test triple (s, p, o), models answer the link prediction queries $(s, p, ?)$ and $(?, p, o)$ by ranking all possible answers to each query by their scores, after filtering other known answers. Metrics such as mean reciprocal rank (MRR) and Hits@K are then computed based on the rank of the answers s and o, resp. As an evaluation method, entity ranking has been questioned in prior work (Zhou et al., 2022; Tiwari et al., 2021; Safavi and Koutra, 2020; Wang et al., 2019). In this work, we focus mostly on other evaluation tasks.

Other training approaches. Nickel et al. (2011) and Li et al. (2021) trained on the *reconstruction* task, which aims at reconstructing the training set using cost functions such as $\sum_{t \in \mathcal{G}_{\text{train}}} \|I[(t)] - s(s,p,o)\|_2^2$, for training set $\mathcal{G}_{\text{train}}$ and where $I[\cdot]$ is a 0/1 indicator. We do not consider such methods due to excessive training costs. Chen et al. (2021b) augmented the link prediction task with *relation prediction* during training (but not evaluation). We extend this work by including additional pre-training tasks and by focusing on graph-structure prediction and downstream task performance instead.

Other evaluation approaches. Some works evaluate KGE models using *triple classification* (Socher et al., 2013; Lin et al., 2015; Wang et al., 2022b). We do not consider this task because performance estimates are typically overly optimistic and misleading unless hard negatives are used (Safavi and 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 evaluation task in this work. There is also work on probing KGE models (Meilicke et al., 2018; Allen et al., 2021; Rim et al., 2021), which focus on link prediction performance across different types of relations, e.g. symmetric. In contrast, we focus on studying whether models provide useful representations, i.e. we focus on embedding quality, not just on link prediction performance. In addition, pre-trained KGE models have been used as components in language models (He et al., 2020; Zhang et al., 2019), visual models (Baier et al., 2017), recommender systems (El-Kishky et al., 2022; Wang et al., 2018), or question answering systems (Ilyas et al., 2022). Similarly, some studies have evaluated pre-trained KGE models for entity classification or regression tasks (Pezeshkpour et al., 2018; Jain et al., 2021), as we do. We extend this line of work with a larger set of downstream tasks, and by being the first (to our knowledge) to study the impact of different pre-training methods on downstream task performance.

3 Graph Structure Prediction

In this section, we describe the new graph-structure tasks used in our study. Specifically, how we use them to test whether KGE models preserve known properties in a KG, and how we adapted KGEs to efficiently train on these tasks.

3.1 Graph-Structure Tasks

An example and summary of the graph-structure tasks that we use in our study is given in Table 1. We describe 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 $(s, ?, *)$) and where it occurs as object (denoted $(*, ?, o)$).

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 illustrate these new tasks in Figure 2 in Appendix A.

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 $(?, p, ?)$) or reconstruction (Nickel et al., 2011) (denoted $(?, ?, ?)$). In our experimental study in Sec. 4, we found that the exclusion of some of the above pre-training tasks (e.g., LP) often improves downstream task performance. The optimal choice of tasks depends on dataset, KGE model, and downstream task, however. We leave the exploration of task selection as well as on exploring additional pre-training tasks to future work.

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.

3.2 Multi-Task Ranking

To intrinsically evaluate whether KGE models preserve properties that are known to exist in a KG, we use the set of graph-structure prediction tasks described above to generalize the entity ranking (ER) protocol for link prediction (see Sec. 2) to a multi-task ranking (MTR) protocol. Intuitively, for each of the nine queries (LP/REL/DOM/NBE/NBR for both subject and object targets), we construct a query from each test triple, obtain a ranking of the prediction targets that do not already occur in the training/validation/test data (filtered setting), and use metrics such as MRR or Hits@K. The final metric is the micro-average over all nine queries.

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 $p' \in \mathcal{R}$ such that $(s, p', o) \notin \mathcal{G}_{\text{train}}$ in descending order of their scores $s(s, p', o)$. 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 data for evaluation. 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, p', o'), where $p' \in \mathcal{R}$ 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). Generally, we make use of an *extended score function* that accepts wildcards (described in Algorithm 2 in Appendix A). The approach just described corresponds to using $s(s, p', *) = \max_{o' \in \mathcal{E}} s(s, p', o')$, i.e, the score of a relation p' 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/validation/test data during model evaluation (as done in the literature), we remove all relations p' such that $(s, p', o') \in \mathcal{G}_{\text{train}}$ for some $o' \in \mathcal{E}$; i.e., we remove all prediction targets that are already implied by the filtering splits. 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 $O(|\mathcal{E}||\mathcal{R}|)$. Similarly, predicting target relations costs $O(|\mathcal{R}||\mathcal{E}|)$ and $O(|\mathcal{R}|)$ with and without wildcards, respectively. We discuss in the next section how to reduce the additional cost of using wildcards to $O(|\mathcal{E}|)$ or $O(|\mathcal{R}|)$.

3.3 Multi-Task Training

We generalize standard KGE model training to all of the graph-structure prediction tasks, called multi-task training (MTT). Our goal is to be able to train KGE models on multiple tasks simultaneously, while 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 linear combination of the task-specific costs (and additional regularization terms), where the weights are hyperparameters (costs increase only linearly in the number of tasks, see Table 12). We formalize the MTT training objective in Appendix A.

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 our proposed tasks involving wildcards, we proceed differently. Instead of performing some form of (costly) score aggregation during training, we "convert" these tasks with wildcards into tasks without wildcards. To do so, we make use of three virtual *wildcard objects*—one for subjects (any_S), one for relations (any_R), and one for objects (any_O)—and learn embeddings for these objects. During training, we conceptually replace wildcards by their 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 convert the NBR task into a REL task. We also use these wildcard embeddings during inference in the same way; e.g., we set $s(s, p', *) = s(s, p', any_O)$. Instead of performing score aggregation, the model thus directly learns extended scores at the same cost (per task) as standard link prediction, i.e. $O(|\mathcal{E}|)$ for target entities, and $O(|\mathcal{R}|)$ for target relations.

4 Experimental Study

To our knowledge, no prior work has studied the impact that different training objectives have on KG embedding quality, despite this being common practice, e.g. in language models (Raffel et al., 2020; Liu et al., 2019). We conducted a large experimental study with the following goals: (i) to assess whether KGE models capture various properties of a KG by intrinsically evaluating their performance on new graph-structure prediction tasks, (ii) to determine whether (and by how much) KGEs improve their performance on these tasks when simultaneously trained for them, and (iii) to assess the impact that different pre-training approaches have on downstream tasks by extrinsically evaluating pre-trained KGE models. We briefly describe our experimental setup here, for details, see Sec. B.1.

Pre-Training Setup. For training and evaluating KGEs, we closely follow Ruffinelli et al. (2020). We implemented everything in LibKGE (Broscheit et al., 2020), used four benchmark datasets commonly used in recent work (Ge et al., 2023; Xiao et al., 2022; Zhu et al., 2022), all models were trained under the same conditions (as much as possible) and tuned with a large hyperparameter space using random search. For MTT training, we used all tasks in Table 1 (LP, REL, DOM, NBE, NBR), and evaluated models on each of these tasks using filtered MRR, and aggregated these metrics into multi-task ranking MRR (MTR). We selected standard (STD) models with LP and MTT models with both the LP and MTR task, all on validation.

Choice of KGE models. We focused on models that provide entity representations, so we may test their quality in downstream tasks, as done in the industry (El-Kishky et al., 2022; Ilyas et al., 2022). We chose four popular models: TransE (Bordes et al., 2013), DistMult (Yang et al., 2015), RotatE (Sun et al., 2019) and ComplEx (Trouillon et al., 2016). These are not the latest KGE models, but they are common baselines in recent work (Ge et al., 2023; Gui et al., 2022; Chao et al., 2021), and are common choices for pre-trained models (Zhu et al., 2022; El-Kishky et al., 2022; Ilyas et al., 2022). They can also reach SOTA performance with reasonable embedding sizes (Ruffinelli et al., 2020), allowing us to scale our study, and with larger embedding size (Lacroix et al., 2018) and additional training objectives (Chen et al., 2021b), ComplEx outperforms more involved models, e.g. the transformer-based HittER model (Chen et al., 2021a). Some recent models achieve better performance on link prediction, but focus exclusively on that task and do not directly provide entity representations for downstream tasks, e.g. HittER (Chen et al., 2021a) and NBFNet (Zhu et al., 2021).

Downstream Tasks Setup. To extrinsically evaluate our pre-trained models, we collected/created data for 35 downstream tasks on FB15K-237, YAGO3-10 and WIKI5M (examples in Table 2). For downstream models, we used scikit-learn (Pedregosa et al., 2011) models that use only entity embeddings from pre-trained KGE models as input features. We used multilayer perceptrons (MLP), logistic regression, KNN, and random forests for classification, and linear regression and MLP for regression, and treated the choice of downstream model as a hyperparameter. For entity classification, we report *weighted F1* (as Jain et al. (2021)) aggregated across all classification tasks (denoted EC). For regression, we chose relative squared error (RSE) (defined in Sec. B.2), as it allows meaningful averaging across different regression tasks (denoted REG, lower values are better). We report mean and standard deviation over 3 training runs. As baseline, we included KE-GCN (Yu et al., 2021), a state-of-the-art GNN for entity classification. In contrast to KGEs, this model is directly trained on the downstream task (i.e., no pre-training) and uses the KG for inference. Tuning, training, evaluation was done as with KGEs and downstream models.

	Benchmark	Name	Train Size
EC	FB15K-237	Entity Type	6 719
		Profession	2 537
	YAGO3-10	Entity Type	69 592
		Player Type	33 928
REG	FB15K-237	Birth Year	3 538
		Latitude	2 568
	YAGO3-10	Born on Year	60 409
		Created on Year	23 896

Table 2: Some datasets for entity classification (EC) and regression (REG) downstream tasks used to evaluate pre-trained KGEs. See Appendix B for a complete list.

4.1 Results on Graph-Structure Prediction

In Table 3, we report test MRR of graph-structure prediction tasks using standard (STD) and MTT training. We report both training approaches with LP for model selection, as we found this to often produce better downstream performance with MTT (such "cross-over" selection was not useful for STD training, see Table 11). We report MTT with MTR for model selection, and results for WNRR and WIKI5M in Appendix C (Tables 13 and 14, resp.)

Every model is able to capture more information about the KG when trained on multiple tasks simultaneously. For a given model, the improvement can be large, often by a factor of 2x and up to 10x depending on model, task and dataset (or even larger when MTT is used with MTR for model selection, see Table 13). This suggests that, unless trained for it, **KGE models often fail to capture graph structure beyond what is necessary to perform link prediction**. MTT models had slightly lower performance on LP, but the decrease was often small and outweighed by significantly improved performance on other tasks. Moreover, the best models for LP with STD training are often far outperformed on other tasks by other STD models with lower LP performance, suggesting that good LP performance is not indicative of general KG representation. For example, the best LP performance on FB15K-237 is ComplEx STD, but RotatE STD outperforms it considerably on REL and TransE STD on DOM. Similar observations also hold for the best models on MTR, but the compromise on other tasks is significanly smaller. **In general, MTT improved significantly on STD for graph structure prediction and can thus be used so models simultaneously learn more properties in a KG.** Note that our performance on LP, even with MTT, is comparable and sometimes better than recently published works that use comparable embedding sizes (Yang et al., 2022; Dong et al., 2022), or even larger embedding sizes (Gui et al., 2022).

Discussion. From a training perspective, these results are not surprising, as STD training only focuses on the LP task. However, these results do challenge studies that describe KGEs as generally capturing semantic properties of a KG (Ge et al., 2023; Xiao et al., 2022; Gui et al., 2022; Nickel et al., 2015; Bordes et al., 2013), which were likely inspired by work on capturing properties of words despite not directly training for it (Mikolov et al., 2013; Bordes et al., 2013). In addition, some of the new tasks are similar enough to link prediction that the results are indeed unexpected. For example, a good link prediction model may be able to answer *(Austin, capital of, ?)* and *(?, capital of, Texas)*, yet it may not be able to predict that *capital of* is a relation connected to Austin and/or Texas (NBR). Similar arguments can be made for other tasks. Generally, if the goal is *purely* link prediction, STD training is more suitable. But we show empirically in Sec. 4.3 that the choice of training objective has an impact on the learned representations and that including the LP task during pre-training is often detrimental for downstream performance.

4.2 Results on Downstream Tasks

Table 3 also reports downstream performance using models pre-trained with STD and MTT. The EC column reports mean weighted F1 across all classification datasets, and REG column reports mean RSE across all regression datasets. We report on individual downstream tasks in Appendix C.

The best overall downstream task performance across all KGE models was achieved by MTT in all cases, and often combined with LP for model selection. While the margin compared to STD was sometimes small (e.g., EC on YAGO3-10) and sometimes large (e.g., REG on FB1K-237), training only for link prediction (STD) resulted in worse average downstream performance compared to MTT more often than not (especially when considering MTT with MTR selection, see Table 13). Nevertheless, for a given KGE model, STD training did perform better at times. In addition, we found that the best models for both LP and MTR are often not the best models in downstream applications. Perhaps more importantly, **the best downstream performance often comes from models with weaker LP**

		Train.	Sel.	Graph-structure prediction (↑)						Downstream tasks	
				LP	REL	DOM	NBE	NBR	MTR	EC (↑)	REG (↓)
FB15K-237	ComplEx	STD	LP	**.346**	.805	.423	.016	.046	.274	.844±.008	.447±.051
		MTT	LP	.336	.964	.557	**.195**	.794	.525	.858±.005	**.394±.057**
	DistMult	STD	LP	.342	.388	.045	.009	.036	.139	.873±.009	.551±.062
		MTT	LP	.334	.944	.557	.139	.818	.516	.865±.005	.472±.026
	RotatE	STD	LP	.312	.919	.581	.051	.136	.342	.868±.003	.797±.286
		MTT	LP	.319	**.965**	.758	.136	**.880**	**.572**	**.890±.003**	.573±.062
	TransE	STD	LP	.330	.900	.624	.038	.054	.332	.873±.015	.742±.287
		MTT	LP	.317	.963	.653	.152	.855	.547	.855±.007	.795±.257
	KE-GCN			–	–	–	–	–	–	.829±.526	.501±.001
YAGO3-10	ComplEx	STD	LP	**.550**	.900	.120	.215	.517	.411	.712±.008	.589±.023
		MTT	LP	.538	.941	.836	.591	**.978**	.759	.729±.005	.466±.017
	DistMult	STD	LP	.539	.881	.010	.327	.613	.429	.734±.003	.519±.019
		MTT	LP	.536	**.945**	**.861**	.581	**.978**	**.762**	**.746±.006**	.472±.029
	RotatE	STD	LP	.436	.809	.046	.400	.656	.432	.701±.002	.696±.018
		MTT	LP	.509	.918	.011	**.609**	.366	.434	.708±.002	.659±.059
	TransE	STD	LP	.504	.860	.178	.287	.175	.349	.742±.002	.447±.036
		MTT	LP	.462	.940	.037	.476	.338	.396	.723±.004	.441±.029
	KE-GCN			–	–	–	–	–	–	.700±.223	**.398±.008**

Table 3: Performance on test data of graph-structure prediction and downstream tasks. Bold entries show performance per task and dataset. Underlined entries show best performance between STD and MTT.

performance (e.g. RotatE on EC in FB15K-237) or weaker MTR performance (e.g. ComplEx on REG in FB15K-237). This is more clearly visible in Table 15 in Appendix C. **This is problematic, as it suggests that MTR and, perhaps more importantly, LP are often inadequate tasks to guide the choice of the more suitable KGE models for downstream applications.** Ultimately, we conclude that the choice of pre-training objective clearly has an impact on downstream performance, but it is unclear how to make this choice in order to maximize downstream performance.

Downstream Baseline Performance. Compared to KE-GCN, KGE models clearly outperform KE-GCN almost every time (except in REG on YAGO3-10) These results suggest that the information captured by KGE models during pre-training is useful for simple downstream models to be competitive with, and even outperform, more involved downstream models that train directly on the task.

4.3 Impact of Pre-Training Task Selection

Table 4 summarizes our results about the impact that pre-training task selection has on downstream tasks. To keep computational costs feasible, we focused on FB15K-237. We explored performance using MTT without either the LP, REL, DOM, NBE, or NBR pre-training task, and without LP+REL or without DOM+NBR. We report models and sets of tasks relevant for our discussion. For details, see Table 16.

Impact on Graph-Structure Tasks. We found that for graph-structure predictions, excluding a task generally led to lower performance on that task, as expected. It may also, however, lead to a boost in performance on other tasks. For example, RotatE performs best on DOM when the standard LP task is excluded from the training objective.

Impact on Downstream Tasks. For downstream performance, the choice of pre-training tasks has a significant impact, but good choices differ between KGE models and the type of downstream task. For example, compared to full MTT training, using a subset of tasks led to improvements almost every time. Surprisingly, excluding the LP task during pre-training improved downstream performance half of the time compared to STD and full MTT training, suggesting that pre-training with LP can often be detrimental to downstream performance.

	Train.	*Sel.*	*Graph-structure prediction* (↑)						*Downstream tasks*	
			LP	REL	DOM	NBE	NBR	MTR	EC (↑)	REG (↓)
ComplEx	STD	LP	**.346**	.805	.423	.016	.046	.274	.844±.008	.447±.051
	MTT	MTR	.331	**.977**	.773	**.210**	.925	**.606**	.843±.002	**.412±.037**
	w/o LP	MTR	.154	.972	.831	.200	.932	.579	.870±.002	.512±.044
	w/o NBE	MTR	.315	.958	**.850**	.005	.936	.575	.856±.002	.562±.038
	w/o LP+REL	MTR	.001	.009	.843	.177	**.939**	.436	.849±.011	.542±.054
RotatE	STD	LP	.312	.919	.581	.051	.136	.342	.868±.003	.797±.286
	MTT	MTR	.314	.964	.813	.160	.922	.598	.847±.001	.704±.060
	w/o LP	MTR	.204	.914	.842	.126	.928	.568	.874±.000	.661±.043
	w/o DOM	MTR	.319	.965	.661	.170	.883	.559	**.898±.001**	.593±.078
	w/o NBR	MTR	.318	.964	.710	.168	.673	.522	.863±.007	.552±.035

Table 4: Performance on test data of graph-structure and downstream tasks for FB15K-237 of STD and various MTT objectives. Objectives such as w/o LP are MTT objectives with all tasks in Table 1 except one, e.g. LP.

4.4 Data Efficiency Tests

To see whether KGE models that capture more information during pre-training are more beneficial as downstream data becomes scarce, we tested models in a few-shot scenario. For classification, we sampled n positive and n negative examples per class, where $n \in \{3, 5, 10\}$. Figure 1 shows the results for the YAGO3-10 classification tasks (higher is better). We found that as less data becomes available, the average performance of STD models becomes considerably lower compared to pre-trained MTT models, except TransE, where performance difference is not as significant. We observed the same pattern in FB15K-237 (see Fig. 3).

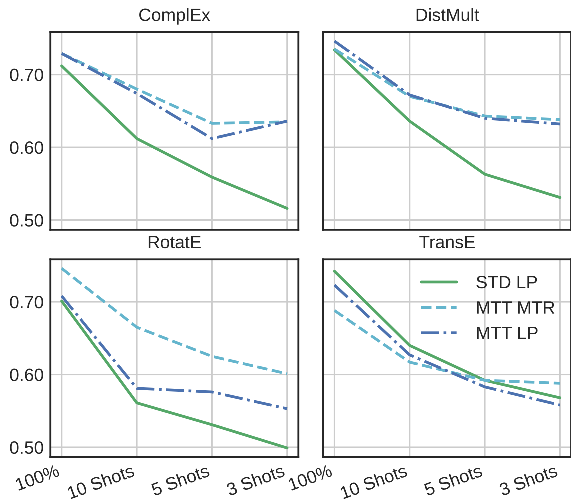


Figure 1: Few-shot performance of entity classification tasks for YAGO3-10 (higher is better). Each n-shot training set consists of n sampled positive and negative examples for each class.

The few-shot scenario applied to regression tasks produced unsatisfactory models almost every time. We thus constructed a different scenario with scarce training data. We randomly sampled $n\%$ of the training set, where again $n \in \{3, 5, 10\}$ (see Figures 5 and 7 in Appendix C, lower is better). For most models, the trend observed with a complete training set is mostly maintained, suggesting that pre-trained MTT models are not always more beneficial with less training data. Still, at no point do models pre-trained with STD become a better choice. **Overall, although not every time, we observed the clear trend that MTT models are more data efficient than STD models, especially for the classification tasks in our tests.**

5 Conclusion

To explore KGE models as general-purpose representions of KGs, we designed a new set of graph-structure prediction tasks for intrinsic evaluation. We found that standard KGE models are not good at predicting simple structures in the graph, challenging the intuition that these models generally capture properties in a KG. In addition, we extrinsically evaluated pre-trained KGE models on several entity-level downstream tasks. We found that link prediction was not indicative of good downstream performance, and that multi-task pre-training was generally better for downstream tasks, often when excluding link prediction during pre-training. However, the best choice of pre-training tasks depends on both KGE model and downstream task, suggesting more research is needed into pre-training KGEs to obtain generally-useful KG representations.

6 Limitations

In our study, we explored the use of different self-supervised tasks for training KGE models. However, as a first step, we tested models using only a limited set of simple pre-training tasks. Aside from the link prediction task that is almost exclusively used in the literature, we also included the relation prediction task (as already done by Chen et al. (2021b)), as well a new set of tasks that we proposed (see Table 1). However, other pre-training tasks are possible and should be explored, e.g. self-supervised tasks such as predicting the n-hop neighborhood of an entity, or even objectives that resemble downstream tasks, such as predicting the size of a neighborhood. It is also possible to combine such objectives with supervised training objectives during training, as already done in previous work (Aribandi et al., 2022).

Another limitation of our work is the small variety in types of downstream tasks. While we focused on entity-level classification and regression tasks, the impact of different pre-training approaches on more involved downstream applications should be explored. Some examples would be testing the use of pre-trained KGEs in recommender systems as in El-Kishky et al. (2022), or question answering systems as in Ilyas et al. (2022).

Finally, while we take the first steps into exploring alternatives for pre-training KGE models, our work does not find a concrete solution to the problem, which may indeed by challenging, as models need to encode hundreds or thousands of different, and often uncorrelated, relation types between entities. We observed the impact that different pre-training tasks have both on capturing properties of a graph, as well as in downstream application performance. In particular, we found that training with more tasks is beneficial for capturing more properties of a KG, and often for improving downstream performance. However, we have no concrete suggestions on how to pre-train KGE models more generally. Different pre-training tasks should be explored in the context of different types of downstream tasks, so that we may better understand the relation between pre-training KGEs and their quality as KG representations in downstream applications. As part of our work, we provide all of our code as well as our collection of downstream task data, to create opportunities for future research into this unexplored question.

References

Mehdi Ali, Max Berrendorf, Charles Tapley Hoyt, Laurent Vermue, Sahand Sharifzadeh, Volker Tresp, and Jens Lehmann. 2021. PyKEEN 1.0: A python library for training and evaluating knowledge graph embeddings. *J. Mach. Learn. Res.*

Carl Allen, Ivana Balazevic, and Timothy Hospedales. 2021. Interpreting knowlege graph relation representation from word embeddings. In *Ninth International Conference on Learning Representations 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. 2022. ExT5: Towards extreme multi-task scaling for transfer learning. In *International Conference on Learning Representations*.

Jiaxin Bai, Zihao Wang, Hongming Zhang, and Yangqiu Song. 2022. Query2particles: Knowledge graph reasoning with particle embeddings. *Findings of the Association for Computational Linguistics: NAACL 2022-Findings*.

Stephan Baier, Yunpu Ma, and Volker Tresp. 2017. Improving visual relationship detection using semantic modeling of scene descriptions. In *ISWC*.

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

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

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Proceedings of the AAAI conference on artificial intelligence*.

Samuel Broscheit, Daniel Ruffinelli, Adrian Kochsiek, Patrick Betz, and Rainer Gemulla. 2020. LibKGE-a knowledge graph embedding library for reproducible research. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

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

Linlin Chao, Jianshan He, Taifeng Wang, and Wei Chu. 2021. Pairre: Knowledge graph embeddings via paired relation vectors. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4360–4369.

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

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

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.

Yao Dong, Lei Wang, Ji Xiang, Xiaobo Guo, and Yuqiang Xie. 2022. Rotatect: Knowledge graph embedding by rotation and coordinate transformation in complex space. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4918–4932.

Ahmed El-Kishky, Thomas Markovich, Serim Park, Chetan Verma, Baekjin Kim, Ramy Eskander, Yury Malkov, Frank Portman, Sofía Samaniego, Ying Xiao, and Aria Haghighi. 2022. TwHIN: Embedding the twitter heterogeneous information network for personalized recommendation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*.

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

Xiou Ge, Yun Cheng Wang, Bin Wang, and C-C Jay Kuo. 2023. Compounding geometric operations for knowledge graph completion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*.

Xiangyu Gui, Feng Zhao, Langjunqing Jin, and Hai Jin. 2022. Optice: A coherence theory-based model for link prediction. In *Proceedings of the 29th International Conference on Computational Linguistics*.

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

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

Ihab F Ilyas, Theodoros Rekatsinas, Vishnu Konda, Jeffrey Pound, Xiaoguang Qi, and Mohamed Soliman. 2022. Saga: A platform for continuous construction and serving of knowledge at scale.

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

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

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. In *International Conference on Machine Learning*.

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

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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

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

Tomáš Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*.

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

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

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. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*.

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

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*.

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

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

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

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

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

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

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

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

Theo Van Veen. 2019. Wikidata. *Information technology and libraries*.

Feiyang Wang, Zhongbao Zhang, Li Sun, Junda Ye, and Yang Yan. 2022a. Dirie: knowledge graph embedding with dirichlet distribution. In *Proceedings of the ACM Web Conference 2022*, pages 3082–3091.

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

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

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*.

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

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

Huiru Xiao, Xin Liu, Yangqiu Song, Ginny Y. Wong, and Simon See. 2022. Complex hyperbolic knowledge graph embeddings with fast fourier transform. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*.

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

Jinfa Yang, Xianghua Ying, Yongjie Shi, Xin Tong, Ruibin Wang, Taiyan Chen, and Bowei Xing. 2022. Knowledge graph embedding by adaptive limit scoring loss using dynamic weighting strategy. In *Findings of the Association for Computational Linguistics: ACL 2022*.

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

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

Ying Zhou, Xuanang Chen, Ben He, Zheng Ye, and Le Sun. 2022. 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*.

Yushan Zhu, Wen Zhang, Mingyang Chen, Hui Chen, Xu Cheng, Wei Zhang, and Huajun Chen. 2022. Dualde: Dually distilling knowledge graph embedding for faster and cheaper reasoning. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*.

Zhaocheng Zhu, Zuobai Zhang, Louis-Pascal Xhonneux, and Jian Tang. 2021. Neural bellman-ford networks: A general graph neural network framework for link prediction.

A Multi-Task Training and Evaluation

We illustrate how training objectives are constructed using more than one training task, i.e. query. To this end, we define both the standard training objective (STD) based on link prediction and our proposed multi-task objective (MTT) as follows. Let $T_o = \{(t,l)\}$ be the set of relevant positive and negative examples t and corresponding label l induced by the link prediction query $(s,p,?)$ in a given training set. Let T_s be the analogous set of examples for query $(?,p,o)$. For some loss function L, the STD training approach optimizes the following objective function (we omit the penalty term for brevity):

$$f(\theta) = \operatorname*{argmin}_{\theta} \left(\frac{1}{|T_s|} \sum_{(t,l) \in T_s} L(s(t),l) + \frac{1}{|T_o|} \sum_{(t,l) \in T_o} L(s(t),l) \right) \quad (1)$$

where s is a KGE score function parameterized by model parameters θ. We generalize this objective to define the following multi-task training (MTT) objective:

$$f(\theta) = \operatorname*{argmin}_{\theta} \frac{1}{N} \sum_{T_i \in \mathcal{T}} \sum_{(t,l) \in T_i} \lambda_i L(s(t),l) \quad (2)$$

where $\mathcal{T} = \{T_1, T_2, \dots\}$ is a superset of training examples for queries T_i, N is the sum of the cardinalities of each T_i and λ_i a hyperparameter that controls the impact of query i in the training objective. Chen et al. (2021b) have already followed this training approach by adding the relation prediction task, i.e. $(i,?,j)$ to Eq. 1. They set $\lambda_s = \lambda_o = 1$ and tune λ_r. Note that Equations 1 and 2 do not describe the exact training objective with some loss functions, e.g. some losses require a positive and corresponding set of negatives to compute a loss value. However, the MTT objective can be reformulated for every loss function commonly used to train KGE models. We provide such a general description of the MTT approach in Algorithm 1. As with loss functions, the MTT approach is agnostic to the choice of model and training task.

W.r.t. to evaluation, Algorithm 2 describes the extended score function described in Section 3.2.

B Experimental Settings

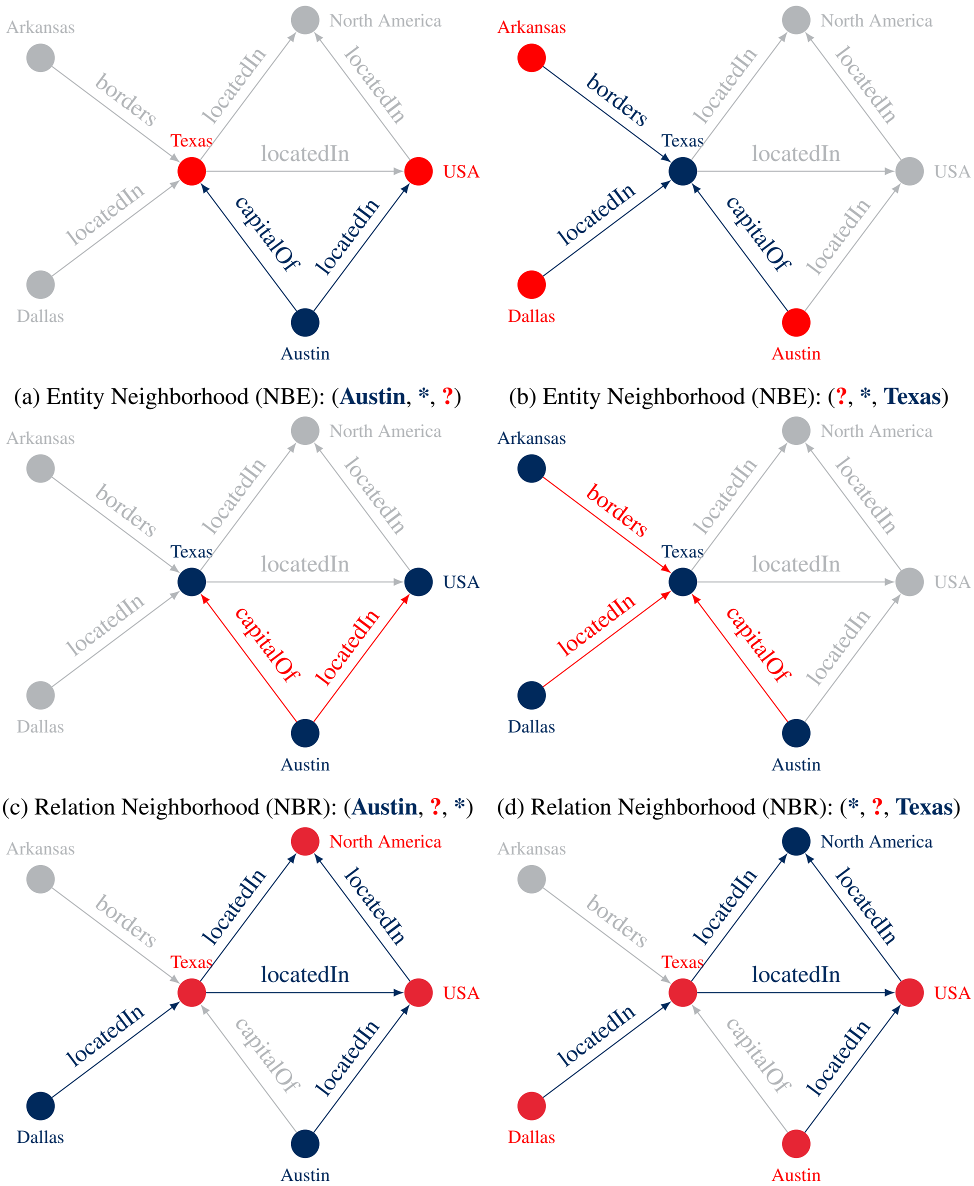
B.1 Experimental Setup: Pre-Training KGEs

Knowledge graphs. We chose four commonly used benchmark datasets for evaluating KGE models: FB15K-237 (Toutanova and Chen, 2015), WNRR (Dettmers et al., 2018), YAGO3-10 (Mahdisoltani et al., 2014), and WIKIDATA5M (WIKI5M) (Wang et al., 2021). 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 and WIKI5M are considerably larger. Dataset statistics are summarized in Table 5.

KGE training. We used LibKGE (Broscheit et al., 2020) for *STD training* (LP only) as a baseline and added MTT/MTR model training/evaluation. 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* training with FB15K-237 and WNRR (to achieve good results) and *NegSamp* with YAGO3-10 and WIKI5M to scale to these larger datasets. Models were selected w.r.t. performance (MRR) on the validation data. We selected STD models with LP task and MTT models with the MTR task. For MTT training, we used all tasks in Table 1.

KGE evaluation. As with training, we evaluate KGE models with respect to each of the five graph-structure prediction tasks in Table 1 (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.



(a) Entity Neighborhood (NBE): (**Austin**, *, **?**)
(b) Entity Neighborhood (NBE): (**?**, *, **Texas**)
(c) Relation Neighborhood (NBR): (**Austin**, **?**, *)
(d) Relation Neighborhood (NBR): (*, **?**, **Texas**)
(e) Domain (DOM): (*, **locatedIn**, **?**)
(f) Domain (DOM): (**?**, **locatedIn**, *)

Figure 2: Visualization of all proposed prediction tasks that use wildcards introduced in Table 1.

Algorithm 1: Multi-task Training (MTT)

Require: $\mathcal{T}$: set of training triples,
$\mathcal{E}$: set of entities in knowledge graph $\mathcal{K}$
θ: model parameters,
$\mathcal{Q}$: set of (q, w) pairs of training queries and corresponding weights

Ensure: Updated model parameters θ

foreach $q, w \in \mathcal{Q}$ **do**
- $N \leftarrow$ construct set of negatives for q using $\mathcal{T}$
- $\mathcal{T}_{\text{all}} \leftarrow \mathcal{T} \cup N$
- $s_{\text{all}} \leftarrow \text{COMPUTE\_SCORES}(\mathcal{T}_{\text{all}})$
- $l_q \leftarrow w * \text{COMPUTE\_LOSS}(s_{\text{all}}, \mathcal{T}_{\text{all}})$ `// loss weighted by` w
- $\theta \leftarrow \text{UPDATE\_PARAMETERS}(\theta, l_q)$

Algorithm 2: Extended Score Function (accepts wildcards)

Require: t: (i, k, j) triple to compute score
q: task query, e.g. $(i, k, *)$
s: model score function
$\mathcal{C}$: set of candidates for wildcard slot
Ensure: Score of given triple t

$max\_score \leftarrow 0$
if q does not have a wildcard **then**
  $max\_score \leftarrow s(t)$
else
  foreach $c \in \mathcal{C}$ **do**
    $candidate\_t = (i, k, c)$ // e.g. for $q = (i, k, *)$
    $candidate\_score = s(candidate\_t)$
    if $candidate\_score \geq max\_score$ **then**
      $max\_score \leftarrow candidate\_score$
return max_score

Dataset	Entities	Relations	Train	Valid	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
WIKIDATA5M	4 818 679	828	21 343 681	5 357	5 321

Table 5: Statistics of benchmark datasets for pre-training knowledge graph embeddings.

We performed 30 random trials using SOBOL sampling (Bergstra and 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 6.

B.2 Relative Squared Error

For evaluating regression performance, we chose relative squared error (RSE), defined as follows:

$$\text{RSE} = \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{3}$$

where N is the number of evaluation examples, y_i are targets to predict, $\hat{y}_i$ are model predictions, and $\bar{y} = \frac{1}{N}\sum_n y_i$, i.e. the mean of targets to predict. We chose 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 predicts the average of the dependent variable in the evaluation data; lower values are better.

B.3 Experimental Setup: Downstream Tasks

Downstream tasks. We collected or created data for 35 downstream tasks on FB15K-237, YAGO3-10 or WIKI5M (see Tables 7 and 8). This includes the datasets of Jain et al. (2021) for entity classification on FB15K-237 and YAGO3-10, which aim 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). To create regression tasks for WIKI5M, we followed the same approach using numerical relations extracted from Wikidata (Van Veen, 2019). Datasets statistics are given in Tables 7 and 8.

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	$[0.0, 0.5]$
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

[†] For RotatE, embedding size is fixed 128 on WNRR and set to either 128 or 256 for YAGO3-10. For Transe, this is set to either 128 or 256 for FB15K-237 and fixed to 128 for WNRR and 1024 for YAGO3-10.
[*] For RotatE, batch size is fixed to 256 in YAGO3-10 and to 128 on FB15K-237 and WNRR. For Transe, this is set to either 128 or 256 on YAGO3-10.

Table 6: Hyperparameter search space for pre-training KGE models. Restrictions for RotatE and TransE are due to higher memory consumption and runtime.

Benchmark	Name	Train	Validation	Test
FB15K-237	Entity Type	6 719	–	1 680
	Profession	2 537	–	635
	Organization Type	342	–	86
	Writer Type	136	–	34
YAGO3-10	Entity Type	69 592	–	17 398
	Player Type	33 928	–	8 483
	Profession	14 480	–	3 621
	Writer Type	4 870	–	1 218
	Scientist Type	2 041	–	511
	Organization Type	1 248	–	312
	Artists Type	520	–	130
	Waterbody Type	195	–	49

Table 7: 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	Train	Validation	Test
FB15K-237	Node Importance	9 877	1 380	2 823
	Birth Year	3 538	442	444
	Latitude	2 568	321	322
	Longitude	2 560	320	322
	Person Height	2 295	287	288
	Size Area	1 731	216	218
	Population	1 543	193	193
	Film Release Year	1 493	186	188
	Org Year Founded	985	123	124
	Film Rating	591	73	75
YAGO3-10	Born on Year	60 409	–	6 730
	Created on Year	23 896	–	2 638
	Died on Year	13 582	–	1 513
	Destroyed on Year	1 630	–	186
	Happened on Year	749	–	73
WIKI5M	Date of Birth	992 126	124 015	124 017
	Album Publication	29 156	3 644	3 645
	Asteroid Magnitude	16 722	2 090	2 091
	River Length	10 092	1 261	1 262
	Airport Elevation	9 054	1 131	1 133
	Sports Season Start	7 631	953	955
	Village Population	3 691	461	462
	Municipality Area	3 158	394	396

Table 8: 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 and WIKI5M datasets were created by us, except node importance, created by Huang et al. (2021).

KGE models. Since we are interested in pre-trained KGE models, we used the KGE models trained for the experiments discussed in Sec. 4.1. Thus, no information from downstream tasks was used for KGE model training and selection; i.e. the same KGE model is used for all downstream tasks in each experiment. For model selection, we selected STD models with LP task (the standard approach), but combined MTT models with the LP task or the MTR task. Further improvements may be made by using downstream tasks during training (Aribandi et al., 2022) at the cost, perhaps, of obtaining less general representations; we leave such exploration to future work.

Downstream models. We use scikit-learn (Pedregosa et al., 2011) using only the node embeddings of the pre-trained KG model as input features. For classification, we use multilayer perceptrons (MLP), logistic regression, KNN, and random forests. For regression, we use 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 for each downstream model 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), aggregated across all classification tasks (denoted 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 predicts the average of the dependent variable in the evaluation 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 include KE-GCN (Yu et al., 2021), a recent GNN with state-of-the-art results for graph alignment and entity classification. In contrast to KGEs, this model is directly trained on the downstream task (i.e., no pre-training) and needs to access the KG to perform predictions. For regression tasks, we use a linear layer after the final convolutional layer of KE-GCN, as this led to better performance in our experiments compared to using a single dimensional output in the final convolution layer as done by Huang et al. (2021). We tune hyperparameters using 30 SOBOL trials (as for KGE models); the search space is shown in Table 9. For training, evaluation, and model selection, we follow the approach for our downstream models (e.g,. 5-fold CV).

C Additional Experimental Results

Model selection using downstream information. To explore whether results can improve by using downstream information to select models, Table 10 reports performance on FB15K-237 of some KGE models using both training approaches in combination with either LP for model selection (which consistently provided better results for these models with both training approaches) or by selecting directly on the metric used to evaluate the downstream task (weighted F1 for entity classification and RSE for regression). We found that model selection with the downstream task metric provides only marginal benefits for both STD and MTT and can in fact be detrimental, likely due to overfitting on validation data. This indicates that model selection without information about downstream tasks—i.e., using LP or MTR—may be preferrable to using downstream information. This is beneficial, as including downstream information during pre-training or model selection would likely make the resulting representations less general.

Overall, we found that full MTT training with LP for model selection was a suitable choice, but further improvements are possible by dataset-, model- and task-specific choices of pre-training task and validation objective, as discussed in the next section.

Further model selection approaches. For completeness, we also explored the impact of further combinations of model selection methods with both STD and MTT training. To explore whether there would be improvements in STD models when selecting them based on performance on the MTR task, Table 11 reports downstream performance of some KGE models using STD training combined with either LP or MTR for model selection. We see that the combination of STD with MTR leads to lower downstream performance almost every time.

Model	Hyperparameter	Values
MLP	Hidden Layer	{(100,), (10,), (100, 100), (10, 10)}
	Alpha	$[0.00001, 0.001]$
	Learning Rate	$[0.001, 0.01]$
	Solver	$[\text{Adam}, \text{LBFGS}]$
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.

		Selection Method			
		EC - Weighted F1		*REG - RSE*	
		LP	Weighted F1	LP	RSE
ComplEx	STD	.844	.850	.447	.437
	MTT	.858	.827	.394	**.393**
DistMult	STD	**.873**	.846	.551	.539
	MTT	.865	.864	.472	.476

Table 10: Performance on FB15K-237 downstream tasks for different KGE model training (STD and MTT) and two model selection approaches: LP and weighted F1 (higher is better) or RSE (lower is better). Using downstream task data for model selection provides only marginal gains and is sometimes detrimental to downstream performance, likely due to overfitting on validation data.

		Selection Method			
		EC - Weighted F1		*REG - RSE*	
		LP	MTR	LP	MTR
ComplEx	STD	.844	.858	**.447**	.545
DistMult	STD	**.873**	.836	.551	.686

Table 11: Performance on FB15K-237 downstream tasks for STD training and two model selection approaches: LP and MTR. On both types of tasks, the best performance is obtained by combining STD training with LP model selection.

		Avg. epoch time in seconds			
		FB-237	YAGO	WNRR	WIKI5M
ComplEx	STD	4.92	97.88	2.32	823.80
	MTT	10.83	137.13	8.13	1635.90
TransE	STD	78.76	141.62	98.45	1115.65
	MTT	245.05	219.42	278.60	2124.29

Table 12: Average training epoch time in seconds over first 5 epochs of best 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.

		Train.	*Sel.*	*Graph-structure prediction (↑)*						*Downstream tasks*	
				LP	REL	DOM	NBE	NBR	MTR	EC (↑)	REG (↓)
FB15K-237	ComplEx	STD	LP	**.346**	.805	.423	.016	.046	.274	.844±.008	.447±.051
		MTT	LP	.336	.964	.557	.195	.794	.525	.858±.005	**.394±.057**
		MTT	MTR	.331	**.977**	.773	**.210**	**.925**	**.606**	.843±.002	.412±.037
	DistMult	STD	LP	.342	.388	.045	.009	.036	.139	.873±.009	.551±.062
		MTT	LP	.334	.944	.557	.139	.818	.516	.865±.005	.472±.026
		MTT	MTR	.327	.939	.780	.142	.879	.577	.857±.006	.482±.026
	RotatE	STD	LP	.312	.919	.581	.051	.136	.342	.868±.003	.797±.286
		MTT	LP	.319	.965	.758	.136	.880	.572	**.890±.003**	.573±.062
		MTT	MTR	.314	.964	**.813**	.160	.922	.598	.847±.001	.704±.060
	TransE	STD	LP	.330	.900	.624	.038	.054	.332	.873±.015	.742±.287
		MTT	LP	.317	.963	.653	.152	.855	.547	.855±.007	.795±.257
		MTT	MTR	.288	.960	.708	.112	.911	.555	.878±.009	.681±.095
	KE-GCN			–	–	–	–	–	–	.829±.526	.501±.001
YAGO3-10	ComplEx	STD	LP	**.550**	.900	.120	.215	.517	.411	.712±.008	.589±.023
		MTT	LP	.538	.941	.836	.591	**.978**	.759	.729±.005	.466±.017
		MTT	MTR	.538	.930	.836	.591	.940	.749	.729±.005	.459±.020
	DistMult	STD	LP	.539	.881	.010	.327	.613	.429	.734±.003	.519±.019
		MTT	LP	.536	.945	**.861**	.581	**.978**	**.762**	**.746±.006**	.472±.029
		MTT	MTR	.536	.941	**.861**	.581	.967	.759	.735±.004	.466±.021
	RotatE	STD	LP	.436	.809	.046	.400	.656	.432	.701±.002	.696±.018
		MTT	LP	.509	.918	.011	**.609**	.366	.434	.708±.002	.659±.059
		MTT	MTR	.427	.933	.032	.550	.694	.482	**.746±.001**	.470±.017
	TransE	STD	LP	.504	.860	.178	.287	.175	.349	.742±.002	.447±.036
		MTT	LP	.462	.940	.037	.476	.338	.396	.723±.004	.441±.029
		MTT	MTR	.048	**.954**	.686	.046	.798	.457	.688±.005	.680±.026
	KE-GCN			–	–	–	–	–	–	.700±.223	**.398±.008**

Table 13: Performance on test data of graph-structure prediction and downstream tasks. Bold entries show best performance per task and dataset. Underlined entries show best performance between STD and MTT.

		Train.	Sel.	*Graph-structure prediction (↑)* LP	REL	DOM	NBE	NBR	MTR	*Downstream tasks* EC (↑)	REG (↓)
WNRR	ComplEx	STD	LP	**.474**	.782	.396	.246	.690	.488	–	–
		MTT	MTR	.459	.831	**.593**	.426	.953	**.633**	–	–
	DistMult	STD	LP	.447	.767	.081	.253	.702	.415	–	–
		MTT	MTR	.431	.804	.573	.342	.952	.600	–	–
	RotatE	STD	LP	.469	.794	.311	**.432**	.881	.553	–	–
		MTT	MTR	.431	**.874**	.512	.239	**.955**	.572	–	–
	TransE	STD	LP	.174	.707	.044	.171	.332	.239	–	–
		MTT	MTR	.094	.603	.476	.095	.827	.399	–	–
WIKI5M	ComplEx	STD*	LP	**.288**	–	–	–	–	–	–	.687±.032
		MTT	LP	.204	.680	.028	.130	.197	.200	–	.706±.025
		MTT	MTR	.215	.804	.087	.136	.342	.263	–	.720±.023
	TransE	STD*	LP	**.288**	–	–	–	–	–	–	**.596±.011**
		MTT	LP	**.250**	**.908**	**.185**	**.169**	**.503**	**.347**	–	.636±.025
		MTT	MTR	**.250**	**.908**	**.185**	**.169**	**.503**	**.347**	–	.650±.018
	KE-GCN†			–	–	–	–	–	–	–	–

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

Table 14: Performance on test data of graph-structure prediction and downstream tasks with MTT training and two model selection methods: LP and MTR. Due to high cost, we trained only two models for WIKI5M: ComplEx and TransE. Bold entries show best performance per task and dataset. Underlined entries show best performance between STD and MTT. For entity classification (EC) we report weighted F1 (higher is better), and for regression (REG) we report relative squared error (lower is better).

Model Performance Sorted in Decreasing Order for each Pre-Training and Downstream Task

	Graph-structure LP (↑)			MTR (↑)			*Downstream Tasks* EC (↑)			REG (↓)		
FB15K-237	ComplEx	STD	.346	ComplEx	MTT	.606	RotatE	MTT	.890±.003	ComplEx	MTT	.394±.057
	DistMult	STD	.342	RotatE	MTT	.598	TransE	MTT	.878±.009	ComplEx	STD	.447±.051
	ComplEx	MTT	.331	DistMult	MTT	.577	TransE	STD	.873±.015	DistMult	MTT	.472±.026
	TransE	STD	.330	TransE	MTT	.555	DistMult	STD	.873±.009	KE-GCN		.501±.001
	DistMult	MTT	.327	RotatE	STD	.342	RotatE	STD	.868±003	DistMult	STD	.551±.062
	RotatE	MTT	.314	TransE	STD	.332	DistMult	MTT	.865±009	RotatE	MTT	.573±.062
	RotatE	STD	.312	ComplEx	STD	.274	ComplEx	MTT	.858±005	TransE	MTT	.681±.095
	TransE	MTT	.288	DistMult	STD	.139	ComplEx	STD	.844±008	TransE	STD	.742±.287
							KE-GCN		.829±.526	RotatE	STD	.797±.286
YAGO3-10	ComplEx	STD	.550	DistMult	MTT	.759	DistMult	MTT	.746±.006	KE-GCN		.398±.008
	DistMult	STD	.539	ComplEx	MTT	.749	RotatE	MTT	.746±.001	TransE	MTT	.441±.029
	ComplEx	MTT	.538	RotatE	MTT	.482	TransE	STD	.742±.002	TransE	STD	.447±.036
	DistMult	MTT	.536	TransE	MTT	.457	DistMult	STD	.734±.003	ComplEx	MTT	.459±.020
	TransE	STD	.504	RotatE	STD	.432	ComplEx	MTT	.729±.005	RotatE	MTT	.470±.017
	RotatE	STD	.436	DistMult	STD	.429	TransE	MTT	.723±.004	DistMult	MTT	.472±.029
	RotatE	MTT	.427	ComplEx	STD	.411	ComplEx	STD	.712±.008	DistMult	STD	.519±.019
	TransE	MTT	.048	TransE	STD	.349	RotatE	STD	.701±.002	ComplEx	STD	.589±.023
							KE-GCN		.700±.223	RotatE	STD	.696±.018

Table 15: Sorted performance on test data of graph-structure prediction tasks and downstream tasks of all KGE models we tested, as well as KE-GCN by Yu et al. (2021). The ranking of models given by their LP or MTR performance is not the same as the ranking of models given by their downstream performance, which suggests that more work is needed to understand how to pre-train KGE models to optimize downstream performance.

	Train.	Sel.	Graph-structure prediction (↑)						Downstream tasks	
			LP	REL	DOM	NBE	NBR	MTR	EC (↑)	REG (↓)
ComplEx	STD	LP	**.346**	.805	.423	.016	.046	.274	.844±.008	.447±.051
	MTT	MTR	.331	**.977**	.773	**.210**	.925	**.606**	.843±.002	**.412±.037**
	w/o LP	MTR	.154	.972	.831	.200	.932	.579	.870±.002	.512±.044
	w/o REL	MTR	.322	.831	.831	.159	.927	.590	.851±.005	.486±.035
	w/o DOM	MTR	.327	.966	.713	.198	.915	.586	.851±.003	.479±.029
	w/o NBE	MTR	.315	.958	**.850**	.005	.936	.575	.856±.002	.562±.038
	w/o NBR	MTR	.325	.967	.795	.199	.874	.595	.858±.000	.459±.062
	w/o LP+REL	MTR	.001	.009	.843	.177	**.939**	.436	.849±.011	.542±.054
	w/o DOM+NBR	MTR	.330	.970	.074	.199	.107	.266	.856±.001	.415±.029
DistMult	STD	LP	.342	.388	.045	.009	.036	.139	.873±.009	.551±.062
	MTT	MTR	.327	.939	.780	.142	.879	.577	.857±.006	.482±.026
	w/o LP	MTR	.159	.954	.826	.087	.937	.553	.861±.008	.522±.067
	w/o REL	MTR	.323	.857	.827	.057	.932	.571	.868±.008	.536±.077
	w/o DOM	MTR	.323	.948	.703	.106	.914	.560	.849±.002	.478±.027
	w/o NBE	MTR	.316	.928	.848	.003	.937	.571	.844±.002	.524±.047
	w/o NBR	MTR	.325	.956	.801	.112	.775	.554	.859±.002	.493±.043
	w/o LP+REL	MTR	.000	.019	.837	.108	.937	.420	.856±.001	.572±.085
	w/o DOM+NBR	MTR	.307	.955	.136	.147	.279	.299	.839±.001	.545±.060
RotatE	STD	LP	.312	.919	.581	.051	.136	.342	.868±.003	.797±.286
	MTT	MTR	.314	.964	.813	.160	.922	.598	.847±.001	.704±.060
	w/o LP	MTR	.204	.914	.842	.126	.928	.568	.874±.000	.661±.043
	w/o REL	MTR	.272	.887	.846	.137	.924	.583	.862±.003	.692±.079
	w/o DOM	MTR	.319	.965	.661	.170	.883	.559	**.898±.001**	.593±.078
	w/o NBE	MTR	.301	.960	.813	.003	.912	.558	.862±.003	.558±.050
	w/o NBR	MTR	.318	.964	.710	.168	.673	.522	.863±.007	.552±.035
	w/o LP+REL	MTR	.012	.031	.842	.124	.916	.424	.864±.001	.743±.123
	w/o DOM+NBR	MTR	.322	.945	.016	.166	.019	.221	.854±.001	.809±.249
TransE	STD	LP	.330	.900	.624	.038	.054	.332	.873±.015	.742±.287
	MTT	MTR	.288	.960	.708	.112	.911	.555	.878±.009	.681±.095
	w/o LP	MTR	.271	.968	.781	.138	.901	.572	.870±.000	.486±.027
	w/o REL	MTR	.307	.944	.698	.124	.906	.557	.856±.001	.622±.061
	w/o DOM	MTR	.325	.965	.626	.126	.879	.542	.863±.000	.539±.052
	w/o NBE	MTR	.330	.966	.801	.012	.904	.562	.884±.002	.463±.032
	w/o NBR	MTR	.329	.966	.723	.125	.790	.545	.857±.007	.458±.024
	w/o LP+REL	MTR	.149	.930	.821	.116	.924	.550	.860±.001	.594±.032
	w/o DOM+NBR	MTR	.312	.962	.360	.129	.580	.414	.864±.001	.497±.057

Table 16: Performance on test data of graph-structure prediction and downstream tasks for FB15K-237 of STD with LP model selection and various forms of multi-task training, all using MTR for model selection. Objectives such as w/o LP are MTT objectives with all tasks in Table 1 except one, in this case, LP. Our results show that excluding the LP task during pre-training often results in improved downstream performance, and that using all pre-training tasks is often not the best choice.

		FB15K-237 *Entity Classification (Weighted F1 - higher is better)* Type	Profession	Organization	Writer
ComplEx	STD+LP	.986±.001	.808±.011	.921±.021	.661±.000
	MTT+LP	.986±.000	.820±.005	.946±.003	.682±.012
	MTT+MTR	.986±.000	.802±.004	.944±.003	.641±.000
DistMult	STD+LP	.984±.000	.811±.007	.912±.009	.785±.020
	MTT+LP	.987±.000	.810±.016	.974±.002	.690±.000
	MTT+MTR	.986±.000	.785±.006	.890±.000	.768±.018
RotatE	STD+LP	.985±.000	.797±.000	.908±.013	.781±.000
	MTT+LP	.989±.001	.807±.000	.934±.012	.828±.000
	MTT+MTR	.989±.000	.810±.000	.931±.003	.658±.000
TransE	STD+LP	.984±.001	.791±.005	.913±.032	.806±.021
	MTT+LP	.987±.000	.805±.006	.946±.009	.681±.014
	MTT+MTR	.987±.000	.796±.000	.942±.000	.789±.034
KE-GCN		.988±.000	.738±.000	.906±.002	.685±.020

Table 17: 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. 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
ComplEx	STD+LP	.994±.000	.918±.001	.753±.004	.575±.006	.518±.013	.789±.005	.480±.018	.673±.015
	MTT+LP	.997±.000	.919±.002	.790±.002	.619±.006	.553±.011	.877±.003	.466±.013	.614±.000
	MTT+MTR	.996±.000	.914±.001	.776±.000	.617±.009	.556±.007	.871±.005	.491±.021	.614±.000
DistMult	STD+LP	.994±.000	.919±.001	.764±.003	.577±.000	.529±.003	.814±.011	.535±.007	.738±.000
	MTT+LP	.996±.000	.919±.002	.789±.002	.634±.019	.556±.003	.890±.010	.495±.010	.691±.000
	MTT+MTR	.996±.000	.918±.002	.776±.000	.622±.006	.539±.009	.876±.005	.462±.006	.691±.000
RotatE	STD+LP	.973±.001	.914±.000	.706±.002	.611±.000	.545±.000	.734±.014	.530±.000	.593±.000
	MTT+LP	.990±.001	.913±.001	.733±.000	.605±.000	.469±.009	.793±.005	.413±.000	.751±.000
	MTT+MTR	.994±.000	.919±.001	.768±.000	.643±.000	.576±.000	.830±.011	.534±.000	.707±.000
TransE	STD+LP	.993±.000	.919±.001	.762±.000	.623±.000	.630±.000	.833±.000	.507±.015	.670±.000
	MTT+LP	.991±.000	.912±.000	.728±.005	.583±.000	.603±.000	.804±.011	.506±.007	.654±.006
	MTT+MTR	.992±.000	.892±.000	.750±.000	.580±.000	.401±.012	.809±.003	.464±.015	.614±.012
KE-GCN		.996±.000	.896±.001	.709±.000	.582±.005	.610±.006	.853±.006	.463±.014	.488±.014

Table 18: 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. 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
ComplEx	STD+LP	.870±.048	.601±.239	.172±.013	.089±.010	.678±.010
	MTT+LP	.918±.142	.477±.190	.145±.015	.066±.008	.661±.011
	MTT+MTR	.909±.086	.214±.050	.143±.009	.096±.008	.678±.000
DistMult	STD+LP	.807±.023	.844±.042	.182±.031	.088±.005	.669±.003
	MTT+LP	.788±.006	.827±.065	.143±.001	.083±.013	.651±.009
	MTT+MTR	.802±.049	.701±.052	.232±.053	.070±.006	.691±.000
RotatE	STD+LP	.913±.000	.872±.027	.498±.057	.279±.003	.657±.000
	MTT+LP	.834±.016	.797±.069	.313±.014	.173±.003	.813±.136
	MTT+MTR	.856±.003	.811±.005	.411±.022	.225±.096	.847±.000
TransE	STD+LP	.886±.035	.836±.041	.170±.022	.084±.006	.722±.003
	MTT+LP	.833±.018	.812±.012	.078±.011	.061±.003	.769±.009
	MTT+MTR	.897±.044	.655±.053	.088±.005	.052±.006	.824±.000
KE-GCN		.804±.005	.376±.035	.218±.023	.113±.003	.748±.002

Table 19: Part 1: 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. Models with RSE above 1 are considered unsatisfactory. Datasets are sorted by decreasing size of the training set from left to right.

		FB15K-237				
		Regression (RSE - lower is better)				
		Size Area	Population	Film Year	Date Founded	Film Rating
ComplEx	STD+LP	.234±.018	.442±.071	.156±.016	.494±.042	.736±.046
	MTT+LP	.046±.026	.260±.064	.138±.007	.431±.047	.795±.058
	MTT+MTR	.049±.021	.493±.097	.126±.003	.605±.033	.804±.065
DistMult	STD+LP	.412±.318	.914±.093	.152±.003	.627±.036	.813±.062
	MTT+LP	.435±.046	.503±.004	.134±.012	.429±.045	.728±.063
	MTT+MTR	.025±.008	.540±.030	.146±.005	.718±.012	.894±.043
RotatE	STD+LP	.700±.223	.463±.429	.176±.004	.618±.024	.792±.089
	MTT+LP	.708±.112	.537±.035	.146±.008	.514±.055	.897±.168
	MTT+MTR	.440±.190	.710±.158	.157±.010	.631±.060	.949±.056
TransE	STD+LP	.326±.075	.906±.574	.153±.019	.499±.046	.839±.045
	MTT+LP	.041±.744	.227±.730	.141±.004	.300±.013	.690±.031
	MTT+MTR	.833±.684	.675±.109	.130±.012	.708±.022	.946±.018
KE-GCN		.754±.0180	.664±.051	.144±.008	.498±.034	.691±.009

Table 20: Part 2: 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. 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
ComplEx	STD+LP	.519±.001	.672±.033	.555±.014	.872±.060	.324±.006
	MTT+LP	.345±.025	.603±.009	.377±.005	.709±.009	.296±.036
	MTT+MTR	.363±.010	.643±.016	.406±.023	.605±.029	.277±.023
DistMult	STD+LP	.432±.013	.612±.024	.466±.025	.773±.004	.311±.030
	MTT+LP	.345±.023	.565±.015	.416±.023	.724±.044	.312±.040
	MTT+MTR	.352±.006	.648±.016	.438±.035	.677±.024	.214±.022
RotatE	STD+LP	.689±.027	.800±.009	.849±.000	.913±.000	.227±.055
	MTT+LP	.538±.006	.717±.008	.657±.018	.886±.031	.497±.233
	MTT+MTR	.421±.016	.706±.012	.468±.003	.616±.043	.137±.013
TransE	STD+LP	.422±.018	.647±.008	.351±.037	.513±.057	.300±.059
	MTT+LP	.371±.006	.725±.022	.434±.009	.573±.081	.100±.027
	MTT+MTR	.494±.017	.777±.000	.521±.038	.942±.048	.666±.024
KE-GCN		.256±.009	.611±.008	.299±.011	.657±.045	.167±.001

Table 21: 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. Models with RSE above 1 are considered unsatisfactory. Datasets are sorted by decreasing size of the training set from left to right.

		WIKIDATA5M			
		Regression (RSE - lower is better)			
		Date of Birth	Album Pub.	Asteroid Mag.	River Length
ComplEx	STD+LP	.475±.003	.760±.009	.436±.014	.559±.022
	MTT+LP	.481±.006	.844±.009	.519±.026	.540±.007
	MTT+MTR	.468±.010	.813±.006	.518±.014	.659±.025
TransE	STD+LP	.373±.002	.555±.004	.377±.015	.444±.016
	MTT+LP	.434±.007	.669±.003	.439±.013	.433±.029
	MTT+MTR	.455±.005	.667±.010	.455±.021	.418±.021

Table 22: Part 1: 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 WIKIDATA5M; Models with RSE above 1 are considered unsatisfactory. Datasets are sorted by decreasing size of the training set from left to right.

		WIKIDATA5M			
		Regression (RSE - lower is better)			
		Airport Elev.	Season Start	Population	Munic. Area
ComplEx	STD+LP	.849±.007	.596±.002	.019±.197	.801±.000
	MTT+LP	.917±.000	.695±.014	.785±.139	.867±.000
	MTT+MTR	.928±.000	.657±.040	.841±.086	.877±.000
TransE	STD+LP	.734±.019	.546±.029	.928±.000	.811±.000
	MTT+LP	.894±.037	.654±.024	.739±.087	.825±.000
	MTT+MTR	.873±.000	.610±.011	.896±.080	.825±.000

Table 23: Part 2: 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 WIKIDATA5M; Models with RSE above 1 are considered unsatisfactory. Datasets are sorted by decreasing size of the training set from left to right.

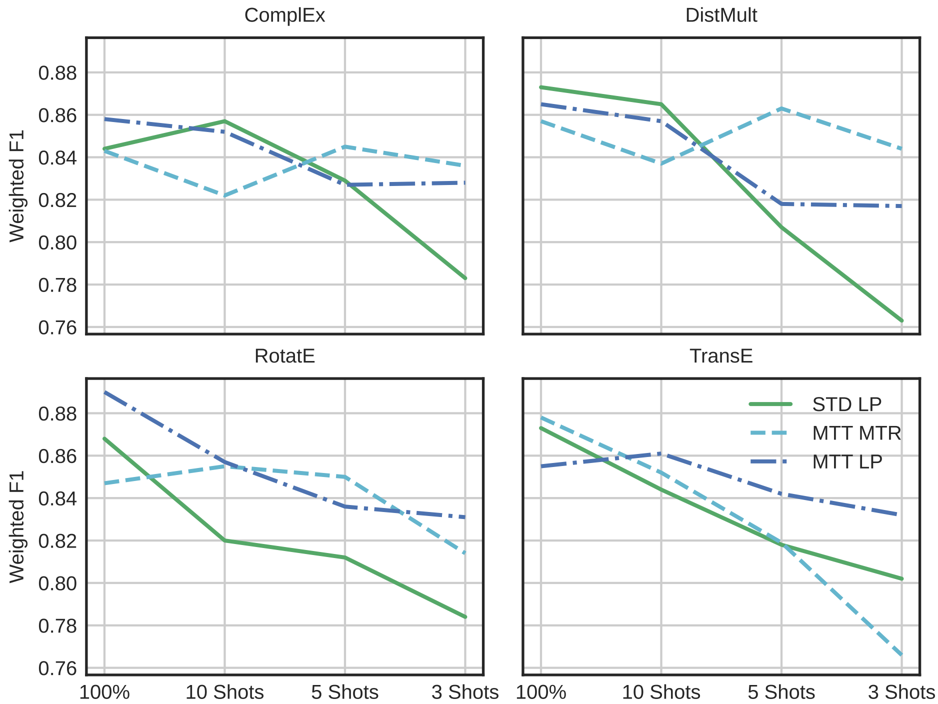


Figure 3: Few-shot performance of entity classification tasks for FB15K-237 (higher is better). Each n-shot training set consists of n sampled positive and negative examples for each class.

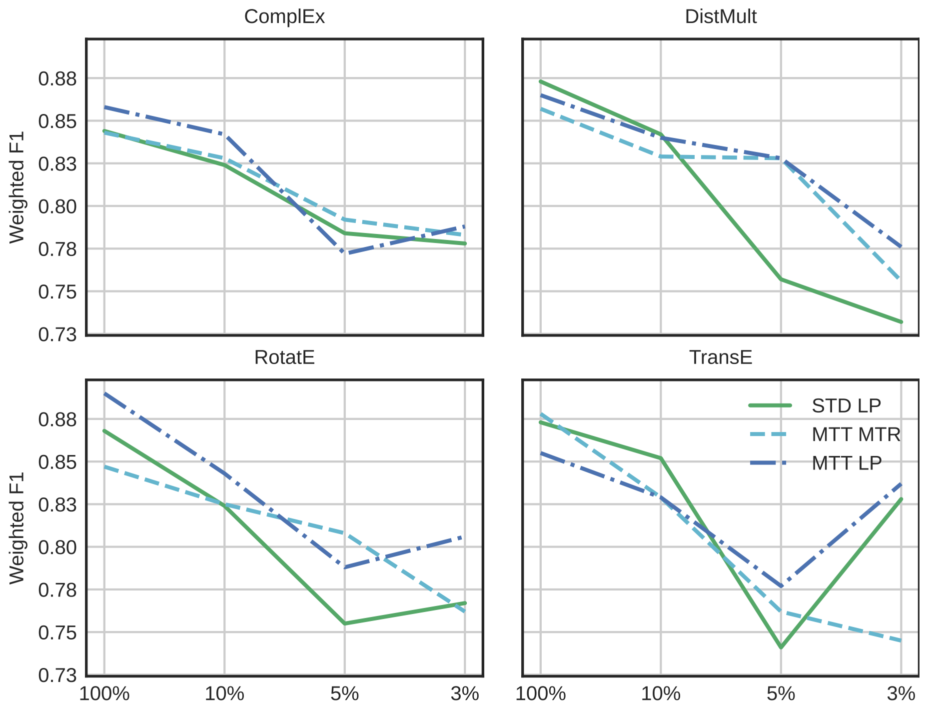


Figure 4: Performance on entity classification for FB15K-237 with downsampled training sets (higher is better). Each training set was constructed by sampling (stratified) a percentage of the training set.

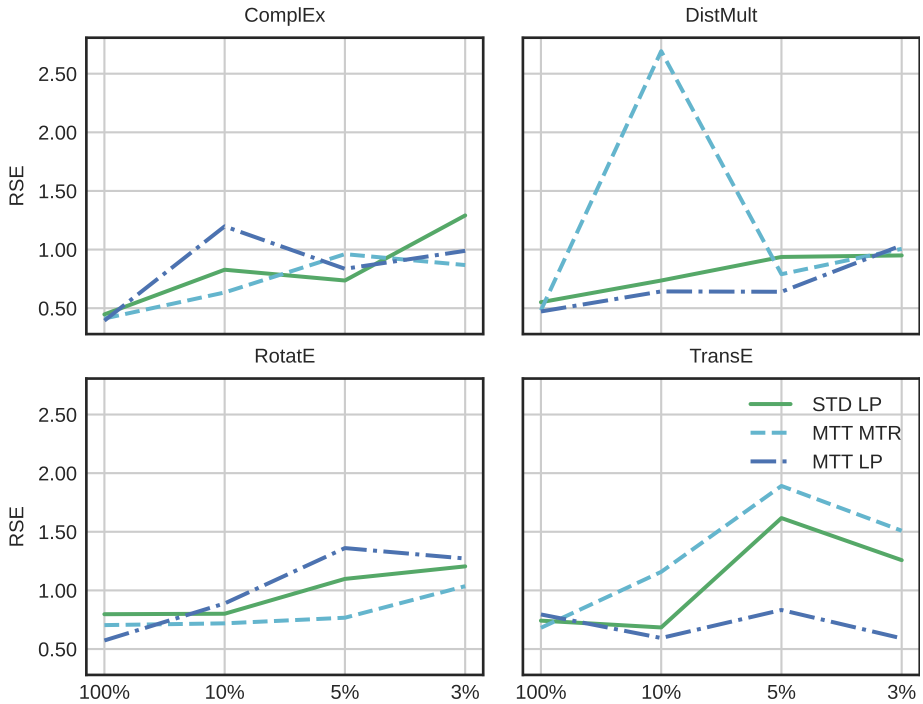


Figure 5: Performance of regression tasks for FB15K-237 with downsampled training sets (lower is better). Each training set was constructed by sampling a percentage of the training set.

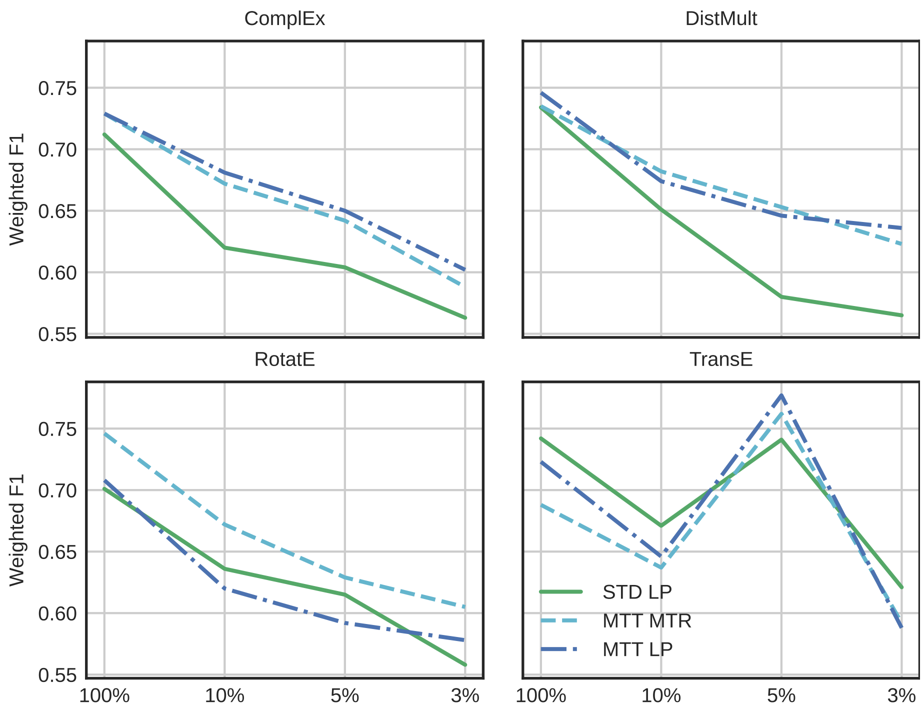


Figure 6: Performance on entity classification for YAGO3-10 with downsampled training sets (higher is better). Each training set was constructed by sampling (stratified) a percentage of the training set.

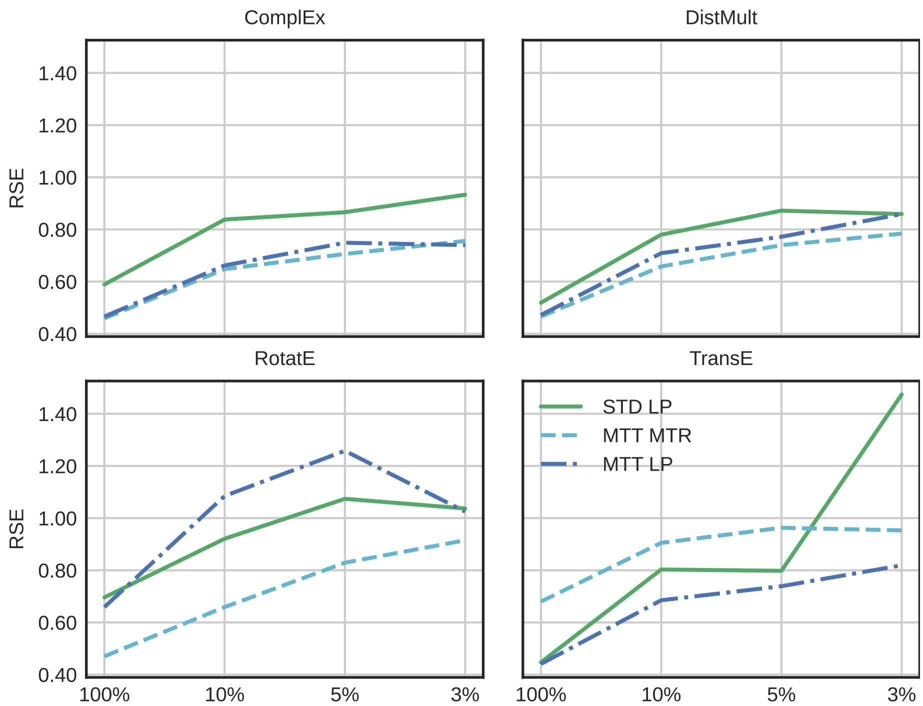


Figure 7: Performance of regression tasks for YAGO3-10 with downsampled training sets (lower is better). Each training set was constructed by sampling a percentage of the training set. The gap in performance between MTT and STD models becomes larger as training data becomes less available.