Journal of Artificial Intelligence Research 78 (2023) 37-68 Submitted 12/2022; published 09/2023

A Benchmark Study on Knowledge Graphs Enrichment and Pruning Methods in the Presence of Noisy Relationships

Stefano Faralli faralli@di.uniroma1.it
Andrea Lenzi lenzi@di.uniroma1.it
Paola Velardi velardi@di.uniroma1.it
Computer Science Department, Sapienza University of Rome,
Via Salaria 113, Rome (00198), Italy

Abstract

In the past few years, knowledge graphs (KGs), as a form of structured human intelligence, have attracted considerable research attention from academia and industry. In this very active field of study, a widely explored problem is that of link prediction, the task of predicting whether two nodes should be connected, based on node attributes and local or global graph connectivity properties. The state of the art in this area is represented by techniques based on graph embeddings. However, KGs, especially those acquired using automated or partly automated techniques, are often riddled with noise, e.g., wrong relationships, which makes the problem of link deletion as important as that of link prediction. In this paper, we address three main research questions. The first is about the true effectiveness of different knowledge graph embedding models under the presence of an increasing number of wrong links. The second is to asses if methods that can predict unknown relationships effectively, work equally well in recognizing incorrect relations. The third is to verify if there are systems robust enough to maintain primacy in all experimental conditions. To answer these research questions, we performed a systematic benchmark study in which the experimental setting includes ten state-of-the-art models, three common KG datasets with different structural properties and three downstream tasks: the widely explored tasks of *link prediction* and *triple classification*, and the less popular task of *link deletion*. Comparative studies often yield contradictory results, where the same systems score better or worse depending on the experimental context. In our work, in order to facilitate the discovery of clear performance patterns and their interpretation, we select and/or aggregate performance data to highlight each specific comparison dimension: dataset complexity, type of task, category of models, and robustness against noise.

1. Introduction

Knowledge graphs (KGs) have rapidly emerged as an important area in AI over the last ten years, both in academia and industry. A KG is commonly defined as a structured representation of facts, consisting of entities and relationships annotated with semantic labels (Ji et al., 2022; Hogan et al., 2021).

Essentially, KGs are represented as collections of real-world triples, where each triple or fact *(h, r, t)* denotes some relation r between a head entity h and a tail entity t. KGs can be formalized as directed multi-relational graphs, in which:

- ***nodes*** correspond to entities (they can be real-world objects or abstract concepts);

- ***edges*** encode various kinds of relationships and possess semantic meanings based on the specified label (they represent the relations between entities).

Popular KG examples are both domain-specific, as the GeneOntology (Ashburner et al., 2000) and generic, like WordNet (Fellbaum, 1998), YAGO (Suchanek et al., 2007), Freebase (Bollacker et al., 2008), NELL (Carlson et al., 2010) and DBpedia (Lehmann et al., 2015). There are also a number of popular commercial KGs, such as Google's Knowledge Graph, Microsoft's Satori and Facebook's Open Graph (Nguyen, 2017).

Recent advances in KG-based research focus on Knowledge Representation Learning (KRL) or Knowledge Graph Embedding (KGE) by mapping entities and relations into low-dimensional vectors while capturing their semantic meanings (Ji et al., 2022). Leveraging their embedded representation, KGs can be used for various tasks, such as: *link prediction* (Bordes et al., 2013; Dettmers et al., 2018; Schlichtkrull et al., 2018; Vashishth et al., 2020; Yao et al., 2019; Daza et al., 2021), *triple classification* (Wang et al., 2014a; Socher et al., 2013; Yao et al., 2019), entity recognition (Mehta et al., 2021), *clustering* (Petzold et al., 2021), and relation extraction (Lin et al., 2015; Weston et al., 2013).

In link prediction, KGE-based methods represent the large majority.

In real-world scenarios, creating and maintaining KGs is a costly and time-consuming process involving experts and competencies across different areas of knowledge. To address the knowledge bottleneck problem and to cope with creation costs, one can rely on automated ontology learning techniques (Khadir et al., 2021) or on the Semantic Web and Linked Open Data principles, guidelines, and tools for the interlinking of existing KGs, thus promoting the reuse of existing dictionaries (Saha & Mandal, 2021; Zeng et al., 2021).

However, automatically acquired or interlinked KGs may present a substantial amount of wrong (hereafter referred to as "noisy") and missing information, due to the fact that knowledge acquisition approaches leverage heterogeneous and often unstructured sources like Web pages, relational databases, NoSQL databases, other KGs, online forums, etc. Essentially, the two major problems affecting KGs are *incompleteness* (missing links) and *incorrectness* (wrong links). The root causes of the presence of wrong links in real-world KGs are many. In general, wrong links are the result of automated mining methodologies devoted to inducing KGs from scratch or as alignment between existing (and sometimes heterogeneous) KGs. For instance, during the two editions of the SemEval Taxonomy Extraction Evaluation task (Bordea et al., 2015, 2016), a relevant portion of noise (ranging from 51% to 91%) has been estimated[1] from the resulting automatically induced taxonomies. Other popular examples of noisy KGs are, among others, YAGO (Suchanek et al., 2007) with a 95% accuracy and NELL (Carlson et al., 2010) with about 55% of correct triples in its original release (after 1K iterations). More recently, (Faralli et al., 2022) have estimated the presence of about 20% wrong triples in a resource consisting of a large knowledge graph automatically acquired by linking the Italian cultural heritage entities.

To mitigate these problems, several link prediction and error detection methods have been proposed. However, few methods explicitly focus on error *correction* (Melo & Paulheim, 2017a), i.e., the task of identifying and deleting wrong links, rather than simply classifying a link as being correct or wrong - which is referred to as triple classification. Indeed, while the vast majority of state-of-the-art methods concentrate on link prediction -

1. The evaluation has been carried out by human experts on a sample of triples.

to reliably extend the KG with new triples - or triple classification, the study of link deletion did not receive as much attention, as also noted in (Melo & Paulheim, 2017b).

In this study, our main purpose is to investigate whether best-performing state-of-the-art *link prediction* and *triple classification* methods perform equally well in the task of *link deletion*, and furthermore, if they are robust with regard to both *link prediction* and *deletion*, in contexts where the KGs contain noise. To this end, differently from previous literature, we consider the following research questions (RQs):

1. *RQ1*: Do methods that can predict unknown relationships effectively (*link prediction*), work equally well in recognizing incorrect associations (*link deletion*)?
2. *RQ2*: To what extent are the best-performing Knowledge Graph Embedding (KGE) methods (according to the literature) robust in the presence of an increasing amount of noisy triples?
3. *RQ3:* Are there systems, or categories of approaches, that show primacy over the others with different tasks, datasets, and increasing levels of noise? If so, what are their "winning" features?

In particular, our RQs are specifically concerned with studying the effect of noise and with the task of graph pruning, rather than only graph enrichment (Hogan et al., 2021).

To answer our research questions, we propose a benchmark study of different KGE models for the three tasks of *link prediction*, *triple classification* and *link deletion*, in the presence of an increasing level of noise, and on three common KG datasets with different connectivity properties.

A summary of the contributions of this paper follows:

- We perform a benchmark study of KGE methods with an increasing noise level. Usually, in other comparative studies, the amount of injected noise is not considered a high-level hyperparameter. With the proposed experiments, we can analyze the robustness to noise of single systems, as well as compare systems with respect to their ability to identify new existing links or to prune wrong ones in complex, real-world contexts;
- To avoid inconclusive results, as is often the case when comparing different systems on different tasks and different datasets, and to *facilitate the detection and interpretation of performance patterns*, we group the compared systems in three categories of models, and we analyze the performance along four dimensions: the type of task, the model category, the amount of injected noise, and the peculiarities of each dataset;
- In line with previous comparative studies (see Section 2), we performed hyperparameter tuning individually for all the analyzed models, applying Bayesian optimization techniques. Given the dimensions of the considered datasets (KGs), this has implied the use of intensive computational resources;
- To support the replicability and reproducibility of the study, we provide a package with all the useful resources: original datasets, noisy datasets, optimized hyperparameters, trained models, experiments scripts, etc.

The remaining of this paper is organized as follows: Section 2 describes the literature about existing benchmarks; Section 3 describes the experimental settings, i.e., the datasets (see Section 3.1), the generation of incrementally noisy datasets (see Section 3.2), the models (see Section 3.4) and the hyperparameters tuning (see Section 3.6). Then, in Section 4, we discuss the results in relation to the three above listed Research Questions, and finally, in Section 5, we summarize our findings and discuss future research directions.

2. Related Works

As discussed in Section 1, performances of real-world knowledge-based applications strongly depend on the availability of large formal knowledge representations such as KGs (Feigenbaum, 1984). To cope with the knowledge bottleneck problem, which mainly affects domain-specific applications, novel KGs can be automatically acquired (Khadir et al., 2021) or obtained by augmenting and interlinking existing KGs (Saha & Mandal, 2021; Zeng et al., 2021). Both approaches are error-prone, and as a result, noisy or absent triples are generated affecting both precision and recall of downstream tasks.

In this Section, we discuss the literature concerned with benchmark studies analyzing state-of-the-art models for *link prediction* and *triple classification*. To the best of our knowledge, no comparative studies have addressed *link deletion*, as we do.

Kadlec, Bajgar, and Kleindienst (2017) performed an experimental study in which they compared 29 KGE models using the standard metric of *Hits@10* on two standard KG datasets: *WN18* (Dettmers et al., 2018) (see Section 3.1.2) and *FB15k* (Toutanova & Chen, 2015) (see Section 3.1.1). They showed that even one of the simplest and oldest models, such as *DistMult* (Yang et al., 2015) (see Section 3.4), with a proper and well-tuned set of hyperparameters, could outperform most models (even the most recent ones). From their results, it has been questioned whether the performance improvements in recent models are due to architectural changes, or rather to hyperparameter tuning, opening up for reconsideration of how model performance should be evaluated and reported in future research.

Similarly, Ruffinelli, Broscheit, and Gemulla (2020) summarized and empirically quantified the impact of different model architectures and training strategies on model performance. To this end, they performed an extensive experimental study using popular KGE models across a wide range of hyperparameter settings in a common setup. Similarly to (Kadlec et al., 2017), they showed that when trained appropriately with the right set of hyperparameters, even the performance of early KGE models can be competitive or superior to that of more recent architectures. Moreover, they also found that a good model configuration can be found by exploring relatively few random samples from a large hyperparameter space through random search followed by Bayesian optimization. These studies have inspired our current work, in particular, the way in which we conducted hyperparameter tuning for all compared models (see Section 3.6).

Akrami, Saeef, Zhang, Hu, and Li (2020) conducted a systematic study with the main objective of assessing the true effectiveness of embedding models when inverted triples are removed[2]. Essentially, they illustrated the performance comparison of some representa-

2. (h, r, t) and (t, r^{-1}, h) are two inverted triples. Ex: *(avatar, directed by, James Cameron)* and *(James Cameron, director, Avatar)*.

tive KGE models on the popular datasets *WN18* and *FB15k* and their respective versions without reverse triples, i.e., *WN18RR* and *FB15k-237* (see Sections 3.1.2 and 3.1.1 for more details). The experimental results show that these models are much less accurate than what expected, and their poor accuracy renders *link prediction* a task without truly effective automated solutions.

Ali et al. (Ali et al., 2021a) performed a large-scale benchmark study on 21 KGE models over four datasets on the *link prediction* task. They performed several thousands of experiments and 24,804 GPU hours of computation time on link prediction task. Under a unified framework, they evaluated all the models based on different hyperparameters, training approaches, loss functions, optimizers, and the explicit modeling of inverse relations. Once more, they provided evidence that several architectures can obtain competitive results when configured carefully.

Further details on the datasets and KGE-based models considered in this study are reported in Sections 3.1 and 3.4 respectively.

To the best of our knowledge, our research represents the first benchmark study analyzing the performances and the behaviours, in the presence of incremental noise levels, of state-of-the-art KGE models in *link prediction*, *link deletion* and *triple classification* downstream tasks.

3. Experiments

In this Section, we describe the experimental setup and process we designed to answer our RQs (see Section 1).

To conduct our study, we relied on the *PyKEEN*[3] (Ali et al., 2021b) Framework. With respect to *PyKEEN* and other existing benchmarks (see Section 2), we additionally release:

- a new collection of datasets with an incremental amount of noisy triples;
- the code to reproduce all the steps of our experimental workflow, i.e., tuning, training, and testing for the three tasks of *link prediction*, *link deletion*, and *triple classification*.

All the resources (i.e., code, datasets, models, hyperparameters configurations) are made available to the research community at the following *url*: `https://github.com/stefanofaralli/noisy-kgs-benchmark/`. The repository includes the source code and a shared cloud folder[4] with datasets, optimal configurations and trained models. The entire experimentation was carried out on a dedicated cluster[5] managed through *HTCondor*[6] (Thain et al., 2005).

Finally, a number of additional fine-tuned experiments and information to ensure the reproducibility of all experiments are made available in the supplementary material, see Sections A and B, respectively.

3. *PyKEEN* (Python KnowlEdge EmbeddiNgs) is a Python package designed to train and evaluate Knowledge Graph Embedding models (`https://github.com/pykeen/pykeen`).
4. `https://drive.google.com/drive/folders/1h_B_0Kent6_F9j8xghKmgAejFF2vRyH-?usp=sharing`.
5. Experiments were considerably time-consuming, involving about two weeks of intensive computation on cluster nodes equipped with dedicated graphics processors.
6. *HTCondor* is an open-source software system that creates a High-Throughput Computing (HTC) environment. It is used to efficiently manage workload on a dedicated cluster of computers (`https://github.com/htcondor/htcondor`).

In the remaining of this Section, we describe the considered datasets (see Section 3.1 and Section 3.2 for a description of the original datasets and the generated noisy ones, respectively), the experimented models (see Section 3.4), the downstream tasks (see Section 3.5), and finally, the hyperparameters tuning methodology (see Section 3.6).

3.1 Datasets

We selected three heterogeneous datasets, thus enabling an evaluation of KGEs' performances with different graph dimensions, i.e., number of nodes and triples, density, and domain of interest. We selected the following KG datasets frequently used in the literature: *CoDEx Small* (see Section 3.1.3), *WN18RR* (see Section 3.1.2), and *FB15k-237* (see Section 3.1.1). The two latter datasets are an extension of *WN18* and *FB15k*, respectively. It was first noted by (Toutanova & Chen, 2015) that *WN18* and *FB15k* suffer from test leakage through inverse relations: a large number of test triples can be obtained simply by inverting triples in the training set. For example, the test set might contain a triple such as *(Bill Gates, founds, Microsoft)*, while the training set presents its inverse triple *(Microsoft, founded by, Bill Gates)*. This undesirable property can lead to performance overestimation. To create a dataset without this property, (Toutanova & Chen, 2015) and (Dettmers et al., 2018) respectively introduced *FB15k-237* and *WN18RR*, two improved versions of *FB15k* and *WN18RR* without inverse relations, more suitable for unbiased evaluation of various algorithms.

The three selected datasets are provided as sets of triples (*head*, *relation*, *tail*), hereafter (*h,r,l*), already divided into three splits: training, validation and testing. Further down, the datasets are described in more detail.

3.1.1 FB15k-237

FB15k-237 (Toutanova & Chen, 2015) is a link prediction dataset created from *FB15k* (Bordes et al., 2013). In the same manner of *WN18*, *FB15k* (Bordes et al., 2013) suffers from test leakage through inverse relations. For example, the test set frequently contains triples such as *(s, hyponym, o)* while the training set contains its inverse *(o, hypernym, s)*. To create a dataset without this property, (Toutanova & Chen, 2015) introduced *FB15k-237*. *FB15k-237* is essentially composed of asymmetric, unreflexive and intransitive relationships (see the example excerpt in Table 1).

3.1.2 WN18RR

WN18RR (Dettmers et al., 2018) is a link prediction dataset created from *WN18* (Bordes et al., 2013), which is a subset of *WordNet*[7] (a lexical knowledge base in which entities represent terms and are called synsets, and relations represent conceptual-semantic or lexical relationships) (Fellbaum, 1998). The *WN18RR* dataset was created to ensure that the evaluation dataset does not have inverse relation test leakage. From the distribution of relationship types, it appears that *WN18RR* is essentially composed of asymmetrical, transitive and irreflexive relationships (i.e., "_hypernym". and "_derivationally_related_from").

7. https://wordnet.princeton.edu/.

Table 1: Examples of asymmetric, unreflexive and intransitive relationships from *FB15k-237*.

/award/award_nominee/award_nominations./award/award_nomination/award_nominee
/film/film/release_date_s./film/film_regional_release_date/film_release_region
/award/award_nominee/award_nominations./award/award_nomination/award

3.1.3 CoDEx Small

CoDEx Small (Safavi & Koutra, 2020) is the downsized version of a set of datasets for knowledge graph completion extracted from *Wikidata*[8] and *Wikipedia*[9]. These datasets improve existing benchmarks for completing knowledge graphs with regard to scope and difficulty level. Regarding the scope, *CoDEx* includes three knowledge graphs (small, medium, and large) that vary both in size and structure. In this study, we tested the selected KGE models on *CoDEx small*. As pointed out by the authors, *CoDEx* datasets represent a remarkably difficult benchmark for *link prediction*, compared with *FB15k-237*, since *CoDEx* contains fewer skewed[10] and fixed-set[11] relationships. Furthermore, from the distribution of relationship types it appears that *CoDEx Small* is essentially composed of asymmetric, unreflexive and intransitive relationships (i.e., "occupation","dimplomatic_relation").

3.2 Generation of Incrementally Noisy Datasets

To specifically answer RQ2 (see Section 1) and assess the robustness of KGE models with noisy knowledge graphs, we started from the original datasets described in the previous Section and generated noisy versions by adding an incremental percentage of noisy triples.

To automatically generate a noisy triple $(\widehat{h}, \widehat{r}, \widehat{t})$ with a very high probability of not being true in the knowledge graph universe, we randomly sampled the head entity $\widehat{h} \in E$, the relation $\widehat{r} \in R$, and the tail entity $\widehat{t} \in E$. On the training set, we performed balanced random sampling; while on the validation and test sets we performed random sampling. We adopted this sampling strategy following (Yu et al., 2010) and (Park & Marcotte, 2011). According to the authors of these studies, during training, it is preferable to have a balanced subset to reduce the bias caused by the Pareto distribution of node degrees[12]. On the other side, during testing, it is preferable to have a closer-to-reality subset, so that the estimate of predictive performance can be safely assumed to generalize to the population level. We remark that rarely this problem is addressed in link prediction experiments, leading to biased performance estimations.

To address RQ2, for each of the three datasets, we generated four new noisy datasets (see Table 2 for a detailed structural analysis). The first has a low amount of noisy triples (i.e., 10%), the second a medium amount of noisy triples (i.e., 20%), the third a high amount

8. `https://www.wikidata.org/`.
9. `https://www.wikipedia.org/`.
10. A skewed relation has only one unique tail entity.
11. A fixed-set relation connects entities to fixed sets of values.
12. Unbalanced random sampling would sample with higher probability nodes with the largest degrees, thus inducing a bias during training.

of noisy triples (i.e., 30%) and finally a fourth completely random dataset where all the original triples have been replaced with noisy triples.

Table 2 also summarizes the main structural and semantic (number of different relations) properties of the datasets.

3.3 Datasets Comparison

From the general descriptions introduced in Section 3.1, and the measurement shown in Table 2, we observe the following facts:

- CoDEx Small is the densest graph (density 0.008837224), followed by FB15K-237 (density 0.0015191507). The less-dense graph is WN18RR (density 0.0000562817).
- The majority of triples in CoDEx Small and FB15K-237 are asymmetric, irreflexive and intransitive relationships, while in WN18RR the most frequent relationships are asymmetric, irreflexive and transitive, such as the "_hypernym" relations which provides a taxonomic backbone to the entire knowledge graph. Thus, WN18RR is more structured than the other datasets, which instead exhibit a typical "small world" network structure.
- WN18RR is the graph with the lowest average degree (2.28), while CoDEx Small and FB15k-237, when compared to WN18RR, have comparable average degree (i.e., 17.96 and 21.47).
- FB15K-237 has, by large, the highest number of different relationships (about 230 against 11 of WN18RR and around 40 of CoDEx Small).

3.4 Models

Knowledge Graph Embeddings (KGEs) are approaches to transform the nodes and the edges of a KG into a low dimensional continuous vector space that preserves various topological and structural features of the graph. By leveraging this embedded representation, it is possible to perform various sub-tasks on the KG, such as *link prediction*, *triple classification*, *entity recognition*, *clustering*, and *relation extraction* (Ji et al., 2022; Hogan et al., 2021). A typical KG embedding model generally consists of three steps: (*i*) representing entities and relations, (*ii*) defining a scoring function, and (*iii*) learning entity and relation representations (Wang et al., 2017). The first step specifies the form in which entities and relations are represented in a continuous vector space. Then, in the second step, a scoring function is defined on each triple (h, r, t) to measure its plausibility (observed triples in KG tend to have higher scores than unobserved triples). Finally, to learn embeddings representations, the third step solves an optimization problem that maximizes the total plausibility of all triples observed in the graph. Numerous KGE models have been proposed in the literature. As reported by (Wang et al., 2017), such embedding techniques can be broadly classified into two groups:

- ***translational models***: they exploit distance-based scoring functions, measuring the plausibility of a triple as the distance between the two entities, usually after a translation carried out by the relation. The basic idea is that relationships are interpreted as

Table 2: Structural analysis. For each dataset we report the density and for each dataset split (i.e., training, test and validation), we report the number of nodes, the number of relation types, and the total amount of resulting triples with the corresponding total of correct and wrong (noisy) triples.

Dataset	Density and Avg. Degree	Split	Nodes	Relation Types	Triples	Correct	Wrong (noisy)
CoDEx Small original	0.0088372241 17.96	train	2,034	42	32,888	32,888	0
		test	1,390	36	1,828	1,828	0
		validation	1,390	33	1,827	1,827	0
CoDEx Small 10% noisy	0.0097211158 19.76	train	2,034	42	36,177	32,888	3,289
		test	1,446	36	2,011	1,828	183
		validation	1,450	33	2,010	1,827	183
CoDEx Small 20% noisy	0.0106050075 21.56	train	2,034	42	39,466	32,888	6,578
		test	1,496	36	2,194	1,828	366
		validation	1,498	33	2,193	1,827	366
CoDEx Small 30% noisy	0.0114888992 23.35	train	2,034	42	42,755	32,888	9,867
		test	1,552	36	2,377	1,828	549
		validation	1,550	33	2,376	1,827	549
CoDEx Small random	0.0087356552 17.76	train	2,034	41	32,473	0	32,473
		test	1,407	36	1,824	0	1,824
		validation	1,391	34	1,826	0	1,826
WN18RR original	0.0000562817 2.28	train	40,559	11	86,835	86,835	0
		test	4,987	11	2,924	2,924	0
		validation	4,897	11	2,824	2,824	0
WN18RR 10% noisy	0.0000619109 2.51	train	40,559	11	95,519	86,835	8,684
		test	5,423	11	3,217	2,924	293
		validation	5,262	11	3,107	2,824	283
WN18RR 20% noisy	0.0000675383 2.73	train	40,559	11	104,202	86,835	17,367
		test	5,834	11	3,509	2,924	585
		validation	5,674	11	3,389	2,824	565
WN18RR 30% noisy	0.0000731675 2.97	train	40,559	11	112,886	86,835	26,051
		test	6,236	11	3,802	2,924	878
		validation	6,077	11	3,672	2,824	848
WN18RR random	0.0000687741 2.55	train	36,255	11	86,800	0	86,800
		test	5,056	11	2,924	0	2,924
		validation	4,835	11	2,824	0	2,824
FB15K237 original	0.0015191507 21.47	train	14,128	235	266,655	266,655	0
		test	10,064	222	19,902	19,902	0
		validation	9,571	221	17,074	17,074	0
FB15K237 10% noisy	0.0016710753 23.62	train	14,130	235	293,321	266,655	26,666
		test	10,438	230	21,893	19,902	1,991
		validation	9,951	232	18,782	17,074	1,708
FB15K237 20% noisy	0.0018229849 25.77	train	14,130	235	319,986	266,655	53,331
		test	10,762	233	23,883	19,902	3,981
		validation	10,288	234	20,489	17,074	3,415
FB15K237 30% noisy	0.0019749045 27.92	train	14,132	235	346,652	266,655	79,997
		test	11,026	234	25,873	19,902	5,971
		validation	10,541	235	22,197	17,074	5,123
FB15K237 random	0.0015497862 21.68	train	13,967	235	266,377	0	266,377
		test	10,415	235	19,898	0	19,898
		validation	10,415	235	17,071	0	19,898

translations in the embedding space: if (h, r, t) holds, then the embedding of the tail entity t should be close to the embedding of the head entity h plus some vector that

depends on the relationship (Bordes et al., 2013). All the models in this category are based on algorithms that train "some" combination of the head and relation vectors to be equal to the tail vector.

- ***semantic matching models***: they exploit similarity-based scoring functions, measuring the plausibility of triples by matching the latent semantics of entities and relations embodied in their vector space representations. Basically, these models are based on tensor factorization. The initial algorithm proposed using this technique was *RESCAL* (Nickel et al., 2011). In this model, first, a three-way tensor of size $n \times n \times m$ is constructed[13], containing value 1 when there is a relationship between two entities and value 0 otherwise. Afterward, the embeddings are calculated by factorizing the three-way tensor. Since this approach is computationally expensive for large graphs, various semantic matching models (*DistMult*, *HolE*, *ComplEx*, etc.) have been proposed in the literature to overcome this computational problem.

Additionally, there is a third category including models - i.e., *AutoSF* (see Section 3.4.8) and *ConvE* (see Section 3.4.6) models - not belonging to the previous categories. In the remaining of this paper, we will refer to the third category as ***others***, in line with previous comparative studies.

In the next Sections, we summarize the ten state-of-the-art KGE models we included in our benchmark study. In Table 3, we also provide an overview of the models by including a short description, the dataset and the downstream tasks experimented in the original studies.

3.4.1 *TransE* (translational)

Bordes et al. (2013) proposed *TransE* an energy-based method that models relations by interpreting them as translations in the embedding space. *TransE* embodies entities and relations as vectors in the same semantic space of dimension $\mathbb{R}^d$, where d is a hyper-parameter that represents the dimension of the target space with reduced dimension. The basic idea of this model is to make the sum of the head vector and relation vector as close as possible to the tail vector. In the original paper, *TransE* was tested for link prediction task on *WN18*, *FB15k* and *FB1M* datasets (Bordes et al., 2013).

3.4.2 *TransH* (translational)

Wang et al. (2014b) presented *TransH* a translational model that represents an evolution of *TransE*. *TransH* models a relation as a hyperplane together with a translation operation on it. In this way, without sacrificing efficiency in the process, this model is able to preserve the following mapping properties of relations: reflexive, one-to-many, many-to-one, and many-to-many. In the original paper, *TransH* was tested for link prediction, triple classification and relation extraction on the following datasets: *WN18*, *FB15k*, *FB13* (Socher et al., 2013), *WN11* (Socher et al., 2013), and *FB5M* (Wang et al., 2014b).

13. n is the number of entities and m is the number of relationships.

Table 3: Summary of the models included in our experimental setup. For each model, we report a brief description, the datasets and the downstream tasks experimented in the original publication.

model	**approach**	**datasets**	**downstream tasks**	**category**
TransE (Bordes et al., 2013)	models relationships by interpreting them as translations operating on the low-dimensional embeddings of the entities	WN18, FB15K, FB1M	*link prediction*	translational
TransH (Wang et al., 2014b)	models a relation as a hyperplane together with a translation operation on it	FB13, FB15k, FB5M, WN11, WN18	*link prediction*, *triple classification*, *relation extraction*	translational
DistMult (Yang et al., 2015)	a bilinear model based on the factorization of a three-way tensor	WN18, FB15k	*link prediction*, *rule extraction*	semantic matching
ComlpEx (Trouillon et al., 2016)	based on latent factorization that uses complex-valued embeddings and the Hermitian dot product	WN18, FB15k	*link prediction*	semantic matching
HolE (Nickel et al., 2016)	based on holographic embeddings and the idea of circular correlation	Countries, FB15k, WN18	*link prediction*	semantic matching
ConvE (Dettmers et al., 2018)	multi-layer convolutional network model able to scale on large KGs and learn expressive feature	Countries, FB15k, FB15k-237, WN18, WN18RR, YAGO3-10	*link prediction*	other
RotatE (Sun et al., 2019)	defines each relation as a rotation from the source entity to the target entity in the complex vector space	FB15k, FB15k-237, WN18, WN18RR	*link prediction*	translational
AutoSF (Zhang et al., 2020)	automatically designs scoring functions (SFs) for distinct KGs by the AutoML techniques	WN18, FB15k, WN18RR, FB15k237, YAGO3-10	*link prediction*	other
BoxE (Abboud et al., 2020)	embeds entities as points, and relations as a set of hyper-rectangles (or boxes), which spatially characterize basic logical properties	FB15k-237, WN18RR, YAGO3-10, JF-17K, SportsNELL	*knowledge graph completion*, *knowledge base completion*, *rule injection*	translational
PairRE (Chao et al., 2021)	a model with paired vectors for each relation representation. The paired vectors enable an adaptive adjustment of the margin in loss function to fit for complex relations	ogbl-wikikg2, ogbl-biokg, FB15k, FB15k-237, DB100k, SportsNELL	*link prediction*	translational

3.4.3 *DistMult* (semantic matching)

Yang et al. (2015) designed *DistMult*, a bilinear model based on the factorization of a three-way tensor. It captures the latent semantics of a knowledge graph by associating entities with vectors. This method represents each relation as a single matrix that models pairwise interaction between entities. It simplifies *RESCAL* (Nickel et al., 2011) by restricting the relations matrix from a general asymmetric and $r \times r$ matrix [14] to a diagonal square matrix, thus reducing the number of parameters per relation. In the original paper, *DistMult* was tested for link prediction and rule extraction tasks on *WN18* and *FB15k* datasets.

14. r represents the number of relations in the KG.

3.4.4 *ComlpEx* (semantic matching)

Trouillon et al. (2016) presented *ComlpEx* a method based on latent factorization that uses complex-valued embeddings and the Hermitian dot product (the complex counterpart of the standard dot product between real vectors). *ComlpEx* remains linear in both space and time. Moreover, The composition of complex embeddings can handle a large variety of binary relations, among them symmetric and anti-symmetric relations. In the original paper, *ComlpEx* was tested for link prediction task on *WN18* and *FB15k* datasets.

3.4.5 *HolE* (semantic matching)

Nickel et al. (2016) proposed *HolE* a model based on holographic embeddings and the idea of circular correlation. It learns compositional vector space representations from the entire knowledge graphs. This method is related to holographic models of associative memory in that it employs circular correlation to create compositional representations. By using correlation as the compositional operator, *HolE* can capture rich interactions but at the same time, it remains efficient to compute, easy to train, and scalable to large datasets. In the original paper, *HolE* was tested for link prediction task on *WN18*, *FB15k* and *Countries* (Bouchard et al., 2015) datasets.

3.4.6 *ConvE* (other)

Dettmers et al. (2018) proposed *ConvE* a multi-layer convolutional neural network able to scale on large KGs and learn expressive features. *ConvE* reshapes the numerical representations of entities and relations in the form of an image and then applies the convolution filters to extract the features, thus learning the final embeddings. This model is highly parameter-efficient, yielding the same performance of *DistMult* with 8x fewer parameters. Moreover, it is particularly effective at modeling nodes with high in-degree (these types of nodes are very common in complex and highly connected KGs). In the original paper, *ConvE* was tested for link prediction task on the following datasets: *WN18*, *WN18RR*, *FB15k*, *FB15k-237*, *Countries*, and *YAGO3-10* (Mahdisoltani et al., 2015). As remarked by the authors, in contrast to other models, ConvE requires explicit samples of symmetric triples in the training dataset. *In our benchmarks, we do not provide such additional information, to avoid a positive bias toward this system.*

3.4.7 *RotatE* (translational)

Sun et al. (2019) developed *RotatE* a method for generating graph embeddings that can model and infer various relation patterns, including: symmetry/anti-symmetry, inversion, and composition. Specifically, the *RotatE* model defines each relation as a rotation from the source entity to the target entity in the complex vector space. The *RotatE* model is trained using a self-adversarial negative sampling technique. Inspired by translational models, *RotatE* maps entity representations into complex vector space. The model is motivated by Euler's identity and defines relations as rotation from head to tail. In the original paper, *RotatE* was tested for link prediction task on the following datasets: *WN18*, *WN18RR*, *FB15k* and *FB15k-237*.

3.4.8 *AutoSF*(other)

Zhang et al. (2020) proposed *AutoSF* a method inspired by the recent success of Automated Machine Learning (AutoML)[15]. It automatically designs *scoring functions* (SFs) for distinct KGs by the AutoML techniques. Many SFs, which target capturing different kinds of relations in KGs, have been designed by humans in recent years. However, as relations can exhibit complex patterns that are hard to infer before training, none of them can consistently perform better than others on existing benchmark data sets. Firstly, *AutoSF* set up a search space over popularly used SFs. Then, through a greedy algorithm, it searches in this space efficiently. In the original paper, *AutoSF* was tested for link prediction task on the following datasets: *WN18*, *WN18RR*, *FB15k*, *FB15k-237*, and *YAGO3-10*.

3.4.9 *BoxE*(translational)

Abboud et al. (2020) presented *BoxE* a spatial-translational model that embeds entities as points, and relations as a set of hyper-rectangles (or boxes), which spatially characterize basic logical properties. Facts are scored based on the positions of entity embeddings with respect to relation boxes. This apparently simple abstraction yields a fully expressive model offering a natural encoding for many desired logical properties. *BoxE* simultaneously addresses all the following limitations: theoretical non-expressivity, lack of support for prominent inference patterns (e.g., hierarchies), and lack of support for incorporating logical rules. In the original paper, *BoxE* was tested for link prediction and rule injection tasks on the following datasets: *WN18RR*, *FB15k-237*, *SportsNELL* (Wang et al., 2015), and *JF17K* (Wen et al., 2016).

3.4.10 *PairRE*(translational)

Chao et al. (2021) designed *PairRE* a model with two (paired) vectors for each relation representation. These two vectors project the corresponding head and tail entities to Euclidean space, where the distance between the projected vectors is minimized. The paired vectors enable an adaptive adjustment of the margin in the loss function to fit different complex relations. Besides, *PairRE* is capable of encoding three important relation patterns, symmetry/antisymmetry, inverse and composition. Given simple constraints on relation representations, *PairRE* can encode subrelation further. In the original paper, *PairRE* was tested for link prediction task on the following datasets: *FB15k*, *FB15k-237*, *SportsNELL*, *ogbl-wikikg2* (Hu et al., 2020), *ogbl-biokg* (Hu et al., 2020), and *DB100K* (Ding et al., 2018).

3.5 Downstream Tasks

Once the KGE models were trained, we evaluated their performance in three different tasks: *link prediction* (see Section 3.5.1), *link deletion* (see Section 3.5.2) and *triple classification*(see Section 3.5.3).

The three tasks have been included in our experimental setting, to assess the robustness of the application of KGE models in the presence of noisy triples (see RQ2 in Section 1). Moreover, the downstream tasks of *link prediction* and *link deletion* have been included

15. `https://www.automl.org/`

to specifically address the RQ1 (see Section 1) related to analyzing the interoperability of these two tasks.

The evaluation has been performed on test sets consisting of triples not included in the training phase. Hereafter, the three tasks analyzed are described in more detail.

3.5.1 Link Prediction

The task of link prediction is defined in literature as predicting the missing head entity h in a triple $(?, r, t)$ or the missing tail entity t in a triple $(h, r, ?)$.

On the *evaluation* side, for each test triple (h, r, t), the head entity h is replaced with every other entity $h' \in E$ in the dataset, to form corrupted triples. The original test triple and its corresponding corrupted triples are ranked by their scores according to the score functions and the rank of the original test triple is denoted $rank_h$. The same procedure is used to calculate $rank_t$ for the tail entity t. A method with the ideal ranking function should rank the test triple at the top.

Usually, the accuracy of different KGE models is measured using ranking *metrics* like: Hits at K ($Hits@1$, $Hits@3$, $Hits@5$, $Hits@10$), Mean Rank (MR), and Mean Reciprocal Rank (MRR)):

- $Hits@K$ is the percentage of top K results that are correct;
- MR is the mean of the test triples' ranks;
- MRR is the average inverse of the harmonic mean of the test triples' ranks.

See (Akrami et al., 2020) for a formal definition of these metrics. We note that, by definition, higher $Hits@K$ and MRR indicate better link prediction accuracy; while MR is a decreasing metric (the lower the better).

3.5.2 Link Deletion

Link deletion represents the symmetric task of *link prediction*. In the literature, almost all KGE models are evaluated on the *link prediction* task (see Table 3). However, in this study, to answer the second research question, we included in our analysis the *link deletion* task, defined as the detection of triples with the wrong head entity $(\widehat{h}, r, t)$ or the incorrect tail entity $(h, r, \widehat{t})$.

Regarding the *evaluation*, for each test triple (h, r, t): the correct head entity h is replaced with a "fake" (wrong) head entity $\widehat{h} \in E$, to form a synthetic corrupted triple $(\widehat{h}, r, t)$; afterward, this synthetic fake triple and all the original correct test triples are ranked by their scores according to the KGE model score function; the rank of the fake test triple is denoted $rank_h$. In analogy with link prediction, systems should rank the wrong triple with the highest score. The same procedure is used to calculate $rank_t$ for the tail entity t.

For this task, the accuracy of different KGE models can be measured using the same ranking *metrics* of the link prediction task ($Hits@K$, MR, MRR).

3.5.3 Triple Classification

The *triple classification* task is used to decide the boolean truth value of an unknown input triple, and is therefore considered much simpler that the previous two tasks. Given an input triple, the trained KGE model evaluates the plausibility of the triple to determine if this triple is true (it belongs to the underlying KG) or false. The decision is made with the KGE model score function and a given threshold. *Note that triple classification represents a much simpler task with respect to link prediction and link deletion*, since it is a binary classification problem. On the *evaluation* side, we used a balanced test set (50% of correct triples and 50% of fake triples generated with random permutations) and the standard classification *metrics* ($F1_{macro}$, $F1_{negative}$, $F1_{positive}$). Moreover, as an additional metric, we considered the *normalized distance* between the mean of the correct testing triples scores and the mean of the fake testing triples scores as a measure of the degree of separability (see an example in Figure 1). As a normalization factor, we considered the difference between the maximum score calculated on the correct training triples and the minimum score computed on the fake testing triples. Let T be the scores associated with the true (real) triples of a specific partition (*training*, *validation*, *testing*) and F be the scores associated with the noisy (wrong) triples of a target partition, then the *normalized distance* can be defined in the following way:

$$norm\_distance = \frac{|mean\{T_{testing}\} - mean\{F_{testing}\}|}{max\{T_{training}\} - min\{F_{testing}\}}$$

3.6 Hyperparameters Tuning

Building on previous results of link prediction benchmark studies (see Section 2), emphasizing the fact that often simplest models with a proper and well-tuned set of hyperparameters could outperform the most recent and complex models (Kadlec et al., 2017), (Ruffinelli et al., 2020), (Ali et al., 2021a), we performed extensive hyperparameters tuning phase. For all the considered KGE methods (see Section 3.4), we applied a methodological hyperparameters optimization strategy. To this aim, we exploited the hyperparameter optimization pipeline provided by *PyKEEN* (Ali et al., 2021b), which in turn, it is based on *Optuna* (Akiba et al., 2019)[16]. Every model in *PyKEEN* has default values for its hyperparameters, chosen from the best-reported values in each model's original paper. Determining the optimal hyperparameters is not an easy task and requires exploring a complex search area with multiple experiments. Thus, *PyKEEN* tries to find a good set of hyperparameters in a limited number of trials, thanks to Sequential Model-Based Optimization (SMBO) with Tree of Parzen Estimators (TPE) (Bergstra et al., 2011). For each optimization (target model for a specific dataset), we ran a simulation consisting of 25 total trials (the initial start-up phase involved 18 random trials) with a 70th percentile pruner. We remark that, for *ConvE* on the *FB15k-237* dataset, we did not find a set of hyperparameters capable of returning a convergent model with significant results.

16. *Optuna* is an open source optimization framework provided as Python library to automate hyperparameters search for black-box functions (`https://github.com/optuna/optuna`).

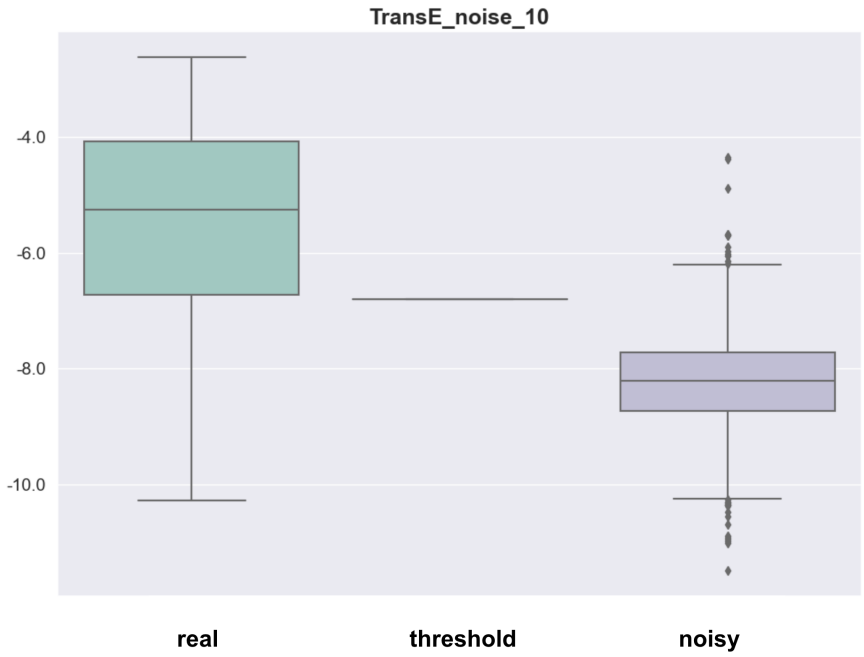

Figure 1: Diagram showing an example of the distance between the scores of test true triples and test noisy triples for *TransE* model on *WN18RR* dataset with 10% noise.

In the shared remote folder[17], the interested reader can access the JSON files providing detailed information such as optimal parameters identified, computation time, scores according to various metrics, etc.

4. Results

In this Section, we summarize the results obtained from all the experiments conducted in our benchmark study. Additional experiments are reported in the Supplementary Material (Sections A and B).

To make it easier to find and interpret clear performance patterns, each table or graph presented in this Section focuses on only one or two comparison dimensions, among the four considered in this study: i) structure of the dataset, ii) type of task, iii) amount of injected noise during training, iv) category of predictive methods.

4.1 Comparing Link Prediction vs Link Deletion Performance of Individual Systems on Different Datasets

We begin with Table 4, which helps answer RQ 1: to what extent do systems that perform well on LP perform equally well on LD? The Table shows a summary excerpt of the per-

17. `https://drive.google.com/drive/folders/1h_B_0Kent6_F9j8xghKmgAejFF2vRyH-?usp=sharing`.

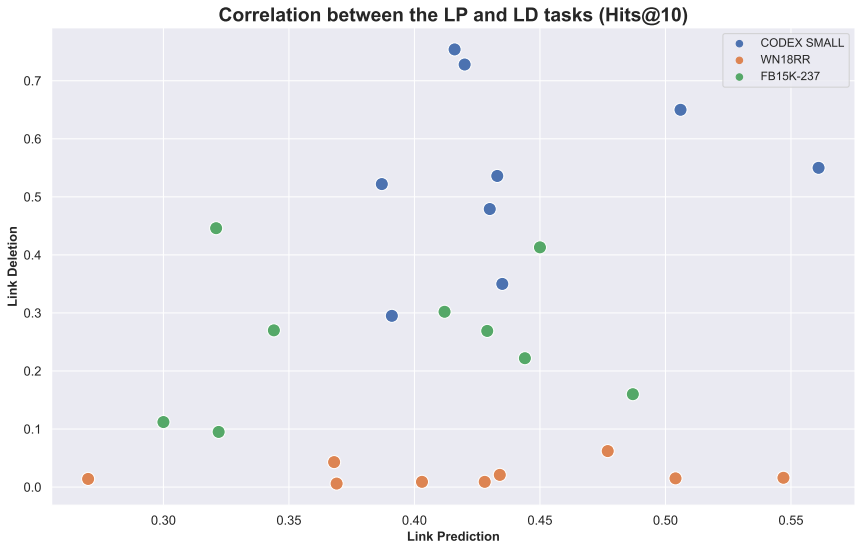

Figure 2: Scatter plot depicting the correlation between the performance (Hits@10) of the LP and LD tasks.

formances obtained by the selected KGE models (see Section 3.4) in the three downstream tasks of Link Prediction (LP), Link Deletion (LD) and Triple Classification (TC), and the three datasets.

The Table does not consider the effect of increasing levels of noisy relations and shows only one performance measure (hits@10 for LP and LD, and f1_macro for TP). The corresponding extended tables with additional measures and increasing levels of noise, along with more detailed discussions, are available in the Supplementary Material (Tables 1, 2, 3, Sections A.1 and A.2).

Analyzing the Table, we observe the following:

1. The majority of systems individually exhibit similar performance in LP and TC across different datasets, as shown in the LP table, with a slight decrease with *WN18RR*.

2. Contrarily, for LD we observe strong differences: for several systems, the performance may even increase w.r.t. LP with *CoDEx*, but they decrease with *FB15k-237* and considerably decrease with *WN18RR*.

3. Comparatively, systems with higher performance in LP and TC, are also the best in LD. However, we note that performance in LP and LD for all systems are not statistically correlated, as shown in Figure 2.

4. In general, the best systems are *translational*, in particular *RotatE*, *TransE*, *TransH*.

Table 4: Task-wise and dataset-wise performance of all systems when trained with the original (uncorrupted) datasets. As remarked in Section 3.6 it was not possible to find an appropriate set of hyperparameters for *ConvE* on *FB15k-237*.

	AutoSF	**BoxE**	**ComplEx**	**ConvE**	**DistMult**	**HolE**	**PairRE**	**RotatE**	**TransE**	**TransH**
Link Prediction										
CoDEx Small ***hits@10***	.420	.391	.433	.286	.387	.435	.430	**.561**	.506	.416
WN18RR ***hits@10***	.403	.504	.369	.439	.270	.428	.434	**.547**	.477	.368
FB15k-237 ***hits@10***	.412	.444	.300	n.a.	.321	.322	.429	**.487**	.344	.450
Link Deletion										
CoDEx Small ***hits@10***	.728	.295	.536	.648	.522	.350	.479	.550	.650	**.754**
WN18RR ***hits@10***	.009	.015	.006	.018	.014	.009	.021	.016	**.062**	.043
FB15k-237 ***hits@10***	.302	.222	.112	n.a.	**.446**	.095	.269	.160	.270	.413
Triple Classification										
CoDEx Small ***f1_macro***	.928	.928	.890	.695	.904	.879	.906	.908	**.937**	.932
WN18RR ***f1_macro***	.716	.795	.689	.801	.663	.786	.780	.801	**.889**	.754
FB15k-237 ***f1_macro***	.972	.974	.932	n.a.	.975	.943	.965	.943	.976	**.981**

To interpret these results[18], we first recall the main properties of each dataset (see Section 3.3): *CoDEx* has the highest density, *FB1k-237* the highest number of different relations, and *WN18RR* has a prevalence of transitive relations (such as hypernymy). The taxonomic structure induced on the hypernymy relations produces a single contribution on the degree of nodes, making the observations "scattered" on single pairs of nodes. It follows that introducing wrong relationships in *WN18RR* may have a disruptive impact on its hierarchical structures, making the task of identifying wrong relationships much more complicated, especially since all systems rely mainly on an embedded representation of local, rather than global, topological properties.

We further note that translational models, *TransE* and *TransH* in particular, show comparatively better performance in LD on *WN18RR* (although still degraded w.r.t. LP). Our interpretation is that, by forcing a combination of the h and r vectors to be similar to the t vector, these methods impose a directionality to the learned embeddings, which seems to be helpful for representing transitive relations. Finally, all systems perform better in all tasks with the *CoDEx* dataset. The ability of the various systems to capture local structural properties of the network is not hindered, but rather favored, by the high density and *small world* structure of *CoDEx*.

4.2 Analysing Performances of All Tasks in the Presence of Increasing Levels of Noise

Next, we face the second objective of our investigation: (RQ2) how do the various systems behave in the presence of increasing levels of noise injected in the training data?

In Figure 3 we show, for all translational systems, tasks and datasets, the changes in performance for *hits@10* (LP and LD) and *f1_macro* (TC). Figure 4 shows the same data for semantic matching and other systems.

The Figures highlight two results:

18. Note that the considerations of this paragraph apply also to the other performance measures not shown in Table 4. The interested reader may inspect the complete tables in Appendix A, Tables 1, 2, 3.

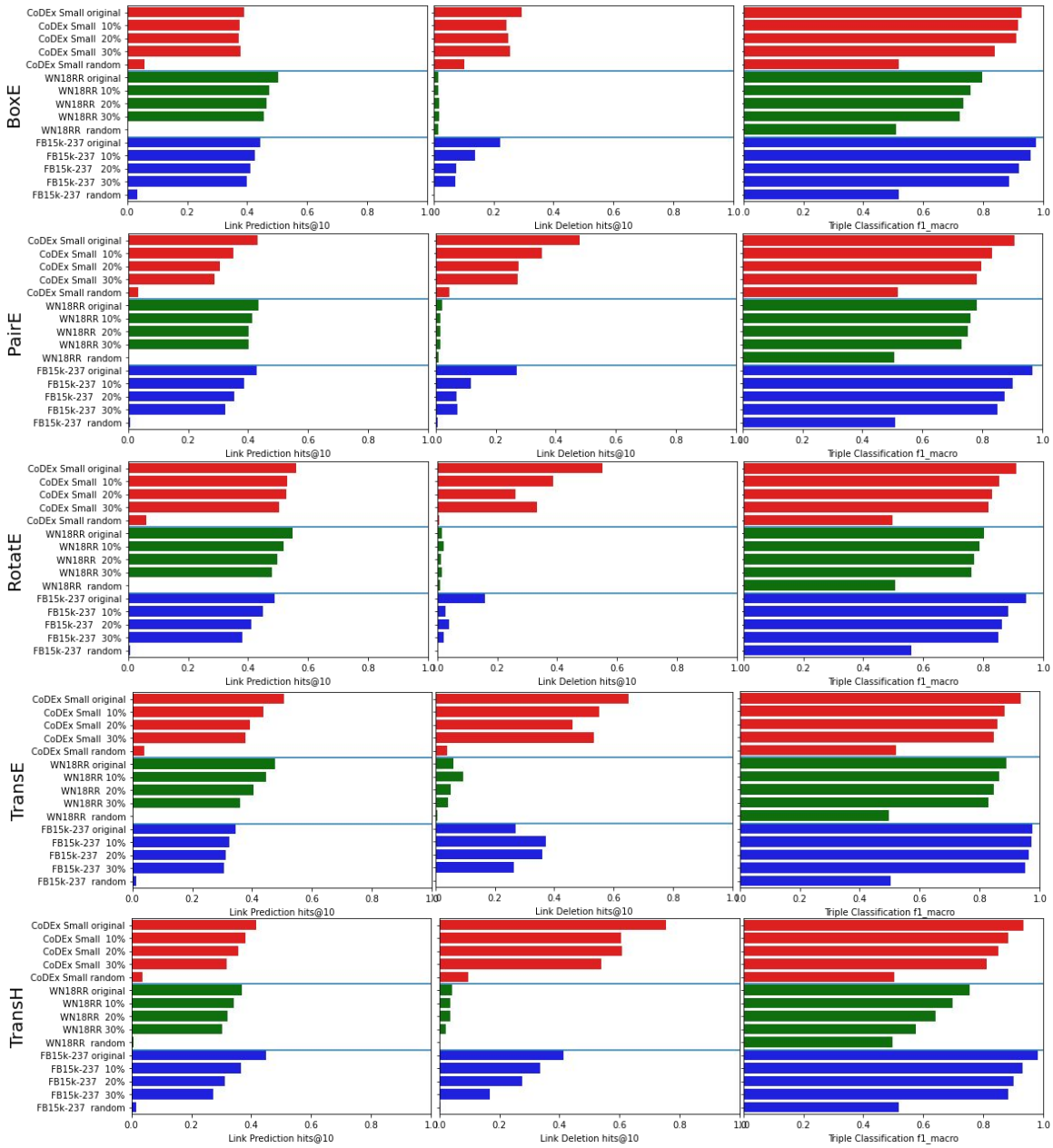

Figure 3: Performance of *translational* systems with increasing levels of noise in training data.

1. In general, a performance decay is evident but not particularly strong for almost all systems, tasks and datasets.

2. However, Link Deletion in general, and particularly on *WN18RR*, represents an experimental context in which all systems suffer the most from noise.

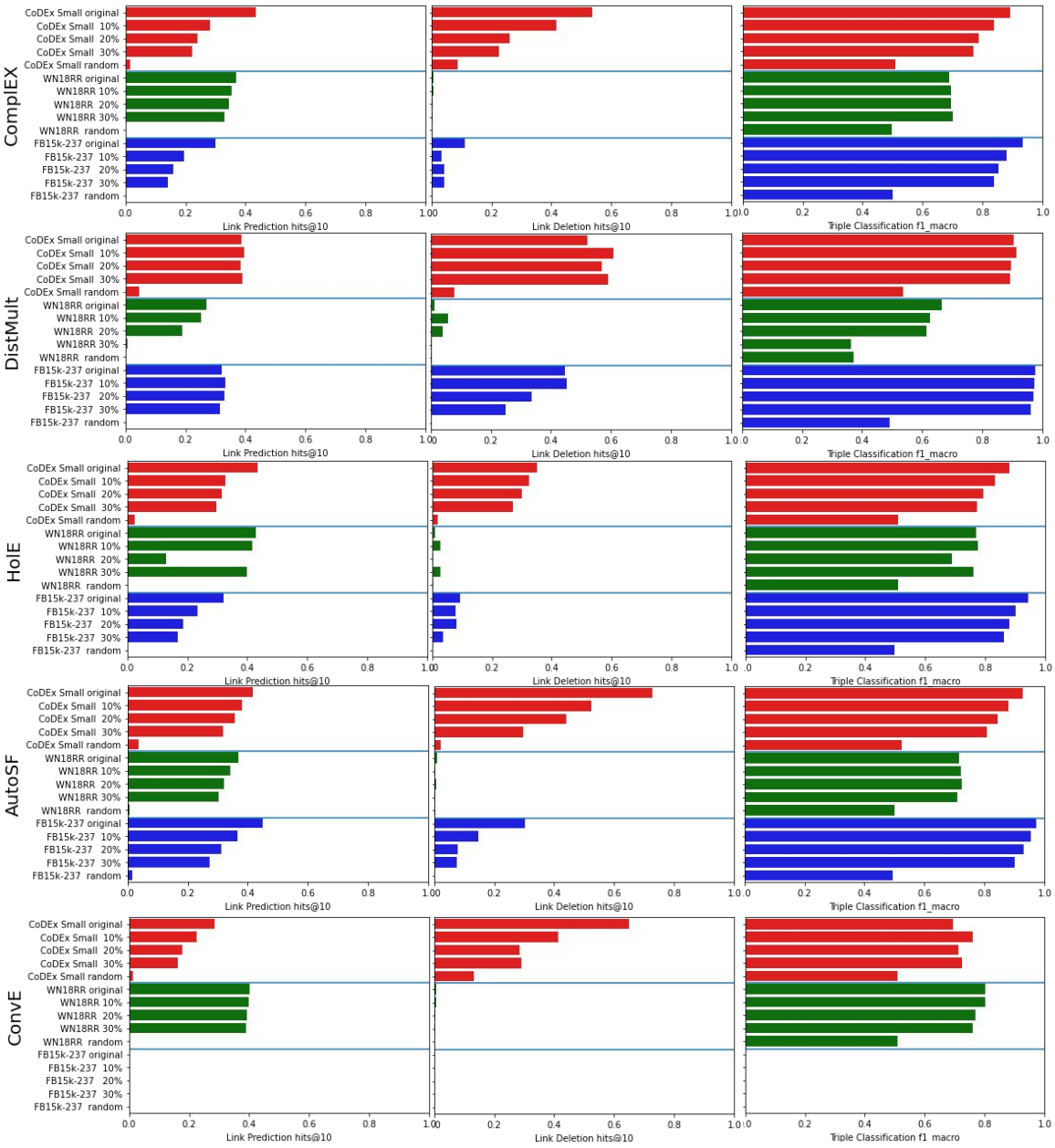

Figure 4: Performance of *semantic matching* and *other* systems with increasing levels of noise in training data. For *ConvE* on the *FB15k-237* dataset, we did not find a set of hyperparameters capable of returning a convergent model with significant results. The results of this model on dataset *FB15k-237* are not reported.

We also observe that, apart from the case of *WN18RR*, *DistMult* is much more robust to noise than all the other systems, in all tasks, to the point that in some cases, performance also increases with noise (see Figure 4, the second row of histograms). To shed more light

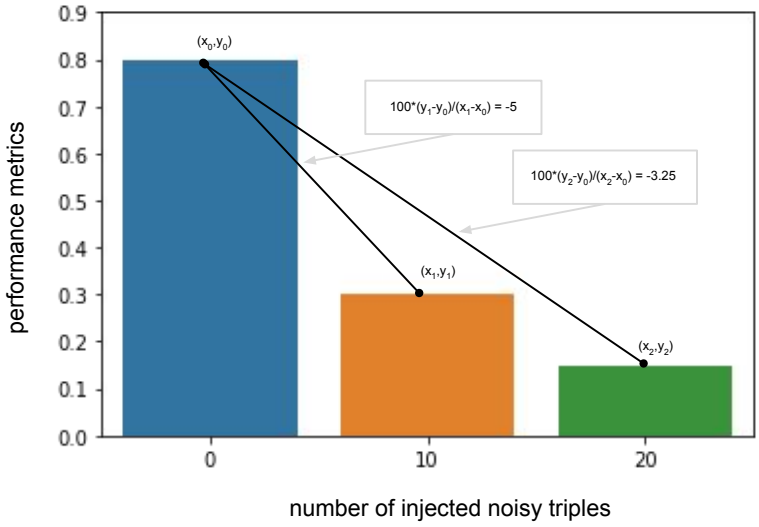

Figure 5: Example of the percentage slopes computed between the performance obtained with a dataset originally containing 0 noisy triples and then incrementally augmented with 10, and 20 noisy triples.

on this and other aspects, in the next Section, we move on to a comparative analysis of all the systems.

To summarize the performances of all systems also with respect to the injection of noise, we introduce the notion of *slopes*. For *hits@10* and *f1_macro*, we compute the percentage slope of the resulting performance obtained with the addition of noisy triples, with respect to the original dataset. The percentage slope is obtained as:

$$100 \times (\frac{y_1 - y_0}{x_1 - x_0}) \tag{1}$$

where: y_0 and y_1 are the performance measures obtained in the original and the noisy datasets, respectively, and x_0, x_1 are the amounts of noisy triples (if x_0 refers to the original dataset, then $x_0 = 0$). The intuition of this formula is depicted in Figure 5: the slopes are the angular coefficients of the lines shown in the Figure since they are always calculated with reference to the original dataset.

The results of this analysis are shown in Tables 5, 6 and 7. Colors in all Tables follow the so-called "objective and key result" (OKR)[19] scale, partitioning the performance in three levels: best (green: 1st, 2nd, 3rd ranks) middle (yellow: 4th, 5th, 6th and 7th ranks) and worse (red: 8th, 9th and 10th ranks). Note that, since systems in the "other" category (*AutoSF* and *ConvE*) have near-to-zero performance in some tasks and datasets (see Figure 4), they have been eliminated from the tables[20].

19. OKR scales are commonly used by very successful companies: `https://www.whatmatters.com/faqs/how-to-grade-okrs` `https://conceptboard.com/blog/okr-google-goal-setting-success/`.
20. With such bad performance, the slope is also close to zero, which would be misleading.

Table 5: Average slopes of all systems, aggregated task-wise.

	BoxE	ComplEx	DistMult	HolE	PairRE	RotatE	TransE	TransH
Link Pred	3.1	6.3	**2.9**	6.1	5.4	4.8	5.6	5.7
Triple Clf	4.8	4.5	**3.4**	4.8	7.2	5.5	4.2	7.3
Link Del	3.4	4.2	2.6	**2.4**	6	5.3	3.5	6.6

Table 6: Average slopes of all systems, aggregated dataset-wise.

	BoxE	ComplEx	DistMult	HolE	PairRE	RotatE	TransE	TransH
CoDEx Small	2.3	7.3	**1.2**	5	7.4	5	5	5.3
WN18RR	5.2	**2.4**	5.8	3.9	4.7	5.1	6.6	7.6
FB15k-237	3.8	5.3	1.9	4.4	6.5	5.5	**1.7**	6.7

Concerning slopes, which summarize resistance to noise, we note that translational models lose their primacy. The reason is that these systems project the representation of relationships onto a latent, dense space. Such compact representation, although it increases the robustness of translational models with respect to different tasks and features of the datasets, makes them more fragile in the presence of noise during training. In contrast, more scattered representations, such as those of semantic matching models, suffer less from noisy data.

Table 7: Average Ranking of Slopes on KGE Models families.

	semantic matching models	translation models
Link Pred	5.1	**4.92**
Triple Clf	**4.23**	5.8
Link Del	**3.07**	4.96
CoDEx Small	**4.5**	5
WN18RR	**4.03**	5.84
FB15k-237	**3.87**	4.84

4.3 Systems Comparison Along Different Dimensions

In the previous Sections, we analyzed the changes in the performance of individual models when changing the datasets, tasks and noise level in the training data. In order to answer RQ3 (are there systems with a clear competitive advantage, in which contexts, and why?), we now concentrate on a comparative analysis of the models. To facilitate comparative analysis, we tried to summarize all the rankings of the models in the various experiments in a single measure. For this purpose, for each model, we aggregated their positions in the final ranking. To do so, we computed the arithmetic mean among the rankings r_j^i obtained with the considered model i in the three different tasks LP, LD and TC for a subset of selected metrics. For example, if a system ranked 1^{st} in the first experiment, 4^{th} in a second, etc., its final rank over K experiments is:

Table 8: Aggregated ranking on the CoDEx Small dataset, all tasks.

	original	**noise 10%**	**noise 20%**	**noise 30%**
1°	TransE (1.542)	DistMult (1.667)	DistMult (1.667)	DistMult (1.208)
2°	TransH (2.292)	TransE (2.333)	TransE (2.083)	TransE (2.125)
3°	RotatE (2.833)	TransH (2.875)	TransH (2.792)	RotatE (2.792)
4°	AutoSF (3.458)	RotatE (2.917)	BoxE (3.083)	TransH (2.958)
5°	PairRE (4.667)	AutoSF (3.542)	RotatE (3.25)	BoxE (3.333)
6°	BoxE (4.75)	BoxE (3.833)	AutoSF (4.0)	AutoSF (4.583)
7°	DistMult (4.792)	ComplEx (5.458)	HolE (5.625)	HolE (5.542)
8°	ComplEx (4.917)	HolE (5.792)	PairRE (5.708)	PairRE (5.708)
9°	HolE (5.25)	PairRE (6.0)	ComplEx (6.125)	ComplEx (6.333)
10°	ConvE (6.417)	ConvE (6.75)	ConvE (6.833)	ConvE (6.667)

$$\sum_{j=1}^{K} \left(\frac{r_j^i}{K} \right) i = 1 \dots M \tag{2}$$

where M is the number of different compared models.

We selected *MR*, *MRR* and *HITS@10* for the *link prediction* and *link deletion* tasks; while, we considered *f1_macro* and *norm_dist* for the *triple classification* task. The resulting rankings achieved with the aggregated position scores are shown in the Tables 8, 9 and 10 for the *CoDEx Small*, *WN18RR* and *FB15k-237* datasets, respectively.

Table 9: Aggregated ranking on the WN18RR dataset, all tasks.

	original	**noise 10%**	**noise 20%**	**noise 30%**
1°	TransE (1.5)	TransE (1.583)	TransE (1.417)	TransE (1.833)
2°	RotatE (2.333)	RotatE (2.042)	RotatE (2.0)	RotatE (2.167)
3°	BoxE (2.583)	BoxE (3.292)	BoxE (2.75)	HolE (2.375)
4°	ConvE (3.083)	HolE (3.458)	PairRE (2.917)	PairRE (2.958)
5°	PairRE (3.417)	PairRE (3.625)	ConvE (4.292)	BoxE (3.292)
6°	HolE (4.417)	ConvE (3.833)	AutoSF (4.625)	ConvE (4.333)
7°	TransH (4.708)	TransH (5.25)	TransH (5.083)	AutoSF (5.125)
8°	AutoSF (5.625)	AutoSF (5.5)	ComplEx (5.542)	TransH (5.583)
9°	ComplEx (6.583)	DistMult (6.0)	DistMult (5.875)	ComplEx (5.917)
10°	DistMult (6.75)	ComplEx (6.333)	HolE (6.583)	DistMult (7.417)

The three tables clearly show that the best systems are *TransE* (translational) and *DistMult* (semantic), followed by other translational models, such as *RotatE* and *BoxE*. We also note that *DistMult* performs quite badly on the *WN18RR* dataset while it ranks 1^{st} for six times in the other two datasets, but always in the presence of noise. Our interpretation is the following: both *CoDEx* and *FB15k-237* have a very high number of different relationships and a "small world", local structure. The second property favors all models, on the contrary, the first penalizes all systems, except *DistMult*, the only semantic matching method in which relations are represented as diagonal matrices (see Section 3.4.3). Using

Table 10: Aggregated ranking on the FB15k-237 dataset, all tasks.

	original	**noise 10%**	**noise 20%**	**noise 30%**
1°	TransH (1.708)	DistMult (2.0)	DistMult (1.833)	DistMult (1.833)
2°	BoxE (2.708)	TransE (2.292)	TransE (2.042)	TransE (1.958)
3°	PairRE (3.208)	BoxE (2.5)	BoxE (2.75)	BoxE (2.75)
4°	DistMult (3.25)	TransH (2.792)	TransH (2.958)	TransH (3.125)
5°	TransE (3.417)	AutoSF (3.542)	AutoSF (3.5)	AutoSF (3.5)
6°	RotatE (3.667)	PairRE (4.208)	PairRE (4.375)	PairRE (4.125)
7°	AutoSF (3.75)	RotatE (4.75)	RotatE (4.917)	RotatE (4.833)
8°	HolE (5.625)	HolE (5.375)	HolE (5.125)	HolE (5.458)
9°	ComplEx (6.417)	ComplEx (6.167)	ComplEx (6.167)	ComplEx (6.0)

sparse representations to model KG relations in the presence of noise can offer several advantages, among which is robustness to noise and improved generalization. As also noted in (Ahmad & Scheinkman, 2019) (among others), by emphasizing sparse patterns, a model can filter out irrelevant or noisy information. Furthermore, sparse representations can help improve generalization capabilities by focusing on the most discriminative features of each relation, rather than relying on all available information. This can lead to better predictive performance, especially when dealing with noisy or incomplete data. Concerning the lower performance of semantic matching systems with *WN18RR*, this has been already discussed in Section 4.1.

Getting back to Section 4.2, where we put the magnifying glass on the noise sensitivity issue, what we just observed on *DistMult* also explains its relatively stable, and sometimes increased performance under noise, previously observed in Figure 4.

Next, to further aggregate our experimental results, we grouped the performance of all systems first, task-wise (Table 11), then, dataset-wise (Table 12), and finally, category-wise, according to model families (Table 13). The experiments reported in these Tables have been conducted on the original datasets, without noise.

Table 11: Average rankings of all systems, aggregated task-wise

	AutoSF	**BoxE**	**ComplEx**	**ConvE**	**DistMult**	**HolE**	**PairRE**	**RotatE**	**TransE**	**TransH**
Link Pred	5.667	4.444	5.778	6.556	6.889	5.111	4.556	**3.667**	5.667	6.444
Triple Clf	6.667	2.5	8.333	8	5.833	6.833	5.5	5.333	**1.833**	4.167
Link Del	5.222	6.222	6.111	5	4.222	7.444	4.778	7,333	4,556	**3.556**

Table 12: Average rankings of all systems, aggregated dataset-wise

	AutoSF	**BoxE**	**ComplEx**	**ConvE**	**DistMult**	**HolE**	**PairRE**	**RotatE**	**TransE**	**TransH**
CoDeX_Small	5.75	5.125	5.75	6.125	6.25	7.125	5	5.5	4.125	**3.75**
WN18RR	6.25	4.5	7	5.125	6.625	5.75	**4.25**	4.625	4.375	6.125
FB15k-237	5.25	4.25	6.875	7.75	**4**	6.375	5.375	6.25	4.375	4.5

The results come out rather clearly from the observation of these Tables, especially from Table 13, and summarize, in a compact way, what we already observed in previous Sections. Translational systems, and in particular *TransE*, are the best-performing ones in

Table 13: Average Ranking on KGE Models families

	semantic matching models	translation models	other models
Link Pred	5.93	**4.96**	6.11
Triple Clf	7	**3.87**	7.33
Link Del	5.93	5.29	**5.11**
CoDEx Small	6.38	**4.7**	5.94
WN18RR	6.46	**4.78**	5.69
FB15k-237	5.75	**4.95**	6.5

absence of noise. *TransE* is particularly robust with respect to variance in the data (Table 12). Semantic Matching models perform worse, with the exception of *DistMult* which we already motivated. The *other* category lies in between[21].

4.4 Discussion

Hereafter, we summarize our results with respect to the initial Research Questions.

Summary for RQ1: *Do methods that can predict unknown relationships effectively (link prediction), work equally well in recognizing incorrect associations (link deletion)?*

Answer: Best performing methods are translational methods, which not only create embedded (dense) representations of relationships, but also incorporate a notion of direction of the relation (see Section 3.4). Since they are able to generate fine-grained representations of relationships, they obtain better scores both when predicting new links and detecting wrong ones (see e.g. Table 11). However, depending on the specific features of the dataset, these systems may degrade their performance (while often keeping their primacy) in the presence of a very high number of different relationships or in the case of transitive relationships. The motivations have been provided in Section 4.1. The answer to RQ1 is: *yes, best link prediction methods tend to hold their lead in the link deletion task as well.*

Summary for RQ2: *To what extent are the best performing Knowledge Graph Embedding (KGE) methods (according to the literature) robust in the presence of an increasing amount of noisy triples?*

Answer: As we said, translational models project relationships onto a lower-dimensional, dense, semantic space, generating more precise representations. For the very same reason, these systems are also more sensitive to the presence of noise in the training data. Contrarily, semantic matching systems adopt more scattered representations, which turn out to be more robust to noise. The answer to RQ1 is: *best performing methods are less robust to noise in the data.*

Summary for RQ3: *Are there systems that individually perform better with different tasks, datasets, and increasing levels of noise? If so, what are their "winning" features?*

21. Note that the primacy of "other" in the LD task occurs in a context in which all models perform more or less the same, therefore is not particularly significant.

Answer: No one single system holds the lead when considering all the possible dimensions explored in this paper: different tasks, different structural properties of the datasets, different representation models, and robustness to increasing levels of noise. However, two systems clearly emerge above the others. *TransE* (see Section 3.4.1) adds to the already summarized advantages of translational systems a simple yet apparently more effective way of combining the h and r vectors, forcing them to be "similar to" the t vector. This incorporates a notion of directionality into the representation, which especially helps with datasets, such as *WN18RR*, with transitive relations and a hierarchical backbone. The second notable model is *DisMult*, the only semantic matching system that occupies competitive positions in the various rankings presented in this paper. The sparse matrix representation adopted by *DisMult* seems to improve both robustness to noise and generalization capabilities. This results in a competitive advantage particularly for datasets, such as *FB15k-237*, with a very high number of relationships (see, among the others, Table 10). The answer to RQ3 is then: *the winning features for the best systems appear to be i) learning representations incorporating a notion of directionality, and ii) learning sparse matricial representations by relation type. On the other hand, since all systems train their models exploiting local connectivity properties, they are equally unable to work well on more structured KGs, such as WN18RR.* The latter represents a missing feature of state-of-the-art systems.

5. Conclusion

In this paper, we conducted a systematic benchmark study with the primary objective of assessing the true effectiveness of various Knowledge Graph Embedding (KGE) models in both tasks of graph enrichment and graph pruning. To this end, we defined three specific RQs (see Section 1) and performed experiments (see Section 3) where, after a systematic tuning phase, we evaluated alternative KGE models under the presence of an increasing level of noise on three standard datasets, i.e., *CoDEx Small*, *WN18RR* and *FB15k-237* (see Section 3.1). In our benchmark, we considered three different KG downstream tasks, i.e., *link prediction*, *triple classification* and *link deletion* (see Section 3.5).

Our benchmark study has a number of unique aspects as compared with other similar works:

1. First, rather than focusing only on link prediction and triple classification, we compare systems also with respect to the much less considered task of link deletion.

2. Second, we also analyze the robustness of systems when injecting in the training set a progressively higher number of wrong links. This is a realistic setting since many KGs are automatically or semi-automatically acquired.

3. Third, to avoid inconclusive results, with some systems working better or worse depending on specific experimental conditions, we designed experiments that allowed us to focus on one comparison dimension at a time. We considered four dimensions: structural characteristics of the KG, type of task, robustness to noise, and category of the predictive model.

4. Finally, we made an effort to interpret the results, in order to justify why some categories of models, or single models, perform better or worse under specific experimental conditions.

We found that - and explained why - translational models, and in particular *TransE*, perform systematically better than systems based on semantic matching, although slightly worse in the link deletion task. Furthermore, translational systems maintain their primacy with datasets with different structural properties. Among semantic matching models, we found that - and explained why - only one system, *DistMult*, achieves top performances when tested against semantically rich knowledge graphs with many different types of relationships, a context in which all the other systems perform poorly. *DistMult* also shows higher robustness to increasing levels of noise in training data. Finally, we ascertained that the effect of noise on highly structured knowledge graphs (i.e., those with a taxonomic backbone) is dreadful for all systems.

In addition to producing clear answers to our initial research questions, we identified two "winning" and one "missing" feature (see Section 4.4) that, to date, are not yet integrated into a single system, paving the way for possible improvement of KGE models.

Acknowledgments

We acknowledge the financial support from PNRR MUR project PE0000013-FAIR.

References

Abboud, R., Ceylan, u. u., Lukasiewicz, T., & Salvatori, T. (2020). Boxe: A box embedding model for knowledge base completion. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA. Curran Associates Inc.

Ahmad, S., & Scheinkman, L. (2019). How can we be so dense? the benefits of using highly sparse representations..

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Akrami, F., Saeef, M. S., Zhang, Q., Hu, W., & Li, C. (2020). Realistic re-evaluation of knowledge graph completion methods: An experimental study. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, SIGMOD '20, p. 1995–2010, New York, NY, USA. Association for Computing Machinery.

Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Galkin, M., Sharifzadeh, S., Fischer, A., Tresp, V., & Lehmann, J. (2021a). Bringing light into the dark: A large-scale evaluation of knowledge graph embedding models under a unified framework. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *PP*, 1–1.

Ali, M., Berrendorf, M., Hoyt, C. T., Vermue, L., Sharifzadeh, S., Tresp, V., & Lehmann, J. (2021b). PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. *Journal of Machine Learning Research*, *22*(82), 1–6.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., Harris, M. A., Hill, D. P., Issel-Tarver, L., Kasarskis, A., Lewis, S., Matese, J. C., Richardson, J. E., Ringwald, M., Rubin, G. M., & Sherlock, G. (2000). Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, *25*(1), 25–29.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, p. 2546–2554, Red Hook, NY, USA. Curran Associates Inc.

Bollacker, K., Evans, C., Paritosh, P., Sturge, T., & Taylor, J. (2008). Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, p. 1247–1250, New York, NY, USA. Association for Computing Machinery.

Bordea, G., Buitelaar, P., Faralli, S., & Navigli, R. (2015). SemEval-2015 task 17: Taxonomy extraction evaluation (TExEval). In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pp. 902–910, Denver, Colorado. Association for Computational Linguistics.

Bordea, G., Lefever, E., & Buitelaar, P. (2016). SemEval-2016 task 13: Taxonomy extraction evaluation (TExEval-2). In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pp. 1081–1091, San Diego, California. Association for Computational Linguistics.

Bordes, A., Usunier, N., Garcia-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, p. 2787–2795, Red Hook, NY, USA. Curran Associates Inc.

Bouchard, G., Singh, S., & Trouillon, T. (2015). On approximate reasoning capabilities of low-rank vector spaces. In *Knowledge Representation and Reasoning: Integrating Symbolic and Neural Approaches, AAAI Spring Symposium Series*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., & Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, p. 1306–1313. AAAI Press.

Chao, L., He, J., Wang, T., & Chu, W. (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)*, pp. 4360–4369.

Daza, D., Cochez, M., & Groth, P. (2021). Inductive entity representations from text via link prediction. In *Proceedings of the Web Conference 2021*, WWW '21, p. 798–808, New York, NY, USA. Association for Computing Machinery.

Dettmers, T., Minervini, P., Stenetorp, P., & Riedel, S. (2018). Convolutional 2d knowledge graph embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Ding, B., Wang, Q., Wang, B., & Guo, L. (2018). Improving knowledge graph embedding using simple constraints. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 110–121, Melbourne, Australia. Association for Computational Linguistics.

Faralli, S., Lenzi, A., & Velardi, P. (2022). A large interlinked knowledge graph of the Italian cultural heritage. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 6280–6289, Marseille, France. European Language Resources Association.

Feigenbaum, E. A. (1984). Knowledge engineering. *Annals of the New York Academy of Sciences*, *426*(1), 91–107.

Fellbaum, C. (Ed.). (1998). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutiérrez, C., Kirrane, S., Gayo, J. E. L., Navigli, R., Neumaier, S., Ngomo, A. N., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). *Knowledge Graphs*. Synthesis Lectures on Data, Semantics, and Knowledge. Morgan & Claypool Publishers.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., & Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. *CoRR*, *abs/2005.00687*.

Ji, S., Pan, S., Cambria, E., Marttinen, P., & Yu, P. S. (2022). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(2), 494–514.

Kadlec, R., Bajgar, O., & Kleindienst, J. (2017). Knowledge base completion: Baselines strike back. In Blunsom, P., Bordes, A., Cho, K., Cohen, S. B., Dyer, C., Grefenstette, E., Hermann, K. M., Rimell, L., Weston, J., & Yih, S. (Eds.), *Proceedings of the 2nd Workshop on Representation Learning for NLP, Rep4NLP@ACL 2017, Vancouver, Canada, August 3, 2017*, pp. 69–74. Association for Computational Linguistics.

Khadir, A. C., Aliane, H., & Guessoum, A. (2021). Ontology learning: Grand tour and challenges. *Computer Science Review*, *39*, 100339.

Lehmann, J., Isele, R., Jakob, M., Jentzsch, A., Kontokostas, D., Mendes, P. N., Hellmann, S., Morsey, M., van Kleef, P., Auer, S., & Bizer, C. (2015). DBpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web Journal*, *6*(2), 167–195.

Lin, Y., Liu, Z., Sun, M., Liu, Y., & Zhu, X. (2015). Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, p. 2181–2187. AAAI Press.

Mahdisoltani, F., Biega, J., & Suchanek, F. M. (2015). YAGO3: A knowledge base from multilingual wikipedias. In *Seventh Biennial Conference on Innovative Data Systems Research, CIDR 2015, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings*. www.cidrdb.org.

Mehta, S., Radke, M., & Sunkle, S. (2021). Named entity recognition using knowledge graph embeddings and distilbert. In *2021 5th International Conference on Natural Language Processing and Information Retrieval (NLPIR)*, NLPIR 2021, p. 146–150, New York, NY, USA. Association for Computing Machinery.

Melo, A., & Paulheim, H. (2017a). An approach to correction of erroneous links in knowledge graphs. In Tiddi, I. (Ed.), *K-CAPSAT-2017 : Proceedings of Workshops and Tutorials of the 9th International Conference on Knowledge Capture (K-CAP2017) Austin, Texas, December 4th, 2017*, Vol. 2065, pp. 54–57, Aachen, Germany. RWTH Aachen.

Melo, A., & Paulheim, H. (2017b). Detection of relation assertion errors in knowledge graphs. In *Proceedings of the Knowledge Capture Conference*, K-CAP 2017, New York, NY, USA. Association for Computing Machinery.

Nguyen, D. Q. (2017). An overview of embedding models of entities and relationships for knowledge base completion. *CoRR*, *abs/1703.08098*.

Nickel, M., Rosasco, L., & Poggio, T. (2016). Holographic embeddings of knowledge graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, p. 1955–1961. AAAI Press.

Nickel, M., Tresp, V., & Kriegel, H.-P. (2011). A three-way model for collective learning on multi-relational data. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, p. 809–816, Madison, WI, USA. Omnipress.

Park, Y., & Marcotte, E. (2011). Revisiting the negative example sampling problem for predicting protein-protein interactions. *Bioinformatics (Oxford, England)*, *27*, 3024–8.

Petzold, R., Gesese, G. A., Bogdanova, V., Zylowski, T., Sack, H., & Alam, M. (2021). Challenges of applying knowledge graph and their embeddings to a real-world use-case. In *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2021), co-located with the 20th International Semantic Web Conference (ISWC 2021): Virtual Conference, online, October 25, 2021. Ed.: M. Alam*, Vol. 3034 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Ruffinelli, D., Broscheit, S., & Gemulla, R. (2020). You CAN teach an old dog new tricks! on training knowledge graph embeddings. In *International Conference on Learning Representations*.

Safavi, T., & Koutra, D. (2020). CoDEx: A Comprehensive Knowledge Graph Completion Benchmark. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8328–8350, Online. Association for Computational Linguistics.

Saha, S., & Mandal, S. (2021). Application of tools to support linked open data. *Library Hi Tech News*, *38*(6), 21–24.

Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In Gangemi, A., Navigli, R., Vidal, M.-E., Hitzler, P., Troncy, R., Hollink, L., Tordai, A., & Alam, M. (Eds.), *The Semantic Web*, pp. 593–607, Cham. Springer International Publishing.

Socher, R., Chen, D., Manning, C. D., & Ng, A. Y. (2013). Reasoning with neural tensor networks for knowledge base completion. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, p. 926–934, Red Hook, NY, USA. Curran Associates Inc.

Suchanek, F. M., Kasneci, G., & Weikum, G. (2007). Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, p. 697–706, New York, NY, USA. Association for Computing Machinery.

Sun, Z., Deng, Z., Nie, J., & Tang, J. (2019). Rotate: Knowledge graph embedding by relational rotation in complex space. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Thain, D., Tannenbaum, T., & Livny, M. (2005). Distributed computing in practice: The condor experience: Research articles. *Concurr. Comput. : Pract. Exper.*, *17*(2-4), 323–356.

Toutanova, K., & Chen, D. (2015). Observed versus latent features for knowledge base and text inference. In *3rd Workshop on Continuous Vector Space Models and Their Compositionality* (3rd Workshop on Continuous Vector Space Models and Their Compositionality edition). ACL - Association for Computational Linguistics.

Trouillon, T., Welbl, J., Riedel, S., Gaussier, E., & Bouchard, G. (2016). Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, p. 2071–2080. JMLR.org.

Vashishth, S., Sanyal, S., Nitin, V., & Talukdar, P. (2020). Composition-based multi-relational graph convolutional networks. In *Proceedings of the 10th International Conference on Learning Representations, ICLR 2022, Virtual Event*.

Wang, Q., Mao, Z., Wang, B., & Guo, L. (2017). Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, *29*(12), 2724–2743.

Wang, Q., Wang, B., & Guo, L. (2015). Knowledge base completion using embeddings and rules. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, p. 1859–1865. AAAI Press.

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014a). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, p. 1112–1119. AAAI Press.

Wang, Z., Zhang, J., Feng, J., & Chen, Z. (2014b). Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, p. 1112–1119. AAAI Press.

Wen, J., Li, J., Mao, Y., Chen, S., & Zhang, R. (2016). On the representation and embedding of knowledge bases beyond binary relations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, p. 1300–1307. AAAI Press.

Weston, J., Bordes, A., Yakhnenko, O., & Usunier, N. (2013). Connecting language and knowledge bases with embedding models for relation extraction. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1366–1371, Seattle, Washington, USA. Association for Computational Linguistics.

Yang, B., Yih, W., He, X., Gao, J., & Deng, L. (2015). Embedding entities and relations for learning and inference in knowledge bases. In Bengio, Y., & LeCun, Y. (Eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Yao, L., Mao, C., & Luo, Y. (2019). KG-BERT: BERT for knowledge graph completion. In *arXiv preprint arXiv:1909.03193*.

Yu, J., Guo, M., Needham, C. J., Huang, Y., Cai, L., & Westhead, D. R. (2010). Simple sequence-based kernels do not predict protein–protein interactions. *Bioinformatics*, *26*(20), 2610–2614.

Zeng, K., Li, C., Hou, L., Li, J., & Feng, L. (2021). A comprehensive survey of entity alignment for knowledge graphs. *AI Open*, *2*, 1–13.

Zhang, Y., Yao, Q., Dai, W., & Chen, L. (2020). Autosf: Searching scoring functions for knowledge graph embedding. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pp. 433–444.