Journal of Artificial Intelligence Research 78 (2023) 1-15 Submitted 12/2022; published 09/2023

Supplementary Materials: 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

Appendix A. Results

In this Section we provide additional results and experiments not included in the main manuscript.

We begin with Tables 1, 2 and 3 showing the complete results of the experiments described in Section 3 of the main paper. Here, we show the performance of all systems, when varying the dataset, the task, the adopted performance metrics, and the amount of injected noise.

Table 1: Results obtained on CoDEx Small dataset.

	AutoSF	BoxE	ComplEx	ConvE	DistMult	HolE	PairRE	RotatE	TransE	TransH
Link Prediction										
hits@10	.420	.391	.433	.286	.387	.435	.430	**.561**	.506	.416
hits@10 (10%)	.387(-.090)	.374(-.046)	.281(-.416)	.225(-.167)	.397(+.027)	.328(-.293)	.351(-.315)	**.531**(-.082)	.438(-.186)	.380(-.098)
hits@10 (20%)	.357(-.086)	.372(-.026)	.240(-.264)	.177(-.149)	.385(-.003)	.314(-.165)	.305(-.171)	**.527**(-.046)	.393(-.154)	.356(-.082)
hits@10 (30%)	.317(-.094)	.377(-.013)	.222(-.192)	.164(-.111)	.389(+.002)	.297(-.126)	.289(-.129)	**.503**(-.053)	.378(-.117)	.319(-.088)
hits@10 (rand)	.039(-.105)	.059(-.092)	.016(-.115)	.013(-.076)	.045(-.095)	.025(-.113)	.033(-.110)	.062(-.138)	.041(-.123)	.038(-.105)
mrr	.206	.205	.204	.123	.196	.206	.215	**.327**	.269	.221
mrr (10%)	.18	.202	.113	.098	.207	.134	.177	**.309**	.223	.197
mrr (20%)	.157	.202	.114	.081	.2	.144	.145	**.297**	.19	.171
mrr (30%)	.142	.206	.106	.069	.199	.134	.146	**.293**	.18	.155
mrr (rand)	.025	.029	.01	.009	.024	.013	.022	.041	.02	.02
Triple Class.										
f1_macro	.928	.928	.890	.695	.904	.879	.906	.908	**.937**	.932
f1_macro (10%)	.878(-.137)	**.915**(-.035)	.839(-.139)	.761(+.180)	.913(+.025)	.833(-.126))	.831(-.205)	.854(-.148)	.883(-.148)	.881(-.139)
f1_macro (20%)	.844(-.115)	**.908**(-.027)	.788(-.139)	.711(+.022)	.894(-.014)	.792(-.119)	.796(-.150)	.828(-.109)	.859(-.107)	.851(-.111)
f1_macro (30%)	.809(-.108)	.837(-.083)	.770(-.109)	.724(+.026)	**.890**(-.013)	.771(-.098)	.780(-.115)	.816(-.084)	.847(-.082)	.811(-.110)
f1_macro (rand)	.525(-.111)	.517(-.114)	.509(-.105)	.508(-.052)	.535(-.102)	.510(-.102)	.517(-.108)	.497(-.114)	.520(-.115)	.504(-.118)
norm_dist	.224	**.525**	.204	.052	.331	.119	.227	.251	.270	.279
norm_dist (10%)	.195	**.492**	.167	.068	.286	.105	.168	.233	.247	.218
norm_dist (20%)	.154	**.466**	.128	.066	.283	.096	.156	.225	.256	.216
norm_dist (30%)	.145	**.41**	.127	.064	.261	.096	.125	.160	.232	.193
norm_dist (rand)	.010	.045	.000	.006	.029	.002	.001	.000	.008	.000
Link Deletion										
hits@10	.728	.295	.536	.648	.522	.350	.479	.550	.650	**.754**
hits@10 (10%)	.524(-.558)	.224(-.194)	.415(-.331)	.414(-.640)	**.607**(+.232)	.324(-.071)	.355(-.339)	.386(-.449)	.550(-.273)	.605(-.408)
hits@10 (20%)	.441(-.393)	.249(-.063)	.261(-.376)	.286(-.495)	.570(+.065)	.301(-.067)	.277(-.276)	.260(-.397)	.460(-.260)	**.609**(-.198)
hits@10 (30%)	.298(-.392)	.255(-.036)	.225(-.283)	.291(-.325)	**.591**(+.063)	.271(-.072)	.274(-.187)	.334(-.197)	.534(-.106)	.538(-.197)
hits@10 (rand)	.023(-.195)	.102(-.053)	.087(-.124)	.133(-.142)	.078(-.123)	.019(-.092)	.046(-.120)	.006(-.150)	.041(-.168)	.097(-.182)
mrr	.283	.218	.320	.365	.240	.123	.288	.189	.433	**.551**
mrr (10%)	.335	.13	.26	.229	.303	.209	.147	.199	.198	**.496**
mrr (20%)	.234	.148	.206	.173	.266	.13	.126	.127	.244	**.311**
mrr (30%)	.141	.182	.134	.203	**.452**	.201	.159	.113	.22	.276
mrr (rand)	.013	.038	.054	.046	.022	.009	.028	.005	.020	.037

Table 2: Results obtained on WN18RR dataset.

	AutoSF	BoxE	ComplEx	ConvE	DistMult	HolE	PairRE	RotatE	TransE	TransH
Link Prediction										
hits@10	.403	.504	.369	.439	.270	.428	.434	**.547**	.477	.368
hits@10 (10%)	.400(-.003)	.474(-.032)	.355(-.015)	.415(-.026)	.253(-.018)	.417(-.012)	.412(-.024)	**.519**(-.030)	.447(-.032)	.343(-.027)
hits@10 (20%)	.393(-.005)	.463(-.022)	.344(-.013)	.387(-.028)	.190(-.043)	.129(-.161)	.401(-.018)	**.496**(-.027)	.405(-.039)	.322(-.025)
hits@10 (30%)	.389(-.005)	.455(-.018)	.330(-.014)	.378(-.022)	.007(-.095)	.400(-.010)	.401(-.012)	**.478**(-.025)	.361(-.042)	.304(-.023)
hits@10 (rand)	.001(-.043)	.002(-.054)	.001(-.040)	.001(-.047)	.000(-.029)	.000(-.046)	.001(-.047)	.000(-.059)	.000(-.051)	.007(-.039)
mrr	.380	.453	.351	.283	.182	.385	.403	**.479**	.185	.149
mrr (10%)	.371	.425	.312	.260	.176	.377	.381	**.454**	.171	.136
mrr (20%)	.346	.408	.295	.231	.128	.097	.365	**.430**	.146	.124
mrr (30%)	.326	.393	.265	.209	.004	.338	.357	**.411**	.128	.115
mrr (rand)	.001	.001	.001	.001	.000	.000	.001	.000	.000	.004
Triple Class.										
f1_macro	.716	.795	.689	.801	.663	.786	.780	.801	**.889**	.754
f1_macro (10%)	.721(-.005)	.756(-.042)	.694(+.005)	.802(+.001)	.625(-.041)	.774(-.013)	.761(-.020)	.786(-.016)	**.863**(-.028)	.697(-.062)
f1_macro (20%)	.723(-.004)	.734(-.033)	.694(+.003)	.770(-.017)	.614(-.026)	.689(-.052)	.752(-.015)	.770(-.017)	**.848**(-.022)	.641(-.061)
f1_macro (30%)	.710(-.002)	.720(-.027)	.699(+.004)	.761(-.014)	.364(-.108)	.761(-.009)	.730(-.018)	.760(-.015)	**.830**(-.021)	.576(-.064)
f1_macro (rand)	.501(-.023)	.508(-.031)	.497(-.021)	.508(-.032)	.372(-.031)	.508(-.030)	.506(-.030)	.505(-.032)	.497(-.042)	.498(-.028)
norm_dist	.180	.270	.155	.185	.117	.219	.225	.254	**.300**	.156
norm_dist (10%)	.176	.231	.136	.180	.118	.207	.208	.236	**.254**	.139
norm_dist (20%)	.160	.195	.143	.102	.096	.093	.193	.215	**.230**	.131
norm_dist (30%)	.165	.171	.125	.129	.004	.184	.177	.204	**.214**	.115
norm_dist (rand)	.000	.005	.000	.002	.000	.002	.002	.003	.000	.000
Link Deletion										
hits@10	.009	.015	.006	.018	.014	.009	.021	.016	**.062**	.043
hits@10 (10%)	.005(-.004)	.017(+.002)	.007(+.001)	.005(-.014)	.058(+.047)	.028(+.020)	.017(-.004)	.022(+.006)	**.094**(+.034)	.036(-.007)
hits@10 (20%)	.007(-.001)	.019(+.002)	.005(-.000)	.011(-.004)	.041(+.014)	.005(-.002)	.016(-.003)	.013(-.002)	**.056**(-.003)	.038(-.003)
hits@10 (30%)	.005(-.001)	.019(+.001)	.005(-.000)	.011(-.002)	.004(-.004)	.027(+.006)	.017(-.001)	.017(+.000)	**.042**(-.007)	.022(-.007)
hits@10 (rand)	.003(-.001)	.015(.000)	.002(-.000)	.007(-.001)	.004(-.001)	.005(-.000)	.009(-.001)	.009(-.001)	.008(-.006)	.002(-.004)
mrr	.007	.011	.005	.014	.011	.008	.014	.010	**.028**	.018
mrr (10%)	.005	.011	.005	.007	.022	.014	.011	.012	**.050**	.017
mrr (20%)	.005	.013	.005	.007	.018	.006	.012	.010	**.046**	.021
mrr (30%)	.004	.012	.005	.007	.004	.017	.013	.010	**.025**	.014
mrr (rand)	.002	.011	.002	.006	.003	.003	.005	.006	.005	.003

A.1 Model vs. Tasks vs. Dataset Analysis

In the next Sections, we discuss in detail the performance of each benchmarked model.

A.1.1 TransE (translational)

link prediction: although the model works well in general, the best performances are obtained with datasets having a low number of relation types. The model is resulting less sensitive to the noise with dense graphs.

link deletion: the best performances are obtained with very dense graphs (e.g., CoDEx Small) and is the outperforming model and more robust to noise with medium and low-density datasets, thus is more sensitive to noise with dense graphs.

triple classification: the model outperforms the others with WNR18RR, is the best one with the original CoDEx Small, and with FB15k-237 with a 10% of noisy triples. Lower sensitivity to noise has been observed with medium-dense graphs (i.e., FB15K-237), while with high-dense and low-dense graphs the model, shows average robustness to noise.

A.1.2 TransH (translational)

link prediction: although the model has average/low performances in general, the best performances are obtained with the FB15k-237 dataset. The model is resulting more sensitive to the noise with dense graphs.

Table 3: Results obtained on FB15k-237 dataset.

	AutoSF	BoxE	ComplEx	DistMult	HolE	PairRE	RotatE	TransE	TransH
Link Prediction									
hits@10	.412	.444	.300	.321	.322	.429	**.487**	.344	.450
hits@10 (10%)	.381(-.010)	.425(-.006)	.194(-.034)	.333(+.004)	.234(-.029)	.387(-.014)	**.449**(-.012)	.325(-.006)	.366(-.028)
hits@10 (20%)	.356(-.009)	**.411**(-.005)	.159(-.023)	.330(+.001)	.187(-.022)	.355(-.012)	.409(-.013)	.312(-.005)	.312(-.023)
hits@10 (30%)	.335(-.008)	**.398**(-.005)	.142(-.017)	.316(-.000)	.169(-.017)	.324(-.011)	.380(-.012)	.306(-.004)	.274(-.019)
hits@10 (rand)	.004(-.013)	.035(-.013)	.002(-.010)	.002(-.010)	.002(-.010)	.007(-.014)	.007(-.016)	.012(-.011)	.017(-.014)
mrr	.246	.275	.156	.185	.176	.251	**.290**	.201	.268
mrr (10%)	.224	**.261**	.095	.195	.123	.212	.254	.184	.169
mrr (20%)	.203	**.252**	.080	.194	.097	.191	.228	.174	.121
mrr (30%)	.187	**.244**	.072	.185	.086	.170	.214	.167	.100
mrr (rand)	.003	.017	.001	.001	.001	.004	.005	.009	.011
Triple Class.									
f1_macro	.972	.974	.932	.975	.943	.965	.943	.976	**.981**
f1_macro (10%)	.953(-.006)	.956(-.006)	.879(-.017)	**.972**(-.001)	.899(-.014)	.899(-.022)	.881(-.020)	**.972**(-.001)	.931(-.016)
f1_macro (20%)	.929(-.007)	.917(-.009)	.854(-.013)	**.968**(-.001)	.879(-.010)	.872(-.015)	.861(-.013)	.962(-.002)	.901(-.013)
f1_macro (30%)	.901(-.008)	.885(-.010)	.839(-.010)	**.960**(-.002)	.862(-.009)	.851(-.012)	.850(-.010)	.952(-.003)	.882(-.010)
f1_macro (rand)	.495(-.016)	.519(-.015)	.500(-.014)	.492(-.016)	.496(-.015)	.509(-.015)	.506(-.014)	.502(-.016)	.519(-.015)
norm_dist	.202	**.512**	.240	.381	.310	.313	.216	.460	.362
norm_dist (10%)	.114	**.441**	.161	.296	.257	.223	.151	.429	.282
norm_dist (20%)	.115	.401	.165	.319	.231	.182	.132	**.408**	.251
norm_dist (30%)	.120	.373	.138	.309	.225	.159	.130	**.392**	.256
norm_dist (rand)	.000	.014	.000	.000	.000	.004	.000	.005	.007
Link Deletion									
hits@10	.302	.222	.112	**.446**	.095	.269	.160	.270	.413
hits@10 (10%)	.148(-.051)	.138(-.028)	.034(-.026)	**.452**(-.002)	.080(-.005)	.117(-.050)	.028(-.043)	.373(+.034)	.335(-.026)
hits@10 (20%)	.079(-.037)	.077(-.024)	.044(-.011)	.336(-.018)	.082(-.002)	.071(-.033)	.040(-.020)	**.360**(+.015)	.275(-.023)
hits@10 (30%)	.075(-.025)	.073(-.016)	.043(-.007)	.249(-.022)	.036(-.006)	.073(-.021)	.021(-.015)	**.263**(-.001)	.168(-.027)
hits@10 (rand)	.000(-.010)	.001(-.007)	.000(-.004)	.000(-.015)	.001(-.003)	.007(-.009)	.001(-.005)	.002(-.009)	.003(-.013)
mrr	.147	.084	.040	**.155**	.045	.145	.052	.072	.134
mrr (10%)	.052	.057	.017	**.201**	.036	.044	.016	.107	.071
mrr (20%)	.041	.046	.019	**.179**	.037	.039	.017	.090	.064
mrr (30%)	.033	.042	.020	**.146**	.020	.034	.013	.068	.053
mrr (rand)	.001	.002	.000	.000	.001	.005	.001	.002	.003

link deletion: the best performances are obtained with the densest datasets (i.e., CoDEx Small and FB15k-237). The model is resulting has medium sensitivity to the noise, dataset-wise.

triple classification: overall, the model is not robust to noise. The best performances are obtained with dense graphs but with a low amount of noisy triples.

A.1.3 DistMult (semantic matching)

link prediction: although the model has low performances in general, it is resulting very sensitive to the noise with the low-dense WN18RR dataset, and it is one of the most robust otherwise.

link deletion: the best performances are obtained with dense and medium-dense datasets while is performing badly with WN18RR. The higher the dataset density is more robust to noisy triples model behavior.

triple classification: overall, the model is robust to noise and performs well with dense graphs. With WN18RR the model obtained the worse performances.

A.1.4 ComplEx (semantic matching)

link prediction: although the model has low/medium performances in general, the best performances are obtained with the very dense CoDEx Small dataset. The model is resulting very sensitive to noise with dense datasets.

link deletion: the best performances are obtained with the densest dataset (i.e., CoDEx Small), while, it performs badly with low dense graph (i.e., WN18RR). The model is resulting more sensitive to noise with highly dense graphs.

triple classification: overall, the model is not robust to noise with dense graphs and performs with low performances in all the settings.

A.1.5 HolE (semantic matching)

link prediction: the model has medium/low performances in general. It also results in being sensitive to noise with dense datasets.

link deletion: HolE performs with medium/low performances of *triple classification*. The model is resulting more sensitive to noise with highly dense graphs.

triple classification: overall, the model has medium performance. It is very sensitive to noisy triples with the densest graph.

A.1.6 ConvE (Convolutional Neural Network)

link prediction: the model has worse performances with dense datasets - we remark that the model was not capable of converging with the FB15k-237 dataset - and medium performances with WN18RR. The model is also resulting sensitive to noise datasets-wise.

link deletion: the model has medium-low performances. Additionally, it is resulting very sensitive to the noise with the CoDEx Small dataset.

triple classification: overall, the model performs better with low-dense datasets. While it results in the worse-performing model with dense datasets. With CoDEx Small, the addition of noisy triples significantly improves the performances.

A.1.7 RotatE (translational)

link prediction: although the model obtained the best performances in *link prediction*, it works better with the datasets having medium/low amounts of relation types. The model is also resulting sensitive to the noise with all datasets.

link deletion: the model obtained medium to low performances of *link deletion*. The model is resulting more sensitive to noise with highly dense graphs.

triple classification: overall, the model has medium to low performances with dense graphs. It is more suitable when applied to low-dense graphs with a low number of relation types, such as WN18RR. It is medium sensitive to noise datasets-wise.

A.1.8 AutoSF (no category. It's based on AutoML)

link prediction: the model has medium performances in general and is more sensitive to noise with dense datasets.

link deletion: the model is capable of obtaining good performances with dense datasets. The best performances are obtained with the densest dataset (i.e., CoDEx Small). The model is also resulting in the most sensitive to noise with dense graphs.

triple classification: good performances have been obtained with CoDEx Small dataset (the densest one), medium to low performances are obtained otherwise. Overall, the model is not robust to noise. More sensitivity has been observed with the CoDEx Small and the WN18RR datasets (the datasets with a medium and a low number of relation types, respectively).

A.1.9 BoxE (translational)

link prediction: the model has good performances with dense and low-dense datasets, where it is also more sensitive to noise.

link deletion: good performances are obtained with the FB15k-237, while medium to low performance is obtained otherwise. The model is also resulting sensitive to noise with dense graphs.

triple classification: it is more suitable when applied to dense graphs, i.e., CoDEx Small. The model is sensitive to noise, datasets-wise.

A.1.10 PairRE (translational)

link prediction: the model has medium performances with the three datasets and is also results sensitive to the noise, datasets-wise.

link deletion: the model has medium/low performances datasets-wise. The model results from sensitive to very sensitive to noise depending on the datasets' densities.

triple classification: the model obtained from medium to low performances. It is more suitable when applied to medium-dense and high-dense graphs. Overall, the model is one of the less robust to noise.

A.2 Datasets vs. Tasks vs. Models Analysis

In the next Sections, for each dataset, we provide an analysis of the models' behaviors across the three considered tasks.

A.2.1 CoDEx Small

Tasks-wise, with the original CoDEx Small (the datasets having the highest density and a medium amount of relations types) the majority of the translational models outperform the others. In particular, RotatE is the best-performing model in *link prediction* w/ and w/o noisy triples. In *triple classification* and in the absence of noisy triples, TransE is the best-performing model, while BoxE and DistMult are the most affordable in the presence of noise. In *link deletion*, instead, TransH and DistMult are the outperforming models in the absence and in the presence of noise, respectively. Thus, translational models work better in the absence of noise triples and DistMult is the most robust to the noise in two of the tasks.

A.2.2 WN18RR

With the WN18RR (the dataset with: a low density, a predominant taxonomic structure, a high standard deviation on the average node degree, and a low number of relations types), TransE is both the outperforming and more robust model for the *link prediction* task. While RotateE is the best in terms of performance and robustness in both the *triple classification* and *link deletion* tasks.

A.2.3 FB15k-237

With FB15k-237 (in our benchmark, a dense graph with the highest number of relation types), we observed a comparable models' performance, where translational models (i.e., BoxE and RotatE) outperform the others in the task of *link prediction* by means of *hits@10* and sensitivity to noisy triples. In the *triple classification* task, with low amounts of noise TransH and TransE are the best-performing models, while, with a higher amount of noisy triples, DistMult is the most affordable model. In the *link deletion* task, DistMult outperforms the others in the presence of low amounts of noise, while TransE is the less sensitive model.

Appendix B. License and Reproducibility

B.1 Links to Access the Dataset and its Metadata

Both source code and data of our benchmark are publicly available at the following GitHub repository: `https://github.com/stefanofaralli/noisy-kgs-benchmark`.

B.2 Data Format

Datasets are released as standard tabbed separated values (.tsv) files. To simplify the access, and due to the size, data are grouped into individual compressed files (.zip).

B.3 Long-term Preservation

The source code is shared, preserved and maintained through the GitHub repository. Due to the size, the data (i.e., the datasets, the pre-trained models, and the results) are linked from the GitHub repository and stored in a Google drive folder.

B.4 License:

Our code and data are released with a MIT License `https://github.com/stefanofaralli/noisy-kgs-benchmark/blob/main/LICENSE`.

B.5 Metadata and Persistent Dereferenceable Identifier:

To provide both metadata and a persistent dereferenceable identifier we created the following page: `https://figshare.com/articles/dataset/noisy-kgs-benchmark/22778945/1`. As a result, the benchmark is accessible through a DOI (i.e., *"10.6084/m9.figshare.22778945.v1"*) and through the major Web search engines supporting microdata indexing.

B.6 Reproducibility of Results:

All results can be easily reproduced, i.e., all necessary datasets, code, and evaluation procedures are accessible and documented in our code repository: `https://github.com/stefanofaralli/noisy-kgs-benchmark`.

Acknowledgments

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