Estimating Data Split Error for Training Knowledge Graph Embeddings

Aleksander Suchorab[1][0009−0001−5006−6492] ✉, Igor Wojnicki[1][0000−0002−4406−4992], and Andrzej Bielecki[1][0000−0002−0192−3785]

AGH University of Krakow, al.Mickiewicza 30, 30-059 Kraków, Poland
asuchorab@agh.edu.pl, wojnicki@agh.edu.pl, bielecki@agh.edu.pl

Abstract. When graph data is divided into a train-test split, which is common in machine learning, it alters the graph structure. In particular, some splits may result in a training set that is unable to distinguish certain groups of vertices as a consequence of graph topology. It can lead to an increased number of errors in evaluation when the test set requires a distinction between those groups but the training set provides none. We propose methods to identify, explain, and estimate the split error of a graph dataset with a particular assignment of triples into splits. The method uses graph entropy differences, based on automorphism of labeled graphs. We examine the approach on a variety of knowledge graph embedding models and datasets. Various datasets were found to have differing potential for split errors, depending on their structure.

Keywords: Knowledge Graph Embedding · Information theory · Structural information · Network analysis · Training-test split.

1 Introduction and Motivation

Knowledge graphs use vertices to represent conceptual entities, while the edges define the relationships between them. However, graph structures are unfit for processing using many machine learning algorithms which operate on tensors. In order to resolve that issue, Knowledge Graph Embedding (KGE) techniques can be used, which transform graph relationships into fixed-length, lower-dimensional vectors, aiming to preserve structural information and properties.

There are many surveys [4] and benchmarks [8] of various KGE algorithms, although a fair comparison is difficult [8]. KGE methods are often evaluated on dense, curated datasets that do not represent real-world graphs, and sparsity has been shown to degrade performance [11].

For most machine learning tasks, including KGE, a dataset is divided into at least two subsets: training and testing. The naive approach to dividing a dataset involves random sampling. Such sampling might lead to bias, which has been investigated for non-graph machine learning tasks [12, 5].

Due to the nature of graph data, some vertices may not be distinguishable in the graph topology. Redundant vertices have been shown to degrade machine

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46

learning performance [10]. Graph entropy [6] can be used, among other things, to quantitatively describe topologically indistinguishable vertices and has been extensively investigated with many applications to machine learning [7, 13, 3].

We posit that these vertices introduce topological bias in train-test splits. We propose an approach that uses graph entropy to describe the information difference between the data subsets that leads to increased evaluation error, which is the main contribution of this paper. Our framework includes a method of detecting and explaining this class of error, which we call split error, and a method of estimating its effect on evaluation error in a particular train-test split, which is also able to determine which test triples contribute to this error.

The proposed approach is tested on several KGE models and datasets by evaluating the models on the provided data splits, and then on test sets without triples that contribute to split error. Then, we compare the performance of the models with error estimates described by our framework.

2 Formulation of Split Error in Graph Data

A graph dataset G is understood as a set of vertices or entities V and a set of labelled edges or triples E. The isomorphism of edge-labeled graphs, similarly as in [9], is one that preserves not only the graph structure but also edge labels. In particular, we will consider automorphism of this kind. A pair of nodes $x, y \in V$ is said to be indistinguishable if they belong to one automorphism orbit.

One way to measure degree to which the graph contains indistinguishable vertices, used in [10], is to use the proportion of vertices with singleton orbits, to all vertices, which will be denoted as p_s. This method does not take into account the distribution of nodes into orbits of varying sizes greater than 1. To better describe this, graph entropy[6] can be used:

$$H = -\sum_{i=1}^{k} \frac{n_i}{n} log_2 \frac{n_i}{n}, \tag{1}$$

where k is the number of automorphism orbits and n_i is the size of the i-th orbit. The maximum possible entropy for a graph with n nodes is $log_2 n$ and this value could be used to normalize the entropy to the range $[0, 1]$, written as H_{norm}.

We examined various graph datasets available in the PyKEEN library[2]. The datasets exhibit varying amounts of indistinguishable vertex pairs. H_{norm} ranges from 0.8729 for DBpedia50, through 0.9670 for WN18, to values near 1 like 0.9994 for PharmKG8k, while p_s ranges from low 0.5224 for DBpedia50, through 0.7674 for WN18 to 0.9928 for PharmKG8k. Another commonly used benchmarking set, FB15k, is also characterized by values near 1. Similar findings were made by [10] on a different selection of datasets.

We examined the structural patterns that characterize indistinguishable vertices. The findings are summarized in Tab. 1. They are concentrated in sparse parts of the graphs, as illustrated by mean and median amount of incident edges. Another insight is that the orbits typically consist of vertices within 2 hops. This

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46

Table 1. Properties of vertices that are in non-singleton orbits in the training parts of datasets, p_s is the proportion of singleton orbit vertices, and n_{ns} is the number of vertices in non-singleton orbits. The "incident" columns describe the mean and median amount of edges incident to these vertices vertex with "expect" describing expected edges per vertex as estimated by $\frac{2|E|}{|V|}$ for the whole graph. The "compactness" columns describe the proportion of classes of which all elements are within d distance or hops.

Dataset	p_s	n_{ns}	incident			compactness	
			expect	mean	median	d=1	d=2
PharmKG8k	0.9928	14	106.7	2.43	3	0.0	1.0
FB15k237	0.9739	347	37.52	4.12	1	0.0	0.977
FB15k	0.9651	101	64.63	3.35	2	0.036	1.0
OpenEA	0.8306	3056	4.08	1.44	1	0.003	0.935
CoDExMedium	0.9987	6	21.8	9.67	10	0.0	1.0
DBpedia50	0.5224	12470	2.62	1.30	1	0.010	0.773
WD50KT	0.8348	7897	8.21	1.38	1	0.0	0.993
WN18RR	0.7723	8823	4.28	1.67	2	0.003	0.976
WN18	0.7674	6906	6.91	3.15	2	0.003	0.987
AristoV4	0.7262	11615	11.55	1.19	1	0.0	0.993
CoDExLarge	0.9713	1625	14.14	5.16	5	0.017	1.0
DB100K	0.8823	8954	12.00	2.82	3	0.002	0.980

corresponds to a pattern where they are connected to one or a few hub vertices. This and similar patterns that involve common connections should result in embedding vectors in one orbit to converge, as they experience identical influence from neighbors. Therefore, they should be rated similarly in head or tail completion. There are factors that may cause them to not converge, such as insufficient training or regularization. Any significant deviations of embeddings in one orbit are likely to be noise, rather than informed in any way.

When KGE models are trained on data with a training-test split, the model learns embeddings on just the triples present in the training set. This set, when interpreted as a graph, has a different topology than the initial set. In particular, distinguishability of a pair of nodes can change between those sets, and this discrepancy is the cause of the split error.

We consider transductive[1] link prediction, often used to evaluate KGE models, although these considerations apply to any task in which vertices are chosen or rated. The transductive setup also requires that the test set only contains entities and triples present in the training set, which is ensured by default in PyKEEN, and it is assumed in further considerations.

Let (V_{train}, E_{train}) be the training set and (V_{test}, E_{test}) be the test set, $V_{test} \subseteq V_{train}$, $E_{train} \cap E_{test} = \emptyset$. Also, $E_{full} = E_{train} \cup E_{test}$, which is the totality of data available at the point of evaluation. We compute the automorphism orbits of E_{train} and E_{full}, and interpret them as partitions of V_{train} and filter them, so they only contain vertices from V_{test}, and denote them P_{train} and P_{full}. Finally, let us compute the intersection of these partitions $P_{inter} = \{x : x = a \cap b, x \neq \emptyset, a \in P_{train}, b \in P_{full}\}$.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46

Finally, the entropy of partitions P_{train}, P_{inter} and P_{full}, denoted as H_{train}, H_{inter} and H_{full}, can be computed using Eq. 1, where $n = |V_{test}|$ and n_i are filtered to exclude vertices outside of V_{test}. Then, $\Delta H = H_{full} - H_{train}$ describes the gain or loss of vertex distinguishability in evaluation compared to training. However, it is insensitive to gain in one part of the graph and loss in another. Since P_{inter} always contains the more granular divisions out of both P_{train} and P_{full}, $H_{inter} \geq \max(H_{train}, H_{full})$. We define $\Delta_H^+ = H_{inter} - H_{train}$ and $\Delta_H^- = H_{inter} - H_{full}$, which can be used to measure only the gain and loss of distinguishability information respectively.

We posit that any gain in this information, that is, an increase in the ability to distinguish vertices compared to the training data, is detrimental in evaluation, so $\Delta_H^+ > 0$ is an indication of the split error. This can be illustrated using Fig. 1, where case c) displays such scenario. The influence of Δ_H^- is uncertain.

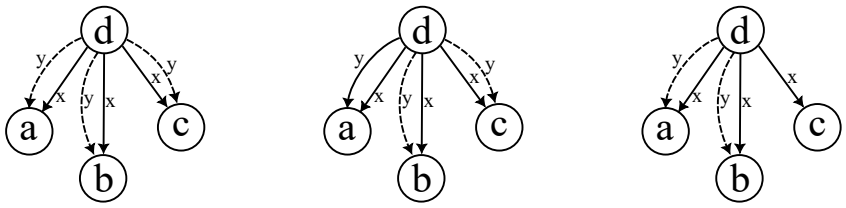
The difference in distinguishability information measured by Δ_H^+ indicates the presence of the split error but does not make a prediction of the error. We propose a procedure for estimating the split error from the perspective of each triple in the test set, suited for link prediction. A vertex that belongs to a smaller orbit in E_{inter} than in E_{train}, as discussed above, will be called ambiguous.

The procedure assigns a score from 0 to 1 for each vertex match, head and tail separately, where 1 is the score in absence of split error, and lower values otherwise,. This assumes that the model is able to correctly predict any triple without ambiguous vertices, in other words there are no other sources of error.

For each triple, head and tail predictions are performed, and their expected scores depend on the ambiguity of the head and tail vertices, so the score for vertex x is defined as

$$s_v(x) = \frac{|\mathrm{orb}_{inter}(x)|}{|\mathrm{orb}_{train}(x)|}, \tag{2}$$

where orb_{inter} and orb_{train} are the orbits of a vertex in the respective partitions. The estimated split error factor using scores for the head and tail vertices for



a) $\Delta H = \Delta_H^+ = \Delta_H^- = 0$,
$P_{train} = \{\{a, b, c\}, \{d\}\}$,
$P_{full} = \{\{a, b, c\}, \{d\}\}$,

b) $\Delta_H^+ = 0$, $\Delta_H^- > 0$,
$P_{train} = \{\{a\}, \{b, c\}, \{d\}\}$,
$P_{full} = \{\{a, b, c\}, \{d\}\}$,

c) $\Delta_H^+ > 0$, $\Delta_H^- = 0$,
$P_{train} = \{\{a, b, c\}, \{d\}\}$,
$P_{full} = \{\{a, b\}, \{c\}, \{d\}\}$,

Fig. 1. Example cases of edge patterns present in training and test sets. The solid and dashed lines represent the training and test sets respectively, and "x" and "y" are distinct relations. The examples should be understood as parts of a larger graph, where a model should be able to infer some relationships. In case c), the model has no information that could differentiate a or b from c, leading to split error.

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46

each edge can be calculated as

$$s = \frac{1}{|E_{test}|} \sum_{(h,r,t) \in E_{test}} \left(1 - \frac{s_v(h) + s_v(t)}{2}\right). \tag{3}$$

This estimation is not derived from any scoring metric, such as Mean Reciprocal Rank (MRR) or Hit Rate at k (hits@k), but instead summarizes the effect of ambiguous vertices weighted by the occurrence of triples that contain them.

This estimate was further adjusted experimentally to improve its ability to predict evaluation differences. One considered technique was using squares of s_v:

$$s_v(x)' = \frac{|\mathrm{orb}_{inter}(x)|^2}{|\mathrm{orb}_{train}(x)|^2}. \tag{4}$$

The split error factor can also be adjusted with the computed metric m such as MRR or hits@10, with scores in range [0, 1]. This is not fully explained, but can be motivated by the fact that for lower scores, noisy predictions from ambiguous vertices will not lower the score by much.

$$s' = ms \tag{5}$$

3 Comparison of Split Error Estimation and Embedding Evaluation

We conducted KGE training using the PyKEEN[2] library. We considered KGE models RESCAL, TransE, DistMult, ComplEx, MuRE, RotatE, CrossE, TuckER, TorusE, SimplE, BoxE and AutoSF. We used datasets AristoV4, CoDExLarge, DB100K, DBpedia50, FB15k237, OpenEA, PharmKG8k, WD50KT, and WN18RR, as provided by PyKEEN. Some datasets from Tab. 1, that is CoDExMedium and FB15k, are not examined further, as they do not contain ambiguous vertices. When loading the datasets with PyKEEN, filtering is performed by default, removing entities and relations not appearing in the training set. All computations were performed on those filtered datasets.

Each combination of model and dataset has been run with a grid hyperparameter search, as influenced by some publications on KGE models[14]. For all, stochastic Local Closed World Assumption (sLCWA) training loop available in PyKEEN was used, with batch size 512, Adagrad optimizer with learning rate in $\{0.002, 0.1, 0.5\}$, "basic" negative sampler with ratio of 10, L2 regularizer with weight in $\{0, 0.0001, 0.01\}$. The embedding dimensions were chosen depending on size of the dataset, with $\{50, 100\}$ if $|E| < 10000$ else $\{100, 150\}$ if $|E| < 50000$ else $\{150, 200\}$. If a model supports different relation and embedding dimensions, the relation dimension was set to the same value. An early stopper was used, evaluating once every 10 epochs with patience 4 and relative delta 0.002, with max epochs set to 1000.

We evaluated the models on link prediction using RankBasedEvaluator in PyKEEN, on the test set with filtered evaluation. We calculated MRR and hits@10

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46

Table 2. Results of KGE evaluation averaged across all models, before and after the processing to reduce split error.

Dataset	hits@10		MRR	
	baseline	change	baseline	change
PharmKG8k	0.28321	$8.9 \cdot 10^{-6}$	0.14657	$5.2 \cdot 10^{-6}$
FB15k237	0.36719	0.00028	0.21305	0.00016
OpenEA	0.31560	0.00939	0.21844	0.00686
DBpedia50	0.32182	0.02907	0.24445	0.02193
WD50KT	0.24431	0.00410	0.13723	0.00245
WN18RR	0.42778	0.02161	0.35027	0.01910
WN18	0.91208	0.00279	0.76966	0.00237
AristoV4	0.24131	0.00092	0.13696	0.00038
CoDExLarge	0.23860	0.00034	0.13870	0.00017
DB100K	0.31281	0.00106	0.18134	0.00043

Table 3. Sizes of the test sets, amount of removed test triples during processing, entropy differences and error estimates. The error estimates use Eq. 3, although e uses Eq. 2 for vertex scores while e_{sqr} uses Eq. 4.

Dataset	$\lvert E_{test}\rvert$	removed	ΔH	Δ_H^+	e	e_{sqr}
PharmKG8k	49764	2	0.0000	$3.0 \cdot 10^{-4}$	$2.2 \cdot 10^{-5}$	$3.2 \cdot 10^{-5}$
FB15k237	20438	23	0.0017	0.0034	$7.5 \cdot 10^{-4}$	$9.1 \cdot 10^{-4}$
OpenEA	3826	242	0.0645	0.0670	0.0445	0.0588
DBpedia50	2095	521	0.1220	0.2545	0.1693	0.1996
WD50KT	45284	1998	0.0755	0.1205	0.0200	0.0259
WN18RR	2924	174	0.0411	0.0571	0.0426	0.0538
WN18	5000	18	-0.0375	0.0018	0.0077	0.0112
AristoV4	18414	116	0.0043	0.0173	0.0100	0.0125
CoDExLarge	30622	159	-0.0052	0.0054	0.0030	0.0042
DB100K	50000	672	-0.0117	0.0205	0.0091	0.0119

metrics. The full data can be provided upon request and the averages can be seen in Tab. 2 in the "baseline" columns.

To determine the effect of the split error, we processed the test sets so that they do not contain any ambiguous vertices, to force $\Delta_H^+ = 0$. This was done by removing all edges incident to them. The summary of the process and error estimates can be seen in Tab. 3. We evaluated the models again on the processed test sets. The results can be provided upon request and the effect on averages across models can be seen in Tab. 2 in the "change" columns.

We compared Δ_H^+ with the changes in evaluation. The entropy difference does not predict the change well, although the datasets with larger evaluation differences are generally not close to $\Delta_H^+ = 0$. Additionally, datasets with high entropy differences are concentrated among datasets that do not perform well.

We tested the ability our error estimation methods to predict changes in evaluation. The estimate from Eq. 3 shows the ability to predict the evaluation difference using linear regression. We tested some variations of the estimate,

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46

including adjustments from Eq. 4 and Eq. 5. The estimation with both adjustments performs best, although the unadjusted estimate still displays prediction capability. Fitting $f(x) = mx$, the best estimate achieves $MAE = 1.80 \cdot 10^{-3}$ and $MRE = 0.42$ with $m = 0.506$ for hits@10 and $MAE = 1.68 \cdot 10^{-3}$ and $MRE = 0.56$ with $m = 0.520$ for MRR. Therefore, a recommendation for roughly estimating the error could be to multiply the adjusted estimate by 0.5.

4 Discussion

The variety of datasets and models should show wide applicability of our framework. We tried to balance equal treatment of all models and datasets, with adjustment or search for each combination, and also time and hardware limitations. We acknowledge that these results do not represent the best performance, and some models or datasets may have been put at a disadvantage.

The method used to eliminate the split error was to remove all triples that contain an ambiguous vertex. It could be possible to remove just enough triples to achieve appropriate automorphism orbits. We made an attempt to isolate the edges causing the error, but all approaches resulted in retaining very few additional edges, as the test sets are already sparse.

The evaluation results on the unprocessed and processed test sets may not be directly comparable. The removal of some triples may introduce a new source of bias. However, removal of the ambiguous vertices did improve the metrics by a predictable magnitude in all cases, which shows the efficacy of our framework.

5 Conclusion

We proposed a framework for explaining and estimating a class of evaluation errors in KGE training that arise due to differences in topology in the training-test split. The split error estimate uses differences of vertex distinguishability based on automorphism orbits. We have demonstrated the ability to predict differences in link prediction scores using KGE models.

The proposed concepts can be applied to perform a more informed KGE evaluation, either by removing ambiguous vertices from test data, finding a better data split, or accounting for effects of split error. It is especially important when comparing multiple datasets, where each can have different characteristics.

To further explore this topic, more thorough testing could be performed on various datasets and KGE models. The proposed method could also be applicable to Graph Neural Networks. The effect of split error on validation sets could be evaluated. We also expect to develop a method of creating better train-test splits without removing parts of the data.

References

1. Ali, M., Berrendorf, M., Galkin, M., Thost, V., Ma, T., Tresp, V., Lehmann, J.: Improving inductive link prediction using hyper-relational facts (2021), https://arxiv.org/abs/2107.04894

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46

2. Ali, M., Berrendorf, M., Hoyt, C.T., Vermue, L., Sharifzadeh, S., Tresp, V., Lehmann, J.: PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. Journal of Machine Learning Research **22**(82), 1–6 (2021), http://jmlr.org/papers/v22/20-825.html
3. Bielecki, A., Stocki, R.: The concept of structural information and possible applications. Philosophical Problems in Science (Zagadnienia Filozoficzne w Nauce) (75), 157–183 (Dec 2023). https://doi.org/10.59203/zfn.75.654, https://zfn.edu.pl/index.php/zfn/article/view/654
4. Cai, H., Zheng, V.W., Chang, K.C.C.: A Comprehensive Survey of Graph Embedding: Problems, Techniques and Applications (Feb 2018)
5. Cervellera, C., Macciò, D.: Distribution-Preserving Stratified Sampling for Learning Problems. IEEE Transactions on Neural Networks and Learning Systems **29**(7), 2886–2895 (Jul 2018). https://doi.org/10.1109/TNNLS.2017.2706964, https://ieeexplore.ieee.org/document/7945296
6. Dehmer, M., Mowshowitz, A.: A history of graph entropy measures. Information Sciences **181**(1), 57–78 (Jan 2011). https://doi.org/10.1016/j.ins.2010.08.041, https://www.sciencedirect.com/science/article/pii/S0020025510004147
7. Duan, L., Chen, X., Liu, W., Liu, D., Yue, K., Li, A.: Structural Entropy Based Graph Structure Learning for Node Classification. Proceedings of the AAAI Conference on Artificial Intelligence **38**(8), 8372–8379 (Mar 2024). https://doi.org/10.1609/aaai.v38i8.28679, https://ojs.aaai.org/index.php/AAAI/article/view/28679
8. Goyal, P., Huang, D., Goswami, A., Chhetri, S.R., Canedo, A., Ferrara, E.: Benchmarks for Graph Embedding Evaluation (Aug 2019). https://doi.org/10.48550/arXiv.1908.06543
9. Hsieh, S.M., Hsu, C.C., Hsu, L.F.: Efficient method to perform isomorphism testing of labeled graphs. In: Gavrilova, M.L., Gervasi, O., Kumar, V., Tan, C.J.K., Taniar, D., Laganá, A., Mun, Y., Choo, H. (eds.) Computational Science and Its Applications - ICCSA 2006. pp. 422–431. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
10. Ivanov, S., Sviridov, S., Burnaev, E.: Rethinking graph classification problem in presence of isomorphism. Doklady Mathematics **110**, 312—-S331 (2024). https://doi.org/10.1134/S1064562424602385
11. Pujara, J., Augustine, E., Getoor, L.: Sparsity and Noise: Where Knowledge Graph Embeddings Fall Short. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 1751–1756. Association for Computational Linguistics, Copenhagen, Denmark (Sep 2017). https://doi.org/10.18653/v1/D17-1184
12. Singh, V., Pencina, M., Einstein, A.J., Liang, J.X., Berman, D.S., Slomka, P.: Impact of train/test sample regimen on performance estimate stability of machine learning in cardiovascular imaging. Scientific Reports **11**(1), 14490 (Jul 2021). https://doi.org/10.1038/s41598-021-93651-5, https://www.nature.com/articles/s41598-021-93651-5
13. Sun, Z., Wang, X., Ding, C., Fan, J.: Learning graph representation via graph entropy maximization. In: Proceedings of the 41st International Conference on Machine Learning. ICML'24, vol. 235, pp. 47133–47158. JMLR.org, Vienna, Austria (Jul 2024)
14. Trouillon, T., Welbl, J., Riedel, S., Éric Gaussier, Bouchard, G.: Complex embeddings for simple link prediction (2016), https://arxiv.org/abs/1606.06357

ICCS Camera Ready Version 2026
To cite this paper please use the final published version:
DOI: 10.1007/978-3-032-29924-6_46