Exploring the impact of literal transformations within Knowledge Graphs for Link Prediction

Moritz Blum
mblum@techfak.uni-bielefeld.de
Bielefeld University
Germany

Basil Ell
bell@techfak.uni-bielefeld.de
Bielefeld University
Germany
University of Oslo
Norway

Philipp Cimiano
cimianom@techfak.uni-bielefeld.de
Bielefeld University
Germany

ABSTRACT

Knowledge Graphs are relevant for many applications, but are inherently incomplete. Thus, Link Prediction methods have been proposed to infer new triples in order to complete a given Knowledge Graph. Many Link Prediction methods ignore literals, in spite of the fact that literals can express important information about entities not encoded in relations between entities. The existing methods that do incorporate literal information (e. g., *LiteralE*) introduce complex architectures by modifying the model or the loss-function. In our research paper, we propose a new approach that relies on graph transformations to transform a graph in such a way that existing Link Prediction methods can leverage the literal information. In particular, we define three transformations and evaluate them in comparison to state-of-the-art approaches. In most cases, the additional triples generated by our transformations lead to a performance increase and even state-of-the-art performance can be reached when comparing against *LiteralE*. It turned out that even a reductionistic transformation is able to archive comparable results like current, more complex, state-of-the-art approaches which incorporate literals.

CCS CONCEPTS

• **Theory of computation → Models of computation**; • **Networks → Network algorithms**; • **Information systems → Graph-based database models**.

KEYWORDS

Link Prediction, Literals, Knowledge Graph Embeddings

ACM Reference Format:
Moritz Blum, Basil Ell, and Philipp Cimiano. 2018. Exploring the impact of literal transformations within Knowledge Graphs for Link Prediction. In *Proceedings of International Joint Conference on Knowledge Graphs (IJCKG-2022).* ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Knowledge Graphs (KGs) are directed labelled multigraphs described by a set of triples of the form (subject, relation, object/literal) encoding structured information about entities. They have become a relevant way to represent knowledge and are used in various domains and applications, e. g., in entity disambiguation [21] or in question answering [3, 20]. As KGs are inherently incomplete, and are missing facts due to various reasons, there have been many attempts to develop methods that can predict missing triples, a task that is called Link Prediction.

Most of the Link Prediction models and benchmark datasets do not take literal information of attributive triples (i. e., triples with a literal as object) into account, as contained in many RDF KGs such as *Freebase* or *DBpedia*, and only consider relational triples (i. e., triples with an entity as object). The reason might be that literals add a complexity to the task, requiring more advanced models which are able to handle both, graph-structured and unstructured literal data. However, the information encoded in literals can be very valuable for the task of Link Prediction, as some approaches have demonstrated [8]. Consider the example of *Leo Tolstoy* and *Sophia Tolstoya*, illustrated in Figure 1; for these two entities, a relation can be predicted based only on the similar family name and the shared wedding day. We use this example involving *Leo Tolstoy* and *Sophia Tolstoya* throughout the paper to illustrate our transformations.

State-of-the-art methods such as *LiteralE* [8] embed literals into vectorial representations and use these representations as additional input to the model or the scoring function. In this research paper, we propose a different method, consisting of enriching the neighborhood structure of entity nodes by transforming literal information into relational triples. This allows to reuse existing Link Prediction models without modifications in comparison to other existing methods such as, e. g., *LiteralE*. Therefore, we motivate and formally describe three different transformations as set-operations: *Literal2Entity* uses the complete literal information without modifications to create a URI, *Datatype2Entity* reduces the literal information to datatype and language tag and then creates a URI, and *Value2Shingles* shingles string literals and enriches the graph by these shingles transformed to URIs. We apply the transformations to the most common Link Prediction benchmark datasets, and evaluate models trained with these created datasets. We compare to three established models as baselines and show that the best graph transformation is a filtered version of *Datatype2Entity*, which improves results by 11% compared to *DistMult* that can not take into account literals. Furthermore, our approach achieves comparable results

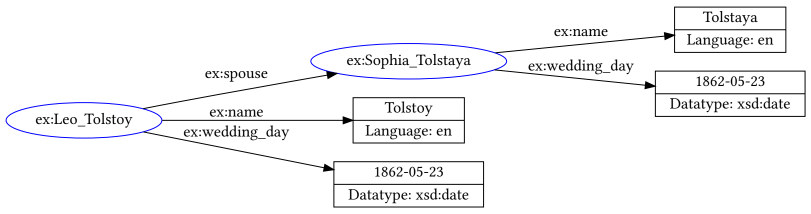

Figure 1: RDF graph containing literals of type $xsd{:}date$ **and** $xsd{:}string$ **with language tag** en**. One can see that** $ex{:}LeoTolstoy$ **and** $ex{:}SophiaTolstaya$ **share the same wedding day and have a similar name.**

to *LiteralE*, which can be considered as state-of-the-art. It turned out that in some cases even the reductionistic *Datatype2Entity* transformation is able to achieve comparable results like current state-of-the-art approaches which incorporate literals. This raises the question, whether the complexity of state-of-the-art architectures like *LiteralE* is needed to allow the usage of literal information in Link Prediction.

2 RELATED WORK

Link Prediction is a research task that has received considerable attention by the Semantic Web Community in the last five years. Whereas rule-based approaches aim at identifying frequent patterns in the KG, e. g., *AMIE* proposed by Galárraga et al. [6], statistical approaches make use of machine learning principles. Moreover, hybrid approaches try to combine the advantages of both methods, e. g., Wang et al. [16] make use of logical rules to refine embedding models. The focus of current research is mainly on machine learning approaches, as they have shown good scalability and generalizability on large KGs. For example, *DistMult*, proposed by Yang et al. [19], uses a multiplicative scoring function for embedding relational triples. *ComplEx*, proposed by Trouillon et al. [14], modifies *DistMult* to work with vectors with complex values, which leads to an increased performance in modeling asymmetric relationships.

Even though many large KGs such as *DBpedia* [9], *Freebase* [1] and *Wikidata* [15] contain literals,[1] relatively little attention has been devoted so far to methods taking literal information into account, as they introduce an additional complexity to the Link Prediction task.

Literals can be multi-modal (e. g., textual, numerical), requiring models that are able to handle multiple types of data. More recently, incorporating literals into KG embeddings or Link Prediction models has become a research topic as several approaches like the ones by Kristiadiet al. [8] or Xie et al. [18] pointed out the value of literals for Link Prediction task.

The majority of methods using literals can only handle literals of a certain datatype, such as textual literals, e. g., textual descriptions of entities. One example is the approach proposed by Xie et al. [18], where structure-based and description-based representations are learned simultaneously in the same vector space by a Link Prediction model, which uses text embeddings if textual features are available. State-of-the-art methods have been developed with the purpose to alleviate the restriction to textual data, allowing multimodal literal input. Pezeshkpour et al. [12] proposed a method that combines existing relational models with separate datatype-specific neural encoders. For attributive triples, these encoders are used to obtain a literal representation, allowing all triples to be scored by relational Link Prediction models. *LiteralE*, proposed by Kristiadi et al. [8], is a multi-modal approach, too, that enriches embeddings with literal feature vectors by a learnable parametrized function. This function maps an entity embedding and a feature vector to a new vector of the same dimension as the entity embedding, which is then used as input for the Link Prediction scoring function. The approach can be applied to, e. g., *DistMult* and *ComplEx*.

The existing methods propose either completely new architectures [18] or modifications of the models [8] or loss functions [17] and are thus only applicable to certain models and frequently require a new implementation or re-implementation of core components.

Transformations of KGs have been proposed before. E. g., Chen et al. [4] propose a method for literal canonicalization by replacing literals with existing entities from the KG or with new entities typed by using classes from the KG. In their approach, the string literal ”*Germany*”^^@*en* would be replaced by its Wikidata entity *Q*183. However, the impact of these methods on Link Prediction has not been investigated so far to the best of our knowledge.

Our approach makes use of transformations, too: it takes unstructured literal data and represents it in a structured way by enriching the neighborhood structure of entity nodes in order to be able to use state-of-the-art Link Prediction models, like *DistMult* or *ComplEx*. Furthermore, we investigate the impact of added triples throughout transformations of literal data to widely-used Link Prediction models.

3 GRAPH TRANSFORMATIONS

KGs are directed multigraphs with labeled edges connecting entities to entities via relational triples and connecting entities to literals via attributive triples. For simplicity, we only consider RDF graphs, but our approach is applicable to property graphs as well.

An RDF graph G is a set of triples $(s, p, o) \in (\mathcal{U} \cup \mathcal{B}) \times \mathcal{U} \times (\mathcal{U} \cup \mathcal{B} \cup \mathcal{L})$, where $\mathcal{U}$, $\mathcal{B}$ and $\mathcal{L}$ are pairwise disjoint sets of URIs, blank nodes and literal values with $\mathcal{L} = \mathcal{L}_d \cup \mathcal{L}_l$. $\mathcal{L}_d$ is the set of datatyped values, i.e., tuples of the form $(v, d) \in \Sigma^* \times \mathcal{D}$ where Σ^* is the set of strings over the alphabet Σ and $\mathcal{D}$ is a set of URIs of datatypes. $\mathcal{L}_l$ is the set of language-tagged values, i.e, tuples of the form $(v, l) \in \Sigma^* \times \mathcal{T}$ where $\mathcal{T}$ is a set of language tags. Without loss of generality, we assume G to adhere to the RDF 1.1. recommendation, such that G does not contain plain literals. If it

[1]Amount of attributive triples: *DBpedia* ≈40%, *Freebase* ≈57%, *Wikidata* ≈41% [5]

does, however, one can transform each plain literal into a datatyped literal by assigning the datatype *xsd*:*string*.

Given an RDF graph G that contains relational triples G_E and attributive triples in G_L. Thus:

$$G_E = \{(s, p, o) \mid (s, p, o) \in G \text{ s.t. } o \in \mathcal{U} \cup \mathcal{B}\}$$

$$G_L = \{(s, p, o) \mid (s, p, o) \in G \text{ s.t. } o \in \mathcal{L}\}$$

The core idea of our approach consists in the application of certain transformations to G_L, yielding a transformed set G'_L. The RDF graph on which Link Prediction is carried out is then the union of G_E and G'_L, that is $G' = G_E \cup G'_L$. In the following, we present three transformations to create G'_L. We use one example throughout the paper to illustrate these transformations, that is the triple representing the wedding date of Leo Tolstoy, which is given by: (*ex*:*Leo_Tolstoy*, *ex*:*wedding_day*, "1862−05−23"^^*xsd*:*date*) with the literal "1862−05−23"^^*xsd*:*date*.

Transformation 1 – Literal2Entity (abbreviated as L2E): The *Literal2Entity* transformation, as the name suggests, transforms every literal into an entity, thus creating a URI. This ensures that entities that share the same literal value with the same datatype are connected in the resulting graph. In our example, the result of the transformation is shown in Figure 2, that is, the literal "1862−05−23"^^*xsd*:*date* transformed into the URI representation *ex*:1862−05−023_*xsd_date*.

More formally, the resulting transformed graph G'_L is defined as follows:

$$G'_L = \{(s, p, \mathsf{urify}(v \oplus x)) \mid (s, p, o) \in G \land o = (v, x) \in \mathcal{L}\} \quad (1)$$

The bijective function $\oplus$ concatenates two strings by inserting an unambiguous separator, and the bijective function urify is used to transform a character sequence into a valid URI.

In terms of complexity, in the worst case, the number of entities grows by $|G_L|$. Furthermore, the number of additional relations equals the number of attributive relations, and, therefore, the number of created triples is $|G_L|$.

Transformation 2 – Datatype2Entity (abbreviated as D2E): This transformation represents the literal's datatype as an entity and sets it into relation to the subject entity according to the attributive triple. As a result, all entities connected with a literal of the same datatype become connected. In our outlined example, for the literal "1862−05−23"^^*xsd*:*date*, the URI *ex*:*xsd_date* is created. Figure 3 shows the result of the *Datatype2Entity* transformation to our example graph about *Leo Tolstoy* and *Sophia Tolstaya*.

To be precise, the *Datatype2Entity* transformation is defined by:

$$G'_L = \{(s, p, \mathsf{urify}(x)) \mid (s, p, o) \in G \land o = (v, x) \in \mathcal{L}\} \quad (2)$$

As the concrete literal value gets lost, the number of additional entities is $|\mathcal{D}_G \cup \mathcal{T}_G|$, where D_G denotes the set of language tags and $\mathcal{T}_G$ denotes the set of URIs of datatypes in the considered graph G. Nevertheless, in the worst case, the number of created triples is $|G_L|$.

Transformation 3 – Value2Shingles (abbreviated as V2S): This transformation relies on the computation of the k-shingles occurring in any textual literal, introducing a URI for each distinct shingle and linking the corresponding subject entity to each of these shingle entities by the attributive relation. k-shingles are character subsequences of length k which can be used to compute a similarity score for two different strings. Instead, we connect entities sharing a certain shingle in the literals. Assuming shingles of size 4, in our example *ex*:*Tolstoy* would be linked to four entities *ex*:*Tols*, *ex*:*olst*, *ex*:*lsto*, and *ex*:*stoy*, corresponding to the 4-shingles of "Tolstoy", as shown in Figure 4.

Value2Shingles is formally defined as follows:

$$G'_L = \{(s, p, \mathsf{urify}(w)) \mid (s, p, o) \in G \land o = (v, x) \in \mathcal{L} \\ \land\ x = xsd{:}string \land w \in \omega(k, v)\} \quad (3)$$

Here, $\omega(k, v)$ is the set of all k-shingles that occur in the literal value v.

The upper bound on the number of additionally created entities is $(m - k) * |G_L|$, where m denotes the maximum length of a string literal. As for the other transformations, the number of additional relations equals the number of attributive relations for this transformation. However, for *Value2Shingles*, the number of triples can grow linearly with the given amount of literal data: $(m - k) * |G_L|$, where again m denotes the maximum length of a string literal.

4 EXPERIMENTAL SETTINGS

In this section, we describe the datasets typically used for the evaluation of Link Prediction and the datasets we have obtained by enriching these datasets with literals. Further, we describe the Link Prediction methods that we use in our experiments to explore the impact of our graph transformations: *DistMult*, *ComplEx*, and *LiteralE*. Finally, we describe the evaluation methodology that makes use of MRR (Mean Reciprocal Rank) and Hits@k according to Borders et al. [2].

4.1 Datasets

Our experiments are evaluated on the Link Prediction datasets *FB15k* [2], *FB15k-237* [13], *YAGO3-10* [11], and *LitWD48K* [7].

FB15k and *FB15k-237* are two of the most widely used datasets for evaluating Link Prediction approaches and are derived from *Freebase*. *FB15k-237* is a subset of *FB15k*, created by removing inverse relations. We enriched the subset of the KG intended for training by attributive triples contained in *Freebase*. The most frequent relation *rdf.freebase.com/ns/type.object.key* does not contain information relevant for our approach, as its literal value can not be usefully interpreted, therefore we removed it from the dataset. *YAGO3-10*, also an established Link Prediction dataset, is a subset of *YAGO3* [10] that only contains entities that occur in at least ten relations. Even though *YAGO3-10* does not contain literals, they can be derived from *YAGO3*. We used the literals as published by Pezeshkpour et al. [12]. In contrast to these datasets, the recently published *LitWD48K* dataset is a subset of *Wikidata* and is explicitly designed to contain literals for Link Prediction.

For the triples induced by *Literal2Entity*, some new entities are connected only to one entity. These entities do not add knowledge that can be used by our approach, and, therefore, we remove them to simplify the learning and reduce feature dimensionality and memory consumption. Furthermore, we found out that *Datatype2Entity*

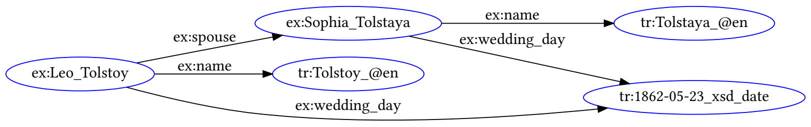

Figure 2: Example: *Literal2Entity*, applied to the graph depicted in Figure 1. This transformation creates a connection between $ex{:}LeoTolstoy$ and $ex{:}SophiaTolstaya$ by the shared entity $tr{:}1862{-}05{-}23\_xsd\_date$ because both share exactly the same wedding day.

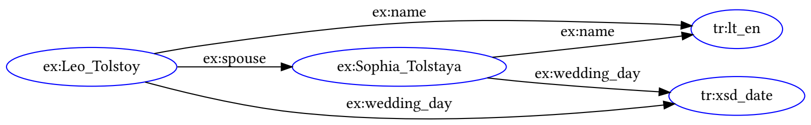

Figure 3: *Datatype2Entity*, applied to the graph depicted in Figure 1. The objects $tr{:}@en$ and $tr{:}xsd\_date$ representing the literal's datatypes and language tags are connected to the subjects of the attributive triples through the attributive relation.

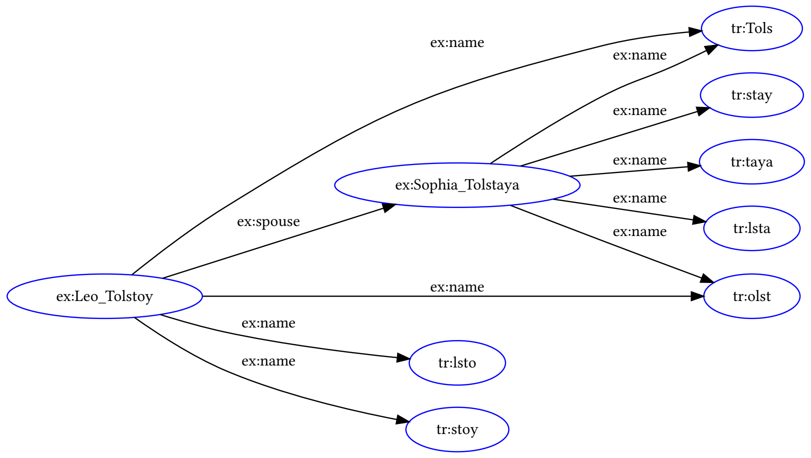

Figure 4: Example: *Value2Shingles*, applied to the graph depicted in Fig. 1. Entities are created by shingling the string literals with a shingle-size of 4. As the names of $ex{:}LeoTolstoy$ and $ex{:}SophiaTolstaya$ are very similar, they share some shingles that are represented by entities.

creates numerous triples and increases the size of the graph. Therefore, in addition to the complete graph *FB15k-237*, we consider different subsets obtained by filtering the triples. We decided to build three subsets based on the frequency of each relation, what has a major impact on the number of triples. The chosen subsets are: only very frequent relations ($> 10k$), the second subset contains only rare ones ($< 1k$), and the third subset is in-between, filtering out the most frequent and the most infrequent ones ($1k < x < 10k$). For *Value2Shingles* we consider shingles of size 7 and filter out all shingles occurring less than $1k$ times across all datasets. However, the shingle-size is a parameter we expect to be worth to be tuned in future investigations. Furthermore, we evaluated the impact of text normalization, lowercasing and lemmatizing, applied to the literals before the transformations are carried out. Table 1 shows an overview of the characteristics of all KGs that we used for training in our experiments.

Actually, *Datatype2Entity* creates more triples than *Literal2Entity*. Whereas for *Literal2Entity* numerous triples with entities connected to only one other entity can be removed, this pattern does not occur frequently for *Datatype2Entity*.

4.2 Link Prediction Models

We evaluate our approach on different models: the semantic matching models *DistMult* and *ComplEx*, and one of the most prominent Link Prediction methods to incorporate literals: *LiteralE*. We used the implementation published with the *LiteralE* paper[2] with the same hyperparameters as described in the corresponding publications [8, 14] for *DistMult, Complex, LiteralE*: learning rate 0.001, batch size 128, embedding dimensionality 200, optimizer adam, negative sampling ratio 1, embedding dropout probability 0.2, label smoothing 0.1. As our transformations introduce additional triples, the training time increases, even though the maximum number of

[2]https://github.com/SmartDataAnalytics/LiteralE

Dataset	#entities	#relations	#triples
FB15k Base	14,951	1,345	483,142
FB15k L2E	86,472	1,875	702,420
FB15k D2E	15,043	2,201	2,896,350
FB15k V2S$_{\text{size=7; >1k}}$	22,084	1,425	9,059,600
FB15k-237 Base	14,505	237	272,115
FB15k-237 L2E	86,059	767	491,393
FB15k-237 L2E lc.	86,497	779	495,900
FB15k-237 L2E lemm.	86,684	786	498,330
FB15k-237 D2E	14,633	1,093	2,685,323
FB15k-237 D2E$_{>1k}$	14,633	423	2,640,896
FB15k-237 D2E$_{<10k}$	14,562	1,023	821,602
FB15k-237 D2E$_{1k<x<10k}$	14,633	423	2,640,896
FB15k-237 V2S$_{\text{size=7; >1k}}$	21,457	317	8,854,676
FB15k-237 V2S$_{\text{size=7; >1k}}$ lc.	22,195	324	9,766,405
FB15k-237 V2S$_{\text{size=7; >1k}}$ lemm.	23,020	323	11,418,696
YAGO3-10 Base	123143	37	1,079,040
YAGO3-10 L2E	124,700	43	1,191,101
YAGO3-10 D2E	123,173	43	1,297,765
YAGO3-10 V2S$_{\text{size=7; >1k}}$	127,237	38	11,297,860
LitWD48K Base	62,135	258	373857
LitWD48K L2E	96,082	509	608607
LitWD48K D2E	62,147	553	1,008,722
LitWD48K V2S$_{\text{size=7; >1k}}$	69,071	371	17,124,453

Table 1: Characteristics in terms of number of entities, number of relations, and number of triples of the training datasets derived from *FB15k, FB15k-237, YAGO3-10,* and *LitWD48k.* The datasets denoted with Base are the datasets without literal data. *Literal2Entity* is abbreviated as *L2E, Datatype2Entity* is abbreviated as *D2E,* and *Value2Shingles* is abbreviated as *V2S.* Moreover, *lc.* denotes lowercasing, and *lemm.* denotes lemmatization.

epochs is set to 100 (200 for *LitWD48K*) as in the original experiments. For early stopping, the Mean Reciprocal Rank (MRR) on the validation set is used.

The following evaluation is performed on the defined test sets of the considered datasets. The enrichment and transformation is only applied to the training set, since in the considered transductive learning setting the relevant information is learned as entity and relation embeddings. Therefore, the setting of predicting links between entities is the same as for *DistMult, Complex* or *LiteralE.*

4.3 Evaluation Method

The models are compared via the filtered mean reciprocal rank (MRR) metric, as proposed by Bordes et al. [2]. For each triple in the test set, first the subject and second the object entity is corrupted by replacing it by all other entities in the graph. Then, the score for each triple is computed and used to rank the test triple among all of those triples not already contained in the graph by sorting ascending. Only triples not contained in the graph are considered during ranking, to not cause valid triples to increase the rank of the test triples. As a result, the MRR is the mean of the multiplicative inverse of all computed ranks:

$$MRR = \frac{1}{|I|} \sum\nolimits_{r \in I} r^{-1} \tag{4}$$

, where I is the set of all computed ranks. In addition, the filtered Hits@k scores are used for comparison. By using the computed ranks, the Hits@k are defined as:

$$Hits@k = |\{r \in I \mid r \leq k\}| \tag{5}$$

5 RESULTS

We report results comparing the runs of the two methods (*DistMult* and *ComplEx*) with and without transformed literals in order to investigate the impact of including literal information into these existing models. We report MRR, Hits@10, Hits@3, and Hits@1 scores for all configurations and runs.

For *DistMult* and *ComplEx*, the transformations lead to an up to 11% increased MRR in comparison to the base models. Especially in the case where the performance of the baseline is low, we observe a substantial performance increase reached through our transformations. For example, the largest improvement yielded by our transformation-based approach without additional filtering or normalization was achieved on the *LitWD48K* dataset using the *DistMult* model. In particular, for the *LitWD48k* dataset, we observe that our *Literal2Entity* and *Datatype2Entity* transformations outperform both the baseline and *LiteralE.* This shows the strength of our method on datasets that natively include literals. On *YAGO3-10*, however, our transformation-based approaches performs worse than the baseline for some *ComplEx* models. Interestingly, this is the case for *LiteralE*, too. We suspect that the relational triples already contain the important and reliable information to archive high scores, and the literals do not add knowledge which is usable by *ComplEx.* On *FB15k-237*, *LiteralE* achieves the best performance when *DistMult* is used. When considering *ComplEx*, however, our transformation-based approach outperforms *LiteralE.*

Overall, our transformation-based approach and *LiteralE* show comparable performance, with both outperforming each other in some settings. All scores are shown in Table 2; it can be appreciated that all scores (MRR, Hits@10, Hits@3, Hits@1) show the same trend, corroborating our results. It was unexpected for us that *Datatype2Entity*, which does not take the actual value into account, is able to outperform state-of-the-art *LiteralE* in some settings.

Datatype2Entity achieves the best scores across all transformations and, therefore, we investigate the effect of filters. Overall, the best absolute improvement upon the baseline of 11% in MRR is achieved by *Datatype2Entity* with filtering across all transformations. *DistMult*, trained with *Datatype2Entity*$_{1k<x<10k}$, reaches an MRR score of 0.313 which is comparable to *LiteralE.* The reason for the increased performance might be that the filters restrict the available data and therefore might reduce data that does not add useful information. Overall, the filtering of *Datatype2Entity* increases the performance in most cases with respect to the baseline on all scores, as shown in Table 3.

Furthermore, we investigate the impact of lexical normalization operations to conflate some URIs created for literals by applying lowercasing and lemmatization prior to the graph transformations. We perform this investigation on the dataset with the lowest MRR scores, *FB15k-237*, and report the results for *Literal2Entity* and

	FB15k		FB15k-237		YAGO3-10		LitWD48K	
	DistMult	ComplEx	DistMult	ComplEx	DistMult	ComplEx	DistMult	ComplEx
	MRR							
Base	**0.670**	0.693	0.281	0.290	0.479	**0.507**	0.336	0.315
LiteralE	0.550	**0.750**	**0.316**	0.272	0.466	0.473	0.333	0.268
L2E	0.607	0.658	0.284	0.297	0.471	0.506	0.354	**0.339**
D2E	0.466	0.534	0.297	0.308	**0.500**	0.459	**0.359**	0.338
V2S	0.524	0.533	0.303	**0.311**	0.409	0.414	0.291	0.283
	Hits@10							
Base	**0.818**	0.832	0.437	0.447	0.639	**0.659**	0.480	0.442
LiteralE	0.738	**0.855**	**0.485**	0.432	0.617	0.610	0.476	0.393
L2E	0.770	0.809	0.444	0.458	0.629	0.657	0.500	0.465
D2E	0.661	0.706	0.455	0.467	**0.653**	0.649	**0.520**	**0.483**
V2S	0.706	0.699	0.462	**0.470**	0.571	0.597	0.426	0.404
	Hits@3							
Base	**0.721**	0.744	0.307	0.320	0.524	**0.552**	0.360	0.341
LiteralE	0.616	**0.789**	**0.348**	0.299	0.510	0.516	0.354	0.293
L2E	0.659	0.708	0.312	0.326	0.515	0.548	0.378	0.364
D2E	0.525	0.581	0.328	0.336	**0.543**	0.535	**0.384**	**0.365**
V2S	0.578	0.577	0.327	**0.342**	0.456	0.461	0.310	0.302
	Hits@1							
Base	**0.589**	0.615	0.202	**0.299**	0.396	0.427	0.264	0.248
LiteralE	0.444	**0.690**	**0.230**	0.193	0.385	0.397	0.262	0.203
L2E	0.518	0.575	0.203	0.216	0.388	**0.428**	**0.280**	**0.271**
D2E	0.358	0.443	0.216	0.227	**0.420**	0.413	**0.280**	0.263
V2S	0.426	0.445	0.223	0.229	0.322	0.324	0.223	0.222

Table 2: Scores (MRR, Hits@10, Hits@3, Hits@1) obtained for the datasets *FB15k*, *FB15k-237*, *YAGO3-10*, and *LitWD48K*, without filter or normalization. Base denotes the dataset containing only the relational triples as originally published. All experiments were run on our machines with the same settings. One exception are the models trained on *LitWD48K* which are run for 200 epochs.

	DistMult	ComplEx	DistMult	ComplEx
	MRR		Hits@10	
Base	0.281	0.290	0.437	0.447
LiteralE	**0.316**	0.272	0.485	0.432
D2E	0.297	0.308	0.455	0.467
$D2E_{>1k}$	0.303	0.306	0.468	0.468
$D2E_{<10k}$	0.310	**0.313**	**0.490**	**0.479**
$D2E_{1k<x<10k}$	0.313	0.311	0.483	0.476
	Hits@3		Hits@1	
Base	0.307	0.320	0.202	**0.299**
LiteralE	0.348	0.299	0.230	0.193
D2E	0.328	0.336	0.216	0.227
$D2E_{>1k}$	0.334	0.337	0.219	0.223
$D2E_{<10k}$	**0.356**	**0.345**	**0.232**	0.228
$D2E_{1k<x<10k}$	0.346	0.343	0.227	0.227

Table 3: Scores (MRR, Hits@10, Hits@3, Hits@1) obtained for different Datatype2Entity filters (*Datatype2Entity*$_{>1k}$, *Datatype2Entity*$_{<10k}$, *Datatype2Entity*$_{1k<x<10k}$) on the dataset *FB15k-237*.

	L2E	V2S	L2E	V2S
	MRR		Hits@10	
no modification	0.285	**0.303**	0.444	**0.462**
lowercasing	**0.293**	0.301	**0.451**	0.459
lemmatization	0.280	0.298	0.439	0.452
	Hits@3		Hits@1	
no modification	0.312	**0.327**	0.203	**0.223**
lowercasing	**0.322**	**0.327**	**0.212**	0.221
lemmatization	0.307	0.325	0.200	0.219

Table 4: Scores (MRR, Hits@10, Hits@3, Hits@1) obtained for the normalization upfront Literal2Entity and Value2Shingles on the dataset *FB15k-237*.

Value2Shingles. Lowercasing slightly improves the results in most cases. The strongest positive impact of lowercasing is achieved for the *Literal2Entity* transformation, leading to a 3% increased MRR. In contrast, lemmatization in most cases leads to worse performance. The scores of this evaluation are shown in Table 4.

To analyze the impact of the transformations on Link Prediction models, we analyzed the specific predictions made by the different models on the test dataset. Overall, *Datatype2Entity*$_{<10k}$ increases the MRR score in 82.5% (185/224) of all relations in *FB15k-237* in comparison to the base model. Especially for the more frequent relations, the model shows a great increase in performance. However, since the relation-wise MRR shows a high variance, even for frequent relations, further analysis of the predictions has shown to be difficult. A comparison of three runs on the same *FB15k-237* graph has shown that the intersection of triples ranked in Hits@10 is only around 40% of all triples. The same observation was made for our proposed transformation datasets.

6 CONCLUSION

In this paper, we presented three graph transformations that encode literal information in the Knowledge Graphs (KGs) via additional entities and relations such that they can be leveraged by Link Prediction methods. In comparison to other methods using latent literal data representations, we apply a transformation directly on the KG, thus modifying the input, without requiring extensions to a model. This makes our method usable with state-of-the-art Link Prediction models. We have examined the impact of our graph transformation approach with respect to four existing Link Prediction methods: *DistMult*, *ComplEx*, *LiteralE*. The achieved MRR score shows a performance increase of up to 11% in comparison to the base model that does not use literal information. Comparing to *LiteralE* as state-of-the-art Link Prediction model incorporating literal information, we show that our method achieves comparable results without requiring any model extension. We thus provide a strong new baseline for the inclusion of literal information. As in some cases, even the reductionistic *Datatype2Entity* transformation is able to achieve comparable results like current state-of-the-art approaches which incorporate literals, it is debatable whether the complexity of these methods is needed to allow Link Prediction the usage of literal information.

In our work, only *LitWD48K* natively includes literals. We have indeed shown a strong positive impact of exploiting literal information in this dataset. Future work should thus invest into the creation of further datasets with literals to support a robust analysis and evaluation of Link Prediction methods exploiting literals. Moreover, future work will be devoted to the development of further transformations. For string literals, advanced NLP methods, e. g., entity linking, could be used in new transformations. For numerical datatypes, the discretization of values could be examined. Furthermore, it would be interesting to take datatype semantics into account, e. g., literals of type $xsd{:}dateTime$ can be transformed into literals of type $xsd{:}date$.

Supplemental Material. The code and data to reproduce our experiments is published on GitHub. [3]

Acknowledgments. This research was carried out in the context of DiProMag, a BMBF (German Federal Ministry of Education and Research) funded research project under Grant No. 13XP5120B. Furthermore, this work has been supported by the SIRIUS centre: Norwegian Research Council project No 237898.

REFERENCES

[1] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data.* 1247–1250.

[2] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013).

[3] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *Joint European conference on machine learning and knowledge discovery in databases.* Springer, 165–180.

[4] Jiaoyan Chen, Ernesto Jiménez-Ruiz, and Ian Horrocks. 2019. Canonicalizing knowledge base literals. In *International semantic web conference.* Springer, 110–127.

[5] Michael Färber, Frederic Bartscherer, Carsten Menne, and Achim Rettinger. 2018. Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and Yago. *Semantic Web* 9 (2018), 77–129.

[6] Luis Antonio Galárraga, Christina Teflioudi, Katja Hose, and Fabian Suchanek. 2013. AMIE: association rule mining under incomplete evidence in ontological knowledge bases. In *Proceedings of the 22nd international conference on World Wide Web.* 413–422.

[7] Genet Asefa Gesese, Mehwish Alam, and Harald Sack. 2021. LiterallyWikidata - A Benchmark for Knowledge Graph Completion Using Literals. In *International Semantic Web Conference.* Springer, 511–527.

[8] Agustinus Kristiadi, Mohammad Asif Khan, Denis Lukovnikov, Jens Lehmann, and Asja Fischer. 2019. Incorporating literals into knowledge graph embeddings. In *International Semantic Web Conference.* Springer, 347–363.

[9] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. 2015. DBpedia – a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web* 6, 2 (2015), 167–195.

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

[11] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2012. Factorizing yago: scalable machine learning for linked data. In *Proceedings of the 21st international conference on World Wide Web.* 271–280.

[12] Pouya Pezeshkpour, Liyan Chen, and Sameer Singh. 2018. Embedding multimodal relational data for knowledge base completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing.* Association for Computational Linguistics, 3208–3218.

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

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

[15] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.

[16] Quan Wang, Bin Wang, and Li Guo. 2015. Knowledge base completion using embeddings and rules. In *Twenty-fourth international joint conference on artificial intelligence.*

[17] Yanrong Wu and Zhichun Wang. 2018. Knowledge graph embedding with numeric attributes of entities. In *Proceedings of The Third Workshop on Representation Learning for NLP.* 132–136.

[18] Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[19] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[20] Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with Freebase. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.* 956–966.

[21] Zhicheng Zheng, Xiance Si, Fangtao Li, Edward Y Chang, and Xiaoyan Zhu. 2012. Entity disambiguation with Freebase. In *ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, Vol. 1. IEEE, 82–89.

[3] https://github.com/moritzblum/literal_transformations