Augmenting Knowledge Graphs for Better Link Prediction

Jiang Wang, Filip Ilievski, Pedro Szekely and Ke-Thia Yao
USC Information Sciences Institute
{jiangwan, ilievski, pszekely, kyao}@isi.edu

Abstract
Embedding methods have demonstrated robust performance on the task of link prediction in knowledge graphs, by mostly encoding entity relationships. Recent methods propose to enhance the loss function with a literal-aware term. In this paper, we propose KGA: a knowledge graph augmentation method that incorporates literals in an embedding model without modifying its loss function. KGA discretizes quantity and year values into bins, and chains these bins both horizontally, modeling neighboring values, and vertically, modeling multiple levels of granularity. KGA is scalable and can be used as a pre-processing step for any existing knowledge graph embedding model. Experiments on legacy benchmarks and a new large benchmark, DWD, show that augmenting the knowledge graph with quantities and years is beneficial for predicting both entities and numbers, as KGA outperforms the vanilla models and other relevant baselines. Our ablation studies confirm that both quantities and years contribute to KGA’s performance, and that its performance depends on the discretization and binning settings. We make the code, models, and the DWD benchmark publicly available to facilitate reproducibility and future research.

1 Introduction
Hyperrelational knowledge graphs (KGs), like Wikidata [Vrandečić and Krötzsch, 2014], formalize knowledge as statements. A statement consists of a triple with key-value qualifiers and references that support its veracity. A statement represents either a relationship between two entities, or attributes a literal value (date, quantity, or string) to an entity. Nearly half of the statements in Wikidata are entity relationships, a third of them are string-valued, and the remaining (around 15%) statements attribute quantities and dates to entities. Intuitively, entity and literal statements complement each other to form a comprehensive view of an entity.

Considering the sparsity and the inherent incompleteness of KGs, the task of link prediction (LP) has been very popular, producing methods based on matrix factorization/decomposition, KG paths, and embeddings [Wang et al., 2021]. While most LP methods only consider statements about two entities (Qnodes), some methods recognize the relevance of literals [Gesese et al., 2019]. Literal-aware methods [Xiao et al., 2017] predominantly learn a representation about the textual descriptions of an entity and combining it with its structured representation.

Quantities and dates have seldom been considered, despite their critical role in contextualizing the LP task. A person’s date of birth or a company’s founding year anchor the prediction context to a specific time period, whereas the population of a country indicates how big a country is. Recent methods that incorporate numeric literals into KG embeddings add a literal-aware term to the scoring function [García-Durán and Niepert, 2017; Kristiadi et al., 2019], or modify the loss function of the base model in order to balance capturing the KG structure and numeric literals [Wu and Wang, 2018; Tay et al., 2017; Feng et al., 2019]. These methods leverage the knowledge encoded in literals in order to improve link prediction performance, but they introduce additional parameters and model-specific clauses, which limits their scalability and generalizability across embedding models.

In this paper, we investigate how to incorporate quantity and date literals into embedding-based link prediction models without modifying their loss function. Our contributions are:
1. **KGA**: a Knowledge Graph Augmentation method for incorporating quantity and year literals into KGs, by chaining the literals vertically, for different granularities of discretization, and horizontally, for neighboring values to each granularity level. KGA is scalable and can be used as a pre-processing step for any existing KG embedding model.
2. **DWD**, an LP benchmark that is orders of magnitude larger than existing ones, and includes quantities and years, thus addressing evaluation challenges with size and overfitting.
3. An extensive LP evaluation, showing the superiority of KGA in terms of generalizability, scalability, and accuracy. Ablations study the individual impact of quantities and years, and the effect of discretization strategies and bin sizes.

1Extended version of our paper with supplementary material can be found on arXiv: https://arxiv.org/abs/2203.13965.
2https://tinyurl.com/56vyj2y, accessed on 13/01/22.

3We focus on embedding-based methods, as these are superior over matrix factorization and path-based methods [Lu et al., 2020].
We define knowledge graph with numeric triples benchmark: https://github.com/Otamio/KGA/.

4. Public release of our code, resulting models, and the DWD binning for the attribute triple (Q76, P569, “1961”), which specifies the year of birth of Barack Obama. We use $b = 4$ bins, quantile-based binning, and chaining in each level mode.

2 Task Definition

We define knowledge graph with numeric triples as a union of entity-valued and numeric statements, formally, $G = (s, r, o) \cup (e, a, v)$, where $(s, r, o) \in \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ and $(e, a, v) \in \mathcal{E} \times \mathcal{R} \times \mathbb{R}$. The entity triples can be formalized as a set of triples (facts), each consisting of a relationship $r \in \mathcal{R}$ and two entities $s, o \in \mathcal{E}$, referred to as the subject and object of the triple. The numeric triples consist of an attribute $a \in \mathcal{R}$ with one entity $e \in \mathcal{E}$ and one value $v \in \mathbb{R}$. We formalize entity link prediction as a point-wise learning-to-rank task, with an objective to learn a scoring function $\psi : (\mathcal{E} \times \mathcal{R} \times \mathcal{E}) \to \mathbb{R}$. Given an input triple $x = (s, r, o)$, its score $\psi(x) \in \mathbb{R}$ is proportional to the likelihood that the fact encoded by $x$ is true. We pose numeric link prediction as a point-wise prediction task, with an objective to learn a function $\phi : (\mathcal{E} \times \mathcal{R}) \to \mathbb{R}$. Given an entity-attribute pair, $(e, a)$, the output $\phi(e, a)$ is the numeric value of attribute $a$ of entity $e$.

3 KGA

The key idea of KGA is to augment the KG with quantities and years before training. KGA consists of two steps: 1) discretization and bin creation; and 2) graph augmentation.

The goal of the first step is to discretize the entire space of values for a KG attribute into bins. Binning significantly decreases the number of distinct values that a model needs to learn, thus reducing the representational and computational complexity, and improving the model performance for relationships with sparse data. For instance, the entire set of birth date (P569) values in Wikidata could be split based on value frequency into four bins: [1 – 1882], [1882 – 1935], [1935 – 1966], and [1966 – 2021] (Figure 1, top).4 In total, KGA defines two different bin interval settings, based on either interval width (fixed) or value frequency (quantile), and three methods to set the bin levels: single, overlapping, and hierarchy. In this example, we use frequency to set the bin intervals and we use a single series of bins. While discretizing numeric values into bins is not new, KGA is the first method that uses binning of numeric values to improve LP performance.

In the augmentation step, each original value is assigned to its bin, and its binned value is used to augment the KG. For example, Barack Obama’s year of birth, 1961, belongs to the third bin ([1935 – 1966]), so the original triple (Q76, P569, “1961”) will be translated into (Q76, P569, [1935 – 1966]). Besides the assignment of the correct bin to its entity, KGA also allows for the neighboring bins to be chained to each other. The augmented KG, containing the original entity relationships and the added binned literal knowledge, can then be used to train any standard off-the-shelf embedding model.

We detail KGA’s two-step approach, and its use for LP.

3.1 Discretization and Bin Creation

We start by obtaining the set of values $v_i$ which are associated with entities $e \in \mathcal{E}$ of a given attribute $a$, namely $(e, a, v_i) \in G$. As dates come in different granularities (e.g., year, month, or day), we transform all dates with a granularity of year or finer (e.g., month) to year values. We sort these values in an ascending order, obtaining $V = \{v_1, v_2, ..., v_m\}$, such that $v_1 \leq v_{i+1}$. We next describe how we create bins based on the values in $V$, resulting in a dictionary $D : V \to \mathcal{E}$ per attribute, which maps each value in $V$ to one or multiple bin entities in $\mathcal{E}$. The discretization of KGA consists of two components: 1) definition of bin intervals; and 2) specification of bin levels.

1. Bin intervals specify the start and end values for the bins of a given attribute. Formally, given a minimal attribute value of $z^-$, a maximal value of $z^+$, and a set of $b$ bins, we seek to define $[z^-_i, z^+_i)$, for each bin $i$, where $i \in [0, b)$. Here, $b$ is a predefined parameter. We investigate two bin interval settings: fixed intervals and quantile-based intervals.5

Our fixed (equal width) binning strategy divides the entire space of values into bins of equal width. The length of each bin $i$ is $k = \frac{z^+ - z^-}{b}$, and the $i$-th bin contains the values $z_i = [z^- + ik, z^+ + (i + 1)k]$.

The quantile (equal frequency) binning strategy splits $[z^-, z^+]$ based on the distributional density, by ensuring that each bin $i$ covers the values for the same number of entities. This allows denser areas of the distribution to be represented with more bins. For a attribute $a$ defined for $N_a$ entities, each bin will cover approximately $n = \frac{N_a}{b}$ entities.

2. Bin levels specify whether the binning method will create a single set of bins, two overlapping sets of bins, or a hierarchy with several levels of bins.

The single strategy creates only one set of disjoint bins, as described previously.

---

4 All year values in Wikidata are positive. Years BC are also positive, but include an additional qualifier to indicate the era.

5 We also experimented with methods that group values based on their (cross-)group (dis)similarity like k-means clustering [Lloyd, 1982], but we do not report these given their low performance.
The overlapping strategy creates bins with intervals that overlap. Starting from a single auxiliary set of $2b$ bins, the overlapping bins are created by merging two neighboring bins at a time. The $i$-th merged bin consists of merging bin $b_i$ and $b_{i+1}$, the $(i+1)$-th merges $b_{i+1}$ and $b_{i+2}$, and so on. This process results in $2b - 1$ overlapping bins. Overlapping bins are illustrated in Figure 1, middle. Here, the auxiliary bins (not shown) would be $[1, 1826], [1826, 1882], [1882, 1912], \ldots$. The derived overlapping bins merge two neighboring bins at a time - merging the first two bins produces $[1, 1882]$, merging the second and the third produces $[1826, 1912]$, etc.

The hierarchy strategy creates multiple binning levels, signifying different granularity levels. A level $l$ has $b^l$ bins. The 0-th level is the coarsest, and it contains a single bin with the entire set of values $[z^-, z^+]$. The levels grow further in terms of detail: considering $b = 2$, the first level has 2 bins, the second one has 4, and the third one has 8. An example for a quantile-based hierarchy binning is shown in Figure 1 bottom. The entire set of birth years at level 0 ([$[1, 2021]$]) is split into two bins at level 1: $[1, 1935]$ and $[1935, 2021]$. Level 2 splits the bins at level 1 into even smaller, more precise bins: $[1, 1882], [1882, 1935], [1935, 1966]$ and $[1966, 2021]$. 

### 3.2 Graph Augmentation

The created bins provide a mapping from a value $v_i \in V$ to a set of bins $b_i \in B$. In order to augment the original graph $G$, we perform two operations: chaining of bins and linking bins to corresponding entities.

1. **Bin chaining** defines connections between two bins $b_i$ and $b_j$, where $b_i, b_j \in B$. The links between bins depend on the bin level setting. Both single bins and overlapping bins are chained by connecting each bin $b_i$ to its predecessor $b_{i-1}$ and successor $b_{i+1}$, see Figure 1 (top and middle). The bins in the hierarchy augmentation mode are connected both horizontally and vertically: 1) horizontally each bin $b_i$ connects to its predecessor $b_{i-1}$ and successor $b_{i+1}$; 2) vertically each bin is connected to its coarser level and finer level correspondents (see Figure 1 bottom).

2. **Bin assignment** links the generated bins $B(v)$ for a given attribute value $v$ to its corresponding entity $e$. A numeric triple $(e, a, v)$ is replaced with a set of triples $(e, a, b)$, where $b \in B(v)$. The updated set of triples modifies the original graph $G$ into an augmented graph $G'$. Figure 1 presents examples of bin assignment. The birth year (attribute P569) of Barack Obama (entity Q76) is 1961. For single level bins (Figure 1, top), the birth year of Obama is placed into bin $[1935, 1966]$. For overlapping bins (Figure 1, middle), the birth year is placed into the bins $[1935, 1966]$ and $[1951, 1982]$. And, when using hierarchy bin levels (Figure 1, bottom), the birth year is placed into bins $[1, 2021], [1935, 2021]$ and $[1935, 1966]$.

### 3.3 Link Prediction

$G'$ can now be used to perform entity and numeric LP. KGA is trained by simply replacing $G$ with $G'$ in the input to the base embedding model. Link prediction of entities with KGA is performed by selecting the entity node $e$ with the highest probability. Numeric link prediction is performed by selecting the bin $b$ with the highest score as a predicted range; and obtaining its median, $\bar{m}$, as an approximate predicted value.

### 4 Experimental Setup

#### 4.1 Datasets and Evaluation

We use benchmarks based on three KGs to evaluate LP performance. We show data statistics in Table 2.

1. **FB15K-237** [Toutanova and Chen, 2015] is a subset of Freebase [Bollacker et al., 2008], which mainly covers facts about movies, actors, awards, sports, and sports teams. FB15K-237 is derived from the benchmark FB15K by removing inverse relations, because a significant number of test triples in FB15K could be inferred by simply reversing the triples in the training set [Dettmers et al., 2018]. We use the same split as [Toutanova and Chen, 2015]. For numeric prediction we follow [Kotnis and García-Durán, 2019].

2. **YAGO15K** [Liu et al., 2019] is a subset of YAGO [Suchanek et al., 2007], which is also a general-domain KG. YAGO15K contains 40% of the data in the FB15K-237 dataset. However, YAGO15K contains more valid numeric triples than FB15K-237. For entity LP, we use the data split proposed in [Lacroix et al., 2020], while for numeric prediction we follow [Kotnis and García-Durán, 2019].

3. **DWD**. We introduce DARPA Wikidata (DWD), a large subset of Wikidata [Vrandečić and Krötzsch, 2014] that excludes prominent domain-specific Wikidata classes such as review article (Q7318358), scholarly Article (Q13442814), and chemical compound (Q11173). DWD describes over 37M items (42% of all items in Wikidata) with approximately 166M statements. DWD is several orders of magnitude larger than any of the previous LP benchmarks, thus providing a realistic evaluation dataset with sparsity and size akin to the size of modern hyperrelational KGs. We split the DWD benchmark at a 98-1-1 ratio, for both entity and numeric link prediction. Given the size of the DWD benchmark, 1% corresponds to a large number of data points (over 1M statements).

For entity LP on FB15K-237 and YAGO15K, we preserve a checkpoint of the model every 5 epochs for DistMult, ComplEx, every 10 epochs for ConvE, TuckER, and every 50 steps for TransE, RotatE. We use the MRR on the validation set to select the best model checkpoint. We report the filtered MRR, Hits@1, and Hits@10 of the best model checkpoint. For DWD, we preserve a checkpoint for each epoch. We use the MRR on the validation set to select the best model checkpoint, and report unfiltered MRR and Hits@10 [Lerer et al., 2019]. Filtered MRR evaluation requires discarding all positive edges when generating corrupted triples, which is not scalable for large KGs. For DWD, we report these metrics by ranking positive edges among $C = 500$ randomly sampled
corrupted edges. For numeric LP, we report MAE for each KG attribute.

### 4.2 Models

For entity link prediction, we evaluate KGA with the following six embedding models: TransE [Bordes et al., 2013], DistMult [Yang et al., 2014], ComplEx [Trouillon et al., 2016], ConvE [Dettmers et al., 2018], RotatE [Sun et al., 2019], and TuckER [Balázsévić et al., 2019]. We run KGA with \( b \in \{2, 4, 8, 16, 32\} \) and show the best result for each model. We compare KGA to the embedding-based LP predictions on the original graph, and to prior literal-aware methods: KBLN [García-Durán and Niepert, 2017], MTKGNN [Tay et al., 2017], and LiteralE [Kristiadi et al., 2019]. We compare our numeric LP performance to the methods NAP++ [Kotnis et al., 2017] and MrAP [Bayram et al., 2021]. As NAP++ is based on TransE, we focus on this embedding model. As mentioned above, we take the median of the most probable bin to be the predicted value by KGA.

The computationally demanding embedding models (ConvE, RotatE, and TuckER) cannot be run on DWD. The size of DWD is prohibitive for ConvE and TuckER because they depend on \( 1-N \) sampling, where batch training requires to load the entire entity embedding into memory.\(^6\) RotatE is even more memory-intensive, because of its hidden size of 2000, which requires an order of magnitude more memory than ConvE. As LiteralE and KBLN are based on these methods, they can also not run on DWD. Thus, on DWD, we compare KGA’s performance on the models ComplEx, TransE, and DistMult against their base model performance.

### 5 Results

#### 5.1 Entity Link Prediction Results

**Main results** LP results on FB15K-237 and YAGO15K are shown in Table 2. KGA outperforms the vanilla model and the literal-aware LP methods by a comfortable margin with all six embedding models. The best overall performance is obtained with the TuckER and RotatE models for FB15K-237, and the ConvE and RotatE models for YAGO15K, which is in line with the relative results of the original models. On FB15K237, the largest performance gain is obtained for DistMult (2.7 MRR points), and the performance gain is around 1 MRR point for the remaining models. The comparatively smaller improvement for TuckER on FB15K-237, brought by KGA and other literal-aware methods, could be due to TuckER’s model parameters being particularly tuned for this dataset. This intuition is supported by the much higher contribution of KGA for TuckER on the YAGO15K dataset. KGA brings a higher MRR gain on the YAGO15K dataset, which shows that KGA is able to learn valuable information from the additional 24% literal triples which we added to YAGO15K. On YAGO15K, KGA improves the performance of the most recent models (ConvE, RotatE, and TuckER) by over 2 MRR points. These results highlight the impact of

\[^6\text{With a hidden size of 200, loading the entity embedding in memory requires at least 42.5M * 200 * 4 = 34 GB GPU memory, which is challenging for most GPU devices today.}\]

<table>
<thead>
<tr>
<th>Method</th>
<th>FB15K-237</th>
<th>YAGO15K</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>H@1</td>
</tr>
<tr>
<td>TransE</td>
<td>0.315</td>
<td>0.217</td>
</tr>
<tr>
<td>+LiteralE</td>
<td>0.315</td>
<td>0.218</td>
</tr>
<tr>
<td>+KBLN</td>
<td>0.308</td>
<td>0.210</td>
</tr>
<tr>
<td>+KGA</td>
<td>0.321</td>
<td>0.223</td>
</tr>
</tbody>
</table>

Table 2: LP results on FB15K-237 and YAGO15K. We compare KGA to the original model (-), and the baselines LiteralE and KBLN. We report the reproduced results for all baseline methods, and provide the original results in the appendix. For KGA, we show the best results across discretization strategies (single, overlapping, hierarchy) and numbers of bins (2, 4, 8, 16, 32). We bold the best overall result per metric, and underline the best result per model.

<table>
<thead>
<tr>
<th>Method</th>
<th>TransE</th>
<th>DistMult</th>
<th>ComplEx</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MRR</td>
<td>H@10</td>
<td>MRR</td>
</tr>
<tr>
<td>-</td>
<td>0.580</td>
<td>0.762</td>
<td>0.559</td>
</tr>
<tr>
<td>Quantity</td>
<td>0.582</td>
<td>0.764</td>
<td>0.564</td>
</tr>
<tr>
<td>Year</td>
<td>0.580</td>
<td>0.763</td>
<td>0.562</td>
</tr>
<tr>
<td>KGA</td>
<td>0.583</td>
<td>0.764</td>
<td>0.566</td>
</tr>
</tbody>
</table>

Table 3: LP results on DWD. We show the performance (MRR and Hits@10) of the vanilla embedding model (-), and KGA with binned quantities, with years, and the full KGA. We use 32-bin KGA with QOC (quantile, overlapping, and chaining) discretization.

KGA’s approach of augmenting KG embedding models with discretized literal values.

**Scalability and knowledge ablations** We test the ability of KGA to perform LP on DWD. The results (Table 3) reveal that KGA, unlike prior literal-aware methods, scales well to much larger LP benchmarks. Furthermore, we observe that KGA brings steady improvement across the models and the metrics. Our knowledge ablation study shows that integrating either quantities or years is better than the vanilla model, that quantities are marginally more informative than years, and that integrating both of them yields best performance. We conclude that quantities and years are informative and mutually complementary for LP by embedding models.

However, the improvement brought by KGA for the models TransE, DistMult, and ComplEx is between 0.3 and 0.7 MRR points, which is much lower than its impact on the
smaller datasets. We note that the results in Table 2 show
the best configuration for KGA, while the results in Table 3
show a single configuration (QOC with 32 bins). Therefore,
we hypothesize that the performance of KGA on DWD can
be improved with further tuning of the number of bins and
the discretization strategies applied. For computational
reasons, we investigate the impact bin sizes and discretization
strategies on the FB15K dataset, and leave the analogous in-
vestigation for DWD to future work.

### Binning ablations

We study different variants of discretiza-
tion (Fixed and Quantile-based), bin levels (Single, Overlap-
ping, and Hierarchy), and bin sizes (2, 4, 8, 16, and 32) on
the FB15K-237 benchmark. The results (Table 4) show that
all of the discretization variants of KGA consistently improve
over the baseline model. We observe that quantile-based binn-
ing is superior over fixed width binning, which is intuitive
because quantile binning considers the density of the value
distribution. Among the bin levels, we see that hierarchy bin
levels performs better than overlapping binning, which in turn
performs better than using a single bin. This relative order of
performance correlates with the expressivity of each bin
levels strategy. Comparing the quantile-overlapping versions
with and without chaining, we typically observe a small ben-
efit of chaining the bins horizontally. Table 5 provides results
of KGA with different bin sizes (2, 4, 8, 16, 32) for all six
models. We observe that finer-grained binning is generally
preferred, as 32 bins performs best for 4 of the 6 models.
Yet, we observe that for RotatE, the best performance is ob-
tained with 16 bins, whereas for TransE, the number of bins
has no measurable impact. We conclude that the performance
of KGA depends on selecting the best discretization strategy
and bin size. While more expressive discretization and fine-

### 5.2 Numeric Link Prediction Results

#### Main results

Next, we investigate the ability of KGA to pre-
dict quantity and year values directly. We compare KGA-
QOC with 32 bins to the baselines MrAP and NAP++ on
the FB15K-237 and YAGO15K datasets. Numbers indicate MAE. Values of NAP++ and
MrAP are taken from [Bayram et al., 2021]. We show results for KGA with TransE for a fair comparison to NAP++.

<table>
<thead>
<tr>
<th>model</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>0.321</td>
<td>0.320</td>
<td>0.321</td>
<td>0.321</td>
<td>0.321</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.306</td>
<td>0.308</td>
<td>0.314</td>
<td>0.317</td>
<td>0.322</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.294</td>
<td>0.295</td>
<td>0.300</td>
<td>0.304</td>
<td>0.305</td>
</tr>
<tr>
<td>ConvE</td>
<td>0.321</td>
<td>0.320</td>
<td>0.325</td>
<td>0.325</td>
<td>0.329</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.327</td>
<td>0.326</td>
<td>0.332</td>
<td>0.335</td>
<td>0.334</td>
</tr>
<tr>
<td>TuckER</td>
<td>0.354</td>
<td>0.356</td>
<td>0.355</td>
<td>0.356</td>
<td>0.357</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>model</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>0.321</td>
<td>0.320</td>
<td>0.321</td>
<td>0.321</td>
<td>0.321</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.306</td>
<td>0.308</td>
<td>0.314</td>
<td>0.317</td>
<td>0.322</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.294</td>
<td>0.295</td>
<td>0.300</td>
<td>0.304</td>
<td>0.305</td>
</tr>
<tr>
<td>ConvE</td>
<td>0.321</td>
<td>0.320</td>
<td>0.325</td>
<td>0.325</td>
<td>0.329</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.327</td>
<td>0.326</td>
<td>0.332</td>
<td>0.335</td>
<td>0.334</td>
</tr>
<tr>
<td>TuckER</td>
<td>0.354</td>
<td>0.356</td>
<td>0.355</td>
<td>0.356</td>
<td>0.357</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>model</th>
<th>2</th>
<th>4</th>
<th>8</th>
<th>16</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>TransE</td>
<td>0.321</td>
<td>0.320</td>
<td>0.321</td>
<td>0.321</td>
<td>0.321</td>
</tr>
<tr>
<td>DistMult</td>
<td>0.306</td>
<td>0.308</td>
<td>0.314</td>
<td>0.317</td>
<td>0.322</td>
</tr>
<tr>
<td>ComplEx</td>
<td>0.294</td>
<td>0.295</td>
<td>0.300</td>
<td>0.304</td>
<td>0.305</td>
</tr>
<tr>
<td>ConvE</td>
<td>0.321</td>
<td>0.320</td>
<td>0.325</td>
<td>0.325</td>
<td>0.329</td>
</tr>
<tr>
<td>RotatE</td>
<td>0.327</td>
<td>0.326</td>
<td>0.332</td>
<td>0.335</td>
<td>0.334</td>
</tr>
<tr>
<td>TuckER</td>
<td>0.354</td>
<td>0.356</td>
<td>0.355</td>
<td>0.356</td>
<td>0.357</td>
</tr>
</tbody>
</table>

Table 4: Ablation study on modes of graph augmentation with link prediction on FB15K-237. KGA variants: '-' represents the original graph
(no augmentation), F = Fixed Size, Q = Quantile, S = Single, O = Overlapping, H = Hierarchy, C = Chaining, N = No Chaining. The best
result for each column is marked in bold. We show the best results among the different numbers of bins (2, 4, 16, 32).

Table 5: Effect of bin size on the performance of different models on
FB15K-237. We show results for the best discretization strategy. We experiment with 2, 4, 8, 16, and 32 bins. Numbers indicate MRR.

Table 6: Performance of our numeric predictor with different
choices of base model on graph augmented with 32-bin QOC,
when compared to existing SOTA methods on the FB15K-237 and
YAGO15K dataset. Numbers indicate MAE. Values of NAP++ and
MrAP are taken from [Bayram et al., 2021]. We show results for KGA with TransE for a fair comparison to NAP++.
### Table 7: Performance of our numeric predictor KGA-QOC on DWD compared to a linear regression (LR) model and a median baseline. We use 32 bins for both quantities and years. Numbers indicate MAE reduction percentages against a median value baseline. We report results for the five most populous 5 properties for both quantities and years, with identifiers: P1087, P6258|Q28390, P2044|Q11573, P6257|Q28390, P1215, P569, P570, P577, P571, and P585.

<table>
<thead>
<tr>
<th>attribute</th>
<th>Median</th>
<th>LR</th>
<th>KGA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elo rating</td>
<td>119.03</td>
<td>86.09</td>
<td>55.20</td>
</tr>
<tr>
<td>declination (degree)</td>
<td>18.68</td>
<td>9.83</td>
<td>18.53</td>
</tr>
<tr>
<td>elevation above sea level</td>
<td>466.51</td>
<td>366.64</td>
<td>459.48</td>
</tr>
<tr>
<td>right ascension (degree)</td>
<td>82.98</td>
<td>40.90</td>
<td>82.51</td>
</tr>
<tr>
<td>apparent magnitude</td>
<td>3.02</td>
<td>2.00</td>
<td>2.37</td>
</tr>
<tr>
<td>date of birth</td>
<td>62.71</td>
<td>49.70</td>
<td>58.59</td>
</tr>
<tr>
<td>date of death</td>
<td>90.68</td>
<td>78.10</td>
<td>79.38</td>
</tr>
<tr>
<td>publication date</td>
<td>28.33</td>
<td>17.37</td>
<td>28.27</td>
</tr>
<tr>
<td>inception</td>
<td>72.84</td>
<td>61.45</td>
<td>72.27</td>
</tr>
<tr>
<td>point in time</td>
<td>88.76</td>
<td>81.65</td>
<td>83.70</td>
</tr>
</tbody>
</table>

6 Related Work

Graph Embedding with Literals

Literal-aware LP methods [Xiao et al., 2017] predominantly focus on strings, by learning a representation of the textual descriptions of an entity and combining it with its structured representation [Gesese et al., 2019]. Considering string-valued triples is a natural future extension of KGA. Several efforts incorporate numeric triples into KG embeddings by adding a literal-aware term to the scoring function of the embedding model. LiteralE [Kristiadi et al., 2019] incorporates literals by passing literal-enriched embeddings to the scoring function. Assuming that the difference between the numeric values for a relation is a good indicator of the existence of a relation, KBLN [García-Durán and Niepert, 2017] adds a separate scoring function for literals. Our experiments show that KGA performs better than both LiteralE and KBLN on entity LP. Instead of modifying the scoring function, several methods modify the loss function of the base model to balance between predicting numeric and entity values. TransEA [Wu and Wang, 2018] extends TransE with a regression penalty on the base model, while MTKGNN [Tay et al., 2017] uses multitask learning and extends a neural representation learning baseline by introducing separate training steps that use embedding to predict numeric values. MARINE [Feng et al., 2019] extends these methods by adding a proximity loss, which preserves the embedding similarity based on shared neighbors between two nodes. We do not compare against TransEA and MARINE because their reported performance is lower than recent base models or KBLN, whereas comparison to MTKGNN would require re-implementation of the model, as the original work is evaluated on different datasets and has its own code base. In contrast to prior work, KGA augments the structure of the original KG, leaving the loss function of the base model intact. As a consequence, KGA can be directly reused to new embedding methods without customizing the base algorithm or the scoring function, and it can be computed on large KGs with the size of Wikidata. Furthermore, the explicitly represented literal range values are intrinsically meaningful as intuitive approximation, corresponding to how humans perceive numbers [Dehaene, 2011].

7 Conclusions

This paper proposed a knowledge graph augmentation (KGA) method, which incorporates literals into embedding-based link prediction systems in a pre-processing step. KGA does not modify the scoring or the loss function of the model, instead, it enhances the original KG with discretized quantities and years. Thus, KGA is designed to generalize to any embedding model and KG. We formulated variants of KGA that differ in terms of their interval definition, binning levels, number of bins, and link formalization. Evaluation showed the superiority of KGA over vanilla embedding models and baseline methods on both entity and numeric link prediction. Unlike prior baselines, KGA scaled to a Wikidata-sized KG, as it performs a minor adaptation of the original model. The performance of KGA depends on the selected number of bins and discretization strategy. While more expressive discretization and binning usually fares better, optimal performance is model-dependent and should be investigated further. Future work should also extend KGA to include string literals, and to enhance link prediction on other graphs, like DBpedia.

References


