A Closer Look at Sum-based Embedding Aggregation for Knowledge Graphs Containing Procedural Knowledge

Richard Nordsieck[1,*], Michael Heider[2], Anton Hummel[1] and Jörg Hähner[2]

[1]*XITASO GmbH IT & Software Solutions, Austraße 35, 86153 Augsburg, Germany*
[2]*Organic Computing Group, University of Augsburg, Am Technologiezentrum 8, 86159 Augsburg, Germany*

Abstract
While knowledge graphs and their embedding into low dimensional vectors are established fields of research, they mostly cover factual knowledge. However, to improve downstream models, e. g. for predictive quality in real-world industrial use cases, embeddings of procedural knowledge, available in the form of rules, could be utilized. As such, we investigate which properties of embedding algorithms could prove beneficial in this scenario and evaluate which established embedding methodologies are suited to form the basis of sum-based embeddings of different representations of procedural knowledge.

Keywords
knowledge graph embedding, industrial knowledge graph, expert knowledge

1. Introduction

Knowledge graphs are frequently used to represent a multitude of heterogeneous information from different sources [1]. Since they are often non-exhaustive, knowledge graph completion, e. g. through link prediction, is a widely researched field which usually relies on knowledge graph embeddings, i. e. low dimensional vector representations [2]. A separate strand of research utilizes knowledge graph embeddings to infuse background knowledge into downstream models such as information retrieval, recommender systems or predicting variables [3]. However, most of these use cases deal with factual (i. e. terminology, specific details or elements [4]) or conceptual (i. e. classifications, categories, principles and generalisations [4]) knowledge, such as *located in* or *president of* relations.

In contrast to this, procedural (i. e. knowledge of skills, techniques, methods and 'criteria for determining when to use appropriate procedures' [4]) and metacognitive (i. e. strategic, contextual and conditional) knowledge plays a significant role in industries, such as manufacturing, where parameters have to be adjusted to achieve target quality criteria or mitigate occurring quality defects. Training learning systems for these predictive quality use cases is hampered by a scarcity in observable data since parametrisation processes are executed relatively seldom

ISWC 2022: Deep Learning for Knowledge Graphs (DL4KG), October 24–28, virtual
*Corresponding author.
richard.nordsieck@xitaso.com (R. Nordsieck); michael.heider@informatik.uni-augsburg.de (M. Heider); anton.hummel@xitaso.com (A. Hummel); joerg.haehner@informatik.uni-augsburg.de (J. Hähner)
0000-0003-3140-1993 (M. Heider); 0000-0003-0107-264X (J. Hähner)

which leads to few data points that contain relevant information compared to overall process iterations since the processes have been manually optimised to a high degree. Still, utilising learning systems as assistance for relatively untrained labour could be one factor to adapt to the rapidly changing workplace and demographic changes. To mitigate the previously mentioned challenges, we envision a combination of procedural knowledge with learning systems. Building on Nordsieck et al. [5], that have explored how procedural knowledge can be represented in knowledge graphs and provided a proof of concept embedding methodology, we now investigate which combination of representations and embedding methods are best suited for embedding knowledge graphs containing procedural knowledge. To this end, we analyse the following research questions from a theoretical as well as an empirical perspective:

- RQ 1: What is the impact of representing quantified values as literals or entities?
- RQ 2: What is the impact of chained binary relations on the benefit of quantified values?
- RQ 3: Which embedding methodologies are able to deal with the indirections introduced through more detailed representations?
- RQ 4: Which embedding method is best suited to embed procedural knowledge?

2. Related Work

Industrial knowledge graphs are gaining traction in practice but are frequently dealing only with factual knowledge [1]. Nordsieck et al. [5] address this problem, providing modelling patterns for procedural knowledge. However, the investigated modelling pattern is not applicable to knowledge graphs with multiple kinds of relations, which are common in practice. We address this shortcoming by providing alternative and more detailed modelling patterns.

Knowledge graph embedding methods for link prediction have been extensively researched [6, 7, 8]. Each of these approaches exhibits strengths and weaknesses. Gutiérrez-Basulto and Schockaert [9] mention that previous approaches are not suited to embed rules, which are an integral part of representing procedural knowledge. Furthermore, they provide theoretical considerations on how this could be alleviated. In contrast to this, Abboud et al. [2] present an implemented approach that is able to represent existential rules. Of special interest to embed quantifiable procedural knowledge are methods that explicitly cater for literals [10]. However, whether direct transfer of the results achieved using these methods to graphs containing procedural knowledge is possible needs to be validated. Portisch and Paulheim [11] relate link prediction embedding methods to those encountered in data mining, e. g. RDF2Vec, and conclude that link prediction methods could also be used for downstream tasks.

3. Embedding Procedural Knowledge Graphs

In the following, we extend and evaluate an approach, presented by Nordsieck et al. [5], to embed procedural knowledge to allow for its use in downstream predictive scenarios. The approach is shown in Figure 1. To be able to use knowledge graph embedding methods a suitable modelling pattern for a graph representation that determines how to represent the different elements of procedural knowledge in a graph has to be chosen (see Section 3.1). Based on this

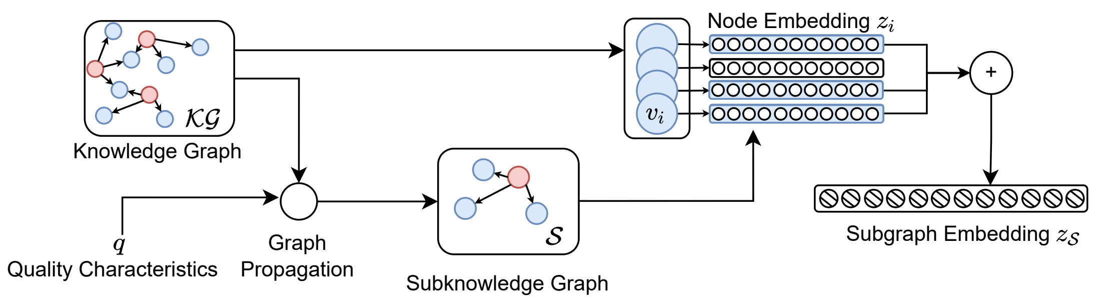
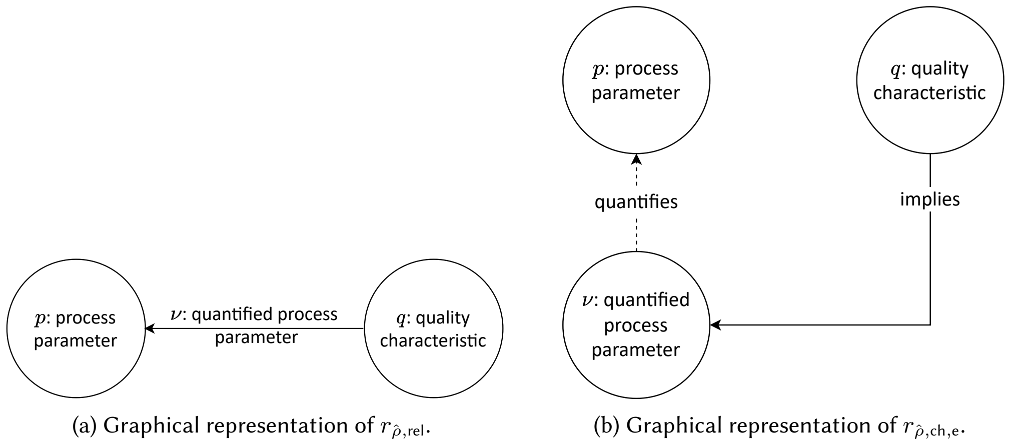


Figure 1: *Illustration of the embedding methodology.* Node embeddings are calculated for the knowledge graph. The quality characteristic q determines the starting node to propagate from. Node embeddings of nodes present in the resulting subgraph are aggregated to one subgraph embedding $z_{\mathcal{S}}$.


p: process parameter
ν: quantified process parameter
q: quality characteristic
p: process parameter
q: quality characteristic
quantifies
implies
ν: quantified process parameter

(a) Graphical representation of $r_{\hat{\rho},\text{rel}}$.

(b) Graphical representation of $r_{\hat{\rho},\text{ch,e}}$.

Figure 2: Two exemplary graphical representations of modelling patterns for ternary rules with quantified conclusions: $r_{\hat{\rho},\text{rel}}$ modelling the quantification as individual relations and $r_{\hat{\rho},\text{ch,e}}$, which uses chained binary relations with the conclusion quantification treated as an entity.

representation the procedural knowledge, which is usually available in the form of *if-then* rules [12], is transferred into a knowledge graph. For this graph, node embeddings can be calculated using standard knowledge graph embedding methods, e. g. TransE, BoxE or RDF2Vec. In our perspective downstream task of predicting parameter adjustments it is necessary to select a portion of the graph that is relevant for the specific input to the system by propagating from the node representing the input. The resulting subgraph is then embedded using a sum-based approach utilising the previously computed node embeddings.

3.1. Representation

Procedural knowledge can be available in different levels of detail, ranging from high-level *implies* notions between conditions or conclusions of a rule to quantified versions of it, i. e. *implies* relations between quality defects and the responsible process parameters, to relations between quantified parameters and the inclusion of production data underlying the quantifications in a manufacturing scenario [5]. These different levels of detail have increasing demands on the embedding methods due to their properties, e. g. arity of relations or literals. In the following, we explore modelling alternatives for graph representations of different levels of detail of procedural knowledge. As more fine-granular levels of detail do not introduce new properties but merely increase their relevance, we focus on the following levels of details: unquantified procedural knowledge and quantified conclusions. Unquantified knowledge describes high-level *implies* relations or unquantified rules, e. g. *If quality characteristic q is unsatisfactory then adjust process parameter p*, and can be directly represented as a triple $r_\eta = (q, \langle\text{implies}\rangle, p)$ for $p \in P$ and $q \in Q$ [5].

In contrast to this, procedural knowledge with quantified conclusions, e. g. *If quality characteristic q is unsatisfactory then adjust process parameter p by* ν with $\nu \in \mathbb{R}$, results in a quadruple $r_{\hat{\rho}} = (q, \langle\text{implies}\rangle, p, \nu)$ [5]. This means that the relation $r_{\hat{\rho}}$ underlying the quantified conclusions is ternary as opposed to the binary relations that are usually represented in knowledge graphs and subsequently knowledge graph embedding methods apart from BoxE [2]. Nordsieck et al. [5] dealt with this by treating the quantification ν, a numeric literal, as a weight to the *implies* relation, a representation we will refer to as $r_{\hat{\rho},\text{rel}}$. However, since common knowledge graph embedding methods do not support literals of relations, the quantifications were modelled as separate relations. This incurs a loss of semantics which is especially disadvantageous if knowledge from multiple sources or of multiple levels of detail is represented in the graph, as is usually the case in practice.

To address this issue, we rely on the fact that the ternary relation can be written as a chain of two binary relations, i. e. $r_{\hat{\rho},\text{ch}} = (q, \langle\text{implies}\rangle, (\nu, \langle\text{quantifies}\rangle, p))$, which can then be directly represented in the graph as can be seen in Figure 2b. However, this indirection could be challenging for embedding methods to pick up.

At the quantification level, we're confronted with two modelling alternatives: treating quantifications as entities, $r_{\hat{\rho},\text{ch,e}}$, which diminishes the semantic value of the respective nodes or explicitly modelling them as literals, $r_{\hat{\rho},\text{ch,l}}$, which keeps their semantics largely intact but provides further challenges for established knowledge graph embedding methods, that lack explicit support for literals [10]. The second alternative brings with it the need to keep the high-level *implies* between p and q entities as in the unquantified case since otherwise there is no direct connection between them in the graph. We denote this literal-based representation as $r_{\hat{\rho},\text{ch,l},\eta}$. This modelling decision leads to this representation exhibiting a compositional property since $(p, \nu) \in \langle\text{implies}\rangle \wedge (\nu, q) \in \langle\text{quantifies}\rangle \implies (p, q) \in \langle\text{implies}\rangle$. For consistency, we introduce a modelling alternative with the same characteristic to the entity based modelling alternative, resulting in $r_{\hat{\rho},\text{ch,e},\eta}$. Figure 2 allows a graphical comparison of representatives of the previously described modelling alternatives.

An overview of properties exhibited by at least one of the representations can be seen in Table 1. While the properties composition, indirection and literals have been previously described, asymmetry directly results from the modelling decisions made, i. e. if $(p, q) \in \langle\text{implies}\rangle \implies (q, p) \notin \langle\text{implies}\rangle$. Also, all representations exhibit heterogeneous entities since process parameters, quality characteristics and quantifications belong to different classes.

Table 1
Overview of properties which representations exhibit and embedding methods address on a theoretical level. For the pattern underlying representation names refer to Section 3.1.

		Asymmetry	Composition	Literals	Indirection
Repres.	r_η	✓	✗	✗	✗
	$r_{\hat{\rho},\text{rel}}$	✓	✗	✗	✗
	$r_{\hat{\rho},\text{ch,e}}$	✓	✗	✗	✓
	$r_{\hat{\rho},\text{ch,e},\eta}$	✓	✓	✗	✓
	$r_{\hat{\rho},\text{ch,l},\eta}$	✓	✓	✓	✓
Embedding	TransE	✓	✓	✗	✗
	ComplEx	✓	✗	✗	✗
	ComplEx-LiteralE	✓	✗	✓	✗
	DistMult	✗	✗	✗	✗
	DistMult-LiteralE$_g$	✗	✗	✓	✗
	RotatE	✓	✓	✗	✗
	BoxE	✓	✗	✗	✗
	RDF2Vec	✓	✓	✗	✓

3.2. Individual Embeddings

Individual node embeddings are calculated by established knowledge graph embedding methods. Since Portisch et al. [3] concluded that embeddings for link prediction are, to a certain extent, also suited for downstream tasks, we chose both, methods initially intended for embeddings for link prediction as well as downstream tasks in our investigation to determine which is suited best for embedding procedural knowledge. We decided on a set of knowledge graph embedding methods to investigate, namely TransE, ComplEx, ComplEx-LiteralE, DistMult-LiteralE$_g$, DistMult, RotatE, BoxE and RDF2Vec, which includes embedding methods of different complexity that satisfy one or more of the properties exhibited by the representations (cf. Section 3.1). Table 1 shows an overview of which properties are addressed by the respective embedding method.

RDF2Vec seems to be the best suited method from a theoretical perspective. Due to its execution of random walks on the graph, it is likely able to capture relational properties [13]. Also, the indirections introduced by modelling the quantification as a chained binary relations, a property with which other embedding methods struggle, can be embedded if the walks are executed to a sufficient depth. In addition, link prediction methods outperform RDF2Vec on well curated datasets [3]. Since industrial datasets are notoriously skewed, biased and incomplete this could be a further benefit on real world data.

Several works mention that the embedding methods' ability to capture composition in the graph could have a detrimental effect on the embedding quality [14, 2]. To investigate this in our scenario, we selected a number of embedding methods that encode composition and a number which do not.

Also, while there exists a broad spectrum of embedding methods that are able to embed entities, there are only relatively few which explicitly take literals into consideration [10]. As such, we added LiteralE [15] to our evaluation that builds on ComplEx and gated DistMult for which [10] report good results on numeric literals. The other properties are shared with the respective underlying embedding method.

As we can see, there is not a single embedding method that is a perfect fit for the properties exhibited by the representations. Therefore, we rely on the experimental evaluation in Section 4 for indications for the importance of different properties.

3.3. Sum-based Embedding Aggregation

To allow for the computation of sum-based embeddings the individual embeddings have to be identified. Section 3.3.1 presents the employed methods for selecting the relevant subgraphs while Section 3.3.2 details the aggregation methodology.

3.3.1. Node Selection Strategy

To select the most relevant nodes that can be aggregated for a given input, i. e. a quality characteristic in our case, a subgraph $\mathcal{S}_q$ is generated by propagating from the node corresponding to the input. The depth until which the propagation is executed is limited by n, which is chosen depending on the respective representation. In our case $n = 1$ for unquantified conclusions, r_η, and quantified conclusions modelled through relations, $r_{\hat{\rho},\text{rel}}$ and $n = 2$ for entity-based representations that model the ternary relation through a combination of two relations. For the literal-based representation, $r_{\hat{\rho},\text{ch,l},\eta}$, $n = 1$ since no node embedding for the literal is computed—it only influences the embedding of the nodes connected to it.

3.3.2. Aggregation

Individual node embeddings z of nodes v, with $v \in \mathcal{S}_n$, where $\mathcal{S}_n$ is the subgraph corresponding to the input q, form the basis of the subgraph's embedding and are aggregated following a sum-based approach. Nordsieck et al. [5] argue that since the head node does not provide additional semantic information it should not be considered. This results in

$$z_{\mathcal{S}}^{\bar{h}} = \sum_{(v_i,v_j)\in\mathcal{S}} z_j \otimes D(z_i, z_j),$$

where v_i, v_j are the pairs of head and tail nodes resulting from the graph propagation and $D(z_i, z_j)$ is a distance measure between the node embeddings of v_i and v_j. However, one could also include the head node in the subgraph embedding. Therefore, we define an aggregation variant including the head node, z_0, via:

$$z_{\mathcal{S}}^{h} = z_0 + \sum_{(v_i,v_j)\in\mathcal{S}} z_j \otimes D(z_i, z_j).$$

Due to the pattern-based nature of knowledge graphs in our scenario, we expect the relations to be the same independent of the starting node. Also, they have previously been considered by the knowledge graph embedding methods. Therefore, we do not consider the relations' embedding in the aggregation.

Since it is not clear whether the head node contains semantic information and should consequently be included in the computation of the subgraph embedding $z_{\mathcal{S}}$, we treat it as a parameter along with the distance measure and conduct an experiment in Section 4 to evaluate its effect.

4. Evaluation

To evaluate which method performs best for embedding procedural knowledge for downstream tasks, we divert from metrics frequently encountered in link predictions scenarios (i. e. *hits@k*, that measures the fraction of hits for which an entity appears under the first k entries of the sorted list of individual rank scores) and utilize the metric *matches@k*, which measures the mean overlap between the k closest quality characteristics in embedding and graph space [5]. As such, it is suited to establish whether subgraphs are close in embedding and graph space. Closest is in this case defined as the highest overlap in related process parameters. A high value—up to a maximum of k—indicates that the k nearest quality characteristics of the graph space have been correctly identified in the embedding space.

To evaluate the representations we rely on a synthetic dataset that simulates parametrisation processes in manufacturing scenarios[1], i. e. process iterations with parameters and the resulting quality characteristics that can be used as target variables in predictive quality systems. This allows us to control the effect of noise that is typical for many real world datasets and might lead to misjudging methods due to the influence of said noise. Since in the envisioned use-case, generalisation of the learnt embeddings to new knowledge is not needed we do not need to employ a separate test set and can validate convergence in-sample. The dataset results in a knowledge graph containing 42 to 90 vertices with 48 to 144 edges depending on the representation. The average neighbour degrees range between 1.67 and 4.11.

The embeddings have been trained using PyKEEN[2] for link prediction node embeddings with an Adam optimizer with learning rate 4×10^{-4} and weight decay 1×10^{-5} as well as the default parametrization of the embedding methods, while RDF2Vec was trained using pyRDF2Vec[3], and a random walker with maximal depth and maximal walks set to 4 and 100, respectively. All embedding methods apart from BoxE and RDF2Vec were trained for 750 epochs. BoxE and RDF2Vec were trained for 1500 and 1000 epochs, respectively, as they did not yet converge after 750. We chose to evaluate 48 dimensional embeddings since these allow for direct utilisation in a down-stream predictive system.

Inspecting the results shown in Table 2, we can observe that the theoretically best-suited method, RDF2Vec, produces uncompetitive results in practice—the exception being representation $r_{\hat{\rho},\text{rel}}$. Interestingly, it is also unable to cope with modelling ternary relations through chained binary relations (present from $r_{\hat{\rho},\text{ch,e}}$ onward), which we expected to be the case as the random walks conducted can span more than the required two relations. Overall, BoxE produced the worst results, performing slightly worse than RDF2Vec for most configurations. Regarding the variations in the sum-based aggregation, euclidean distance outperforms jaccard for all representations and embedding methods apart from RDF2Vec. Whether to include the

[1]Code, data and RDF representations are available at https://github.com/0x14d/embedding-operator-knowledge
[2]https://pykeen.readthedocs.io/
[3]https://pyRDF2Vec.readthedocs.io/

Table 2
Results of various embedding methods and aggregation configurations for different representations as measured with *matches@3* over 30 runs (displaying mean and standard deviation). ComplEx and ComplEx-LiteralE are omitted due to space constraints and unremarkable results. Best results per representation in bold. Results for $r_{\hat{\rho},\text{ch},\text{l},\eta}$ are only available for embedding methods supporting literals.

Rep.	Head	Dist.	TransE	RotatE	DistMult	DM-LiteralE$_g$	BoxE	RDF2Vec
r_η	h	euc	2.08 ± 0.18	2.24 ± 0.12	2.05 ± 0.15	1.91 ± 0.21	1.65 ± 0.12	1.59 ± 0.16
		jac	1.93 ± 0.22	2.14 ± 0.17	1.98 ± 0.19	1.83 ± 0.2	1.68 ± 0.09	1.61 ± 0.16
	$\bar{h}$	euc	2.17 ± 0.17	**2.25 ± 0.1**	2.04 ± 0.13	1.98 ± 0.23	1.61 ± 0.13	1.85 ± 0.15
		jac	1.91 ± 0.22	2.16 ± 0.16	1.97 ± 0.17	1.88 ± 0.23	1.63 ± 0.11	1.88 ± 0.1
$r_{\hat{\rho},\text{rel}}$	h	euc	2.07 ± 0.14	2.1 ± 0.16	1.92 ± 0.18	1.87 ± 0.22	1.68 ± 0.14	1.81 ± 0.19
		jac	1.86 ± 0.23	2.01 ± 0.22	1.89 ± 0.21	1.81 ± 0.2	1.7 ± 0.12	1.82 ± 0.19
	$\bar{h}$	euc	**2.14 ± 0.16**	2.09 ± 0.16	1.97 ± 0.18	1.95 ± 0.2	1.59 ± 0.14	2.06 ± 0.16
		jac	1.9 ± 0.21	2.07 ± 0.2	1.89 ± 0.25	1.86 ± 0.24	1.61 ± 0.17	2.07 ± 0.16
$r_{\hat{\rho},\text{ch},\text{e}}$	h	euc	1.78 ± 0.15	**1.96 ± 0.16**	1.88 ± 0.17	1.77 ± 0.16	1.34 ± 0.22	1.68 ± 0.07
		jac	1.74 ± 0.2	1.87 ± 0.18	1.84 ± 0.24	1.74 ± 0.22	1.42 ± 0.15	1.72 ± 0.14
	$\bar{h}$	euc	1.83 ± 0.2	1.93 ± 0.18	1.92 ± 0.17	1.81 ± 0.18	1.27 ± 0.21	1.62 ± 0.1
		jac	1.77 ± 0.24	1.88 ± 0.19	1.85 ± 0.23	1.74 ± 0.22	1.28 ± 0.2	1.57 ± 0.15
$r_{\hat{\rho},\text{ch},\text{e},\eta}$	h	euc	1.98 ± 0.16	**2.22 ± 0.12**	2.08 ± 0.16	2.01 ± 0.17	1.53 ± 0.23	1.82 ± 0.12
		jac	1.86 ± 0.2	2.08 ± 0.17	1.98 ± 0.18	1.92 ± 0.21	1.56 ± 0.2	1.77 ± 0.16
	$\bar{h}$	euc	2.0 ± 0.18	2.2 ± 0.12	2.05 ± 0.16	1.99 ± 0.17	1.5 ± 0.22	1.74 ± 0.11
		jac	1.87 ± 0.2	2.08 ± 0.16	1.96 ± 0.19	1.91 ± 0.23	1.48 ± 0.21	1.62 ± 0.17
$r_{\hat{\rho},\text{ch},\text{l},\eta}$	h	euc				**1.93 ± 0.23**		
		jac				1.89 ± 0.22		
	$\bar{h}$	euc				2.0 ± 0.21		
		jac				**1.93 ± 0.23**		

head node in the aggregation, however, remains inconclusive. For representations without indirections or chained binary relations (r_η and $r_{\hat{\rho},\text{rel}}$) no difference is discernible. For representations with chained binary relations, the head node is included in a majority of the best results per representation. This indicates that the indirections introduced by chained binary relations profit of a stronger sense of context which is provided by adding the head node. This can be interpreted as an indication that the quantification and its connections to the respective parameter can be embedded. Regarding RQ4, we conclude that a combination of RotatE with euclidean distance and included head node is likely to be the embedding method best suited to embed procedural knowledge.

To answer RQ1, whether quantified values should better be represented as literals or entities, we refer to RotatE's performance on $r_{\hat{\rho},\text{ch},\text{e},\eta}$ versus DistMult-LiteralE$_g$ on $r_{\hat{\rho},\text{ch},\text{l},\eta}$. Here, RotatE outperforms DistMult-LiteralE$_g$ by 15.03% for the respective best configuration. In general, we observe that the literal enabled method DistMult-LiteralE$_g$ performs consistently worse than its literal agnostic base DistMult with Complex-LiteralE showing the same behaviour. Therefore, we conclude that representing quantified values as entities is the better choice for the investigated methods.

Regarding RQ2, no benefit of including quantified values in the respective representations can be discerned using the *matches@k* metric. As such, RQ2 cannot be definitively answered. Consequently, this question should be re-evaluated on an actual downstream scenario which includes a stronger reliance on quantified values than *matches@k*.

The representation containing only chained binary relations, $r_{\hat{\rho},\mathrm{ch},\mathrm{e}}$, provides mostly worse results compared to representations (a) not considering the quantification, (b) representing the quantification as a separate relation or (c) adding η for all evaluated embedding methods. As such, we conclude that none of the evaluated embedding methods is able to sufficiently deal with the indirections (RQ3). Consequently, alternative approaches to represent ternary relations in knowledge graphs should be explored.

5. Future Work

On a conceptual level, the presented representations all concern non-temporal aspects of procedural knowledge. However, in practice the order in which actions are executed often plays a significant role. As such, we intend to provide representations that address this aspect of procedural knowledge and evaluate whether ordered RDF2Vec provides a benefit over other embedding methods in this scenario. Also, since BoxE is able to deal with higher arities it could be used directly to gain a better understanding of the implications of modelling the ternary relation as chained binary relations. As the detrimental effect of lower dimensional embeddings on *matches@k* visible in preliminary experiments was not evenly distributed over embedding methods, e. g. RDF2Vec was able to handle it better than its competitors, a more thorough investigation in this direction is planned. Furthermore, different aggregation methods than sum-based aggregations will be considered. While we argued that to evaluate the quality of the embeddings in a downstream scenario a specific metric, i. e. *matches@k*, is beneficial, an evaluation on standard link prediction metrics, i. e. *hits@k*, could be used to quantitatively evaluate the node embedding quality. Furthermore, an evaluation with an actual downstream task might give further insights into the expressiveness of *matches@k* and the achieved quality of the respective embedding method. Furthermore, we plan to evaluate the approach on more datasets.

6. Conclusion

In this paper, we took a closer look at embeddings of procedural knowledge. From exploring more detailed modelling patterns than were previously published [5], over analysing the properties they exhibit, to evaluating—on a theoretical and experimental level—which embedding methods are best suited to embed them. Additionally, subgraph selection strategies were adapted and alternative aggregation strategies evaluated. We discovered that the theoretical inspection of the properties of embedding methods is not confirmed by the experimental results. Especially, methods capable of embedding literals did not provide a benefit over using literal agnostic methods. Most of the evaluated methods seem to have difficulties of capturing the indirections introduced by modelling ternary relations as chained binary relations. However, explicitly adding the implied high-level relation seems to mitigate this problem. An evaluation on an actual downstream task will provide further insights into our findings.

References

[1] G. Buchgeher, D. Gabauer, J. Martinez-Gil, L. Ehrlinger, Knowledge graphs in manufacturing and production: A systematic literature review, IEEE Access 9 (2021) 55537–55554.

[2] R. Abboud, I. Ceylan, T. Lukasiewicz, T. Salvatori, Boxe: A box embedding model for knowledge base completion, Advances in Neural Information Processing Systems 33 (2020) 9649–9661.

[3] J. Portisch, N. Heist, H. Paulheim, Knowledge graph embedding for data mining vs. knowledge graph embedding for link prediction–two sides of the same coin?, Semantic Web 13 (2022) 399–422.

[4] D. R. Krathwohl, A revision of bloom's taxonomy: An overview, Theory into practice 41 (2002) 212–218.

[5] R. Nordsieck, A. Hummel, M. Heider, A. Hoffmann, J. Hähner, Towards conceptual and procedural models of operator knowledge in industrial information models, First International Workshop On Semantic Industrial Information Modelling (SemIIM) at the 19th Extended Semantic Web Conference (ESWC 2022) (2022).

[6] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Advances in Neural Information Processing Systems 26 (2013).

[7] T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, G. Bouchard, Complex embeddings for simple link prediction, in: International conference on machine learning, 2016, pp. 2071–2080.

[8] Z. Sun, Z.-H. Deng, J.-Y. Nie, J. Tang, Rotate: Knowledge graph embedding by relational rotation in complex space, arXiv preprint arXiv:1902.10197 (2019).

[9] V. Gutiérrez-Basulto, S. Schockaert, From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules, in: Sixteenth International Conference on Principles of Knowledge Representation and Reasoning, 2018.

[10] G. A. Gesese, R. Biswas, M. Alam, H. Sack, A survey on knowledge graph embeddings with literals: Which model links better literal-ly?, Semantic Web 12 (2021) 617–647.

[11] J. Portisch, H. Paulheim, Walk this way! entity walks and property walks for rdf2vec, arXiv preprint arXiv:2204.02777 (2022).

[12] L. Hörner, M. Schamberger, F. Bodendorf, Externalisierung von prozess-spezifischem mitarbeiterwissen im produktionsumfeld, Zeitschrift für wirtschaftlichen Fabrikbetrieb 115 (2020) 413–417.

[13] J. Portisch, H. Paulheim, Putting rdf2vec in order, arXiv preprint arXiv:2108.05280 (2021).

[14] S. Zhang, Y. Tay, L. Yao, Q. Liu, Quaternion knowledge graph embeddings, Advances in Neural Information Processing Systems 32 (2019).

[15] A. Kristiadi, M. A. Khan, D. Lukovnikov, J. Lehmann, A. Fischer, Incorporating literals into knowledge graph embeddings, in: International Semantic Web Conference, 2019, pp. 347–363.