Leveraging Semantic Representations via Knowledge Graph Embeddings

Franz Krause, Kabul Kurniawan, Elmar Kiesling, Jorge Martinez-Gil, Thomas Hoch, Mario Pichler, Bernhard Heinzl, and Bernhard Moser

1 Introduction

Knowledge graphs are becoming increasingly recognized as a valuable tool in data-driven domains like healthcare [1], finance [2], and manufacturing [3], where they have gained considerable popularity in recent research. They are commonly employed to represent and integrate both structured and unstructured data, providing a standardized approach to encode domain knowledge [4]. Built on ontologies that conceptualize domain classes, relations, and logical inference rules, KGs represent specific instantiations of ontological models and their inherent semantic characteristics. Typically, KGs are divided into two modules: a terminological TBox containing concepts (such as the class of a manufacturing process) and an assertive ABox containing real-world instances (such as unique executions of a manufacturing process).

F. Krause (✉)
University of Mannheim, Data and Web Science Group, Mannheim, Germany
e-mail: franz.krause@uni-mannheim.de

K. Kurniawan
WU, Institute for Data, Process and Knowledge Management, Vienna, Austria

Austrian Center for Digital Production (CDP), Vienna, Austria
e-mail: kabul.kurniawan@wu.ac.at

E. Kiesling
WU, Institute for Data, Process and Knowledge Management, Vienna, Austria
e-mail: elmar.kiesling@wu.ac.at

J. Martinez-Gil · T. Hoch · M. Pichler · B. Heinzl · B. Moser
Software Competence Center Hagenberg GmbH, Hagenberg, Austria
e-mail: Jorge.Martinez-Gil@scch.at; thomas.hoch@scch.at; Mario.Pichler@scch.at; bernhard.heinzl@scch.at; Bernhard.Moser@scch.at

J. Soldatos (ed.), *Artificial Intelligence in Manufacturing*,
https://doi.org/10.1007/978-3-031-46452-2_5

We adopt the notion of a (standard) KG $\mathcal{G} = (V, E)$ as described in [5], which is represented by a set of nodes V (also referred to as vertices) and a set of triples $E \subseteq V \times R \times V$ consisting of directed and labeled edges. Here, R denotes the set of valid relation types defined in the underlying ontology. Thus, an edge in the form of a triple $(s, p, o) \in E$ implies an outgoing relation from the subject $s \in V$ to the object $o \in V$ via the predicate $p \in R$. Given such a KG, embedding techniques aim to exploit the topology of the graph to generate latent feature representations

$$\gamma : V \to \Gamma \tag{1}$$

of its nodes V in a latent representation space Γ, e.g., $\Gamma = \mathbb{R}^d$ with $d \in \mathbb{N}$, thereby enabling their utilization in downstream applications, e.g., graph-based machine learning (ML). However, the findings of this work can be applied almost analogously to the most well-known KG extensions, such as labeled property graphs like Neo4j [6].

In addition to the improved applicability of graph-based data in tasks like recommendation systems [7] or question answering [8], embedding formalisms have also proven to be valuable as intrinsic complements to graph-structured data. This is due to their ability to provide an empirical approach for enhancing the expressivity of graph topologies by means of downstream tasks like entity linking [9] and link prediction [10]. Consequently, related areas such as relational ML are receiving significant attention in both literature and applications [11].

In this chapter, we first provide a brief overview of representation learning as the enabler of KG embeddings, addressing state-of-the-art embedding formalisms for generating lean feature representations and describing their functionalities. An analysis of the advantages and drawbacks of employing KG embeddings is provided, along with a discussion of associated open research questions. We focus specifically on potential challenges and risks that may hinder the usage of KG embeddings in the highly dynamic manufacturing domain. Accordingly, we present the methodologies developed within the Teaming. AI project to address those problems. In this context, we describe the applicability and potential benefits of KG embeddings in the human–AI-based manufacturing use cases of the project. Furthermore, we showcase the Navi approach as an enabler of dynamic KG embeddings that allows for real-time and structure-preserving computations of new or updated node representations.

2 Knowledge Graph Embeddings

The generation of KG embeddings as per Eq. (1) denotes a subdiscipline of representation learning. In the context of KGs, representation learning is applied to determine lean feature representations that are able to capture inherent semantic relationships between KG elements. Thus, we first provide a general overview of representation learning to subsequently describe its application in KG embeddings.

3 Representation Learning

Representation learning comprises techniques for the automatic detection of appropriate feature representations that can be employed by downstream models or tasks, such as machine learning models [12]. Thus, the main objective of representation learning is to eliminate the need for preprocessing raw input data. Given a set of observable variables V with semantic representations $\pi : V \rightarrow \Pi$ within an inherent representation space Π (which is not necessarily compatible with the downstream model), these techniques aim to generate an alternative feature mapping $\gamma : V \rightarrow \Gamma$ into a representation space Γ that satisfies the requirements of the desired task.

Representation learning can be performed in a supervised, unsupervised, or self-supervised manner. One example of a supervised approach for learning latent feature representations is the training of deep neural networks on labeled input data. Namely, given an input feature $\pi(v)$ for some $v \in V$, the hidden layer outputs (and also the output layer) obtained from the forward pass of the network can be considered as alternative representations $\gamma(v)$, as illustrated in Fig. 1.

Contrarily, unsupervised representation learning techniques can be utilized for unlabeled representations $\pi(v)$. Methods like principal component analysis or auto-encoders intend to reduce the dimensionality of high-dimensional input features. Accordingly, the goal of these algorithms is to determine alternative, low-dimensional representations without the consideration of any target feature except the input feature $\pi(v)$ itself. For example, auto-encoders feed a representation $\pi(v) \in \mathbb{R}^{d'}$ into a deep neural network and attempt to reconstruct it, i.e., $\pi(v)$ also serves as the output feature. However, the hidden layers are assumed to be low-dimensional to serve as alternative representations $\gamma(v) \in \mathbb{R}^d$ of $v \in V$ with $d \ll d'$ as depicted in Fig. 2.

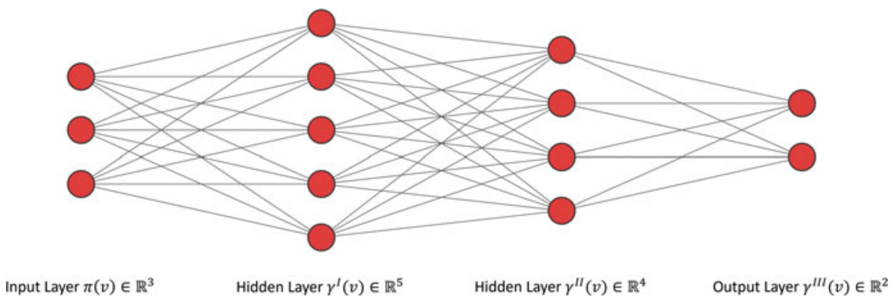

Input Layer $\pi(v) \in \mathbb{R}^3$
Hidden Layer $\gamma^{I}(v) \in \mathbb{R}^5$
Hidden Layer $\gamma^{II}(v) \in \mathbb{R}^4$
Output Layer $\gamma^{III}(v) \in \mathbb{R}^2$

Fig. 1 Deep neural networks as supervised representation learning formalisms

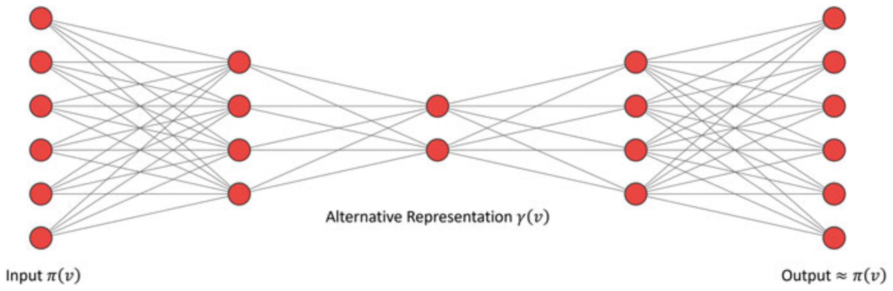
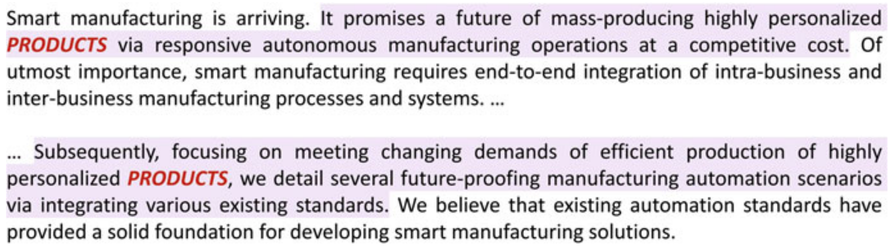


Fig. 2 Auto-encoders as unsupervised representation learning formalisms



Smart manufacturing is arriving. It promises a future of mass-producing highly personalized ***PRODUCTS*** via responsive autonomous manufacturing operations at a competitive cost. Of utmost importance, smart manufacturing requires end-to-end integration of intra-business and inter-business manufacturing processes and systems. ...

... Subsequently, focusing on meeting changing demands of efficient production of highly personalized ***PRODUCTS***, we detail several future-proofing manufacturing automation scenarios via integrating various existing standards. We believe that existing automation standards have provided a solid foundation for developing smart manufacturing solutions.

Fig. 3 Extract from the abstract in [15]. The semantics of the word *products* is encoded within the sentences that contain it

Finally, self-supervised representation learning aims to leverage the underlying structure $\mathcal{S}_V$ of unlabeled data that contains the variables $v \in V$ and which allows for deriving meaningful initial representations $\pi(v)$. For example, a word $v \in V$ may appear in a set of sentences $\pi(v)$ within a shared text corpus $\mathcal{S}_V$, as exemplified in Fig. 3. While state-of-the-art NLP models like BERT [13] usually split words into frequently occurring subword tokens via subword segmentation algorithms such as Wordpiece [14], the inherent methods can be applied analogously to sets of complete words. In the course of training such NLP models, numerical embeddings $\gamma(v) \in \mathbb{R}^d$ are assigned to the domain variables $v \in V$ with respect to their original representations $\pi(v)$. These alternative representations are optimized by backpropagating the output of the LLM for at least one element of its initial representation $\pi(v)$.

Analogously, most NLP techniques can be applied to KG structures $\mathcal{G} = (V, E)$ by characterizing directed graph walks $(v_1, p_1, v_2, p_2, v_3, \ldots, v_{l-1}, p_{l-1}, v_l)$ of depth $l - 1 \in \mathbb{N}$ as sentences that are composed of edges $(v_i, p_i, v_{i+1}) \in E$. For instance, the sample manufacturing KG depicted in Fig. 4 contains the 4-hop walk

*(**John**, executes, **Task 1**, output, **Product 1**, input, **Task 2**, output, **Product 2**).*

One of these transfer approaches is RDF2Vec [16], which utilizes random graph walks to generate input data for the NLP-based Word2Vec algorithm [17]. By doing so, a mapping $\overline{\gamma} : V \cup R \to \mathbb{R}^d$ is trained and thus, alternative representations of the graph nodes in V, but also for the relation types in R as well. Therefore,

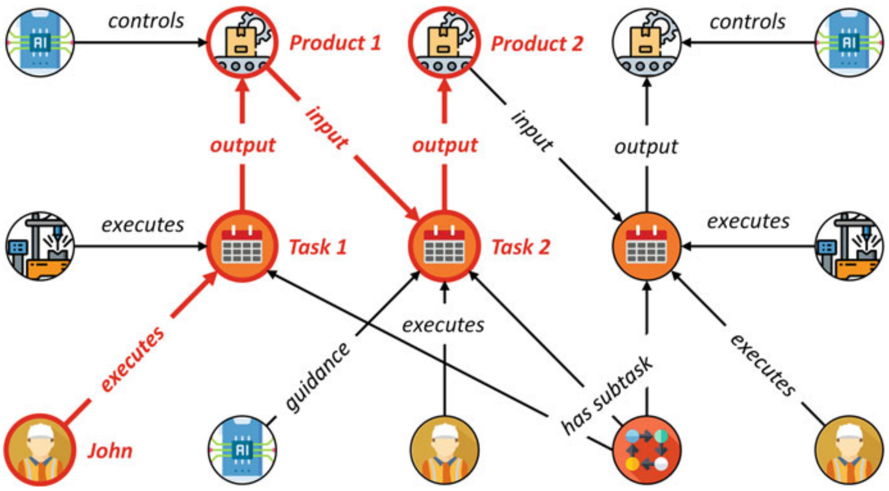


Fig. 4 Sample KG containing process flows within a production process

node embeddings can be derived via $\gamma(v) := \overline{\gamma}(v)$. Besides transfer approaches like RDF2Vec, various embedding algorithms exist, which are specifically tailored toward KG structures. These are further discussed in the following.

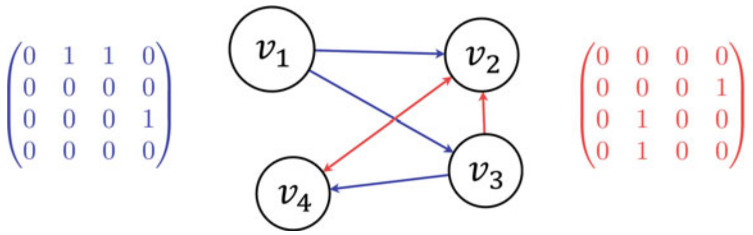
3.1 Representation Learning for Knowledge Graphs

KG embedding techniques denote a subdiscipline of representation learning, taking into account KG structures as initial input data. Given a KG $\mathcal{G} = (V, E)$, these approaches intend to provide numerical representations $\gamma : V \rightarrow \Gamma$ as per Eq. (1). However, as exemplified by RDF2Vec, KG embeddings may contain alternative representations of graph elements $y \notin V$ as well, such as embeddings of relations, but also edges or subgraphs. Thus, in general, a KG embedding is a mapping $\overline{\gamma} : \Omega \rightarrow \Gamma$, where Ω represents a collection of KG elements pertaining to $\mathcal{G}$. The node embedding of some $v \in V$ is accordingly obtained by restricting $\overline{\gamma}$ to V, i.e., $\gamma(v) = \overline{\gamma}(v)$.

Based on the research conducted in [10], KG embedding methods can be categorized into three model families, namely tensor decomposition models, geometric models, and deep learning models. We adopt this subdivision in the following.

3.1.1 Tensor Decomposition Models

Tensor decomposition models for KG embeddings are based on the concept of tensor decompositions within the area of multilinear algebra [18]. These attempt



$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \qquad \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Fig. 5 Sample KG with $n = 4$ nodes and $k = 2$ relations r_1 (blue) and r_2 (red), including their respective adjacency matrices $\mathcal{A}_1$ and $\mathcal{A}_2$

to characterize tensors via sequences of simplified tensor operations. For a KG $\mathcal{G}$, this approach is applied to its unique adjacency tensor $\mathcal{A} \in \{0, 1\}^{k \times n \times n}$, defined as

$$\mathcal{A}_{h,i,j} = 1 \iff \left(v_i, r_h, v_j\right) \in E.$$

Here, $k \in \mathbb{N}$ denotes the cardinality of the underlying relation set R and $n \in \mathbb{N}$ is the number of nodes in V. Accordingly, without loss of generality, we may assume labeled sets $R = \{r_1, \ldots, r_k\}$ and $V = \{v_1, \ldots, v_n\}$, as exemplified in Fig. 5.

Accordingly, tensor decomposition-based KG embedding methods intend to approximate $\mathcal{A}$ by a sequence of lower dimensional tensor operations. Among these methods, RESCAL [19] is considered to be the first work to apply this methodology for determining KG embeddings. Regarding $\mathcal{A}$, it proposes a rank-d factorization

$$\mathcal{A}_h \approx \mathcal{X} \cdot \mathcal{R}_h \cdot \mathcal{X}^T$$

of its h-th slice $\mathcal{A}_h \in \{0, 1\}^{n \times n}$ by means of matrices $\mathcal{X} \in \mathbb{R}^{n \times d}$ and $\mathcal{R}_h \in \mathbb{R}^{d \times d}$ with $d \ll n$. Therefore, the i-th row of the matrix $\mathcal{X}$ contains an alternative representation $\gamma(v_i) := \left(\mathcal{X}_{i,1}, \ldots, \mathcal{X}_{i,d}\right) \in \mathbb{R}^d$ of $v_i \in V$. The optimization of the matrices $\mathcal{X}$ and $(\mathcal{R}_h)_{1 \le h \le k}$ is accordingly achieved by solving the minimization problems

$$\min_{\mathcal{X},\mathcal{R}_h} f(\mathcal{X}, \mathcal{R}_h) \text{ for } f(\mathcal{X}, \mathcal{R}_h) = \frac{1}{2}\left(\sum\nolimits_{h=1}^{k} \|\mathcal{A}_h - \mathcal{X} \cdot \mathcal{R}_h \cdot \mathcal{X}^T\|_F^2\right),$$

with the Frobenius norm $\|\cdot\|_F$ and the corresponding element-wise operations

$$f(h, i, j) = \frac{1}{2}\left(\mathcal{A}_{h,i,j} - \gamma(v_i)^T \cdot \mathcal{R}_h \cdot \gamma(v_j)\right)^2.$$

To reduce the complexity of these optimizations, DistMult proposes to use diagonal matrices $(\mathcal{R}_h)_{1 \le h \le k}$ [20]. However, by doing so, DistMult is limited to symmetric relations. ComplEx solves this problem by employing $\mathbb{C}$-valued embedding spaces [21]. In addition to the mentioned models, numerous other tensor decomposition models for KG embeddings exist, including ANALOGY [22], SimplE [23], and HolE [24].

3.1.2 Geometric Models

Geometric KG embedding models represent semantic relations as geometric transformations within a corresponding embedding space. In contrast to tensor decomposition models, embeddings are not determined based on characteristics of the unique adjacency tensor $\mathcal{A}$, but with respect to individual facts $(s, p, o) \in E$.

As outlined in [10], transformations $\tau_p(s) := \tau\left(\gamma(s), \overline{\gamma}(p)\right) \in \Gamma$ are applied for subject nodes $s \in V$ regarding predicates $p \in R$. Accordingly, based on a distance measure $\delta : \Gamma \times \Gamma \rightarrow \mathbb{R}_{\geq 0}$, KG embeddings are computed via score functions

$$f(s, p, o) := \delta\left(\tau_p(s), \gamma(o)\right).$$

Among the family of geometric KG embedding methods, TransE [25] constitutes the most famous approach. As a translational model, it approximates object representations $\gamma(o)$ via $\gamma(o) \approx \tau_p(s) = \gamma(s) + \overline{\gamma}(p)$. Various geometric KG embedding models build upon the idea of TransE, improving the representation of nodes and relations by introducing additional components or transformations, such as

- Relationship-specific hyperplanes to capture complex interactions between nodes and relationships more effectively (TransH) [26]
- Relationship-specific node projection matrices to handle entities and relationships with different characteristics more flexibly (TransR) [27]
- Adaptive projection matrices regarding differing node-relation-pairs (TransD) [28]
- Relationship clustering to group similar relations (TransG) [29]

For a comprehensive overview of these methods, we refer to [10]. This work also introduces negative sampling as a common obstacle of KG embedding formalisms. Due to the open-world assumption of KGs, $(s, p, o) \notin E$ does not necessarily imply that the fact is false. Rather, it means that the KG does not contain information about its validity. Thus, negative sampling is applied to create a set of false facts $E_{neg} \subseteq V \times R \times V$ with $E \cap E_{neg} = \emptyset$ to train the embeddings in a supervised way.

3.1.3 Deep Learning Models

Graph-based deep learning (DL) approaches, also referred to as Graph Neural Networks (GNNs), exist for some time already, especially in the context of complex network systems and their underlying undirected graph structures [30]. However, the application of such algorithms on directed and labeled KGs may lead to a loss of relevant information. To address this issue, Graph Convolutional Networks

(GCNs) were first introduced to account for directed edges [31]. Furthermore, to accommodate different relation types, Relational Graph Convolutional Networks (RGCNs) were elaborated as extensions of GCNs [32], which were subsequently extended by means of attention mechanisms [33] in Relational Graph Attention Networks (RGATs) [34].

In contrast to geometric KG embedding models that apply score functions to individual triples and tensor decomposition models that intend to reduce the dimensionality of the adjacency tensor $\mathcal{A}$, DL-based models perform predictions for labeled nodes $v \in \overline{V}$, taking into account itself and its outgoing neighbors

$$N(v) := \{y \in V \mid \exists (s, p, o) \in E : (s = y \wedge o = v) \vee (s = v \wedge o = y)\}.$$

These labels can be derived from the KG itself via node assertions or link assignments, or they can be external, such as numerical or nominal node attributes. Adjacent node representations are meant to be aggregated to receive a composite node representation of v. By backpropagating a suitable loss, initial embeddings of v and its neighbors are optimized. This process is repeated for each labeled training node to generate latent feature representations for all $v \in \overline{V} \cup \{N(v) : v \in \overline{V}\}$. The formalism proposed in [32] subdivides $N(v)$ into relation-specific neighborhoods

$$N_r(v) := \{y \in V \mid \exists (s, p, o) \in E : (s = y \wedge o = v) \vee (s = v \wedge o = y) \wedge p = r\},$$

regarding relation types $r \in R$. Thus, given a matrix of (initial) feature representations $\mathcal{X} \in \mathbb{R}^{n \times d}$ (i.e., the i-th row of $\mathcal{X}$ is an embedding of $v_i \in V$), embeddings of outgoing neighbors can be incorporated in the forward pass of a GNN via

$$\mathcal{A}_h \cdot \mathcal{X} \in \mathbb{R}^{n \times d},$$

where $\mathcal{A}_h$ denotes the h-th slice of $\mathcal{A}$. For instance, in the context of the KG from Fig. 5, the composite representation of v_1 regarding the relation r_1 equals the sum of the initial embeddings of v_2 and v_3. To account for differing impacts of incoming and outgoing edges, R is typically extended via inverse relations r' for each $r \in R$. Some works also consider a self-relation r_0. Accordingly, by taking into account the adjacency matrices $\mathcal{A}_0 = Id$ and $\mathcal{A}_{2h} = \mathcal{A}_h^T$ for $1 \leq h \leq k$, we extend the set R via

$$\widehat{R} := R \cup \left\{r' \mid r \in R\right\} \cup \{r_0\} \quad \text{with} \quad r'_h = r_{2h}.$$

By doing so, GNN models capture the semantics of directed and labeled graphs by summing up weighted composite representations to receive a convoluted matrix

$$\sum_{h=0}^{2k} \widehat{\mathcal{A}}_h \cdot \mathcal{X} \cdot \mathcal{W}_h \in \mathbb{R}^{n \times d'},$$

including relation-specific weight matrices $\mathcal{W}_h \in \mathbb{R}^{d \times d'}$. Moreover, the extended adjacency tensor $\widehat{\mathcal{A}} \in \mathbb{R}^{(2k+1) \times n \times n}$ is not necessarily $\{0, 1\}$-valued. Rather, it is intended to contain normalization constants or attention scores to encode the significance of individual nodes and relations to the forward pass of a GNN. However,

$$\left(v_i, r_h, v_j\right) \notin E \Rightarrow \widehat{\mathcal{A}}_{h,i,j} = 0$$

still holds. If no normalization constants or attention mechanisms are to be implemented, this tensor can be directly derived from $\mathcal{A} \in \{0, 1\}^{k \times n \times n}$ by means of matrix transpositions and the insertion of an additional identity matrix. Finally, by introducing an activation function $\sigma : \mathbb{R} \to \mathbb{R}$ such as ReLu, the generalized forward pass of a GNN layer (including RGCNs and RGATs) can be defined as

$$\sigma\left(\sum_{h=0}^{2k} \widehat{\mathcal{A}}_h \cdot \mathcal{X} \cdot \mathcal{W}_h\right) =: \mathcal{X}' \in \mathbb{R}^{n \times d'}. \qquad (2)$$

4 Industrial Applications of Knowledge Graph Embeddings

The lack of use case scenarios poses a significant challenge to the application of KGs and corresponding KG embeddings in the manufacturing domain. Without specific applications, it becomes difficult to identify the relevant data sources, design appropriate KG structures, and create meaningful embeddings that capture the intricate relationships within manufacturing processes. Thus, the absence of concrete use cases hinders the exploration of the full potential of KGs and KG embeddings in improving efficiency, decision-making, and knowledge sharing within this domain.

As a result of the research conducted within the Teaming.AI project, which aims to enhance flexibility in Industry 4.0, while prioritizing human involvement and collaboration in maintaining and advancing AI systems, we identified several application scenarios within the manufacturing domain that can be leveraged by introducing industrial KGs and KG embeddings. These are introduced in the following.

Data Integration and Fusion Manufacturing involves diverse and complex data from various sources, such as sensors, process logs, or maintenance records. While KGs can integrate these heterogeneous data sources, KG embeddings map them into a shared representation space. By representing KG nodes and their relationships in this shared embedding space, it becomes easier to combine and analyze data from different sources, leading to enhanced data fusion capabilities.

Semantic Similarity and Recommendation KG embeddings allow for quantifying the semantic similarity between nodes. In the manufacturing domain, this can be useful for recommending similar products, materials, or processes based on their embeddings. For example, embeddings can help to identify alternative materials with desired properties or characteristics, thereby aiding in material selection.

Supply Chain Management Effective supply chain management is crucial for manufacturing. KGs and corresponding KG embeddings can help model and analyze complex supply chain networks by representing suppliers, products, transportation routes, and inventory levels as graph entities. By considering their semantic relations, embeddings can facilitate supply chain optimization, demand forecasting, and identifying potential risks in the supply chain.

Decision Support Systems KG embeddings and relational ML techniques can serve as a foundation for developing decision support systems in manufacturing. By learning from empirical semantic observations, these systems can provide recommendations, insights, and decision-making support to operators, engineers, and managers. For example, based on the current state of the manufacturing environment, the system can suggest optimal operating conditions or maintenance actions. Moreover, models can be learned to recommend ML models for AI activities, given the current manufacturing environment.

Fault Detection and Diagnosis KG embeddings combined with relational ML techniques can aid in fault detection and diagnosis in manufacturing systems. By analyzing historical data and capturing the relationships between machines, process variables, and failure events, embeddings can be used to build systems that identify faults or failures in advance. This facilitates proactive maintenance, reduces downtime, and improves overall effectiveness.

In conclusion, KGs allow for representing manufacturing concepts and entities (such as processes, machines, and human workers) and their semantic relationships. KG embeddings, on the other hand, capture inherent semantics in lean numerical representations which facilitate (i) the analysis of existing manufacturing knowledge and (ii) the extraction of new manufacturing knowledge based on empirical observations. As a powerful tool for representing domain knowledge in a human- and machine-interpretable way, KGs enable the combination of human comprehensibility with the computational capabilities of machines. This synergy of human and machine intelligence enables effective collaboration, decision-making, and efficient problem solving in the manufacturing domain. Moreover, it represents a step toward optimized human-in-the-loop scenarios [35] and human-centric Industry 5.0 [36].

However, the manufacturing domain is inherently dynamic, with continuous changes in its processes, equipment, materials, and market demands. Therefore, it is crucial to incorporate these dynamics into KG embeddings, which are typically designed for static snapshots of a domain (cf. Sect. 3.1). In the end, KG embeddings should be able to capture the evolving relationships, dependencies, and contextual information, preferably in real time. By incorporating dynamics, the embeddings can adapt to changes in manufacturing operations, such as process modifications, equipment upgrades, or variations in product requirements. This enables the representations to accurately reflect the current state of the manufacturing system and to capture the evolving aspects of runtime observations and data.

5 The Navi Approach: Dynamic Knowledge Graph Embeddings via Local Embedding Reconstructions

Most of the existing works on dynamic graph embeddings do not account for directed and labeled graphs. Rather, they are designed to be applicable to undirected and/or unlabeled graphs [37, 38], or they aim to embed temporally enhanced snapshots of non-dynamic graphs [39, 40]. Moreover, approaches like the one proposed in [41] exist that intend to perform an online training of KG embeddings by focusing on regions of the graph which were actually affected by KG updates. However, the overall embedding structure is still affected, leading to a need for continuous adjustments of embedding-based downstream tasks, such as graph-based ML models. Thus, we require a dynamic KG embedding formalism that (i) can produce real-time embeddings for dynamic KGs and (ii) is able to preserve the original structure of KG embeddings to allow for consistent downstream applications.

We propose to utilize the dynamic Navi approach [42], which is based on the core idea of GNNs as per Eq. (2). Given an initial KG $\mathcal{G}_{t_0} = (V_{t_0}, E_{t_0})$ at timestamp t_0, we assume an embedding $\widetilde{\gamma}_{t_0} : V_{t_0} \to \mathbb{R}^d$ based on some state-of-the-art KG embedding method from Sect. 3.1. Accordingly, a dynamic KG is defined as a family of stationary snapshots $(\mathcal{G}_t)_{t\in\mathcal{T}}$ with respect to some time set $\mathcal{T}$. Given a future timestamp $t > t_0$, the Navi approach provides a consistent embedding $\gamma_t : V_t \to \mathbb{R}^d$ so that previously trained downstream models can still be employed.

Since we leverage the idea of GNNs to reconstruct $\widetilde{\gamma}_{t_0}(v)$ through local neighborhoods, these reconstructions are based on the unique adjacency tensors $(\mathcal{A}(t))_{t\in\mathcal{T}}$ with $\mathcal{A}(t) \in \mathbb{R}^{k\times n_t\times n_t}$. Here, $n_t = \left|\bigcup_{\tau\leq t} V_\tau\right|$ denotes the number of nodes that were known to exist since the graph's initialization and thus $n_t \geq n_{t_0}$ holds. Thus, we assume an initial embedding matrix $\widetilde{\mathcal{X}}_{t_0} \in \mathbb{R}^{n_{t_0}\times d}$ that contains the initial embeddings as per $\widetilde{\gamma}_{t_0}$. This matrix is then reconstructed based on itself via a single-layer GNN

$$\sigma\left(\widehat{\mathcal{A}}(t_0)_0 \cdot \Theta_{t_0} \cdot \mathcal{W}_0 + \sum\nolimits_{h=1}^{2k} \widehat{\mathcal{A}}(t_0)_h \cdot \widetilde{\mathcal{X}}_{t_0} \cdot \mathcal{W}_h\right) =: \mathcal{X}_{t_0} \approx \widetilde{\mathcal{X}}_{t_0}$$

by taking into account the extended adjacency tensor $\widehat{\mathcal{A}}(t_0)$ (cf. Sect. 3.1.3). During the training process, a global embedding $\gamma_{r_0} \in \mathbb{R}^d$ is implemented regarding the self-relation r_0 so that $\Theta_{t_0} \in \mathbb{R}^{n_{t_0}\times d}$ contains n_{t_0} copies of γ_{r_0}. Moreover, instead of zero-value dropouts, overfitting is prevented by randomly replacing node embeddings with γ_{r_0} in the input layer, simulating the semantic impact of nodes that are not known at time t_0. It is also used to represent self-loops, enabling reconstructions that are independent of the (potentially unknown) initial representations. A detailed overview, including training settings and benchmark evaluation results, can be found in [42]. The evaluation implies that, given a timestamp $t > t_0$, this approach allows for high-qualitative and consistent embeddings $\gamma_t : V_t \to \mathbb{R}^d$ that are computed via

$$\sigma\left(\widehat{\mathcal{A}}(t)_0 \cdot \Theta_t \cdot \mathcal{W}_0 + \sum\nolimits_{h=1}^{2k} \widehat{\mathcal{A}}(t)_h \cdot \widetilde{\mathcal{X}}_t \cdot \mathcal{W}_h\right) =: \mathcal{X}_t,$$

i.e., the i-th row of $\mathcal{X}_t$ represents the embedding $\gamma_t(v_i)$ of the node $v_i \in V_t$. In the case of new nodes, $\widetilde{\mathcal{X}}_t$ and Θ_t are the extensions of $\widetilde{\mathcal{X}}_{t_0}$ and Θ_{t_0} by inserting copies of γ_{r_0}, respectively. Moreover, the update of the adjacency tensor can be performed via

$$\mathcal{A}(t)_h = I(t_0, t)^T \cdot \mathcal{A}(t_0)_h \cdot I(t_0, t) + \mathcal{B}(t_0, t)_h.$$

First, the matrix $I(t_0, t) \in \{0, 1\}^{n_{t_0} \times n_t}$ accounts for newly inserted nodes, i.e.,

$$I(t_0, t)_{i,j} = 1 \iff i = j.$$

Second, the update matrices $\mathcal{B}(t_0, t)_h \in \{-1, 0, 1\}^{n_t \times n_t}$ identify KG updates

$$\mathcal{B}(t_0, t)_{i,j} == \begin{cases} 1 & \iff \text{the edge } (v_i, r_h, v_j) \text{ was inserted between } t_0 \text{ and } t \\ -1 & \iff \text{the edge } (v_i, r_h, v_j) \text{ was deleted between } t_0 \text{ and } t. \end{cases}$$

After the KG update, a synchronizing assistant is to provide (i) the number of nodes n_t and (ii) the update tensor $\mathcal{B}(t_0, t) \in \{-1, 0, 1\}^{k \times n_t \times n_t}$. For instance, given an Apache Jena Fuseki[1] KG, existing logging tools like rdf-delta[2] can be extended to use them as synchronizing assistants. Moreover, while we focus on a single update at time $t \in \mathcal{T}$, transitions between arbitrary timestamps can be handled as well, i.e.,

$$\mathcal{A}(t')_h = I(t, t')^T \cdot \mathcal{A}(t)_h \cdot I(t, t') + \mathcal{B}(t, t')_h \text{ for } t_0 < t < t'.$$

In conclusion, the late shaping of KG embeddings via Navi reconstructions represents a promising approach for incorporating dynamic KG updates and semantic evolutions into KG embeddings as lean feature representations of domain concepts and instances. Besides the ability to allow for consistent embeddings, the results in [42] even showed that the reconstruction of existing embeddings often leads to an improved performance in downstream tasks like link predictions and entity classifications as key enablers of the industrial use case applications outlined in Sect. 4.

[1] Apache Software Foundation, 2021. Apache Jena, Available at https://jena.apache.org/.

[2] https://afs.github.io/rdf-delta/.

6 Conclusions

In this work, we highlighted the increasing importance of representing and exploiting semantics, with a specific emphasis on the manufacturing domain. While industrial KGs are already employed and utilized to integrate and standardize domain knowledge, the generation and application of KG embeddings as lean feature representations of graph elements have been largely overlooked. Existing KGs lack either domain dynamics or contextuality, limiting the applicability of context-dependent embedding algorithms. Thus, we provide an overview of state-of-the-art KG embedding techniques, including their characteristics and prerequisites. In this context, we emphasized the need for dynamic embedding methods and their implementation in concrete manufacturing scenarios, describing potential KG embedding applications in industrial environments, which were identified as a result of the Teaming.AI project. Furthermore, we introduced the concept of Navi reconstructions as a real-time and structure-preserving approach for generating dynamic KG embeddings.

To summarize, KGs and KG embeddings offer significant advantages for the manufacturing domain. The structured representation of complex relationships in KGs enables context-awareness, dynamic analysis, and efficient information retrieval. Furthermore, the utilization of KG embeddings promotes process optimization, leading to improved product quality, reduced errors, and an increased overall productivity.

Acknowledgments This work is part of the TEAMING.AI project which receives funding in the European Commission's Horizon 2020 Research Program under Grant Agreement Number 957402 (www.teamingai-project.eu).

References

1. Mohamed, S.K., Nováček, V., Nounu, A.: Discovering protein drug targets using knowledge graph embeddings. Bioinformatics **36**(2), 603–610 (2020)
2. Fu, X., Ren, X., et al.: Stochastic optimization for market return prediction using financial knowledge graph. In: IEEE International Conference on Big Knowledge, ICBK, pp. 25–32. IEEE Computer Society, New York (2018)
3. Buchgeher, G., Gabauer, D., et al.: Knowledge graphs in manufacturing and production: a systematic literature review. IEEE Access **9**, 55537–55554 (2021)
4. Hogan, A., Blomqvist, E., et al.: Knowledge graphs. ACM Comput. Surv. (CSUR) **54**(4), 1–37 (2021)
5. Krause, F., Weller, T., Paulheim, H.: On a generalized framework for time-aware knowledge graphs. In: Towards a Knowledge-Aware AI—Proceedings of the 18th International Conference on Semantic Systems, vol. 55, pp. 69–74. IOS Press, New York (2022)
6. Neo4j: Neo4j—the world's leading graph database (2012)
7. Palumbo, E., Rizzo, G., et al.: Knowledge graph embeddings with node2vec for item recommendation, In: The Semantic Web: ESWC Satellite Events, pp. 117–120 (2018)

8. Diefenbach, D., Giménez-García, J., et al.: Qanswer KG: designing a portable question answering system over RDF data. In: The Semantic Web: ESWC 2020, pp. 429–445 (2020)
9. Sun, Z., Hu, W., et al.: Bootstrapping entity alignment with knowledge graph embedding. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, pp. 4396–4402. AAAI Press, New York (2018)
10. Rossi, A., Barbosa, D., et al.: Knowledge graph embedding for link prediction: a comparative analysis. ACM Trans. Knowl. Discov. Data **15**(2), 1–49 (2021)
11. Nickel, M., Murphy, K., et al.: A review of relational machine learning for knowledge graphs. Proc. IEEE **104**(1), 11–33 (2016)
12. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. **35**(8), 1798–1828 (2013)
13. Devlin, J., Chang, M.-W., et al.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4171–4186. Association for Computational Linguistics, Kerrville (2019)
14. Schuster, M., Nakajima, K.: Japanese and Korean voice search. In ICASSP, pp. 5149–5152. IEEE, New York (2012)
15. Lu, Y., Xu, X., Wang, L.: Smart manufacturing process and system automation—a critical review of the standards and envisioned scenarios. J. Manuf. Syst. **56**, 312–325 (2020)
16. Ristoski, P., Rosati, J., et al.: Rdf2vec: RDF graph embeddings and their applications. Semantic Web **10**, 721–752 (2019)
17. Mikolov, T., Sutskever, I., et al.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, vol. 26. Curran Associates, Inc., New York (2013)
18. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
19. Nickel, M., Tresp, V., Kriegel, H.-P.: A three-way model for collective learning on multi-relational data. In: Proceedings of the 28th International Conference on International Conference on Machine Learning, pp. 809–816. Omnipress, New York (2011)
20. Yang, B., Yih, W.-T., et al.: Embedding entities and relations for learning and inference in knowledge bases. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, Conference Track Proceedings (2015)
21. Trouillon, T., Welbl, J., et al.: Complex embeddings for simple link prediction. In: Proceedings of The 33rd International Conference on Machine Learning, vol. 48, pp. 2071–2080. PMLR, New York (2016)
22. Liu, H., Wu, Y., Yang, Y.: Analogical inference for multi-relational embeddings. In: Proceedings of the 34th International Conference on Machine Learning, vol. 70, pp. 2168–2178 (2017). JMLR.org
23. Kazemi, S.M., Poole, D.: Simple embedding for link prediction in knowledge graphs. In: NeurIPS, pp. 4289–4300. Curran Associates Inc., New York (2018)
24. Nickel, M., Rosasco, L., Poggio, T.: Holographic embeddings of knowledge graphs. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, pp. 1955–1961. AAAI Press, New York (2016)
25. Bordes, A., Usunier, N., et al.: Translating embeddings for modeling multi-relational data. In: Proceedings of the 26th International Conference on Neural Information Processing Systems, vol. 2, pp. 2787–2795. Curran Associates Inc., New York (2013)
26. Wang, Z., Zhang, J., et al.: Knowledge graph embedding by translating on hyperplanes. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 28(1) (2014)
27. Lin, Y., Liu, Z., et al.: Learning entity and relation embeddings for knowledge graph completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 2181–2187. AAAI Press, New York (2015)
28. Ji, G., He, S., et al.: Knowledge graph embedding via dynamic mapping matrix. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics, pages 687–696. Association for Computational Linguistics, New York (2015)

29. Xiao, H., Huang, M., Zhu, X.: TransG: a generative model for knowledge graph embedding. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, pp. 2316–2325. Association for Computational Linguistics, New York (2016)
30. Wu, Z., Pan, S., et al.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. **32**(1), 4–24 (2021)
31. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)
32. Schlichtkrull, M., Kipf, T.N., et al.: Modeling relational data with graph convolutional networks. In: The Semantic Web ESWC, pp. 593–607. Springer, Berlin (2018)
33. Vaswani, A., Shazeer, N., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, pp. 5998–6008 (2017)
34. Busbridge, D., Sherburn, D., et al.: Relational Graph Attention Networks (2019)
35. Schirner, G., Erdogmus, D., et al.: The future of human-in-the-loop cyber-physical systems. Computer **46**(1), 36–45 (2013)
36. Leng, J., Sha, W., et al.: Industry 5.0: prospect and retrospect. J. Manuf. Syst. **65**, 279–295 (2022)
37. Pareja, A., Domeniconi, G., et al.: EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence AAAI, The Thirty-Second Innovative Applications of Artificial Intelligence Conference IAAI, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence EAAI, pp. 5363–5370. AAAI Press, New York (2020)
38. Trivedi, R., Farajtabar, M., Biswal, P., Zha, H.: DyRep: learning representations over dynamic graphs. In: International Conference on Learning Representations (2019)
39. Dasgupta, S.S., Ray, S.N., Talukdar, P.: HyTE: hyperplane-based temporally aware knowledge graph embedding. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, pp. 2001–2011. Association for Computational Linguistics, Belgium (2018)
40. Liao, S., Liang, S., et al.: Learning dynamic embeddings for temporal knowledge graphs. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, pp. 535–543. Association for Computing Machinery, New York (2021)
41. Wewer, C., Lemmerich, F., Cochez, M.: Updating embeddings for dynamic knowledge graphs. CoRR, abs/2109.10896 (2021)
42. Krause, F.: Dynamic knowledge graph embeddings via local embedding reconstructions. In: The Semantic Web: ESWC Satellite Events, pp. 215–223. Springer, Berlin (2022)