Graph Embedding-based Automatic Domain Ontology Completion

Xiaofeng Zhu, Haijiang Li*, Xiaoyu Liu
Cardiff University, UK
Zhux29@cardiff.ac.uk

Abstract. As an essential form of knowledge representation in the AEC industry, ontologies play an irreplaceable role in engineering applications. Due to complex logic within domain knowledge, the manual inspection of constructed domain ontologies remains time-consuming and labour-intensive. Therefore, an automated ontology completion method is pivotal. To fill this gap, a graph embedding-based ontology completion model is proposed in this research, which can automatically supplement plausible relations in domain ontologies. The proposed completion model is developed with an encoder-decoder structure. A binary feature representation and reverse relation addition are adopted to enhance graph embeddings. This completion model achieves promising results in the testing of several domain ontologies. The proposed automatic ontology completion method can significantly save time and labour on ontology inspection and potentially improve the reliability of ontology-driven engineering applications.

1 Introduction

The graph is a structured representation to describe the existence of relationships between objects. It is composed of vertices (nodes) and edges (directed or undirected), which are used to represent objects and relationships respectively. The idea of representing knowledge through a graph was first introduced by Frans in 1988 [1] and gained great attention after its usage in Google's search engine in 2012. So far, a large number of open knowledge graph datasets have been established, such as DBPedia, FreeBase [2], YAGO [3], Wikidata, etc. The knowledge graph stores objective information as structured RDF-style triples in the form of (head, relation, tail) or (subject, predicate, object) [4]. Ontologies are another type of graph-based domain knowledge representation. It encompasses domain concepts, entities, and relations that show the properties of a set of concepts and how they are related [5]. Within the last decade, ontology engineering has infiltrated many facets of research and is now clearly ubiquitous even within the architecture, engineering, and construction (AEC) domain [6]. These domain ontologies play significant roles in design optimisation, compliance checking, knowledge inference, holistic decision-making, etc. Currently, the construction of ontologies is still largely dependent on the manual work of domain experts, where subjectivity and uncertainty are inevitable. Although some guidelines for ontology construction, such as Ontology Development 101 [7], Best Practices of Ontology Development [8], etc., have been published to standardise the procedure, these domain ontologies are usually incomplete and require manual inspections to complete the missing relations among entities.

The process of completing incomplete triples (i.e., (Beam, ? , Building Element)) is called graph completion (GC). Many approaches have been proposed in recent years to supplement plausible missing facts in knowledge graphs. However, ontology-oriented knowledge completion in the AEC domain is still a vacuum. To fill this gap, the authors propose a graph embedding-based automatic domain ontology completion method in this paper. The proposed completion model follows an encoder-decoder structure, and a pre-processor is built into the model to convert graph data into computable tensor representation. The binary feature representation and reverse relation addition mechanism are utilised in the model to enhance the precision of graph

embedding. The proposed method has been tested on several domain ontologies and achieved promising results on ontology completion. The arises of automatic ontology completion can considerably save the time and resources spent on ontology inspection and reduce the proportion of manual work. Furthermore, the proposed ontology completion model can minimise the uncertainty in ontology construction and greatly improve the reliability of ontology-driven applications.

2 Related Work

2.1 Ontologies in AEC domain

In the last decade, plenty of ontologies and vocabularies have been created, investigated, and proposed [9]. These ontologies cover diverse areas in the AEC domain. For instance, there is the long-standing IFC ontology or vocabulary that is available in the Web Ontology Language since 2009 [10]. In parallel with the IFC ontology, the concept of link building data (LBD) was introduced. It gathers and shares building data via a set of available vocabularies and its ontology is diverse combinations of Building Topology Ontology (BOT), Building Element Ontology (BEO), Building Product Ontology (BPO), MEP ontology, the Ontology for Property Management (OPM), the File Ontology for Geometry Formats (FOG), and the Ontology for Managing Geometry (OMG). When considering asset management, diverse other ontologies can be of additional use, such as SAREF4BLDG, the Damage Topology Ontology (DOT), Real Estate Core (REC), and so forth. For ecosystems, the Brick ontology is the most predominant, which consists of Location, Equipment, and System classes. These classes hold a considerable inheritance tree and group the definition of Building, Space, Floor, Site, and Zone. As such, the Brick ontology is very well capable of defining the ecosystem of a building. The Flow System Ontology (FSO) is aimed entirely at the flows, ports, and interfaces in an HVAC system. It allows to define systems, components, and connections, including energy storage devices, supply units, and fluid and heat flows. The FSO can be used to compute flows through a system and optimise HVAC systems. Furthermore, some other existing ontologies, such as QUDT, SSN/SOSA, O&M, Time, etc. can easily be combined with the above-mentioned ontologies to form a comprehensive knowledge base.

2.2 Graph completion

To realize automatic graph completion, many approaches have been proposed in the computer science domain to find plausible missing relations in graphs. A common strategy is to rely on graph embedding, which aims to identify plausible triples by representing entities as vectors in a low-dimensional vector space and learning a parametrized scoring function for each relation. The graph embedding models can be broadly classified into the translational model, tensor decompositional model, and deep learning-based models.

The translation model construes the relation as a simple translation over a hidden entity representation [11]. It constructs a low-dimensional vector representation by drawing on information about the translation of entities. TransE [12] is one of the representative translational models proposed by Bordes et al. in 2013, where both entities and relations are considered vectors in the same space. However, it performs not well in one-to-many, many-to-one, and many-to-many relations [13,14]. To overcome this issue, extensions of TransE including TransH, TransR, TransD, TransM, and TransW have been recently proposed, which

have different relation embeddings and scoring functions. RotatE [15] is another translational-based representation learning approach introduced by Sun et al. in 2019, which can infer different relation patterns of symmetry and antisymmetry. Similar to RotatE, Zhang et al. proposed a translational distance model named HAKE [16], which explicitly models modulus information and considers the depth of the tree as moduli.

Tensor decompositional models use tensor products to capture rich interactions. RESCAL [17] is one of the representative tensor decomposition models, whose extension models are more trending. RESCAL tackles uncertainty and complex relational structures based on a statistical relational learning approach. It captures the underlying semantics of each entity using a vector representation and represents the pairwise interactions between potential factors as a matrix [18]. To simplify RESCAL, Yang et al. developed the DistMult [19], which uses bilinear diagonal matrices and reduces the number of parameters. ComplEx [20] proposed complex-valued embeddings to improve asymmetric relations modelling between entities, where entity and relation embeddings are in a complex space rather than one real space.

Deep learning models generally use an artificial neural network to learn how the head, relation, and tail embeddings interact. Convolutional neural networks normally yield the same performance as previous models with much fewer parameters. ConvE is the most representative convolutional model for graph completion and many extensions of ConvE are developed to enhance the ability in feature captures, such as InteractE, ConvKB and ConEx. In addition to the convolutional model, graph neural networks (GNN) also have been proven to be a powerful model that can learn the representation of an entity by aggregation of the features of the entities and neighbours. Hence, plenty of GNN-based approaches have been proposed for graph completion. For example, Graph Convolutional Network (GCN) [21] based approaches are a new trend for knowledge base completion. Li et al. [22] proposed a GCN-based model to complete ontologies selected from different domains, including wine, economy, transport and so on. Since then, many variants of GCN have been proposed to further improve the performance, such as RGCN, RAGCN, COMPGCN, etc. Furthermore, some researchers introduced hybrid models that combine the translational model with GCN. RotatE-GCN and TranE-GCN are representative of this type of model. Attention mechanisms are also introduced to help the above models get more contextualize embeddings.

According to the above review, ontologies is the most widely used knowledge representation in the AEC domain [9]. When constructing new ontologies or combining multiple existing ontologies, considerable labour is spent on completing missing relations. Hence, there is a demand for automatic ontology completion in the AEC domain. Although some existing solutions in the computer science domain have shown great potential to address this problem, no study in the AEC field has applied these approaches to automate the complementation of plausible relations in domain ontologies. Therefore, there is still a vacuum in automatic domain ontology completion.

3 Methodology

To fill the abovementioned research gap, the authors developed a decompositional ontology completion model that can automatically complement missing relations and nodes in the AEC-related ontologies. The proposed ontology completion model is comprised of a tuple converter, a graph embedding encoder and a triplet prediction decoder. Figure 1 presents the overall architecture of the proposed ontology completion model.

Although both are graph-based knowledge models, ontologies and knowledge graphs differ in the size of the model and the types of relationships. As a model focusing on domain knowledge representation, the number of nodes and relations in an ontology is much smaller than in a knowledge graph. In addition, the types of relationships between nodes in the ontology are also more limited. Therefore, the performance of knowledge graph complementation approaches is not ideal for ontology complementation due to limited input data. Considering these characteristics, the authors borrowed the idea from SimplE and use binary representation to describe the entity feature in an ontology. In the proposed ontology completion model, the embedding for each entity e is composed of two vectors $h_e, t_e \in \mathbb{R}^d$, while the relations r are still expressed as a single vector $v_r \in \mathbb{R}^d$. The vector h_e captures the entity's behaviour as the head of a relation and t_e captures the entity's behaviour as the tail of a relation. The representation of a triplet (e_1, r, e_2) is $< h_{e_1}, v_r, t_{e_2} >$. These two embedding vectors (h_e and t_e) for entities are learned independently of each other during the training: observing $(e_1, r, e_2) \in \zeta$ only updates h_{e_1} and t_{e_2} , not t_{e_1} and h_{e_2}.

Since ontology is a directed graph structure, entities that appear at the head and tail of a tuple are different in feature representation. This binary entity representation method separates the entity feature into a head feature and a tail feature. The increment of the feature dimensions enables a more accurate feature representation of the entities in the ontology, which compensates for the lack of feature information due to the small size of the ontology graph.

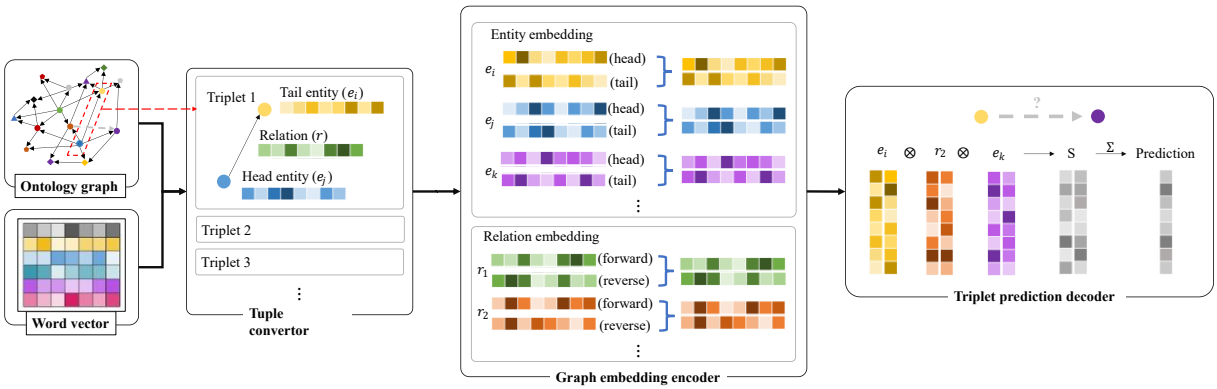


Figure 1: The architecture of the proposed ontology completion model.

4 Development

4.1 Tuple convertor

Graphs are data structures that cannot be directly processed by computers. Furthermore, all the instances and properties in the ontology are defined as unique Internationalized Resource Identifiers (IRIs). In some ontologies, entities and properties are expressed with the CamelCase naming convention. These characteristics make it more difficult to obtain semantic information.

To address the above problems, the authors developed a tuple converter to transfer graph information into processible triplets. Figure 2 shows the pre-processing of a tuple example.

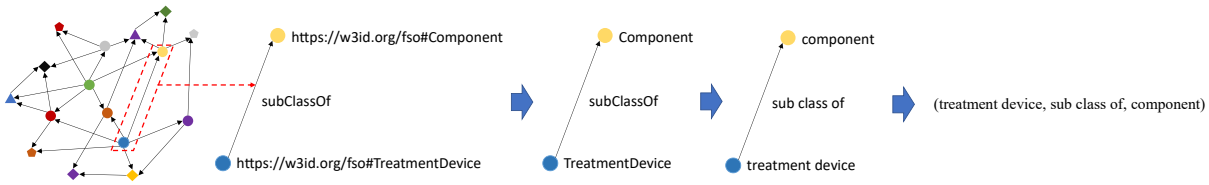


Figure 2: The pre-processing of a tuple example in the FSO.

The tuple converter first parses the ontology into a series of triples composed of IRIs with the help of a python toolkit (RDFLib). Then, regular expressions are used to remove redundant information from the IRI, only retaining the names of the instances or properties. The convert finally checks if the elements in the tuple are expressed in the Camel case and convert all Camel case expression into lowercase.

4.2 Graph embedding encoder

After pre-processing, the ontology has been converted to a set of textual triplets (e_i, r_j, e_k), which are still not calculable. To vectorize the triplets, a word-level tokenizer is adopted to split the name of entities and relations into tokens and the GloVe [23] word vectors are utilised to set the word embedding for each token. The initial embedding of the entities and relations is determined by the mean value of their token embeddings. The two vectors (h_e and t_e) are set as the same initial embedding at the beginning of training and then updated separately during the training. To improve the inferential capability, reverse relation addition [24] is introduced in the proposed model. That is, for each triplet (e_i, r, e_j), another vectorized tuple $< t_{e_j}, v_{r^{-1}}, h_{e_i} >$ is built to refine the representation. If this inverse relation (r^{-1}) has already been defined in the ontology, its embedding will be directly used as the initial embedding of r^{-1}. For inverse relations that are not predefined, their initial embeddings are set the same as the embedding of r.

4.3 Triplet prediction decoder

The decoder in the proposed model is essentially a probability-based predictor. It takes the embeddings of the triplet element (head entity, relation, tail entity) as input and outputs a predicted existence probability of this triplet. To estimate the plausibility of an arbitrary triplet, a scoring function [25] is predefined in the proposed decoder, which combines both forward and reverse triplet representations. The plausibility score of a specific triplet (e_i, r, e_j) is set to the average of the Canonical Polyadic scores for (e_i, r, e_j) and (e_j, r^{-1}, e_i) and calculated according to the following equations:

$$\phi(e_i, r, e_j) = \frac{1}{2}\left(h_{e_i} \times v_r \times t_{e_j} + t_{e_j} \times v_{r^{-1}} \times h_{e_i} \right)$$

where $\phi(e_i, r, e_j)$ represents for the plausibility score of the triplet (e_i, r, e_j), and h_{e_i}, v_r, t_{e_j}, $v_{r^{-1}}$ represents for the head embedding of entity e_i, embedding of forward relation r, the head embedding of entity e_j and embedding of reverse relation r respectively.

4.4 Model training

During the training process, a batch of positive triplets is iteratively taken from the domain ontology. To create the same number of negative triplets, the authors use the corruption method proposed by Bordes et al. [26]. For each positive triplet in the batch, a single negative triplet is generated by corrupting the positive triple. The specific corrupting procedure is as follows:

1. For a positive triplet (e_i, r, e_j), the head entity e_i or the tail entity e_j is randomly corrupted.
2. If the head entity is selected, another entity e_k will be randomly selected from $\mathcal{E} - \{e_i\}$ ($\mathcal{E}$ stands for the set of all entities) to replace e_i and form the corrupted triple (e_k, r, e_j).
3. If the tail entity is selected, another entity e_l will be randomly selected from $\mathcal{E} - \{e_j\}$ to replace e_j and form the corrupted triple (e_i, r, e_l).
4. A labelled batch (LB) will be generated, where positive triples are labelled as +1 and negative triples as −1.

In terms of the loss function, the margin-based loss functions have been proven to be useful for graph completion and widely used in the previous models (e.g., TransE, TransR, STransE, etc.). However, this type of loss function is more prone to overfitting compared to log-likelihood [27]. Therefore, the authors use the L2 regularized negative log-likelihood as the loss function in the proposed model, which can be calculated on each labelled batch (LB) through the following equation:

$$\mathcal{L}_{\text{Logistic}}(\theta) = \sum_{((h,r,t),l)\in \mathbf{LB}} log\ (1 + \exp\ (-l \cdot \phi(h,r,t))) + \ \lambda \|\theta\|_2^2$$

where θ represents the parameters of the model (the parameters in the entity and relation embeddings), l represents the label of a triple, $\phi(h,r,t)$ represents the plausibility score for triplet (h,r,t), λ is the regularization hyperparameter.

To avoid being stuck in a local optimum when updating the embedding parameters, the authors adopted stochastic gradient descent with mini-batches as the optimizer. The initial value of the learning rate and regularization hyperparameter is set as 0.01 and 0.02 respectively. Considering the proposed model will be tested by different domain ontologies, other hyperparameters associated with the dataset, such as training epoch and batch size, are not preset.

5 Experiment

In order to validate the practical performance of the proposed model in domain ontology completion. the authors selected 7 widely recognized ontologies as test examples. These selected ontologies cover various aspects of the AEC domain, including building products, building elements, flow systems, damage topology, property management, etc. More details about the selected ontologies can be found in Table 1.

Table 1: Basic statistics of the selected domain ontologies.

Ontology name	Classes	Object property	Data property	Individual	Annotation property

Building Element Ontology (BEO)	186	0	1	1	18
Building Topology Ontology (BOT)	10	16	1	5	21
Building Product Ontology (BPO)	25	22	6	0	15
Brick	1452	86	21	2566	58
Damage Topology Ontology (DOT)	16	13	3	1	15
Flow System Ontology (FSO)	14	23	0	1	13
Ontology for Property Management (OPM)	17	8	4	1	18

The above ontologies were processed by the tuple convertor, where all the IRIs are simplified as lower-case element names. These elements are then reorganized as RDF triplets in the format of (subject, predicate, object). Table 2 presents the number of entities, relations and triplets converted from each domain ontology.

Table 2: Number of the entities, relations, and triplets in each dataset.

Dataset	**Entity**	**Relation**	**Triplets**
Building Element Ontology (BEO)	1102	16	1530
Building Topology Ontology (BOT)	785	32	957
Building Product Ontology (BPO)	214	31	382
Brick	14283	77	52113
Damage Topology Ontology (DOT)	184	27	274
Flow System Ontology (FSO)	189	22	301
Ontology for Property Management (OPM)	156	23	177

The converted triplets were considered as positive triplets and divided into training, validation and test set according to the ratio of 80%, 10% and 10%. The negative triplets were automatically generated and loaded into the model together with the positive triplets. To obtain the optimum model, the model parameters were saved separately after a fixed epoch of training. The model parameters that perform best on the validation set were treated as optimum parameters and validated on the test set to show the practical performance.

Commonly used evaluation indicators for graph completion are Hits@k, Mean Rank (MR), and Mean Reciprocal Rank (MRR) [28]. Hits@k indicates the probability of correct prediction in the top k candidate triples calculated by the algorithm. The value of Hits@k is between 0 and 1. A larger value represents a better performance of the model. Mean Rank is the average value of the ranking of predictions/recommendations among all candidates. The smaller the value of MR, the better the prediction effect of the model. Mean Reciprocal Rank scores the predicted triples based on whether they are true or not. If the first predicted triple is true, its score is 1, and the second true score is 0.5, and so on. When the n-th triplet is established, it is scored $\frac{1}{n}$, and the final MRR value is the sum of all the scores. The larger the MRR value, the better the model effect [29]. The calculation formula of Hits@k, Mean Rank (MR), and Mean Reciprocal Rank (MRR) are shown as below.

$$H@k = \frac{|\{q \in Q : q < k\}|}{|Q|}$$

$$MR = \frac{1}{|Q|}\sum_{q \in Q} q$$

$$MRR = \frac{1}{|Q|}\sum_{q \in Q} \frac{1}{q}$$

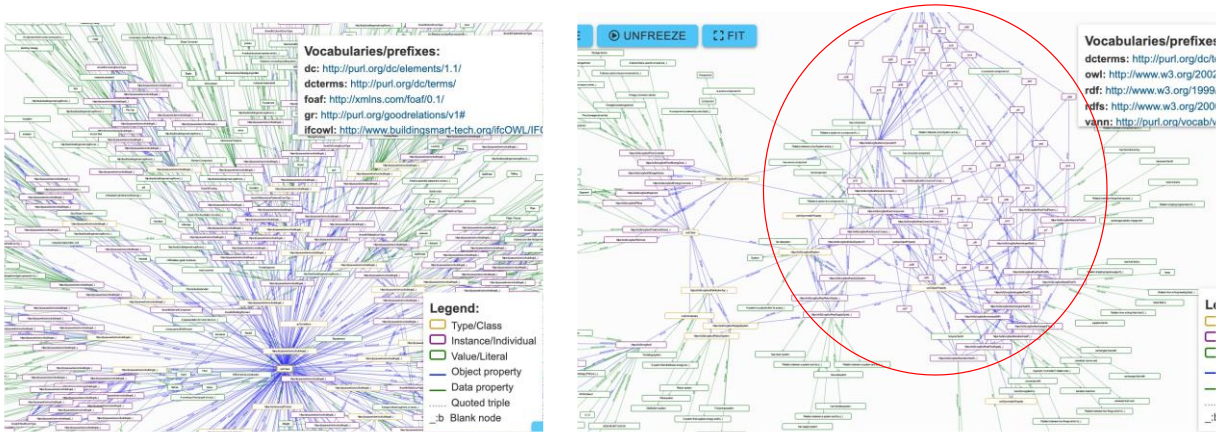
in which, q represents the prediction item, and Q represents all the prediction items given by the model.

Table 3: The completion results of each selected ontologies.

Dataset	Hits@1	Hits@3	MR	MRR
Building Element Ontology (BEO)	0.533	0.539	163.997	0.54
Building Topology Ontology (BOT)	0.599	0.609	108.078	0.608
Building Product Ontology (BPO)	0.553	0.566	29.3	0.568
Brick	0.558	0.598	1301.37	0.573
Damage Topology Ontology (DOT)	0.704	0.704	16.481	0.712
Flow System Ontology (FSO)	0.767	0.833	12.316	0.789
Ontology for Property Management (OPM)	0.556	0.556	25.083	0.568

Table 3 shows the completion results of all selected ontologies against the above evaluation indicators. It can be observed that the performance of the proposed completion model on Hits@1 generally remains between 55% and 60%. For some specific ontologies (e.g., DOT and FSO), the accuracy of completion can even exceed 70%. According to the above test results, the proposed model has been proven to be promising for automatic domain ontology completion.

To figure out the interconnection between ontology and the accuracy of completion, the authors compared statistics of the tested ontology dataset. However, the results do not show an obvious correlation with the number of entities, relations, or triplets. After visualising all the tested ontologies, the authors found the potential reason that might account for this difference. Figure 3 presents a visualization of the two tested ontologies (i.e., BEO and FSO).



Building Element Ontology (BEO) Flow System Ontology (FSO)

Figure 3: A visualization of the BEO and FSO.

It can be found that the entities in the FSO are well-connected, especially in the marked area. The BEO is largely treelike and with a lot of leaves. During the learning process, the embedding of entities is calculated based on their neighbour entities. The relation embeddings follow the same mechanism. Therefore, the embedding of entities and relations in the FSO can get more information during the training process. This makes the feature representation of elements in the FSO more precise than the BEO and enables a higher accuracy in ontology completion. Given the above analysis, the proposed ontology completion model can achieve promising performance on well-connected ontologies and pretty good results on dendritic ontologies with a baseline of 55% accuracy.

6 Conclusion

In this research, the authors proposed a novel ontology completion model, which can automatically supplement plausible relations between defined entities. This completion model is comprised of a tuple convertor, a graph embedding encoder, and a triplet prediction decoder. The binary feature representation and reverse relation addition are applied to improve the precision of the graph embedding of each entity and relation. The proposed model is tested on several domain ontologies and achieves promising performance on well-connected ontologies. Due to the structural sensitivity, the result of dendritic ontologies completion is not fair well. A weight system for connectivity might be introduced in future work. In addition, more AEC domain ontologies will be used to test the proposed model to obtain a comprehensive result of practical performance. The advent of automatic ontology completion may have a profound impact on the AEC industry. It can significantly reduce the time and labour spent on ontological inspections. More importantly, this ontology completion model can greatly enhance the reliability of all types of engineering applications driven by ontologies.

References

[1] F.N. Stokman, P.H. de Vries, Structuring Knowledge in a Graph, Hum Comput Interact. (1988) 186–206. https://doi.org/10.1007/978-3-642-73402-1_12.

[2] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, Proceedings of the ACM SIGMOD International Conference on Management of Data. (2008) 1247–1249. https://doi.org/10.1145/1376616.1376746.

[3] M. Nickel, V. Tresp, H.P. Kriegel, Factorizing YAGO : Scalable machine learning for linked data, WWW'12 - Proceedings of the 21st Annual Conference on World Wide Web. (2012) 271–280. https://doi.org/10.1145/2187836.2187874.

[4] Frank Manola, Eric Miller, Brian McBride, RDF Primer, W3C Recommendation. (2004). https://www.w3.org/TR/rdf-primer/ (accessed May 6, 2023).

[5] N. Guarino, D. Oberle, S. Staab, What Is an Ontology? Nicola, Handbook on Ontologies. (2009). http://www.gbv.de/du/services/toc/bs/368354474 (accessed March 14, 2023).

[6] L.J. McGibbney, B. Kumar, A Framework for Regulatory Ontology Construction within AEC Domain, Ontology in the AEC Industry: A Decade of Research and Development in Architecture, Engineering, and Construction. (2015) 193–215. https://doi.org/10.1061/9780784413906.CH09.

[7] N. Noy, Ontology Development 101: A Guide to Creating Your First Ontology, (2001).

[8] R. Rudnicki, Best Practices of Ontology Development, (2016). http://www.w3.org/2001/sw/wiki/Tools (accessed March 22, 2023).

[9] P. Pauwels, A. Costin, M.H. Rasmussen, Knowledge Graphs and Linked Data for the Built Environment, Structural Integrity. 20 (2022) 157–183. https://doi.org/10.1007/978-3-030-82430-3_7.

[10] J. Beetz, J. Van Leeuwen, B. De Vries, IfcOWL: A case of transforming EXPRESS schemas into ontologies, AI EDAM. 23 (2009) 89–101. https://doi.org/10.1017/S0890060409000122.

[11] M. Zamini, H. Reza, M. Rabiei, A Review of Knowledge Graph Completion, Information (Switzerland). 13 (2022) 396. https://doi.org/10.3390/info13080396.

[12] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, Adv Neural Inf Process Syst. 26 (2013).

[13] Z. Wang, J. Zhang, J. Feng, Z. Chen, Knowledge Graph Embedding by Translating on Hyperplanes, Proceedings of the National Conference on Artificial Intelligence. 2 (2014) 1112–1119. https://doi.org/10.1609/AAAI.V28I1.8870.

[14] Y. Lin, Z. Liu, M. Sun, Y. Liu, X. Zhu, Learning Entity and Relation Embeddings for Knowledge Graph Completion, (n.d.). www.aaai.org (accessed March 22, 2023).

[15] Z. Sun, Z.H. Deng, J.Y. Nie, J. Tang, RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space, 7th International Conference on Learning Representations, ICLR 2019. (2019). https://arxiv.org/abs/1902.10197v1 (accessed March 22, 2023).

[16] Z. Zhang, J. Cai, Y. Zhang, J. Wang, Learning Hierarchy-Aware Knowledge Graph Embeddings for Link Prediction, AAAI 2020 - 34th AAAI Conference on Artificial Intelligence. (2019) 3065–3072. https://doi.org/10.1609/aaai.v34i03.5701.

[17] M. Nickel, V. Tresp, H.-P. Kriegel, A Three-Way Model for Collective Learning on Multi-Relational Data, (2011).

[18] Q. Wang, Z. Mao, B. Wang, L. Guo, Knowledge graph embedding: A survey of approaches and applications, IEEE Trans Knowl Data Eng. 29 (2017) 2724–2743. https://doi.org/10.1109/TKDE.2017.2754499.

[19] B. Yang, W. tau Yih, X. He, J. Gao, L. Deng, Embedding Entities and Relations for Learning and Inference in Knowledge Bases, 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings. (2014). https://arxiv.org/abs/1412.6575v4 (accessed March 22, 2023).

[20] T. Trouillon, J. Welbl, S. Riedel, E. Ciaussier, G. Bouchard, Complex Embeddings for Simple Link Prediction, 33rd International Conference on Machine Learning, ICML 2016. 5 (2016) 3021–3032. https://arxiv.org/abs/1606.06357v1 (accessed March 22, 2023).

[21] T.N. Kipf, M. Welling, Semi-Supervised Classification with Graph Convolutional Networks, 5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings. (2016). https://arxiv.org/abs/1609.02907v4 (accessed March 25, 2023).

[22] N. Li, Z. Bouraoui, S. Schockaert, Ontology Completion Using Graph Convolutional Networks, (n.d.).

[23] J. Pennington, R. Socher, C.D. Manning, GloVe: Global Vectors for Word Representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014: pp. 1532–1543. http://nlp. (accessed February 16, 2022).

[24] Y. Lin, Z. Liu, H. Luan, M. Sun, S. Rao, S. Liu, Modeling Relation Paths for Representation Learning of Knowledge Bases, Conference Proceedings - EMNLP 2015: Conference on Empirical Methods in Natural Language Processing. (2015) 705–714. https://doi.org/10.18653/v1/d15-1082.

[25] A. Rossi, D. Barbosa, P. Merialdo, D. Firmani, A. Matinata, Knowledge Graph Embedding for Link Prediction: A Comparative Analysis, ACM Trans. Knowl. Discov. Data. 1, 1, Article. 1 (2020) 47. https://github.com/merialdo/research.lpca. (accessed March 30, 2023).

[26] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating Embeddings for Modeling Multi-relational Data, (n.d.).

[27] U.G. Alpes, M. Nickel, Complex and Holographic Embeddings of Knowledge Graphs: A Comparison, (2017). https://arxiv.org/abs/1707.01475v2 (accessed March 30, 2023).

[28] F. Akrami, M.S. Saeef, Q. Zhang, W. Hu, C. Li, Realistic Re-evaluation of Knowledge Graph Completion Methods: An Experimental Study, Proceedings of the ACM SIGMOD International Conference on Management of Data. (2020) 1995–2010. https://doi.org/10.1145/3318464.3380599.

[29] X. Zeng-lin, S. Yong-pan, H. Li-rong, W. Ya-fang, Review on Knowledge Graph Techniques, (2016).