Machine Learning-Based Enterprise Modeling Assistance: Approach and Potentials

Nikolay Shilov, Walaa Othman, Michael Fellmann, Kurt Sandkuhl

Machine Learning-Based Enterprise Modeling Assistance: Approach and Potentials

Nikolay Shilov[1*[0000-0002-9264-9127]], Walaa Othman[2[0000-0002-8581-1333]], Michael Fellmann[3[0000-0003-0593-4956]], Kurt Sandkuhl[3[0000-0002-7431-8412]]

[1]SPC RAS, St.Petersbuerg, Russia
[2]ITMO University, St.Petersbuerg, Russia
[3]University of Rostock, Rostock, Germany
nick@iias.spb.su, walaa.s.othman@gmail.com,
{michael.fellmann, kurt.sandkuhl}@uni-rostock.de

Abstract. Today, enterprise modeling is still a highly manual task that requires substantial human effort. Human modelers are not only assigned the creative component of the process, but they also need to perform routine work related to comparing the being developed model with the existing ones. Although the huge amount of information available today (big data) makes it possible to analyze more best practices, it also introduces difficulties since a person is often not able to analyze all of it. In this work, we analyze the potential of using machine learning methods for assistance during enterprise modeling. An illustrative case study proves the feasibility and potentials of the proposed approach, which can potentially significantly affect the modern modeling methods, and also has long-term prospects for the creation of new technologies, products, and services.

Keywords: enterprise modeling, assisted modeling, machine learning, graph neural networks, decision support.

1 Introduction

Today, the high speed of scientific and technological progress as well as globalization have led to the need to often develop and modify enterprise models (e.g., [1]), which are usually described using graph-based structures. At the same time, even though the human engineer is usually assigned the creative component of the process, he/she still needs to perform routine work related to comparing the being developed model with the existing ones (best practice). Although the huge amount of information available today (big data) makes it possible to analyze more existing models, it also introduces difficulties since a person is often not able to analyze all of it.

Usage of modeling patterns is currently one of the trends [2, 3]. However, patterns sometimes can be an inefficient solution due to their diversity, specialization for specific conditions, and the same need to analyze the available patterns "manually".

Efficient use of big data is a global challenge today, which is emphasized by the popularity of research in this area. In the view of significant development of

information technologies, machine learning methods using, for example, deep neural networks, have made a significant qualitative leap in the past few years.

As mentioned in [4] "…a human actor will more easily adapt decision making to context specific factors, while an automated service will only address context if this was explicitly included in its design.". The capability of such machine learning models as deep neural networks to take into account and generalize the whole available information (e.g., the entire enterprise model) might help to overcome this limitation. As a result, the application of such machine learning techniques to support design decisions might enable modelers to take into account the context and various aspects of the being built model when comparing it with numerous (hundreds, thousands, or even millions) available models (best practices). So, slightly rephrasing the question from [4], the research question considered is "Can graph-based machine learning discover tacit enterprise model patterns from existing solutions to support enterprise modelers?".

The goal of the presented work is to analyze the potential of using machine learning methods for assistance during enterprise modeling. This approach can potentially significantly affect the modern modeling methods, and also has long-term prospects for the creation of new technologies, products, and services.

The paper is structured as follows. The next section introduces the state of the art in the areas of decision support and assistance for enterprise modeling and similar domains, as well as machine learning techniques capable of dealing with graph-based structures. It is followed by the approach description. Section 4 presents the experimental evaluation of the developed approach. The research results are discussed in sec. 5 followed by conclusions.

2 State of the art Review

2.1 Recommender System Techniques in Conceptual Modelling

Today, assistance for configuration and modeling processes is in great demand, since in addition to the creative component, such processes include tasks related to the analysis of existing solutions for possible reuse, which is a very laborious process. Existing research efforts intending to aid the modeler when designing an enterprise model are works in the intersection of Recommender Systems (RS) and conceptual modeling. In the most general sense, RS "generate meaningful recommendations to a collection of users" [15]. While RS are quite well-known in domains such as e-commerce, not much research is available so far in the domain of enterprise modeling. Modeling is still a highly manual task that requires substantial human effort e.g. in order to decide which label is appropriate for a model element, to determine where to start and stop modeling (scope of the model), and to model on a consistent abstraction level since guidance is lacking in current tools [16].

Hence some initial techniques and prototypes have been developed that are capable of generating suggestions on how to complete a model currently being edited. These approaches mainly have been developed in the context of business process modeling. They are geared towards different assistance features such as to *ease the completion of*

a partially constructed model e.g. by presenting relevant fragments of already existing models [17], using pattern-based knowledge for completion [18], or other auto-completion mechanisms [19], by adding required information for model execution [20] or assist in applying the modeling syntax [21].

Another form of recommendation-based support is *suggestions for concrete modeling actions* also denoted as auto-suggest features [22]. Regarding the latter, an initial evaluation of a knowledge-based paradigm for suggestion generation led to promising results [23]. Finally, some approaches also intend to *improve the model quality* [24] or *provide domain-specific knowledge-support* in the case of modeling for software engineering where constraints have to be satisfied [25] or where support regarding model changes is sought e.g. to inform the user about untypical model changes that potentially lead to errors based on a large data set about model evolutions [26]. In general, the cited works develop task-specific recommendation approaches that often differ from classical recommender system implementations and paradigms. Recently, also the incorporation of machine learning e.g. to learn parameters for the recommendation approach is discussed as future work [25].

All the above approaches are mainly based on utilizing predefined patterns, what on the one hand makes the modeling process more reliable and predictable, but on the other hand less flexible and creative. This limitation is addressed by the proposed use of machine learning.

2.2 Machine Learning in Modeling

Application of machine learning (which is basically about finding tacit modeling patterns) to support configuration tasks resulted from usage of pattern libraries built manually or semi-automatically. For example, in paper [5], the ability to use patterns is considered an essential functionality of enterprise modeling tools in the modern era of digitalization. The authors of [3] propose a complex system for organizing patterns with an evaluation of their consistency, and the authors of [6] – a system of recommendations for specific patterns or known models. The efficiency and promising outlook of using patterns in enterprise modeling is also emphasized in work [2].

The patterns are also used in other modeling domains, for example, when configuring complex products [7]. In work [8], co-authored by one of the authors of this work, a method was proposed for configuring complex systems (namely, a network of information and computing resources to solve a specific problem) based on an analysis of the functionality of resources and existing constraints. However, since a significant part of the resources remained unchanged from task to task, and the tasks belonged to one problem area, the use of machine learning models could potentially speed up the configuration process, but this opportunity was not considered. A recommender system proposed in [9] is focused on supporting the configuration process (configuring production for the release of a software product) and uses collaborative filtering techniques to select similar resources for the release of similar software products. Quite interesting is the work [10], the authors of which represent robotic industrial systems in the form of graphs and use the analysis of embeddings to select the most suitable components. This approach is the closest to the idea proposed

in this paper, however, it uses algorithmic optimization and not machine learning methods.

A review of methodologies for creating ontologies (which are semantic models of a problem area described as graph-like structures) in an industrial context [11] showed that only one of the considered methodologies used today in the field of enterprise modeling considers methods of reusing ontologies, and only on the level of standards and fixed fragments. The authors of [12] provide an overview of various existing systems for supporting business process modeling, but machine learning methods are not mentioned among them. The same is also true for the survey of research in enterprise modeling [13].

The necessity of balancing between pattern flexibility/diversity and the number of patterns to be analyzed manually is emphasized in paper [14], the authors of which proposed to expand the patterns to the concept of "bag of fragments" (a set of fragments), which are compared with the current configuration. It can be said that this work serves as a kind of link between the use of patterns and machine learning methods. Machine learning methods that allow both to identify such patterns by analyzing and summarizing the available big data, and to select and offer them to the user, depending on the situation, can be useful for solving this problem.

Today there already exist machine learning models, including those based on deep neural networks, which are focused on working with network-like structures and with graphs as particular. Paper [27] proposes a method for representing a graph in a form suitable for use in machine learning models and supporting parallelization of the learning process for efficient use of GPUs in order to increase the speed and performance of the learning process. In [28], the authors use the random forest machine learning paradigm to analyze the main flows within a transport network, representing the latter in the form of a graph. Such models are often successfully used today in chemistry. In [29], the authors use a graph to represent chemical compounds to use machine learning methods to generalize and predict their properties. A similar problem is solved in [30], with the use of machine learning methods. The authors of [31] apply machine learning to assist modeling particle processes.

Graph Neural Networks (GNN). Analyzing graphs using machine learning has received a lot of attention due to the great expressive power of the graphs. Graph neural networks can be defined as deep learning-based methods that operate on graphs. According to [32], GNN can be categorized into four groups: convolutional graph neural networks, recurrent graph neural networks, graph autoencoders, and spatial-temporal graph neural networks. The main tasks for graph learning are:

1. Node level: includes node classification, regression, and clustering tasks. While the first task aims to classify the nodes into several classes, the regression task predicts a continuous value of the node. The clustering task splits nodes into disjoint groups.
2. Edge-level: the main tasks are to predict whether there is an edge between two nodes (edge prediction or link prediction) and to classify edges.
3. Graph-level: includes graph matching, graph classification, and graph regression.

To solve the *edge prediction* task for knowledge graph, there have been developed several models with the main focus on transforming the graph into low dimensional

space while preserving the semantic information or graph embedding based models. These models can be categorized into three groups [33]:

1. Translational-distance-based models inspired by word2vec [34] based on representing each word by a vector of length *n* that represents the coordinates of this word in the *n*-dimensional space (like TransE [35], TransH [36], TransM [37], and TransR [38]), however, these models are reported to have a low capability of capturing the semantic information.
2. Semantic-matching-based models (like DistMult [39] and Complex [40]) embed both the relations and the entities into a unified vector space and define a scoring function to measure the validity. Although these models capture more semantic information, they still have some drawbacks which make them inefficient when deep semantic information is needed.
3. Neural-network-based models (like ConvE [41], HypER [42], CNN-BiLSTM [43]).

The neural-network-based models take into consideration the node type and the path information, and use convolution layers and attention mechanisms for enhancing the embeddings. ConvE was the first model to use a convolutional neural network for graph completion. It uses 2D convolution for the entity and relation embedding after reshaping and concatenating them. HypER introduces "hypernetworks" based on ConvE to generate convolutional filter weights for each relation. HypER uses 1D filters for entity embeddings to simplify the interaction between entities and relational embeddings. CompGCN [44] is a novel GCN that performs a composition operator over each edge in the neighborhood of the central node. The composed embeddings are convolved using two filters representing the inverse and the original relations. The aggregated messages of the neighbors represent the updated embedding of the central node. CNN-BiLSTM combines bidirectional long short-term memory (BiLSTM) and convolutional neural network (CNN) modules with an attention mechanism. The CNN followed by BILSTM modules are used to embed relations into a low space dimension. The attention layer is used to capture the semantic correlation between the candidate relation and each path between the two entities, and extracts reasoning evidence from the representation of the paths to predict whether two nodes should be connected by the candidate relation or not.

While multiple models aimed at operating on graphs exist, there is still a lack of those aimed at work with enterprise models. With this research, we would like to bridge the gap between the models themselves and the application area of EM, and in this paper we are checking the fundamental possibility of this.

3 Proposed Approach

3.1 Machine Learning-Based Assistance of EM

In this section, we suggest main potential assistance scenarios during EM that can be supported by machine learning technologies based on the application of GNN.

Generally, GNNs are aimed at solving the following four tasks taking the entire available graph as the input:

1. Edge prediction: the model calculates the probability of the given edge.
2. Edge class prediction: the model classifies the given edge.
3. Node class prediction: the model classifies the given node.
4. Graph classification: the model classifies the entire graph.

Based on these tasks, the following EM tasks that can be potentially assisted with machine learning have been identified (Table 1):

Table 1. Association between tasks solved by GNN and EM processes

Task solved by GNN	Associated EM assistance suggestions
Edge prediction.	1. Suggestion of edges. 2. Identification of likely wrong edges.
Edge class prediction	3. Suggestion of edge class. 4. Identification of edges of likely wrong types or with wrong labels.
Node class prediction.	5. Suggestion of node class. 6. Identification of nodes of likely wrong types or with wrong labels. 7. Suggestion of nodes.
Graph classification.	8. Model verification. 9. Model validation.

1. Suggestion of possibly existing connections: given a partially defined EM, a number of connections can be randomly generated (or all possible connections can be generated) and those with high probability can be suggested to the modeler.
2. Identification of likely wrong edges: the probability of a newly added by the modeler edge can be evaluated and if it is low enough, the corresponding warning can be presented to the modeler.
3. Suggestion of the edge class: the model can suggest the class of the newly added edge (edge type, label, etc.).

At this point, it is worth mentioning that the "class" from the machine learning point of view is not necessarily only the class (or type) of an item in the modeling domain. This term can equally apply to various other characteristics of the item such as functions it performs, requirements it meets, or associated text labels. Besides, an item can be related to several classes simultaneously.

1. Identification of edges of likely wrong types or with wrong labels: the probability of the type or text label of a newly added by the modeler edge can be evaluated and if it is low enough, the corresponding warning can be presented to the modeler together with a suggestion of better fitting type or label.
2. Suggestion of the node class: the model can suggest the class of the newly added node (node type, label, etc.).

3. Identification of nodes of likely wrong types or with wrong labels: the probability of the type or text label of a newly added by the modeler node can be evaluated and if it is low enough, the corresponding warning can be presented to the modeler together with a suggestion of better fitting type or label.
4. Suggestion of nodes: in this scenario, a node needs to be generated first (for example, connected to a newly added by the modeler node), then classified, and then the probability of its edge(s) is evaluated; if the probability is high enough, the appropriate suggestion is shown to the modeler.
5. Model verification (checking for consistency, correctness, and meeting given standards): classification of the model to consistent/inconsistent, correct/incorrect, meeting the given standards or not.
6. Model validation (checking that the model fulfills the requirements and achieves the goal): classification of the model against the requirements.

While the first seven scenarios are clear and potentially possible given the sufficient training data (that does not seem to be impossible), the last two scenarios are subject of more remote future, since they require trusted training data in rather narrow domains, and the question "what data sources can be credibly used in this process" raised in [4] becomes important.

Summarizing the above scenarios, Fig. 1 illustrates the approach as a whole. The numbers in parenthesis correspond to the scenarios described above. The text labeling of edges and nodes can additionally be extended with using Natural Language Processing (NLP) machine learning models for text analysis, matching, and suggesting better naming during the modeling (cf. [45]), however, this issue is currently out of the scope of the current research.

3.2 GNN Models for Approach Feasibility Evaluation

In order to evaluate the validity of the above-described approach, two models have been built: the node classification model and the edge prediction model.

Node classification model. The node classification model is used to determine the node class (e.g., concept, resource, rule, etc.). The input of the model is the enterprise model graph represented by the node names and the edges connecting them. The output is the node class.

Before feeding the graph into the network, the label encoding has to be used to encode both the node name and the node's class name.

The model consists of two sage convolution layers [46] followed by three fully connected layers. The rectified linear unit (relu) function is used as the activation function for all the layers except the last layer where the sigmoid function is used. The sage convolution layers (SAGEConv) get the information from the neighbors and aggregate them using the mean function. The architecture used is shown in Fig .2.

Edge prediction model. The edge prediction model is aimed at the prediction of missing edges in the enterprise model. The input of the model is a graph, where the nodes of the graph represent the main entities of the enterprise model (like concepts, organizational units, resources, rules, etc.) and the edges represent the connections

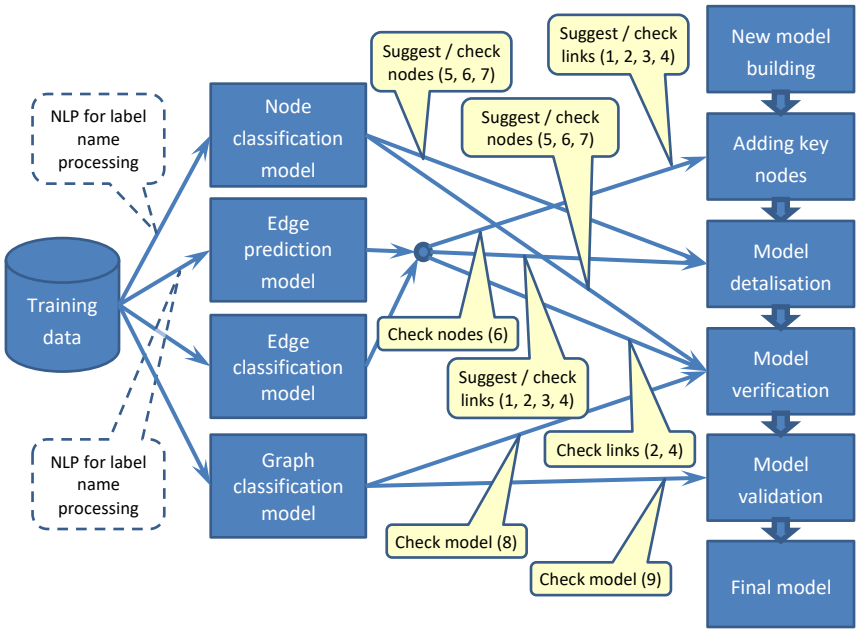

Fig. 1. Machine learning-based support of different EM stages.

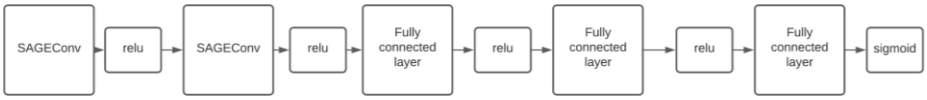

Fig. 2. The architecture of the neural network used for node classification.

between these entities. For each node, there are two attributes that describe the node (the class name, and the node description).

Before feeding the graph into the model, first, all its nodes have to be embedded. In the current version of the model, all nodes' names and the nodes' attributes in the graph are embedded using word2vec technique with Skip-gram architecture [47] using the word window of size 2.

For node description, the text is preprocessed by dropping all the stop words and mapping different forms of the word to the source using Snowball's stemming algorithm. Although this embedding is simple, which is essential for the proof of concept, it has a number of drawbacks for use in production. The main one is the need to recompute the embedding, every time a new node is added to the graph.

To keep things simple, in our approach, for now, we drop the description information and only use the node name alongside with the node class. The label encoding method is used to encode both the node class and the node name.

The edge prediction model consists of three graph convolutional layers (GCNConv) with rectified linear unit (relu) activation function. The similarity between the nodes is calculated using the cosine measure (Fig. 3).

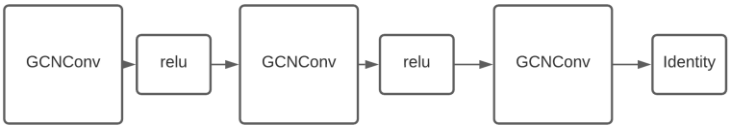

Fig. 3. The architecture of the neural network used for edge prediction task.

4 Experimental Evaluation

The overall experiment scheme is shown in Fig. 4.

The dataset based on models built during the EM university course consists of 55 enterprise models with 1728 edges and 35 node classes. Fig. 5 shows a sample model from the dataset. This set is not sufficient for productive use but can be still enough for the proof of the concept. Based on this test, two models have been trained: the node classification model and the edge prediction model.

The node classification model was trained for 200 epochs with the loss function negative log likelihood loss, Adam optimizer, and learning rate 0.005. Fig. 6 shows the loss/accuracy graph during the training of node classification model.

The edge prediction model was trained for 1000 epochs with the loss function Binary Cross Entropy, Adam optimizer, and learning rate 0.01. Fig. 7 shows the loss/accuracy graph during the edge prediction model training.

At this point, no hyperparameters tuning has been done. In future work, such hyperparameters as the learning rate, the number of epochs, and the batch size ca be adjusted.

For testing purposes, several enterprise models were built with definitely correct and wrong elements. Out of 1000 tested elements, the models correctly classified 962 elements resulting in an accuracy of 96.2%, precision of 0.9354, and the F1 score of 0.956.

5 Discussion

In our work, we presented a machine learning-based enterprise modeling support approach and analyzed its potentials. Our approach consists in application of machine learning models (GNN in particular) to support the modeler during the enterprise

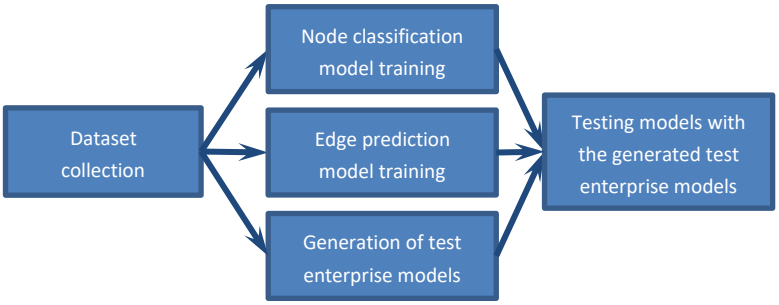

Fig. 4. The scheme of experimentation for testing the feasibility of the approach.

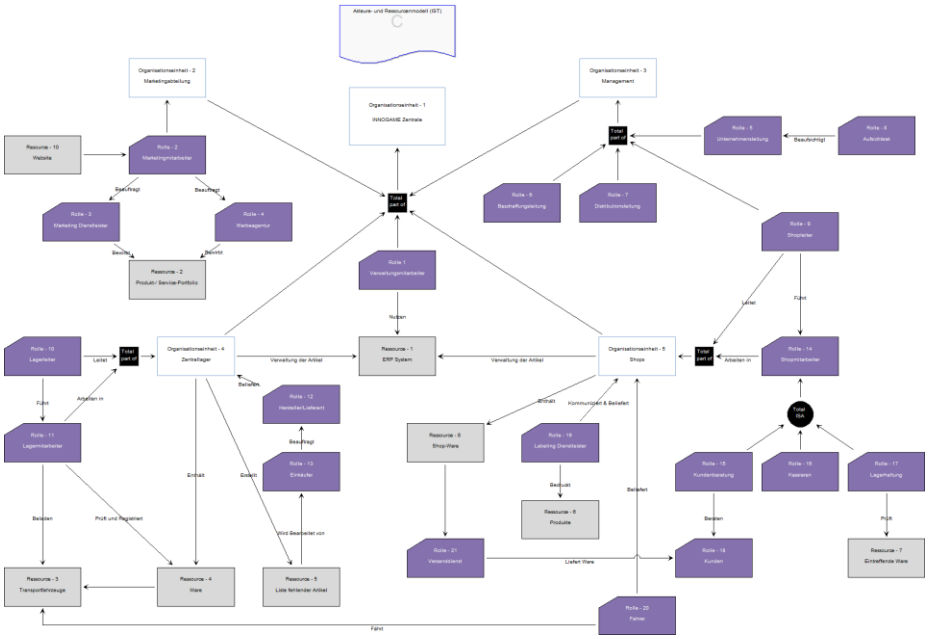

Fig. 5. Sample enterprise model example from the dataset used.

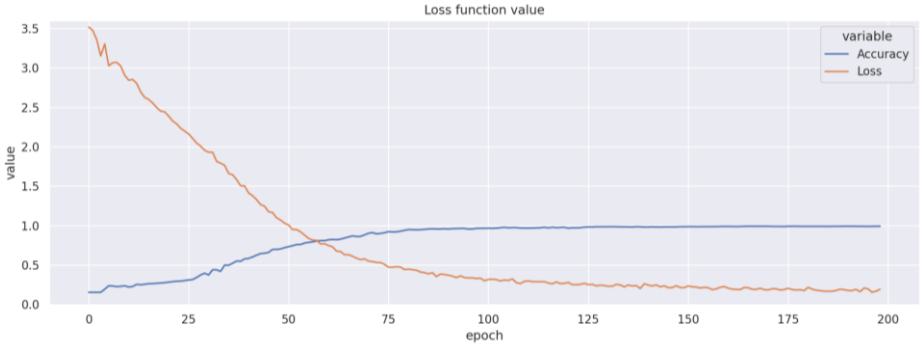

Fig. 6. The accuracy/loss graph during node classification model training.

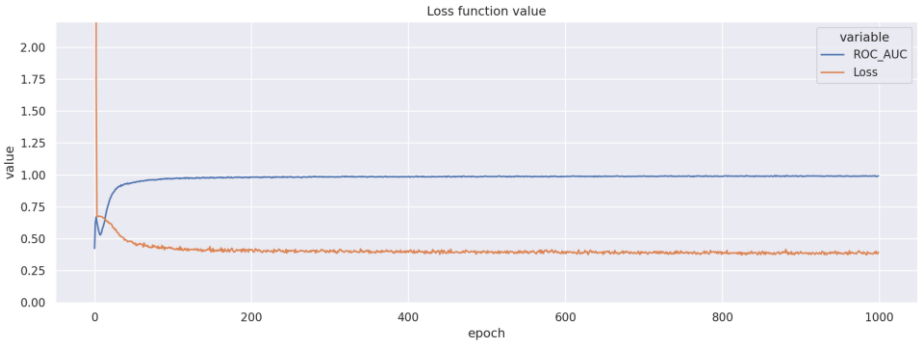

Fig. 7. The accuracy/loss graph during edge prediction model training.

modelling process. We have also outlined the potential assistance scenarios and carried out proof of concept experiment.

All in all, we have demonstrated the feasibility and potentials of leveraging machine learning for enterprise modeling. On the positive side, such an approach would be capable of analyzing large amounts of available models up to a volume that could be considered as "Big Data" (i.e. thousands of models). This could be used to extract common patterns or to extract best practices, if any indication of the model's quality or other performance attribute is available. It would also prevent the modeler from untypical or rare solutions that might be prone to errors or simply superseded by improved solutions that the modeler is not aware of.

On the negative side however, such features always bear the risk of "nudging" the modeler towards mainstream solutions. This could lead to novel, original and innovative solutions being unnoticed or, even worse, considered as inferior or non-efficient. Hence additional research is required how to improve this situation.

The model can also suffer from the problem of data imbalance. By that we mean, that some enterprise models can have elements that occur only once like some resources and processes, and others that appear much often like some organizational units. This can lead to poor performance on the low frequent elements. To overcome this problem, as a start, we included data from several enterprises' models, so each training model consists of several similar models. Also in some models, we duplicated these elements. In that case, we increased the number of the low-frequent elements. In future work, more experiments will be conducted to overcome this problem.

Besides, the approach can be affected by the "cold start problem". That is, a significant amount of source data is required for the approach to become efficient. The evident solution for this is to create larger, company-spanning model repositories which however bears the risk of unintentionally exposing confidential information to competitors. This in turn creates the need for fine-grained visibility concepts or mechanisms to anonymize parts of the models on different levels of confidentiality, (e.g. blind model element labels but still leverage the model structure for machine learning).

Finally, in order to improve the quality and relevance of the modelling support, it seems to be important to know as much as possible about the context of the model under construction. Context parameters could be its *goal* or *purpose* (e.g. ensuring

compliance, strategic decision making), *requirements* to the being developed model (e.g., ensuring model's flexibility), *subject matter* (e.g. procurement, production, marketing, finance, HR), enterprise-specific *enforced modeling conventions and constraints* (e.g. preferred terms, styles of expression, colors, formatting) or *intended addressees* (e.g. business experts, developers, lawyers, customers). Even though such context can potentially be identified based on already existing model elements, the problem is still critical when the modeler begins to model and the model contains only a few elements. In this situation, it is next to impossible to provide adequate modelling support without knowing this context parameters. Having the context as one of the input parameters for machine learning models is possible, however, this makes the problem of cold start more critical and requires additional research.

6 Conclusions and Future Work

The paper proposes application of machine learning (graph neural networks in particular) to assist a modeler during the enterprise modelling process and analyses its potentials. The main assistance scenarios associated with tasks solved by GNN have been identified. Some of them have been implemented to evaluate the feasibility of the approach.

It has been found that modelling support using machine learning techniques is feasible and has a significant potential. However, it still suffers the problems of suggesting mainstream solutions and identification of novel or unusual ones as incorrect. Besides, there is a problem of finding enough credible training data that is essential for building reliable and efficient models.

Planned future work is aimed at two main directions. First, deeper text analysis techniques can be used for dealing with labels and descriptions of enterprise model elements. This can be addressed with application of natural language processing techniques. The other direction is developing a prototype and collecting feedback from modelers to evaluate the effectiveness of the approach.

Acknowledgements. The paper is due to State Research no. 0073-2019-0005.

References

1. Riss U V., Maus H, Javaid S, Jilek C (2020) Digital Twins of an Organization for Enterprise Modeling. In: PoEM 2020: The Practice of Enterprise Modeling. Lecture Notes in Business Information Processing. Springer, pp 25–40
2. Fayoumi A (2018) Toward an Adaptive Enterprise Modelling Platform. Lecture Notes in Business Information Processing 335:362–371. https://doi.org/10.1007/978-3-030-02302-7_23
3. Awadid A, Bork D, Karagiannis D, Nurcan S (2018) Toward Generic Consistency Patterns in Multi-View Enterprise Modelling. In: ECIS 2018 Proceedings. AIS eLibrary, p 146
4. Snoeck M, Stirna J, Weigand H, Proper HA (2019) Panel Discussion: Artificial Intelligence meets Enterprise Modelling. In: The 12th IFIP Working Conference on The

Practice of Enterprise Modeling, PoEM 2019. CEUR
5. van Gils B, Proper HA (2018) Enterprise Modelling in the Age of Digital Transformation. Lecture Notes in Business Information Processing 335:257–273. https://doi.org/10.1007/978-3-030-02302-7_16
6. Khider H, Hammoudi S, Meziane A (2020) Business Process Model Recommendation as a Transformation Process in MDE: Conceptualization and First Experiments. In: Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development. SciTePress, pp 65–75
7. Rasmussen JB, Hvam L, Kristjansdottir K, Mortensen NH (2020) Guidelines for Structuring Object-Oriented Product Configuration Models in Standard Configuration Software. Journal of Universal Computer Science 26:374–401
8. Smirnov A, Shchekotov M, Shilov N, Ponomarev A (2018) Decision Support Service Based on Dynamic Resource Network Configuration in Human-Computer Cloud. In: 2018 23rd Conference of Open Innovations Association (FRUCT). IEEE, pp 362–368
9. Pereira JA, Schulze S, Krieter S, et al (2018) A Context-Aware Recommender System for Extended Software Product Line Configurations. In: Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems. ACM, New York, NY, USA, pp 97–104
10. Hildebrandt M, Sunder SS, Mogoreanu S, et al (2019) Configuration of Industrial Automation Solutions Using Multi-relational Recommender Systems. Lecture Notes in Computer Science 11053:271–287. https://doi.org/10.1007/978-3-030-10997-4_17
11. Tarasov V, Seigerroth U, Sandkuhl K (2019) Ontology Development Strategies in Industrial Contexts. Lecture Notes in Business Information Processing 339:156–167. https://doi.org/10.1007/978-3-030-04849-5_14
12. Elkindy AIA (2019) Survey of Business Process Modeling Recommender Systems. University of Koblenz - Landau
13. Vernadat F (2020) Enterprise modelling: Research review and outlook. Computers in Industry 122:103265. https://doi.org/10.1016/j.compind.2020.103265
14. Wang J, Gui S, Cao B (2019) A process recommendation method using bag-of-fragments. International Journal of Intelligent Internet of Things Computing 1:32. https://doi.org/10.1504/IJIITC.2019.104734
15. Melville P, Sindhwani V (2011) Recommender Systems. In: Sammut C, Webb GI (eds) Encyclopedia of Machine Learning. Springer US, Boston, MA, pp 829–838
16. Fellmann M, Metzger D, Jannaber S, et al (2018) Process Modeling Recommender Systems - A Generic Data Model and Its Application to a Smart Glasses-Based Modeling Environment. Business & Information Systems Engineering 60:21–38
17. Koschmider A, Hornung T, Oberweis A (2011) Recommendation-based editor for business process modeling. Data & Knowledge Engineering 70:483–503. https://doi.org/10.1016/j.datak.2011.02.002
18. Kuschke T, Mäder P (2014) Pattern-based auto-completion of UML modeling activities. In: Proceedings of the 29th ACM/IEEE international conference on Automated software engineering. ACM, New York, NY, USA, pp 551–556
19. Wieloch K, Filipowska A, Kaczmarek M (2011) Autocompletion for Business Process Modelling. Lecture Notes in Business Information Processing 97:30–40. https://doi.org/10.1007/978-3-642-25370-6_4
20. Born M, Brelage C, Markovic I, et al (2009) Auto-completion for Executable Business Process Models. Lecture Notes in Business Information Processing 17:510–515. https://doi.org/10.1007/978-3-642-00328-8_51
21. Mazanek S, Minas M (2009) Business Process Models as a Showcase for Syntax-Based Assistance in Diagram Editors. Lecture Notes in Computer Science 5795:322–336. https://doi.org/10.1007/978-3-642-04425-0_24
22. Clever N, Holler J, Shitkova M, Becker J (2013) Towards Auto-Suggested Process

Modeling – Prototypical Development of an Auto-Suggest Component for Process Modeling Tools. In: Enterprise Modelling and Information Systems Architectures (EMISA 2013). Gesellschaft für Informatik e.V., pp 133–145
23. Fellmann M, Zarvić N, Thomas O (2017) Business Processes Modelling Assistance by Recommender Functionalities: A First Evaluation from Potential Users. Lecture Notes in Business Information Processing 295:79–92. https://doi.org/10.1007/978-3-319-64930-6_6
24. Li Y, Cao B, Xu L, et al (2014) An Efficient Recommendation Method for Improving Business Process Modeling. IEEE Transactions on Industrial Informatics 10:502–513. https://doi.org/10.1109/TII.2013.2258677
25. Nair A, Ning X, Hill JH (2021) Using recommender systems to improve proactive modeling. Software and Systems Modeling. https://doi.org/10.1007/s10270-020-00841-2
26. Kögel S (2017) Recommender system for model driven software development. In: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. ACM, New York, NY, USA, pp 1026–1029
27. Jangda A, Polisetty S, Guha A, Serafini M (2021) Accelerating graph sampling for graph machine learning using GPUs. In: Proceedings of the Sixteenth European Conference on Computer Systems. ACM, New York, NY, USA, pp 311–326
28. Valera M, Guo Z, Kelly P, et al (2018) Machine learning for graph-based representations of three-dimensional discrete fracture networks. Computational Geosciences 22:695–710. https://doi.org/10.1007/s10596-018-9720-1
29. Chen C, Ye W, Zuo Y, et al (2019) Graph Networks as a Universal Machine Learning Framework for Molecules and Crystals. Chemistry of Materials 31:3564–3572. https://doi.org/10.1021/acs.chemmater.9b01294
30. Na GS, Chang H, Kim HW (2020) Machine-guided representation for accurate graph-based molecular machine learning. Physical Chemistry Chemical Physics 22:18526–18535. https://doi.org/10.1039/D0CP02709J
31. Nielsen RF, Nazemzadeh N, Sillesen LW, et al (2020) Hybrid machine learning assisted modelling framework for particle processes. Computers & Chemical Engineering 140:106916. https://doi.org/10.1016/j.compchemeng.2020.106916
32. Wu Z, Pan S, Chen F, et al (2021) A Comprehensive Survey on Graph Neural Networks. IEEE Transactions on Neural Networks and Learning Systems 32:4–24. https://doi.org/10.1109/TNNLS.2020.2978386
33. Wang M, Qiu L, Wang X (2021) A Survey on Knowledge Graph Embeddings for Link Prediction. Symmetry 13:485. https://doi.org/10.3390/sym13030485
34. Mikolov T, Sutskever I, Chen K, et al (2013) Distributed Representations of Words and Phrases and their Compositionality
35. Bordes A, Usunier N, Garcia-Duran A, et al (2013) Translating Embeddings for Modeling Multi-relational Dat. Advances in Neural Information Processing Systems (NIPS 2013) 26:
36. Wang Z, Zhang J, Feng J, Chen Z (2014) Knowledge graph embedding by translating on hyperplanes. In: AAAI'14: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence. pp 1112–1119
37. Fan M, Zhou Q, Chang E, Zheng TF (2014) Transition-based Knowledge Graph Embedding with Relational Mapping Properties. In: Proceedings of the 28th Pacific Asia Conference on Language, Information and Computing. Department of Linguistics, Chulalongkorn University, pp 328–337
38. Lin Y, Liu1 Z, Sun M, et al (2015) Learning Entity and Relation Embeddings for Knowledge Graph Completion. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. pp 2181–2187
39. Yang B, Yih W, He X, et al (2014) Embedding Entities and Relations for Learning and

Inference in Knowledge Bases
40. Trouillon T, Welbl J, Riedel S, et al (2016) Complex Embeddings for Simple Link Prediction
41. Dettmers T, Minervini P, Stenetorp P, Riedel S (2017) Convolutional 2D Knowledge Graph Embeddings
42. Balažević I, Allen C, Hospedales TM (2019) Hypernetwork Knowledge Graph Embeddings. Lecture Notes in Computer Science 11731:553–565. https://doi.org/10.1007/978-3-030-30493-5_52
43. Jagvaral B, Lee W-K, Roh J-S, et al (2020) Path-based reasoning approach for knowledge graph completion using CNN-BiLSTM with attention mechanism. Expert Systems with Applications 142:112960. https://doi.org/10.1016/j.eswa.2019.112960
44. Vashishth S, Sanyal S, Nitin V, Talukdar P (2019) Composition-based Multi-Relational Graph Convolutional Networks
45. Sonntag A, Hake P, Fettke P, Loos P (2016) An Approach for Semantic Business Process Model Matching Using Supervised Machine Learning. In: European Conference on Information Systems (ECIS)
46. Hamilton WL, Ying R, Leskovec J (2017) Inductive Representation Learning on Large Graphs
47. Mikolov T, Chen K, Corrado G, Dean J (2013) Efficient Estimation of Word Representations in Vector Space. In: Proceedings of the International Conference on Learning Representations (ICLR 2013)