Enterprise Modelling Assistance: Edge Prediction Improvement Using Textual Information

Walaa Othman
ITMO University
St.Petersburg, Russia
walaa_othman@itmo.ru

Nikolay Shilov
SPC RAS
St.Petersburg, Russia
nick@iias.spb.su

***Abstract*—Today, enterprise modelling is still a highly manual task. There are exist some assistance techniques but they are mostly limited to pattern libraries and pre-defined rules, which limits their functionality and makes them non-flexible. Application of machine learning techniques to support enterprise modelers is a promising approach. However, one of the main problems in this area today is the absence of model repositories that could be used for training what causes the necessity to train machine learning models on small data. In this paper we study which textual information from the model and how can be used to increase the efficiency of the edge prediction task, which is one of the key tasks in graph-structured problems like enterprise modelling. The comparative analysis shows that application of FastText method provides a better result for node names embedding, and consideration of node names and descriptions significantly increases the edge prediction quality. The built model has been successfully validated on a test case scenario simulating the enterprise model building process.**

I. Introduction

Enterprise modelling is currently of high demand due to increasing progress pace and intensive penetration of information technologies into various business processes [1]. These processes cause continuous changes in companies going along with such trends as digitalization (integration of digital technologies into everyday life [2]), introduction of smart product-service systems (intelligent products and services integrated into through information and communication technologies [3]), quantified products (products capable of collecting and analyzing various usage data during operation, not only on the level of a single product instance but for a complete fleet, which allows for new data services and service ecosystems [4]), etc. All these trends require companies to adapt and change their architectures and business processes. As a result, various enterprise models have to be designed quite often. However, enterprise modelling is still a highly manual process that requires the modeler to analyze multiple existing models to evaluate the efficiency of solutions [5].

There have been developed different techniques to support the modeler, such as pattern libraries, automated model syntax checking, autocompletion, domain dictionaries, and others. However, currently all of them provide very limited level of automation and are based on pre-defined libraries or rule bases. In the previous publication [6] the authors of this paper proposed application of machine learning techniques to support enterprise modelers at various stages of model building. The research was aimed at studying enterprise modelling assistance possibilities that could be provided by graph neural networks. Conducted experiments have demonstrated that such possibilities are promising. However, one of the main problems today is the absence of repositories of multiple enterprise models of decent quality so it was necessary to collect own training dataset consisting of about 100 models and experiment with small data.

One of the tasks to be solved with the use of graph neural networks outlined in the paper was edge prediction (the task of predicting whether two nodes in a graph are likely to have a link or not). From the enterprise modelling point of view, this task could be applied to such modelling automation tasks as:

- suggestion of possibly existing connections (the modeler adds a new node to the model, and the modelling environment should suggest its connections to other existing nodes of the model);

- identification of likely wrong edges (the modeler adds a wrong connection by mistake, and the modelling environment should identify the problem).

Due to the limited volume of training data, it is important to consider as much information as possible. Textual node information (name, type and description) can be considered as such additional information. Thus, in this paper we report the research on how to use the textual node information in a deep neural network model to solve the edge prediction task in a more efficient way. As a result, the following research questions were formulated:

1. What would be a more efficient embedding method to capture the semantic information from the nodes' names in order to achieve better edge prediction?

2. What textual node attributes should be used to achieve better prediction results?

The rest of the paper is structured as follows. The related work is presented in section 2, Section 3 introduces the research approach including the used dataset and the method used for edge prediction. Experimental evaluation and results

discussion are presented in Section 4. Finally, the conclusions are drawn in section 5.

II. Related work

A. Edge prediction

Existing methods for edge prediction can be classified into three categories: path-based methods, embedding methods, and graph neural networks.

The path-based methods predict the edges by computing the similarity between two nodes based on the weighted count of paths (personalized PageRank [7]) or the length of the shortest path (graph distance [8]). More recent path-based methods like Path-RCNN [9] and PathCon [10] encode each path using recurrent neural networks and aggregate paths for predicting the edge. These methods are limited to short paths that consist of less than four edges.

The embedding methods learn the distributed representation of the nodes and edges in the graph. Some of these methods are applied to homogeneous graphs (e.g., Deepwalk [11] learns the representation by modelling a stream of short random walks). These representations are latent features that capture the neighborhood similarity. Others are applied to knowledge graphs like TransE [12] or RotatE [13]. Further research efforts in this field try to improve the embedding by introducing new score functions [14], [15] to capture the semantic patterns of the relations. However, these methods are not capable of encoding subgraphs between node pairs.

The graph neural networks (GNN) is a type of neural networks designed for learning over graphs. The GNN usually consists of graph convolutional layers and graph aggregation layers. The convolutional layers aim to extract the local substructure features for nodes, while the aggregation layers aggregate the node-level features into graph-level feature vector. One of the most popular methods for GNN is Graph convolutional networks (GCN) [16], The GCN generalize the convolution operation from euclidean data to graphs by using the Fourier basis of the given graph. GraphSAGE [17] samples a fixed-sized of neighborhood nodes and then aggregates neighbor information using different strategies. DGCNN [18] proposes a layer to sort the graph vertices in a consistent order (Sort-Pooling) and then uses a traditional 1-D convolutional neural network. For edge prediction, The authors of paper [19] apply the DGCNN to learn link representation for link prediction after labeling the nodes according to their distances to the source and the the target nodes. the authors of paper [20] use a similar trick as [19] by labeling the nodes based on the shortest distances to target nodes.

B. Word embedding

Word embeddings are a real-valued representation vectors that encode the meaning of the words, where words with similar meaning are also close in the vector space. Representing the textual data in a mathematical form makes it possible for the computers to handle and deal with these data. The word embedding method can be classified into two categories: (1) static models such as GLOVE [21], Word2Vec [22], and FastText [23] don't capture the contextual meaning (in other words, they map the words into vectors ignoring the fact that the same string of letters may have different meanings); (2) contextual embedding models such as BERT [24] embed the the contextual information into the word representations as well.

C. Sentence embedding

Sentence embedding techniques represent the entire sentences as vectors capturing their semantic meaning. This helps the computers to understand the intention and the context in the entire text. Doc2Vec [25] is one of the common techniques and it is considered as an extension to the Word2Vec [22]. It uses the Word2Vec model and adds on it by introducing another vector (Paragraph ID). There are 2 ways to add the paragraph vector to the model: (1) the distributed Memory version of Paragraph vector and (2) the bag of words version. SentenceBERT [26] uses a Siamese network-like architecture [27] to provide 2 sentences as an input. These 2 sentences are then passed to BERT models and a pooling layer to generate their embeddings. Then the embeddings are used for the pair of sentences as inputs to calculate the cosine similarity.

InferSent [28] like SentenceBERT takes a pair of sentences and encodes them to generate the actual sentence embeddings. Then, it extracts the relations between these embeddings using concatenation, element-wise product and absolute element-wise difference.

Universal Sentence Encoder [29] is one of the most well-performing sentence embedding techniques that can be used for Multi-task learning. This encoder is based on two encoder models: Transformer and Deep Averaging Network (DAN). Both models can generate a sentence embedding by lower-casing the sentence, tokenizing it, and converting it into 512-dimensional vector. The DAN computes the unigram and bi-gram embeddings and averages them to get a single embedding that is then passed to a deep neural network to get the final embedding.

The authors of paper [30] proposed an effective knowledge distillation method to compress large transformer-based models by training a student model via mimicking the teacher's self-attention modules. In addition, they proposed to use the self-attention distributions and value relation of the teacher's last transformer layer to guide the training of the student. The resulted student models (such as all-MiniLM-L6-v2) were smaller but achieved high accuracy.

In this paper, we use the pretrained all-MiniLM-L6-v2 model, trained using the XLM-RBase [31] as the teacher. The main reason to choose this model is that it has a high encoding speed and good mean accuracy of 68.06% over 14 different datasets.

III. Research approach

A. Dataset

The used dataset consists of 112 models with total number of edges equal to 3259 and total number of nodes equal to 3058 (see [6] for details). The models in the dataset belong to 8 different classes: (1) Business Process Model, (2) Actors and Resources Model, (3) 4EM General Model, (4) Concepts Model, (5) Technical Components and Requirements Model, (6) Product-Service-Model, (7) Goal Model, and (8) Business Rule Model. Fig.1 shows the data distribution according to the model classes.

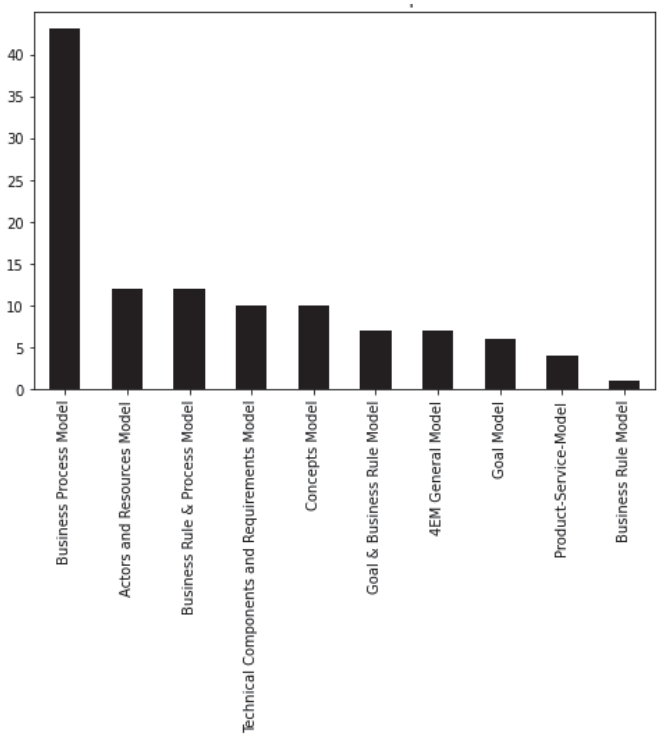

Fig. 1. The distribution of the data according to the model classes

Each node has three textual attributes: name, class name, and description. The description is a sentence that explains the functionality of the node, while the class name is the category of the node and can be one of the following 23 classes: (1) Relation, (2) Attribute, (3) Cause, (4) Comment, (5) Component, (6) Concept, (7) Constraint, (8) Development Action, (9) External Process, (10) Goal, (11) Feature, (12) Organizational Unit, (13) IS Requirement, (14) IS Technical Component, (15) Individual, (16) Information Set, (17) Role, (18) Resource, (19) Rule, (20) Process, (21) Problem, (22) Unspecific/Product/Service, and (23) Opportunity. Fig.2 shows the data distribution according to the nodes class name.

The edges connect the models' nodes and have two attributes: the class name, and the description. The class name describes the category of the edge. In the used dataset all the edges belong to class 4EM_Relation. The edge description is a

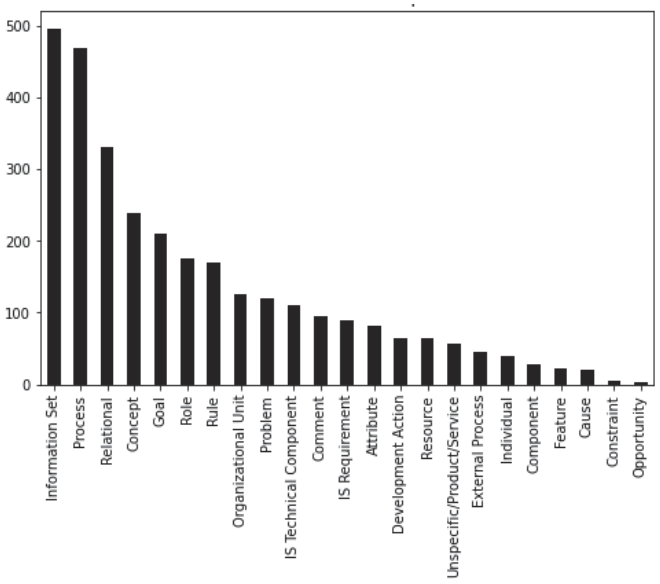

Fig. 2. The distribution of the data based on the nodes class name

sentence that describes the relationship between connected nodes. Only 223 out of 3259 edges in the dataset have a description. So they were excluded from the consideration. Fig.3 shows an example of a model from the dataset that belongs to the class "Business process Model".

B. Methodology

The edge prediction task is aimed to predict the connection between two nodes in an enterprise model. We used graph neural network to solve this task. Its architecture is presented in Fig. 4. The model consists of three GCNConv (graph convolution) layers. It takes the graph represented by the nodes and edges as the input and returns the probability of two nodes to be connected by an edge. As it was mentioned in the Introduction, we investigate two questions: (1) finding the best embedding method to use for capturing the semantic information from the nodes' names in order to achieve better edge prediction (for that purpose, we looked into two methods: the word2vec method and the FastText method); (2) finding the best nodes attributes to use for achieving a better prediction results.

The all-MiniLM-L6-v2 transformer [30] was used for embedding the nodes description attrribute.

1) Word2Vec embedding method: Word2Vec [22] is one of the most popular techniques to learn word embedding using shallow neural network. It can be obtained using two methods: Common Bag Of Words (CBOW) and Skip Gram. The CBOW method takes the context of each word in the neighborhood of the target word as the input and tries to predict the word corresponding to the context. The neighborhood is a window of predefined s ize s urrounding t he t arget w ord. O n t he other hand, the Skip Gram predicts the context for the given word: the target word is fed into the network and the Skip Gram

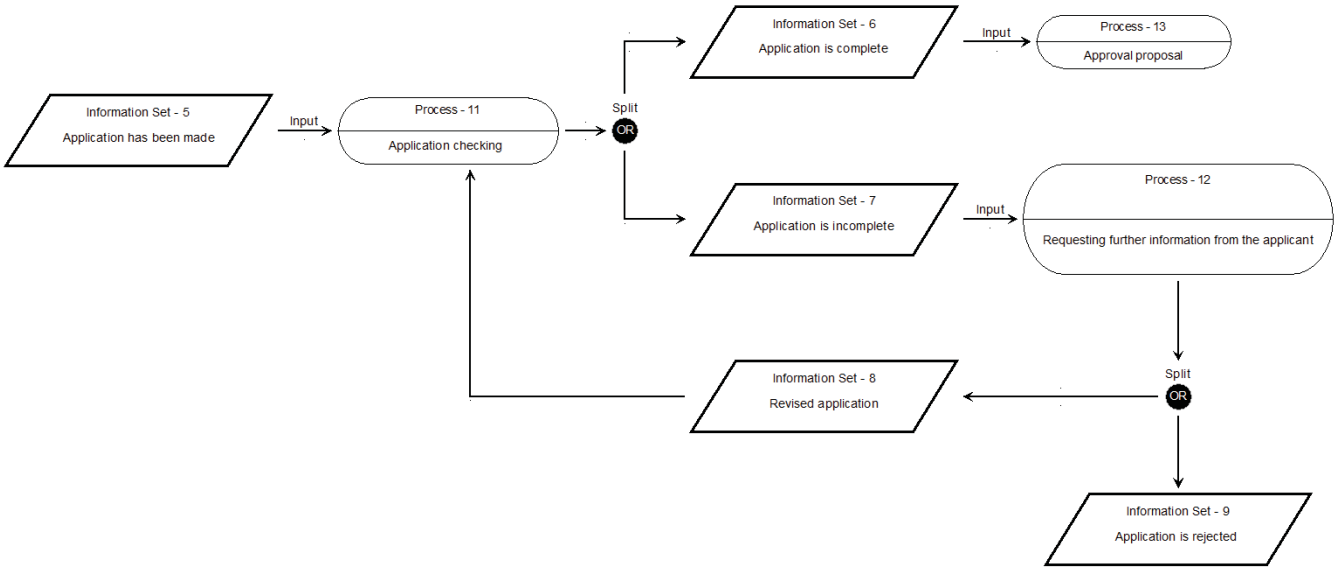

Fig. 3. A sample model from the dataset

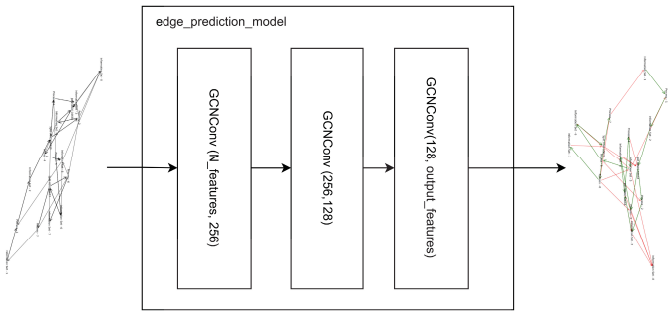

Fig. 4. The model architecture used for edge prediction

predict its location in the sentence. While the first method is faster and has better representations for more frequent words, the latter works better with small amount of data and is capable of representing rare words.

2) FastText embedding method: FastText [23] is a method for word embedding, which depends on breaking the words into several sub-words (n-grams) instead of feeding them directly to the neural network. In other words, the representations are learnt for the n-gram characters, and the words represented as the sum of the n-gram vectors. After training the neural network, one gets word embeddings for each of the n-grams. This helps in representing rare words as their n-grams are more likely appear in n-grams for other words. It also helps to understand the suffixes and prefixes.

IV. Experiments and Evaluation

In the conducted experiments, the dataset was split into 98 graphs for training and 14 graphs for testing. No data augmentation were performed. One suggestion for data augmentation was to use the graph truncation approach to generate more data. However, such an approach can lead to an over-fitting, because the features fed to the model are textual. In other words, the same textual information (nodes' names and nodes' descriptions) will appear in the training dataset several times (in the original sample and in all truncated samples), and this will led to over-fitting on these data. To answer the first research question we conducted three experiments using the model shown in Fig.4 with Adam optimizer, learning rate 1e-4, batch size 16, and the binary cross entropy(BCE) as a loss function. We chose the BCE because the edge prediction problem is a classification problem (one needs to classify whether the edge exists (True) or not (False)). In the experiments, the input to the model was the graph topology and the embedded nodes' names. In the first experiment, the nodes' names were embedded using the Word2Vec skip-gram method, while in the second experiment the used embedding method was FastText skip-gram, and in the last experiment we used the pre-trained all-MiniLM-L6-v2 sentence embedding model to embed the nodes' names. The word2vec and the FastText were trained on all the words that appears in the dataset including the words in the nodes' descriptions, names, class names, and the edges' descriptions. The reason we chose the skip-gram model for both FastText and Word2Vec is that it works well with a small amount of the training data such as our case, and it has well representation for rare words.

Table I shows the testing performance represented by the area under the receiver operating characteristic curve (ROC AUC) for Entriprise modelling edge prediction task based on the graph topology and the nodes' names embedded using the Word2Vec, FastText and the all-MiniLM-L6-v2 methods. The all-MiniLM-L6-v2 embedding produced better results compared to the other classical word embedding methods (just slightly better than FastText). However, this pre-trained model

embeds each word into a vector of 384 dimensions, while the other two were trained to embed the word to a vector of 9 dimensions, which is more efficient, thus in the further experiments, we will go with the FastText embedding method. The vector length affects the model size. The higher the vector length, the higher computational power required. we chose the length to be 9 because it was the minimum length that achieves good results. In addition, for fastText, the regular unigram word handling was used, thus the word wasn't broken into parts, but used as it is.

TABLE I. COMPARISION BETWEEN WORD2VEC, FASTTEXT, AND ALL-MINILM-L6-V2 EMBEDDING METHODS

The used embedding method	ROC AUC
Word2Vec	0.8592
FastText	**0.9196**
all-MiniLM-L6-v2	**0.9218**

To answer the second research question we conducted eight experiments using the neural network model shown in Fig.4 with Adam optimizer, learning rate 1e-4, batch size 16, and BCE as a loss function.

For each experiment, different features were used. While in the first experiment, we only used the graph topology, by that we mean the graph represented by its nodes and the connections between these nodes (the edges), without taking into consideration any other attributes. In the rest of experiments, we used in addition to the graph topolgy, different nodes' attributes. For node name, the FastText method were used for embedding as it showed better results compared to the Word2Vec. While the all-MiniLM-L6-v2 sentence embedding [32] was used for embedding the nodes' description into a vector of 384 dimensions. For the nodes' class name, a simple label encoding was used.

Table. II shows the features used in each experiments in addition to the testing ROC AUC. It can be seen that node name and node description turn out to be the most valuable textual information for edge prediction, whereas node class name doesn't help and even introduces some disturbance.

Besides the model evaluation above an additional experiment involving a domain expert has been carried out in order to analyse if the built model can be potentially useful for an enterprise modeler. For evaluation of the built GNN model, a test case model shown in Fig.5 was chosen. It is introduced by [33], is not related to the training set, and has never been "seen" by the GNN model. The evaluation was aimed at simulation of the modeler activity (step by step enterprise model building). For this purpose we generated several "partial" models that illustrate the test case model at different building stages with a missing edge. The pre-trained GNN model shown in Fig.4 with features (graph topology, nodes' names, and nodes' descriptions) was used to predict to which of the test case model nodes the newly added (not connected) node should be connected. For this purpose, all the possible edges between the existing nodes and the newly added node are generated, the GNN predicts the probability for each edge, and the edge with the highest probability is suggested to the user.

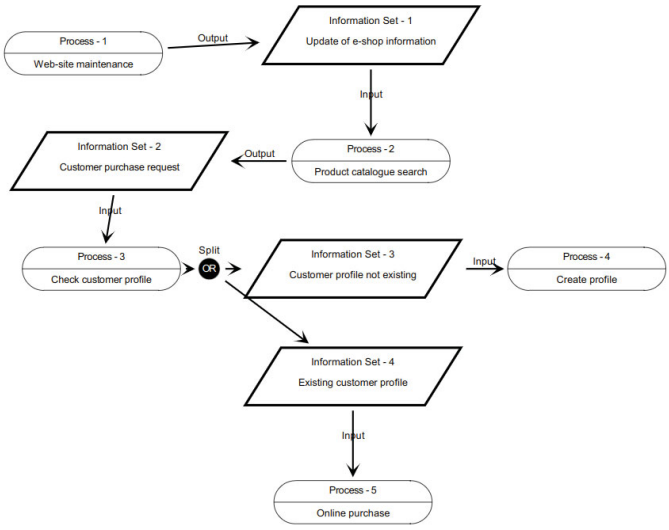

Fig. 5. The test case model

Fig. 6 - Fig. 10 show the partial models fed to the GNN with missing edges illustrated by dashed lines. Table III - Table VII show the probabilities of the edges between the model nodes and the newly added node.

One can see, that the edge predictions for partial models 1, 2, and 4 the predictions were correct. For partial model 3 the prediction was wrong, however, the correct edge also has a high probability (0.859). In the implementation this situation can be processed as top n suggestions, or several suggestions with the probability above a certain threshold. For partial model 5 the result is also wrong, and the correct edge again has the second highest probability. The domain expert confirmed that provided recommendations can indeed facilitate the model development process.

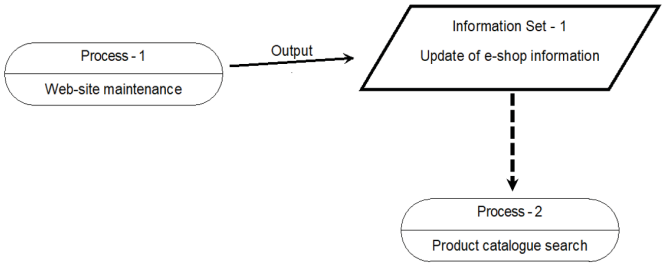

Fig. 6. Partial model 1 with a missing edge from Information Set-1 to Process-2

TABLE II. CONDUCTED EXPERIMENTS TO CHOOSE THE BEST FEATURES FOR EDGE PREDICTION TASK

Experiment	Used features	number of features	ROC AUC
1	graph topology	0	0.5463
2	graph topology + node class name	1	0.6238
3	graph topology + node name	9	0.9196
4	graph topology + node name+ node class name	10	0.6299
5	graph topology + node description	384	0.9619
6	graph topology+ node class name+ node description	385	0.7643
7	**graph topology + node name+ node description**	**393**	**0.9851**
8	graph topology+ node name+ node class name+ node description	394	0.9747

TABLE III. THE RESULTS OF THE GNN FOR PREDICTING THE MISSING EDGE OF PARTIAL MODEL 1 (FIG.6)

source node	target node	Probability	Suggestion
Information Set - 1	Process - 2	0.4828	**
Process - 1	Process - 2	0.4776	

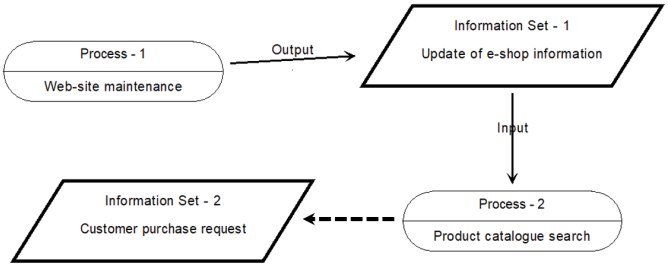

Fig. 7. Partial model 2 with a missing edge from Process-2 to Information set-2

TABLE IV. THE RESULTS OF THE GNN FOR PREDICTING THE MISSING EDGE OF PARTIAL MODEL 2 (FIG.7)

source node	target node	Probability	Suggestion
Information Set - 1	Information Set - 2	0.4545	
Process - 1	Information Set - 2	0.4766	
Process - 2	Information Set - 2	0.4828	**

TABLE V. THE RESULTS OF THE GNN FOR PREDICTING THE MISSING EDGE OF PARTIAL MODEL 3 (FIG.8)

source node	target node	Probability	Suggestion
Information Set - 1	Information Set - 3	0.4682	
Information Set - 2	Information Set - 3	0.6827	
Process - 1	Information Set - 3	0.4550	
Process - 2	Information Set - 3	0.4727	
Process - 3	Information Set - 3	0.9244	**
Split (OR)-15645	Information Set - 3	0.8590	

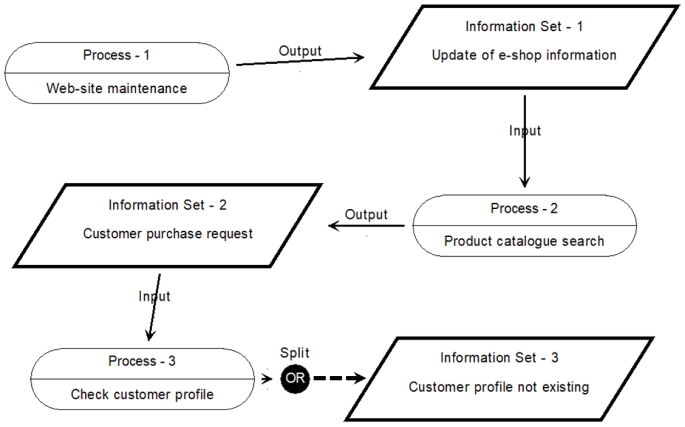

Fig. 8. Partial model 3 with a missing edge from the Split node to Information set-3

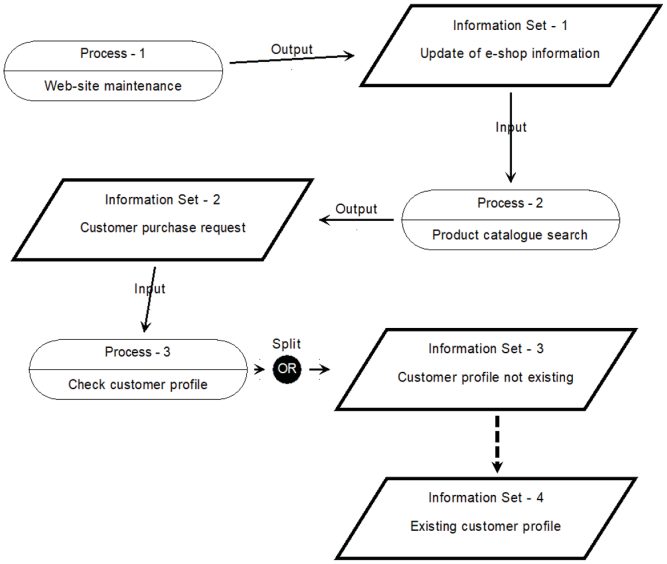

Fig. 9. Partial model 4 with a missing edge from the Split node to Information set-4

V. CONCLUSION

The paper analyzes how solving the edge prediction task for enterprise models using GNN can be improved in the conditions of training on small data. We answer the following two research questions: (1) what would be a more efficient embedding method to capture the semantic information from the nodes' names in order to achieve better edge prediction, and (2) what textual node attributes should be used to achieve better prediction results. The experiments showed that application of FastText method provides better results for

TABLE VI. THE RESULTS OF THE GNN FOR PREDICTING THE MISSING EDGE OF PARTIAL MODEL 4 (FIG.9)

source node	target node	Probability	Suggestion
Information Set - 1	Information Set - 4	0.4822	
Information Set - 2	Information Set - 4	0.5373	
Information Set - 3	Information Set - 4	0.4793	
Process - 1	Information Set - 4	0.6076	
Process - 2	Information Set - 4	0.6138	
Process - 3	Information Set - 4	0.4940	
Split (OR)-15645	Information Set - 4	0.6407	**

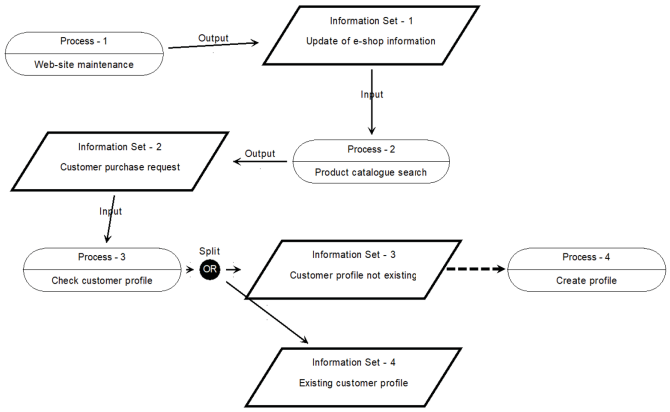

Fig. 10. Partial model 5 with a missing edge from Information Set-3 to Process-4

TABLE VII. THE RESULTS OF THE GNN FOR PREDICTING THE MISSING EDGE OF PARTIAL MODEL 5 (FIG.10)

source node	target node	Probability	Suggestion
Information Set - 1	Process - 4	0.5590	**
Information Set - 2	Process - 4	0.5142	
Information Set - 3	Process - 4	0.5568	
Information Set - 4	Process - 4	0.5520	
Process - 1	Process - 4	0.5310	
Process - 2	Process - 4	0.3950	
Process - 3	Process - 4	0.4286	
Split (OR)-15645	Process - 4	0.3871	

node names embedding than application of Word2Vec. For the second research question it was found that node name and node description turned out to be the most valuable textual information for edge prediction, whereas node class name doesn't help and even introduces some disturbance. To evaluate the assistance potential of the built GNN model, an experiment simulating the modeler activity has been carried out. Out of five cases, three edges were predicted correctly, and for the other two cases the correct edges had the second highest probability, what can be interpreted as a successful result.

Future work will be aimed to experimenting with other graph tasks, as well as developing a modeler assistance tool incorporating developed machine learning models.

ACKNOWLEDGMENT

The research is funded by the Russian Science Foundation (project # 22-21-00790). The authors thank colleagues from the University of Rostock for provided training data and result evaluation assistance.

REFERENCES

[1] A. Zimmermann, R. Schmidt, K. Sandkuhl, D. Jugel, C. Schweda, and J. Bogner, *Architecting Digital Products and Services*, 01 2021, pp. 181–197.

[2] M. Mondejar, R. Avtar, H. Baños, R. Dubey, J. Esteban, A. Gómez-Morales, B. Hallam, N. Mbungu, C. Okolo, A. Kumar, Q. She, and S. Garcia-Segura, "Digitalization to achieve sustainable development goals: Steps towards a smart green planet," *Science of The Total Environment*, vol. 794, p. 148539, 06 2021.

[3] W. Song, Z. Niu, and P. Zheng, "Design concept evaluation of smart product-service systems considering sustainability: An integrated method," *Computers Industrial Engineering*, vol. 159, 06 2021.

[4] K. Sandkuhl, N. Shilov, U. Seigerroth, and A. Smirnov, "Towards the quantified product-product lifecycle support by multi-aspect ontologies," *IFAC-PapersOnLine*, vol. 55, no. 2, pp. 187–192, 2022, 14th IFAC Workshop on Intelligent Manufacturing Systems IMS 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896322001926

[5] M. Fellmann, D. Metzger, S. Jannaber, N. Zarvic, and O. Thomas, "Process modeling recommender systems - a generic data model and its application to a smart glasses-based modeling environment," *Business Information Systems Engineering*, vol. 60, no. 1, 2018.

[6] N. Shilov, W. Othman, M. Fellmann, and K. Sandkuhl, "Machine learning-based enterprise modeling assistance : Approach and potentials," in *The Practice of Enterprise Modeling : 14th IFIP WG 8.1 Working Conference, PoEM 2021, Riga, Latvia, November 24-26, 2021, Proceedings*, ser. Lecture Notes in Business Information Processing, vol. 432, no. 432, 2021, pp. 19–33.

[7] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Technical Report 1999-66, November 1999, previous number = SIDL-WP-1999-0120. [Online]. Available: http://ilpubs.stanford.edu:8090/422/

[8] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.

[9] A. Neelakantan, B. Roth, and A. McCallum, "Compositional vector space models for knowledge base completion," *CoRR*, vol. abs/1504.06662, 2015. [Online]. Available: http://arxiv.org/abs/1504.06662

[10] H. Wang, H. Ren, and J. Leskovec, "Entity context and relational paths for knowledge graph completion," *CoRR*, vol. abs/2002.06757, 2020. [Online]. Available: https://arxiv.org/abs/2002.06757

[11] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[12] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Advances in Neural Information Processing Systems*, C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, Eds., vol. 26. Curran Associates, Inc., 2013. [Online]. Available: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf

[13] Z. Sun, Z. Deng, J. Nie, and J. Tang, "Rotate: Knowledge graph embedding by relational rotation in complex space," *CoRR*, vol. abs/1902.10197, 2019. [Online]. Available: http://arxiv.org/abs/1902.10197

[14] Y. Tang, J. Huang, G. Wang, X. He, and B. Zhou, "Orthogonal relation transforms with graph context modeling for knowledge graph embedding," *CoRR*, vol. abs/1911.04910, 2019. [Online]. Available: http://arxiv.org/abs/1911.04910

[15] Z. Zhang, J. Cai, Y. Zhang, and J. Wang, "Learning hierarchy-aware knowledge graph embeddings for link prediction," *CoRR*, vol. abs/1911.09419, 2019. [Online]. Available: http://arxiv.org/abs/1911.09419

[16] J. Bruna, W. Zaremba, A. Szlam, and Y. Lecun, "Spectral networks and locally connected networks on graphs," 12 2013.

[17] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *CoRR*, vol. abs/1706.02216, 2017. [Online]. Available: http://arxiv.org/abs/1706.02216

[18] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *CoRR*, vol. abs/1801.07829, 2018. [Online]. Available: http://arxiv.org/abs/1801.07829

[19] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," *CoRR*, vol. abs/1802.09691, 2018. [Online]. Available: http://arxiv.org/abs/1802.09691

[20] P. Li, Y. Wang, H. Wang, and J. Leskovec, "Distance encoding - design provably more powerful graph neural networks for structural representation learning," *CoRR*, vol. abs/2009.00142, 2020. [Online]. Available: https://arxiv.org/abs/2009.00142

[21] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," vol. 14, 01 2014, pp. 1532–1543.

[22] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

[23] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *CoRR*, vol. abs/1607.04606, 2016. [Online]. Available: http://arxiv.org/abs/1607.04606

[30] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," *CoRR*, vol. abs/2002.10957, 2020. [Online]. Available: https://arxiv.org/abs/2002.10957

[24] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[25] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," *31st International Conference on Machine Learning, ICML 2014*, vol. 4, 05 2014.

[26] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *CoRR*, vol. abs/1908.10084, 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[27] G. Koch, R. Zemel, R. Salakhutdinov *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, vol. 2. Lille, 2015, p. 0.

[28] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, September 2017, pp. 670–680. [Online]. Available: https://www.aclweb.org/anthology/D17-1070

[29] D. Cer, Y. Yang, S. Kong, N. Hua, N. Limtiaco, R. S. John, N. Constant, M. Guajardo-Cespedes, S. Yuan, C. Tar, Y. Sung, B. Strope, and R. Kurzweil, "Universal sentence encoder," *CoRR*, vol. abs/1803.11175, 2018. [Online]. Available: http://arxiv.org/abs/1803.11175

[31] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzm, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *CoRR*, vol. abs/1911.02116, 2019. [Online]. Available: http://arxiv.org/abs/1911.02116

[32] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov, "Unsupervised cross-lingual representation learning at scale," *CoRR*, vol. abs/1911.02116, 2019. [Online]. Available: http://arxiv.org/abs/1911.02116

[33] J. Stirna, B. Lantow, and D. Bork, "Modeling method conceptualization within omilab: The 4em case," in *POEM*, 2017.