Graph Embeddings-based Link Prediction for Bridge Maintenance Knowledge

Guanyu Xiong[1], Yan Gao[1], Haijiang Li*

BIM for Smart Engineering Centre, School of Engineering, Cardiff University, United Kingdom

[1]The authors contributed equally to this work.

xiongg@cardiff.ac.uk

Abstract. Infrastructure owners and operators around the world are facing escalated maintenance requirements of large number of bridge assets. The bridge maintenance decision-making requires reviewing numerous bridge inspection data while these data sets are fragmented in different systems or formats, and their value in bridge management are not fully explored. This study presents a solution for bridge defect data storage in graph database supported by graph-based bridge element model and maintenance method recommendation informed by historical data using binary classification-based link prediction model. The supervised machine learning pipeline in Neo4j is utilised and features are engineered with GraphSAGE to suit the bridge engineering practices. The solution is validated via real-world bridge maintenance dataset, demonstrating that the proposed method is capable of recommending potential repair methods based on historical maintenance decision-making records.

1 Introduction

Decision-making in bridge maintenance requires the thorough review of extensive bridge inspection data, combining with specialised domain knowledge of bridge practitioners. The process can be labour extensive and cognitively demanding, while less rewarding due to its repetitive nature. So there exists both demand and potential to automate the process with software tools and machine learning.

To represent the domain knowledge of bridge practitioners, some efforts pivoted in ontology for knowledge representation (Zangeneh and McCabe, 2020) and extraction of links among knowledge entities (Xu *et al.*, 2023). While the ontology-based knowledge graph (KG) frameworks are not designed for effective integration of large number of inspection data and subsequential data-driven machine learning tasks. Meanwhile, Neo4j as a graph database management software tool, has been utilised to store bridge inspection data and fabricate bridge management systems on the basis of bridge maintenance KG. Two recent studies using Neo4j are maintenance cost estimation (Lee and Chi, 2023) and data mining of bridge networks and environmental information (Lin, 2020), but both lack in discussing capturing features from the dataset and learning a model from extracted features.

Notably, a study with similar vision presented by Tiwary, Patro and Sahoo (2022) leveraged DeepWalk (Perozzi, Al-Rfou and Skiena, 2014) embedding technique for variables and relative features on the high-level summarisation of bridge structural performance. This approach can be effective for relatively static database where the nodes and their attributes have been fixed.

For a dynamic and evolving graph database, the transudative learning algorithm exposes shortcoming as it needs re-run when new nodes are added to obtain embeddings of all nodes. Therefore, the potentials and practicability of building a bridge maintenance database with connected machine learning pipelines has not been fully investigated.

While the scenario of combing graph-based modeling of bridge and bridge inspection data has not been much explored. Hence, this study proposes a schema for integrating bridge graph-based modeling, bridge defect data storage and link prediction from defects instances to repair methods. Neo4j is selected as the graph database implementation tool as it supports various graph analytics in the built-in machine learning pipelines.

2 Formation of Graph-based Bridge Element Model

The conceptual development of bridge maintenance KG is a combination of industrial practices and functions of knowledge graph. The Network Rail standards NR/L2/CIV/035 (Network Rail, 2022a) and NR/L3/CIV/006 (Network Rail, 2022b) together describe a methodology for bridge structural assessment - each bridge is composed of hierarchical levels of elements (e.g. major elements and minor elements) which are assessed individually and then the repair method is determined respectively according to the inspection results. This engineering practice motivates the building of an element-wise bridge KG with incorporated defect and repair information.

Various types of KGs are available for this task. A hierarchical KG present components in hierarchical relationships while a labelled property KG can be used to query basic information, such as latest maintenance log and corresponding status, stored in each node (Xia *et al.*, 2022). Hence, a combined hierarchical and property KG is considered as an ideal solution to represent the bridge structure model and stored bridge inspection data.

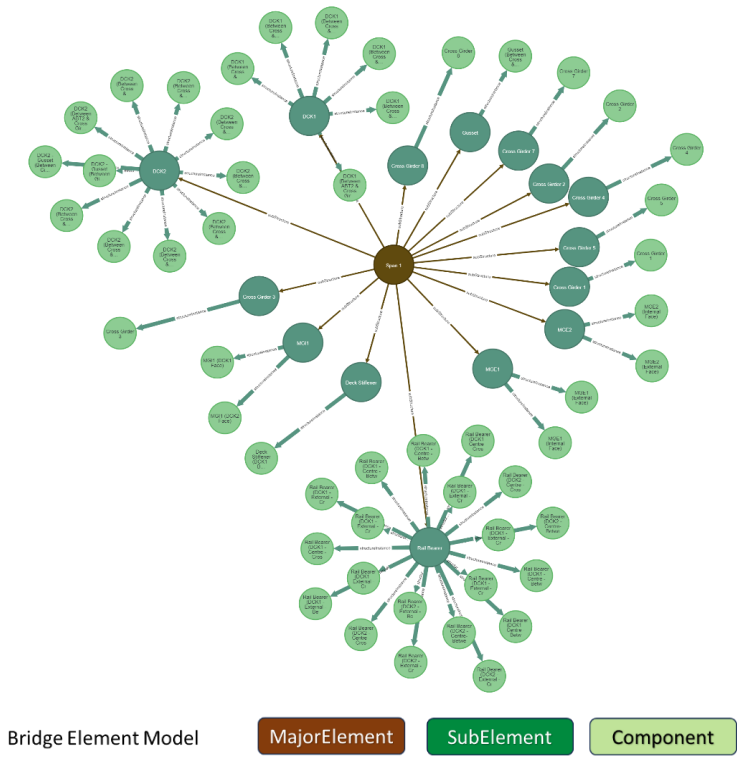

Figure 1: Implementation of Graph-based Bridge Element Model in Neo4j

Several options exist for incorporating defects information on the graph-based bridge element model:

- As descriptions in the attribute of the bridge element node where the defect is identified.
- Each defect type as a node while every defect instance as a link/edge between the element node and the defect node.
- Each defect instance as a node connecting to the bridge element node where the defect is identified.

The first option introduces negligible featuers into the graph as neither nodes/edges nor the layout of the graph is affected. The second and the third options, based on the two graph modeling approaches - TimeTree Model (Graphaware, 2014) and Entity-State Model (Neo4j, 2017), consider defects as states of the bridge element. The two options can largely manipulate graph features so that graph learning can be enabled. Unlike a trouble-shooting proposed KG proposed by Xia *et al.* (2023) where a specific type of defect is indicated as one node, in this study the KG is implemented as graph database to store defect information, so the third option is considered optimal to encompass the benefits of the other two alternatives. As the final step of the graph creation, repair techniques are added as a separate layer of nodes in the graph and each of them is linked to the corresponding node of defect instance.

To digest the conceptual design explained above, the desired graph shall be partitioned into three layers: 1) bridge structure layer which represents graph-based bridge model; 2) defect information layer describing the type, extent and severity of the defect and 3) maintenance layer which reflects repair methods. With the bridge maintenance dataset (samples shown in Table 1), where there are 178 'Defect' nodes and 21 'Proposal' nodes, a Graph-based and Maintenance-oriented Bridge Element Model is created in Neo4j as per the illustration in Figure 2.

Table 1: Samples of Bridge Maintenance Dataset

Major Element	**Sub Element**	**Component**	**Location**	**Defect**	**Proposal**
Span 1	Main Girder 2	MGE2 (External Face)	Stiffener Crank (MGE2 - EF)	Section Loss - Over 50%	Cut out corroded section and weld a new plate in its place like for like.
Span 1	Cross Girder 1	Cross Girder 1	Bottom Flange (XG1)	Section Loss & Rivet Head Loss and Thinning	Carry out a flange repair Full width of the bottom flange to NR/CIV/SD/828 AFC Repair type B.
Span 1	Deck 2	DCK2 (Between Cross Girder 3 & 4)	Deck Plate (DK2 - XG3/XG4)	Holes - New	Weld a new plate to the underside of the steel deck using a 6mm CFW; new plate to extend a minimum of 100mm past the defect area.

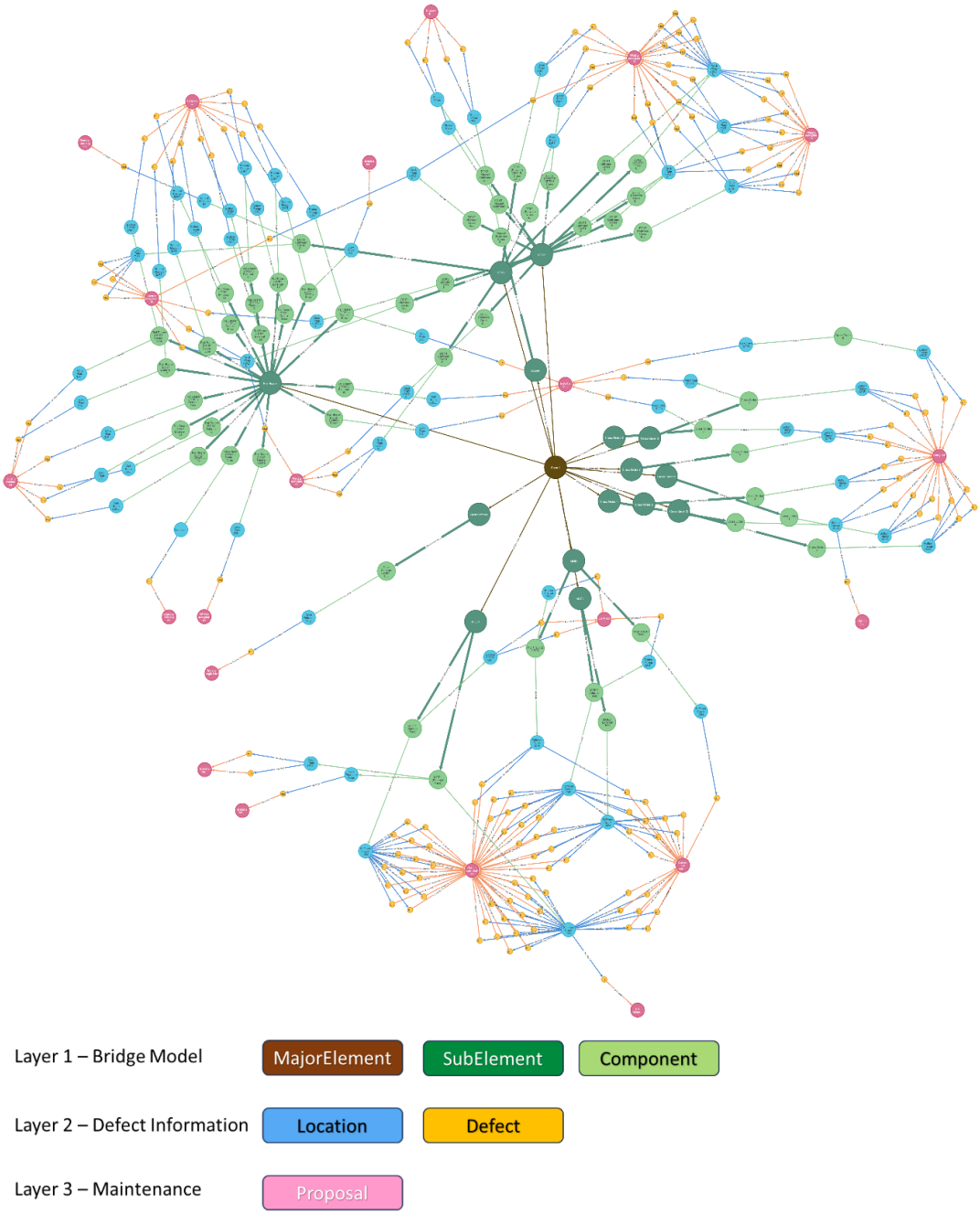

Figure 2: Implementation of Graph-based and Maintenance-oriented Bridge Element Model in Neo4j

3 Graph Learning for Link Prediction

The business logics of the original dataset is to document all the defects identified in a bridge inspection and recommend repair method for each of the defect, preparing for review of other stakeholders. With the Graph-based and Maintenance-oriented Bridge Model, the second stage of the study is to automate the process of generating recommendation with graph machine learning.

From the original dataset and the established graph-based bridge model, the following patterns can be observed: 1) The appearance of defects follow some simple patterns (e.g. same type of defect tend to appear on certain types of bridge element); 2) There are only a limited number

of potential defect repair techniques and 3) the selection of repair techniques depends on the bridge element where the defect is identified and the type (including severity and extent) of the defect. Based on the reasons described above, the recommendation procedure can be formulated as a link prediction based supervised machine learning task - training a model to learn the features of the targeted node-pairs in the graph, where relationships should exist. Thus, the trained model could match the newly added 'Defect' node to the most likely 'Proposal' node.

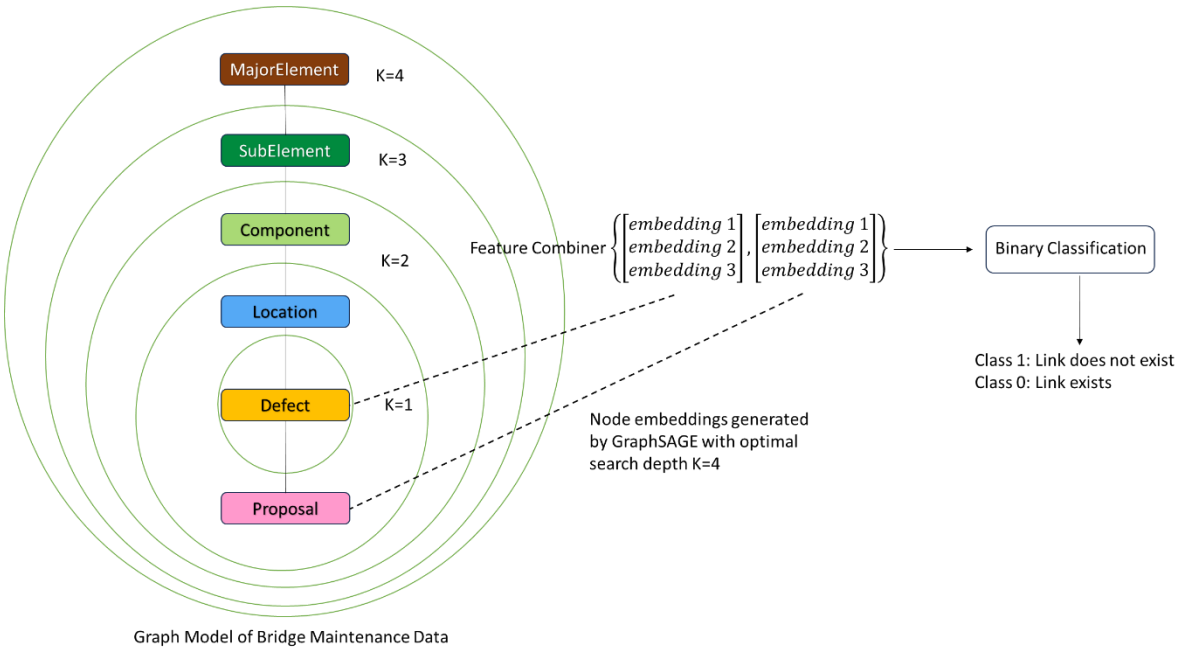

Figure 3: Illustration of Graph Embeddings and Link Prediction Pipeline

As identified from the original dataset, maintenance method decision-making is determined by two types of features: 1) defect location – which bridge component that the defect is developed upon, and 2) defect evaluation in terms of extent and severity. Therefore, the proposed approach of feature engineering is to capture the above two characteristics by obtaining the topological representation of the defective component (i.e. 'Location' node for the defect instance) and the defect type. The procedures of obtaining the graph features and conducting binary classification are illustrated in Figure 4 and explained in Sections 3.1 and Section 3.2.

3.1 Graph Embeddings of Nodes

As a pre-processing step, graph data shall be engineered to produce features that are amenable to machine learning algorithms. In this case, the location information and the defect evaluation are considered as categorical variables, so encoding technique for categorical data shall be applied to convert them into numerical form for downstream machine learning tasks.

One-hot encoding is selected to convert string properties to numeric arrays. Though this conversion increases dimensionality of data, it is effective in preventing unintended biases based on the order of categories (Oyebamiji Micheal, 2023). This advantage is a more important factor compared with the possible growth in computational complexity introduced by encoding.

After the encoding of types of bridge elements and defects, there are a few algorithms available to obtain the embeddings of the node, such as Fast Random Projection (Chen *et al.*, 2019), node2vec (Grover and Leskovec, 2016), HG-GNN (Chen *et al.*, 2023), etc. However, most of

the node embedding generation frameworks are transudative and do not efficiently generalise to unseen nodes or across different graphs. To meet the requirement of this application where new nodes and relationships are constantly added in the graph, GraphSAGE algorithm (Hamilton, Ying and Leskovec, 2017) is utilised. As an inductive learning algorithm, it can generate embeddings for unseen nodes without re-producing embeddings of the existing nodes. In addition, it is able to convert not only the topology of the neighbourhood of the node, but also the properties of the node and its neighbouring nodes (i.e. defect information), into embedding vectors.

The basic principle of GaphSAGE is to initiate the nodes aggregate information of their neighbouring nodes layer by layer and reach further parts of the graph with iteration of this process and finally generate node representation for the whole graph. The number of layers for information aggregation is referred to as Search Depth k, the aggregating architecture can be implemented by various architectures including mean, pool, etc.

3.2 Model Training for Link Prediction

The bridge maintenance knowledge graph is heterogeneous therefore the link to be predicted is between nodes of different types (e.g. bridge element, defect instance, maintenance proposal, etc.), this is where traditional graph algorithms (e.g. graph traversal, topological sort, etc.) can expose limited capability. Whereas supervised machine learning algorithm can be directed to learn only a specific relationship type between specific source and target nodes.

In the link prediction model training pipeline, the existing links in the graph are labelled as positive samples and some non-existent links are created as negative samples. Then the positive and the negative samples are separated into training and test sets. After that, link features are derived by combining node properties of the node-pair. Three feature combining techniques are trialed to evaluate the predication performance: Cosine, L2 and Hadamard product.

In this study, link prediction is transformed to a binary classification problem:

$$Y = CE\big(softmax(Wx + b)\big) \tag{1}$$

where Y is possibility of the existence of the link, X is the combined feature of the node-pair, W is the coefficients for the features of the link and b is bias. The classification model is trained by minimizing a loss function which depends on a weight matrix and on the training data. The loss function of the Adam optimiser, a gradient descent type algorithm, can be defined as:

$$Loss\ Function = CE\big(softmax(Wx + b)\big) \tag{2}$$

where CE is the cross-entropy loss and x is a feature vector training sample. As a gradient descent-based training, the best weights for the model is sought after, through processing training dataset to compute the loss and the gradient of the weights in each epoch. These gradients are then used to update the weights.

This model shall be trained for “Defect - Proposal” node pairs. The trained model can then be applied to the graph to create new relationships containing the predicted links. These potential links shall have an attribute to store the predicted probability of the link, which can be seen as a relative measure of the model’s prediction performance. A threshold can be defined to include only predictions with probability greater than certain value (i.e. 50%) and those with high probability are recommended as proposed maintenance methods.

4 Experiment and Results

The experiment was conducted with different settings of the pipeline, including hyperparameters of the GraphSAGE model (shown in the Table 2) and feature combining algorithms (Cosine, L2 and Hadamard) to provide quantitative insight of the proposed approach as trials and errors. The objective is to identify how the proposed schema performs and which set of modeling settings is optimal.

GraphSAGE		**Logistic Regression**	
Hyperparameters	**Values**	**Hyperparameters**	**Values**
Embedding Dimension	128	Penalty	[0.001, 10000]
Learning Rate	0.1	Patience	2
Batch Size	100	Negative Class Weight	1
Maximum Epochs	1000	Maximum Epochs	5000

Table 2: Configuration of GraphSAGE Model and Logistic Regression Model

The performance of link prediction pipeline, consisting of embeddings generated by GraphSAGE, feature combing algorithms and logistic regression based binary classification, was evaluated with the Receiver Operating Characteristic (ROC) and the Area Under the ROC Curve (AUC). Starting with the four possible outcomes of binary classification: true positive (TP), false positive (FP), true negative (TN) and false negative (FN), ROC is plotted on x-y coordinates where x-axis is false positive rate (FPR) and y-axis is the true positive rate (TPR). FPR and TPR are defined as below:

$$FPR = \frac{FP}{TN + FP} \tag{3}$$

$$TPR = \frac{TP}{TP + FN} \tag{4}$$

Then AUC can be calculated by:

$$AUC = \frac{\sum(p_i, n_j)}{P \times N} \tag{5}$$

Where P is the number of positive examples, N is the number of negative examples, p_i is the prediction score of a positive example and n_j is the prediction score of a negative example. It can be interpreted as the probability that a positive example score is greater than a negative example score when they are both sampled randomly. A value of AUC closer to 1 indicates better performance while a value around 0.5 suggests a random classifier.

The link prediction performance of different pipeline settings are summarised in Figure 4 and Figure 5. Other parameters were taken as default values as shown in Table 2.

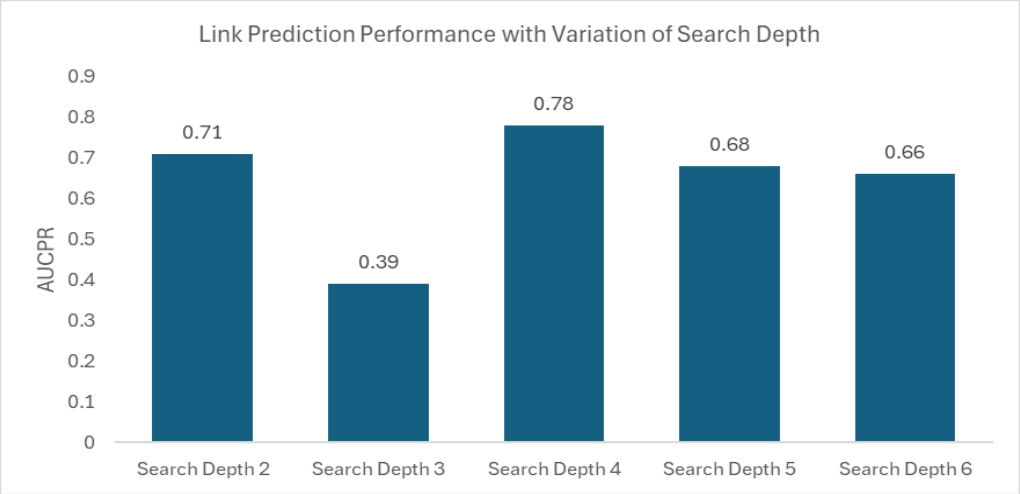

Figure 4: Link Prediction Performance of Different Pipeline Settings Part 1

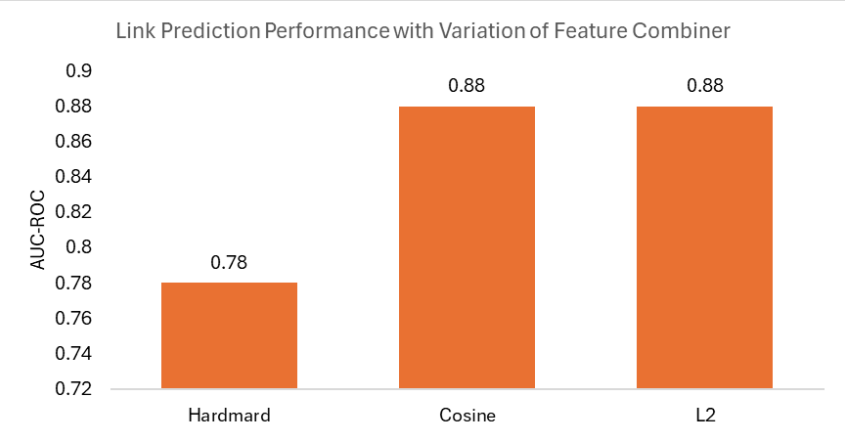

Figure 5: Link Prediction Performance of Different Pipeline Settings Part 2

It is interpreted that Search Depth 4 is optimal and L2 or Cosine feature combing algorithms are better than Hardmard. In addition, on the current dataset using GraphSAGE embeddings, Mean Aggregator outperforms in comparison with Pool Aggregator. Variation of Activation Function was also tested where Sigmoid outperforms ReLU. It is considered that the search depth K is largely graph dependant (as illustrated in Figure 3) while the feature combiner and the aggregator architecture is more likely to generalise.

5 Conclusion

This paper presented a workflow for storage of bridge maintenance data on a graph database enabled bridge model and predict the link of the 'Defect - Proposal' node-pair via graph learning of embeddings generated by GraphSAGE.

The proposed approach was tested with the following variations to find the optimal performance of the machine learning model: 1) Tuning of GraphSAGE model and 2) Node pair feature combining algorithms. This holistic schema allows for efficient management of the identified defects and development of remediation plans, as well as the automation of bridge assessment procedures defined in the bridge maintenance standards.

The major limitation of the presented solution is that predictions are derived from statistical principles and probabilistic reasoning without incorporating validation of the actual success of the predicted repair method. To mitigate this issue, feedback from bridge practitioners can be collected after each prediction to update the graph model accordingly.

For the future work, the encoding technique applied for categorical data (i.e. bridge component and defect type) can be upgraded to more advanced encoding techniques to capture semantics of different bridge element and defect information. On the other hand, the same pipeline and learnings can be experimented with dataset from another bridge, then the reasons for performance variation of different model hyperparameters can be further investigated. Providing the bridge with the same structural arrangement (e.g. steel girder bridge), the pattern of appearance of defects captured in embeddings could be migrated so that the knowledge can be accumulated and transferred.

6 Acknowledgements

This work was supported by the DigiBridge KTP (Grant Reference Number 10003208) and the dataset was contributed by Jarrod Richards, Centregreat Rail Ltd.

7 Reference

Chen, H. *et al.* (2019) 'Fast and Accurate Network Embeddings via Very Sparse Random Projection', *International Conference on Information and Knowledge Management, Proceedings*, pp. 399–408. doi: 10.1145/3357384.3357879.

Chen, H. *et al.* (2023) 'Hessian-aware Quantized Node Embeddings for Recommendation', *arXiv*, (23). doi: 10.1145/3604915.3608826.

Graphaware (2014) *GitHub - graphaware/neo4j-timetree: Java and REST APIs for working with time-representing tree in Neo4j*, *Github*. Available at: https://github.com/graphaware/neo4j-timetree (Accessed: 10 April 2024).

Grover, A. and Leskovec, J. (2016) 'node2vec: Scalable Feature Learning for Networks', *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. doi: 10.1145/2939672.2939754.

Hamilton, W. L., Ying, R. and Leskovec, J. (2017) 'Inductive Representation Learning on Large Graphs', *Advances in Neural Information Processing Systems*, 2017-December, pp. 1025–1035. Available at: https://arxiv.org/abs/1706.02216v4 (Accessed: 18 March 2024).

Lee, G. and Chi, S. (2023) 'Graph-based Clustering of Bridge Management System Data for Bridge Maintenance Cost Estimation', in *EG-ICE 2023 Conference*.

Lin, J. R. (2020) 'Openbridgegraph: Integrating open government data for bridge management', in *Proceedings of the 37th International Symposium on Automation and Robotics in Construction, ISARC 2020: From Demonstration to Practical Use - To New Stage of Construction Robot*, pp. 1255–1262. doi: 10.22260/isarc2020/0172.

Neo4j (2017) *GitHub Entity-State model managed by Neo4j Procedures*, *Githuib*. Available at: https://github.com/h-omer/neo4j-versioner-core (Accessed: 10 April 2024).

Network Rail (2022a) *Level2 Business Process Management of Structures*.

Network Rail (2022b) *Level3 Manual Structures, Tunnels and Operational Property Examinations Approvals*.

Oyebamiji Micheal (2023) *A Comprehensive Comparison between One-Hot and Ordinal Encoding | by Oyebamiji Micheal | Medium*, *Medium*. Available at: https://medium.com/@oyebamijimicheal10/a-comprehensive-comparison-between-one-hot-and-ordinal-encoding-6f899c4f08b3 (Accessed: 26 March 2024).

Perozzi, B., Al-Rfou, R. and Skiena, S. (2014) 'DeepWalk: Online Learning of Social Representations', *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 701–710. doi: 10.1145/2623330.2623732.

Tiwary, K., Patro, S. K. and Sahoo, B. (2022) 'Bridgebase: A Knowledge Graph Framework for Monitoring and Analysis of Bridges', *Lecture Notes in Civil Engineering*, 244, pp. 409–420. doi: 10.1007/978-981-19-0656-5_34/FIGURES/5.

Xia, L. *et al.* (2022) 'Toward cognitive predictive maintenance: A survey of graph-based approaches', *Journal of Manufacturing Systems*, 64, pp. 107–120. doi: 10.1016/J.JMSY.2022.06.002.

Xia, L. *et al.* (2023) 'Maintenance planning recommendation of complex industrial equipment based on knowledge graph and graph neural network', *Reliability Engineering & System Safety*, 232, p. 109068. doi: 10.1016/J.RESS.2022.109068.

Xu, H. *et al.* (2023) 'Knowledge graph and CBR-based approach for automated analysis of bridge operational accidents: Case representation and retrieval', *PLOS ONE*, 18(11), p. e0294130. doi: 10.1371/JOURNAL.PONE.0294130.

Zangeneh, P. and McCabe, B. (2020) 'Ontology-based knowledge representation for industrial megaprojects analytics using linked data and the semantic web', *Advanced Engineering Informatics*, 46, p. 101164. doi: 10.1016/J.AEI.2020.101164.