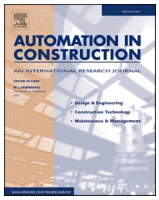
Automation in Construction 166 (2024) 105634

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon



Exploring bridge maintenance knowledge graph by leveraging GrapshSAGE and text encoding

Yan Gao [a], Guanyu Xiong [a], Haijiang Li [a,*], Jarrod Richards [b]

[a] BIM for Smart Engineering Centre, School of Engineering, Cardiff University, Cardiff CF24 3AA, UK
[b] Centregreat Rail Limited, Bridgend CF31 2AD, UK

ARTICLE INFO

Keywords:
Bridge maintenance knowledge graph
Text encoding
Graph neural networks
Node classification
Link prediction

ABSTRACT

Knowledge graphs (KGs) are crucial in documenting bridge maintenance expertise. However, existing KG schemas lack integration of bridge design and practical inspection insights. Meanwhile, traditional methods for node feature initialization, relying on meticulous manual encoding or word embeddings, are inadequate for real-world maintenance textual data. To address these challenges, this paper introduces a bridge maintenance-oriented KG (BMKG) schema and approaches for graph data mining, including node-layer classification and link prediction. These methods leverage large language model (LLM)-based text encoding combined with GraphSAGE, demonstrating excellent performance in semantic enrichment and KG completion on deficient BMKGs. Additionally, ablation studies reveal the superiority of the pre-trained BERT text encoder and the L2 distance pairwise scoring calculator. Furthermore, a practical implementation framework integrating these approaches is developed for routine bridge maintenance, which can facilitate various practical applications, such as maintenance planning, and has the potential to enhance the efficiency of engineers' documentation work.

1. Introduction

In recent years, the use of knowledge graphs (KGs) has emerged as a highly effective tool in capturing the specialized knowledge related to bridge maintenance. Efforts have been concentrated on automatically creating bridge ontologies and KGs using natural language processing (NLP) techniques like named entity recognition [1,2] and relationship extraction [3,4], but few studies focused on graph data mining after that. Moreover, there is rarely a KG schema that incorporates both the bridge design diagram and the practical inspection insights. Additionally, traditional methods for node feature representations in bridge KGs, such as manual one-hot encoding or word embeddings like word2vec, have limitations. The former requires meticulous design for differentiation, which can be labour-intensive, while the latter is ill-suited to capturing sentence-level context. Therefore, it is advantageous to develop a bridge KG schema rooted in practical maintenance, and knowledge mining approaches by leveraging graph neural networks (GNNs) and advanced text encoding techniques.

To address these challenges, this study introduces a bridge maintenance-oriented KG (BMKG) schema by incorporating structural designs and real-world maintenance reports. It also proposes methods for node-layer classification and link prediction through inductive learning and contrastive learning, respectively, in which the text embeddings derived from the pre-trained text encoders or large language modes (LLMs) are innovatively combined with GraphSAGE neural networks for data mining on BMKGs.

Using the proposed schema, a BMKG with 188 nodes and 263 relationships was constructed from practical semi-structured bridge inspection reports. The above methods were then tested on incomplete BMKGs, which are generated by randomly removing a certain proportion of links from the original intact one. The experiments demonstrate that the proposed methodologies can achieve promising performance and generalizability for node-layer classification and link prediction, reaching an average test accuracy of 96.51% and an average test AUC of 0.8577, respectively, through cross-validation on the target dataset. Furthermore, the ablation studies reveal the superiority of the pre-trained BERT encoder and the L2 distance-based score calculator for downstream graph data mining tasks on the target dataset.

Finally, a practical implementation framework that incorporates the above approaches is developed for effective bridge maintenance. This framework cannot only mitigate the impact of pseudo-negative examples (generated by simple random repairing) on contrastive learning in a

* Corresponding author at: BIM for Smart Engineering Centre, School of Engineering, Cardiff University, Cardiff CF24 3AA, UK.
E-mail address: LiH@cardiff.ac.uk (H. Li).

https://doi.org/10.1016/j.autcon.2024.105634
Received 13 August 2023; Received in revised form 15 July 2024; Accepted 15 July 2024
Available online 25 July 2024

closed-loop system but can also facilitate various practical applications such as semantic enrichment, compliance checks, dependency analysis, and generating repair recommendations. Furthermore, it also has the potential to enhance the efficiency of engineers' documentation processes through the plugs for text prompting or automatic maintenance planning.

The contribution of this study is four-fold as below:

1) This paper introduces a hierarchical KG schema tailored for bridge maintenance based on real-world textual reports. The schema is organized into 5 layers, including main structure, components, elements, defects, and repair actions, with node connections limited in adjacent layers.
2) A node-layer classification approach by employing GraphSAGE and LLM-based text encoding is proposed for semantic enrichment in a deficient BMKG.
3) Moreover, a link prediction approach by leveraging GraphSAGE and contrastive learning is proposed for graph completion in a deficient BMKG, which can achieve excellent performance.
4) An implementation framework that integrates the above schema and approaches is designed for routine bridge maintenance, which can not only facilitate practical applications but also mitigate the influence of pseudo-negative examples on contrastive learning.

2. Literature review

2.1. Bridge maintenance knowledge engineering

Knowledge engineering (KE) is a field of artificial intelligence (AI) that creates rules by applying data to imitate the thought process of a human expert. It looks at the structure of a task or a decision to identify how a conclusion is reached [5]. In the AEC industry, KE refers to knowledge representation, acquisition, reasoning, decision-making, and application in building, operation, and maintenance. Currently, a knowledge graph (KG) has become one of the most effective tools for knowledge management and integration. For most studies in the construction fields, KGs are usually termed ontological semantic networks based on graphics with domain entities as nodes, and the defined entities relationships as edges [6].

To represent the intricate knowledge of bridge maintenance, Ren et al. [7] developed an ontology for bridge maintenance (BrMontology) based on web ontology language (OWL), which covers bridge structure, damage (and causes), solutions, and big events. It can enable automatic rule-based maintenance planning and holistic decision-making. Liu and ElGohary [8] proposed a bridge ontology (BridgeOnto) based on routine inspection reports and maintenance manuals in the US, in which the bridge elements, types of bridge elements, and bridge defects are elaborately decomposed. Zhang et al. [9] proposed a comprehensive bridge maintenance knowledge graph (BMKG) by involving expenses, which used semantic triples for knowledge organization and a Neo4j graph database for data storage. Similarly, Lee et al. [10] established a graph database to organize data from bridge management systems (BMS) and employed graph clustering for maintenance cost estimation. Wu et al. [11] developed an ontology for project management of bridge rehabilitation, which covers restoration tasks and constraints.

Additionally, several studies focused on the knowledge-based bridge management system (BMS) by leveraging bridge real-time monitoring and pattern recognition. For example, Li et al. [12] designed a bridge structure health monitoring (SHM) system based on a fine-grained ontology to integrate heterogenous data from various sensors. Yang et al. [13] developed a framework for bridge management based on a big-data knowledge engineering paradigm, consisting of both data (including sources, storage, and computing) and knowledge (including representation, computing, and services) layers. This framework can facilitate intelligent bridge maintenance by leveraging the big data within the entire bridge life cycle.

Moreover, some efforts have been made for automatic ontology generation and KG completion, such as named entity recognition (NER) for target classes [1,2] (i.e., bridge element, deficiency, cause, repair action) and dependency parsing [3,4] (i.e., relation extraction) from textual reports. NLP approaches are utilized in these works to build or complete semantic triples (i.e., ontologies). However, subsequent data mining still relies on classic algorithms through simple graph structures, such as centrality, clustering, and pathfinding. Meanwhile, there have been a few studies exploring graphs based on infrastructure architecture, but they are mainly focused on modular design or generation, such as mass customization of precast bridge systems [14] and building structure layouts [15,16]. Therefore, it would be advantageous to create a comprehensive BMKG by integrating knowledge from both bridge structure design and maintenance experience gleaned from textual reports. Additionally, there is a need to investigate the application of emerging AI techniques, such as Graph Neural Networks (GNNs), for practical implementations within this knowledge framework.

2.2. Graph neural networks and applications

Although the classic deep neural networks (DNNs) achieve great success for latent embedding from Euclidean spatial data, they cannot perform satisfactorily in processing non-Euclidean data, such as graphs. Hence, graph neural networks (GNNs) are proposed to solve this issue (i.e., graph embedding), which are defined as an optimizable transformation on all attributes of the graph (nodes, edges, global context) that preserves graph symmetries (permutation invariances) [17]. Graph embedding is a process to generate a vector from graph features and attributes but tries to preserve graph information as far as possible so that the downstream graph analytic tasks can be achieved easily using the off-the-self machine learning (ML) algorithms [18]. Embedding aggregation and message passing are the primary attributes of GNNs. The common GNN architectures [19] include graph convolutional networks (GCN), graph attention networks (GAT), GraphSAGE, spatial-temporal graph neural networks (STGNN), etc. Most GNNs are inherently transductive, in which the model is trained through the whole graph and can only generate graph embeddings for a single fixed graph, so they are not efficiently suitable for involving graphs and cannot learn to generalize across different graphs. In contrast, a few others, like GraphSAGE, are based on inductive learning, in which the model can only see the training data and is trained through a sample partial graph or a set of subgraphs. Thus, the generated model and graph embeddings can be generalized to unseen nodes.

As GNNs function on graph data, they are typically utilized for tasks such as graph classification, node classification, clustering, and link prediction. In the AEC industry, Collins et al. [20] first applied graph classification through GCN to classify IFC building objects based on their geometry. Wang et al. [21] proposed an improved GraphSAGE algorithm (SAGE-E), by involving edge features in aggregation, for semantic enrichment of BIM models (i.e., identify room types) using node classification. Additionally, link prediction is usually utilized for KG completion, relation prediction, and recommendation. Hence, in the realm of maintenance for manufacturing systems, Xia et al. [22] developed an approach to recommend potential solutions and explain the fault for oil drilling equipment by using linkage prediction through an attention-based compressed relational GCN (ACRGCN). Its model was pre-trained with word2vec embeddings as node features and the machine components within the graph were organized as only a single layer.

Nevertheless, the hierarchical bridge architecture suggests that the multi-layer structure is likely more suitable for constructing its KG. The modular bridge architecture often exhibits repetitiveness and symmetry, which can be observed from textural inspection reports. Meanwhile, the content of practical bridge maintenance reports, especially repair proposals, is unlikely to adopt straightforward phrases used in the study [22]. In these reports, the content tends to be more complex and

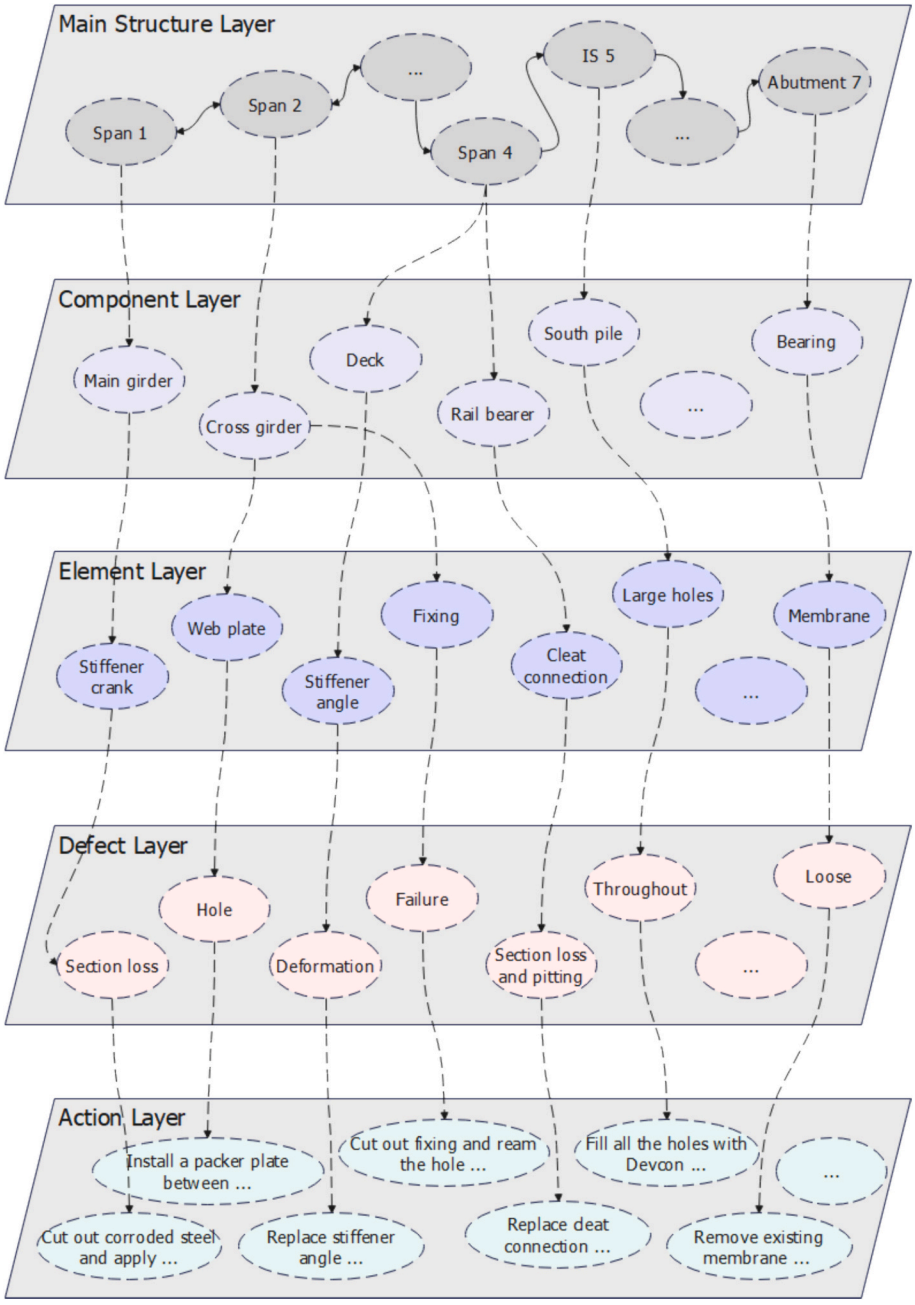


Fig. 1. Proposed maintenance-oriented knowledge-graph schema for bridges.

comprehensive and presented in sentence form. Furthermore, GNN has demonstrated its potential in typical NLP tasks, like computing semantic textual similarity and sentence completion. Hence, it would be advantageous to 1) propose a hierarchical maintenance-oriented KG schema based on bridge architectures and practical textural reports, and 2) explore different text embeddings as well as GNN algorithms for graph data mining on the generated KG to enhance practical applicability, e.g., node classification for properties comply check or semantic enrichment, link prediction for maintenance recommendation.

2.3. Word and sentence embedding

As is known, machines have remarkable proficiency in handling numerical data, but they encounter challenges when processing text input (usually functioning less efficiently). Therefore, it is imperative to convert the text into a format that machines can effectively interpret. Various static word embedding techniques, including Bag of Words [23], TF-IDF [24], Word2Vec [25] and GloVe [26], have been developed to produce n-dimensional vectorized representations (i.e., word embeddings) of individual words by looking up through the created "tables". These embeddings aim to capture the meanings and the semantic relations of the words, as well as the diverse contexts in which they are utilized, such as a classic example [25]: by removing the man attribute from 'King' and incorporating the woman characteristic, appropriate word embeddings can arrive at 'Queen', i.e., $King - Man + Woman = Queen$, which encapsulates the comparison between them. They are supposed to maintain attributes of the individual words from the training set by usually projecting similar ones as neighbours in a hyperspace. Word embeddings can be adopted for different downstream tasks, such as sentiment classification and name entity recognition (NER). Moreover, through transfer learning, the pre-trained word embedding models can be extended to a wide range of applications.

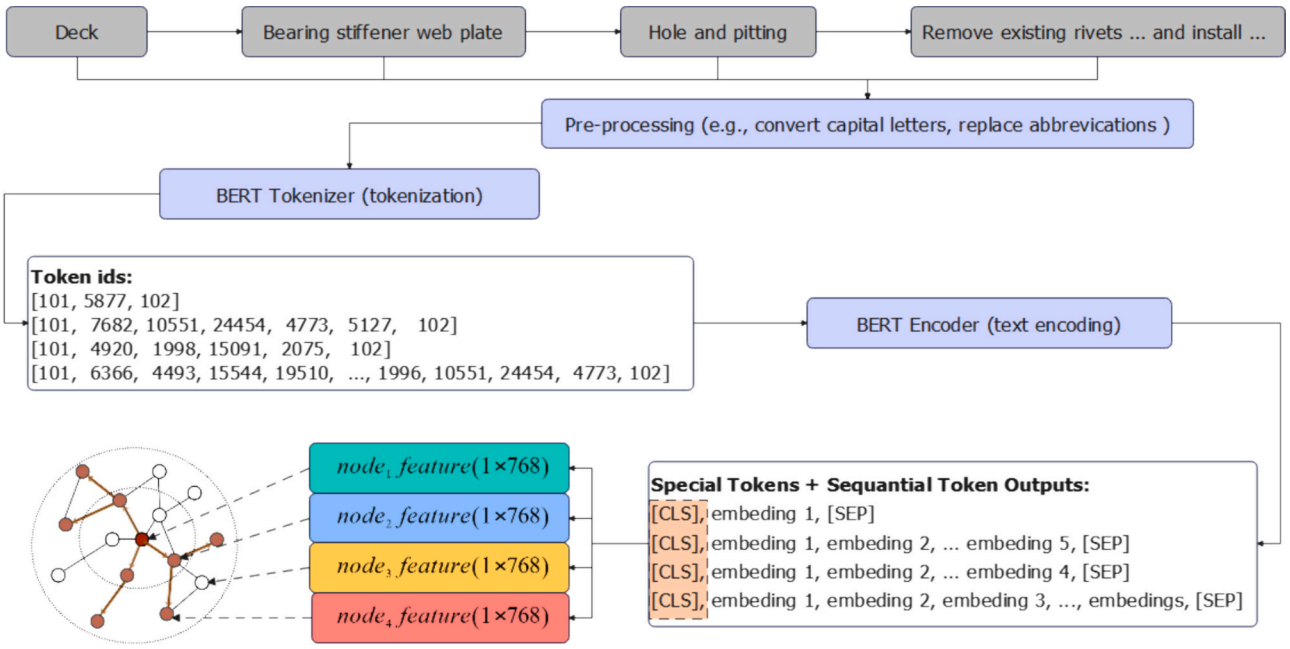


Fig. 2. Procedures of text encoding to generate node features.

However, word embeddings encounter challenges in dealing with a sentence, e.g., when manipulating multiple word embedding vectors to produce a vectorized representation of a sentence. Hence, it is necessary to develop techniques to convert the sentence to a n-dimensional vectorized representation, i.e., sentence embedding.

Word and sentence embeddings are both subsets of text embeddings. The ideal text embeddings (or text representations) can retain the semantic information of the text as much as possible. Texts with the same semantics but different expressions should be mapped to the same position (or neighbours), while texts with different semantics should maintain corresponding distances in the vector space. Currently, numerous approaches and pre-trained models have been created for text embedding, e.g., fastText [27], BERT [28], T5 [29], Instructor [30], as well as the latest large language models (LLMs) such as ChatGPT [31] and Llama [32,33]. Most of them are based on the transformer architecture and trained on two major objective tasks in natural language processing (NLP), including masked language modelling (MLM) and next sentence prediction (NSP).

In practice, text embedding vectors (i.e., text representations) are usually derived through the following methods: 1) with the vectorized representation of the [CLS] token as text embedding, i.e., [CLS] token embedding; 2) with the vectorized representation of the [CLS] token after processing through multilayer perceptron (MLP) as text embedding, i.e., [CLS] + MLP embedding; 3) with all the tokens' vectorized representations after pooling (mean or max pooling) as text embedding, i.e., all the tokens + pooling embedding; 4) with all the tokens' vectorized representations after pooling (mean or max pooling) and MLP processing as text embedding, i.e., all the tokens + pooling + MLP embedding. The evaluation of text embeddings is based on their performance in different downstream tasks, such as classification, clustering, pair classification, reranking, retrieval, semantic textual similarity (STS), summarization, and bitext mining.

3. Methodology

3.1. Problem statement

This research aims to develop a hierarchical Knowledge Graph (KG) focused on bridge maintenance, referred to as BMKG. The goal is to create a structured framework that incorporates information from bridge designs and actual textual maintenance reports. Furthermore, the study will explore different text embedding techniques and Graph Neural Network (GNN) models for efficient data mining within the BMKG, thus increasing its utility in real-world applications.

The BMKG is designed to evolve by integrating ongoing inspection reports throughout a bridge's lifecycle. It typically functions as an incomplete graph, marked by the introduction of new nodes without the corresponding links. For example, a newly identified defect might not have a corresponding repair proposal, whether the defect is newly discovered or an existing one found in a new location. In such cases, experienced engineers can utilize the accumulated maintenance records (i.e., a knowledge base) to pinpoint similar defects and suggest appropriate repair actions. This study is inspired by the above observation and the specific objectives of this study can be listed below:

1. Propose a hierarchical KG schema oriented towards bridge maintenance incorporating bridge architectures and practical textual reports.
2. Identify the node hierarchy (i.e., node classification) in an incomplete BMKG with missing links through pre-trained text encoding and GNN algorithms for compliance check and semantic enrichment.
3. Predict potential connections (i.e., link prediction) in a deficient BMKG by leveraging text encoding and GNN neural networks for KG completion.
4. Design a practical implementation framework that incorporates the above strategies and approaches for effective bridge maintenance.

3.2. BMKG schema

Modular bridges typically display recurring patterns and symmetries in both their structure and defects. These patterns can be observed from design documents like CAD drawings or Revit models, as well as from textual inspection reports, through semantic textual similarities and consistent demonstrative relationships. For example, bridge spans of the same type usually have similar components (such as main girders, cross girders, soffits, and rail bearers) and tend to show similar types of defects and repair needs (e.g., stiffener cranks with medium section loss <50% and remarkable section loss >50% may require progressively extensive repair actions). In light of this, this study introduces a bridge maintenance-oriented knowledge graph (BMKG) schema, derived from

both bridge design and practical maintenance reports, as depicted in Fig. 1. This schema consists of five distinct layers: the main structure layer, component layer, element (or sub-component) layer, defect (or fault) layer, and repair action (or proposal) layer.

In this schema, nodes in the main structure layer are interconnected. However, other connections (links) are limited to adjacent layers, ensuring no cross-layer or intra-layer connections (except in the top layer). This structure allows for effective graph aggregation within a single BMKG. All connections between different layers are directed from the upper to the lower layers. Previous research has extensively focused on target classification (i.e., Named Entity Recognition, NER) [1,2] and dependency parsing (i.e., relation extraction) [3,4] in the context of bridge inspection reports. As a result, the necessary BMKG can be automatically generated from historical textual reports (whether unstructured or semi-structured), following the proposed KG schema. Furthermore, this BMKG can be dynamically expanded with regular inspections, to facilitate bridge maintenance activities such as information storage, retrieval, and clustering.

3.3. Text encoding and GraphSAGE

3.3.1. Text encoding

Bridge inspection reports are usually unstructured or semi-structured. Previous studies have shown that these textural reports can be transformed into structured data through NLP techniques. This structured data can then be used to create a directed, hierarchical Bridge Maintenance Knowledge Graph (BMKG) following a specific schema. In this schema, textual information, such as "Span 1" → "Deck" → "Bearing stiffener web plate" → "Hole and pitting" → "Remove existing rivets using burning equipment and install a new steel plate to the stiffener web", serve as nodes. Traditionally, node features are represented through manual encoding methods like one-hot encoding or word embeddings such as word2vec. The former requires meticulous design for differentiation, which can be labour-intensive, while the latter may not effectively capture sentence-level context. Therefore, using sentence embeddings to represent node features, especially to capture semantic textual similarity in bridge maintenance reports, is advantageous. These embeddings can be obtained from pre-trained text encoders or language models, like BERT. This process can be illustrated with BERT and [CLS] embeddings in Fig. 2.

Different text encoders can lead to varying performance when their embeddings are used as node features in GNNs. The performance difference can be attributed to several factors below:

1) Quality of embeddings: different text encoders may capture semantic information with varying levels of effectiveness. Some encoders may be better at context understanding or semantic relationship encoding, which can significantly affect the performance of the GNNs that use these embeddings.
2) Pre-training data and objectives: the pre-trained text encoders are based on specific datasets and tasks (such as language modelling, sentence prediction, etc.). It can influence how well the embeddings meet the specific requirements of the downstream task. For example, an encoder trained on a dataset like the application domain may produce more relevant embeddings.
3) Dimensions: different encoders may produce embeddings of different sizes. In principle, higher-dimensional embeddings can potentially capture more information but at the cost of increased computational complexity. Conversely, lower-dimensional embeddings are computationally efficient but may lack expressiveness.
4) Model architecture: the architecture of the text encoder (e.g., LSTM, transformer) can affect task performance. For example, transformer-based models like BERT are known for capturing contextual information effectively, which may be beneficial for certain GNN tasks.
5) Robustness: some encoders are more robust to variations in specific text, like slang and typos. The robustness of embeddings can affect the model's ability for generalization.

3.3.2. GraphSAGE

Traditional transductive GNNs, like the Graph Convolutional Network (GCN), are limited to creating representations through all the nodes for a single fixed graph [34], which is difficult to generalize for unseen nodes in an evolving graph with new links, as well as different graphs. GraphSAGE addresses these limitations by learning a dynamic representation method across subsets of graphs through inductive training, which is particularly beneficial for graphs that evolve with rich node attribute information [35]. Once trained, GraphSAGE can generate representations for new nodes or entirely different graphs, provided they share the attribute schema of the training data. Additionally, the learned node representation can change along with the neighbour relationship variation in an evolving graph. Therefore, in this study, GraphSAGE is employed to create low-dimensional vector representations (graph embeddings) for nodes in the proposed BMKG. The algorithm for GraphSAGE embedding is outlined in the following pseudo-code.

Algorithm 1. Generating node representation based on graphSAGE

Input: Graph $G(V,E)$; input feature $x_v, \forall\ v \in V$; depth K; weight matrices $W^k, \forall\ k \in (1,...,K)$; non-linearity σ; differentiable functions $AGGREGATE_k, \forall\ k \in (1,...,K)$; neighborhood function N

Output: Vector representation z_v for all $v \in V$

$h_v^0 \leftarrow x_v, \forall\ v \in V$
while $k \leq K$ **do**
  while $v \in V$ **do**
    $h_{N(v)}^{k-1} \leftarrow AGGREGATE_k(h_u^{k-1}, \forall\ u \in N\ (v))$;
    $h_v^k \leftarrow \sigma(W^k \cdot CONCAT(h_v^{k-1}, h_{N(v)}^{k-1}))$
  $h_v^k \leftarrow h_v^k / \|h_v^k\|_2, \forall\ v \in V$
return $z_v \leftarrow h_v^K, \forall\ v \in V$;

During the GraphSAGE embedding process, nodes progressively aggregate more information from their neighbours and extend their reach across the graph through iterative processes. Suppose there are K aggregation functions, denoted as $AGGREGATE_k$ ($\forall k \in \{1, \ldots, K\}$), which are used for aggregating features from node neighbours. These functions are associated with a set of weight matrices, denoted as $\boldsymbol{W}^k$ ($\forall k \in \{1, \ldots, K\}$), which are utilized for message propagation across different layers of the model [35]. The variable k represents the current

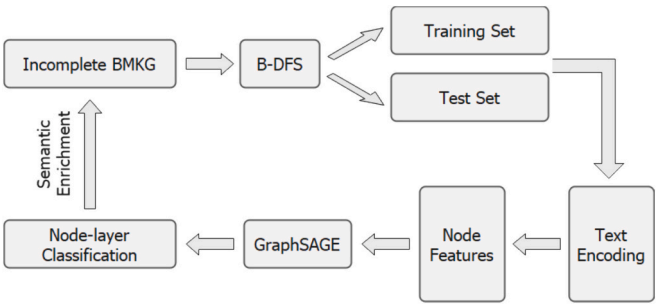


Fig. 3. Proposed pipeline for NLC in an incomplete BMKG.

iteration step (or the search depth) and h^k denotes the latent embedding at step k (or the hidden layer k). Initially, $k = 0$, indicating the input node features.

At first, each node $v \in V$ aggregates its neighbourhood embeddings, i.e., $h_u^{k-1}\ \forall u \in N(v)$, into a single vector $h_{N(v)}^{k-1}$. Then, GraphSAGE concatenate the node's embedding h_v^{k-1} with the aggregated neighbourhood embedding $h_{N(v)}^{k-1}$, and feed the result into a fully connected neural network (FCN) with a nonlinear activating function σ to generate the representation vector (i.e., h_v^k). Finally, the representation output at the depth K is obtained after normalization as $z_v = h_v^K,\ \forall v \in V$. In practice, the aggregation of neighbour embeddings, i.e., $AGGREGATE_k$, can be implemented through diverse architectures [35], including mean, pooling, GCN, and LSTM. The mean and pooling aggregators are adopted in this study, as indicated in Eq. 1.

$$\boldsymbol{h}_v^k \leftarrow \sigma\left(\boldsymbol{W}\cdot MEAN\left(\left\{\boldsymbol{h}_v^{k-1}\right\} \cup \left\{\boldsymbol{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\right\}\right)\right)\# \tag{1}$$

$$AGGREGATE_k^{\text{pool}} = max\left(\left\{\sigma\left(\boldsymbol{W}_{\text{pool}}\boldsymbol{h}_{u_i}^k + \boldsymbol{b}\right), \forall u_i \in \mathcal{N}(v)\right\}\right)\# \tag{2}$$

3.4. Proposed approaches and framework

3.4.1. Node-layer classification

Node-Layer Classification (NLC) can be applied for compliance checks or semantic enrichment within a BMKG. This process typically begins with a Depth-First Search (DFS), a traditional method used for finding the shortest path between two nodes in a graph. The effectiveness of DFS relies on graph integrity. For example, in an ideal, complete BMKG, once the nodes in the top layer (i.e., the main structure layer) are identified, the layers of the other nodes can be determined using bidirectional depth-first search (B-DFS) according to the 5-layer schema configuration. However, BMKGs based on real-world inspection reports might be incomplete due to data gaps or inconsistencies, like missing entities or relationships. In such scenarios, some nodes may remain unclassified after B-DFS due to absent connections. To address this challenge in an incomplete BMKG, a method that combines pre-trained text encoding with GraphSAGE is proposed. This approach utilizes contextual information such as semantic textual similarity and parts of speech, along with the benefits of GNNs. The methodology for this process is depicted in Fig. 3.

In the beginning, bidirectional B-DFS is used for node-layer classification in the incomplete BMKG with missing links. It starts from the given node in the top layer to the bottom layer (i.e., top-down DFS), and then reverses from the classified nodes in the bottom layer to the upper layer (i.e., down-top DFS), which aims to identify the missing nodes due to absent connections in the top-down DFS. Subsequently, the classified nodes are taken as the training set and the remaining unidentified nodes are adopted as the test set. After encoding through an appropriate pre-trained text encoder (or language model), the generated text embeddings are utilized as features for each node. Finally, a model consisting of multiple GraphSAGE layers (for graph embedding or aggregation) and a Softmax layer (as the classifier) is trained through inductive learning for NLC.

The loss function is based on multi-class cross-entropy (MCE), indicated in Eq. 2.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{M} y_{ic} log(p_{ic})\# \tag{3}$$

Where, L is the average loss for N examples; M is the number of classes and $c \in [1, M]$; y_{ic} is a binary value (0 or 1) – if the ith example's true class is c, $y_{ic} = 1$, and vice versa; p_{ic} is the probability of the ith example belonging to the class c, which is derived from the Softmax function (Eq. 3).

The performance of NLC is assessed using accuracy, indicated in Eq. 4.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}\# \tag{4}$$

Where, TP – true positive; TN – true negative; FP – false positive; FN – false negative.

3.4.2. Contrastive link prediction

In an evolving Bridge Maintenance Knowledge Graph (BMKG), which often resembles an incomplete graph with missing links due to data scarcity, determining the connections between new and existing nodes is a challenge of link prediction. This has various practical applications, such as suggesting repairs and creating dependencies. This study proposes a method based on contrastive learning to address this challenge, comprising the following steps [36]:

1) Existing and missing links in the BMKG are considered positive examples, labelled as {1, 1, …, 1}. Existing links form the positive training set while missing links are used as the positive test set.

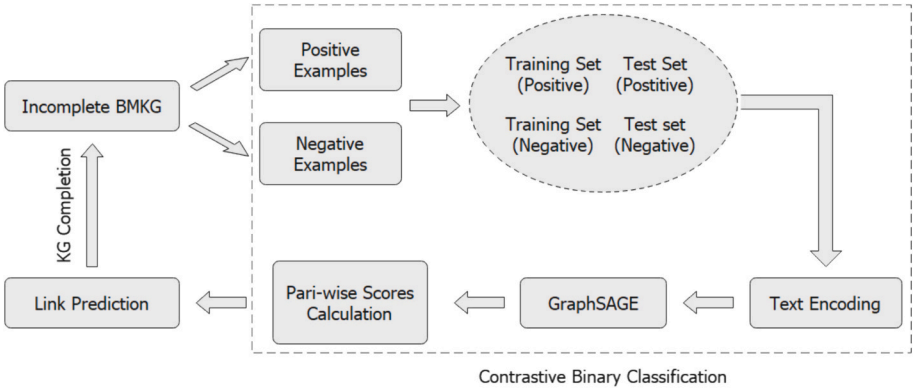


Fig. 4. Proposed pipeline for link prediction in an incomplete BMKG.

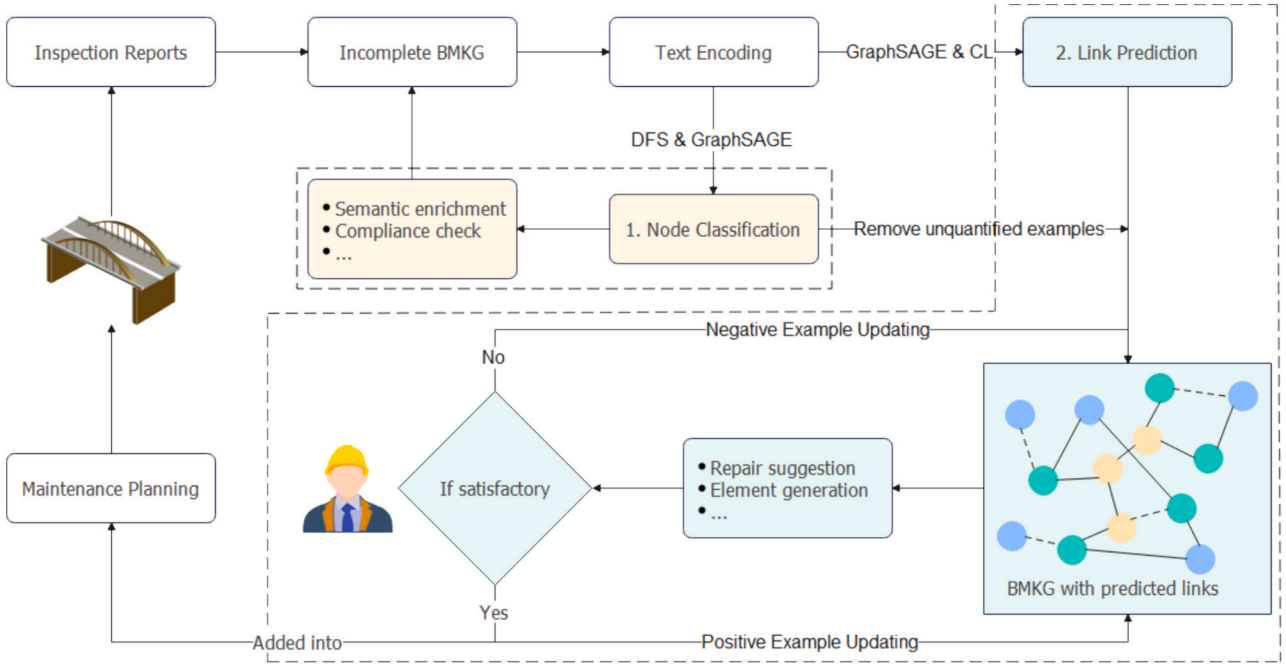


Fig. 5. Framework applying NLC and link prediction for practical implementation.

2) Non-existent edges in the current BMKG, representing node pairs without edges, are randomly sampled as negative examples and labelled as {0, 0, …, 0}. Then, they are divided into training and test sets.
3) All nodes in both sets are encoded using a pre-trained text encoder or language model. The generated text embeddings become node features for graph embedding.
4) A binary classification model, designed for link prediction, is trained using inductive learning. This model includes graph embedding based on GraphSAGE architecture and a pair-wise score calculation.
5) The score is derived using various metric-based methods like L2 distance, dot product, cosine similarity, element-wise product (sum or mean), and MLP.
6) Finally, a binary classification model based on graph embedding and pair-wise score calculation is trained through inductive learning for link prediction.

The link prediction model is composed of two parts: 1) Graph embedding, which involves aggregating node information using the GraphSAGE architecture, and 2) Pair-wise scoring, where a scalar value is generated for each edge or node pair. This scoring is based on the features of the connected nodes after aggregation. The calculation of this score employs various metrics, including L2 distance (Eq. 5), dot product (Eq. 6), cosine similarity (Eq. 7), element-wise product (Eq. 8), and Multilayer Perceptron (MLP) (Eq. 9).

$$score = \sqrt{(h_i - h_j)^2 + (h_i - h_j)^2}\# \tag{5}$$

$$score = h_i^T h_j\# \tag{6}$$

$$score = \frac{h_i^T h_j}{\|h_i\|_2 \odot \|h_j\|_2}\# \tag{7}$$

$$score = SUM(h_i \odot h_j) or\ MEAN(h_i \odot h_j)\# \tag{8}$$

$$score = W(h_i \| h_j) + b\# \tag{9}$$

Where, *score* is the pair-wise score of the edge between node $i,j \in V$; h_i, h_j are graph embeddings for node $i,j \in V$.

The comprehensive process for link prediction using contrastive learning is illustrated in Fig. 4. Notably, this involves constructing two types of graphs: a positive graph using positive examples and a negative graph using negative examples. Both these graphs share the same nodes as the original BMKG. Consequently, this allows for the calculation of a scalar score for each edge or node pair, utilizing the same node representations post-GraphSAGE embedding. This scoring is consistent across various graphs, encompassing the original, incomplete BMKG as well as the newly created positive and negative graphs.

It's important to note that the negative examples are generated from node pairs that are not connected in the current BMKG. These examples must be filtered by eliminating unqualified links that violate the restrictions in the proposed BMKG schema, such as connections that span beyond adjacent layers. Moreover, due to limitations in the dataset, i.e., the BMKG derived from existing inspection reports cannot cover all scenarios, the generated negative examples can be divided into two categories:

1) Authentic negative examples, which are pairs of nodes that should not be linked essentially, such as non-existing components (or elements) and inappropriate repair solutions.
2) Pseudo-negative examples, which are pairs of nodes that might be connected but were not recorded in historical reports. Examples include components (or elements) that were not found to be defective and defects that were not detected.

To mitigate the impact of pseudo-negative examples in contrastive learning, negative examples are randomly generated in the same quantity as positive examples, as recommended in [37]. Then, the binary cross-entropy (BCE), indicated in Eq. 10, is used as the loss function.

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \cdot log(p_i) + (1 - y_i) \cdot log(1 - p_i)]\# \tag{10}$$

Where L represents the average loss for N examples; y_i is the label of the example i; p_i is the predicted probability of the example i.

The effectiveness of the model is assessed using the ROC (Receiver Operating Characteristic Curve) and AUC (Area Under the ROC Curve) metrics. The ROC is a graphical representation that displays the classi-

Original Inspection Reports

Date	Location	Defect	Query	Proposal
01/03/2021	Span 1 - MGE2 (External Face)	Crank Defect - Over 50%	Main Girder Stiffener Crank Section Loss	Cut out corroded section and weld a new plate in its place like for like. (Only use in cases of section loss over 50%.
05/07/2022	Span 2	Gap Between Rail Bearer (RB) Flange and Stool		Insert Packer between RB flange and stool then tack weld in place.
05/07/2022		Loose membrane below Rail Bearer on Abutment		Remove existing membrane and insert a new rubber pad then shim until tight.
26/03/2023	IS5	Large Holes throughout South Caisson		Fill all Holes with Devcon.

Processed Textural Reports

Main Structure	Component	Element	Defect	Repair Action
span 1	main girder external	stiffener lower crank	section loss over 50%	cut out corroded section and weld a new plate in its place like for like (only use in cases of section loss over 50%).
span 2	rail bearer	between flange and stool	gap	insert packer between flange and stool, then fix it in place with tack weld.
abutment 7	bearing below rail bearer	membrane	loose	remove existing membrane and insert a new rubber pad then shim it until tight.
intermediate support 5	south caisson pile	throughout surface	large holes	fill all the holes with Devcon coating product.

Fig. 6. Examples of inspection reports preprocessing for BMKG establishment.

fication model's performance across various thresholds, with the x-axis indicating the False Positive Rate (FPR) (Eq. 11) and the y-axis showing the True Positive Rate (TPR) (Eq. 12).

$$\mathrm{FPR} = \frac{FP}{FP + TN}\# \tag{11}$$

$$\mathrm{TPR} = \mathrm{Recall} = \frac{TP}{TP + FN}\# \tag{12}$$

AUC measures the entire two-dimensional area underneath the entire ROC curve, which denotes the probability that the score of a positive example exceeds that of a negative example when both are randomly sampled from the data, indicated in Eq. 13.

$$AUC = \frac{\sum (p_i, n_j)_{p_i > n_j}}{P^*N}\# \tag{13}$$

Where, P – the number of positive examples; N – the number of negative examples; p_i – the prediction score for a positive example; n_j – the prediction score for a negative example.

Determining the best threshold based on known True Positive Rates (TPRs) and False Positive Rates (FPRs) varies according to specific goals or priorities. For instance, if the aim is to increase the TPR while minimizing the FPR, the ideal threshold is that which corresponds to the point nearest to the top left corner of the ROC plot. Conversely, if the goal is to achieve a balance between TPR and FPR, the preferred threshold is one that optimizes the Youden Index (J) in the ROC curve. In this study, the latter approach has been chosen for determining the most effective threshold for evaluating model performance, as indicated in Eq. 14 and 15.

$$J(t) = TPR(t) - FPR(t)\# \tag{14}$$

$$t^* = \underset{t}{argmax}\, J(t)\# \tag{15}$$

Where, t^* – optimal threshold; t – threshold.

Additionally, precision and F1 score are also utilized to evaluate model performance in this work, as indicated in Eqs. 16 and 17.

$$\mathrm{Precision} = \frac{TP}{TP + FP}\# \tag{16}$$

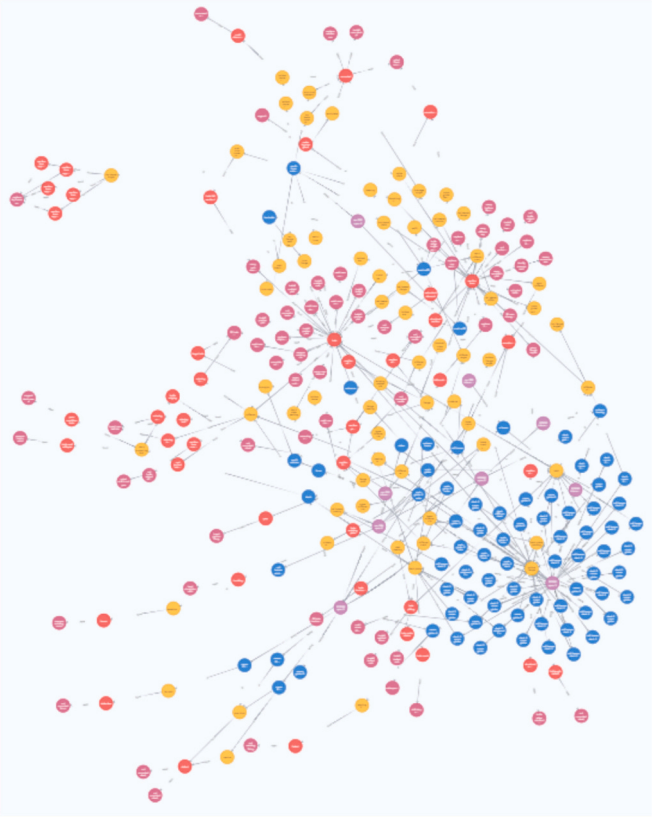


Fig. 7. Established intact BMKG in Neo4j (264 nodes, 451 edges, and 5 node layers).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}\# \quad (17)$$

3.4.3. Implementation framework

The proposed methods can be widely used in bridge maintenance knowledge engineering, covering areas like KG completion, semantic enrichment, node dependency analysis, and compliance checks. These methods can improve the efficiency of engineers in bridge maintenance planning, e.g., by offering automated repair suggestions. This involves transferring knowledge from experienced engineers to junior ones, enabling them to come up with viable repair suggestions using historical data and BMKG. This is made possible through text encoding and GNNs, which use semantic textual similarities, label propagation, and graph embedding. A framework for this implementation is illustrated in Fig. 5, integrating node-layer classification and link prediction into practical bridge maintenance procedures.

Fig. 5 illustrates the framework for applying Node-Layer Classification (NLC) and link prediction in practical use. The framework begins with BMKG derived from textual reports (either unstructured or semi-structured), obtained either manually or through NLP. The BMKG may have missing links due to incomplete data or new inspection findings. Initially, each node's content (words, phrases, sentences) is encoded using a pre-trained text encoder or language model. The derived text embeddings serve as node features. Compared to traditional encoding methods like one-hot encoding, it can enhance node encoding efficiency within the GNNs. Node-layer classification is then achieved using B-DFS and GraphSAGE, aiding in semantic enrichment of the BMKG and compliance checks. Additionally, link prediction is accomplished using GraphSAGE and contrastive learning. The results of node-layer classification can help filter out unqualified negative examples, increasing the accuracy of link prediction.

It's important to note that the context-based predicted node connections may not always be reliable due to data limitations in the textual reports. Therefore, a manual review of link prediction results is required. For instance, a link from a component to an element should be verified against the bridge design diagram. Similarly, proposed repair solutions, i.e., links between defect and repair action nodes, require manual verification as well. Verified predicted connections can be incorporated into maintenance planning or used as text prompts in a plugin, thereby improving the efficiency of engineers' documentation tasks. In contrast, unsatisfactory predictions can be used as new negative examples for further contrastive learning, enhancing the model's performance in a closed-loop system.

4. Proof of concept

4.1. BMKG construction

To verify the practical effectiveness of the proposed methodologies, real-world maintenance reports for multiple railway truss bridges in the UK, which are semi-structured containing 953 records in terms of "date", "location", "defect", "query", and "proposal", are employed to create the BMKG. Then, the textural reports are manually preprocessed (e.g., abbreviation replacement, lowercase conversion, misspelling correction) and manipulated into "main structure", "component", "element", "defect", and "repair action" layers according to the proposed BMKG schema, as illustrated in Fig. 6. Because of irregularity in routine inspection reports, such as blank spaces, special abbreviations, misspellings (or misuse of punctuation), this pre-processing step becomes necessary. It is also promising to be accomplished through NLP techniques (e.g., NER and relations extraction) like in previous research [1–4].

Consequently, a BMKG with 264 nodes and 451 relationships (i.e., edges) is created according to the proposed schema for the following experiments and illustrated via Neo4j, as shown in Fig. 7.

4.2. Node layer classification for incomplete BMKGs

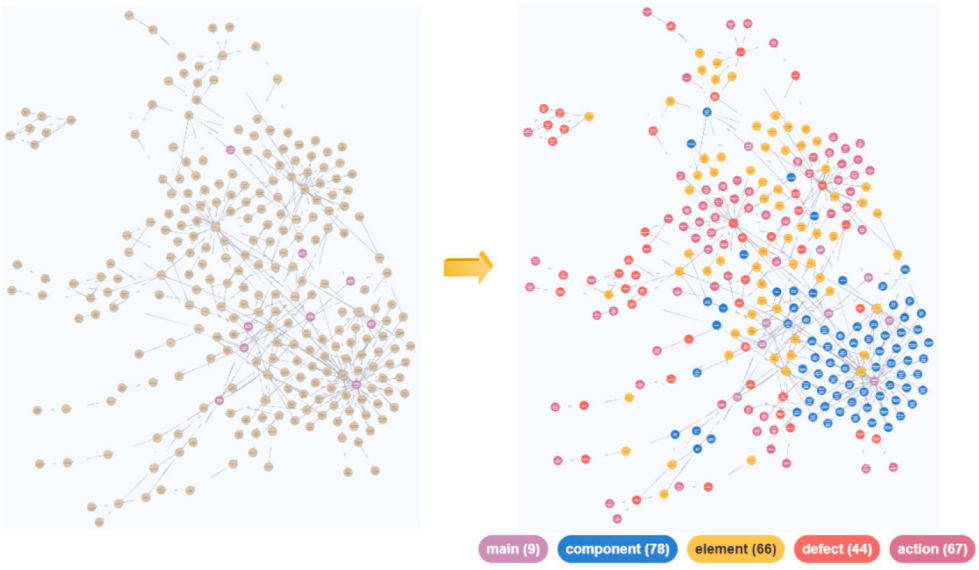
4.2.1. Experiment preparation for NLC

As for a complete directed BMKG, the B-DFS method can effectively determine the layers of every node, provided the hierarchy of at least one node is known. This is depicted in Fig. 8 (1). However, in the case of an incomplete BMKG with missing links, the B-DFS method can only identify the layers of partial nodes due to the absent connections, as illustrated in Fig. 8 (2). To address this challenge, the approach described in Section 3.4.1 is proposed, which integrates text embeddings as initial node features into GraphSAGE, leveraging both contextual similarities and the strengths of GNNs. The experiment is designed to test the proposed approach on a deficient BMKG with missing links. A deficient or incomplete BMKG is created from the original intact BMKG in Fig. 8 by randomly removing a certain proportion of its connection (e.g., 20%).

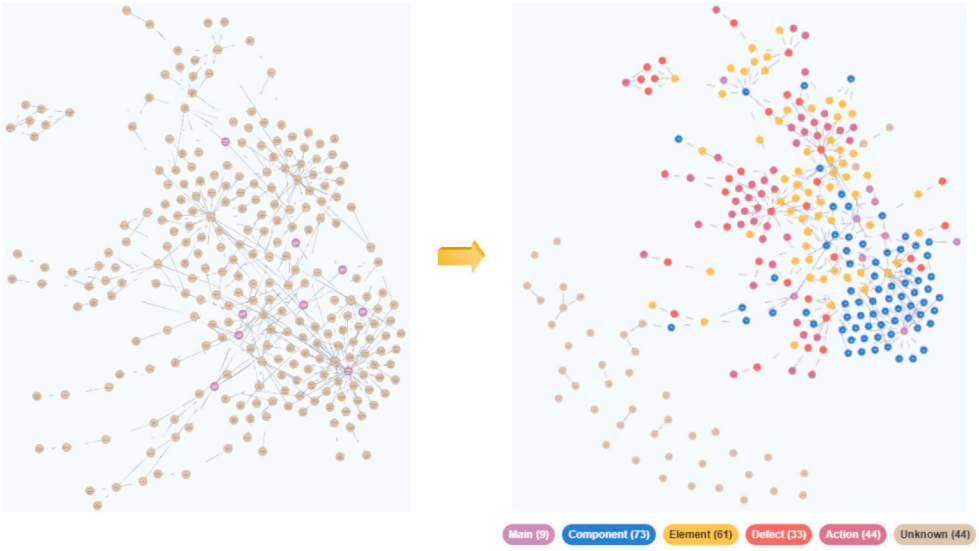
The experiment is carried out on Google CodeLabs utilizing PyTorch and the Deep Graph Library (DGL). A GraphSAGE architecture, as detailed in Table 1, is proposed for graph learning. The NLC model is developed by incorporating a Softmax layer as the classifier at the end of the proposed neural network. The experiment's objective is to identify the layers of unknown nodes in the deficient BMKG with absent connections, as illustrated in Fig. 8 (2), using the information and attributes of the recognized nodes.

4.2.2. Ablation study for NLC

4.2.2.1. Different text encoders for NLC. In graph neural networks (GNNs), the initial node features can significantly affect the model performance in downstream tasks. Different text encoding methods produce various embeddings, leanding to diverse outcomes. This



(1) intact BMKG after B-DFS (fully successful node-layer classification)



(2) BMKG with missing links after B-DFS (partially successful node-layer classification)

Fig. 8. NLC for intact and deficient BMKGs after B-DFS.

Table 1
GraphSAGE model architecture for NLC.

Search depth	Hidden layer dimension	Aggregation	Activation function	Dropout	Classifier
$k = 3$	128	mean	ReLU	0.3	Softmax

Table 2
Training configuration for NLC.

Optimiser	Learning rate	Weight_decay	AMSGrad	Epochs
AdamW	0.0001	0.001	True	3000

ablation study aims to identify which text encoder can generate the best performance in the proposed BMKG (shown in Fig. 7) for NLC. Then, an experiment was conducted to compare traditional word embedding methods and multiple pre-trained LLMs in node feature initialization for NLC. Traditional word embedding methods include bag of words (BOW) [18], TF-IDF [19], Skip-gram [38], and GloVe [21], which are trained on the corpus derived from the same maintenance reports used to establish the BMKG. LLM-based text encoders include bart-base (BART) [39], bert-base-uncased (BERT) [40], CLIP text encoder [41], deberta-v3-base (DeBERTaV3) [42], gtr-t5-base (GTR) [43], text-embedding-ada-002 (OpenAI) [44], sentence-t5-base (T5) [45], UAE-Large-V1(UAE) [46], and xlm-mlm-en-2048 (XLM) [47].

In the experiment, the identified nodes after B-DFS and their layer information serve as the training set, while the remaining unknown nodes are treated as the test set. As illustrated in Fig. 8, there are 5 classes of node labels, i.e., main, component, element, defect, action. The experiment configuration is detailed in Table 2.

The experimental results are presented in Fig. 9 and 10. As can be seen, multiple LLM text encoders, like BERT, DeBERTaV3, GTR, OpenAI-ada, UAE, overperform all the traditional word embedding methods in the experiment. Especially, BERT achieves the top test accuracy of 97.73%. Hence, BERT is adopted as the text encoder to initialize node

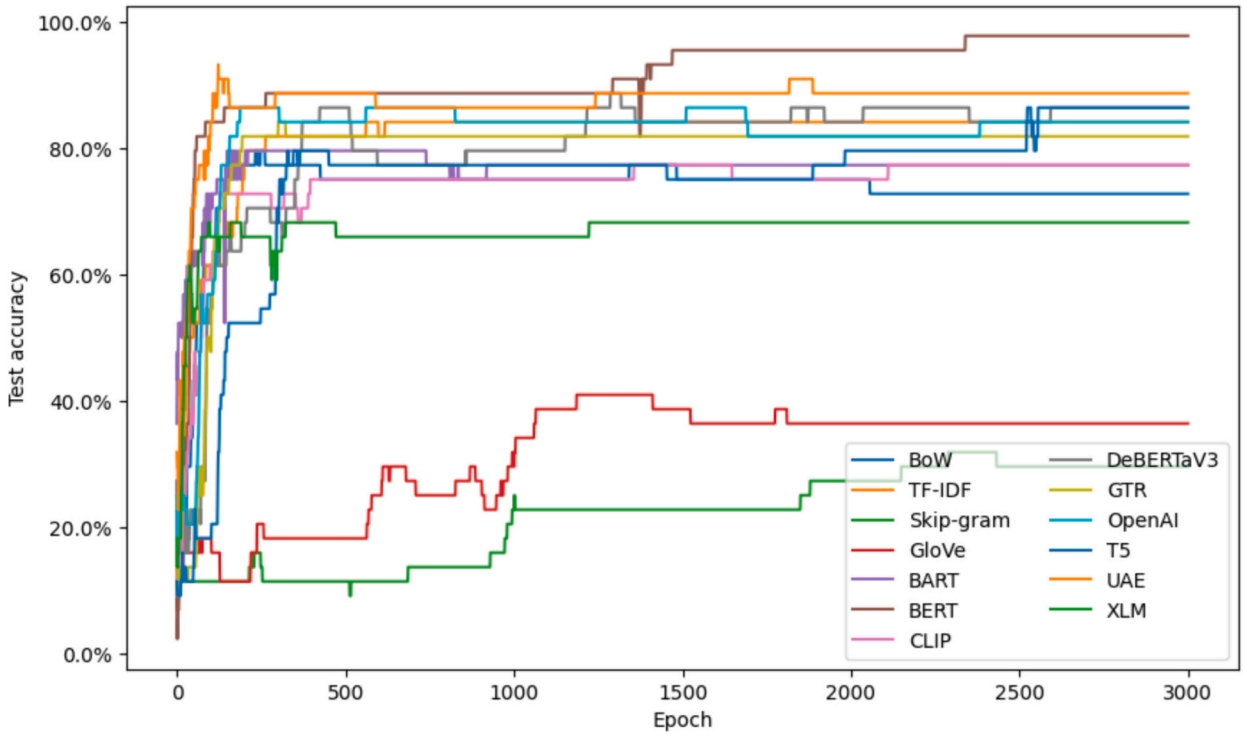


Fig. 9. Training process based on different text encoders for NLC.

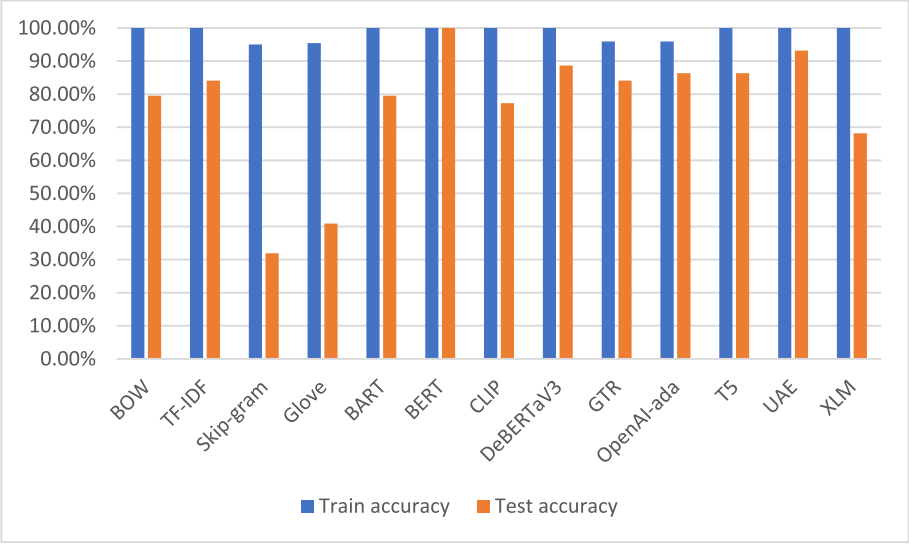


Fig. 10. Training and test performance based on different text encoders for NLC.

features for following NLC experiment. Additionally, most of the LLMs used in the experiment perform well for the NLC task (i.e., 5-class classification) in the target BMKG, demonstrating the effectiveness of proposed NCL method (see section 3.4.1) and GraphSAGE architecture (detailed in Table 1). The difference in results is most likely due to the different training data used for each LLM. The BERT model's superior performance could be attributed to its robust pre-training on a corpus closely related to bridge maintenance textual reports, which enhances its ability to generalize across this specific task.

4.2.2.2. Varying BMKG completeness for NLC. An experiment is conducted for NLC on incomplete BMKGs across a range of missing connections from 10% to 50%, leading to 20, 44, 62, 106 and 121 unknown node labels after B-DFS, respectively. The experiment configuration is detailed in Table 2 and the initial node features are generated by text encoding through the pre-trained BERT model. The training and test accuracies are displayed in Fig. 11.

As can be seen in Fig. 11, test accuracy decreases as the missing rate of connections increases in the BMKG, indicating that the robustness of the proposed NLC approach is significantly affected by the completeness of the BMKG. Furthermore, in the deficient BMKG with 50% missing links (leading to 121 unknown nodes of 234 nodes after B-DFS), test accuracy can achieve an accuracy of over 86%, demonstrating the effectiveness and robustness of the proposed NLC method and GraphSAGE architecture. It is also evident that ensuring a lower rate of missing connections can lead to better performance in downstream tasks.

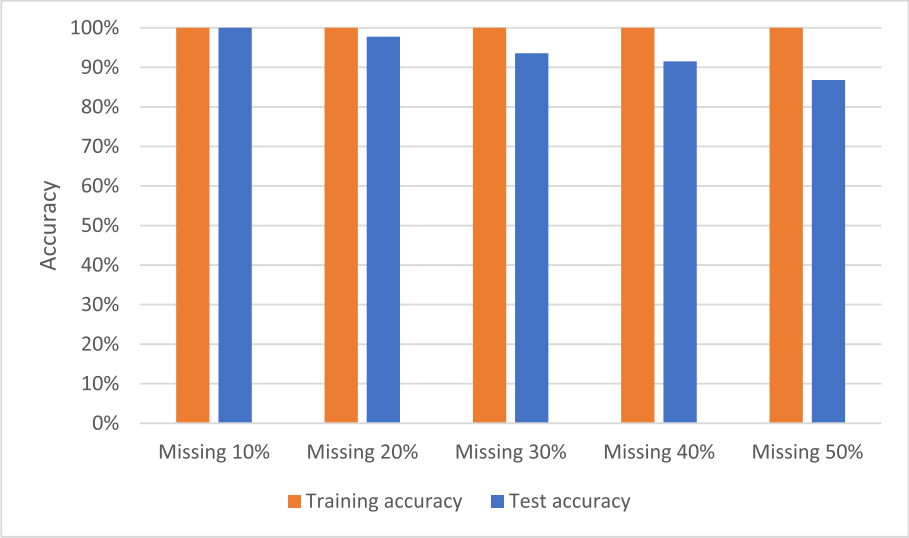


Fig. 11. NLC performance based on varying BMKG completeness.

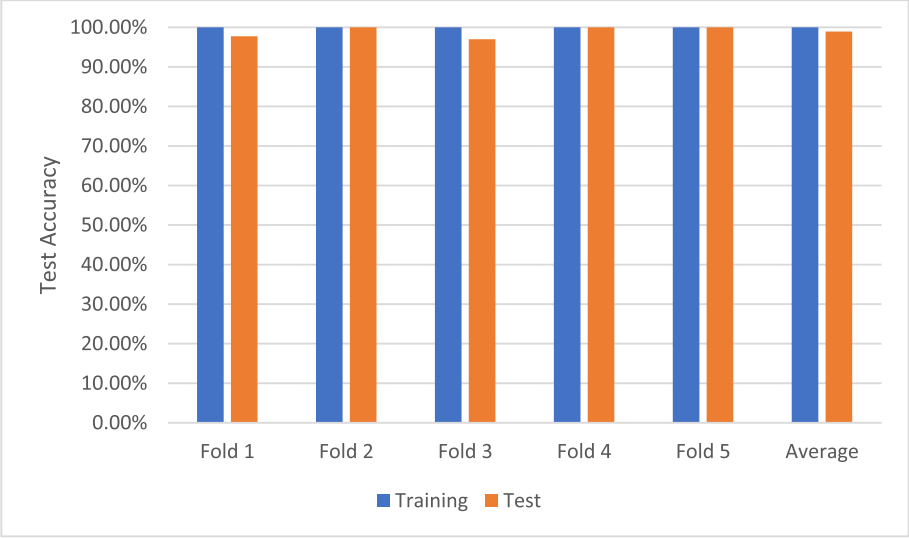


Fig. 12. 5-fold cross-validation for NLC.

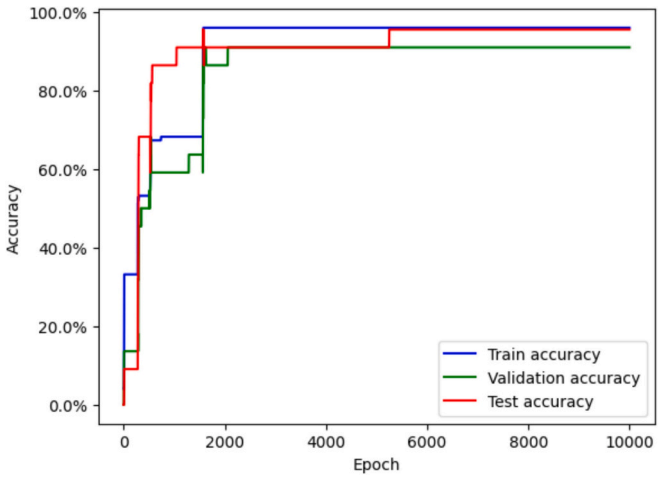


Fig. 13. Training, validation, and test performance in a deficient BMKG for NLC.

Therefore, it is essential to consider the impact of missing links' proportions when applying the approach to the BMKG generated from the real-world maintenance textual reports that usually has data deficiencies and irregularities.

4.2.3. *Cross-validation for NLC*

5-fold cross-validation (i.e., removing 20% of links in each fold) is performed to evaluate the model's generalization capability for NLC. The experiment follows the configuration detailed in Table 2. The initial nodes features are generated by encoding through the pre-trained BERT model. The results are shown in Fig. 12, achieving an average accuracy of 98.94% on the test set. This demonstrates that the proposed NLC method and GraphSAGE with node features initialized from BERT text encoding cannot only achieve excellent performance in the target BMKG but also exhibit strong generalization capability. Hence, the approach is promising for significantly improving the efficiency, accuracy, and accessibility of knowledge engineering and documentation work in bridge maintenance, including tasks such as compliance checking and knowledge graph generation.

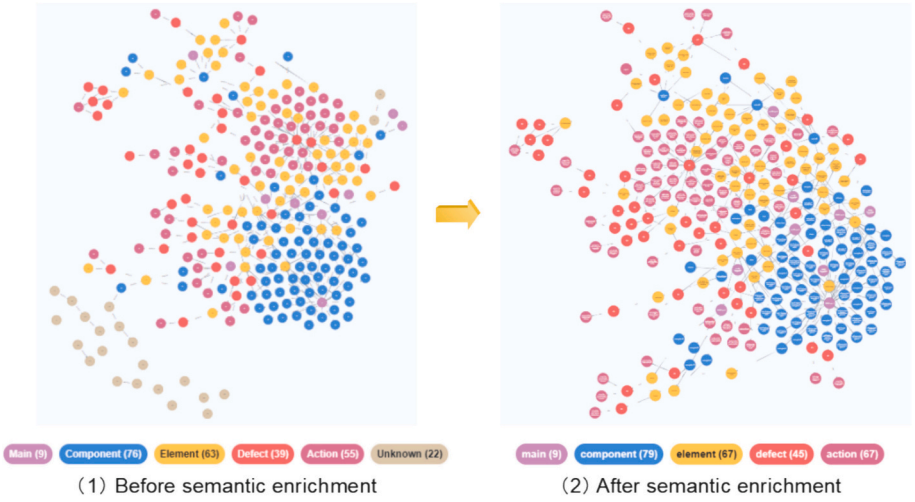


Fig. 14. Deficient BMKG before and after semantic enrichment via NLC.

Table 3
GraphSAGE architecture for link prediction.

Search depth	Hidden layer dimension	Aggregation	activation function
$k = 3$	128	pool	Sigmoid

Table 4
Training configuration for link prediction.

Optimiser	Learning rate	Weight_decay	StepLR		Epochs
			step_size	gamma	
AdamW	1×10^{-5}	0.01	10	0.98	2000

4.2.4. Semantic enrichment

BMKGs derived from real-world maintenance reports are usually deficient with absent connections due to data deficiencies and irregularities, which results in missing node layer information of many nodes after B-DFS. Hence, it would be beneficial to enrich such semantic information in an incomplete BMKG using various methods. An experiment is conducted to validate the proposed approach for such NLC semantic enrichment tasks. The deficient BMKG is created with 10% of its links missing randomly, leading to 22 problematic nodes without layer information after B-DFS. The other nodes with known layers are split into the training set (i.e., 220 nodes) and validation set (22 nodes), respectively, while the problematic nodes are adopted as the test set (i.e., 20 nodes). The experiment configuration is detailed in Table. 2 with the number of epochs set to 10,000. Initial node features are derived by text encoding through the pre-trained BERT model. Finally, the model training process and the experiment results are shown in Fig. 13.

In Fig. 13, training, validation, and test accuracies can achieve 95.91%, 90.91%, and 95.45%, respectively. Notably, due to a limited number of samples, each back-propagation update during training can cause significant changes in accuracies, leading to the stepwise pattern observed. This indicates the sensitivity of the model to the data volume. Especially for a relatively small dataset, each data point can disproportionately affect the training process.

Furthermore, the node-layer results of the BMKG before and after semantic enrichment are displayed in Fig. 14. It demonstrates the effectiveness of the proposed method in semantic enrichment (i.e., node layer information) in a deficient BMKG with missing links by leveraging available textual information through GNNs. As can be seen, this approach is beneficial for practical applications in bridge maintenance documentation work, such as facilitating easier retrieval and understanding of data, ensuring that all critical information is captured and structured, and reducing the likelihood of errors and omissions.

4.3. Link prediction for incomplete BMKGs

4.3.1. Experiment preparation for link prediction

The GraphSAGE architecture for link prediction is detailed in Table 3, where the Softmax layer in the NLC model is replaced with a function for calculating pairwise scores. The experiment is conducted on Google CodeLabs using PyTorch and the Deep Graph Library (DGL).

In the experiment, a portion of links were randomly removed from the intact BMKG in Fig. 7 to simulate a deficient BMKG due to the data deficiencies or irregularities commonly found in practical maintenance reports. The positive and negative examples are generated using the method proposed in Section 3.4.2. Binary cross-entropy (BCE), as indicated in Eq. 9, is used as the loss function.

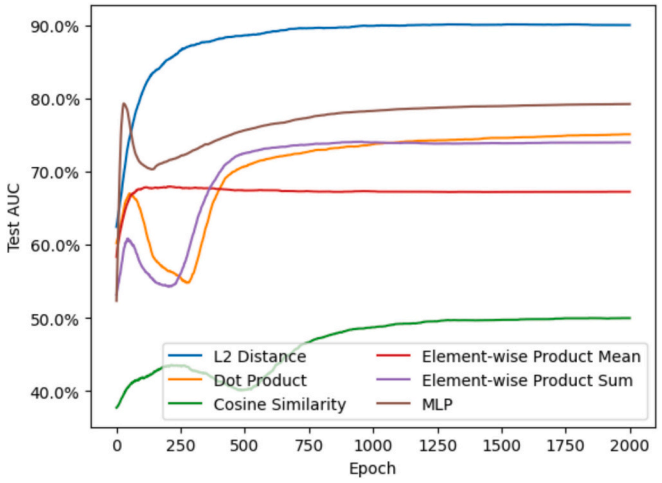


Fig. 15. Training process based on different pairwise scoring methods for link prediction.

4.3.2. Ablation study for link prediction

4.3.2.1. Different edge pair-scoring methods for link prediction. To identify the most effective edge pair-scoring method for link prediction, five different metric-based pairwise score calculators are compared in the experiment, including L2 distance, dot product, cosine similarity,

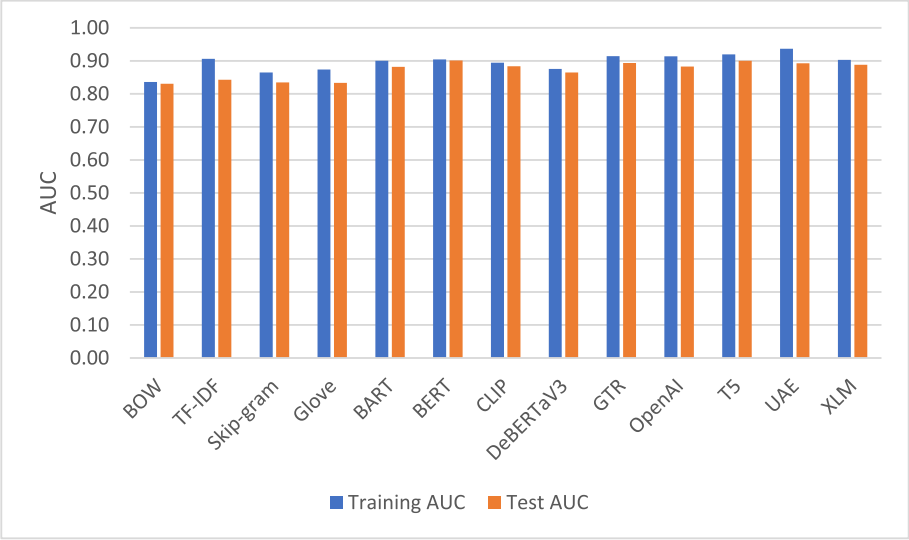


Fig. 16. Training and test performance based on different text encoders for link prediction.

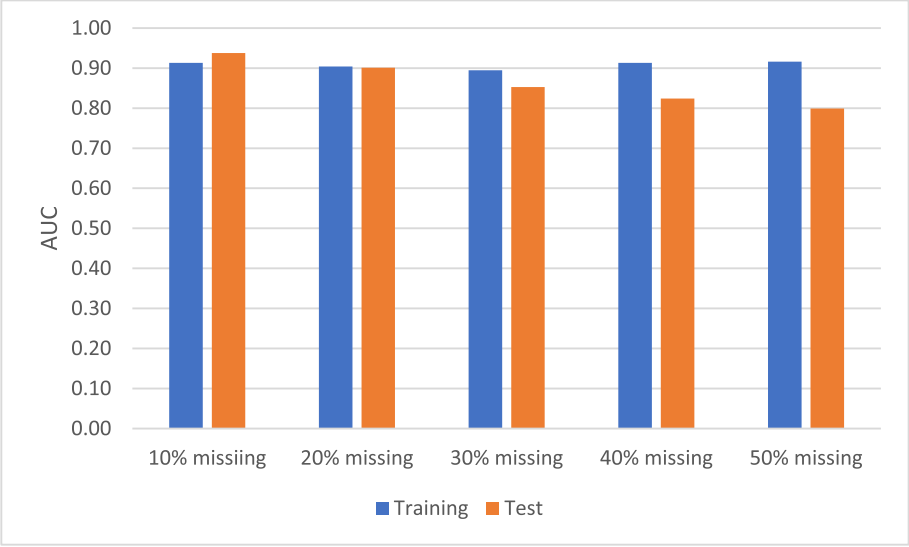


Fig. 17. Link prediction performance based on varying BMKG completeness.

Hadamard product (both summation and average), and an MLP based on concatenation. The incomplete BMKG is created by randomly removing 20% of the links from the original BMKG (Fig. 7). The node features are initialized through a pre-trained BERT model (i.e., bert-base-uncased [40]). The experimental configuration is detailed in Table 4.

The model performance for link prediction is evaluated using the AUC metric (Eq. 12). The results are presented in Fig. 15. As can be seen, the choice of edge score calculation method can significantly affect model performance. The L2 distance (i.e., Euclidean distance) achieves the highest test AUC (i.e., 0.90), showing its superior effectiveness over the other metric-based pairwise score calculators in capturing the underlying relationships within the proposed BMKG. Consequently, the L2 distance method is employed in the following experiment.

4.3.2.2. Different text encoders for link prediction. An experiment is conducted to compare traditional word embedding methods and multiple pre-trained LLMs in node feature initialization for link prediction, including the models utilized in Section 4.2.2.1. The experiment configuration is detailed in Table 4. The results are shown in Fig. 16.

As can be seen from Fig. 16, most LLM text encoders overperform the traditional word embedding methods in the experiment for link prediction on test AUC. BERT stands out with the best AUC (i.e., 0.90) on the test set, as well as the minimal AUC difference between training and test sets, indicating its robustness and reliability. Consequently, the pre-trained BERT model is employed as the text encoder for node feature initialization in the following link prediction experiment. Additionally, the other LLM text encoders used in the experiment demonstrate excellent performance and strong generalization capabilities as well, which are promising to be further improved by tuning and regularization.

4.3.2.3. Varying BMKG completeness for link prediction. An experiment is conducted for link prediction on incomplete BMKGs across a range of missing links from 10% to 50%, i.e., 45, 90, 135, 180, and 225 edges, respectively. The configuration is detailed in Table 4. The initial node features are generated by text encoding through the pre-trained BERT model. The results are shown in Fig. 17.

As can be seen from Fig. 17, it demonstrates the impact of varying levels of BMKG completeness on the model's performance. As the proportion of missing connections increases from 10% to 50%, a clear

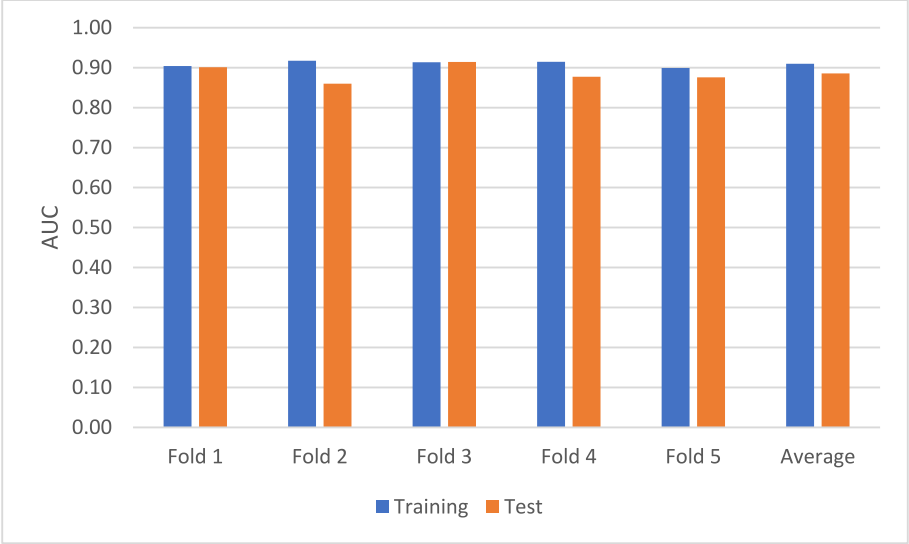


Fig. 18. 5-fold cross-validation for link prediction.

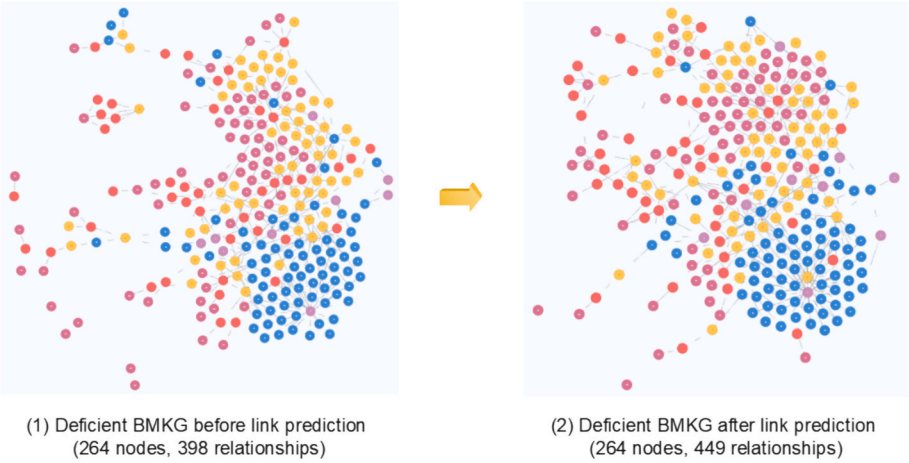


Fig. 19. BMKG completion via link prediction.

decline can be observed from test accuracies. This trend underscores the sensitivity of link prediction models to the availability of complete data in BMKGs. Therefore, in practical applications of link prediction, it is crucial to consider the impact of BMKG completeness for achieving more robust model performance.

4.3.3. *Cross-validation for link prediction*

To evaluate the model's generalization capability for link prediction, a 5-fold cross-validation is performed by removing 20% of links in each fold, generating different training and test sets. The experiment follows the configuration detailed in Table 4. The results are presented in Fig. 18, achieving an average AUC of 0.88 on the test set. This demonstrates that the proposed contrastive link prediction method with BERT text embeddings and the L2 pairwise scoring method, is not only effective in predicting node connections for the target BMKG but also exhibits excellent generalization capability, which is crucial for practical applications.

4.3.4. *BMKG completion*

KG completion plays a crucial role in knowledge engineering and has various practical usages, such as enhancing search engine outcomes, powering recommendation systems, generating scenarios, knowledge discovery, data integration and interoperability [48]. To assess the effectiveness of the proposed link prediction method in BMKG completion, an experiment is conducted following the configuration detailed in Table 4. The Youden index (J), as depicted in Eqs. 13 and 14, is employed as the threshold for pairwise scores to determine whether to accept predicted node connections. An incomplete BMKG is created by removing 10% connections (i.e., 45 edges) randomly for the experiment, and the split of train, validation, and test sets is 80%:10:10%. The updated BMKG after link prediction is shown in Fig. 19.

Moreover, because the proposed contrastive link prediction essentially performs as binary classification (with J as the threshold of pairwise scores), the result confusion matrix can be displayed in Fig. 20 and the prediction performance can be evaluated as shown in Table 5.

As can be seen from Fig. 19, a significant portion of the missing links in the incomplete BMKG have been accurately reestablished after link prediction, i.e., 42 out of 45 edges, achieving a high true positive rate. The performance metrics in Table 5 showcase an AUC of 0.91 and an optimal threshold of 0.42. The model's accuracy stands at 0.84, with a precision of 0.79 and an F1 score of 0.86. Notably, the recall, i.e., true positive rate (TPR), is 0.93, indicating the model's proficiency in identifying positive examples. The false positive rate (FPR) is 0.24, reflecting the model's balance between sensitivity and specificity.

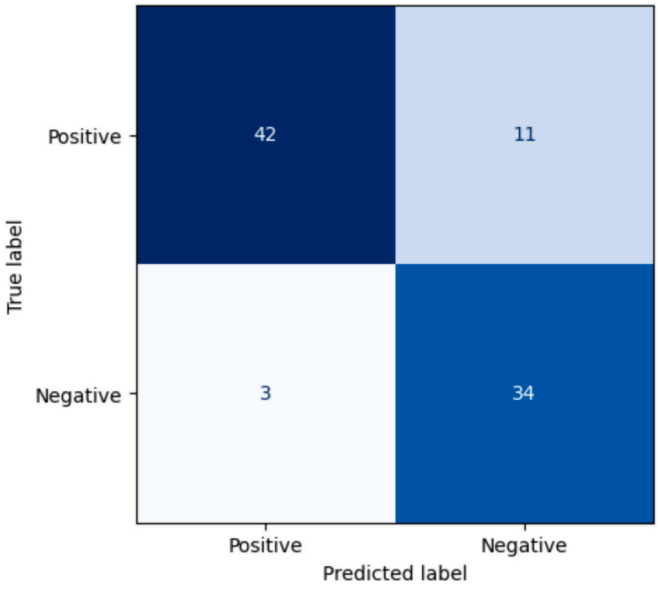


Fig. 20. Confusion matrix for link prediction.

Table 5
Evaluation of link prediction based on the Youden threshold.

AUC	Threshold (*J*)	Accuracy	Precision	F1 score	TPR (Recall)	FPR
0.91	0.42	0.84	0.79	0.86	0.93	0.24

The results demonstrate the effectiveness of the proposed link prediction method in BMKG completion, which can support various practical applications in bridge maintenance knowledge engineering, such as maintenance schedule and repair recommendations, data augmentation, and anomaly detection. By accurately predicting missing links, the method can help to maintain comprehensive and up-to-date BMKGs, ultimately enhancing decision-making in bridge maintenance. It is also worth noting that a few negative example edges were incorrectly identified as positive. This is partially attributed to the presence of pseudo-negative examples (see Section 3.4.2), which requires further exploration in the future.

4.4. Comparison with cutting-edge models

4.4.1. Comparative analysis for NLC

Multiple cutting-edge models are employed in this section to compare with the proposed GraphSAGE model for NLC, including GCN [45], GAT [46], RGCN [47], and GINconv [49]. The target deficient BMKG is generated by removing 20% links randomly, leading to 41 unknown nodes after B-DFS. The experiment is conducted on Google CodeLabs using PyTorch and the Deep Graph Library (DGL), with the model architectures recommended in their respective papers. The configurations are adjusted as needed for the target BMKG, such as setting the number of layers to 5. The results are shown in Fig. 21. As can be seen, the proposed GraphSAGE model overperforms the other SOTA models in test accuracy for node layer classification in BMKG, demonstrating the superiority of the proposed NLC approach.

4.4.2. Comparative analysis for link prediction

Different state-of-art models and approaches are compared with the proposed contrastive link prediction method based on GraphSAGE, including GCN [45], RGCN [47], TAGCN [48], TransE [50], and DistMult [51]. The target deficient BMKG is created by removing 20% links randomly, leading to 90 missing links. The experiment is conducted on Google CodeLabs using PyTorch and the Deep Graph Library (DGL), with the cutting-edge model architectures recommended in their respective papers. The configurations are adjusted as needed for the target BMKG, such as setting the number of layers to 5. The results are shown in Fig. 22.

As can be seen, the proposed GraphSAGE link prediction model by leveraging contrastive learning overperforms the other cutting-edge models and approaches in test AUC, demonstrating the superiority of the proposed link prediction method in the BMKG. Additionally, negative examples in TransE and DistMult are generated via head and tail embeddings' corruption, whereas negative examples in GCN, RGCN, TAGCN, and GraphSAGE are proposed with non-existent edges. The experiment results further validate the effectiveness of using non-existent edges for generating negative examples in downstream tasks within the BMKG.

5. Discussion

The proposed KG schema for bridge maintenance is well-grounded in practical applications, because real-world bridge inspections and repairs rely on the step-by-step details of defect descriptions, such as locations,

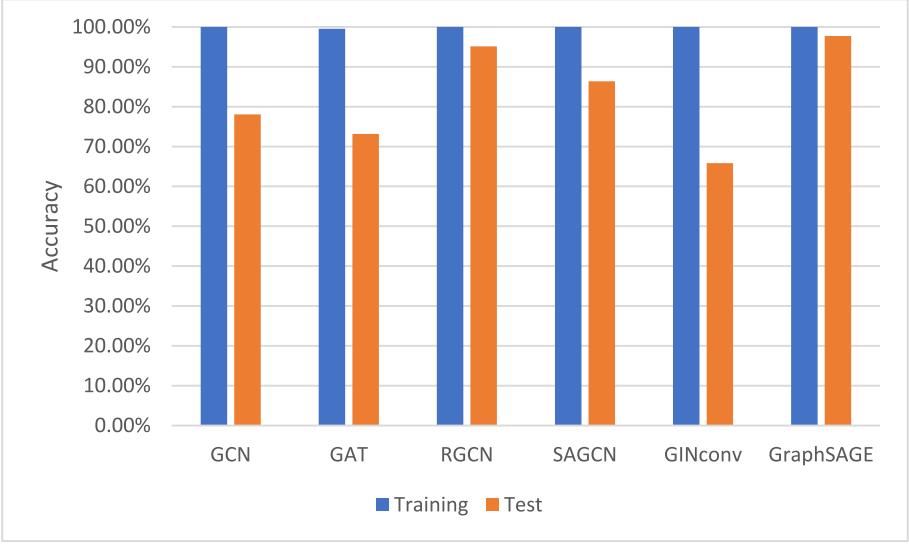


Fig. 21. Comparative analysis for NLC.

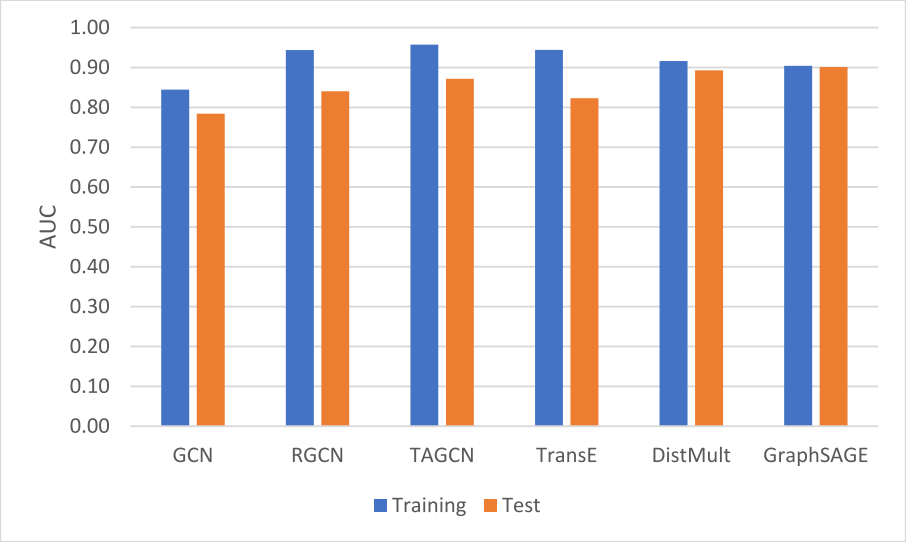


Fig. 22. Comparative analysis for link prediction.

types, and severity, which are typically summarized in routine textual reports. Hence, the textual content of these reports plays a significant role in knowledge engineering for bridge maintenance. For instance, clauses that begin with action verbs like 'insert', 'replace', 'remove', 'install', and 'weld' often indicate repair actions. Similarly, specific phrases such as 'section loss', 'bolt missing', 'corrosion', 'hole', and 'pitting' are commonly used to describe defects. Additionally, the relationships between defects and repair solutions tend to follow regular patterns based on specific phrase pairings. These principles are also applicable to the components and elements of bridge structures and their interdependencies. However, the utility of bridge maintenance reports for knowledge mining is usually hindered by data gaps and inconsistencies.

Moreover, traditional word embedding methods like one-hot or word2vec are limited in capturing the contextual relationships between words in a sentence. They often fail to represent the nuanced meanings and dependencies that arise from word order and syntax, which are crucial for accurately understanding and processing bridge maintenance textual reports. As a result, more advanced techniques, such as text encoding through LLMs, have been developed to address these limitations by generating contextual embeddings for node feature initialization in the BMKG. Furthermore, the GraphSAGE-based NLC and contrastive link prediction approaches are proposed to overcome the challenges posed by data deficiencies or irregularities in practical maintenance reports.

Although the GraphSAGE-based NLC and link prediction approaches proposed in this study has proven effective in semantic enrichment and completing a deficient BMKG, there remains a need for future work in several areas. First, the currently established BMKG is limited in scope, and there is a lack of bridge maintenance-oriented data in academic research. Therefore, it is essential to develop a comprehensive corpus from industrial practices to facilitate the development and validation of AI-assisted approaches in bridge knowledge engineering, including the approaches proposed in this study.

Second, the current method of generating negative examples through simple random pairing leads to numerous pseudo-negative examples. In the study, this issue is addressed with a closed-loop framework that includes manual review, as shown in Section 3.4.3. However, other methods, such as utilizing node attributes, considering common neighbours, sampling from low-probability edges, or applying adversarial techniques may help to mitigate this problem and reduce overfitting in link prediction. Furthermore, exploring transductive learning methods for evolving BMKGs offers a promising direction for future research.

6. Conclusion

By integrating structural designs and real-world maintenance reports into the proposed BMKG schema, the paper addresses the need for a dynamic knowledge graph adapting to isolated nodes and missing connections within bridge routine inspection, which also shows superiority in documenting the domain expertise of bridge maintenance. Then, the research demonstrates that the proposed GrapshSAGE-based NLC and contrastive link-prediction approaches by leveraging contextual embeddings through LLMs can achieve satisfactory performance and generalizability on the deficient BMKGs for semantic enrichment and KG completion. The performance is validated through ablation study and cross-validation.

These approaches can improve the graph data mining in BMKG and facilitate many practical applications in bridge maintenance knowledge engineering, such as semantic enrichment, compliance checks, dependency analysis, and repair recommendations. Experimentally, it is also observed that the pre-trained BERT model, i.e., bert-base-uncased [40], outperforms many other LLMs in text encoding for NLC and link prediction tasks. Similarly, the pairwise scoring method via L2 distance (i.e., Euclidean distance) shows superior effectiveness compared to other metrics for link prediction. A practical implementation framework that integrates the proposed approaches has been designed for effective routine bridge maintenance. Additionally, the areas for future enhancements and potential research directions are thoroughly discussed in the discussion section of the study.

CRediT authorship contribution statement

Yan Gao: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Conceptualization. **Guanyu Xiong:** Investigation, Formal analysis, Data curation. **Haijiang Li:** Supervision, Project administration, Methodology. **Jarrod Richards:** Validation, Resources, Investigation, Data curation.

Declaration of competing interest

The authors declared that they have no conflicts of interest in this work.

Data availability

Data will be made available on request.

Acknowledgement

This work is part of the knowledge transfer partnerships (KTP) project – DIGIBRIDGE: BIM and Digital Twins in support of Smart Bridge Structural Surveying. The project receives funding from Innovate UK with reference number 10003208.

References

[1] R. Li, T. Mo, J. Yang, D. Li, S. Jiang, D. Wang, Bridge inspection named entity recognition via BERT and lexicon augmented machine reading comprehension neural model, Adv. Eng. Inform. 50 (September) (2021) 101416, https://doi.org/10.1016/j.aei.2021.101416.
[2] K. Liu, N. El-Gohary, Ontology-based semi-supervised conditional random fields for automated information extraction from bridge inspection reports, Autom. Constr. 81 (2017) 313–327, https://doi.org/10.1016/j.autcon.2017.02.003.
[3] K. Liu, N. El-Gohary, Semantic neural network ensemble for automated dependency relation extraction from bridge inspection reports, J. Comput. Civ. Eng. 35 (4) (2021), https://doi.org/10.1061/(asce)cp.1943-5487.0000961.
[4] R. Li, et al., Joint extraction of entities and relations via an entity correlated attention neural model, Inf. Sci. 581 (2021) 179–193, https://doi.org/10.1016/j.ins.2021.09.028.
[5] Knowledge Engineering: What it Means, Examples, Accessed: Aug. 09, 2023. [Online]. Available: https://www.investopedia.com/terms/k/knowledge-engineering.asp, 2024.
[6] Y. Zhang, J. Liu, K. Hou, Building a knowledge base of bridge maintenance using knowledge graph, Adv. Civil Eng. 2023 (2023), https://doi.org/10.1155/2023/6047489.
[7] G. Ren, R. Ding, H. Li, Building an ontological knowledgebase for bridge maintenance, Adv. Eng. Softw. 130 (July) (2019) 24–40, https://doi.org/10.1016/j.advengsoft.2019.02.001.
[8] K. Liu, N. El-Gohary, Bridge deterioration knowledge ontology for supporting bridge document analytics, J. Constr. Eng. Manag. 148 (6) (2022) 1–14, https://doi.org/10.1061/(asce)co.1943-7862.0002210.
[9] G. Lee, S. Chi, Graph-based clustering of bridge management system data for bridge maintenance cost estimation, in: EG-ICE 2023 Workshop on Intelligent Computing in Engineering, London, UK, 2023.
[10] C. Wu, P. Wu, J. Wang, R. Jiang, M. Chen, X. Wang, Ontological knowledge base for concrete bridge rehabilitation project management, Autom. Constr. 121 (May) (2021) 103428, https://doi.org/10.1016/j.autcon.2020.103428.
[11] R. Li, T. Mo, J. Yang, S. Jiang, T. Li, Y. Liu, Ontologies-based domain knowledge modeling and heterogeneous sensor data integration for bridge health monitoring systems, IEEE Trans. Industr. Inform. 17 (1) (2021) 321–332, https://doi.org/10.1109/TII.2020.2967561.
[12] J. Yang, et al., Intelligent bridge management via big data knowledge engineering, Autom. Constr. 135 (January) (2022) 104118, https://doi.org/10.1016/j.autcon.2021.104118.
[13] L. Kolbeck, S. Vilgertshofer, A. Borrmann, Graph-based mass customisation of modular precast bridge systems – Methodology for kit development and algorithmic design, in: EG-ICE 2023 Workshop on Intelligent Computing in Engineering, London, UK, 2023.
[14] P. Zhao, W. Liao, Y. Huang, X. Lu, Intelligent design of shear wall layout based on graph neural networks, Adv. Eng. Inform. 55 (January) (2023) 101886, https://doi.org/10.1016/j.aei.2023.101886.
[15] P. Zhao, W. Liao, Y. Huang, X. Lu, Intelligent beam layout design for frame structure based on graph neural networks, J. Building Eng. 63 (PA) (2023) 105499, https://doi.org/10.1016/j.jobe.2022.105499.
[16] B. Sanchez-Lengeling, E. Reif, A. Pearce, A.B. Wiltschko, A gentle introduction to graph neural networks, Distill 6 (9) (Sep. 2021) e33, https://doi.org/10.23915/DISTILL.00033.
[17] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, P.S. Yu, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2021) 4–24, https://doi.org/10.1109/TNNLS.2020.2978386.
[18] J. Zhou, et al., Graph neural networks: a review of methods and applications, AI Open 1 (April) (2020) 57–81, https://doi.org/10.1016/j.aiopen.2021.01.001.
[19] F.C. Collins, A. Braun, M. Ringsquandl, D.M. Hall, A. Borrmann, Assessing IFC classes with means of geometric deep learning on different graph encodings, in: Proceedings of the 2021 European Conference on Computing in Construction vol. 2, 2021, pp. 332–341. Rhodes, Greece, on July 19–28, 10.35490/ec3.2021.168.
[20] Z. Wang, R. Sacks, T. Yeung, Exploring graph neural networks for semantic enrichment: Room type classification, Autom. Constr. 134 (December) (2022) 104039, https://doi.org/10.1016/j.autcon.2021.104039.
[21] L. Xia, Y. Liang, P. Zheng, J. Leng, Maintenance planning recommendation of complex industrial equipment based on knowledge graph and graph neural network, Reliab. Eng. Syst. Saf. 232 (December) (2023) 109068, https://doi.org/10.1016/j.ress.2022.109068.
[22] Y. Zhang, R. Jin, Z.H. Zhou, Understanding bag-of-words model: a statistical framework, Int. J. Mach. Learn. Cybern. 1 (1–4) (2010) 43–52, https://doi.org/10.1007/s13042-010-0001-0.
[23] S. Qaiser, R. Ali, Text mining: use of TF-IDF to examine the relevance of words to documents, Int. J. Comput. Appl. 181 (1) (2018) 25–29, https://doi.org/10.5120/ijca2018917395.
[24] K.W. Church, Emerging Trends: Word2Vec, Nat. Lang. Eng. 23 (1) (2017) 155–162, https://doi.org/10.1017/S1351324916000334.
[25] J. Pennington, R. Socher, C.D. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014, pp. 1532–1543, https://doi.org/10.3115/v1/D14-1162.
[26] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Trans. Assoc. Comput. Linguist. 5 (2017) 135–146, https://doi.org/10.1162/tacl_a_00051.
[27] J. Devlin, M.W. Chang, K. Lee, K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in: NAACL HLT 2019–2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference vol. 1, 2019, pp. 4171–4186, https://doi.org/10.18653/v1/N19-1423, no. Mlm.
[28] C. Raffel, et al., Exploring the limits of transfer learning with a unified text-to-text transformer, J. Mach. Learn. Res. 21 (2020) 1–67. https://dl.acm.org/doi/abs/10.5555/3455716.3455856.
[29] H. Su, et al., One embedder, any task: instruction-finetuned text embeddings, in: Proceedings of the Annual Meeting of the Association for Computational Linguistics, Toronto, Canada, 2023, pp. 1102–1121, https://doi.org/10.18653/v1/2023.findings-acl.71.
[30] Introducing ChatGPT, Accessed: Jan. 13, 2024. [Online]. Available: https://openai.com/blog/chatgpt, 2024.
[31] H. Touvron, et al., Llama 2: Open Foundation and Fine-Tuned Chat Models [Online]. Available: http://arxiv.org/abs/2307.09288, 2023.
[32] H. Touvron, et al., LLaMA: Open and Efficient Foundation Language Models [Online]. Available: http://arxiv.org/abs/2302.13971, 2023.
[33] S. Zhang, H. Tong, J. Xu, R. Maciejewski, Graph convolutional networks: a comprehensive review, Comput. Soc. Netw. 6 (1) (2019), https://doi.org/10.1186/s40649-019-0069-y.
[34] W.L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems vol. 2017, 2017, pp. 1025–1035. Decem, no. Nips.
[35] Link Prediction using Graph Neural Networks — DGL 1.1.1 Documentation, Accessed: Aug. 09, 2023. [Online]. Available: https://docs.dgl.ai/tutorials/blitz/4_link_predict.html#sphx-glr-tutorials-blitz-4-link-predict-py, 2024.
[36] Link Prediction Pipelines - Neo4j Graph Data Science, Accessed: Jan. 29, 2024. [Online]. Available: https://neo4j.com/docs/graph-data-science/current/machine-learning/linkprediction-pipelines/link-prediction/, 2024.
[37] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, J. Dean, Distributed representations ofwords and phrases and their compositionality, in: Advances in Neural Information Processing Systems, 2013, pp. 1–9, https://doi.org/10.5555/2999792.2999959.
[38] Facebook/bart-base·Hugging Face, Accessed: Jan. 13, 2024. [Online]. Available: https://huggingface.co/facebook/bart-base, 2024.
[39] Bert-base-uncased·Hugging Face, Accessed: Jan. 13, 2024. [Online]. Available: https://huggingface.co/bert-base-uncased, 2024.
[40] openai/CLIP: CLIP (Contrastive Language-Image Pretraining), Predict the most relevant text snippet given an image, 2024. Accessed: Jan. 13, 2024. [Online]. Available: https://github.com/openai/CLIP.
[41] microsoft/deberta-v3-base·Hugging Face, Accessed: Jan. 13, 2024. [Online]. Available: https://huggingface.co/microsoft/deberta-v3-base, 2024.
[42] sentence-transformers/gtr-t5-base·Hugging Face, Accessed: Jan. 13, 2024. [Online]. Available: https://huggingface.co/sentence-transformers/gtr-t5-base, 2024.
[43] Embeddings - OpenAI API, Accessed: Jan. 13, 2024. [Online]. Available: https://platform.openai.com/docs/guides/embeddings, 2024.
[44] sentence-transformers/sentence-t5-base·Hugging Face, Accessed: Jan. 13, 2024. [Online]. Available: https://huggingface.co/sentence-transformers/sentence-t5-base, 2024.
[45] WhereIsAI/UAE-Large-V1·Hugging Face, Accessed: Jan. 13, 2024. [Online]. Available: https://huggingface.co/WhereIsAI/UAE-Large-V1, 2024.
[46] xlm-mlm-en-2048·Hugging Face, Accessed: Jan. 13, 2024. [Online]. Available: https://huggingface.co/xlm-mlm-en-2048, 2024.
[47] T. Shen, F. Zhang, J. Cheng, A comprehensive overview of knowledge graph completion, Knowl.-Based Syst. 255 (2022) 109597, https://doi.org/10.1016/j.knosys.2022.109597.
[48] K. Xu, S. Jegelka, W. Hu, J. Leskovec, How powerful are graph neural networks, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, USA, 2019, pp. 1–17, https://doi.org/10.13140/RG.2.2.23687.70561.
[49] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, Adv. Neural Inf. Proces. Syst. 26 (2013), in: https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html.
[50] B. Yang, W. Yih, X. He, J. Gao, L. Deng, Embedding entities and relations for learning and inference in knowledge bases, arXiv preprint arXiv:1412.6575, 2014.