Requirements Management for Design of NPPs using Graph Neural Networks

Byoungchan Han [a*], Byeong-hyeok Ha [a], Byeongmun Ahn [a], Tongkyu Park [a], Sung-Kyun Zee [a], Jaeseok Yoo [b], Young-Jin Oh [b]
[a]*FNC Technology, Institute of Future Energy Technology, 46, Tapsil-ro, Giheung-gu, Yongin-si, Gyeonggi-do, 17084, Republic of Korea*
[b]*KEPCO E&C, 269, Hyeoksin-ro, Gimcheon-si, Gyeongsangbuk-do, 39960, Republic of Korea*
[*]*Corresponding author: bchan007@fnctech.com*

1. Introduction

The recent advances in deep learning are truly groundbreaking. In the last decade, CNN-based models optimized for analyzing grid-type data and RNN-based models for effectively learning sequential data have been studied. Moreover, the Transformer model, which was released in 2017 [1] demonstrated remarkable performance in natural language processing (NLP) at the time by incorporating the mechanism of attention that allows the model to learn which data to focus on, and is still attracting public attention as it has expanded into new machine learning fields outside of NLP such as computer vision and image detection.

On the other hand, most deep learning models studied thus far have the disadvantage of relying on a specific data type. However, as data in the real world becomes more complex and entangled, the need for data analysis with a network structure has emerged. Since these graph data were difficult to analyze using existing deep learning models due to their lack of fixed structure and sequential feature, graph neural network (GNN) models have been actively researched

Graph data can also be applied to nuclear power plant (NPP) design documents as each document is associated with design requirements specified therein. In addition, controlling these design documents becomes more difficult as massive amounts of documents are continuously produced and accumulated. Therefore, we present the applicability of GNNs to NPP design requirements to implement automatic management system in this paper.

2. Background and preliminaries

In this section, we will outline the background of graph neural networks and how their trained knowledge is applied to specific tasks.

2.1. Graph convolutional network (GCN)

In deep learning, the convolution operation, which integrates the products of local features and kernels, is frequently used to aggregate and filter local features. Convolutional neural network is one of the most well-known artificial neural networks that uses convolutional operations. It is widely used in image data analysis, as it can effectively learn images by dividing them into small local zones and aggregating local features. Likewise, the goal of GCN is to extract the most important features for a given task. It filters through the graph to collect features from important nodes and edges [2]. Both CNN and GCN collect information of adjacent data points from target pixels or nodes.

Training principle of GCN is to update the feature by collecting information on the nodes connected to each node. As the matrix multiplication between the feature matrix and the weight matrix extracts features of the next hidden layer, the k-th updated feature matrix (or hidden layer) in which each row represents the feature of each node can be expressed as follows:

$$\boldsymbol{H}^{k+1} = \sigma(\boldsymbol{A}\boldsymbol{H}^{k}\boldsymbol{W}^{k}) \tag{1}$$

However, there are two types of issues with this expression. First, nodes without self-loops do not reflect their own features when the feature matrix is updated because the elements of adjacency matrix corresponding to such nodes are zero. Second, nodes that have many neighbors have high values of feature representations, while nodes with few neighbors have low values. For higher-degree nodes, this phenomenon can result in an exploding gradient problem, while for lower-degree nodes, it can result in a vanishing gradient problem.

The first issue is solved by assigning self-loops to all nodes. In other words, GCN model substitutes $\widetilde{\boldsymbol{A}}$, which allocates 1 to all diagonal elements of the adjacent matrix $\boldsymbol{A}$, for $\boldsymbol{A}$ in expression (1). The second issue is addressed by normalizing the aggregated features to the degree of each node. Normalization can be achieved by multiplying the inverse degree matrix and the adjacency matrix of a graph. To summarize, feature matrix expression is improved as follows.

$$\boldsymbol{H}^{k+1} = \sigma(\boldsymbol{D}^{-1}\widetilde{\boldsymbol{A}}\boldsymbol{H}^{k}\boldsymbol{W}^{k}) \tag{2}$$

2.2. Graph attention network (GAT)

Unlike GCN's feature matrix calculation by graph convolution, GAT computes the feature matrix with a concept called self-attention, which indicates the importance and level of contribution of each neighboring node to the embedding of a given node. [3] The self-attention score of a neighbor node j to the given node i, $\alpha_{i,j}^{k}$ is defined as below. $\boldsymbol{a}_k$ denotes trainable weight and the operator $\oplus$ stands for vector

concatenation. Note that N_i stands for the first-order neighbors of node i which includes node i itself.

$$\alpha_{i,j}^{k} = \frac{\exp\left(\text{LeakyReLU}\left(\boldsymbol{a}_k^T\left[\boldsymbol{W}^k\boldsymbol{h}_i^k \oplus \boldsymbol{W}^k\boldsymbol{h}_j^k\right]\right)\right)}{\sum_{r\in N_i}\exp\left(\text{LeakyReLU}\left(\boldsymbol{a}_k^T\left[\boldsymbol{W}^k\boldsymbol{h}_i^k \oplus \boldsymbol{W}^k\boldsymbol{h}_r^k\right]\right)\right)} \quad (3)$$

The attention score calculated in expression (3) determines the importance of each neighbor node, and updates the input data as expression (4).

$$\boldsymbol{h}_i^{k+1} = \sigma\left(\sum_{j\in N_i}\alpha_{i,j}^{k}\boldsymbol{W}^k\boldsymbol{h}_j^k\right) \quad (4)$$

The most significant advantage of GAT is that it does not necessitate an entire graph structure. In the case of GCN, the node representation required adjacent matrix A from the overall graph. As a result, GCN was vulnerable not only to the computational cost but also to the changes in the graph. On the other hand, GAT is capable to obtain the node feature only with the corresponding neighbors' representations. Such characteristic allows GAT to be applied for not only transductive but also inductive problems by hiding nodes and edges used in validation or test sets.

2.3. GraphSAGE

Although the previously introduced GCN is a representative model of graph neural networks, it has the fatal disadvantage of using the adjacent matrix and the degree matrix as input. It implies that GCN is taking information from the entire graph. Thus, GCN can only make predictions on trained nodes and edges. However, most graph data in real life are constantly changing (e.g., social network, viewing history in streaming service, e-commerce activities). To overcome this limitation, GraphSAGE, named after graph sampling and aggregation, was proposed.

The key concept of GraphSAGE is the use of a sampling technique and a new weighting method instead of the traditional degree and norm methods commonly used in aggregation. [4] GraphSAGE randomly samples a set of neighbors of a fixed size for aggregation, whereas the GCN model aggregates the entire first-order neighbors to evaluate the targeted node's embedding. As a result, GraphSAGE can use neighbor's features in the absence of graph structural information.

3. Methods and results

In this paper, we propose to apply graph neural network models to manage NPP design requirements. To test the applicability of GNN, we conducted an experiment to predict which design documents contain arbitrary design requirements. There are 1,066 design documents and 444,553 sentences to be used as input, which were collected from the U.S. NRC website [5]. Each line contained in the document was separated by a newline letter, and was regarded as a design requirement. It was not used as input data if the number of words in a line was less than ten or the proportion of alphabet letters was less than 70%.

Graph data was constructed using three different types of nodes, and two different types of edges. Nodes were composed of documents, lines, and words. Edges were set between document nodes and the line nodes that comprise corresponding documents, as well as between line nodes and the word nodes that comprise respective lines. Graph was constructed as an undirected graph.

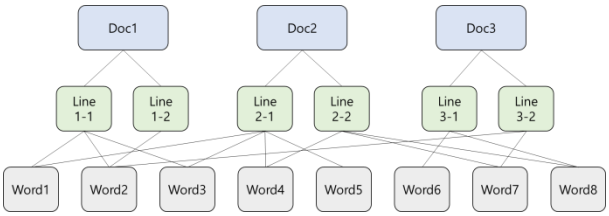

Fig. 1. Graph model of the design requirements data.

All words are converted to their reflected forms with the lemmatization approach so that words with the same root can be considered as a single item. Then Sentence-BERT [6] was used to evaluate the initial embeddings of word nodes and line nodes. Document embeddings were initialized by mean-pooling the line embeddings that comprise the respective documents.

Table I: Model Parameters

GCN	number of layers	2
	hidden layer size	48
	output layer size	6
	dropout rate	0.5
GAT	number of layers	2
	number of heads	8
	hidden layer size	8
	output layer size	6
	dropout rate	0.6
GraphSAGE	number of layers	2
	hidden layer size	48
	output layer size	6
	dropout rate	0.5

To perform inductive learning, we first masked the line nodes and linked edges in the test set during the training phase. GCN and GraphSAGE were used to compute the remaining node embeddings after masking. The prediction task was then performed to estimate the existence probability of missing links between the masked line nodes and the document nodes. The link prediction results are used in binary cross entropy loss and are reflected in backpropagation process. Each model's hyperparameters are presented in Table I.

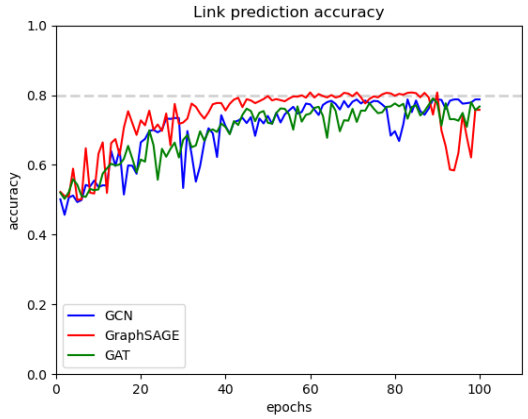

Fig. 2. Accuracy of link prediction task for 100 documents.

First, we measured the accuracy of the model for 100 design documents and 63,134 sentences contained therein. The maximum accuracies obtained were 0.7878 for GCN, 0.8077 for GraphSAGE, and 0.7908 for GAT. Results are shown in Fig. 2. Experiments on all documents were conducted only on GCN and GraphSAGE, while GAT was not conducted in the current study due to a lack of computing resources. Fig. 3 presents the performances of GCN and GraphSAGE for all input data. Training and evaluation were carried out for 100 epochs.

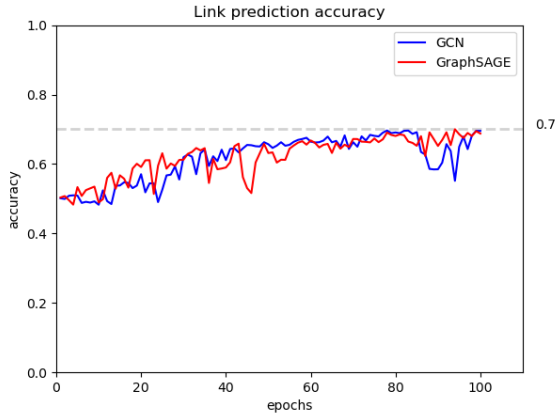

Fig. 3. Accuracy of link prediction task for all input data.

We chose the same number of connected and unconnected node pairs at random for performance evaluation. Models' Accuracies were measured by predicting whether or not given node pairs are connected.

4. Conclusion

This work demonstrated the applicability of GNN models to the NPP design requirements. Although the GNN models only achieved 80% accuracy for 100 documents and 70% accuracy for 1,066 documents, we believe there is a room for improvement because the input text used in this experiment contains a number of general statements rather than well-organized design requirements.

In the future, we will focus on developing a GNN model optimized for bi-partite graphs by testing models that require huge computing resources such as GAT, and implementing a design requirements management system usable in the field by training model with actual design data.

NOMENCLATURE

$\boldsymbol{A}$	Adjacency matrix.
$\boldsymbol{a}_k$	Trainable weight matrix regarding to self-attention of k-th hidden layer.
$\alpha_{i,j}^k$	Self-attention score of the node j to the node i in k-th hidden layer.
$\boldsymbol{D}$	Degree matrix.
$\boldsymbol{H}^k$	k-th hidden layer.
$\boldsymbol{h}_i^k$	Embedding of node i contained in k-th hidden layer.
LeakyReLU	Non-linear function to perform a threshold operation, where any input value less than zero is multiplied by a fixed scalar.
σ	Non-linear function (i.e., ReLU).
$\boldsymbol{W}^k$	Trainable weight matrix of of k-th hidden layer.

ACKNOWLEDGMENTS

This research was supported by KEPCO Engineering & Construction Company, Inc. (KEPCO E&C) (No.TRS29).

REFERENCES

[1] A, Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, Attention is all you need, Advances in neural information processing systems, Vol. 30, pp. 5998-6008, 2017.
[2] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907, 2016.
[3] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903, 2017.
[4] W. Hamilton, Z. Ying, and J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems, Vol. 30, 2017.
[5] U.S.NRC, https://www.nrc.gov.
[6] N. Reimers, and I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084, 2019.