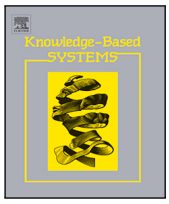
Knowledge-Based Systems 324 (2025) 113883

Contents lists available at ScienceDirect

Knowledge-Based Systems

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



A Structure-Feature Dynamic Decoupling GNN architecture for link prediction

Guowang Li, Zhiqiang Pan, Fei Cai *, Weijie Chen, Langgao Cheng

National Key Laboratory of Information Systems Engineering, National University of Defense Technology, Changsha, 410073, Hunan, China

ARTICLE INFO

Keywords:
Structure-Feature Dynamic Decoupling
Graph Neural Network
Graph contrastive learning
Link prediction

ABSTRACT

Link prediction aims to forecast the missing links within a graph, which is widely applied in various fields such as recommender systems and drug analysis. Graph Neural Networks (GNNs) have emerged as strong baselines for link prediction due to their ability to simultaneously capture the topological structure and node features of graphs. Moreover, existing approaches use the node features as the initial embedding of nodes and input them into GNNs for message passing and updating. However, these methods assume that the features and topology in graphs are homophonic and do not take into account the possible incompatibility that is common and even very severe in some graphs, harming the performance of GNNs in link prediction.

To address this issue, we propose a Structure-Feature Dynamic Decoupling GNN architecture (SFDDGNN), which mainly consists of two decoupled embedding pipelines and the dynamic gate fusion mechanism. Specifically, to avoid the incompatibility, we first utilize a GraphSAGE-based structure encoder to capture the topological structure in one pipeline. Then we construct a graph contrastive learning module to train the node feature embedding in the other pipeline. Finally, we dynamically aggregate the topology and features embedding based on the graph data distribution knowledge. Experimental results on three real-world datasets of link prediction demonstrate that SFDDGNN outperforms the state-of-the-art baselines by up to 3.54% and 6.55% in terms of AP and AUC, respectively.

1. Introduction

The graph is a data structure that regards objects as nodes and describes relations between objects through links [1]. As an important graph task, link prediction aims to predict the missing links in the current graph, which has broad applications in recommender systems [2,3], community detection [4], and knowledge graphs [5]. Graph neural networks (GNNs) have become the leading baselines for graph data mining due to their strong ability to aggregate information from both node features and topological structure, outperforming the heuristic methods [6–10] and the embedding-based methods [11–13] in various graph tasks [14]. Recently, Graph Neural Network for Link Prediction (GNN4LP) has been widely investigated to improve the graph link prediction performance [15–17].

Heuristic methods quantify the association strength between nodes based on predefined topological rules. While computationally efficient, these methods overly rely on manually designed prior assumptions, which limits their flexibility and generalizability. Embedding-based methods implicitly encode graph structure into low-dimensional vectors to support end-to-end learning. However, most existing models primarily focus on one-hop or shallow neighborhood relationships, thereby struggling to capture high-order interaction patterns. Additionally, their ability to effectively integrate feature information with structure information remains constrained. GNN-based methods utilize message-passing mechanisms to aggregate multi-hop neighborhood information, enabling the automatic extraction of complex topological patterns. These methods also facilitate end-to-end joint optimization by incorporating node/edge features. Nevertheless, existing approaches still face challenges in coordinating node features with topological structure.

Existing methods use the node features as initial inputs to GNNs, which assume that graphs are homophonic regarding node features and topological structure, i.e., nodes with similar features and topology tend to establish the links [18]. However, this assumption ignores the possible incompatibility between the node features and topological structure. For example, when two nodes have similar features but different topologies, the repugnant supervisory signals will confuse the GNNs. Consider two feature-similar user nodes A and B in a social network. Node A resides in a dense community with a high degree

* Corresponding author.
E-mail addresses: liguowang@nudt.edu.cn (G. Li), panzhiqiang@nudt.edu.cn (Z. Pan), caifei08@nudt.edu.cn (F. Cai), chenweijie19@nudt.edu.cn (W. Chen), chenglanggao23@nudt.edu.cn (L. Cheng).

https://doi.org/10.1016/j.knosys.2025.113883
Received 5 December 2024; Received in revised form 27 May 2025; Accepted 29 May 2025
Available online 13 June 2025

of centrality, while node B is located at the edge of the network with low centrality. Although feature similarity might suggest a potential link between them, real-world data reveal that their connectivity is predominantly governed by topological constraints: users in dense communities tend to interact internally, whereas sparse users rarely establish new connections due to their structural sparsity. This structural dominance often prevents link formation despite feature affinity. Furthermore, Hou et al. [19] note that neglecting topological heterogeneity may lead to misjudgment of node influence in predictive models. Wang et al. [20] argue that direct integration of node features with topological structure risks covers critical information about structural specificity, compromising model interpretability in network analysis.

Although some works [21,22] are aware of the incompatibility, they only use GNNs to embed the node features and topological structure individually and linearly combine them. For instance, Decouple-SEAL [21] employs two separate SEAL modules to independently embed node features and topological structure for link prediction. The final prediction probability is obtained by linearly combining the outputs from both SEAL modules. Similarly, GNN-BC [22] separately embeds node features and topological structure, concatenating the final-layer embedding to derive the overall node representations for link prediction. Due to the incomplete decoupling and simple fusion approach, these methods are generally effective and only for specific graphs. This leads to negative effects on the model performance for link prediction.

To address the above issues, we propose the Structure-Feature Dynamic Decoupling GNN architecture (SFDDGNN). Specifically, we first initialize the topological structure utilizing the Ensemble Learning Structure Encoder (ELSE) and input it into the GraphSAGE-based encoder for message passing and updating to generate the structure-only embedding. Then, we fine-tune the language model by graph contrastive learning to train the feature encoder and generate the feature-only embedding. Finally, we jointly input the structure-only embedding and feature-only embedding into the dynamic gate fusion mechanism, which fuses the node features and topological structure based on the graph knowledge, to accomplish the link prediction task. Extensive experiments on three datasets show that SFDDGNN outperforms the baselines by up to 3.54% and 6.55% in terms of AP and AUC, respectively.

Our contributions are summarized as follows:

1. We propose a graph representation learning architecture that completely separates the node features from topological structure. Compared to existing works, our approach effectively addresses the incompatibility between structural proximity and feature proximity of graph.
2. We design a dynamic gate function based on the graph large model and multi-head attention mechanisms for integrating the structure and feature embedding. Inspired by the graph large model, SFDDGNN can selectively retain useful information from the global level to accomplish the link prediction task.
3. Experiments conducted on three real-world datasets demonstrate that SFDDGNN outperforms the baselines in link prediction. In particular, in the Arxiv2023 dataset, AP and AUC improve by 3.54% and 6.55%, respectively.

2. Related work

We review the related works from three angles: link prediction, graph contrastive learning and graph data distribution analysis. The three subsections correspond to the main technology stacks of the three pipelines of the SFDDGNN.

2.1. Link prediction

Link prediction is a graph task to discover missing links between nodes, where existing technologies mainly include the heuristic methods, the embedding-based methods, and the GNN-based methods [23].

2.1.1. Heuristic methods

Heuristic methods refer to simple rules or algorithms that do not depend on complex models or global optimization [24]. In detail, they use the intuitive characteristics of graphs, such as node degree, clustering coefficient, and path length, to design feature engineering for predicting potential links between nodes. Several classes of heuristics existing for link prediction include local structural proximity, global structural proximity, and feature proximity [21]. Firstly, Local Structure Proximity(LSP) extracts local neighborhood information between nodes, usually involving neighbor nodes with only fewer hops. For instance, Common Neighbor (CN) [6] predicts the possibility of links between nodes by the number of common neighbors. And Adamic-Adar (AA) [7] considers the sparsity of common neighbors base on CN, i.e., nodes with fewer neighbors contribute more to the prediction. Secondly, Global Structural Proximity (GSP) starts from the node pairs to be predicted and extends to the full graph structure information. The Shortest Path (SP) [25] calculates the shortest path between the node pairs, then the shorter the path, the greater the possibility of forming a link between nodes. Katz [9] evaluates the possibility of establishing links by calculating the weighted sum of all paths with different lengths between node pairs. PageRank [10] predicts the probability of new links based on node centrality. Finally, Feature Proximity (FP) evaluates the feature similarity between nodes. It is considered that nodes with similar characteristics are more likely to generate edges, where the commonly adopted method is cosine similarity.

However, the rules are set in advance and constant, so heuristic methods cannot adapt to the characteristics of different graphs and lack flexibility.

2.1.2. Embedding-based methods

The embedding-based methods map nodes into a low-dimensional vector space to capture the complex relationship between them [23]. The core idea is that if two nodes have similar neighborhood structures or roles in a graph, the distance between their embedding in the low-dimensional space is also close. For example, DeepWalk [11] generates a node sequence through random walk and then uses the Skip-gram [26] to learn the embedding of nodes. Moreover, Node2Vec [12] performs a biased random walk to explore the neighborhood of nodes and then optimizes the embedding by maximizing the co-occurrence probability of nodes in the walk sequence. Furthermore, Matrix Factorization (MF) [13] captures the potential relationship between nodes by decomposing the adjacency matrix into the product of two or more low-dimensional matrices.

However, the embedding-based methods only capture topological structure and are affected by the embedding dimension and noise, resulting in less accurate representation of complex relationships between nodes.

2.1.3. GNN-based methods

Graph Neural Network (GNN) can capture the topological structure and node features of graphs at the same time, which aggregates and updates the message of nodes through iteration, thus obtaining the node embedding [23]. In link prediction, the classical way is to use GNNs for learning the embedding of nodes and design a ReadOut function to evaluate the similarity. The similarity output by the ReadOut function is used as the probability of existence of a link. In general, link prediction with GNNs is solved as a binary classification problem [23]. However, recent works have shown that the vanilla GNNs cannot effectively model the link prediction task [17,27], thus recent works propose a series of graph neural networks for link prediction, namely GNN4LP. Specifically, compared with vanilla GNNs, GNN4LP focuses more on capturing the topological structure and node features of paired nodes and is equipped with a paired decoder as the ReadOut function. For example, SEAL [17] establishes a separate local subgraph for nodes to be predicted and runs GNN on it to learn the embedding. Moreover, NCNC [15] and NBFNet [16] use neural function to learn CN and Katz,

respectively, then the learned heuristic information is supplemented into the node pair embedding to improve the ability of GNN for making link predictions.

However, GNN4LP relies on heuristic information and data preprocessing in link prediction, resulting in inefficient training and high memory burden, as well as lack of enhancement specialized for different graphs. In addition, the incompatibility of node features and topological structure limits GNN4LP, but existing works have not paid sufficient attention to this issue.

2.2. Graph contrastive learning

Contrastive Learning is a self-supervised learning method that learns the representation by maximizing the similarity of positive sample pairs and minimizing the similarity of negative sample pairs. Related researches have made significant progress in Natural Language Processing (NLP) [28–30] and Computer Vision (CV) [31–33]. Inspired by this, recent works have proposed a similar framework to achieve self-supervised training on graphs [34]. ECLiP [35] effectively enhances link prediction by using edges as comparison data and integrating edge information into node representations. LGCL [36] combines SEAL and LGLP by contrastive learning to reduce information loss. When the node feature of the graph data is text, the contrastive learning framework in NLP can be migrated to the graph. For example, the link prediction task is very similar to the text similarity task. Therefore, in the Link Prediction task, we can use the end nodes with links as positive samples, and the node pairs with distant or unreachable paths as negative samples, and then use contrastive learning to embed the text features on them.

In summary, the reason such methods are effective is that comparing paired graph data can effectively capture both the topological structure and node features [37]. Therefore, in the feature-based link prediction pipeline, we fine-tune the language model using graph contrastive learning by combining the link prediction task with the text similarity matching task to model node triples. By this approach, we fully capture the node feature information based on the semantic perspective.

2.3. Graph data distribution analysis

Graph Data Distribution Analysis (GDDA) refers to representing and analyzing data with the graph theory, topology, and other related theories [38]. The purpose of GDDA is to obtain graph patterns and extract data information to complete various graph tasks [39]. Depending on the characteristics of different graphs or the analysis objectives focused in the GDDA, the main methods include Random and Evolutionary Network Models [40,41], Structural and Community Detection Models [42], Weighted, Multidimensional and Flow Models [43], and Network Embedding and Graph Isomorphism Models [44]. Since different tasks focus on different distribution knowledge, the corresponding graph data distribution analysis model should be selected for various tasks. We use the Network Embedding and Graph Isomorphism Models to analyze the distribution of graph data, which maps the whole graph to a low-dimensional space to generate embedding containing information such as graph domain knowledge and graph distribution characteristics. Graph Isomorphism Network (GIN) [45] and GraphGPT [46] are two typical Network Embedding and Graph Isomorphism Models. GIN can obtain graph embedding that reflect graph's global structure and local patterns, thus helping to identify distribution patterns and anomalies in the graph. GraphGPT demonstrates robust generalization in graph data distribution shift detection through its dual-stage graph instruction tuning and Chain-of-Thought distillation techniques

Recent works have found that link prediction mainly focuses on the topological structure and node features of graphs, but there exists incompatibility between them, which is also one of the reasons that mainstream traditional technologies have limited performance in edge-level tasks. For example, the heuristic method and the embedding-based method only consider the topological structure, while the GNNs originates from the message passing research on graphs, thus confusing the structure and feature information. In summary, existing technologies either focus on incomplete graph features or ignore the incompatibility between topological structure and node features. Based on this analysis, we analyze the graph data through the Network Embedding and Graph Isomorphism Models, in order to guide the fusion of topological structure and node features.

3. Approach

In this section, we first state the notations used in this paper and their explanations. Then we detail the SFDDGNN models, which consists of three main components, i.e., GraphSAGE-based structure encoder (see Section 3.1), graph contrastive learning for node feature embedding (see Section 3.2), and dynamic gate fusion mechanism based on graph data distribution analysis (see Section 3.3).

The workflow of SFDDGNN is plotted in Fig. 1. First, the graph is divided into topological structure graphs and a set of node features. In the structure-based link prediction pipeline, the topological structure graphs are input into the Ensemble Learning Structure Encoder for initialization to generate the initial node embedding, then, the initial node embedding is input into GraphSAGE-based encoder for message passing and updating to generate the structure-only embedding. In the feature-based link prediction pipeline, node features are input into the language model (RoBERTa) for embedding, and then we fine-tune the node feature embedding by graph contrastive learning to generate feature-only embedding. After that, the structure-only embedding and feature-only embedding are fused by the dynamic gate fusion mechanism. Note that the dynamic gate fusion mechanism includes graph background knowledge from graph data distribution analysis, so the fusion is specific and can mitigate incompatibilities

We formulate the notations and their explanations as follows. Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ denote an undirected graph, where $\mathcal{V}$ and $\mathcal{E}$ are the set of $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, respectively. Moreover, the node set $\mathcal{V}$ is associated with the feature matrix $X \in \mathbb{R}^{n\times d}$, where d is the feature dimension and $n = |\mathcal{V}|$. And the graph can be denoted as an adjacency matrix $A \in \mathbb{R}^{n\times n}$, where $A_{i,j} = 1$ if $(v_i, v_j) \in \mathcal{E}$ and $A_{i,j} = 0$ otherwise, and $\mathcal{N}_i^k$ denotes the set of the k-hop neighbors of node v_i. In addition, the main notations used in this paper and their explanations are shown in Table 1.

3.1. Structure-based graph representation learning

Section 3.1 is the structure-based link prediction pipeline. Compared to tasks such as node classification (node-level task) and graph matching (graph-level task), the edge-level task relies more on the neighborhood information of pairs of nodes. The edge-based topology is the primary information source for accomplishing the link prediction task [47]. To pay full attention to the topological structure, we construct the Ensemble Learning Structure Encoder (ELSE) for initialization and the GraphSAGE-based structure encoder for message passing and updating.

3.1.1. Ensemble learning structure encoder

When node features are missing, vanilla GNNs generally use one-hot encoding as the initial embedding of nodes. However, one-hot encoding only identifies different nodes and cannot represent the topology. To address this challenge, we construct the Ensemble Learning Structure Encoder (ELSE) for generating initial embedding of nodes to solve the problem of insufficient information of one-hot encoding.

For the node pair (v_i, v_j) to be predicted, we use DeepWalk, Node2Vec and MF for link prediction to train the basis methods of

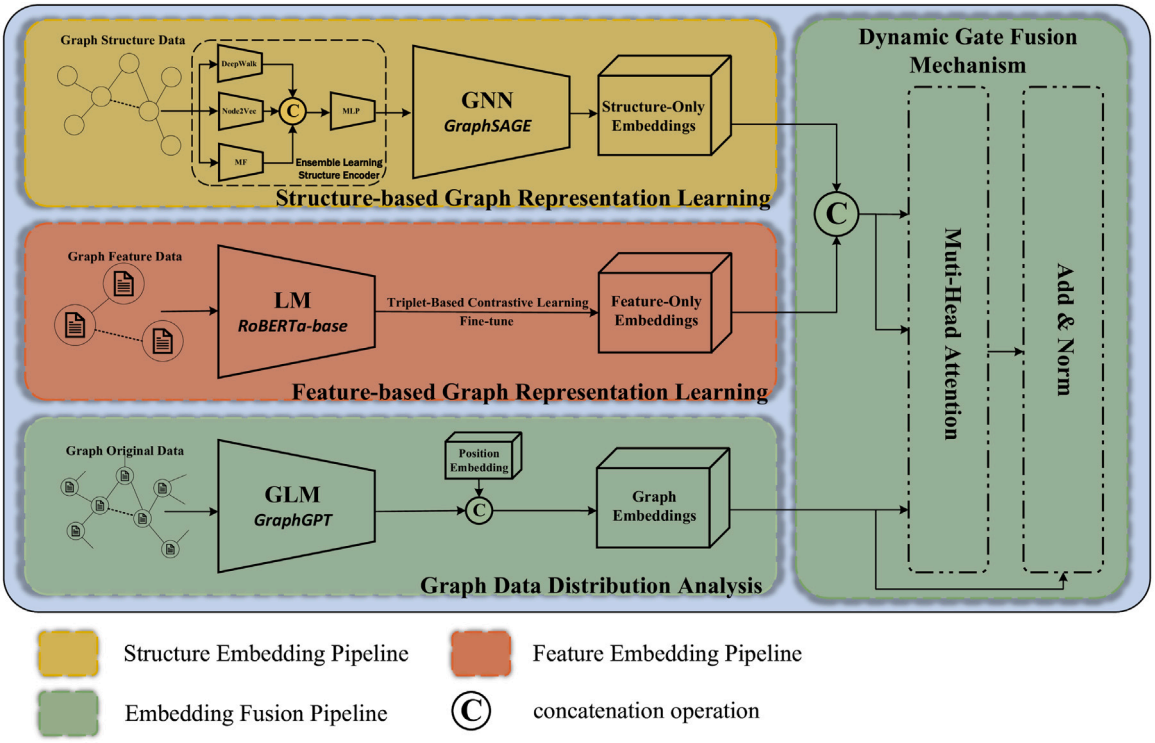


Fig. 1. Structure-Feature Dynamic Decoupling GNN Architecture. Graph Structure Data refers to neighborhood topological structure data organized by node pairs. Graph Feature Data refers to textual data on node triples. Graph Original Data refers to the complete graph data.

Table 1
Notations and Explanations.

Notations	Explanations
$\mathcal{G}$	A graph
$\mathcal{V}$	The set of nodes in graph $\mathcal{G}$
$\mathcal{E}$	The set of edges in graph $\mathcal{G}$
A	The adjacency matrix.
$\mathcal{N}_i^k$	The set of k-hop neighbors of node v_i
$\mathcal{B}$	The set of basis methods of the ELSE.
X^O	The node identity matrix composed of one-hot vectors.
$X^{\mathcal{B}}$, X^S, X^F, $X^{\mathcal{G}}$	The feature matrices in different pipelines
$x_i^{\mathcal{B}}$, x_i^S, x_i^F, $x_i^{\mathcal{G}}$	The feature vector of node v_i in different pipelines
$W_{\mathcal{B}}$, W_E, W_N	The trainable parameters of the ELSE, GNN, and ReadOut Function
H^k	The feature matrix of the kth layer
h_i^k	The feature vector of the kth layer of node v_i
h_{ij}^k	The embedding of the kth layer of edge (v_i, v_j)
H_S, H_F	The structure/feature-only Embedding matrix
$\mathcal{C}$	The training set for contrastive learning
$\mathcal{T}$	The set of text messages on nodes
Ω	The set of positive samples
ω	The set of negative samples
Γ	The set of positive samples and negative samples
p_{ij}	The probability of existence of edge (v_i, v_j)

ELSE. The training process for each basis method is as follows:

$$\begin{cases} \max\limits_{W_{\mathcal{B}}} ReadOut\left(\mathcal{B}\left(A, X^O\right)\right), \\ ReadOut\left(h_i, h_j\right) = \dfrac{h_i \cdot h_j}{\left\|h_i\right\| \left\|h_j\right\|}, \end{cases} \tag{1}$$

where $\mathcal{B} = \{DeepWalk, Node2Vec, MF\}$, $W_{\mathcal{B}}$ denotes trainable parameters in basis models. The $Readout(\cdot)$ denotes the link prediction mechanism that takes the embeddings of the target node pair as input and outputs the probability of link existence between them. Commonly adopted implementations include cosine similarity functions and multi-layer perceptrons (MLPs). In this experiment, we employ the cosine similarity function as the $Readout(\cdot)$. After that, we embed the nodes using the trained basis methods and input them into the gating function for aggregation. The specific process can be formulated as follows:

$$X^{\mathcal{B}} = MLP\left(\Big\|_{\mathcal{B}} \mathcal{B}(A, X)\right), \tag{2}$$

where $X^{\mathcal{B}}$ is the initial node embedding of the GNN model, $\|$ denotes the concatenate operation. $MLP(\cdot)$ is a fully connected layer consisting

of two linear layers for fusing node embeddings from three basis models and integrating feature dimension to match subsequent models.

In summary, we construct the Ensemble Learning Structure Encoder (ELSE), which receives one-hot encoding and outputs the initial topological structure embedding that serves as the input to the subsequent GNN.

3.1.2. Graphsage-based structure encoder

To fully capture the neighborhood topology information of the node pair to be predicted, we use the subgraph composed of node pairs and their k-hop neighborhoods as the training data to train the GraphSAGE-based structure encoder. Compare to the classical GraphSAGE, our model focuses on the link prediction. There are two main improvements.

On the one hand, we update the edge embedding after each messages passing and retain information of shallow networks through residual connections, which helps to encode multi-hop topological structure information and prevents over-smoothing in the edge embedding process. Specifically, during model training, the node embedding is updated as follows:

$$\begin{cases} h_{ij}^0 = \sigma\left(x_i^S \;||\; x_j^S\right), \\ h_u^0 = x_u^S \;, \forall u \in \mathcal{N}_i^h \cup \mathcal{N}_j^h, \\ h_i^k \leftarrow \sigma\left(W_N \cdot Aggregator\left(h_i^{k-1} \cup h_u^{k-1}\right)\right), \\ \forall u \in \mathcal{N}_i^h, \end{cases} \tag{3}$$

where x_i^S denotes the initial topological structure embedding of node v_i, W_N denotes the trainable parameters of GNN. And $Aggregator(\cdot)$ denotes the message aggregation function, which adopts the average pooling function in our model.

Then, the edge embedding is updated by the residual connections:

$$h_{ij}^k \leftarrow \sigma\left(W_E \cdot Aggregator\left(h_{ij}^{k-1}, h_i^k, h_j^k\right)\right), \tag{4}$$

where W_E denotes the trainable parameters of ReadOut Function.

On the other hand, we design an edge-based loss functions to jointly train GNN and ReadOut Function:

$$\mathcal{L}_S = -y log\left(\delta\left(h_{ij}\right)\right) + (1-y)\, log\left(1-\delta\left(h_{ij}\right)\right), \tag{5}$$

where h_{ij} denotes the edge embedding from the final layer of GraphSAGE-based Structure Encoder, $\delta(\cdot)$ is a Sigmoid-based ReadOut Function to predict edge existence probability from edge embedding, and $y \in \{0,1\}$ is the edge label. It is easy to see that optimizing $\mathcal{L}_S$ can sequentially optimize W_N and W_E to achieve the joint training.

Combining the models in Sections 3.1.1 and 3.1.2, we obtain an encoder with only topology information, and then obtain the structure-only embedding as follows:

$$\begin{cases} X^S = ELSE\,(X), \\ H_S = StruE\left(X^S\right), \end{cases} \tag{6}$$

where $StruE_{output}(\cdot)$ is an encoder composed of GNN and ReadOut Function, and H_S only contains the topology information based on the edges and neighborhoods of the nodes to be predicted. The structure-based link prediction pipeline avoids the information loss caused by the rigid fusion of node features and is more beneficial to the performance of link prediction.

3.2. Feature-based graph representation learning

Section 3.2 is a feature-based link prediction pipeline. Although the topological structure plays a major role in the link prediction, the node features have a non-negligible supplementary role in understanding the graph based on an information integrity perspective. Node features can provide semantic information that cannot be modeled by topology, and then capture the potential connections between nodes.

Existing works treat node features as initial embedding and input them into GNNs, relying on the topological structure for message passing and updating. It leads to that the embedding of semantic information over-trust and over-reliance on graph. When the graph is not trustworthy, the obtained embedding will have uncontrollable information bias. To address this challenge, we consider node feature embedding as the text similarity matching task and construct an encoder based on contrastive learning. Our method can focus on semantic relevance and reduce the coupling between features and structures.

We first construct the dataset for contrastive learning. For any three nodes v_i, v_j, v_k on the graph, if v_i is directly connected to v_j, v_i and v_k are not reachable or more than 3 hops away from each other, they can be used as a set of positive and negative sample pairs, which is denoted by $c = \{v_i, v_j, v_k\}$. Specifically, positive samples are directly derived from existing edges in the original graph. For negative sampling, we implement a hybrid strategy that combines degree centrality ranking and topology-aware exploration to ensure both structural diversity and computational efficiency:

1. Degree Centrality Prioritization: Nodes are ranked by degree centrality (measuring connection density) to form a candidate pool of central nodes, prioritizing high-degree hubs as these often serve as critical bridges in graph structure.
2. Alternating Sampling: For top-ranked nodes, we alternately apply: a. Random Sampling to ensure diversity in negative pairs; b. Breadth-First Search (BFS)-Guided Sampling to capture multi-hop topological relationships (3+ hop reachable nodes) and n-hop unreachable nodes, thereby balancing local/global structural awareness.
3. Termination Criteria: a. Each central node generates no more than 5 negative pairs to prevent over-representation; b. Nodes lacking 3+ hop reachable neighbors are excluded from subsequent iterations to avoid trivial samples.

Then, we can construct the positive and negative sample pairs with the number of $|\mathcal{E}|$ as the training dataset $C = \{c_1, c_2, \ldots, c_{|\mathcal{E}|}\}$. After that, for positive and negative sample pairs $c = \{v_i, v_j, v_k\}$, we use a pre-trained language model to generate the initial node feature embedding:

$$\begin{cases} x_i^F = LM\left(t_i\right), \\ x_j^F = LM\left(t_j\right), \\ x_k^F = LM\left(t_k\right), \end{cases} \tag{7}$$

where $t_i, t_j, t_k \in \mathcal{T}$, t_i, t_j, t_k are the texts (node features) on the nodes. Note that the node features of the graph data have a variety of presentation forms, such as text, pictures, etc. We only consider the text information, that is, the node features of the graph are long text. Then, we design the loss function for contrastive learning based on the triples with positive and negative sample pairs to fine-tune the language model. The loss function can be formulated as follows:

$$\begin{aligned} \mathcal{L}_F = & -\frac{1}{|\Omega|}\sum_{|\Omega|}\left(y \log \frac{\exp\left(d\left(x_i^F, x_j^F\right)\right)}{\exp\left(d\left(x_i^F, x_j^F\right)\right) + \exp\left(d\left(x_i^F, x_k^F\right)\right)}\right) \\ & + \frac{1}{|\omega|}\sum_{|\omega|}\left((1-y)\log \frac{\exp\left(d\left(x_k^F, x_j^F\right)\right)}{\exp\left(d\left(x_i^F, x_j^F\right)\right) + \exp\left(d\left(x_k^F, x_j^F\right)\right)}\right) \\ & + \lambda l\,(\theta), \end{aligned} \tag{8}$$

where $d(\cdot)$ denotes the similarity function, we adopt the cosine similarity metric:

$$d\left(x_i, x_j\right) = \frac{x_i \cdot x_j}{\left\|x_i\right\|\left\|x_j\right\|}. \tag{9}$$

And $\lambda l(\theta)$ denotes the regularization term, we use L2 regularization of the form:

$$\lambda l\,(\theta) = \lambda \sum \|\theta\|^2, \tag{10}$$

where θ refers to the parameters of the language model and λ controls the strength of the regularization. Finally, we fine-tune language model to generate the feature-only embedding:

$$H_F = FeatE(\mathcal{T}), \tag{11}$$

where $FeatE(\cdot)$ denotes the fine-tuned language model. Although the information contained in H_F is mainly derived from single-node features, due to the introduction of contrastive learning, the embedding also learn the relationship between nodes from a semantic perspective.

3.3. Dynamic gate fusion mechanism

The incompatibility between topological structure and node features leads to confusing supervised signals in the GNN optimization process. However, if only using one side of the information, it will lead to the problem of incomplete information.

To solve the above contradiction, we propose the dynamic gate fusion mechanism based on graph data distribution analysis. We use graph large model to perform graph data distribution analysis to capture the background knowledge and distributional differences of different graphs, because the graph large model has a huge knowledge reserve due to the pre-training of massive data. After that, the gate mechanism specifically fuses the embedding in Sections 3.1 and 3.2 based on the graph knowledge to selectively retain the contributed knowledge.

We first use the latest graph large model technology $GraphGPT(\cdot)$ [46] to embed the whole graph and extract the specific informations. Specifically, for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ which has $|\mathcal{G}|$ connected components, we use the graph large model to embed its connected components, which can be formulated as follows:

$$\begin{cases} \mathcal{G} = \{g_1, g_2, \ldots, g_{|\mathcal{G}|}\}, \\ X^{\mathcal{G}} = GLM(\mathcal{G}), \end{cases} \tag{12}$$

where $X^{\mathcal{G}}$ is derived from the full graph embedding, and thus contains graph knowledge in a global perspective. $X^{\mathcal{G}}$ contains both general knowledge complemented by graph large model and specific knowledge in different contexts. We use $X^{\mathcal{G}}$ as a guide for the fusion of structure-only embedding with feature-only embedding. Then, we input $X^{\mathcal{G}}$, H_S, and H_F into the dynamic gating function based on the multi-head attention mechanism, which can be formulated as follows:

$$\begin{aligned} H_m^G &= gate\left(X^{\mathcal{G}}, H_S, H_F\right) \\ &= softmax\left(\frac{\sigma\left(\left(X^{\mathcal{G}} \,||\, X\right) \cdot W_m^Q\right) \cdot \sigma\left(\left(H_S \,||\, H_F\right) \cdot W_m^K\right)^T}{\sqrt{d_k}}\right) \\ &\cdot \sigma\left(W_m^V \cdot \left(H_S \,||\, H_F\right)\right), \end{aligned} \tag{13}$$

where $H_S \in \mathbb{R}^{n \times d_S}$, $H_F \in \mathbb{R}^{n \times d_F}$, $X^G \in \mathbb{R}^{n \times d_G}$, and $X \in \mathbb{R}^{n \times d_X}$. W_m^Q, W_m^K, W_m^V denote the trainable parameters, and $W_m^Q \in \mathbb{R}^{|d_G + d_X| \times d_w}$, $W_m^K \in \mathbb{R}^{|d_S + d_F| \times d_w}$, $W_m^K \in \mathbb{R}^{n \times n}$. d_k denotes the dimension of the matrix $\sigma\left(\left(X^{\mathcal{G}} \,||\, X\right) \cdot W_m^Q\right) \cdot \sigma\left(\left(H_S \,||\, H_F\right) \cdot W_m^K\right)$.

Finally, we optimize the gating function in the link prediction task with the following loss function:

$$\begin{cases} p_{ij} = \delta\left(g\left(h_i^G \,||\, h_j^G\right)\right), \\ \mathcal{L}_{gate} = -\dfrac{1}{|\Gamma|} \displaystyle\sum_{(v_i, v_j, y) \in \Gamma} y \log p_{ij} + (1 - y) \log\left(1 - p_{ij}\right), \end{cases} \tag{14}$$

where h_i^G, h_j^G denote the embedding generated by fusion mechanism. When the node pair (v_i, v_j) is a positive sample, $y = 1$, otherwise $y = 0$.

In summary, based on the background knowledge and distribution characteristics embedded in the graph embedding, SFDDGNN selectively and dynamically regulates the fusion of topological structure and node features through the gating function. Thus, SFDDGNN flexibly deals with the incompatibility remain within different graphs, and improves the performance of GNNs in link prediction.

Table 2
Statistics of the datasets used in our experiments.

Statistics	Cora	PubMed	Arxiv2023
Nodes	2708	19,717	19,871
Links	5429	44,337	38,863
Connected Components	79	2	2471
Release Time	2000	2008	2023

4. Experiments

4.1. Research questions

We validate the effectiveness of SFDDGNN by answering the following five research questions:

RQ1 Can our proposed SFDDGNN achieve better performance than the baselines in link prediction?

RQ2 Can SFDDGNN solve the incompatibility between topological structure and node features on different graphs?

RQ3 How is the utility of the dynamic gate fusion mechanism based on graph data distribution analysis?

RQ4 How is the efficiency of our proposed SFDDGNN?

RQ5 How is the robustness of our proposed SFDDGNN?

4.2. Datasets and evaluation metrics

To validate the effectiveness of SFDDGNN in link prediction, we conduct experiments on three real-world datasets of different sizes, i.e., Cora [48], PubMed [49] and Arxiv2023 [5], which are widely adopted in link prediction.

- **Cora** is a citation network dataset containing 2708 papers in the field of computer science and the citation relations between them. Each paper is represented as a node with the node ID, title, abstract, researchers, etc. The citation relations of Cora form a directed graph, where the edges denote the citation relations between papers.
- **PubMed** contains 19,717 biomedical papers, each paper is represented as a node and the edges between the nodes represent the citation relationships between the papers. The node features of the PubMed dataset consist of the node ID, title, and abstract.
- **Arxiv2023** only includes papers published in 2023 or later, which is well beyond the knowledge cutoff for GraphGPT. This dataset prevents the problem of label leakage due to the fact that the same dataset has been used during the training of the graph large model [50]. Arxiv2023 is a subset of the tape-arxiv23 dataset, and Arxiv2023 contains 19,871 paper nodes with 38,863 citation relations. The node features of the Arxiv2023 dataset consist of the node ID, title, abstract, and category.

The statistics of three adopted datasets are shown in Table 2. Moreover, the division ratios of train set, validation set, and test set in the experiments are 80%/10%/10%, respectively. It is noteworthy that the graph datasets employed in our experiments are not the commonly used versions integrated in libraries such as DGL, but rather the raw versions collated by He et al. [5]. Taking the Cora dataset as an example, while the standard version utilizes binary indicators (0 or 1) to denote word presence, our adopted version processed by He et al. [5] preserves node attributes containing original textual information such as paper titles and abstracts. It enables the node feature-based link prediction pipeline in SFDDGNN to capture more comprehensive textual semantic information, which is significant for enhancing the model's capability in understanding contextual relationships between nodes. In addition, following [21,51], average precision (AP) and area under the ROC curve (AUC) are adopted as the evaluation metrics.

4.3. Model summary

The following methods in link prediction are selected as the baselines for comparison with SFDDGNN, including:

1. five heuristic methods, i.e., CN (Common Neighbor) [6], AA (Adamic-Adar) [7], RA (Resource Allocation) [52], SP (Shortest Path) [25], Katz [9];
2. two embedding-based methods, i.e., Node2Vec [12], Matrix Factorization (MF) [13];
3. four vanilla GNNs, i.e., GCN [53], GAT [54], SAGE [55], GIN [45]; and
4. three latest GNN4LP methods, i.e., NEO-GNN [56], SEAL [17], Decouple-SEAL [21].

Five heuristic methods:

- **CN** [6] predicts links between two nodes based on the number of their common neighbors.
- **AA** [7] predicts links between two nodes by considering the degree of their common neighbors, with a particular emphasis on low-degree neighbors.
- **RA** [52] predicts links between two nodes by evaluating the efficiency of resource transfer from node A to node B, taking into account the degree of common neighbors.
- **SP** [25] predicts links between two nodes based on the length of the shortest path connecting them in the graph.
- **Katz** [9] predicts links between two nodes by considering all possible paths between them, with shorter paths being weighted more heavily.

Two embedding-based methods:

- **Node2Vec** [12] employs random walks and the Skip-gram to learn the node embedding, utilizing the similarity in the embedding space to predict links.
- **MF** [13] infers latent features of nodes through matrix decomposition to predict unknown links.

Four vanilla GNNs:

- **GCN** [53] aggregates the neighboring node features through graph convolution operations to update the node representations.
- **GAT** [54] employs an attention mechanism to assign different weights to each neighboring node, thereby considering the importance of nodes during the aggregation of neighbor features.
- **SAGE** [55] generates the node representations by sampling and aggregating features from the neighboring nodes, making it applicable for large-scale graph data.
- **GIN** [45] utilizes a multi-layer perceptron (MLP) to aggregate features from the neighboring nodes, ensuring graph isomorphism and capturing structural information of the graph.

Three latest GNN4LP methods:

- **NEO-GNN** [56] excels in link prediction tasks by extracting useful structural features from the adjacency matrix and leveraging a neighborhood overlap-aware aggregation scheme to estimate the overlapped neighborhoods.
- **SEAL** [17] extracts subgraphs around target links, labels nodes within these subgraphs based on their distances, and then inputs these labeled subgraphs into a GNN for learning.
- **Decouple-SEAL** [21] is a variant of SEAL that separately inputs structure embedding and feature embedding of nodes into different SEALs for link prediction, and linearly combines the predicted probabilities.

4.4. Model implementations

SFDDGNN consists of three parts: structure-based link prediction pipeline, feature-based link prediction pipeline and dynamic gate fusion mechanism. For structure-based link prediction pipeline, we use two layers of MLP to implement the ELSE and two layers of GraphSAGE to generate structure-only embedding. The hidden layers of MLP and GraphSAGE have a dimension of 256 and the output layer of GraphSAGE has a dimension of 128. Following [51], we adopt the Adam optimizer with the learning rate $1e^{-3}$, and the batch size is set to 1024. In addition, the dropout parameter is set to 0.5 and the L2 regularization parameter is set to 1e-5 in order to prevent overfitting. The dimension of structure-only embedding is set to 128. For feature-based link prediction pipeline, we fine-tune the classification layer of the Roberta-base [57] to implement feature encoder, and the batch size is set to 16. In the fine-tuning process, we adopt the Adam optimizer with an initial learning rate $1e^{-3}$ and a decay factor 0.1 for every 3 epochs. We use cosine similarity to control the distance between positive and negative samples to accomplish comparison learning. The dimension of feature-only embedding is set to 128. For dynamic gate fusion mechanism, we use GraphGPT [46] to generate the graph embedding, and we barely adjusted the main internal parameters of the model. The dimension of graph embedding is set to 128. The number of heads in the attention layer is 3. In all three sets of experiments, the ratio between the training set, validation set and test set is 8:1:1. The AP and AUC values shown in Table 3 are the average values of 10 rounds of different random seed experiments. The convergence condition of each experiments is that the performance of the model does not increase in the validation dataset over 50 epochs. We do all of the above experiments on NVIDIA RTX 4090 GPU with 24 GB memory.

5. Results and discussion

5.1. Overall performance (RQ1)

We evaluation the link prediction performance of SFDDGNN and the baselines in Cora, PubMed and Arxiv2023, the results are presented in Table 3. From the experimental results in Table 3, the following conclusions can be drawn:

Among the baselines, the heuristic methods and the embedding-based methods only derive information from the topology, and the superior performance of the embedding-based methods above the heuristics in Table 3 is due to their ability to capture higher-order topology information. Moreover, GNNs propagate node features along the edges. GNNs ignore the incompatibility between node features and topological structure, resulting in a coupled manner, which leads to the unstable performance of the GNNs. In detail, the performance of GNNs is even worse than the embedding-based methods or the heuristic methods in multiple cases. For example, in the Arxiv2023, the AP and AUC of GAT were 78.76%, 72.92%, both lower than Katz. Furthermore, GNN4LP integrates the heuristic information into GNNs, enabling them to capture more comprehensive and beneficial edge-based topological information, leading to their better average performance than other baselines.

We can observe that our proposed SFDDGNN generally achieve better performance than the baselines in most cases in three datasets. This indicates the universal effectiveness of SFDDGNN in link prediction. SFDDGNN can dynamically adapt to graphs from different sources. In detail, compared to other baselines, SFDDGNN achieves top-three levels in both AP and AUC across three datasets. Moreover, in the Arxiv2023 dataset, compared with the baselines with the second best performance, SFDDGNN increases the performance in terms of AP and AUC by 3.54% and 6.55%, respectively.

In addition, for the Cora dataset, we add an additional comparison experiment with Decouple-SEAL. Decouple-SEAL is an improved architecture based on SEAL for reducing the incompatibility between structure and features. This model utilizes GNN to embed node features and topological structure individually, and aggregates them with a linear function. It is similar to SFDDGNN. However, Decouple-SEAL underperforms our model, which may due to the following reasons:

1. Although Decouple-SEAL separates topological structure and node features when embedding nodes through GNN, messages travel along the edges, which may reintroduce topology information, resulting in the incompatibility.
2. Decouple-SEAL adopts the linear gating function for embedding aggregation, which makes it difficult to model the complex relationship between topological structure and node features.

Table 3
Main results on link prediction %). The bolded values indicate the performance metrics of our SFDDGNN across different datasets, while the underlined data denote the optimal performance achieved by the best-performing models in each respective dataset. We only run SEAL and Decouple-SEAL on Cora due to the high memory requirements of the labeling trick.

	Metric	Cora		PubMed		Arxiv2023	
		AP	AUC	AP	AUC	AP	AUC
Heuristic	CN	70.59	70.85	64.87	63.90	70.94	70.96
	AA	71.16	70.96	63.90	63.91	70.96	70.96
	RA	71.14	70.96	63.90	63.92	70.94	70.95
	SP	84.79	81.08	83.68	74.64	81.85	73.23
	Katz	80.08	80.15	79.80	79.94	80.60	79.97
Embedding	Node2Vec	89.02	86.39	88.05	82.63	87.34	82.14
	MF	92.66	91.93	94.09	95.34	69.29	63.86
GNN	GCN	85.92	86.90	86.12	88.86	80.47	77.93
	GAT	86.15	87.83	74.84	75.93	78.76	72.92
	SAGE	88.38	89.56	82.00	84,98	81.89	79.83
	GIN	82.20	82.58	88.20	89.60	84.58	81.33
GNN4LP	NEO-GNN	92.82	92.27	91.61	91.90	85.49	81.50
	SEAL	89.66	88.86	\	\	\	\
	Decouple-SEAL	94.55	94.12	\	\	\	\
SFDDGNN		**95.44**	**94.73**	**92.84**	**89.97**	**90.43**	**87.52**

5.2. Ablation study

5.2.1. Compatibility issues (RQ2)

To explain RQ2, we designed three variants of our proposed SFDDGNN, i.e., making predictions with only structure embedding, only feature embedding, and simple concatenation, respectively. The experimental results are shown in Fig. 2.

From Fig. 2, we can observe that when using only topology information or node features, the experimental results are worse than SFDDGNN. This shows that both the topological structure and the node features contribute to predicting links, and the contribution of their respective information does not completely overlap. Moreover, simple concatenation of structure embedding and feature embedding also causes performance degradation compared to SFDDGNN. It indicates that there exists incompatibility between the topological structure and the node features, and simply connecting the two will cause information conflict. It is noteworthy that the Stru-Only Emb. model in our ablation study demonstrates superior performance compared to the GraphSAGE presented in Table 3. This phenomenon serves two critical reasons:

1. It reinforces the perspective that inherent incompatibility between node feature and topological structure may adversely affect the performance of graph neural networks in handling link prediction tasks;
2. It substantiates the effectiveness of our optimizations to the original GraphSAGE as detailed in Section 3.1.2.

In addition, it should be noted that in the PubMed dataset, the performance of Concatenation Gating decreases more obviously. This phenomenon shows that the topology information plays an important role in the PubMed dataset, and compared with other graph datasets, the incompatibility remain within PubMed is the most obvious, which is consistent with the conclusion in Section 5.1.

5.2.2. Utility of the dynamic gate fusion mechanism (RQ3)

To solve RQ3, we design a self-attention gating function, which serves as a complementary experiment to the ablation experiment. The experimental results are shown in Fig. 2. From Fig. 2, we can see that by comparing different gating functions, the dynamic gate fusion mechanism based on graph data distribution analysis designed in SFDDGNN is the most effective and can better alleviate the incompatibility problem. Moreover, the gating function based on the self-attention mechanism also performs well. This suggests that the information about the incompatibility between node features and structure is hidden in the graph, and mining such deeper information from the graph itself alone is expected to solve the problems posed by the incompatibility. In addition, SFDDGNN mines the information related to incompatibility

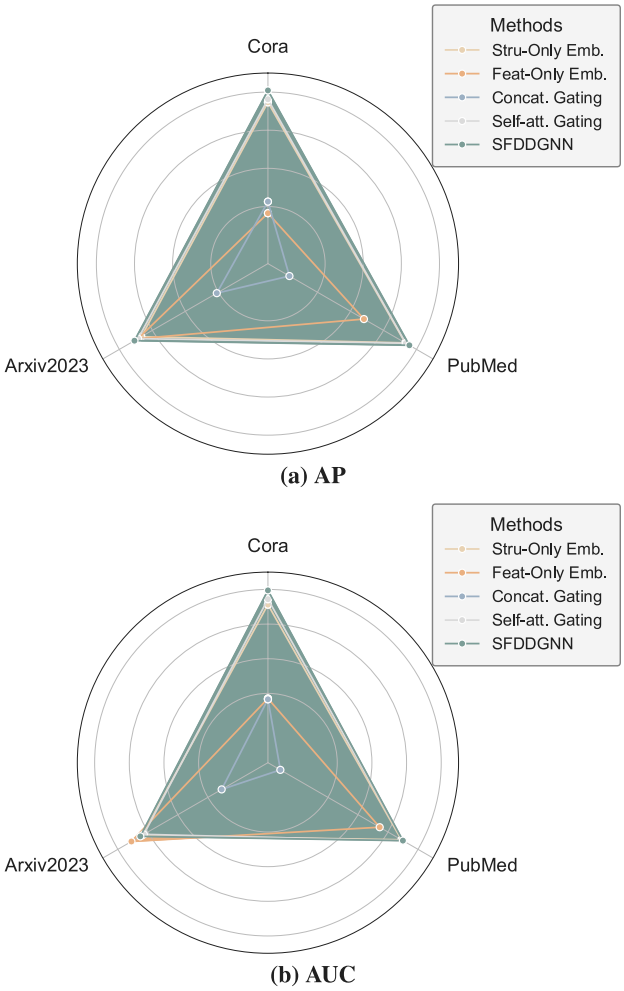


Fig. 2. The results of ablation experiments. Figure (a) shows the AP of different control experiments on different datasets. Figure (b) shows the AUC of different control experiments on different datasets.

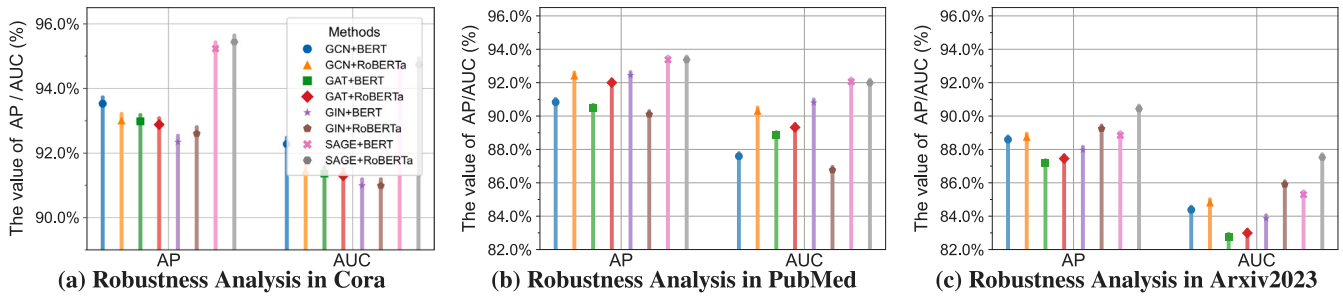


Fig. 3. The results of Robustness Analysis.

Table 4
Model space complexity.

Methods	SEAL	DSEAL	SFDDGNN
Labeling Trick	58.83G	58.83G	\
GNN	36.11G	36.40G	2.95G
LM	\	\	3.13G
Gating Model	\	\	2.78G
Training Memory	58.83G	58.83G	3.13G

by analyzing the graph data distribution, which is used as an explicit guidance to regulate the integration of features and structure, thus leading to the better performance of SFDDGNN.

5.3. Efficiency analysis (RQ4)

To address RQ4, we record the memory occupancy of SFDDGNN, SEAL, and Decouple-SEAL during the experiments to analyze the difference of the space complexity between SFDDGNN and the advanced baselines. The results are shown in Table 4. From Table 4, we can observe that training SFDDGNN requires much less memory than SEAL and Decouple-SEAL. Since SFDDGNN adopts a decoupled approach to capture the node features and topological structure of the graph, it can realize parallelism and reduce the memory occupation during the embedding training process. Meanwhile, the models used in SFDDGNN are classical models with controllable scale, such as GraphSAGE, which ensures the feasibility of its training.

In addition to this, we compare the loss changing during the training of SFDDGNN, SEAL and Decouple-SEAL in link prediction. The results are shown in Fig. 4. It is observed that the loss curve of SFDDGNN decreases faster and smoother, which indicates that the dynamic gate fusion mechanism based on the attention mechanism included in our framework is simple, effective and easy to converge, and thus has good training efficiency.

5.4. Robustness analysis (RQ5)

To address RQ5, we analyze the robustness of SFDDGNN by combining the feature-only embedding and structure-only embedding derived from different models. Specifically, in the structure-based link prediction pipeline, we utilize GCN, GAT, GIN, and SAGE, and in the feature-based link prediction pipeline, we adopt BERT and RoBERTa. Therefore, we design a total of eight controlled experiments with different combinations (including the SAGE+RoBERTa combination used in this paper), namely GCN+BERT, GCN+RoBERTa, GAT+BERT, GAT+RoBERTa, GIN+BERT, GIN+RoBERTa, SAGE+BERT, SAGE+RoBERTa. The experimental results are shown in Fig. 3.

From Fig. 3, we can observe that although the performance of different combinations varies across different datasets, the overall performance is still significantly better than most of the baselines in the current link prediction task. This shows that our framework does not rely on a particular structure embedding or feature embedding approach, demonstrating its robustness.

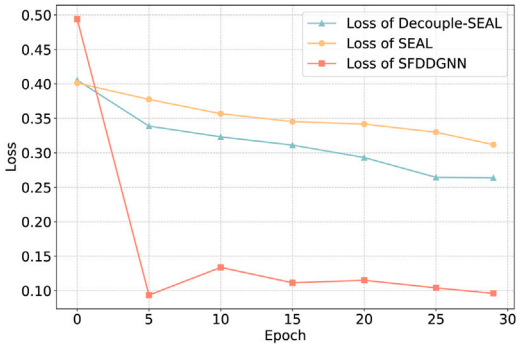


Fig. 4. Training loss.

6. Conclusion

In this paper, to fully capture and effectively aggregate the topological structure and node features of graphs for link prediction, we propose the Structure-Feature Dynamic Decoupling GNN (SFDDGNN) architecture based on the graph data distribution analysis. SFDDGNN consists of two decoupled link prediction pipelines and the dynamic gate fusion mechanism. Structure-based pipeline firstly initializes the node inputs using the Ensemble Learning Structure Encoder (ELSE), and then optimizes the GNN based on the edge-based loss function. Feature-based pipeline embeds node features from a semantic perspective through contrastive learning. The dynamic gate fusion mechanism fuses topological structure and node features based on the graph distribution knowledge to reduce incompatibility. Extensive experiments verify that SFDDGNN performs well in the link prediction task on multiple graphs.

As to future work, on the one hand, we plan to explore the underlying causes of incompatibility so that we can investigate more convenient ways to analyze incompatibility differences across graphs. On the other hand, We plan to compensate for the information loss by supplementing the background knowledge of graphs to improve the GNNs.

CRediT authorship contribution statement

Guowang Li: Writing – original draft, Validation, Methodology, Conceptualization. **Zhiqiang Pan:** Validation, Formal analysis. **Fei Cai:** Writing – review & editing, Resources, Funding acquisition. **Weijie Chen:** Validation, Investigation. **Langgao Cheng:** Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the support by the COSTA: complex system optimization team of the College of System Engineering at NUDT, China.

Data availability

Data will be made available on request.

References

[1] Z. Wu, S. Pan, F. Chen, et al., A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2021) 4–24.
[2] X. Wang, H. Ji, C. Shi, et al., Heterogeneous graph attention network, in: The Web Conference, WWW, ACM, 2019, pp. 2022–2032.
[3] Z. Pan, F. Cai, W. Chen, et al., Star graph neural networks for session-based recommendation, in: Proceedings of the 29th ACM International Conference on Information and Knowledge Management, CIKM, ACM, 2020, pp. 1195–1204.
[4] S. Khanna, S. Bhattacharyya, S. Ghosh, et al., Link prediction for social networks using representation learning and heuristic-based features, 2024, arXiv preprint arXiv:2403.08613.
[5] X. He, X. Bresson, T. Laurent, et al., Harnessing explanations: LLM-to-LM interpreter for enhanced text-attributed graph representation learning, in: 12th International Conference on Learning Representations, ICLR, 2024.
[6] M. Newman, Clustering and preferential attachment in growing networks, Phys. Rev. E 64 (2) (2001) 025102.
[7] L.A. Adamic, E. Adar, Friends and neighbors on the Web, Soc. Netw. 25 (3) (2003) 211–230.
[8] P. Jaccard, Etude de la distribution florale dans une portion des Alpes et du Jura, Bull. Soc. Vaudoise Sci. Nat. 37 (142) (1901) 547–579.
[9] L. Katz, A new status index derived from sociometric analysis, Psychometrika 18 (1) (1953) 39–43.
[10] H. Wang, Z. Wei, J. Gan, et al., Personalized PageRank to a target node, in: The 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, ACM, 2020, pp. 657–667.
[11] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: online learning of social representations, in: The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, ACM, 2014, pp. 701–710.
[12] A. Grover, J. Leskovec, Node2vec: Scalable feature learning for networks, in: The 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD, ACM, 2016, pp. 855–864.
[13] A.K. Menon, C. Elkan, Link prediction via matrix factorization, in: Machine Learning and Knowledge Discovery in Databases - European Conference (ECML/PKDD), Springer, 2011, pp. 437–452.
[14] Z. Liu, X. Yu, Y. Fang, et al., GraphPrompt: Unifying pre-training and downstream tasks for graph neural networks, in: The Web Conference, WWW, ACM, 2023, pp. 417–428.
[15] X. Wang, H. Yang, M. Zhang, Neural common neighbor with completion for link prediction, in: 12th International Conference on Learning Representations, ICLR, 2024.
[16] Z. Zhu, Z. Zhang, L.A.C. Xhonneux, et al., Neural Bellman-Ford networks: A general graph neural network framework for link prediction, in: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS, MIT Press, 2021, pp. 29476–29490.
[17] M. Zhang, P. Li, Y. Xia, et al., Labeling trick: A theory of using graph neural networks for multi-node representation learning, in: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS, MIT Press, 2021, pp. 9061–9073.
[18] Y. Zheng, S. Luan, L. Chen, What is missing in homophily? Disentangling graph homophily for graph neural networks, 2024, arXiv preprint arXiv:2406.18854.
[19] S. Hou, Y. Liu, M. Tang, Identifying influential nodes in spreading process in complex networks by integrating node dynamic propagation features and local structure, Acta Phys. Sin. (2025) 1–25.
[20] B. Wang, B. Cai, J. Sheng, et al., AAGCN: a graph convolutional neural network with adaptive feature and topology learning, Sci. Rep. 14 (1) (2024) 1–12.
[21] H. Mao, J. Li, H. Shomer, et al., Revisiting link prediction: a data perspective, in: 12th International Conference on Learning Representations, ICLR, 2024.
[22] L. Yang, W. Zhou, W. Peng, et al., Graph neural networks beyond compromise between attribute and topology, in: The Web Conference, WWW, ACM, 2022, pp. 1127–1135.
[23] X. Liu, J. Chen, Q. Wen, A survey on graph classification and link prediction based on GNN, 2023, arXiv preprint arXiv:2307.00865.
[24] Y. Wang, T. Zhao, Y. Zhao, et al., A topological perspective on demystifying GNN-based link prediction performance, in: 12th International Conference on Learning Representation, ICLR, 2024.
[25] D. Liben-Nowell, J.M. Kleinberg, The link prediction problem for social networks, in: Proceedings of the 12th ACM International Conference on Information and Knowledge Management, CIKM, ACM, 2003, pp. 556–559.
[26] T. Mikolov, K. Chen, G. Corrado, et al., Efficient estimation of word representations in vector space, in: 1st International Conference on Learning Representations, ICLR, 2013.
[27] B. Srinivasan, B. Ribeiro, On the equivalence between positional node embeddings and structural graph representations, in: 8th International Conference on Learning Representations, ICLR, 2020.
[28] J.M. Giorgi, O. Nitski, B. Wang, et al., DeCLUTR: Deep contrastive learning for unsupervised textual representations, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, ACL, 2021, pp. 879–895.
[29] Y. Chung, Y. Zhang, W. Han, et al., w2v-BERT: Combining contrastive learning and masked language modeling for self-supervised speech pre-training, in: IEEE Automatic Speech Recognition and Understanding Workshop, ASRU, IEEE, 2021, pp. 244–250.
[30] Y. Qin, Y. Lin, R. Takanobu, et al., ERICA: improving entity and relation understanding for pre-trained language models via contrastive learning, in: Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP, ACL, 2021, pp. 3350–3363.
[31] J. Cui, Z. Zhong, S. Liu, et al., Parametric contrastive learning, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV, IEEE, 2021, pp. 695–704.
[32] Y. Meng, L. Zhao, L. Xu, Unsupervised image-to-image translation with patch pyramid dual contrastive learning for cross domain object detection, in: IEEE International Conference on Multimedia and Expo, ICME, IEEE, 2022, pp. 1–6.
[33] M. Zheng, F. Wang, S. You, et al., Weakly supervised contrastive learning, in: 2021 IEEE/CVF International Conference on Computer Vision, ICCV, IEEE, 2021, pp. 10022–10031.
[34] Y. Xie, Z. Xu, J. Zhang, et al., Self-supervised learning of graph neural networks: A unified review, IEEE Trans. Pattern Anal. Mach. Intell. 45 (2) (2023) 2412–2429.
[35] L. Liu, Q. Xie, W. Wen, et al., Edge contrastive learning for link prediction, Inf. Process. Manage. 61 (6) (2024) 103847.
[36] Z. Zhang, S. Sun, G. Ma, et al., Line graph contrastive learning for link prediction, Pattern Recognit. 140 (2023) 109537.
[37] K. Duan, Q. Liu, T. Chua, et al., SimTeG: A frustratingly simple approach improves textual graph learning, 2023, arXiv preprint arXiv:2308.02565.
[38] M. Sun, J. Yang, J. Yang, Topology description for data distributions using a topology graph with divide-and-combine learning strategy, IEEE Trans. Syst. Man Cybern.: Syst. 36 (6) (2006) 1296–1305.
[39] G. Cimini, T. Squartini, F. Saracco, et al., The statistical physics of real-world networks, 2018, arXiv preprint arXiv:1810.05095.
[40] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[41] A.-Í. Barabási, R. Albert, Emergence of scaling in random networks, Science 286 (5439) (1999) 509–512.
[42] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 22 (8) (2000) 888–905.
[43] M.K. Cameron, E. Vanden-Eijnden, Flows in complex networks: Theory, algorithms, and application to Lennard–Jones cluster rearrangement, J. Stat. Phys. 156 (3) (2014) 427–454.
[44] H. Zhang, P. Li, R. Zhang, et al., Embedding graph auto-encoder for graph clustering, IEEE Trans. Neural Netw. Learn. Syst. 34 (11) (2023) 9352–9362.
[45] K. Xu, W. Hu, J. Leskovec, et al., How powerful are graph neural networks? in: 7th International Conference on Learning Representations, ICLR, 2019.
[46] J. Tang, Y. Yang, W. Wei, et al., GraphGPT: Graph instruction tuning for large language models, in: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR, ACM, 2024, pp. 491–500.
[47] X. Zhang, W. Yang, J. Feng, et al., GSpect: Spectral filtering for cross-scale graph classification, 2024, arXiv preprint arXiv:2409.00338.
[48] A.K. McCallum, K. Nigam, J. Rennie, et al., Automating the construction of internet portals with machine learning, Inf. Retr. 3 (2) (2000) 127–163.
[49] P. Sen, G. Namata, M. Bilgic, et al., Collective classification in network data, AI Mag. 29 (3) (2008) 93–106.
[50] A. Srivastava, A. Rastogi, A. Rao, et al., Beyond the imitation game: Quantifying and extrapolating the capabilitiesof language models, 2023, arXiv preprint arXiv:2206.04215.
[51] L. Ma, H. Han, J. Li, et al., Mixture of link predictors, 2024, arXiv preprint arXiv:2402.08583.
[52] T. Zhou, L. Lü, Y.-C. Zhang, Predicting missing links via local information, Eur. Phys. J. B 71 (4) (2009) 623–630.
[53] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, 2016, arXiv preprint arXiv:1609.02907.

[54] P. Velickovic, G. Cucurull, A. Casanova, et al., Graph attention networks, in: 6th International Conference on Learning Representations, ICLR, 2018.
[55] W.L. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, in: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NeurIPS, MIT Press, 2017, pp. 1024–1034.
[56] S. Yun, S. Kim, J. Lee, et al., Neo-GNNs: Neighborhood overlap-aware graph neural networks for link prediction, in: Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, NeurIPS, MIT Press, 2021, pp. 13683–13694.
[57] Y. Liu, M. Ott, N. Goyal, et al., RoBERTa: A robustly optimized BERT pretraining approach, 2019, arXiv preprint arXiv:1907.11692.