SEGNN4SLP: Structure Enhanced Graph Neural Networks for Service Link Prediction

Yuxi Lin
School of Computer Science and Technology
Hainan University
Hainan, China
E-mail: 13654669668@163.com

Nuo Chen
School of Computer Science and Technology
Hainan University
Hainan, China
E-mail: nuochen0107@163.com

Mengfei1 Li
School of Computer Science and Technology
Hainan University
Hainan, China
E-mail: 18346011668@163.com

***Abstract*—For the provision of accurate link prediction, this study's neural network-based method for API recommendation uses structure encoding to capture topological context. SEGNN4SLP, a Graph Neural Network (GNN) framework that integrates node attributes and graph structure to enhance GNNs' link prediction skills, makes a substantial contribution. Utilizing an actual dataset with 21,900 APIs, 6,435 Mashups, and 13, 340 interactions, ProgrammableWeb.com was the source of the evaluation. Eighty percent of the data were test sets and twenty percent were training sets after single API-invocation Mashups were eliminated. The results demonstrate high link prediction accuracy, which is attributed to the incorporation of structural encoding in embedding learning and improved collaborative signal extraction from users and APIs, which improves API recommendation performance overall.**

Keywords-Network Representation; Web Service; Mobile Network; Graph Attention network; Link Prediction

I. INTRODUCTION

The advancement of service computing technologies and the rise of service markets have resulted in the proliferation and consumption of an increasing variety of services (such as APIs and Mashups) in diverse application situations. [1]. Mashup represents a lightweight Web application that consists of multiple existing Web APIs or services in a flexible manner to meet the complex application needs of users.

According to Programmable Web's statistics, there is a concentration of usage since the top 10(200) most often used Web API calls in Mashups account for around 30.6% (99%) of all Mashup calls [2]. Consumers may ignore lesser-known APIs because they frequently rely on these well-known ones [3]. By utilizing implicit co-call records between APIs in past Mashups, it is possible to forecast the likelihood of usage of less popular APIs, which helps to solve the problem of users missing out on potentially useful APIs for their Mashup requirements.

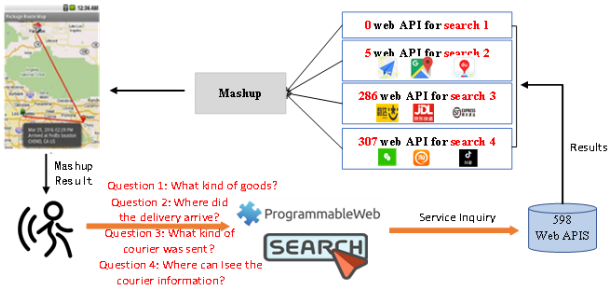

Figure 1 Maup example schematic

The service Link Prediction results can be used to diversify Web API recommendations [4]. With the use of service graph data [4], Graph Neural

Networks (GNNs) have emerged as a potent framework for service management applications. By using a message passing paradigm to recursively collect neighborhood vectors, they are very good at predicting service links [5]. Nevertheless, conventional GNNs merely convey node properties while passing messages; they do not explicitly take topological information into account, which has been shown to be advantageous in topology-based techniques.

The design and encoding of structural elements and their integration into GNNs for service link prediction are the two primary difficulties that this work attempts to solve. A topology-based strategy provides a Path Labeling (PL) method to extract structural information in order to address the first difficulty. An encoder is then used to transform these features into structural embeddings. The Network Topology Structure Enhanced Graph Neural Network (SEGNN4SLP) incorporates configurational embeddings into GNNs to tackle the second challenge. To improve GNN speed, SEGNN4SLP maps structural embeddings to the same space as the original node features using a feature fusion module. By utilizing both structural and attribute information, SEG can optimize link prediction through the joint training of the structural encoder and GNN. The pipeline consists of labeling nodes according to their positional roles, creating structural embeddings, fusing them with GNNs, and extracting a closed 1-hop subgraph surrounding target nodes. This fusion forecasts the presence of linkages between target nodes when paired with the results of the GAT model.

II. METHODOLOGY OF SEGNN4SLP

This research presents the SEGNN4SLP framework, which includes node embedding and structure encoding, for service link prediction (Figure 2). A closed 1-hop subgraph is extracted around two target nodes in order to forecast linkages between them. To extract structural characteristics, each node is labeled according to its positional role in the subgraph; structural encoding is then used to construct structural embeddings. The GAT model receives these embeddings fused with node attributes as input. The presence of a link between nodes a and b is predicted by the correlation between the structural embeddings of the target nodes and the output of the GAT model.

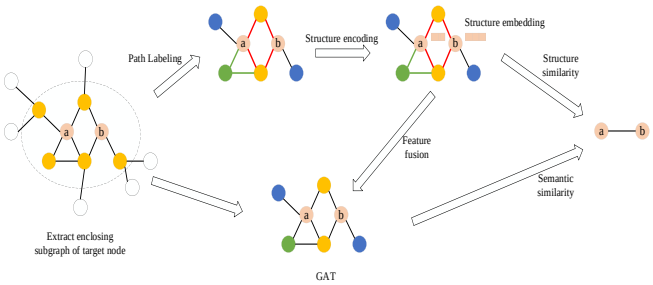

Figure 2 The SEGNN4SLP structure

A. Structure Encoding

Most GNN models primarily utilize edges for message delivery, overlooking network topology. Incorporating network topology as additional input can significantly enhance network embedding quality. This section discusses designing and encoding structural features. Various local patterns around target nodes exist. Assessing both node topology and connecting edges is essential for measuring node correlation. Graph structure methods, including Common Neighbors (CN), Jaccard, Adamic/Adar (AA), and Katz measures, aid in link forecast missions. These approaches can be unified below:

$$s(i,j)=\sum_{l=1}^{\infty} f(l,N(i),N(j))\phi(p_{i,j}^{l}) \quad (1)$$

where $s(i,j)$ denotes the similarity between nodes v_i and v_j; $p_{i,j}^{l}$ denotes the number of nodes v_i and v_j at path length l; $N(i),N(j)$ denotes the neighbors of nodes v_i and v_j.

The pathways of target nodes and their surrounding nodes $p_{i,j}^{l}$ are two crucial elements $N(i),N(j)$ in Equation 1. Although Graph Neural Networks (GNNs) already integrate information from neighboring nodes, routes provide further inputs to GNN models, highlighting their importance. The proposition characterizes the route between two target nodes as a specific form

of topological attribute, described in the following manner:

Path Labeling: Where p_{ij} denotes the path between the target node v_i and v_j. The path has no duplicate vertices and duplicate edges. To represent these paths as topologies available to the nodes, Path Labeling (PL) is proposed to label the nodes under each different path and assign values for example, the nodes of the target node 1 hop neighbors have the same Path Labeling. For nodes that appear in various routes in the meantime, the shortest route is chosen to label the nodes.

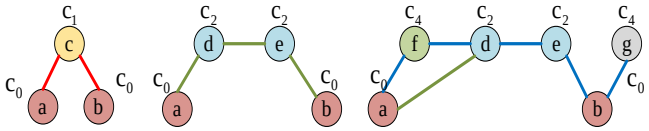

Figure 3 The SEGNN4SLP framework demonstrates the use of route labeling (PL) on a 1-hop subgraph that includes nodes a and b. In (a), we notice a route with a length of 2, in (b) a route with a length of 3, and in (c) a route with a length of 4. Every unique pathway is depicted using a distinct hue. The nodes are labeled and the labels are presented above the nodes. Nodes with different labels are shown in distinct colors.

Figure 3 shows a schematic diagram of the path marking in a subgraph consisting of 1-hop neighbors of two target nodes. As in Figure 3(a), Algorithm 1 : structure encoding

node c is a common neighbor of node a and node b, marked as; as in Figure 3(b), node a and node b are on the path of path length node 3 (a-d-e-b), marking nodes d and e as; as in Figure 3(c), nodes d and e are on the route of path length 3 (a-d-e-b) and length 4 (a-f-d-e-b), and the shorter route is selected to label nodes d and e, labeled as; nodes f and g are on the path of path length 4, labeled as.

By labeling the nodes under different paths by PL, for these topological characteristics from the routes of every pair of target nodes, a structural encoding method is additionally proposed to learn the representation vectors from them. The definition of the encoding approach is shown below:

In Algorithm 1, GCN layer and MLP are utilized to fit f and ϕ in Equation 1. With proper coefficients and sufficient layers, it is hoped that the approximation error can be neglected. In addition, A constant is used λ to truncate paths that are too long, and these differences prevent PL overfitting and make the model more robust.

input: target nodes v_i, v_j; enclosing subgraph G_s	
output: node embedding *z*	
1 /*extracts the routes*/	2 $p_{i,j}$ □(G,i,j)
3 /* generate node structural features */	4 $c_u \leftarrow \{p_{i,j}, G_S\}, \forall_u \in G_S;$
5 $Z_u^{(0)} \leftarrow \text{one}-\text{hot}(\min(c_u, \lambda)), \forall_u \in G_S;$	6 /* encode with a GCN layer and a MLP */
7 for $u \in G_S$ do	8 $Z_u^{(1)} \leftarrow \text{AGGREGATE}(Z_v^{(0)}, \forall_u \in N(u));$
9 end for	10 $Z_u \leftarrow \text{MLP}(z_u), \forall_u \in G_s;$

B. *Node Embedding Representation*

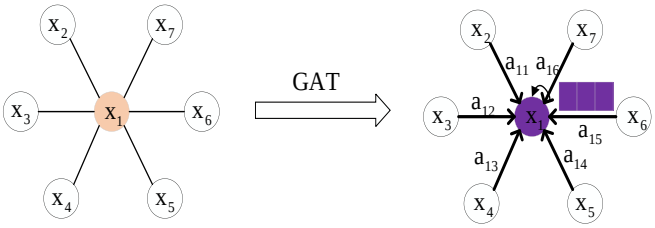

Figure 4 Assign different coefficients between nodes by GAT.

The utilization of an undirected graph to describe a network consisting of online APIs and Mashups enables the application of graph neural network techniques for obtaining vector representations of nodes. One way to accomplish this is by using Graph Attention Networks (GAT) to assign different weight coefficients to nodes and combine the information from neighboring nodes, including the target node's own features, to update and obtain a new vector representation of the

target node. The influence between nodes i and j can be mathematically represented as:

$$e_{ij} = a\left(Wh_i, Wh_j\right) \tag{2}$$

where a denotes the attention parameter of node i on node j, W denotes the weight matrix, and h_i, h_j denotes the vector representation of node i and j. Besides, the impact of node i on node j is not equivalent to the impact of node j on node i , $e_{ij} \neq e_{ji}$.

After obtaining the importance between pairs of nodes based on the same route, the softmax function is used for normalization.

$$a_{ij} = \frac{\exp\left(\sigma\left(e_{ij}\right)\right)}{\sum_{k \in N_i} \exp\left(\sigma\left(e_{ik}\right)\right)} \tag{3}$$

where σ denotes the activation function, $\|$ denotes the splicing operation, a_{ij} denotes the attention coefficients of the lower nodes i and j, and N_i represents the set of neighbors of node i .

The normalized coefficient is used to calculate the weighted mean of the transformed characteristics of neighbor nodes (nonlinear activation function is used) as the new feature vector representation of node i:

$$h_i = \sigma\left(\sum_{j \in N_i} a_{ij} W h_j\right) \tag{4}$$

After obtaining the new vector representation of the node, the next consideration is how to fuse the node vector and the structure information.

C. Feature Fusion

An effective approach to acquire knowledge about both node properties and structural data is to directly feed them into Graph Neural Networks (GNNs) such as SEAL [6]. Nevertheless, these two categories of characteristics have varied meanings: node properties usually provide semantic data, while structural characteristics are directly obtained from the graph topology. Hence, acquiring proficiency in both semantic and structural knowledge presents a considerable obstacle.

A framework called SEGNN4SLP has been developed to combine these two functionalities. Figure 6 depicts the intricate architecture. The SEGNN4SLP architecture consists of two components: a structural encoder and a deep Graph Neural Network (GNN). In the structural encoder, the process begins with encoding the node structural features (c_u) having a single layer of GCNs. The reason for using a single layer of Graph Convolutional Networks (GCNs) is that the multi-hop data is naturally present in the route. Therefore, a single layer of aggregation suffices to upgrade the two desired target nodes. The output of the GCN layer are adopted as the input of a MLP to fit f and ϕ in Equation1. The output vectors are denoted z_u of MLP as the configurational embeddings of v_u. The structure similarity score $S_{structure}$ is predicted with another MLP on basis of the Hadamard product of structural embeddings of target nodes z_i and z_j.

$$S_{structure} = MLP\left(z_i \circ z_j\right) \tag{5}$$

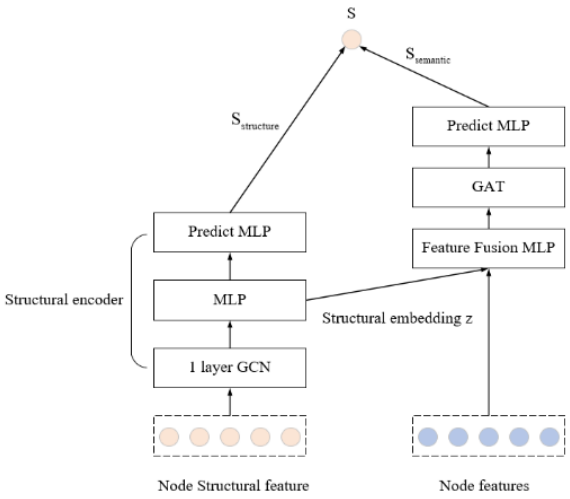

Figure 5 Architecture of SEGNN4SLP

In the deep GNN model, the configurational embeddings z_u and the original node properties x_u are originally combined using a Multilayer Perceptron (MLP). The feature fusion module combines the features of x and z into a unified feature space. The deep GNN model can utilize the fused features as input to learn from both the structural characteristics and the original node characteristics. In this context, a Graph Attention Network (GAT) is selected as the primary Graph Neural Network (GNN) model, and the SortPooling [7] layer is utilized to obtain the ultimate representation output of the two specific nodes of interest. An MLP is used to forecast the high semantic similarity score, represented as $S_{semantic}$. The structural similarity mark and the semantic similarity mark are subsequently merged to ascertain the ultimate probability of link presence:

$$S = S_{structure} + S_{semantic} \quad (6)$$

For better learning of the model parameters, the SEGNN4SLP model uses cross entropy as the loss function, defined below:

$$loss = \frac{1}{N}\sum_{t=1}^{N} -y_t \log s_t + (1-y_t)\log(1-s_t) \quad (7)$$

Where s_t denotes the fraction of possible links t; y_t denotes the label of link t; N denotes the number of training edges. The function reacts similar embedding of friends and dissimilar embedding of enemies. The cross-entropy loss is reduced constantly to update the coefficients, and the vector representation Z of nodes is got when the loss tends to be stable after several optimizations, and the algorithm procedure is specified below.

Stable after several optimizations, and the algorithm procedure is specified below.

Algorithm 2 : SEGNN4SLP

input: target edge (i,j); input graph G ; node characterizes X	
output: forecast score s,	
1 /* extracts enclosing subgraph */	2 $G_s \leftarrow G$
3 $z_u \leftarrow$ Structural Encoding$(G_s, i, j), \forall_u \in G_s$;	4 $S_{structure} \leftarrow MLP(z_i \circ z_j)$
5 /* feature fusion */ $c_u \leftarrow \{p_{i,j}, G_s\}, \forall_u \in G_s$; $\tilde{x}_u^{(0)} \leftarrow x_u^{(0)}, \forall_u \in G_s$; $\tilde{x}_u \leftarrow MLP(\tilde{x}_u^{(0)}), \forall_u \in G_s$; $h_u^{(0)} \leftarrow \tilde{x}_u, \forall_u \in G_s$;	
6 /* GNN message passing */for k=1,2…K do: for $u \in G$ do: $h_u^{(k)} \leftarrow$ *Equation* 4; endend	
7 $h_G \leftarrow SortPool(h_u^{(k)} \mid u \in G_s, k = 1,\ldots,K)$;	8 $S_{semantic} \leftarrow MLP(h_G)$
9 $S = S_{structure} + S_{semantic}$	

III. EXPERIMENTS

A. Dataset description

This study's methodology was meticulously assessed through a series of controlled trials performed on Programmable Web (PW), the largest and most renowned public repository for web APIs. PW functions as an extensive platform that rigorously aggregates and methodically organizes a wide range of data related to web APIs and their corresponding applications. The study

concentrated on the methodical analysis of the web APIs and mashups present on PW, specifically highlighting the assessment of the interactions between these APIs and their users, referred to as mashups.

The dataset employed for these assessments comprises a significant aggregation of 21, 900 individual APIs, 6,435 unique mashups, and a comprehensive account of 13, 340 specific interactions between these mashups and APIs. Table 1 presents a detailed summary of the experimental dataset, encompassing essential measurements and properties vital to the evaluation process. To guarantee the robustness and validity of the evaluation, the study intentionally removed mashups comprising only a single API call from the dataset, thereby concentrating on more intricate and representative interactions.

For the evaluation, the dataset was carefully divided into two separate subsets: 80% of the exchange records were allocated as the training set, employed to construct and enhance the models and methodologies under investigation. Twenty percent of the data was designated for the test set, acting as the essential benchmark for evaluating the performance and effectiveness of the trained models. This stratified method guaranteed a thorough and impartial examination, establishing a solid basis for analyzing the technique's relevance and efficacy in practical situations.

B. Evaluation metrics

User preferences can be output by every model for all APIs. To assess the effectiveness of Top-K recommendation and user preference ranking, two assessment metrics are employed. Recall@K represents the proportion of actual APIs in the top - K API recommendation list to the actual APIs required by user preferences. Its definition is shown below:

$$\mathrm{Re}call@k = \frac{|\{actural\ APIs\} \cap \{\mathrm{topk}\ APIs\}|}{|\{actural\ APIs\}|} \quad (8)$$

nDCG@K gives varying weights to every API in the top - K recommendation list, with higher-ranked APIs receiving bigger weights. One of its commonly adopted definitions is:

$$DCG@\mathrm{k} = \sum_{i=1}^{n} \frac{2^{rel(i)} - 1}{\log_2(i+1)} \quad (9)$$

$$IDCG@\mathrm{k} = \sum_{i=1}^{\mathrm{c}} \frac{1}{\log_2(i+1)} \quad (10)$$

$$nDCG@\mathrm{k} = \frac{DCG@k}{IDCG@k} \quad (11)$$

TABLE I COMPARISON OF DIFFERENT METHODS IN RECALL@K.

	K=5	K=10	K=15	K=20	K=25
Node2vec	0.2185	0.2915	0.3473	0.3761	0.4012
GCN	0.2729	0.3461	0.3684	0.4561	0.4716
GraphSAGE	0.2816	0.3553	0.3941	0.4611	0.4933
GAT	0.2810	0.3513	0.3902	0.4687	0.4910
SEAL	0.2984	0.3588	0.4013	0.4701	0.4987
SEGNN4SLP	0.3514	0.3981	0.4586	0.4981	0.5231

TABLE II COMPARISON OF DIFFERENT METHODS IN NDCG@K.

	K=5	K=10	K=15	K=20	K=25
Node2vec	0.2314	0.2786	0.3278	0.3529	0.3604
GCN	0.2811	0.3378	0.3588	0.3687	0.3786
GraphSAGE	0.2823	0.3468	0.3770	0.3819	0.3793
GAT	0.2811	0.3398	0.3764	0.3987	0.3859
SEAL	0.2994	0.3410	0.3896	0.4055	0.3986
SEGNN4SLP	0.3516	0.3814	0.4156	0.4258	0.4288

C. Baseline methods

In order to verify the effectiveness of our proposed method, we choose the following method to compare with our proposed method:

Node2vec is a traditional graph embedding technique that represents nodes as low-dimensional vectors. Node2vec utilizes random walks to effectively capture both local and global structures inside a graph, rendering it a versatile instrument for many graph-related tasks. After embedding the nodes into low-dimensional vectors, a Multilayer Perceptron (MLP) predictor is subsequently employed. This predictor employs a

combination of the original node features and the Node2vec output vector as input, thereby incorporating both structural information and intrinsic node qualities to improve predictive performance. The MLP adeptly models intricate, nonlinear relationships, efficiently utilizing this enhanced feature set to generate precise predictions.

Conversely, Graph Convolutional Networks (GCNs) are prominent neural networks that characterize graphical convolution via spectrum analysis. Graph Convolutional Networks (GCNs) function by altering and disseminating node attributes via the graph's Laplacian matrix, thereby encapsulating the spectral characteristics of the graph. This methodology enables GCNs to intrinsically comprehend and leverage the graph's topology, rendering them exceptionally proficient for jobs like node classification and graph categorization.

GraphSAGE, a prevalent graph neural network, presents an innovative methodology by utilizing sampling and aggregation techniques to facilitate inductive learning for previously unobserved nodes. In contrast to transductive approaches that necessitate the complete graph during training, GraphSAGE generalizes by acquiring the ability to aggregate feature information from local neighborhoods. The inductive feature of GraphSAGE enables it to manage graphs with changing structures, rendering it especially advantageous in dynamic situations when the graph is not entirely known in advance.

SEAL (Subgraph Embedding Attributed Link prediction) is a link prediction technique that derives link representations from tagged subgraphs using the Deep Graph Convolutional Neural Network (DGCNN). SEAL functions by extracting subgraphs surrounding prospective edges and subsequently employing DGCNN to derive embeddings that include the structural and semantic attributes of these subgraphs. This acquired knowledge is then utilized to forecast the probability of connections, offering a solid and comprehensible method for link prediction.

Graph Attention Networks (GAT) incorporate attention mechanisms into graph neural networks based on spatial domains. GATs dynamically allocate varying weights to adjacent nodes according on their significance to the center node, thus enhancing node attributes through the weighted representation of neighboring nodes. This attention-based methodology enables Graph Attention Networks (GATs) to concentrate on the most significant relationships within the graph, hence improving its efficacy in capturing intricate dependencies and interactions among nodes.

In conclusion, these methods exemplify a range of strategies for utilizing graph structures in diverse machine learning applications. Node2vec, GraphSAGE, and GAT each provide distinct advantages that render them appropriate for certain applications and contexts. Collectively, they constitute a comprehensive toolkit for tackling various graph-related issues.

D. Experimental results

The efficacy of the proposed strategy will be thoroughly assessed in comparison to the previously mentioned baseline approach. The findings in Table 1 unequivocally demonstrate that the suggested method regularly surpasses the baseline in all cases. The technique exhibits a about 13% improvement over the ideal baseline when assessed using Recall@K. This substantial enhancement is especially remarkable considering that, in fact, users generally need less than 5 APIs to create a Mashup. Thus, the marginal advantages of recall tend to decrease as the quantity of suggested APIs rises.

The experimental findings highlight the considerable advantages of integrating higher-order connectivity data, markedly enhancing the recommendation effect. Furthermore, it is clear that the performances of both GAT (Graph Attention Network) and GraphSAGE are inferior to the suggested technique. This comparative research reinforces the practicality and importance of incorporating the network structure's topology into the node representations. The proposed method integrates topological insights with node embeddings, thereby improving the precision of recommendations and facilitating a deeper comprehension of the network's structural dynamics.

In this section, ablation experiments are done for the core components of the model: SEGNN4SLP-1 indicates structural encoding only: SEGNN4SLP-2 removes structural encoding and learns node embedding with GAT only; SEGNN4SLP indicates fusion of structural encoding and node embedding, which is the method we propose. The experimental results are shown in Table Ⅲ. The SEGNN4SLP method has a significant improvement in Recall and nDCG values compared with SEGNN4SLP-1 and SEGNN4SLP-2. This indicates that the fusion structure encoding to node embedding helps to improve the prediction quality of the link.

TABLE III RESULTS FOR SEGNN4SLP, SEGNN4SLP-1，SEGNN4SLP-2.

Methods	Recall		nDCG	
	Recall@5	Recall@25	nDCG@5	nDCG@25
SEGNN4SLP-1	0.3389	0.4844	0.3486	0.3855
SEGNN4SLP-2	0.3284	0.4964	0.3357	0.3746
SEGNN4SLP	0.3598	0.5287	0.3617	0.4137

E. Hyper parameters analysis

In this subsection, we discuss the effect of the model hyperparameters used for data training on the recommendation performance. We fixed the other parameters and changed only the hyperparameters to conduct the experiments. The hyperparameters include user or API embedding dimension d, path control length .

Figure 6 shows the embedding size of the user and API for Recall@25. We can observe that increasing the embedding size of the user and API initially improves recommendation performance. More specifically, when the embedding size increases from 16 to 32, Recall@25 increases from 0.5145 to 0.5297. However, when the embedding size exceeds 32 Recall@25 the value starts to decline rapidly. This observation suggests that a moderate embedding size can provide sufficient information storage space during training. If the embed size is too small, information about some users or apis in the embed may be lost. On the contrary, if the embedding size is too large, it may lead to information redundancy and increase the time overhead of model training.

Figure 6 shows the effect of path length at Recall@25. We can observe that increasing the length of the target node path initially improves the recommended performance. More specifically when the length =1 grows to =3, Recall@25 increases from 0.511 to 0.5264. However, the Recall@25 value starts to decrease rapidly when the length grows. This observation suggests that a moderate path length allows for the best topology efficacy.

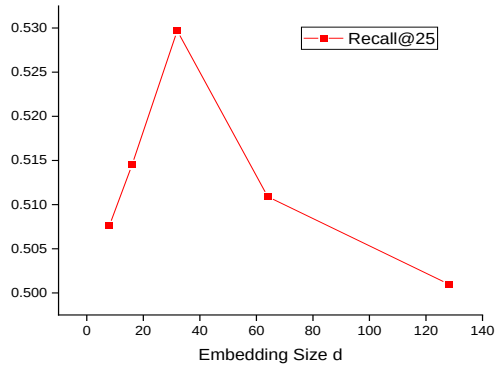

Figure 6 Impact of different embedding size d

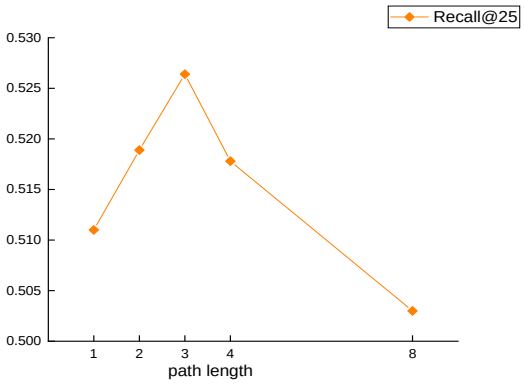

Figure 7 Impact of different path length λ

IV. Related work

Most current research in service categorization and recommendation predominantly focuses on extracting unstructured data using document representation techniques. These strategies typically entail steps such as aligning keywords identified in other service descriptions or assessing the semantic proximity between various services. The classification outcomes often aggregate services with analogous characteristics into a singular category. Nevertheless, these keyword-centric methodologies are significantly dependent on the quality and pertinence of the terms contained inside the database. Furthermore, service descriptions are frequently articulated manually by service providers, potentially leading to inconsistencies and mistakes that undermine the overall precision of service classification.

To address the constraints of keyword-based approaches, researchers have commenced the investigation of diverse semantic-based service categorization methodologies. These methods often entail the extraction of probabilistic topics from service descriptions through sophisticated vector space models to assess the similarity between services and categorize them accordingly. Notable instances of these methodologies encompass the probabilistic topic models PLSA (Probabilistic Latent Semantic Analysis) and LDA (Latent Dirichlet Allocation), in addition to neural network-driven document embedding approaches. These methodologies generally entail initially acquiring prospective subject or functional unstructured vectors to represent service documents. Consequently, suitable classifiers are trained according on the similarity among these vectors. Topic models are particularly efficacious as they can convert the high-dimensional document word vector space into a more tractable low-dimensional unstructured vector space. Nonetheless, a significant shortcoming of these methods is their frequent neglect of the discourse order information embedded in textual data, which is essential for comprehending the context and semantics of service descriptions.

In recent years, Graph Neural Networks (GNN) have developed as a potent deep learning technique for extracting properties of network relationships. Graph Neural Networks (GNNs) have been extensively utilized throughout multiple fields of service computing, encompassing service combination, service recommendation, service clustering, and service categorization. Many of these applications concentrate on deriving network characteristics from service isomorphic graphs. A burgeoning cohort of academics acknowledges the capacity of GNNs to elucidate concealed network structural attributes through the formulation of meta-paths or meta-graphs that integrate various node and edge kinds. This method utilizes diverse information to acquire more efficient and complete service network data. Researchers seek to improve the precision and comprehensiveness of service classification and recommendation systems by integrating GNNs with diverse graph structures, hence offering more sophisticated and contextually enriched insights into service functionality and interrelations.

In conclusion, whereas conventional keyword-based and preliminary semantic-based service categorization approaches possess advantages, they are also accompanied by considerable drawbacks, especially regarding keyword quality and the absence of discourse order information. The emergence of GNNs signifies a substantial advancement in tackling these difficulties, providing a more refined and adaptable method for extracting and employing network properties to enhance service classification and recommendation.

V. Conclusions

This research thoroughly examines a neural network-based API recommendation methodology that utilizes a sophisticated method called structural encoding. This technique effectively collects contextual topological data, which is crucial for link prediction. Link prediction fundamentally seeks to forecast possible relationships among diverse entities inside a network. To enhance the precision of this prediction, the research presents SEGNN4SLP, an innovative and unique GNN (Graph Neural Network) framework. This methodology uniquely integrates node properties with graph structural

data, providing more full insight of the network's complexities.

Extensive testing on real-world datasets has shown that incorporating structural encoding into the embedding learning process markedly improves API recommendation performance. The gathering of cooperative signals from both users and APIs enhances the accuracy and dependability of these recommendations.

Prospectively, numerous intriguing opportunities for future investigation exist. One route entails the integration of more comprehensive information regarding the node attributes of the API. Examining the particular labels linked to these nodes may yield further insights. Moreover, the advancement of more adaptive techniques for computing weight coefficients is a significant area of emphasis. Through the ongoing refinement and evolution of these methodologies, we anticipate increasingly precise and effective API recommendations in the future.

REFERENCES

[1] Ramadhanu P B, Priandika A T. Rancang Bangun Web Service Api Aplikasi Sentralisasi Produk Umkm Pada Uptd Plut Kumkm Provinsi Lampung. Jurnal Teknologi Dan Sistem Informasi, 2021, 2(1): 59-64.

[2] Cao B, Peng M, Xie Z, et al. PRKG: Pre-Training Representation and Knowledge-Graph-Enhanced Web Service Recommendation for Mashup Creation. IEEE Transactions on Network and Service Management, 2024.

[3] Wu S, Shen S, Xu X, et al. Popularity-aware and diverse web APIs recommendation based on correlation graph. IEEE Transactions on Computational Social Systems, 2022, 10(2): 771-782.

[4] Qi L, He Q, Chen F, et al. Data-driven web APIs recommendation for building web applications. IEEE transactions on big data, 2020, 8(3): 685-698.

[5] Li S, Niu D, Wang Y, et al. Hyper scale FPGA-as-a-service architecture for large-scale distributed graph neural network//Proceedings of the 49th Annual International Symposium on Computer Architecture. 2022: 946-961.

[6] Zhang M, Cui Z, Neumann M, et al. An end-to-end deep learning architecture for graph classification//Proceedings of the AAAI conference on artificial intelligence. 2018, 32(1).

[7] Wang Y Q, Dong L Y, Jiang X Q, et al. KG2Vec: A node2vec-based vectorization model for knowledge graph[J]. Plos one, 2021, 16(3): e0248552.