IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received June 18, 2020, accepted July 15, 2020, date of publication July 23, 2020, date of current version August 5, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3011439

GHSCN: A Graph Neural Network-Based API Popularity Prediction Method in Service Ecosystem

ZHONG LI[1,2], XIAOCHEN LIU[1,2], TIANBO WANG[1,3], (Member, IEEE), WENHUI HE[1,2], AND CHUNHE XIA[1,2,4]
[1]Beijing Key Laboratory of Network Technology, Beijing 100191, China
[2]School of Computer Science and Engineering, Beihang University, Beijing 100191, China
[3]School of Cyber Science and Technology, Beihang University, Beijing 100191, China
[4]Guangxi Key Laboratory of Multi-Source Information Mining and Security, School of Computer Science and Information Technology, Guangxi Normal University, Guilin 541004, China

Corresponding author: Chunhe Xia (xch@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant U1636208, Grant 61862008, and Grant 61902013; and in part by the Beihang Youth Top Talent Support Program under Grant YWF-20-BJ-J-1038.

ABSTRACT With the rapid development of technologies in the field of service computing, and increasing of complex business requirements, more and more large-scale service ecosystem emerges. Thus, many researches of service ecosystem focus on issues related to optimization such as service recommendation and load balancing, so the API popularity prediction problem studied in this paper, which is basis for this service ecosystem optimization, becomes a research hotspot in this field. However, many existing researches are predicting the popularity of APIs based on API functions, QoS, history usage patterns and social relationships, which are difficult to obtain and cannot reflect the overall structure of the underlying service ecosystem. Therefore, we propose an innovative API popularity prediction method in service ecosystem based on Graph Neural Network (GNN). Concretely, a Global-Service Ecosystem Network (GSEN) model is proposed firstly, for modeling a given service ecosystem to a network that can depict the complex structure of service ecosystem and the functions, QoS, history usage patterns and social relationships of APIs. Then, a Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) model is proposed to predict the popularity of APIs based on GSEN, and for getting better prediction accuracy, four different Heterogeneous Spatiotemporal Convolutional Kernels are proposed to extract the features of different elements which have different mechanisms to affect the popularity of target API. Finally, extensive experiments based on the data crawled from ProgrammableWeb.com show that our method achieves a superior performance in API popularity prediction, and the importance of the introduction of our model to service ecosystems.

INDEX TERMS API popularity prediction, graph neural network, GHSCN, service ecosystem.

I. INTRODUCTION

With the rapid development of service-oriented architecture, cloud computing and other technologies in the field of service computing, more and more services are published on the network by service developers, so that users of services can form new and more functional services by invoking these services, and these services form a complete service ecosystem. By invoking and combining the services in the service ecosystem, the more complex functional requirements can be met, which makes the whole service ecosystem more and more rich and robust. For example, the largest existing online Web services site, ProgrammableWeb, which records more than 20000 APIs with different functions. Developers can combine and reuse these APIs to form a service composition Mashup that meets the requirements of complex service functions, API and Mashup are the complete embodiment of service programmability. Since 2008, the number of APIs and Mashups are increasing rapidly, and in recent years, the growth trend has accelerated obviously. The rapid growth of the quantity and function of services as well as the increase of the complex relationship between services,

The associate editor coordinating the review of this manuscript and approving it for publication was Michael Lyu.

make it important and meaningful to ensure the security and stability of the whole service ecosystem and optimize the structure of the service ecosystem.

Using the API and Mashup data recorded on the ProgrammableWeb.com, the high-level service application developers would select the more popular APIs for combination according to the historical information, which means they have higher reliability and trust. Through historical data, we find that some APIs are called very frequently. For example, Google Map API is one of the most frequently called API by Mashups. This is because many Mashups need to call the API with Map function, and provider of the API is the world-famous Google company with high reliability. Due to these two factors, Google Map API is the most popular API to provide Map function in the whole service ecosystem. Therefore, it is very important for service developers to calculate and recommend the most popular API, developers can choose the more popular API for service composition so as to provide high reliability service composition. The existing API recommendation methods mainly focus on the service function behavior, QoS, past historical service pattern and social relationship, and the popularity of a service essentially indicates the above four aspects. Predicting the popularity of each service in the service ecosystem and its trend plays an important role in the study of the reliability of the whole service ecosystem. High popularity APIs have a great impact on the whole service ecosystem because they are called by a large number of Mashups, so we give safety measure to high popularity API ensuring its normal operation. At the same time, the popularity of API is not only related to the number of Mashups invoke it, but also related to the function as Category and the service providers, that is, related to the whole service ecosystem structure.

In order to predict the popularity of API in the service ecosystem, we find that it in the ProgrammableWeb has the following characteristics:

1. The popularity of API is related to the number of service combinations they participate in, that is, the more an API participates in, the higher its popularity is. For example, Facebook API participates in a larger number of Mashups than other APIs, so its popularity would remain relatively high in the future.
2. The popularity of API is also related to its competitiveness in functional category, that is, the competitiveness in the same category of API. For example, Paypal API has the highest number of service combinations under the Payments category, but it is not as high as Google Map API in terms of the number of Mashups it participates in. Paypal API is the most competitive API under the Payments category, so the popularity of Paypal would remain at a high level in the future.
3. The competitiveness of the API's provider also determines the popularity. For example, the provider of Google Map API is Google company, because of its own influence, it has produced a large number of high popularity APIs that are recognized by everyone. This shows that developers are willing to give priority to such a large competitive company when using services, so the API under Google Map would also have a high popularity in the future.

Therefore, in the service ecosystem, the relationship between different elements and the structure pose a higher challenge to the prediction of API popularity. It is not only a simple prediction process for APIs and Mashups based on function, QoS, historical service mode and social relationship, but also a prediction method based on the structure of the service ecosystem with analyzing the competition, dependence and complementarity relationship among the elements, these relationship contain the QoS, the reliability of historical service pattern, its trust degree in social relations, and the popularity of service.

While, the existing research of service popularity prediction and service recommendation mostly concentrate on the service function Tag and the obtained QoS data. The API recommended in research [1], [2] can best meet the needs of developers. Its main thought is the functional attributes provided by different APIs, while weakening the historical records of previous services. In reference [3], [4], a QoS prediction method based on machine learning is proposed, which predicts QoS value based on natural language processing method, so as to recommend potential high-quality service combination. Some researchers used machine learning method to learn the previous service composition patterns and usage patterns, so as to recommend the service composition that best meets the high-level requirements [5], [6], and considering based on social relations in the existing service recommendation methods to recommend the service composition that most developers trust [7], [8]. Some service providers predict the popularity of service content based on the service content they provide, such as the popularity of online video in the future [9]–[11], but we can't know the specific content of each service in the service ecosystem, and it's hard to get the specific function behavior, QoS, historical usage mode and social relationship of each API in real service scenario, so these methods are not suitable for us to predict the popularity of API in the service ecosystem. As far as we know, there is no popularity prediction method that can include four factors: API function, QoS, historical service pattern and social relationship. Therefore, we would depict the elements and structure of the service ecosystem based on these four factors with API, Mashup and its corresponding Category and Provider, and then give the popularity of each API based on the structure of the underlying service ecosystem.

However, due to the huge scale of each kind of element, and high complexity of the relationships between elements, in current service ecosystems, it is difficult to construct meaningful feature sets for predicting API popularity manually. Therefore, based on the idea of using deep learning models' advantage in extracting hidden features from massive sample data, this paper proposes a Graph Neural Network-based API popularity prediction model in service ecosystems, the so-called Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) model. Concretely, this paper first constructs a temporal heterogeneous network model of service ecosystems, the so-called Global Service Ecosystem Network (GSEN) model, according to the characteristic of service ecosystems that dynamic change with time and coexistence of different kinds of element; and then, based on the GSEN model, two GHSCN models containing four Heterogeneous Spatiotemporal Convolutional Kernels are designed to predict the popularity of API in global and specific categories respectively. Where, the four Heterogeneous Spatiotemporal Convolutional Kernels are the core of the GHSCN model. They extract the hidden features of elements that have different relationships with target API (the API of which the popularity needs to be predicted) through heterogeneous spatiotemporal convolutional operations, so as to improve the accuracy of target API's popularity prediction. Finally, to verify the effectiveness of the above models, a large number of experiments are carried out using the data crawled from ProgrammableWeb. And the results show that GHSCN model is extremely accurate in predicting API popularity, and can effectively improve fault monitoring efficiency in service ecosystems when guiding the deployment of fault monitoring agents through the predicted APIs' popularity.

In summary, the work and contributions of this paper mainly include the following four points:

1) A temporal heterogeneous network model of service ecosystems, the so-called Global-Service Ecosystem Network (GSEN) model, is presented. This model not only models the relationship between APIs and Mashups, but also depicts Categories, Providers and their relationship with APIs and Mashups, so that it has stronger expression ability, and can express the functions, QoS, historical service patterns and social relationships of services through nodes, edges and their attributes in constructed networks. Therefore, GSEN model will provide effective support for the research of performance, security and stability optimization of service ecosystems;
2) A Graph Neural Network-based API popularity prediction model in service ecosystems, the so-called Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) model, is proposed. This model is different from the existing models that only use the own attributes of APIs and Mashups to generate feature sets for target API's popularity prediction, the attributes of Categories and Providers, and the structural information of the underlying service ecosystem are also used to generate feature sets. Therefore, GHSCN model has better prediction accuracy of API popularity than existing models theoretically;
3) Four different Heterogeneous Spatiotemporal Convolutional Kernels are proposed to extract the features of different elements which have different mechanisms to affect the popularity of target API. In this paper, we divide the elements associated with the target API into four categories: supporting elements, required elements, competitive elements, and inherent elements. And then, we design different convolutional operations to deal these four categories of elements respectively, for improvement of GHSCN's learning efficiency and prediction accuracy;
4) It is verified that the overall security and stability of service ecosystem will be improved by deploying monitoring agents in the APIs with high predicted popularity, since this can improve the scope of monitoring and the accuracy of source identification.

The rest of this paper is organized as follow: Chapter II summary the related work of service ecosystem, service popularity and Graph Neural Networks (GNN); Chapter III gives an overview of the method proposed in this paper; Chapter IV detailly introduces the Graph Neural Network-based API prediction method, GHSCN method; Chapter V shows the accuracy and efficiency of GHSCN method, and illustrates the meaning of API popularity in improving service ecosystem's security and stability; Chapter VI concludes this paper and discusses the future work.

II. RELATED WORK

A. SERVICE OPTIMIZATION BASED ON APIS

In the field of service computing, many researches focus on service discovery, service recommendation, service selection, service composition and other issues, they selected existing service nodes on demand to find the optimal service composition to meet the needs of users. There are more and more researches on the public service data provided by the network as ProgrmmableWeb, which is the largest online API (i.e. service) repository at present. It collects more than 20000 APIs with various functions on the network. Because of API are reusable and programmable, developers can combine existing APIs to create a Mashup (service composition) meeting their needs, so as to realizing the comprehensive functional requirements and enhance the commercial value of the original API. The existing research use these data to study the service selection, recommendation, composition and other issues, such as the existing API selection, composition, forming a new Mashup, so as to accelerate the development of the service ecosystem.

With the emergence of API, more and more researchers turn their attention from WSDL to the functions of API itself and the relationship between them. References [12] and [13] proposed a framework for discovering APIs, which can effectively discover APIs that meet the functional requirements of developers. Reference [14] presented a view of Web API association data, which can help service developers to search association

API from multiple perspectives and combine it into Mashup to promote the rapid development of Web Mashup. References [15] And [16] gives a ranking method of API, which mainly determines the ranking of API according to the number of Mashups that call the API. A method of service recommendation based on user interest and social relationship of service is showed by [17]. Reference [18] build an ecosystem including API and Mashup is built based on ProgramableWeb data. Reference [19] gave a method to obtain the service evolution pattern by using the Latent Dirchlet Allocation (LDT) method and a prediction method based on time series, which is used to predict the invoking relationship between services. At the same time, an innovative Three-phase network prediction approach (NPA) is proposed in [20] for service recommendation. So, putting forward on API become more and more important and meaningful in service optimization field.

B. SERVICE SELECTING AND RECOMMENDING

When selecting and recommending services, it is very important to know the past service history data, trust degree and popularity of services. Predicting the popularity of service is helpful for researchers to select the most appropriate services to achieve better service composition. As far as we know, few studies predict the popularity of API through known service ecosystem structure to make service recommendation and service selection. The existing service selection methods for creating Mashup are mainly divided into four categories: (1) function-based methods. The APIs recommended by these methods can best meet the needs of developers, and its main focus is the functional attributes provided by different APIs while weakening the historical records of previous services [1], [2]. (2) collaborative filtering-based method, this method considers the past service composition patterns and the usage history of each API, it can provide the most consistent service composition with the historical service pattern [1], [5]. (3) QoS-based methods, most of these methods predict QoS values, so as to recommend potential high-quality service combinations [3], [4], [21]. (4) Social-based method, this method takes social factors other than services into account, such as the social relationship of developers, so as to give a more reliable service combination [6]–[8].

Most of the service selection and recommendation are based on the above methods, which are combined to give the optimal results. However, it is difficult for researchers to collect large-scale functional behavior data sets, QoS data sets and other related data sets of actual APIs in service ecosystem. At the same time, the behavior and QoS of APIs and the network status of users would change at any time when be invoked, so if only according to the behavior and QoS of services, service selection and recommendation results would be inaccurate. For example, an unknown service provider provides an API that performs a certain function with a high level of QoS over a period of time, but after a period of time, its QoS value drops significantly, this API should not be selected or recommended. So, the popularity of API is really important because it represents the function of API, the stability of QoS, the historical service mode and social relationship. So how to give a reasonable API popularity prediction method ensure that the popularity can provide the basis for accurate recommendation and selection of services is a very meaningful problem.

C. SERVICE POPULARITY PREDICTION

Now there are also some researches on the popularity prediction of service content, which are mostly used to provide social network services to improve their service quality. Reference [9] predicted the future content popularity by setting the content popularity at a specific time as a reference value and gave the linear correlation between them. Reference [10] presented a prediction model of video popularity based on Reservoir Computing. This model can predict the video popularity in a short time according to the video popularity of the previous days, but this method is easily affected by random effects. Reference [11] showed a linear model to predict the popularity of YouTube video, but not considering the impact of some social policies of YouTube itself on the popularity. Reference [22] predicted the popularity of online content using Cox proportional hazard expression model, which infers the possibility of content popularity through survival analysis model. But in real service scenario, we can't know the specific service content of API, these methods are not suitable for us to predict the popularity of API, so we need to give an API popularity prediction method for the service ecosystem, and the popularity should include four factors: function, QoS, historical service pattern and social relationship of APIs. These four factors are mapped with API, Mashup, and its corresponding Category and Provider to describe the structure of service ecosystem, and the popularity of each API is predicted based on the structure information.

D. GRAPH NEURAL NETWORKS (GNN)

In recent years, driven by the large number of graph data analysis and mining requirements in e-commerce, biomedicine, chemistry, citation networks and many other fields, the research of Graph Neural Network (GNN), proposed by Gori *et al.* in 2005 [23], has become a hotspot in the field of Artificial Intelligence (AI). And according to the difference of learning tasks, five kinds of GNN models have been designed in existing literature, namely, Graph Convolutional Network (GCN), Graph Attention Network (GAN), Graph Auto Encoder (GAE), Graph Generation Network (GGN) and Graph Spatial Temporal Network (GSTN).

Where, as a basic module of other four kinds of GNN models, Graph Convolutional Network (GCN) has the highest research heat, and it is mainly used to solve the problem that is caused by the irregularity of graph data (the number and order of nodes in different graphs are different), so that the traditional convolutional operations and pooling operations (Defined by tensor operations), which are applied to regular European space data processing such as images and videos, become invalid. To solve this problem, in the existing work, researchers have proposed a large number of generalization and redefinition schemes for convolutional operations and pooling operations. For example, in [24]–[27], based on the graph theory, the graph convolution operations are represented as the noise removal processes of graph signal processing, that is, the hidden features are extracted by the means of multiple graph Fourier transforms; [28]–[36] represents the graph convolution operation as the aggregation process of the attributes of target node and its neighbor nodes, that is, the hidden features are extracted by calculating the weighted average of the attributes of target node and its neighbor nodes for many times, through using the operations with neighbor node arrangement invariance as an aggregation function [29], or substituting some selected important neighbor nodes of target node into traditional convolution operations [31]–[33], etc.. In [24], [37], [38], the graph pooling operations are represented as the structure compression processes of graphs, that is, down sample the original graphs by selecting some important nodes in the original graph to recombine a new graph as the input of subsequent operations [24], [37], or using Multi-Layer Perceptron (MLP) to compress the dimensions of the original graph's connection matrix and characteristic matrix [38].

Compared with GCN, the research work of other four kinds of GNN models are relatively less. By introducing the attention mechanism, [39]–[42] proposed the Graph Attention Network (GAN) model to improve prediction accuracy, that is, give different importance to different neighbor nodes of the target node [39], [40], or provide instructions for selecting different models [41] and information propagation paths [42]. In order to use low-dimensional vector to accurately represent the nodes in graphs, [43]–[48] train the Graph Auto Encoder (GAE) models, by using Multi-Layer Perceptron (MLP) to encode and decode the adjacency matrix of graphs without attribute [43], [44] or directly encoding and decoding graphs attribute by GCN [45]–[48]. In [49]–[52], some Graph Generation Network (GGN) models are proposed, and for improving the training efficiency, [49], [50] divide the graph generation process into node and edge generation processes, [51], [52] transform the training process into a game process of graph generation model and discriminant model by using GAN models. To solve the problem of learning tasks in spatiotemporal graph data, [53]–[56] proposed the Graph Spatial Temporal Network (GSTN) models, which obtains the spatial correlation of data through GCNs, and uses the Convolutional Neural Network (CNN), Recurrent Neural Network (RNN) or Long-Short-Term Memory (LSTM) model to obtain the time correlation of data.

In conclusion, it can be seen that the advantage of GNN models in extracting hidden features from graph data makes GNN possible to change the status quo of graph data analysis and processing tasks that rely heavily on artificially constructed feature sets, thereby providing strong technical support for analysis and processing of larger and more complex graph data.

III. METHOD OVERVIEW

A. MOTIVATION

FIGURE 1 shows a complicated service ecosystem in the real world. Software product of Company A provides a variety of APPs. APP1 and APP2 are high-level App service applications which independently combined with existing API services developed by company A. Mashup1 is a composite service provided by Company A through invoking existing APPs for combination. A large number of companies are similar to Company A's service delivery mode, they provide various services to different users through the APP developed by themselves and the Mashup generated by invoking other APIs. These massive APPs, Mashups and their interaction with different APIs and API's providers constitute a complex service ecosystem.

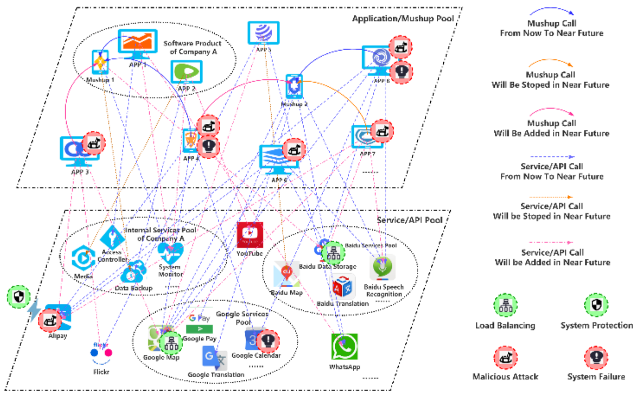


FIGURE 1. The research scenario of service ecosystem.

As FIGURE 1 presented, Google Map, Alipay and Baidu Data Storage are popular APIs in the entire API resource pool, which would be called by a large number of high-level Apps and Mashups, this means that these APIs have high service quality and provide excellent service reliability and stability. When high-level developers choose appropriate APIs for service composition, they will give priority to these high popularity APIs in order to obtain high reliability and better service quality for building APPs and Mashups. When high-level developers need to call an API to realize a certain function of their APP or Mashup, they prefer to select the most popular API among all APIs with such functions. For example, Google Map has a high popularity in the Map function, indicating that the service quality and service reliability of Google Map are leading under the same function API, so calling that API can ensure that APP and Mashup with the map function have high-level reliability. At the same time, the same service provider such as Google and Baidu, the popularity of the API they can provide is not the same, we can see that Google Map and Baidu Data Storage are the most popular APIs among the services provided by their corresponding service providers. This judgment and prediction of popularity can help us to deploy some load balancing measures against these APIs to balance the traffic borne and improve service quality.

When the high popularity API is attacked, the impact on the whole service ecosystem would be larger than that of the low popularity API. For example, if Alipay is maliciously attacked, the APP3, APP4, APP6, APP7, and APP8 which related with Alipay would have an impact, and all of these APPs can only be terminated due to the payment function be fail. This cascading impact would cause the users bad experience of these five Apps. If Google Calendar API fails, due to its low popularity, only APP4 and APP8 have relation with it, so its impact range is smaller than that of Alipay.

Therefore, system protection and defense measures should be priority given to Alipay with high popularity, which can improve the security and stability of the whole service ecosystem. As well as, because the high popularity API participates in more APPs and Mashups, setting monitoring nodes at the high popularity API is also conducive to finding the source of API faults in time and improving the accuracy of traceability when the APPs and Mashup failed.

For service ecosystem, it is very important to study the popularity of API for promoting the development of service ecosystem and ensuring its security and stability. The in-category popularity of API represents the popularity of the API within APIs that have the same functions. The higher in-category popularity indicates that the API has higher reliability and better service quality in the APIs which have same functions, which also means that more and more APPs and Mashups would invoke it preferentially when they need such functions. The global popularity of API represents the popularity of API in the whole service ecosystem. The high global popularity indicates that API plays an important role in the whole service ecosystem, and its service provider and service quality are widely recognized. If the in-category popularity and global popularity of API are accurately predicted, service optimization and corresponding protection measures can be deployed for high popularity APIs to improve the security and stability of the whole cloud service ecosystem.

B. PROBLEM STATEMENT

1) DEFINITION OF API POPULARITY IN SERVICE ECOSYSTEMS

As we know, in many service ecosystems, most of APIs provide multiple functions, and when Mashups choose APIs, they often compare the APIs that provide the same categories of functions. Therefore, the popularity of APIs in service ecosystems is divided into in-category popularity and global popularity, which are respectively defined as follows:

Definition 1: the in-category popularity $p_{f,s}^{IC}(t)$ of a API s in a specific functional category f, means that at a specific moment t, the proportion of Mashups, which call function f provided by service s, in the all Mashups which call function f, i.e.:

$$p_{f,s}^{IC}(t)=\frac{n_{f,s}^{M}(t)}{n_{f}^{M}(t)}$$

where, $n_{f,s}^{M}(t)$ represents the number of Mashups which call function f provided by service s at moment t, $n_{f}^{M}(t)$ represents the number of the all Mashups that call function f at moment t.

For example, there were 162 Mashups calling the Financial function, and of which there were 35 Mashups calling the Financial function provided by PayPal API, in the data recorded on the ProgrammableWeb in January 2019. Therefore, in January 2019, the in-category popularity of PayPal API within the Financial functional category is $^{35}/_{162}= 0.216$.

Definition 2: the global popularity $p_{s}^{G}(t)$ of a API s means that at a specific moment t, the proportion of Mashups, which call API s, in the all Mashups, i.e.:

$$p_{s}^{G}(t)=\frac{n_{s}^{M}(t)}{n^{M}(t)}$$

where, $n_{s}^{M}(t)$ represents the number of Mashups which call API s at time t, and $n^{M}(t)$ represents the number of all Mashups at time t.

Similarly, taking the data of ProgrammableWeb as an example, in January 2019, there were 6408 Mashups in all, and of which there were 35 Mashups calling the PayPal API. Therefore, in January 2019, the global popularity of PayPal API was $^{35}/_{6408}= 0.00546$.

Note that the definition of the popularity of other kinds of elements, such as Providers and Categories, in service ecosystems is the same as the APIs'.

2) API POPULARITY PROBLEM IN SERVICE ECOSYSTEMS

As with most of time sequence prediction problems, in this paper, the goal of API popularity prediction problem in service ecosystems is to predict the popularity of each API at a future moment, based on the data collected in the underlying service ecosystem in a previous period. Thus, if using $P_{SE}(t)=\bigcup_{s\in S^S}\left[p_s^G(t)\cup\left(\bigcup_{f\in S^F}p_{f,s}^{IC}(t)\right)\right]\in R^{(n^F(t)+1)\times n^S(t)}$ to represent the vector formed by concatenating the popularity of all APIs in the underlying service ecosystem at moment t (where, there are $n^F(t)\times n^S(t)$ in-category popularity $p_{f,s}^{IC}(t)$ and $n^S(t)$ global popularity $p_s^G(t)$, and $n^F(t)$ and $n^S(t)$ respectively represent the number of Categories and APIs in the underlying service ecosystem at moment t, S^F and S^S respectively represent the set of Categories and APIs, $\cup$ represents the concatenation operation of vectors here), using $D_{SE}(t)\in R^{n^D(t)}$ to represent the vector formed by concatenating the data collected in the underlying service ecosystem at moment t (where $n^D(t)$ represents the number of data collected at moment t, and $n^D(t)\geq\left(n^F(t)+1\right)\times n^S(t)$), and using $D_{SE}[t_1{:}t_2]=[D_{SE}(t_1),D_{SE}(t_1+1),\cdots,D_{SE}(t_2)]$ to represent the vector formed by concatenating the data collected in the underlying service ecosystem from time t_1 to time t_2. Then, the goal of API popularity prediction problem in service ecosystems is to learn a function $h(\cdot)$, so that the function can map the data $D_{SE}[t-T{:}t-1]$ collected in the T moments before any moment t to the popularity $P_{SE}(t)$ of all APIs at the moment t, i.e.:

$$h:\ D_{SE}[t-T:t-1]\rightarrow P_{SE}(t)$$

C. APPROACH ROADMAP

Aiming at the problems studied in this paper, as with the mainstream research work on sequence prediction problems, we propose a solution based on machine learning method. However, due to the characteristic of service ecosystems that dynamic change with time, coexistence of different kinds of element, and the high complexity of relationships between elements, traditional deep learning models dealing with European space learning problems, like CNN and RNN, not suitable for predicting the popularity of APIs in service ecosystems. There are two reasons for this: (1) with the evolution of service ecosystem, the dimension $n^D(t)$ of collected data $D_{SE}(t)$ will change with time t. Therefore, when using $D_{SE}[t-T{:}t-1] = [D_{SE}\text{(t-T)}, \cdots, D_{SE}(t-2), D_{SE}(t-1)]$ as the input of CNN, RNN or other traditional deep learning models, we need to fill in a large number of invalid data, or cut off a large number of data in $D_{SE}[t-T{:}t-1]$ to make the input dimension of each sample the same. (2) using the Euclidean space vector $D_{SE}(t)$ to represent the data collected from service ecosystems cannot describe the complex relationships between different kinds of elements. Thus, traditional deep learning models such as CNN and RNN are prone to overfitting during the training process, and it will take more time, resources, and samples to achieve good learning results.

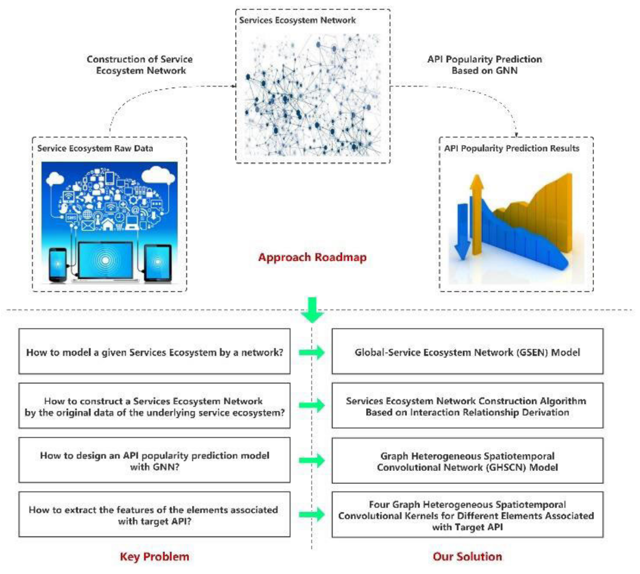

Services Ecosystem Network
Construction of Service Ecosystem Network
API Popularity Prediction Based on GNN
Service Ecosystem Raw Data
API Popularity Prediction Results
Approach Roadmap
How to model a given Services Ecosystem by a network?
Global-Service Ecosystem Network (GSEN) Model
How to construct a Services Ecosystem Network by the original data of the underlying service ecosystem?
Services Ecosystem Network Construction Algorithm Based on Interaction Relationship Derivation
How to design an API popularity prediction model with GNN?
Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) Model
How to extract the features of the elements associated with target API?
Four Graph Heterogeneous Spatiotemporal Convolutional Kernels for Different Elements Associated with Target API
Key Problem
Our Solution

FIGURE 2. Approach roadmap of this paper's research.

To overcome the above two problems, this paper proposes an approach roadmap as shown in the top half of FIGURE 2. First of all, based on the original data $D_{SE}[t-T:t-1] = [D_{SE}(t-T), \cdots, D_{SE}(t-2), D_{SE}(t-1)]$ collected from the underlying service ecosystem, a sequential service ecosystem network $G_{SE}[t-T:t-1] = [G_{SE}(t-T), \cdots, G_{SE}(t-2), G_{SE}(t-1)]$ is constructed, so that the network can depict the relationships between key elements in the underlying service ecosystem at each moment of $[t-T:t-1]$. And then, taking the network $G_{SE}[t-T:t-1]$ as the input, using GNN model which is good at dealing with structured data learning problem to predict the API popularity $P_{SE}(t)$ at next moment t. Thus, the solution proposed in this paper converts the objective function $h(\cdot)$ in Section B of Chapter III into the following two-step mapping:

$$h: D_{SE}[t-T:t-1] \to G_{SE}[t-T:t-1] \to P_{SE}(t)$$

The benefits of the above solution are: (1) the relationships between key elements in the underlying service ecosystem can be reflected in the input of GNN model, so that the learning efficiency of GNN model can improve effectively; (2) since after being converted into a sequential service ecosystem network $G_{SE}[t-T:t-1]$, the data in $D_{SE}[t-T:t-1]$ is scattered in the attributes of each node and each edge in $G_{SE}[t-T:t-1]$, so that when input to the GNN model in batches, there is no need to fill in invalid data or cut off existing data in large scale.

However, as shown in the bottom half of FIGURE 2, there are four key problems need to be solved in designing the API popularity prediction method in service ecosystems along this route: (1) how to model a given service ecosystem as a network? (2) how to construct a service ecosystem network by the original data collected from the underlying service ecosystem? (3) how to design an API popularity prediction model based on GNN? (4) how to extract the features of the elements associated with the target API? To solve these problems, this paper gives four corresponding technologies: Global-Service Ecosystem Network (GSEN) model of service ecosystems, service ecosystem network construction algorithm based on interactive relationship derivation, Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) model for API popularity prediction in service ecosystems, Graph Heterogeneous Spatiotemporal Convolutional Kernels for extracting the features of different elements which have different mechanisms to affect the popularity of target API. Next, we introduce these technologies in detail.

IV. GHSCN METHOD

This chapter introduces the Graph Neural Network-based API popularity prediction method in service ecosystems proposed in this paper. The Global-Service Ecosystem Network model part introduces the service ecosystem network model proposed in this paper, and corresponding algorithm for constructing service ecosystem networks based on the original data collected from service ecosystems; the Graph Neural Network-based API popularity prediction method part introduces the framework of the API popularity prediction method proposed in this paper, and the graph heterogeneous spatiotemporal convolutional operations designed for the characteristics of service ecosystem network.

A. GLOBAL-SERVICE ECOSYSTEM NETWORK MODEL

Ecosystem is a word derived from the field of ecology, which is defined as all animals and plants in a specific area,

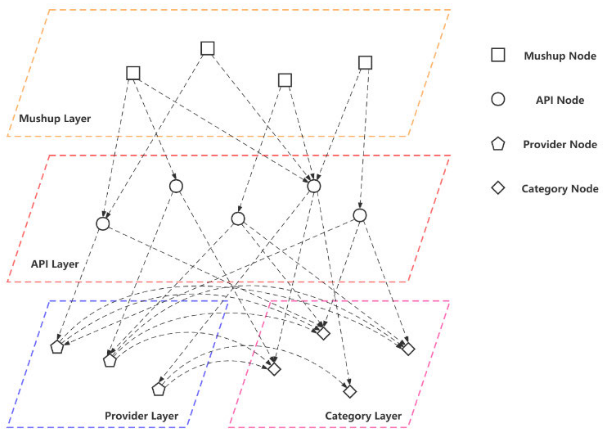


FIGURE 3. **Service ecosystem network model.**

their relationship with each other and their contact with the environment (Longman Dictionary of Contemporary English, 2008). In the ecosystem, animals, plants and environment are related with each other. Their own activities and their interactions lead to the formation, maintenance, development, deterioration and disappearance of the whole ecosystem. In this way, the change of elements and relationships in the whole ecosystem will promote the evolution of the ecosystem. Ecosystem consists of two important components: one is the elements in the system, which can have different characteristics and attributes; the other is the functional relationship between these elements, through which different elements are connected together, a complex network with self-organization characteristics and continuous evolution characteristics can be formed. Reference [57] introduced the concept of ecosystem into modern commerce for the first time, proposed the business ecosystem and its life cycle, and gave the definition of business ecosystem with leading the development of business research. In recent years, service ecosystem has emerged with the attention of researchers in the field of service computing. The earliest definition of service ecosystem is from the perspective of supply based on customer centered, which evolved with the development of other dynamic factors related to customers and their interaction relationship [58]. However, there is no commonly used definition of service ecosystem in the research field until now. G. Scheithauer [59] believes that the service ecosystem is the result of the transformation from the traditional electronic market to the service economy. The goal of the service ecosystem is to make different services interact with each other through the Internet, so as to make services having more commercial value. In this paper, we regard the service ecosystem is influenced by API, Mashup, Provider, Category and their interaction. These basic elements interact with each other through cooperation, competition, invocation, etc. Thus, affecting the direction of the whole service ecosystem development, and making the service ecosystem more complicated.

In order to better depict the service ecosystem, we abstract the service ecosystem into a complex network structure. The elements of the service ecosystem mainly include API, Mashup and Provider. Provider can provide multiple APIs, and Mashup can realize its functions by invoking multiple APIs. Generally, the main indicators used to describe and distinguish between API and Mashup are function and service quality. APIs with different functions can cooperate with each other to form a Mashup with complex functions. The service ecosystem will evolve and develop when users put forward their own requirement, the popularity and reliability of API play a key role in the stability and security of the service ecosystem. Therefore, predicting the popularity of API in the service ecosystem is of great significance to users and the evolution of the service ecosystem.

Definition 3: (Service Ecosystem Network) The service ecosystem network is formalized into $SEnet(t) =< SE(t), R(t)^{SE} >$. where $SE(t) = \{API(t), Mashup(t), Provider(t), Function(t)\}$ refers to the elements of time t in the service ecosystem, and $R^{SE}(t) = \{(se^1(t), se^2(t)) | se^1(t) \in SE(t), se^2(t) \in SE(t)\}$ refers to the interaction of different elements in the service ecosystem at time t.

1) DERIVATION OF INTERACTION RELATIONSHIP

Because the existing service recommendation is mainly based on QoS, function, social relationship and historical service pattern, we hope to build the service ecosystem can contain the above four factors. The following gives the derivation method of $R^{SE}(t)$ relationship at time t, which describes the derivation of the relationship between the elements of cloud service ecosystem at time t.

In order to meet the QoS, function and social relationship in service ecosystem, the concretely derivation process is as follows:

- $SE_{out}(t)$ is the target service element, i.e. the API, Mashup, Provider, Category connected with the existing service element at time t;
- $SE_{in}(t)$ is the existing service element, i.e. API, Mashup, Provider, Category known at time t.
- $S_n(t), S'_n(t)$ is a subset of $SE_{out}(t)$ or $SE_{in}(t)$;
- $e_n(t), e'_n(t)$ is the data element related to API, Mashup, Provider, Category in service ecosystem;
- $D_n(t), D'_n(t)$ is the data element related to API, Mashup, Provider, Category in the service ecosystem, including input data entity and output data entity. These data include QoS, social relations, and historical service patterns. Each data entity includes a collection of data elements $D_n(t) = \left\{e_n^1(t), e_n^2(t), \ldots\right\}$;
- $f(t)$ is the conversion process from $D_n(t)$ to $BD'_n(t)$.
- $S_n(t) \rightarrow D_n(t)$indicates the output relationship of data, i.e., the output data of S_n at time t is D_n.
- $D_n(t) \rightarrow D_n(t)$, indicates the relationship between data, that is, at time t, D_n is a subset of D'_n.
- $D_n(t) \rightarrow S_n(t)$, it indicates the input relationship of data, that is, at time t, $S_n$needsdataofD_n as input.

Based on the above definition, the triple $(R_o^{SE}(t), R_L^{SE}(t), R_I^{SE}(t))$ of a derivation process is given to decide whether API, Mashup, Provider, Category are connected or not, i.e.:

$$R_o^{SE}(t) = \{(S_n(t),\ D_n(t)) | S_n(t) \to D_n(t)\}$$
$$R_L^{SE}(t) = \{(D_n(t),\ D_n{}'(t)) | D_n(t) \to D_n{}'(t)\}$$
$$R_I^{SE}(t) = \{(D_n{}'(t),\ S_n{}'(t)) | D_n{}'(t) \to S_n{}'(t)\}$$

2) BUILDING SERVICE ECOSYSTEM NETWORK

Given a set of service ecosystem elements SE_n, building a service ecosystem network is to build a network $G =< V, E >$ with a series of SE_n by the following algorithm:

Algorithm 1 Service Ecosystem Network Construction

```
Input: $S_{API}, S_{Function}, S_{Provider}, S_{Mushup}$
Output: $G_{SE}$
Procedure: SvcEcsConstruction
$G_{SE} = \emptyset$;
For $t \in \{0, 1, 2, \cdots, T_{SE}^{Max}\}$ Do
$V_{SE}(t) = S_{API}(t) \cup S_{Function}(t) \cup S_{Provider}(t) \cup S_{Mushup}(t)$;
$E_{SE}(t) = \emptyset$;
For $s_{API}(t) \in S_{API}(t), s_{Function}(t) \in S_{Function}(t)$ Do
$R_{s_{API}(t), s_{Function}(t)}$ = ItaRlsDerivation($s_{API}(t), s_{Function}(t)$);
If $R_{s_{API}(t), s_{Function}(t)} \neq NR$ Then
$E_{SE}(t) = E_{SE}(t) \cup \{R_{s_{API}(t), s_{Function}(t)}\}$;
End If
End For
For $s_{Provider}(t) \in S_{Provider}(t), s_{Function}(t) \in S_{Function}(t)$ Do
$R_{s_{Provider}(t), s_{Function}(t)}$ = ItaRlsDerivation($s_{Provider}(t)$,
$s_{Function}(t)$);
If $R_{s_{Provider}(t), s_{Function}(t)} \neq NR$ Then
$E_{SE}(t) = E_{SE}(t) \cup \{R_{s_{Provider}(t), s_{Function}(t)}\}$;
End If
End For
For $s_{API}(t) \in S_{API}(t), s_{Provider}(t) \in S_{Provider}(t)$ Do
$R_{s_{API}(t), s_{Provider}(t)}$ = ItaRlsDerivation($s_{API}(t), s_{Provider}(t)$);
If $R_{s_{API}(t), s_{Provider}(t)} \neq NR$ Then
$E_{SE}(t) = E_{SE}(t) \cup \{R_{s_{API}(t), s_{Provider}(t)}\}$;
End If
End For
For $s_{Mushup}(t) \in S_{Mushup}(t), s_{API}(t) \in S_{API}(t)$ Do
$R_{s_{Mushup}(t), s_{API}(t)}$ = ItaRlsDerivation($s_{Mushup}(t), s_{API}(t)$);
If $R_{s_{Mushup}(t), s_{API}(t)} \neq NR$ Then
$E_{SE}(t) = E_{SE}(t) \cup \{R_{s_{Mushup}(t), s_{API}(t)}\}$;
End If
End For
$G_{SE}(t) = \langle V_{SE}(t), E_{SE}(t) \rangle$;
$G_{SE} = G_{SE} \cup \{G_{SE}(t)\}$;
End For
Return $G_{SE}$;
End Procedure
```

B. GRAPH NEURAL NETWORK-BASED API POPULARITY PREDICTION METHOD

1) GRAPH HETEROGENEOUS SPATIOTEMPORAL CONVOLUTIONAL NETWORK MODEL

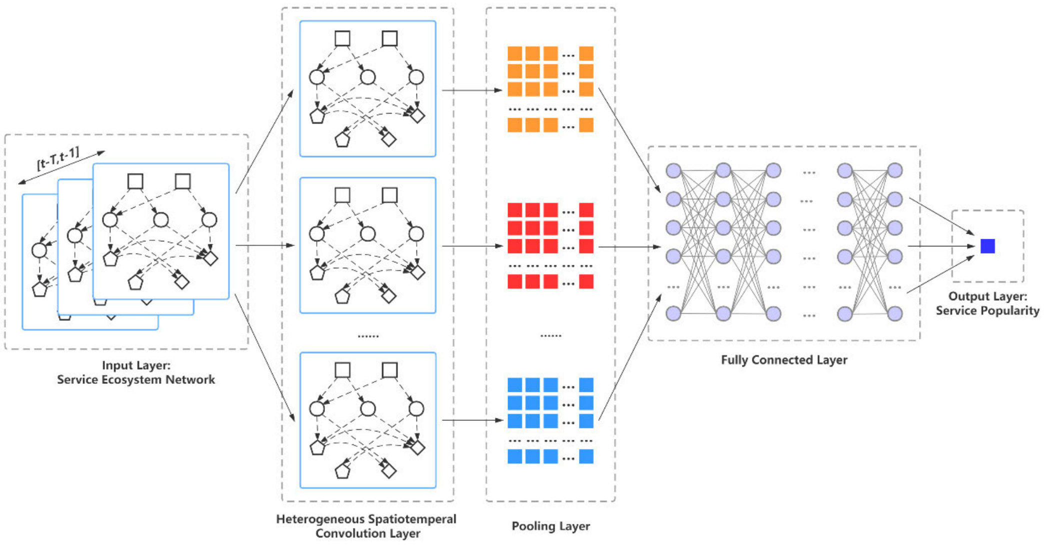
As described in Section C of Chapter III, after converting the original data $D_{SE}[T_{Start} : T_{End}]$ collected from the underlying service ecosystem into sequential network $G_{SE}[T_{Start} : T_{End}]$ through the model and algorithm given in Section A of Chapter IV, this paper predict the popularity of APIs based on GNN model. Concretely, we first decompose the prediction process of overall popularity $P_{SE}(t)$ of the underlying service ecosystem into the prediction process of in-category popularity $p_{f,s}^{IC}(t)$ and global popularity $p_s^G(t)$, i.e., the second step mapping $G_{SE}[t - T : t - 1] \to P_{SE}(t)$ of objective function $h(\cdot)$ in Section C of Chapter III is decomposed into the following mapping set:

$$\left\{s \in S^S \mid G_{SE}[t - T : t - 1] \to p_s^G(t)\right\} \cup \left\{s \in S^S, f \in S^F \mid G_{SE}[t - T : t - 1] \to p_{f,s}^{IC}(t)\right\}$$

There are two reasons for this composition: (1) by dividing the attribute prediction of the whole network (corresponding to the prediction of $P_{SE}(t)$) into the attribute prediction of single node (corresponding to API global popularity prediction $p_s^G(t)$) and single edge (corresponding to API in-category popularity prediction $p_{f,s}^{IC}(t)$), each iteration in model training process no longer needs to input the whole service ecosystem network in a period of time (i.e. $G_{SE}[t - T : t - 1]$), only needs to input the subgraph sequences strongly related to the target API and Category, so that the cost of time and resources can be effectively reduced; (2) in many application scenarios, such as service recommendation and internal resource optimization of Providers, it is only necessary to predict the popularity of some or a single API in the underlying service ecosystem, so the decomposed model has a larger scope of application and higher flexibility.

Furthermore, for the prediction of in-category popularity $p_{f,s}^{IC}(t)$ and global popularity $p_s^G(t)$ of a single API s, we propose a Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) model with three core layers as shown in FIGURE 4. Where, the heterogeneous spatiotemporal convolution layer is the core part of GHSCN model, and it use the Graph Heterogeneous Spatiotemporal Convolutional Kernels designed in this paper to extract the features of different elements which have different mechanisms to affect the popularity of target API; the pooling layer combines the output of the convolutional layer into a complete fixed-dimensional vector; the fully connected layer finally fits the popularity of the target API based on the vector output from the pooling layer.

It is worth noting that in the later sections, for convenience of expression, we use GHSCN-GP and GHSCN-ICP to represent the Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) models for API global popularity



[t-T,t-1]
Input Layer: Service Ecosystem Network
Heterogeneous Spatiotemporal Convolution Layer
Pooling Layer
Fully Connected Layer
Output Layer: Service Popularity

FIGURE 4. The Graph Heterogeneous Spatiotemporal Convolutional Network (GHSCN) Model.

prediction and in-category popularity prediction respectively (the difference between this two models is reflected in the convolutional operations described below).

2) GRAPH HETEROGENEOUS SPATIOTEMPORAL CONVOLUTIONAL KERNELS

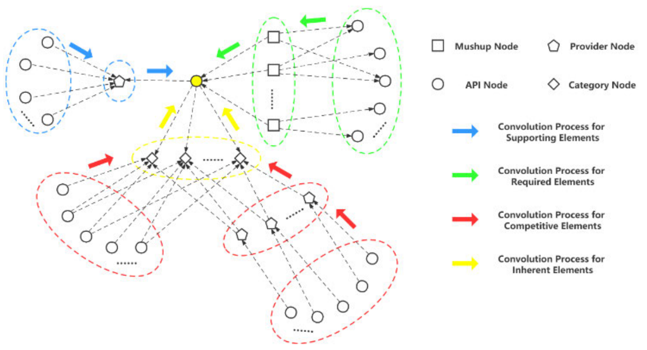
In order to improve the learning efficiency of the GHSCN model designed in this paper, we need to process the underlying service ecosystem's elements, which associate with target API, in the graph convolutional operations in batches, according to their different mechanisms to affect the popularity of target API. Thus, first of all, based on the Calling Relationship (between APIs and Mashups), Subordination Relationship (between APIs and Providers), Providing Relationship (between APIs and Categories) and other native relationships of elements in service ecosystems, this paper defines the following four types of higher-level relationships:

Collaborative Relationship: refers to the relationship between the APIs called by the same Mashups. For example, the relationships between target API (marked in yellow in the figures) and the APIs circled by the green dotted line on the right of FIGURE 5 and FIGURE 6.

Joint Relationship: refers to the relationship between the APIs provided by the same Providers. For example, the relationship between target API and the APIs circled by the blue dotted line on the left of FIGURE 5 and FIGURE 6.

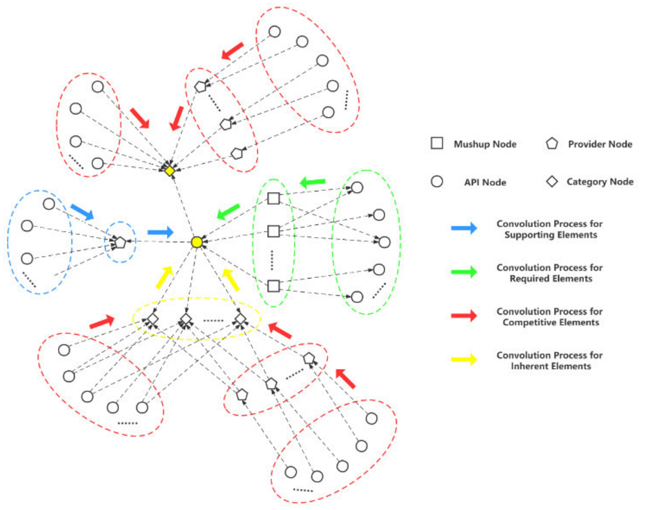
Direct Competitive Relationship: refers to the relationship between APIs where function sets intersect. For example, the relationship between target API and the APIs circled by the red dotted line in the lower left corner of FIGURE 5 and FIGURE 6.

Indirect Competitive Relationship: refers to the relationship between the APIs, which have a Direct Competitive Relationship with a API s, and the APIs that have a Joint



Mushup Node
Provider Node
API Node
Category Node
Convolution Process for Supporting Elements
Convolution Process for Required Elements
Convolution Process for Competitive Elements
Convolution Process for Inherent Elements

FIGURE 5. The Sketch Map of Graph Convolutional Process for API Global Popularity Prediction.



Mushup Node
Provider Node
API Node
Category Node
Convolution Process for Supporting Elements
Convolution Process for Required Elements
Convolution Process for Competitive Elements
Convolution Process for Inherent Elements

FIGURE 6. The Sketch Map of Graph Convolutional Process for API In-category Popularity Prediction.

Relationship with s; and the relationship between the APIs, which have a Direct Competitive Relationship with a API s, and the Provider of s. For example, the relationship between target API and the APIs and Providers circled by the red dotted line in the lower right corner of FIGURE 5 and FIGURE 6.

Further, as shown in FIGURE 5 and FIGURE 6, based on the above four types and original relationships of elements in service ecosystems, this paper divides the elements associated with the target API into four categories, and designs corresponding graph convolution operations to extract their features. Concretely:

1) The target API's Provider provides resources for the target API, and the APIs, which have a Joint Relationship with target API, form functional complement with target API. They are all support for increasing the popularity of target API. Thus, they and the native relationships between them and between them and target API are defined as the first category of elements, Supporting Elements. Just as the elements convolved along the blue arrows in FIGURE 5 and FIGURE 6.
2) The Mashups, which have a Calling Relationship with target API, and the APIs, which have a Collaborative Relationship with target API, reflect the demand level and characteristics of the entire service ecosystem for target API. Thus, they and the native relationships between them and between them and target API are defined as the second category of elements, Required Elements. Just as the elements convolved along the green arrows in FIGURE 5 and FIGURE 6.
3) The APIs, which have a Direct Competitive Relationship with target API, compete with target API for Mashups with the same requirements, and the APIs and Providers, which have an Indirect Competitive Relationship with target API, provide support for the competitors of target API. They all have an inhibitory effect on the popularity of target API. Thus, they and the native relationships between them and between them and target API are defined as the third category of elements, Competitive Elements. Just as the elements convolved along the red arrows in FIGURE 5 and FIGURE 6.
4) The functional Categories provided by target API are the same as the properties of the target API itself, reflecting the inherent characteristics and inner competitiveness of the target API. Thus, they and the native relationships between them and between them and target API are defined as the fourth category of elements, Inherent Elements. Just as the elements convolved along the yellow arrows in FIGURE 5 and FIGURE 6.

Then, based on the above classification results, we design two sets of Graph Heterogeneous Spatiotemporal Convolutional Kernels to extract the features of related elements for the prediction API global popularity and in-category popularity in service ecosystems. The details are as follows:

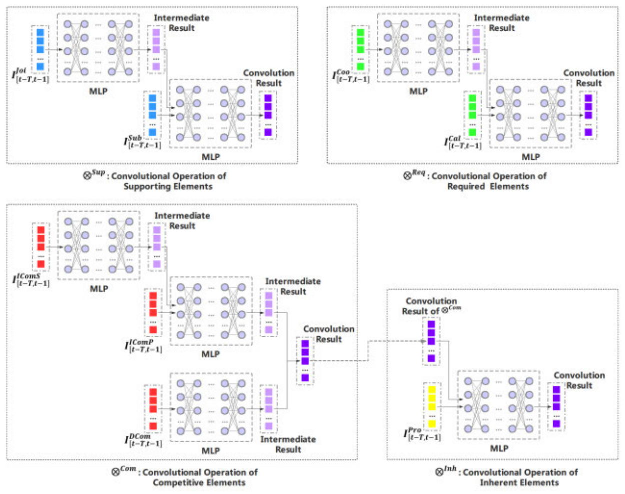

Intermediate Result
Convolution Result
MLP
$I^{Joi}_{[t-T,t-1]}$
$I^{Sub}_{[t-T,t-1]}$
$\otimes^{Sup}$: Convolutional Operation of Supporting Elements
$I^{Coo}_{[t-T,t-1]}$
$I^{Cal}_{[t-T,t-1]}$
$\otimes^{Req}$: Convolutional Operation of Required Elements
$I^{ComS}_{[t-T,t-1]}$
$I^{ComP}_{[t-T,t-1]}$
$I^{DCom}_{[t-T,t-1]}$
$\otimes^{Com}$: Convolutional Operation of Competitive Elements
Convolution Result of $\otimes^{Com}$
$I^{Pro}_{[t-T,t-1]}$
$\otimes^{Inh}$: Convolutional Operation of Inherent Elements

FIGURE 7. **The model of the four graph heterogeneous spatiotemporal convolutional kernels proposed in this paper.**

a: GRAPH CONVOLUTIONAL KERNELS FOR API GLOBAL POPULARITY PREDICTION

As shown in the sketch map of FIGURE 5 and the convolutional operation model of FIGURE 7, this paper uses the following four groups of graph heterogeneous spatiotemporal convolutional operations $\bigotimes^{Sup}$, $\bigotimes^{Req}$, $\bigotimes^{Com}$ and $\bigotimes^{Inh}$, which are constructed by Multi-Layer Perceptron (MLP), to respectively extract the characteristics of Supporting Elements, Required Elements, Competitive Elements and Inherent Elements associated with the target API s^* that need to predict global popularity:

$$\begin{aligned}
\bigotimes^{Sup} &: I^{Joi}_{[t-T,t-1]} \rightarrow_{MLP} R^{I-Sup} \\
&\quad R^{I-Sup}, I^{Sub}_{[t-T,t-1]} \rightarrow_{MLP} R^{C-Sup} \\
\bigotimes^{Req} &: I^{Coo}_{[t-T,t-1]} \rightarrow_{MLP} R^{I-Req} \\
&\quad R^{I-Req}, I^{Cal}_{[t-T,t-1]} \rightarrow_{MLP} R^{C-Req} \\
\bigotimes^{Com} &: I^{IComS}_{[t-T,t-1]} \rightarrow_{MLP} R^{I-Com}_1 \\
&\quad R^{I-Com}_1, I^{IComP}_{[t-T,t-1]} \rightarrow_{MLP} R^{I-Com}_2 \\
&\quad I^{DCom}_{[t-T,t-1]} \rightarrow_{MLP} R^{I-Com}_3 \\
&\quad R^{I-Com}_2, R^{I-Com}_3 \rightarrow R^{C-Com} \\
\bigotimes^{Inh} &: R^{C-Com}, I^{Pro}_{[t-T,t-1]} \rightarrow_{MLP} R^{C-Inh}
\end{aligned}$$

where, R^{I-Sup}, R^{I-Req}, R^{I-Com}_1, R^{I-Com}_2 and R^{I-Com}_3 respectively represent the intermediate results of each convolutional operation, R^{C-Sup}, R^{C-Req}, R^{C-Com} and R^{C-Inh} respectively represent the final results of each convolutional operation, $I^{Joi}_{[t-T,t-1]}$, $I^{Sub}_{[t-T,t-1]}$, $I^{Coo}_{[t-T,t-1]}$, $I^{Cal}_{[t-T,t-1]}$, $I^{IComS}_{[t-T,t-1]}$, $I^{IComP}_{[t-T,t-1]}$, $I^{DCom}_{[t-T,t-1]}$ and $I^{Pro}_{[t-T,t-1]}$ respectively represent the input vectors of each convolutional operation composed of the attributes of the four categories of elements associated with target API s^*, and are respectively defined as follows:

$$I^{Joi}_{[t-T,t-1]} = \begin{pmatrix} a^{Joi}_{s*,t-T,1} & \cdots & a^{Joi}_{s*,t-1,1} \\ \vdots & \ddots & \vdots \\ a^{Joi}_{s*,t-T,N^{Joi}} & \cdots & a^{Joi}_{s*,t-1,N^{Joi}} \end{pmatrix}$$

$$I^{Sub}_{[t-T,t-1]} = \begin{pmatrix} a^{Sub}_{s*,t-T} & \cdots & a^{Sub}_{s*,t-1} \end{pmatrix}$$

$$I^{Coo}_{[t-T,t-1]} = \begin{pmatrix} a^{Coo}_{s*,t-T,1} & \cdots & a^{Coo}_{s*,t-1,1} \\ \vdots & \ddots & \vdots \\ a^{Coo}_{s*,t-T,N^{Coo}} & \cdots & a^{Coo}_{s*,t-1,N^{Coo}} \end{pmatrix}$$

$$I^{Cal}_{[t-T,t-1]} = \begin{pmatrix} a^{Cal}_{s*,t-T,1} & \cdots & a^{Cal}_{s*,t-1,1} \\ \vdots & \ddots & \vdots \\ a^{Cal}_{s*,t-T,N^{Cal}} & \cdots & a^{Cal}_{s*,t-1,N^{Cal}} \end{pmatrix}$$

$$I^{IComS}_{[t-T,t-1]} = \begin{pmatrix} a^{IComS}_{s*,t-T,1} & \cdots & a^{IComS}_{s*,t-1,1} \\ \vdots & \ddots & \vdots \\ a^{IComS}_{s*,t-T,N^{IComS}} & \cdots & a^{IComS}_{s*,t-1,N^{IComS}} \end{pmatrix}$$

$$I^{IComP}_{[t-T,t-1]} = \begin{pmatrix} a^{IComP}_{s*,t-T,1} & \cdots & a^{IComP}_{s*,t-1,1} \\ \vdots & \ddots & \vdots \\ a^{IComP}_{s*,t-T,N^{IComP}} & \cdots & a^{IComP}_{s*,t-1,N^{IComP}} \end{pmatrix}$$

$$I^{DCom}_{[t-T,t-1]} = \begin{pmatrix} a^{DCom}_{s*,t-T,1} & \cdots & a^{DCom}_{s*,t-1,1} \\ \vdots & \ddots & \vdots \\ a^{DCom}_{s*,t-T,N^{DCom}} & \cdots & a^{DCom}_{s*,t-1,N^{DCom}} \end{pmatrix}$$

$$I^{Pro}_{[t-T,t-1]} = \begin{pmatrix} a^{Pro}_{s*,t-T,1} & \cdots & a^{Pro}_{s*,t-1,1} \\ \vdots & \ddots & \vdots \\ a^{Pro}_{s*,t-T,N^{Pro}} & \cdots & a^{Pro}_{s*,t-1,N^{Pro}} \end{pmatrix}$$

where, t represents the moment that need to predict the global popularity of target API $s*$, T represents the length of the time period of the historical data used for the prediction task, N^{Joi} represents the number of APIs that have a Joint Relationship with target API and are selected for using their attributes as the input of graph convolution operations, N^{Coo} represents the number of APIs that have a Collaborative Relationship with target API and are selected for using their attributes as the input of graph convolution operations, N^{Cal} represents the number of Mashups that have a Calling Relationship with target API and are selected for using their attributes as the input of graph convolution operations, N^{DCom} represents the number of APIs that have a Directly Competitive Relationship with target API and are selected for using their attributes as the input of graph convolution operations, N^{IComS} represents the number of APIs that have a Indirectly Competitive Relationship with target API and are selected for using their attributes as the input of graph convolution operations, N^{IComP} represents the number of Providers that have a Indirectly Competitive Relationship with target API and are selected for using their attributes as the input of graph convolution operations, N^{Pro} represents the number of Categories that have a Providing Relationship with target API and are selected for using their attributes as the input of graph convolution operations, $a^{Joi}_{s*,t',i}$, $a^{Sub}_{s*,t'}$, $a^{Coo}_{s*,t',i}$, $a^{Cal}_{s*,t',i}$, $a^{DCom}_{s*,t',i}$, $a^{IComS}_{s*,t',i}$, $a^{IComP}_{s*,t',i}$, $a^{Pro}_{s*,t',i}$, represent the vectors as shown at the bottom of the next page respectively.

Where, $A(e)$ represents the operation of acquiring the vector composed of element e's attributes, $R(e_1, e_2)$ represents the operation of acquiring the relationship between element e_1 and e_2, $P(s)$ represents the operation of acquiring the provider which provides API s, $s^{Joi}_{s*,t',i}$ represents the API ranked i-th, according to a given criteria, in APIs that have a Joint Relationship with target API at moment t', $s^{Coo}_{s*,t',i}$ represents the API ranked i-th, according to a given criteria, in APIs that have a Collaborative Relationship with target API at moment t', $c^{Cal}_{s*,t',i}$ represents the Mashup ranked i-th, according to a given criteria, in Mashups that have a Calling Relationship with target API at moment t', $s^{DCom}_{s*,t',i}$ represents the API ranked i-th, according to a given criteria, in APIs that have a Directly Competitive Relationship with target API at moment t', $s^{ICom}_{s*,t',i}$ represents the API ranked i-th, according to a given criteria, in APIs that have a Indirectly Competitive Relationship with target API at moment t', $p^{ICom}_{s*,t',i}$ represents the Provider ranked i-th, according to a given criteria, in Providers that have a Indirectly Competitive Relationship with target API at moment t', $f^{Pro}_{s*,t'}$ represents the Category ranked i-th, according to a given criteria, in Categories that have a Indirectly Providing Relationship with target API at moment t', $p^{Sub}_{s*,t'}$ represents the Provider of target API at moment t'.

In the above definitions, using the combinations of MLP, which has strong expression ability and high adaptability, to extract the features of the elements associated with target API, makes it not only ensures the accuracy of GHSCN model, but also reduces the training complexity. And at the same time, ranking the elements which have different relationship with target API at each moment in time period $[t-T, t-1]$ according to a given criteria, and selecting the attributes of the top fixed number of elements as the input of vector $I^{Joi}_{[t-T,t-1]}$, $I^{Sub}_{[t-T,t-1]}$, $I^{Coo}_{[t-T,t-1]}$, $I^{Cal}_{[t-T,t-1]}$, $I^{IComS}_{[t-T,t-1]}$, $I^{IComP}_{[t-T,t-1]}$, $I^{DCom}_{[t-T,t-1]}$, $I^{Pro}_{[t-T,t-1]}$, guarantees the rotation invariance characteristic and fixed input dimension of the graph heterogeneous spatiotemporal convolution operations $\bigotimes^{Sup}$, $\bigotimes^{Req}$, $\bigotimes^{Com}$ and $\bigotimes^{Inh}$. Where, for the given criteria of ranking the elements which have different relationship with target API, this paper follows the principle that putting the elements with high correlation with the global popularity and in-category popularity of target API at the front, and design as follow:

1) The APIs that have a Joint Relationship, Collaborative Relationship and Indirect Competition Relationship with target API are sorted in descending order according to their average in-category popularity;
2) The Mashups of target API are sorted in descending order according to their update time;
3) The APIs that have a Directly Competition Relationship with target API, and the Providers that have an Indirectly Competition Relationship with target API,

are sorted in descending order according to the sum of their in-category popularity within the functional Categories provided by target API;

4) The Categories provided by target API are sorted in descending order to their global popularity.

b: GRAPH CONVOLUTIONAL KERNELS FOR API IN-CATEGORY POPULARITY PREDICTION

As shown in the sketch map of FIGURE 6, different from the graph convolutional operations for API global popularity prediction task, the graph convolutional operations for API in-category popularity prediction task need to deal with the target Category f^* (the Category within which the in-category popularity of target API needs to be predicted, and marked in yellow in FIGURE 6) and other Categories provided by target API s^* separately. Therefore, in each API in-category popularity prediction task, the graph heterogeneous spatiotemporal convolutional operation $\bigotimes^{Com}$ needs to be performed twice to respectively extract the features of elements of which the path to target API s^* need go through the target Category f^* and other Categories provided by target API s^* in the service ecosystem network, just as the convolution process of elements circled by the red dotted line in upper and lower parts of FIGURE 6 (It is worth noting that the input vectors $I^{IComS}_{[t-T,t-1]}$, $I^{IComP}_{[t-T,t-1]}$ and $I^{DCom}_{[t-T,t-1]}$ corresponding to $\bigotimes^{Com}$ also need to be constructed twice in accordance with the two different processing objects). The graph convolutional operations $\bigotimes^{Sup}$, $\bigotimes^{Req}$ and $\bigotimes^{Inh}$ are only executed once just like in the API global popularity prediction task, and the only difference is the input part of $\bigotimes^{Inh}$: the vector $I^{Pro}_{[t-T,t-1]}$ only be composed of the attributes of elements associated with other Categories provided by target API s^* except for target Category f^*.

V. EXPERIMENTS AND DISCUSSIONS

In this section, we use the data crawled from ProgrammableWeb to verify the API popularity prediction method in service ecosystems proposed in this paper, and analyze and discuss the experimental results.

A. INTRODUCTION OF DATA SET

1) RAW DATA FROM PROGRAMMABLEWEB

ProgrammableWeb is the largest Internet-based API information and news platform in the world, it records more than 22000 APIs, more than 7000 Mashups, more than 490 Categories, more than 13000 Providers, and a large number of SDKs, Libraries, Frameworks, Sample Sources Codes, Users, Developers, Teaching Documents, Comments and other peripheral data, which form a complex, diverse, relatively stable and complete service ecosystem. Therefore, this paper chooses the service ecosystem data recorded on ProgrammableWeb as the experimental samples.

Concretely, this paper will simply process and transform the original data crawled from the ProgrammableWeb, and save it in the form of service ecosystem network $G^{ES}_{[t_0,t_0+T]} = \left\{G^{ES}_{t_0}, G^{ES}_{t_0+1}, \cdots, G^{ES}_{t_0+T}\right\}$, that is, for each sampling moment t, the APIs, Mashups, Categories and Providers contained in ProgrammableWeb at that moment, as well as the numerical and enumeration data related to them (for example, the global popularity at moment t, the number of sample codes and authentication types, etc.) are saved in four node files in the form of node and node attribute of network G^{ES}_t respectively; four kinds of relationships, namely, Mashup-API, API-Provider, API-Category and Provider-Category, as well as the numerical data and enumeration data related to them (for example, the in-category popularity at moment t) are saved in four edge files in the form of edge and edge attributes of network G^{ES}_t respectively. Where, the sampling moments $t_0, t_0+1, \cdots, t_0+T$ represent the zero hour on the first day of each month in the period from June 1, 2005 to March 1, 2019 (as of the beginning of our experiments).

2) DATA SET SELECTED FOR EXPERIMENTS

As shown in FIGURE 8, the change trend of the number of APIs, Mashups, Categories and Providers on ProgrammableWeb has obviously different characteristics in different time periods. Therefore, in order to avoid the impact on

$$
\begin{aligned}
a^{Joi}_{s^*,t',i} &= \left(A\left(s^{Joi}_{s^*,t',i}\right) \quad A\left(R\left(s^{Joi}_{s^*,t',i}, p^{Sub}_{s^*,t'}\right)\right)\right)\\
a^{Sub}_{s^*,t'} &= \left(A\left(p^{Sub}_{s^*,t'}\right) \quad A\left(R\left(p^{Sub}_{s^*,t'}, s^*\right)\right)\right)\\
a^{Coo}_{s^*,t',i} &= \left(A\left(s^{Coo}_{s^*,t',i}\right) \quad A\left(R\left(s^{Coo}_{s^*,t',i}, c^{Cal}_{s^*,t',1}\right)\right) \quad \cdots \quad A\left(R\left(s^{Coo}_{s^*,t',i}, c^{Cal}_{s^*,t',N^{Cal}}\right)\right) \right)\\
a^{Cal}_{s^*,t',i} &= \left(A\left(c^{Cal}_{s^*,t',i}\right) \quad A\left(R\left(c^{Cal}_{s^*,t',i}, s^*\right)\right)\right)\\
a^{DCom}_{s^*,t',i} &= \left(A\left(s^{DCom}_{s^*,t',i}\right) \quad A\left(R\left(s^{DCom}_{s^*,t',i}, f^{Pro}_{s^*,t',1}\right)\right) \quad \cdots \quad A\left(R\left(s^{DCom}_{s^*,t',i}, f^{Pro}_{s^*,t',N^{Pro}}\right)\right) \right)\\
a^{IComS}_{s^*,t',i} &= \left(A\left(s^{ICom}_{s^*,t',i}\right) \quad A\left(R\left(s^{ICom}_{s^*,t',i}, P\left(s^{ICom}_{s^*,t',i}\right)\right)\right)\right)\\
a^{IComP}_{s^*,t',i} &= \left(A\left(p^{ICom}_{s^*,t',i}\right) \quad A\left(R\left(p^{ICom}_{s^*,t',i}, f^{Pro}_{s^*,t',1}\right)\right) \quad \cdots \quad A\left(R\left(p^{ICom}_{s^*,t',i}, f^{Pro}_{s^*,t',N^{Pro}}\right)\right) \right)\\
a^{Pro}_{s^*,t',i} &= \left(A\left(f^{Pro}_{s^*,t',i}\right) \quad A\left(R\left(f^{Pro}_{s^*,t',i}, s^*\right)\right)\right)
\end{aligned}
$$

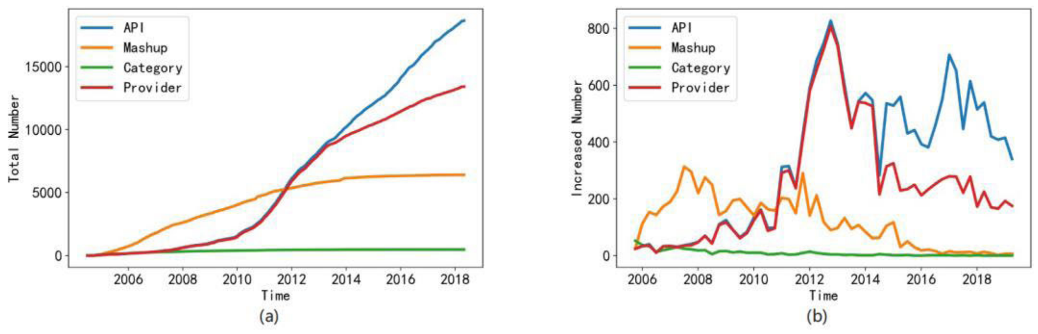


FIGURE 8. The change trend of the number of APIs, Mashups, Categories and Providers on ProgrammableWeb. Where, (a) is the total number growth trend chart, (b) is the quarterly growth statistics chart.

the general trend of API's global popularity and in-category popularity caused by the change of self-promotion and operation strategy of ProgrammableWeb, and the unexpected events (such as the emergence of new development models or tools) in the field of service computing and Internet-based API. In this paper, the service ecosystem network data from January 1, 2006 to January 1, 2011 are selected as the experimental samples (the change trend of the number of APIs, Mashups, Categories and Providers are relatively fixed), for reducing the experimental error caused by data imbalance.

Concretely, at each sampling moment t from January 1, 2006 to January 1, 2011, the global popularity attributes of all APIs in the service ecosystem network G_t^{ES} are taken as sample Y^{GP} of API global popularity prediction model GHSCN-GP, with a total of 148141; the in-category popularity attributes of all API-Category relationships are taken as sample Y^{ICP} of service in-category popularity prediction model GHSCN-ICP, with a total of 55155.

3) EXPERIMENTAL ENVIRONMENT

The experiments in this paper were conducted on a server with CentOS 7 operating system, 8-core CPU (Xeon E5-2640 v3) of Intel, and 125G memory. Further, for the convenience of experiments implementation, the following software development frameworks and tools are used: NumPy 1.18.1, Pandas 0.25.1, Torch 1.4.0, NetworkX 2.4 and Python 3.6.1.

4) MODEL CONFIGURATION

It can be found from Subsection 2 of Section B of Chapter IV that there are eight constants that need to be configurated according to the scale and structure of the underlying service ecosystem (parameters that do not change with the training process of the model), namely T, N^{Joi}, N^{Coo}, N^{Cal}, N^{DCom}, N^{IComS}, N^{IComP} and N^{Pro}. Therefore, before the experiments, we first set the values of these eight constants according to the information about the service ecosystem recorded on ProgrammableWeb from January 1, 2006 to January 1, 2011.

Concretely, as shown in FIGURE 9, to set more reasonable values for N^{Joi}, N^{Coo}, N^{Cal}, N^{DCom}, N^{IComS}, N^{IComP} and N^{Pro}, we respectively counted the number distributions of the APIs that have a Joint Relationship with each API, the APIs that have a Collaborative Relationship with each API, the Mashups that have a Calling Relationship with each API, the APIs that have a Directly Competitive Relationship with each API, the APIs that have a Indirectly Competitive Relationship with each API, the Providers that have a Indirectly Competitive Relationship with each API and the Categories that have a Providing Relationship with each API in the service ecosystem network samples obtained from ProgrammableWeb during the period from January 1, 2006 to January 1, 2011. From the results, it can be seen that these distributions are obviously concentrated in a small interval. Thus, after considering the accuracy and operation efficiency of GHSCN model, we select the non-zero median value of the corresponding distributions as the values of N^{Joi}, N^{Coo}, N^{Cal}, N^{DCom}, N^{IComS}, N^{IComP} and N^{Pro} in our experiments, which are 7, 48, 253, 128, 165, 126 and 2 respectively.

Further, to set reasonable values for T, we firstly calculate the multiple correlation coefficient values (R^{GP-Ipt} and $R^{ICP-Ipt}$) between samples (Y^{GP} and Y^{ICP}) and the corresponding prediction model inputs (i.e., $I_{[t-T,t-1]}^{Joi}$, $I_{[t-T,t-1]}^{Sub}$, $I_{[t-T,t-1]}^{Coo}$, $I_{[t-T,t-1]}^{Cal}$, $I_{[t-T,t-1]}^{IComS}$, $I_{[t-T,t-1]}^{IComP}$, $I_{[t-T,t-1]}^{DCom}$ and $I_{[t-T,t-1]}^{Pro}$ in Subsection 2 of Section B of Chapter IV) when T takes different values. The calculation results are shown in FIGURE 10, and it can be seen that when T is greater than 4, there is no significant increase in R^{GP-Ipt} and $R^{ICP-Ipt}$. Thus, after considering the accuracy and efficiency of GHSCN model, we set $T = 4$ in our experiments.

B. DESIGN AND RESULTS OF EXPERIMENTS

Based on the above data set, computing environment and model configurations, two groups of experiments are designed to verify the performance and application value of the API popularity prediction method in service ecosystems proposed in this paper.

1) PERFORMANCE VALIDATION EXPERIMENTS

As with the methods verifying the performance of GNN models in most existing literature, this group of experiments are divided into two parts: the prediction accuracy verification

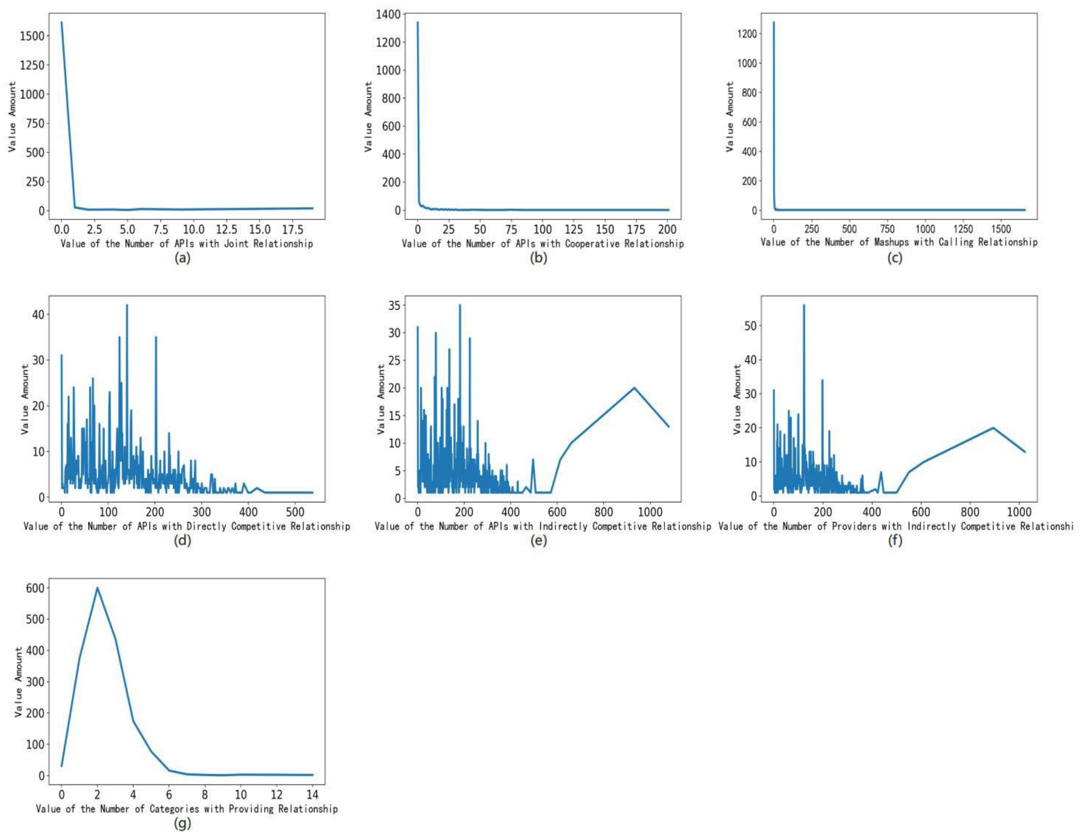


FIGURE 9. Figures (a) to (g) are respectively the number distributions of the APIs that have a Joint Relationship with each API, the APIs that have a Collaborative Relationship with each API, the Mashups that have a Calling Relationship with each API, the APIs that have a Directly Competitive Relationship with each API, the APIs that have a Indirectly Competitive Relationship with each API, the Providers that have a Indirectly Competitive Relationship with each API and the Categories that have a Providing Relationship with each API in the service ecosystem network samples obtained from ProgrammableWeb during the period from January 1, 2006 to January 1, 2011.

experiments and training speed verification experiments, and the details are as follows:

a: PREDICTION ACCURACY VERIFICATION EXPERIMENTS

The purpose of this part of experiments is to verify the accuracy of GHSCN model in predicting the global popularity and in-category popularity of APIs by using the full sample data set (i.e., the service ecosystem data on ProgrammableWeb from January 1, 2006 to January 1, 2011). Thus, whether it is the API global popularity prediction model GHSCN-GP or the in-category popularity prediction model GHSCN-ICP, the steps of experiments are as follows:

1) Randomly divide the full sample data set Y (Y^{GP} for GHSCN-GPF verification experiment, and Y^{ICP} for GHSCN-ICPF verification experiment) into training data set Y^{Trn} and test data set Y^{Tst} in a ratio of $9.5 : 0.5$;
2) Use the tenfold cross-validation method to train the model in the training data set Y^{Trn}, that is, after each hyperparameters adjustment, divide Y^{Trn} into ten parts randomly; then, perform ten model training, and each training uses the data of different parts of Y^{Trn} as the validation data set; finally, use the average value of ten training results as the loss function value of the current hyperparameters;
3) Test the accuracy of the model in test data set Y^{Tst}, that is, count and record the values of Mean Absolute Error γ^{MAE} and Coefficient of Determination γ^{CoD};
4) Judge whether five independent tests (Steps 1) to 3) is an independent and complete test) have been performed, if so, skip to the next step, otherwise, return to Step 1) to start the next test;
5) Calculate the average values ($\overline{\gamma^{MAE}}$ and $\overline{\gamma^{CoD}}$) of Mean Absolute Error γ^{MAE} and Coefficient of Determination γ^{CoD} of the model obtained from five tests, and use them as the final experimental result.

Where, using the tenfold cross-validation method to train model is to avoid the over fitting phenomenon, using the average value of five independent test results as the final experimental result is to reduce random errors, and due to the API popularity prediction in service ecosystems is a regression problem, using the Mean Absolute Error γ^{MAE} and Co-efficient of Determination γ^{CoD} defined as follows as the

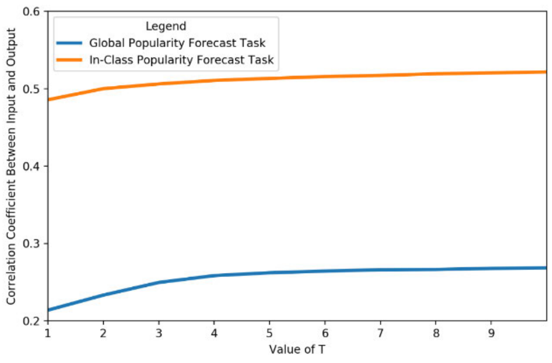


FIGURE 10. The curves of the multiple correlation coefficient values, between the samples Y^{GP} and Y^{ICP} (the output of models) and the corresponding prediction models inputs, with the values of T.

measurement indicators in this part of experiments:

$$\gamma^{MAE} = \frac{1}{N^{Tst}} \sum_{i=1}^{N^{Tst}} \left| y_i^{Tst} - \hat{y}_i^{Tst} \right|$$

$$\gamma^{CoD} = 1 - \frac{\sum_{i=1}^{N^{Tst}} \left(y_i^{Tst} - \hat{y}_i^{Tst} \right)^2}{\sum_{i=1}^{N^{Tst}} \left(y_i^{Tst} - \overline{Y^{Tst}} \right)^2}$$

where, N^{Tst} represents the total number of samples in the test data set, y_i^{Tst} and $\hat{y}_i^{Tst}$ represent the real value and predicted value of the sample i in the test data set, and $\overline{Y^{Tst}}$ represents the expected value of the samples in the test data set.

TABLE 1. The experimental results of prediction accuracy verification.

Model	Mean Absolute Error $\overline{\gamma^{MAE}}$	Coefficient of Determination $\overline{\gamma^{CoD}}$
GHSCN-GPF	0.0211309	0.54668784
GHSCN-ICPF	0.00212808	0.00012791

After completing this part of experiments according to the above steps, we get the experimental results as shown in TABLE 1 and FIGURE 11. It can be seen that both the GHSCN-GP model and the GHSCN-ICP model proposed in this paper perform well in general. They not only have low Mean Absolute Error value, but also have very fast convergence speed (the convergence can be achieved within 100 iterations). Where, the only point that is not outstanding is that the Coefficient of Determination of GHSCN-ICPF model is relatively low, and the reason is that sample Y^{ICP} contains a large number of zero samples, and the difference between non-zero samples is very small, resulting in the value of $\sum_{i=1}^{N^{Tst}} \left(y_i^{Tst} - \overline{Y^{Tst}} \right)^2$ in the formula of Coefficient of Determination is only 0.0002332859583584961.

b: TRAINING SPEED VERIFICATION EXPERIMENTS

The main idea of this part of experiments is to count the time and iterations number spent in training in different size of

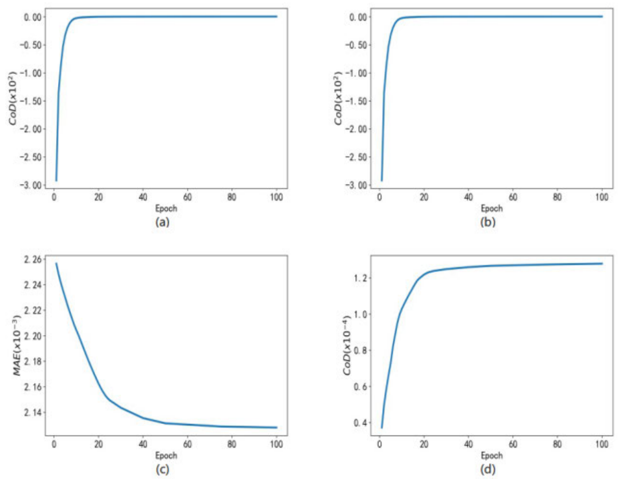


FIGURE 11. The prediction accuracy convergence curve of the model proposed in this paper. (a), (b), (c) and (d) are respectively the convergence curve of Mean Absolute Error of GHSCN-GP model, Coefficient of Determination of GHSCN-GP model, Mean Absolute Error of GHSCN-ICP model and Coefficient of Determination of GHSCN-ICP model with the number of process iterations.

sample data sets to verify the training speed of the model proposed in this paper. Thus, the steps of training speed verification experiments of API global popularity prediction model GHSCN-GP and in-category popularity prediction model GHSCN-ICP are all as follows:

1) Randomly select 100, 500, 1000, 5000, 10000, and 50000 samples from the full sample data set Y (Y^{GP} for GHSCN-GP verification experiment, and Y^{ICP} for GHSCN-ICP verification experiment) to form sub-data sets Y_{100}, Y_{500}, Y_{1000}, Y_{5000}, Y_{10000} and Y_{50000};
2) Carry out eight groups of experiments, and each group of experiments set the hidden layer number, hidden layer neuron number and output layer dimension of each MLP (including the four graph heterogeneous convolutional operations $\bigotimes^{Sup}$, $\bigotimes^{Req}$, $\bigotimes^{Com}$, $\bigotimes^{Inh}$ and the last full connection layer of GHSCN model) according to the parameters in different rows in TABLE 2; then, train the model with Y_{100}, Y_{500}, Y_{1000}, Y_{5000}, Y_{10000} and Y_{50000} as training data sets, respectively; and finally, count the training time t^{Trn} (the value when the model reaches convergence state);
3) Judge whether five independent tests (Steps 1) to 2) is an independent and complete test) have been performed, if so, skip to the next step, otherwise, return to Step 1) to start the next test;
4) Calculate the average value $\overline{t^{Trn}}$ of training time t^{Trn} of the model obtained from the five tests under different settings and different size of data sets, and use them as the final experimental result.

After completing this part of experiments according to the above steps, we get the experimental results as shown in FIGURE 12. It can be seen that the training time of the model proposed in this paper almost increases linearly with the increase of the number of training samples under any

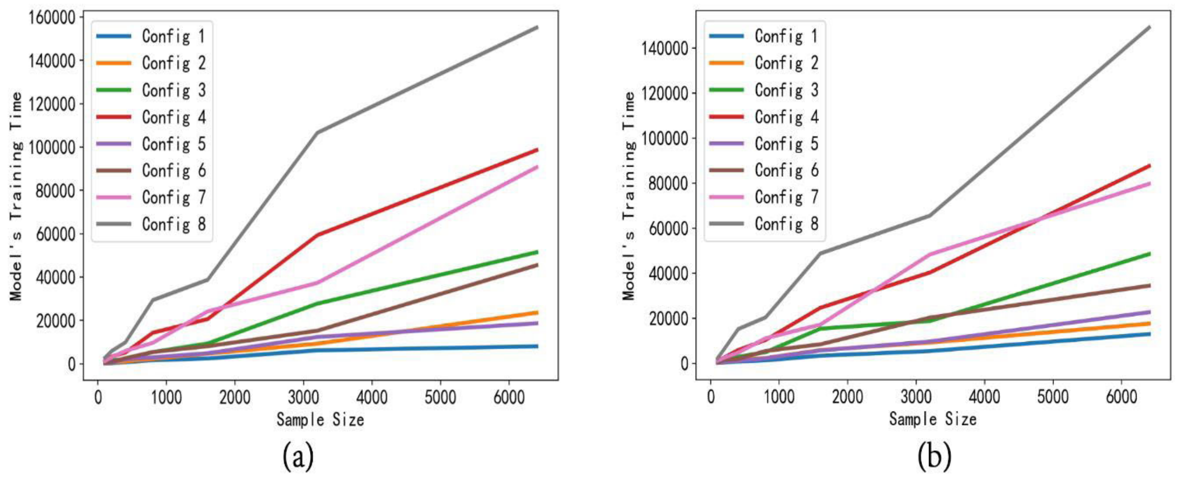


FIGURE 12. Figure (a) and (b) respectively represent the change curves of GHSCN-GP model's and GHSCN-ICP model's training time with the number of training samples, under different configurations.

TABLE 2. The parameters of each MLP in the model, that is, the proportion of the number of their hidden layer layers, the number of their hidden layer neurons and their output layer dimension in the corresponding input layer dimension (it is worth noting that due to the output dimension of the fully connected layer of the model is always 1, so only two other parameters are given).

MLP / Parameters meter / ID	$\otimes^{Sup}$	$\otimes^{Req}$	$\otimes^{Com}$	$\otimes^{Inh}$	Fully Connected Layer
1	$1,\frac{1}{2},\frac{1}{8}$	$1,\frac{1}{2},\frac{1}{8}$	$1,\frac{1}{2},\frac{1}{8}$	$1,\frac{1}{2},\frac{1}{8}$	$2,\frac{1}{2}$
2	$1,\frac{2}{3},\frac{1}{6}$	$1,\frac{2}{3},\frac{1}{6}$	$1,\frac{2}{3},\frac{1}{6}$	$1,\frac{2}{3},\frac{1}{6}$	$2,\frac{2}{3}$
3	$1,1,\frac{1}{4}$	$1,1,\frac{1}{4}$	$1,1,\frac{1}{4}$	$1,1,\frac{1}{4}$	2,1
4	$1,\frac{3}{2},\frac{1}{2}$	$1,\frac{3}{2},\frac{1}{2}$	$1,\frac{3}{2},\frac{1}{2}$	$1,\frac{3}{2},\frac{1}{2}$	$2,\frac{3}{2}$
5	$2,\frac{1}{2},\frac{1}{8}$	$2,\frac{1}{2},\frac{1}{8}$	$2,\frac{1}{2},\frac{1}{8}$	$2,\frac{1}{2},\frac{1}{8}$	$4,\frac{1}{2}$
6	$2,\frac{2}{3},\frac{1}{6}$	$2,\frac{2}{3},\frac{1}{6}$	$2,\frac{2}{3},\frac{1}{6}$	$2,\frac{2}{3},\frac{1}{6}$	$4,\frac{2}{3}$
7	$2,1,\frac{1}{4}$	$2,1,\frac{1}{4}$	$2,1,\frac{1}{4}$	$2,1,\frac{1}{4}$	4,1
8	$2,\frac{3}{2},\frac{1}{2}$	$2,\frac{3}{2},\frac{1}{2}$	$2,\frac{3}{2},\frac{1}{2}$	$2,\frac{3}{2},\frac{1}{2}$	$4,\frac{3}{2}$

configuration. Therefore, through reasonable configuration of parameters, our model can also have high training efficiency in large-scale sample learning tasks. Furthermore, by comparing the training time of the model under different configurations, it can be found that the number of neurons in hidden layers has a greater impact on the performance than the number of hidden layers. Thus, when applying GHSCN model, we suggest to increase the number of hidden layers to improve the prediction accuracy of the model, rather than increase the number of neurons.

2) APPLICATION VALUE VALIDATION EXPERIMENTS

Different from the previous group of experiments to verify the theoretical performance indicators of GHSCN model, this group of experiments aims to demonstrate and verify the application value of GHSCN model in service ecosystems through simulations. Thus, we choose the common problem in the service ecosystem, fault source identification, as an example to demonstrate the importance of the introduction of the model proposed in this paper to service ecosystems.

Concretely, we conducted 30 experiments on different service ecosystem networks sampled in the period from January 1, 2006 to January 1, 2011. In each experiment, we firstly use different metrics (Degree, Betweenness Centrality, Closeness Centrality, PageRank and global popularity predicted by CHSCN model) as the basis for selecting the deployment location of monitoring agents in the underlying service ecosystem network; then, using the SI model with random assignments of propagation probability of edges, conduct 100 times of fault propagation simulation; and finally, perform fault source identification (three classic source identification algorithms, Concentric Center, Jordan Center and DMP, are used respectively) based on the information monitored by monitoring agents selected by different metrics and the finally state of all nodes in the underlying service ecosystem network, respectively, and calculate the corresponding source identification difficulties as the following formula (taking one of them as an example):

$$\gamma_{Pop} = \frac{1}{100}\sum_{i=1}^{100}\frac{\left(r^{TrueSource}_{Pop,Conc,i}+r^{TrueSource}_{Pop,Jord,i}+r^{TrueSource}_{Pop,DMP,i}\right)\Big/3}{|V_{Mashup}|+|V_{API}|+|V_{Provider}|}$$

where, γ_{Pop} represents the source identification difficulty based on the information monitored by monitoring agents selected by global popularity predicted by CHSCN model (γ_{Deg}, γ_{BC}, γ_{CC}, γ_{Pr} and γ_{All} respectively represent the corresponding value for Degree, Betweenness Centrality, Closeness Centrality, PageRank and "all nodes in the underlying service ecosystem network"), V_{Mashup}, V_{API} and $V_{Provider}$ respectively represent the set of Mashups, APIs and Providers in the underlying service ecosystem network, $r^{TrueSource}_{Pop,Conc,i}$,

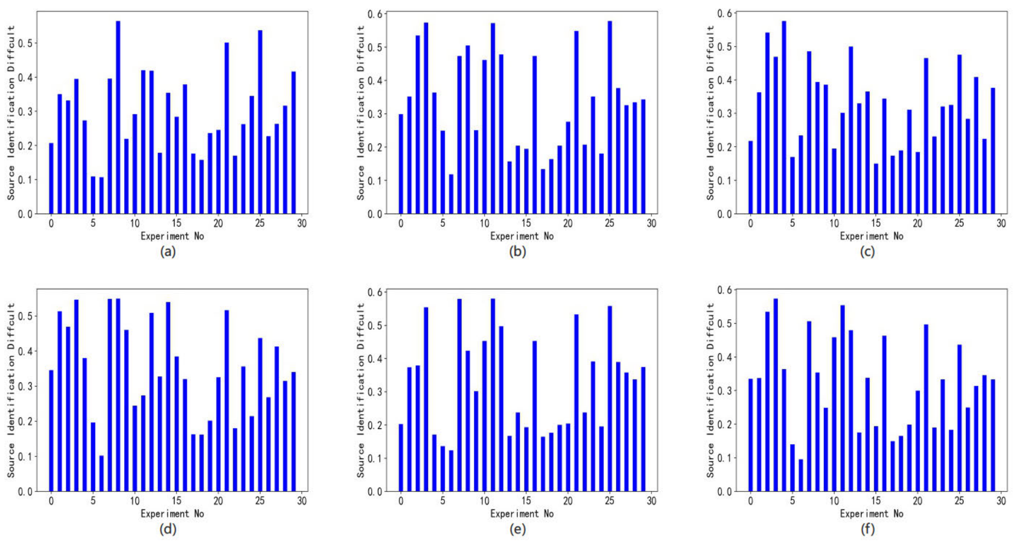


FIGURE 13. Figure (a)∼(f) are respectively the distribution of the difficulty of source identification in 30 experiments, when using different metrics (degree, betweenness centrality, closeness centrality, PageRank and global popularity predicted by CHSCN model) as the basis for selecting the deployment location of monitoring agents in service ecosystem networks.

$r^{TrueSource}_{Pop,Jord,i}$ and $r^{TrueSource}_{Pop,DMP,i}$ respectively represent the ranking values of the real fault source in the i-th fault propagation simulation, among all nodes in the underlying service ecosystem network, for the estimated value (the measure of the probability that a node in the network is the source of fault) calculated in source identifications by Concentric Center, Jordan Center and DMP based on the information monitored by monitoring agents selected by global popularity predicted by CHSCN model.

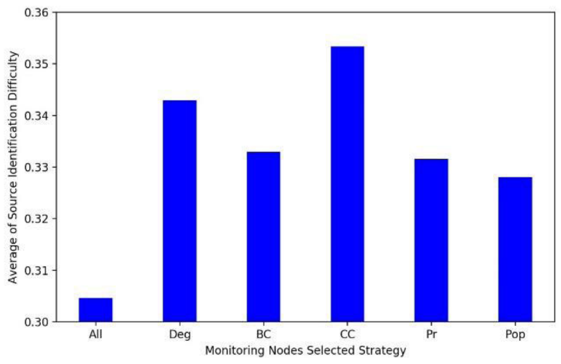


FIGURE 14. The average values of the difficulty of source identification, when using different metrics (degree, betweenness centrality, closeness centrality, PageRank and global popularity predicted by CHSCN model) as the basis for selecting the deployment location of monitoring agents in service ecosystem networks.

After completing this group of experiments according to the above steps, we get the experimental results as shown in FIGURE 13 and FIGURE 14. It can be seen that when a fault propagation occurs in service ecosystems, the difficulty of source identification based on the final state of the top 10% of APIs for global popularity predicted by GHSCN model is lowest, and is closer to the difficulty of source identification based on the final state of all nodes in service ecosystems. Therefore, using the API global popularity predicted by the model proposed in this paper, as the basis for selecting the deployment location of the monitoring agent in service ecosystems, can effectively reduce the difficulty of fault source identification and ensure the security and accountability of the service ecosystem.

VI. CONCLUSION

The APIs, Mashups, Providers corresponding to services, compositions, and providers form an evolving service ecosystem. Understanding the popularity of API which includes QoS, function, historical usage pattern and social relationship can help to recommend adequate services, as well as promote the security and stability of service ecosystem.

In this paper, we propose a Graph Neural Network-based Service Popularity Prediction Method (GHSCN) in service ecosystem, and then carry out extensive experiments on the data set crawled from ProgrammableWeb. The experiments showed that our approach have higher accuracy because four different Heterogeneous Spatiotemporal Convolutional Kernels are proposed to extract the features of different elements which have different mechanisms to affect the popularity of target service, and that the prediction APIs popularity is meaningful for selecting placement of fault monitoring agents in service ecosystem.

For future work, we have the following four suggestions: (1) modeling more elements and information, such as users, SDKs and teaching documents of API, in service ecosystem in the form of a network, just like GSEN model, to further improve the accuracy of GNN-based API popularity prediction methods; (2) introduce some knowledge and experience in the field of service ecosystem, such as the relationships between API popularity and API centrality (PageRank and Closeness Centrality, etc.) in the underlying service ecosystem network, to improve the convolutional kernels of GHSCN model, for further improving the model's accuracy and speed; (3) based on the GHSCN model in this paper, adding a Graph Auto Encoder (GAE) to encode the information of elements related to Mashups, it is can easily design a GNN-based Mashup-API link prediction method. (4) Combining API popularity with other aspects in service optimization field such as compositions or recommendations.

ACKNOWLEDGMENT

(Zhong Li and Xiaochen Liu contributed equally to this work.)

REFERENCES

[1] L. Yao, Q. Z. Sheng, A. Segev, and J. Yu, "Recommending Web services via combining collaborative filtering with content-based features," in *Proc. ICWS*, Santa Clara, CA, USA, Jun. 2013, pp. 42–49.

[2] C. Li, R. Zhang, J. Huai, and H. Sun, "A novel approach for API recommendation in mashup development," in *Proc. ICWS*, Anchorage, AK, USA, Jun. 2014, pp. 289–296.

[3] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "Collaborative Web service QoS prediction via neighborhood integrated matrix factorization," *IEEE Trans. Services Comput.*, vol. 6, no. 3, pp. 289–299, Jul. 2013.

[4] P. Wang, Z. Ding, C. Jiang, M. Zhou, and Y. Zheng, "Automatic Web service composition based on uncertainty execution effects," *IEEE Trans. Services Comput.*, vol. 9, no. 4, pp. 551–565, Jul. 2016.

[5] Y. Jiang, J. Liu, M. Tang, and X. Liu, "An effective Web service recommendation method based on personalized collaborative filtering," in *Proc. ICWS*, Washington, DC, USA, Jul. 2011, pp. 211–218.

[6] B. Tapia, R. Torres, and H. Astudillo, "Simplifying mashup component selection with a combined similarity-and social-based technique," in *Proc. ACM ICPS*, 2011, p. 8.

[7] J. Cao, W. Xu, L. Hu, J. Wang, and M. Li, "A social-aware service recommendation approach for mashup creation," *IJWSR*, vol. 10, no. 1, pp. 53–72, Jan. 2013.

[8] T. Liang, L. Chen, J. Wu, G. Xu, and Z. Wu, "SMS: A framework for service discovery by incorporating social media information," *IEEE Trans. Services Comput.*, vol. 12, no. 3, pp. 384–397, May 2019.

[9] A. Tatar, J. Leguay, P. Antoniadis, A. Limbourg, M. D. de Amorim, and S. Fdida, "Predicting the popularity of online articles based on user comments," in *Proc. WIMS*, 2011, p. 67.

[10] T. Wu, M. Timmers, D. D. Vleeschauwer, and W. V. Leekwijck, "On the use of reservoir computing in popularity prediction," in *Proc. INTERNET*, Valencia, Spain, Sep. 2010, pp. 19–24.

[11] H. Pinto, J. M. Almeida, and M. A. Gonçalves, "Using early view patterns to predict the popularity of youtube videos," in *Proc. WSDM*, Rome, Italy, 2013, pp. 365–374.

[12] D. Bianchini, V. D. Antonellis, and M. Melchiori, "A multi-perspective framework for Web API search in enterprise mashup design," in *Proc. CAiSE*, Valencia, Spain, 2013, pp. 353–368.

[13] R. Torres, B. Tapia, and H. Astudillo, "Improving Web API discovery by leveraging social information," in *Proc. ICWS*, Washington, DC, USA, Jul. 2011, pp. 744–745.

[14] D. Bianchini, V. D. Antonellis, and M. Melchiori, "A linked data perspective for effective exploration of Web APIs repositories," in *Proc. ICWE*, Aalborg, Denmark, 2013, pp. 506–509.

[15] A. Ranabahu, M. Nagarajan, A. P. Sheth, and K. Verma, "A faceted classification based approach to search and rank Web APIs," in *Proc. ICWS*, Beijing, China, Sep. 2008, pp. 177–184.

[16] D. Bianchini, V. D. Antonellis, and M. Melchiori, "Model-based search and ranking of Web apis across multiple repositories," in *Proc. WISE*, Thessaloniki, Greece, 2014, pp. 218–233.

[17] B. Cao, J. Liu, M. Tang, Z. Zheng, and G. Wang, "Mashup service recommendation based on user interest and social network," in *Proc. ICWS*, Santa Clara, CA, USA, Jun. 2013, pp. 99–106.

[18] K. Huang, Y. Fan, and W. Tan, "An empirical study of programmable Web: A network analysis on a service-mashup system," in *Proc. ICWS*, Honolulu, HI, USA, Jun. 2012, pp. 552–559.

[19] Y. Zhong, Y. Fan, K. Huang, W. Tan, and J. Zhang, "Time-aware service recommendation for mashup creation," *IEEE Trans. Serv. Comput.*, vol. 8, no. 3, pp. 356–368, May/Jun. 2015.

[20] K. Huang, Y. Fan, and W. Tan, "Recommendation in an evolving service ecosystem based on network prediction," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 3, pp. 906–920, Jul. 2014.

[21] Z. Zheng, H. Ma, M. R. Lyu, and I. King, "WSRec: A collaborative filtering based Web service recommender system," in *Proc. ICWS*, Los Angeles, CA, USA, Jul. 2009, pp. 437–444.

[22] J. G. Lee, S. Moon, and K. Salamatian, "Modeling and predicting the popularity of online contents with cox proportional hazard regression model," *Neurocomputing*, vol. 76, no. 1, pp. 134–145, Jan. 2012.

[23] M. Gori, G. Monfardini, and F. Scarselli, "A new model for learning in graph domains," in *Proc. IJCNN*, MOntreal, QC, Canada, Jul./Aug. 2005, pp. 729–734.

[24] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. NIPS*, Barcelona, Spain, 2016, pp. 3844–3852.

[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. ICLR*, Toulon, France, 2017, pp. 1–13.

[26] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. ICLR*, Scottsdale, AZ, USA, 2014, pp. 1–14.

[27] R. Li, S. Wang, F. Zhu, and J. Huang, "Adaptive graph convolutional neural networks," in *Proc. AAAI*, New Orleans, LA, USA, 2018, pp. 3546–3553.

[28] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. ICML*, Sydney, NSW, Australia, 2017, pp. 1263–1272.

[29] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NeurIPS*, Long Beach, CA, USA, 2017, pp. 1024–1034.

[30] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Proc. NeurIPS*, Montreal, QC, Canada, 2016, pp. 1993–2001.

[31] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. ICML*, New York, NY, USA, 2016, pp. 2014–2023.

[32] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. KDD*, London, U.K., Jul. 2018, pp. 1416–1424.

[33] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. CVPR*, Honolulu, HI, USA, Jul. 2017, pp. 5425–5434.

[34] Z. Liu, C. Chen, L. Li, J. Zhou, X. Li, L. Song, and Y. Qi, "Geniepath: Graph neural networks with adaptive receptive paths," in *Proc. AAAI*, Honolulu, HI, USA, 2019, pp. 4424–4431.

[35] C. Zhuang and Q. Ma, "Dual graph convolutional networks for graph-based semi-supervised classification," in *Proc. WWW*, Lyon, France, 2018, pp. 499–508.

[36] D. V. Tran, N. Navarin, and A. Sperduti, "On filter size in graph convolutional networks," in *Proc. SSCI*, Bengaluru, India, Nov. 2018, pp. 1534–1541.

[37] M. Zhang, Z. Cui, M. Neumann, and C. Yixin, "An end-to-end deep learning architecture for graph classification," in *Proc. AAAI*, New Orleans, LA, USA, 2018, pp. 4438–4445.

[38] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Proc. NeurIPS*, Montreal, QC, Canada, 2018, pp. 4805–4815.

[39] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, Vancouver, BC, Canada, 2018, pp. 1–12.

[40] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D. Yeung, "GaAN: Gated attention networks for learning on large and spatiotemporal graphs," in *Proc. UAI*, Monterey, CA, USA, 2018, pp. 339–349.

[41] J. B. Lee, R. Rossi, and X. Kong, "Graph classification using structural attention," in *Proc. KDD*, London, U.K., Jul. 2018, pp. 1666–1674.

[42] S. Abu-El-Haija, B. Perozzi, R. Al-Rfou, and A. Alemi, "Watch your step: Learning node embeddings via graph attention," in *Proc. NeurIPS*, Montreal, QC, Canada, 2018, pp. 9180–9190.

[43] S. Cao, W. Lu, and Q. Xu, "Deep neural networks for learning graph representations," in *Proc. AAAI*, Phoenix, AZ, USA, 2016, pp. 1145–1152.

[44] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proc. KDD*, San Francisco, CA, USA, Aug. 2016, pp. 1225–1234.

[45] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016, *arXiv:1611.07308*. [Online]. Available: http://arxiv.org/abs/1611.07308

[46] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2018, *arXiv:1802.04407*. [Online]. Available: https://arxiv.org/abs/1802.04407

[47] W. Yu, C. Zheng, W. Cheng, C. C. Aggarwal, D. Song, B. Zong, H. Chen, and W. Wang, "Learning deep network representations with adversarially regularized autoencoders," in *Proc. KDD*, London, U.K., Jul. 2018, pp. 2663–2671.

[48] K. Tu, P. Cui, X. Wang, P. S. Yu, and W. Zhu, "Deep recursive network embedding with regular equivalence," in *Proc. KDD*, London, U.K., Jul. 2018, pp. 2357–2366.

[49] J. You, R. Ying, X. Ren, W. L. Hamilton, and J. Leskovec, "GraphRNN: A deep generative model for graphs," 2018, *arXiv:1802.08773*. [Online]. Available: https://arxiv.org/abs/1802.08773

[50] Y. Li, O. Vinyals, C. Dyer, R. Pascanu, and P. Battaglia, "Learning deep generative models of graphs," in *Proc. ICLR*, Vancouver, BC, Canada, 2018.

[51] N. De Cao and T. Kipf, "MolGAN: An implicit generative model for small molecular graphs," 2018, *arXiv:1805.11973*. [Online]. Available: http://arxiv.org/abs/1805.11973

[52] A. Bojchevski, O. Shchur, D. Zügner, and S. Günnemann, "NetGAN: Generating graphs via random walks," 2018, *arXiv:1803.00816*. [Online]. Available: http://arxiv.org/abs/1803.00816

[53] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," in *Proc. ICLR*, Vancouver, BC, Canada, 2018, pp. 1–33.

[54] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. IJCAI*, Jul. 2018, pp. 3634–3640.

[55] C. Wu, X.-J. Wu, and J. Kittler, "Spatial residual layer and dense connection block enhanced spatial temporal graph convolutional network for skeleton-based action recognition," in *Proc. ICCVW*, Oct. 2019, pp. 1740–1748.

[56] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-RNN: Deep learning on spatio-temporal graphs," in *Proc. KDD*, Jun. 2016, pp. 5308–5317.

[57] J. F. Moore, "Predators and prey: A new ecology of competition," *Harvard Bus. Rev.*, vol. 71, no. 3, pp. 75–86, 1993.

[58] J. Spohrer, P. P. Maglio, J. Bailey, and D. Gruhl, "Steps toward a science of service systems," *IEEE Computer*, vol. 40, no. 1, pp. 71–77, Jan. 2007.

[59] G. Scheithauer, S. Augustin, and G. Wirtz, "Describing services for service ecosystems," in *Proc. ICSOC*, Sydney, NSW, Australia, 2008, pp. 242–255.

ZHONG LI received the B.S. degree in computer science from Beihang University, where he is currently pursuing the Ph.D. degree. His main research interests include cloud computing technology, social network analysis, and source identification of information diffusion.

XIAOCHEN LIU received the B.S. and M.S. degrees from the School of Computer Science and Technology, Beijing University of Posts and Telecommunications, China. She is currently pursuing the Ph.D. degree with the School of Computer Science and Engineering, Beihang University, Beijing, China. Her research interests include network management, cloud service management, and cloud security analysis.

TIANBO WANG (Member, IEEE) received the Ph.D. degree in computer application from Beihang University, Beijing, China, in 2018. He is currently a Lecturer with Beihang University. He has participated in several national natural science foundations and other research projects. His research interests include network and information security, intrusion detection technology, and information countermeasure.

WENHUI HE received the B.S. degree in information safety from Central South University, Hunan, China, in 2018. She is currently pursuing the master's degree with the School of Computer Science and Engineering, Beihang University. Her research interests include services computing and graph neural networks.

CHUNHE XIA received the Ph.D. degree in computer science and engineering from Beihang University, Beijing, China, in 2003. He is currently the Head of the Beijing Key Laboratory of Network Technology, Beihang University. His research interests include network security, network management, and network measurement.