MFA: Web API recommendation based on service multiple feature aggregation

Guobing Zou*, Chunhua Zeng, Yue Zhu, Pengtao Li, Song Yang and Shengxiang Hu

School of Computer Engineering and Science,
Shanghai University,
Shanghai, China
Email: gbzou@shu.edu.cn
Email: kriya@shu.edu.cn
Email: illescionor278@gmail.com
Email: 22721507@shu.edu.cn
Email: yangsong@shu.edu.cn
Email: shengxianghu@shu.edu.cn
*Corresponding author

Abstract: As the number of web services continues to increase, it has become a challenging problem to provide developers with accurate and efficient web services that meet Mashup requirements. To solve this problem, various methods have been proposed to recommend APIs to match the needs of Mashups, and have achieved great success. However, due to the uneven quality of service descriptions, there are some challenges in feature extraction, utilisation of service meta-information, and textual requirements understanding. Therefore, we propose a Web API recommendation method (FMA) based on service multiple feature aggregation. FMA uses the attention mechanism model to mine the semantic features of similar services and enhance the features of Mashup services and Web API services. The service category is used as the basis for constructing the graph network, and multiple service features are mined and integrated through the hierarchical feature aggregation algorithm of the graph convolution network to further enhance the service features, thereby significantly improving the Web API recommendation effect. We conduct extensive experiments on a large-scale real-world dataset called ProgrammableWeb, and the results show that FMA outperforms existing baseline methods on multiple evaluation metrics.

Keywords: Web API; Mashup service; API recommendation; multiple feature aggregation; attention mechanism.

Reference to this paper should be made as follows: Zou, G., Zeng, C., Zhu, Y., Li, P., Yang, S. and Hu, S. (xxxx) 'MFA: Web API recommendation based on service multiple feature aggregation', *Int. J. Computational Science and Engineering*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Guobing Zou is a Full Professor and the Dean of the Department of Computer Science and Technology, Shanghai University, China. He received his PhD in Computer Science from Tongji University, Shanghai, China, 2012. He has worked as a Visiting Scholar in the Department of Computer Science and Engineering at Washington University in St. Louis from 2009 to 2011, USA. His current research interests mainly focus on services computing, edge computing, recommender systems. He has published more than 100 papers on premier international journals and conferences, including *IEEE TSC*, *IEEE TNSM*, *IEEE ICWS*, *ICSOC*, *IEEE SCC*, *AAAI*, *IJWSR*, *IJWGS*, etc.

Chunhua Zeng is currently a Master's student in the School of Computer Engineering and Science, Shanghai University, China. Prior to that, he received a Bachelor's in Software Engineering from East China Jiaotong University, China, 2021. His research interests include Mashup service recommendation, deep learning and intelligent algorithms. He has led a research and development team to successfully design and implement a service-oriented enterprise application big data platform, which can intelligently classify recycling, cultivate citizens' habit of discarding recyclable items, and generate significant economic and social benefits by providing personalised recommendation algorithms.

Yue Zhu is currently a Master's student in the School of Computer Engineering and Science, Shanghai University, China. Before that, she received a Bachelor's in Information Management and Information System at Shanghai University, China, 2020. Her research interests include Mashup service recommendation and deep learning. She has participated in a research and development group to successfully design a self-adaptive semantics-based framework for web service recommendation for Mashup creation. By combining natural language processing, optimisation algorithm and deep learning, the effectiveness of service recommendation has been improved and the users' requirements haven been met properly.

Pengtao Li is currently a Master's student in the School of Computer Engineering and Science, Shanghai University, China. Before that, he received a Bachelor's in Computer Science and Technology at Shanghai University, China, 2022. His research interests include Web API recommendation, natural language processing and deep learning. He led the research and development team to successfully design and implement a service-oriented anti-tracing cross-border network application platform. By designing a personalised recommendation algorithm, they helped users select and combine various anti-tracing networks, resulting in significant economic and social benefits.

Song Yang is currently a PhD candidate in the School of Computer Engineering and Science, Shanghai University, China. Before that, he received a Bachelor's degree in 2019 and Master's degree in 2022 both in Computer Science and Technology at Shanghai University, respectively. He has been an engineer at ByteDance from 2022 to 2023. His research interests include long-tail service classification, deep neural network and natural language processing. He has published two research papers on *IEEE Transactions on Network and Service Management* and *International Journal of Computational Science and Engineering*. He was responsible for designing an intelligent service platform, which can classify different kinds of recyclables.

Shengxiang Hu is currently a PhD candidate in the School of Computer Engineering and Science at Shanghai University, China. Prior to his ongoing PhD work, he successfully completed his Master's in Computer Science and Technology from the same university in 2021. His primary areas of research encompass quality of service (QoS) prediction, graph neural networks, and natural language processing. Over the course of his academic career, Hu has contributed significantly to the field through his authorship and co-authorship of 15 scholarly papers. These papers have been published in esteemed international journals and presented at prestigious conferences, such as *Knowledge-based Systems*, *IEEE Transactions on Service Computing*, *ICSOC*, *PPSN*, etc.

1 Introduction

With the rapid development of the internet, Web API has become an important entrance for global information transmission and sharing. They are software interfaces that support communication between networks, enabling different applications to exchange data and functionality. These APIs provide developers with standardised ways to leverage the functionality of other applications or services without knowing internal implementation details. In order to make full use of existing Web APIs and avoid developers from repeatedly implementing the same functional requirements in different applications, by creating Mashup services, developers can build new applications by combining different Web APIs without developing all functionality from scratch. To support the construction of Mashups, many service repositories continuously collect various Web APIs and previous Mashups. From these repositories, developers can choose the Web API that suits their needs to aggregate into new applications. However, due to the large number of existing Web APIs, it is often difficult for developers to spend a lot of manpower to find the most suitable Web API to meet their development needs. Therefore, Web API recommendations serve as an effective solution to help developers find their desired Web API for Mashup creation more easily.

However, with the rapid increase in the number of Web APIs, Mashup developers are facing increasing challenges in choosing appropriate Web APIs to build Mashup services. According to surveys, as of the end of 2023, there were 22,642 Web APIs registered on ProgrammableWeb, the largest service sharing platform, covering 499 categories and spanning many fields such as finance, maps, shopping, and weather. However, only a small part of the Web API is used by the Mashup service, and most of the Web API is not used. Especially for Web APIs with similar functions, Mashup services tend to use more popular Web APIs. Therefore, in the face of the explosive growth in the number of Web APIs, how to accurately provide Mashup developers with Web APIs that match their functions has become a hot issue in the field of service computing. The Web API recommendation algorithm has made a significant contribution to solving this problem. It can personalisedly recommend appropriate Web API services to users based on the demand description information provided by developers and the interaction records of the service.

In recent years, different Web API recommendation approaches have been proposed by QoS-aware, collaborative filtering, content-based, and deep neural network. QoS-aware (Liang et al., 2024) service recommendation is designed to help users find services that meet their QoS requirements among a list of functionally equivalent candidate services. The investigations based on collaborative filtering mainly rely on historical invocations to predict Web APIs, where there may be data sparsity problems as the number of Web APIs and Mashup services exponentially becomes larger. To alleviate the sparsity of historical invocations, content-based (Li et al., 2017) Web API recommendation approaches primarily focus on analysing the description feature of Web APIs and Mashups, where text mining and machine learning techniques, such as using topic models (Zhong et al., 2018) to reconstruct service descriptions in an attempt to remove redundant information, link prediction (Bianchini et al., 2017) and text clustering, are applied to extract and match the content features between API and Mashup services. Moreover, deep neural network (Liu et al., 2023) have been leveraged to learn the complex functionality relationships as well as capture non-functional features between APIs and Mashup services, leading to more accurate Web API recommendation.

Although these approachs show certain advantages in terms of Web API recommendation performance, there are still two main problems that need to be resolved: firstly, there are a large number of functionally irrelevant statements in the service descriptions, and existing approachs do not consider removing these redundant descriptions, directly using the original descriptions as a source of information, which limits the full exploration of service feature and affects the improvement of Web API recommendation performance. Secondly, the service has multiple information such as description and category. The existing approachs can not make full use of these information to enhance the service feature together, and only extract the service feature from the unit information, resulting in the limited information contained in the service feature, which limits the improvement of Web API recommendation performance.

To address the above two issues, we propose a Web API recommendation method based on the aggregation of multiple service feature, abbreviated as MFA. it employs the attention mechanism to extract feature from texts that are similar to descriptions. These features are then integrated into service's semantic feature, effectively addressing the problem of insufficient feature information due to poor service description quality. Simultaneously, relationship between services is supplemented through a category prediction task, a rich graph network is constructed in conjunction with service interaction records. Graph convolutional network is utilised to mine features from descriptions and categories. The acquired multi-dimensional features are aggregated through hierarchical feature aggregation algorithm, resulting in information-rich and diverse Mashup service features. This significantly enhances the effectiveness of Web API recommendation.

To demonstrate the effectiveness of MFA, extensive experiments are conducted on a large-scale real-world dataset that has been crawled from ProgrammableWeb, which includes 8484 Mashup services and 22,642 Web APIs. By comparing with several state-of-the-art baselines, the experimental results validate the superiority of MFA on multiple evaluation metrics.

The main contributions of this paper are summarised as follows:

- We propose a novel framework to improve the effectiveness of Web API recommendation, integrating service features and similar text features through a neural network model based on the attention mechanism to enrich the functional feature information of the service. Addresses an issue where service descriptions of varying quality result in insufficient information contained in service feature.
- We use the category of services as the basis for building graph networks, which greatly enriches the relationships between services. The hierarchical feature aggregation algorithm of the graph convolution network is used to further extract the functional features of the Mashup service and improve the accuracy of the Mashup service feature.
- To validate the performance of MFA, we conducted extensive experiments on a real-world dataset. The experimental results show that MFA achieve superior performance of Web API recommendation over competing baselines.

The remainder of this paper is organised as follows. Section 2 defines and formulates API service recommendation problem. Section 3 illustrates the overall proposed framework. Section 4 presents the approach in detail. Section 5 shows and analyses the experimental results. Section 6 reviews the related work. Finally, Section 7 concludes the paper and discusses the future work.

2 Problem formulation

In this section, we first give the formal expression of the Mashup service and Web API service in this article, which leads to the relevant definition of the Web API recommendation problem.

Definition 1 (service category): Given a web service library S. $C = \{c_1, c_2, ..., c_n\}$ are all service categories in the web service library S, where each category c is represented by the word w_c.

Definition 2 (Web API): The web service library S comprises p Web API services, denoted as $S_A = \{a_1, a_2, ..., a_p\}$, where each Web API is represented by $a =< D_a, C_a >$. $D_a = \{w_1, w_2, ...\}$ stands for the description of Web API service, elucidating the functional intricacies of the service, constructed from several words, with each word denoted by w_i; $C_a \subseteq C$ represents the service categories to which a belongs.

Figure 1 The framework of MFA

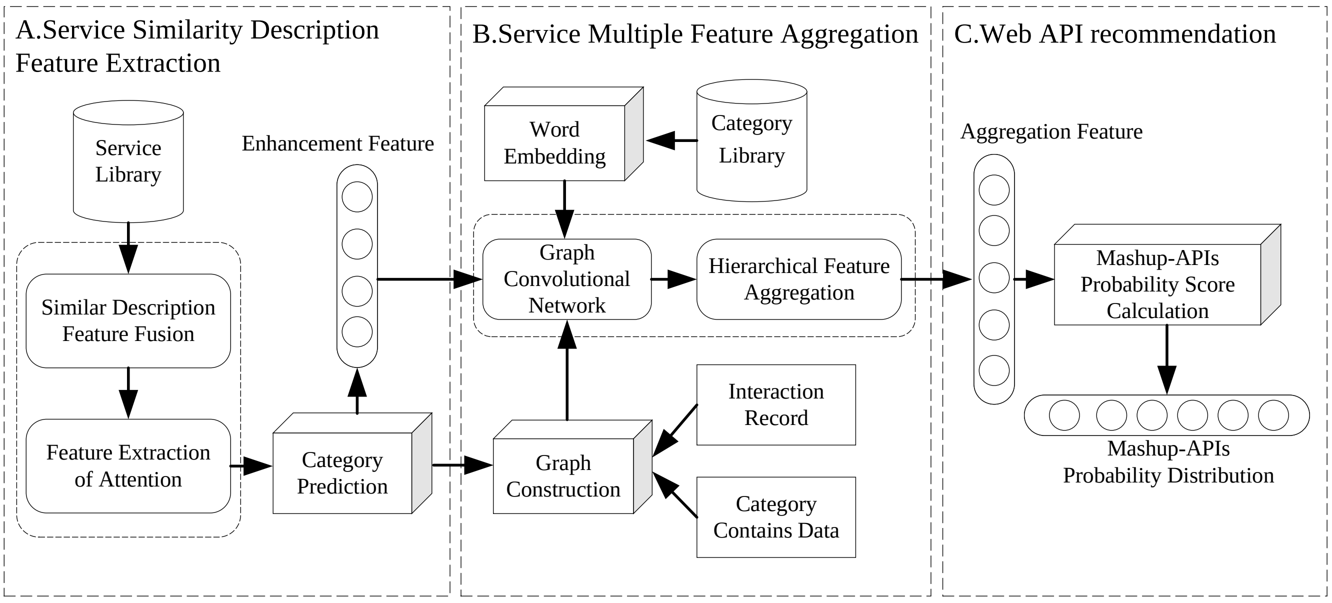
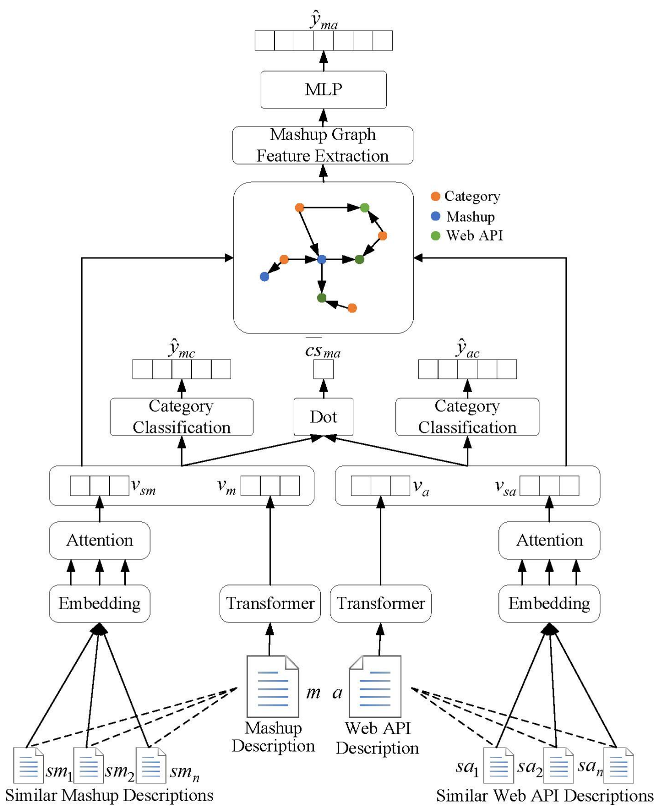

A.Service Similarity Description Feature Extraction
Service Library
Enhancement Feature
Similar Description Feature Fusion
Feature Extraction of Attention
Category Prediction
B.Service Multiple Feature Aggregation
Word Embedding
Category Library
Graph Convolutional Network
Hierarchical Feature Aggregation
Graph Construction
Interaction Record
Category Contains Data
C.Web API recommendation
Aggregation Feature
Mashup-APIs Probability Score Calculation
Mashup-APIs Probability Distribution

Figure 2 The neural network model of MFA (see online version for colours)



Definition 3 (Mashup): The web service repository S comprises q Mashup services, denoted as $S_M = \{m_1, m_2, ..., m_q\}$, where each Mashup is represented by $m = \{D_m, C_m, T_m\}$. $D_m = \{w_1, w_2, ...\}$ stands for the description of a Mashup service, capturing its functional requirements through a series of words denoted as w_i; $C_m \subseteq C$ signifies the service categories to which m belongs. $T_m \subseteq S_A$ represents the collection of Web API services invoked by m.

Definition 4 (Web API recommendation problem): Given a web service repository S, the Web API recommendation conundrum is delineated as $\Omega =< C, S_A, S_M, R >$, where S_A denotes a cohort of Web API services, S_M denotes a suite of Mashup services, and R represents a novel requirement for a Mashup. The resolution to the Web API recommendation poser Ω may be embodied by a set of Web API services $R_A \subseteq S_A$, wherein each Web API within R_A impeccably aligns with the functional requisites of the Mashup R.

3 The framework of MFA

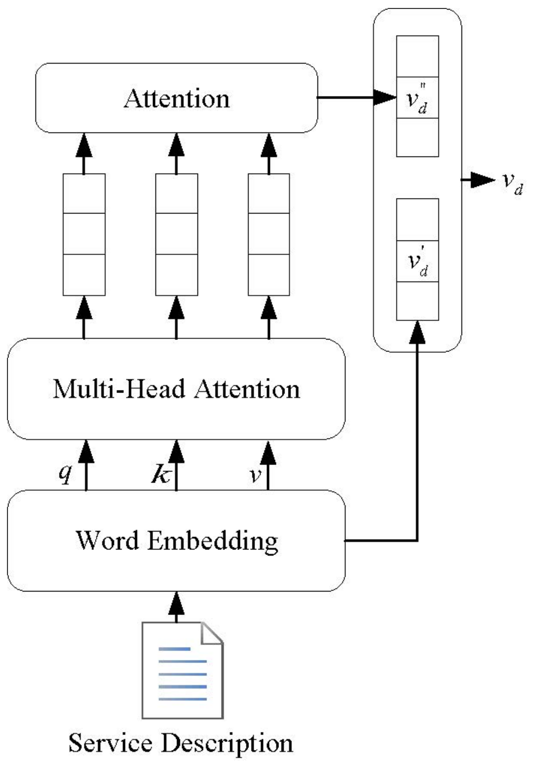
Figure 1 illustrates the overall framework of MFA, a methodology predicated on the learning of multi-element service features. This approach employs an attention mechanism to unearth service semantic features, and uses cosine similarity to locate analogous descriptions of services, thereby enriching the semantic information of the service. This process ensures that the service features pay greater attention to the latent functional features of similar services, thereby enhancing the service feature. By treating service category prediction as a preliminary task, the relationships between services are enriched. A graph network is constructed using the historical invocation of services and category inclusion data. The graph convolution network is then used to mine the Web API service features and service category features from the graph network. Utilising the hierarchical feature aggregation algorithm, these features are integrated into the Mashup service features, resulting in more accurate Mashup service features and enabling precise recommendations of Web API services. The MFA consists of three parts: the service similar description feature extraction module, the service multiple feature aggregation module, and the Web API recommendation module.

- In the service similarity description feature extraction module, it uses an attention mechanism-based neural network model to extract potential features from service descriptions. It also uses this model to extract features from similar description texts and combines them with original service features, emphasising similar functions to enhance semantic features. The predicted service category is used as the optimisation goal, and each service's features are used as input for the hierarchical feature aggregation of the graph convolution network.
- In the service multiple feature aggregation module, it uses service categories, Mashup services, and Web API services to create a graph network from historical and category inclusion data. The Jaccard similarity of service categories is used as edge weight to show node relationship closeness. Deep features of nodes are extracted via a graph convolution network. The hierarchical feature aggregation algorithm combines related Web API service features and service category features into Mashup service features for more accurate potential feature extraction.
- In the Web API recommendation module, it inputs the feature of the Mashup service obtained above into the multi-classifier of the fully connected layer and normalisation operation to obtain the Web API service prediction score, which is used as a basis to make effective Web API recommendation.

4 Approach

Figure 2 shows the Web API recommendation deep neural network model based on service multiple feature aggregation. it unfolds in two distinct stages. Firstly, the model capitalises on the service description and similar descriptions as the foundational data. It extracts semantic features via an attention mechanism and the transformer (Han et al., 2021) separately, and subsequently integrates these to form enhanced service features. Then, it employs three tasks to holistically optimise the service feature: prediction of the Web API category, prediction of the Mashup category, and prediction of click scores between Mashup and Web API. Secondly, it uses categories and service interaction records to construct a graph. It then extracts the latent feature of each node through the graph convolution network (Yu et al., 2023), and integrates the features of the Web API service, category, and Mashup service. These multiple features are hierarchically aggregated, culminating in the derivation of deep features for the Mashup service. Given that the recommendation task involves computing the probability distribution of the Mashup service across all Web API services, a binary cross-entropy loss function is utilised for multi-label classification during the model's training phase.

Figure 3 The improved transformer



4.1 Similar description feature extraction

Since there is a large amount of useless information in the service description, satisfactory results cannot be achieved if used directly for feature mining. Therefore, in order to fully obtain the feature of the service, consider mining relevant feature from description texts similar to the service description to supplement feature of the service. It contains three parts: service feature extraction by attention mechanism. This part uses the attention mechanism model to extract potential feature contained in service descriptions and similar description texts, so that the obtained feature pay more attention to the functions of the original service. Service description feature are enhanced. This part is to fuse service semantic features and similar service semantic feature to solve the problem of incomplete feature information due to the existence of useless descriptions in the original service description feature. Service category prediction. This part calculates the distribution probability of services for all Web API categories, with the optimisation goal of predicting the categories of services, and then enriches the service relationship graph constructed subsequently.

4.1.1 Service feature extraction by attention mechanism

A variety of complex information is implicit in the service description. In order to avoid insufficient information acquisition in the service due to a simple model, Transformer is used to mine the functional feature contained in the description. It uses multi-head attention and fuses the original word embedding vector with deep feature to ensure the generalisation of the model (Ansuini et al., 2019). Figure 3 shows the improved Transformer model. First, the vector $v_d^{'} = [e_1, e_2, ...]$ is obtained by performing a word embedding operation on the service description $D_d = \{w_1, w_2, ...\}$. Then, a multi-head attention network is used to extract the close layers between each word, which can focus on different parts of the input sequence in parallel, thereby better capturing the relationship between each word. This helps the model more fully understand the semantic and contextual information in the service description. The feature vector $head_i^k$ of the i^{th} word learned by the k^{th} attention head is expressed as follows:

$$Q_i^k = e_i W_Q^k, K_i^k = e_i W_K^k, V_i^k = e_i W_V^k \tag{1}$$

$$Attention(Q_i^k, K_i^k) = softmax\left(\frac{Q_i^k (K_i^k)^T}{\sqrt{d_k}}\right) \tag{2}$$

$$head_i^k = Attention(Q_i^k, K_i^k) V_i^k \tag{3}$$

where e_i is the vector of the i^{th} word, W_Q^k, W_K^k and W_V^k are the linear transformation matrices of the k^{th} attention head respectively. d_k is the dimension of the query vector. $Attention(Q_i^k, K_i^k)$ represents the relative importance of word w_i in the entire description. The final feature vector h_i of the i^{th} word is concated from the features learned by all attention heads. The formula is as follows:

$$h_i = [head_i^1; head_i^2; ...; head_i^{\mathcal{H}}] \tag{4}$$

where $\mathcal{H}$ is the number of attention heads, and then uses the word attention network to model the relative importance between different words, and aggregates them into the deep feature vector of the service description $v_d^{''}$, which is expressed as follows:

$$\gamma_i^w = \frac{\exp(q_w^T tanh(U_w h_i + u_w))}{\sum_{j=1}^{\mathcal{H}} \exp(q_w^T tanh(U_w h_j + u_w))} \tag{5}$$

$$v_d^{''} = \sum_{i=1}^{\mathcal{H}} \gamma_i^w h_i \tag{6}$$

where q_w^T, U_w and u_w are trainable parameters in the word attention network. Finally, in order to improve the generalisation ability of the model and avoid losing information, the word embedding vector $v_d^{'}$ of the service description and the deep feature vector $v_d^{''}$ are concated, thus forming the final representation of the service description $v_d = [v_d^{'}; v_d^{''}]$.

4.1.2 Service description feature enhancement

Given that the service description may contain extraneous information, relying solely on these features cannot fully encapsulate the service functions. Therefore, we propose to refine and augment the service features by considering similar descriptions of the service. Initially, we treat all service descriptions as a document collection. Using the BM25 algorithm, we obtain a set of similar descriptions for each service description and select η similar descriptions. These are denoted as $sd = [sd_1, sd_2, ..., sd_\eta]$, where sd_i represents one of the similar descriptions. Subsequently, these similar descriptions are transformed into continuous word vectors, which can be represented as follows:

$$E_{sd} = [ED_1; ED_2; ...; ED_\eta] \tag{7}$$

Recognising that different similar descriptions carry varying degrees of importance for services, we employ an attention mechanism to aggregate the vectors of similar service descriptions. Analogously, the attention weight γ_i^{sd} of a similar description sd_i is calculated using the following formula:

$$\gamma_i^{sd} = \frac{\exp(q_n^T tanh(U_n ED_i + u_n))}{\sum_{q=1}^{\eta} \exp(q_n^T tanh(U_n ED_q + u_n))} \tag{8}$$

where q_n^T, U_n and u_n are trainable parameters that similarly describe the attention network. The final similarity description semantic representation v_{sd} is calculated as $v_{sd} = \sum_{i=1}^{\eta} \gamma_i^{sd}$. After deep feature mining of the original description and similar description, the respective feature expressions v_d and v_{sd} are obtained respectively. v_d and v_{sd} are concated to obtain the enhanced feature of the service $v = [v_d; v_{sd}]$.

4.1.3 Service category prediction

Having enhanced the features of services using similar description features, the Mashup service description and Web API service description have now formed their respective feature expressions, denoted as v_m and v_a. In the realm of recommendation systems, the cosine similarity of two vectors is often calculated to measure the degree of functional matching of a service (Benard Magara et al., 2018). In order to bring the vectors of the Web API service and the Mashup service with similar functions closer together, thereby ensuring that the feature expressions of the Mashup service and the Web API service with matching functions are similar, we can employ cosine similarity. First, the vectors v_m and v_a are normalised, and then the cosine similarity of the two vectors is calculated, as outlined below:

$$\overline{cs}_{ma} = \frac{v_m v_a}{||v_m|| \, ||v_a||} \tag{9}$$

where $\overline{cs}_{ma}$ represents the cosine similarity between the Mashup service and the Web API service. The value range is between –1 and 1. The closer this value is to 1, the closer the characteristic expressions of the Web API service and the Mashup service are to each other.

To refine the parameters of our model and ensure the proximity of the learned service vector, we employ the mean variance as our loss function. This approach facilitates the model's learning and optimisation processes. Here is the corresponding formulation:

$$\mathcal{L}_{cs} = -\frac{1}{\mathcal{F}} \sum_{i=1}^{\mathcal{F}} (cs_{ma} \log(\overline{cs}_{ma}) + (1 - cs_{ma}) \log(1 - \overline{cs}_{ma}))$$

$$cs_{ma} = \begin{cases} -1, & \text{service } m \text{ has not used service } a \\ 1, & \text{service } m \text{ has used service } a \end{cases} \tag{10}$$

where $\mathcal{F}$ denotes the quantity of sample pairs comprising Web API service and Mashup service descriptions. The term $\overline{cs}_{ma}$ represents the computed cosine similarity of the service, while cs_{ma} signifies the actual degree of similarity. During the construction of the Web API service and Mashup service description pair, we perform negative sampling predicated on whether the Mashup service has utilised the Web API service. The ratio of positive to negative samples is maintained at 1:3.

Moreover, we augment the model training process with a service category prediction task, in addition to the existing similarity prediction task. This dual-task approach enables the model to learn from domain-specific information implicitly embedded within the signal. During the backpropagation phase, multi-task learning permits the shared hidden layer features, originally dedicated to a specific task, to be utilised by the other task. This results in implicit data augmentation and co-regularisation, significantly enhancing the training efficiency of deep neural networks (Ruder, 2017). Concurrently, during the subsequent hierarchical feature extraction phase of the graph convolution network, the service category prediction task enriches the relationships among services. This enhancement renders the constructed relationship graph more robust, thereby facilitating the extraction of multi-dimensional service features.

The service category prediction task is reframed as a multi-label classification problem, with each category within the category library considered a distinct label. The objective is to train a classifier capable of determining the probability distribution of each service across all categories. Consequently, binary cross-entropy loss is employed as the loss function. The following presents the calculation formula:

$$\hat{y}_{mc} = \sigma(W_{mc}v_m + b_{mc}), \hat{y}_{ac} = \sigma(W_{ac}v_a + b_{ac}) \tag{11}$$

$$\mathcal{L}_{mc} = -\frac{1}{|C|}\sum_{c\in C}^{C}(y_{mc}\log(\hat{y}_{mc}) + (1-y_{mc})\log(1-\hat{y}_{mc})) \tag{12}$$

$$\mathcal{L}_{ac} = -\frac{1}{|C|}\sum_{c\in C}^{C}(y_{ac}\log(\hat{y}_{ac}) + (1-y_{ac})\log(1-\hat{y}_{ac})) \tag{13}$$

where W_{mc}, b_{mc}, W_{ac}, and b_{ac} represent the weight matrix and bias vector in the category prediction task layer for the Mashup service and Web API service, respectively. $\hat{y}_{mc}$ and $\hat{y}_{ac}$ denote the corresponding category probability prediction values, whereas y_{mc} and y_{ac} are the true labels for their corresponding categories. If a service belongs to category c, then $y_{mc} = 1$ and $y_{ac} = 1$. Otherwise, $y_{mc} = 0$ and $y_{ac} = 0$. $\mathcal{L}_{mc}$ signifies the category prediction loss for the Mashup service, and $\mathcal{L}_{ac}$ denotes the category prediction loss for the Web API service. Considering all these factors, the overall loss of the model during the feature extraction phase of the attention mechanism comprises $\mathcal{L}_{cs}$, $\mathcal{L}_{mc}$, and $\mathcal{L}_{ac}$. The formula is as follows:

$$\mathcal{L}_1 = \mathcal{L}_{cs} + \mathcal{L}_{mc} + \mathcal{L}_{ac} + \mu\|\phi\|_2^2 \tag{14}$$

where $\mathcal{L}_1$ denotes the training loss of the neural network model during the service feature extraction phase of the service attention mechanism, with ϕ representing its corresponding model parameter.

4.2 *Service multiple feature aggregation*

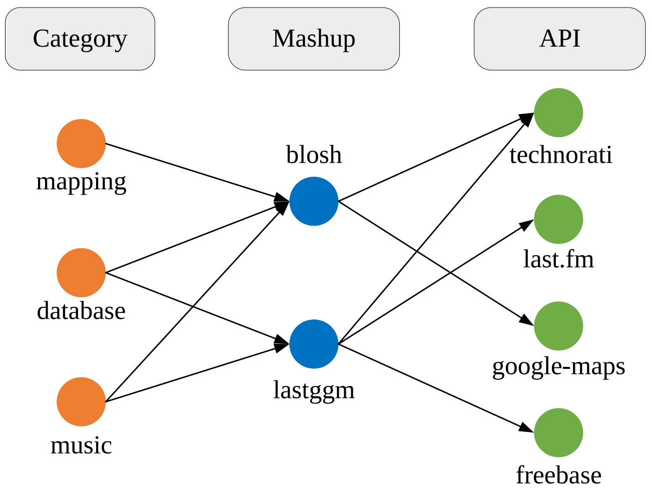
Through the feature mining of service descriptions by the above-mentioned deep neural network model based on the attention mechanism, a relatively sufficient semantic feature vector has been obtained. However, the semantic information of the service alone cannot further improve the recommendation performance. Therefore, the category information of the service is introduced, and the hierarchical feature aggregation algorithm of the graph convolution network is used to mine and aggregate category features. It can alleviate the cold start problem. On the other hand, it makes the service feature contain more information to improve the performance of Web API recommendation. It includes two parts: construction of service and category graph network. This part is based on the Mashup service using Web API service historical data and category containing service data to build a service category graph network, which is used to represent the structural relationship between services. Hierarchical feature aggregation algorithm of graph convolutional network. This part integrates the multiple feature of services from the graph network constructed by categories, Web API services and Mashup services as nodes to promote service semantic integrity.

4.2.1 *Construction of service and category graph network*

To effectively extract the diverse features of both Mashup and Web API services, and concurrently address the cold start problem inherent to recommendation systems, we construct a directed graph network using historical interaction data from the dataset service and category-inclusive service data.

Upon analysing the dataset, we identified three entities: the Mashup service, the Web API service, and the category. Typically, graph network recommendation approaches only consider Mashup services and Web API services as basic nodes. However, in practical applications, the historical records of Mashup services utilising Web API services are relatively sparse. This scarcity hampers significant improvements in recommendation performance. The Web API service maintains a compositional relationship with the Mashup service. From certain perspectives, the Web API service can reflect a portion of the Mashup service's functionalities and can supplement and integrate additional Mashup functional information. Meanwhile, the category serves as a service archive and reference summary. Simultaneously, categories can be considered an additional attribute of services, ensuring compatibility between services to some extent and enriching the inter-service relationships (Hu et al., 2020). Therefore, when constructing the relationship graph, we consider the Web API services, Mashup services, and categories as nodes.

Figure 4 Part of the structure of graph network (see online version for colours)



Once the nodes of the graph have been established, the edges can be determined accordingly. With the three types of nodes mentioned above, we can divide the edge set into two parts. One part is the relationship set of service interactions, denoted as $\mathcal{C}$, and the other part is the relationship set of categories containing services, denoted as $\mathcal{R}$. Each Mashup service has employed several Web API services. Hence, if Mashup service m has used Web API service a, a directed edge from service m to service a will be added to $\mathcal{C}$. This relationship can be expressed as $\{m, r_{ma}, a, w_{ma} \mid m \in S_M, a \in S_A\}$, where m and a represent Mashup service and Web API service, respectively. r_{ma} signifies that service m has used service a, and $w_{ma} = \frac{|C_m \cap C_a|}{|C_m \cup C_a|}$ represents the weight of this edge, calculated using the Jaccard coefficient through the set of categories owned by the service (Bag et al., 2019).

Each service has its own category set. When service s belongs to a certain category c, a directed edge from category c to service s is added to $\mathcal{R}$. This relationship can be represented as $\{c, r_{cs}, s, w_{cs} \mid c \in C, s \in S\}$, where c and s represent categories and services, respectively. r_{cs} indicates that service s belongs to category c, and w_{cs} symbolises the weight of the edge, represented by $\frac{1}{|C_s|}$, which corresponds to the proportion of categories to which the service pertains.

Figure 4 illustrates a selection of nodes and relationships within the network graph. It is apparent that the graph network comprises three types of nodes: Mashup service, Web API service, and category. The edge from the category pointing towards the service primarily represents the category to which the service belongs. Conversely, the edge from the Mashup service pointing to the Web API service indicates the usage record of the Web API service by the Mashup service. Consequently, the graph network can be represented as $G = \{V, E\}$, where the node set $V = S_M \cup S_A \cup C$ and the edge set is $E = \mathcal{C} \cup \mathcal{R}$.

4.2.2 Hierarchical feature aggregation algorithm of graph convolutional network

The semantic features of both Mashup and Web API services have been obtained through the attention mechanism's service description feature extraction. Simultaneously, the category of the new Mashup service has been predicted. By selecting $\mathcal{Q}$ categories as related categories of the service, we significantly enrich the relationships between services. Subsequently, a graph convolution network is employed to further extract complex features from each node in the graph. The Web API service features and category features associated with the Mashup service are aggregated into the Mashup service via a hierarchical feature aggregation algorithm. This approach enables us to obtain a more comprehensive and accurate feature representation of the Mashup service.

In the construction of the graph network, categories are incorporated as nodes. The feature vector v_c of each category is obtained using a word embedding network, which is dimensionally consistent with the feature vector of the service description. Subsequently, we define the adjacency matrix $A^{|V|\times|V|}$, which represents the connection relationship between nodes in the graph, based on the edge set. Initially, this matrix is set to zero for all elements. If a directed edge exists between node x and node y, the corresponding element A_{xy} in the adjacency matrix is set to 1. The node feature matrix X is then constructed, which consists of the concatenated feature vectors of all services and categories. The ith row of this matrix corresponds to the feature vector of the ith node. With this setup, the node features are updated through the graph convolution network as follows:

$$H^{(l+1)} = \sigma(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}) \tag{15}$$

where $H^{(l)}$ is the node representation matrix of the l^{th} layer, where $H^{(0)} = X$, $\hat{A} = A + I$ is the adjacency matrix A plus the self-connection matrix I, ensuring that each node takes its own feature into account, $\hat{D}$ is the diagonal degree matrix of $\hat{A}$, and $W^{(l)}$ is the weight matrix of the l^{th} layer.

Through the feature mining and aggregation of nodes through the graph convolution network, the features from the perspective of categories, Web API services and Mashup services have been further improved. In order to obtain a better representation of the Mashup service, the categories involved in the Mashup service and the information from the Web API service perspective are used to obtain the final multi-aggregation feature f_m. Specifically, the category feature of the upstream Mashup service are used to summarise it, and the Web API service feature downstream of the Mashup service are used to extend it. Its formal expression is as follows:

$$f_m = \frac{1}{|T_m|}\sum_{a\in T_m} w_{ma} g_{out}(concat(H_m^{l+1}, H_a^{l+1})) + \frac{1}{|C_m|}\sum_{c\in C_m} w_{mc} g_{in}(concat(H_m^{l+1}, H_c^{l+1})) \quad (16)$$

where concat represents the concatenation operation, w_{ma} and w_{mc} are the weights corresponding to the edges. $H_m^{(l+1)}$, $H_a^{(l+1)}$, and $H_c^{(l+1)}$ are the Mashup service feature, Web API service feature, and category feature obtained through GNN, respectively. g_{in} and g_{out} are two fully connected networks that map vectors to a specified dimension.

After hierarchical feature aggregation of the graph convolution network, the final feature expression f_m of the Mashup service is obtained. For the Web API recommendation task, it is necessary to calculate the probability score of use between the Mashup service and each Web API service, and make service recommendation based on this score. Therefore, in order to calculate the probability that the Mashup service uses the Web API service and use them to measure the Mashup service's preference for the Web API service, this paper uses a multi-label classification task to transform the Web API service usage probability calculation. The calculation formula is as follows:

$$\hat{s}_m = \sigma(mlp(f_m)) \quad (17)$$

where mlp is a plurality of fully connected layers, which maps the dimensions of f_m to the dimensions of the number of Web APIs. σ is a normalisation function, so that the value of each dimension is in the range of 0 to 1. The value of one dimension represents the usage probability of each Web API service.

In order to optimise the model parameters φ of the hierarchical feature aggregation algorithm module of graph convolution network, binary cross-entropy loss is used as the loss function, which calculates the difference between the predicted probability of each Web API being used and the real value, The formula is as follows:

$$\mathcal{L}_2 = -\frac{1}{|S_A|}\sum_{a\in S_A}(s_{m,a}\log(\hat{s}_{m,a}) + (1-s_{m,a})\log(1-\hat{s}_{m,a})) + \mu\|\varphi\|_2^2 \quad (18)$$

where S_A is the set of all Web API services, $s_{m,a}$ is the actual value of Mashup service m using Web API service a. If m has used a, then $s_{m,a} = 1$, otherwise $s_{m,a} = 0$. $\hat{s}_{m,a}$ is the predicted value of Mashup service m using Web API service a.

5 Experiments

5.1 Experimental setup and dataset

All experiments were conducted on our workstation, which is equipped with two NVIDIA GTX 4090 GPUs, two Intel(R) Xeon(R) Silver 4210R @2.40 GHz CPUs and 1T RAM. The designed models for Web API recommendation in the experiments are implemented by Python3.7.1 and Pytorch1.13.0.

In order to verify the effectiveness of MFA, we crawled web services from ProgrammableWeb. It contains 8,484 Mashup services and 22,642 Web API services. A Mashup service contains four metadata attributes, including name, description, category, and API interaction set; a Web API service contains three metadata attributes, including name, description, and category. In the experiments, we have deleted those Mashup services without source information and interaction, and finally it has 8,217 Mashup services, 1,648 API services, and 499 categories. Table 1 provides detailed statistical data on the PW dataset.

For the parameter setting of MFA, the word embedding dimension of the service description is 200. The service description feature enhancement module under the first-stage attention mechanism predicts the number of service categories selected by the service category Q is 3, and other models The training parameter settings are shown in Table 2.

5.2 Evaluation metrics

In the experiments, we apply cross-validation techniques to evaluate our proposed MFA and competing approaches. It consists of three experimental stages, including model training, validation, and performance testing. During model training and validation, we use a training set and verification set to evaluate the model convergence, and a test set to demonstrate the effectiveness MFA (Coates et al., 2011). After experimental verification, it was determined that setting the ratio to 8:1:1 results in the best recommendation performance for MFA.

To validate the effectiveness of MFA, four different evaluation metrics are used to measure recommendation performance among multiple competing approaches. The precision and recall ratio of the top K recommended Web APIs in the ranking are defined as:

$$Precision@K = \frac{1}{|M|}\sum_{m\in M}\frac{|rcmd(m)\cap real(m)|}{|rcmd(m)|} \quad (19)$$

$$Recall@K = \frac{1}{|M|}\sum_{m\in M}\frac{|rcmd(m)\cap real(m)|}{|real(m)|} \quad (20)$$

where M is the Mashup collection in the test set, and $|M|$ is the size of M. For each Mashup service $m \in M$, $rcmd(m)$ is the list of recommended Web APIs, and $real(m)$ is the ground truth of the included Web APIs.

The mean average precision (MAP) of the top K recommended Web APIs in the ranking is defined as:

$$MAP@K = \frac{1}{|M|}\sum_{m\in M}\frac{1}{N_m}\sum_{i=1}^{K}\left(\frac{N_i}{i}I_i\right) \quad (21)$$

where I_i indicates whether a Web API at position i in the ranking corresponds to the real Web API of m; N_m

indicates the real number of Web APIs in m; N_i represents the number of invocations received by the Web API among the top i API set in the ranking.

Table 1 Statistics of the experimental dataset

Statistics	*Value*
APIs	1648
Mashups	8217
Categories	499
Average Mashup per APIs	10.432
Average API per Mashups	2.091
Categories per API	2.887
Categories per Mashup	2.997

Table 2 Parameter settings of MFA

Parameter	*Value*
Epoch	50
Batch size	1024
Learning rate	0.01
D (word embedding dimension)	200
η (number of similar services)	5
$\mathcal{Q}$ (the predicted number of service categories)	3
$\mathcal{H}$ (number of attention heads of multi-head attention)	200

Additionally, the normalised discounted cumulative gain (NDCG) of the top K recommended Web APIs in the ranking is applied to evaluate the performance of MFA, which is defined as:

$$NDCG@K = \frac{1}{|M|} \sum_{m \in M} \frac{1}{S_m} \sum_{i=1}^{K} \frac{2^{I_i} - 1}{\log(1+i)} \tag{22}$$

where S_m represents the expected maximum DCG score of m.

5.3 Competing methods

To evaluate the performance of MFA, we compare it with eight competitive baselines, including two conventional approaches based on collaborative filtering, three content-based probability and statistics approaches, and three model-based deep neural network aprroaches. They are described as below.

- *CF (Cremonesi et al., 2012):* It is a widely used recommendation algorithm, which calculates the similarity of target Mashup services according to their historical invocation records (Cao et al., 2014), and recommends Web APIs involved in the most similar Mashup services to the target ones.
- *NCF (He et al., 2017b):* It utilises deep neural network models to capture nonlinear relationships between services instead of relying on traditional matrix factorisation. At the same time, by using the collaborative filtering algorithm to calculate the similarity between services, the accuracy and diversity of Web API recommendations can be improved.
- *AFUP (Jain et al., 2015):* It takes into account a range of factors, including feature information, historical interactions, and popularity. By aggregating these factors, the probability that the Mashup service calls the corresponding Web API is calculated, thereby recommending the Web API to create the required Mashup service.
- *SFTN (Samanta and Liu, 2017):* It is improved by their previous work AFUP (Jain et al., 2015) by using hierarchical dirichlet process and probability matrix factorisation that can effectively process content information and historic invocation records to perform better Web APIs recommendation.
- *SPR (Zhong et al., 2018):* It exploits topic model to extract the features of Web APIs from Mashup description, and uses the extracted features to reconstruct aggregated requirement descriptions that removes irrelevant information for satisfying personalised Web API recommendation.
- *RWR (Wang et al., 2019):* It is a recommendation approach based on knowledge graph, where each API is regarded as a node in the graph, and the relationship between two APIs is represented by edges. Then, it applies random walk algorithm to explore the modelled knowledge map that finds the neighbor information of nodes to recommend the appropriate Web APIs.
- *MTFM (Wu et al., 2022):* It takes Mashup service category prediction as an auxiliary task to enhance the ability of network feature extraction, and regards API prediction as a multiple classification problem, resulting in expected recommendation effectiveness.
- *GSR-REC (Fan et al., 2023):* It is a service recommendation approach that can automatically acquire services based on user requirements, where GSR employs reinforcement learning to learn the inter-dependencies among Web APIs and integrate them into service recommendation.

5.4 Experiment results and analyses

Tables 3 and 4 show the comparison between MFA and various baseline approachs. It can be seen that MFA has better recommendation performance than other baseline approachs in all scenarios where K is. It shows that MFA is better than other approachs in Web API recommendation.

For the precision, its value continues to decrease as K increases, but MFA is still 6.9% to 10.6% higher than other approachs, which intuitively shows that MFA has better service recommendation performance superiority. As for the recall, its value increases with the increase of K, but MFA is 1.4% to 4.9% higher than the optimal approach among the baseline approachs. This shows that MFA takes into account the Web API services that Mashup developers are interested in, and matches the Web API services that have the same functional requirements as the Mashup service as much as possible. Judging from the performance of MAP, its value slowly becomes smaller as K increases. However, MFA is still 0.3% to 1.4% higher than other approaches. This shows that the model is able to effectively distinguish between positive and negative examples across multiple categories and performs well for classification tasks across the entire dataset. Regarding the NDCG, its value is proportional to K. Compared with other baseline approaches, MFA outperforms this metric by 0.4% to 3.8%, This means that the recommended service list performs better in the ranking task, provides more relevant services, and these services are ranked higher in the recommended list.

Both CF and NCF use service history interaction records as an information source to obtain service feature. However, the service history interaction record data is relatively sparse, resulting in mediocre recommendation performance. AFUP and SFTN use a variety of information sources, including description information, historical interaction records and popularity, but are still constrained by the sparse historical interaction record, so the service recommendation performance has not been greatly improved. SPR, MTFM, and GSR-REC use neural network models to extract features from service description information, which has a certain improvement in recommendation performance compared to the approachs mentioned above. However, the uneven quality of service descriptions makes it impossible to effectively obtain service feature, making it impossible to further improve recommendation performance.

RWR considers using graph networks to mine relationship features between services. However, the relationship graph between services is relatively simple to construct and cannot fully express the relationships between services, causing recommendation performance to encounter bottlenecks. On the one hand, MFA uses similar descriptions of services to optimise and expand its descriptions, fully capturing its semantic features for the service category prediction task. On the other hand, it constructs a graph network with three types of nodes: categories, Mashup services, and Web API services, making the graph network more rich and diverse. More accurate multiple categories of features are obtained through the expanded graph network. The hierarchical feature aggregation algorithm aggregates service type features and Web API service features into Mashup service features, so that the service features contain a lot of heterogeneous feature information, thus making the Web API recommendation performance has been further improved.

5.5 *Performance impact of parameters*

The API recommendation performance of MFA predominantly hinges upon the quantity of attention heads within the multi-head attention, denoted as $\mathcal{H}$, and the projected count of service categories, symbolised as $\mathcal{Q}$.

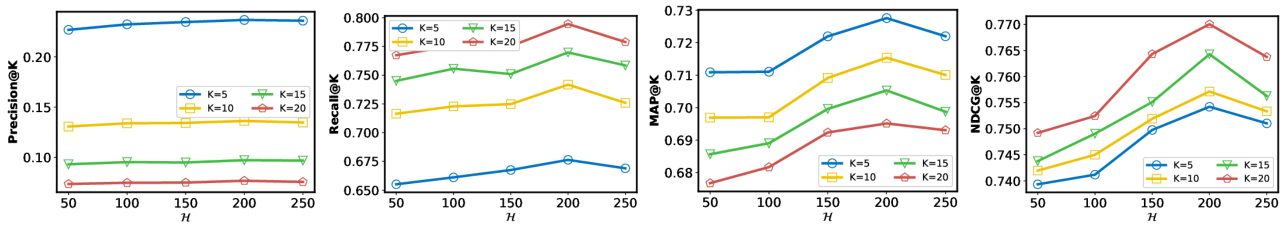
Table 3 Experimental results of different competing approaches under the settings of $K = 5$ and 10

Methods	*K = 5*				*K = 10*			
	Precision	*Recall*	*MAP*	*NDCG*	*Precision*	*Recall*	*MAP*	*NDCG*
CF	0.117	0.327	0.241	0.303	0.100	0.557	0.263	0.380
NCF	0.141	0.415	0.421	0.463	0.091	0.513	0.427	0.494
AFUP	0.142	0.421	0.423	0.466	0.092	0.522	0.429	0.497
SFTN	0.153	0.474	0.503	0.533	0.087	0.543	0.501	0.561
SPR	0.187	0.543	0.549	0.595	0.123	0.645	0.544	0.616
RWR	0.198	0.592	0.581	0.633	0.117	0.682	0.574	0.648
MTFM	0.219	0.602	0.676	0.705	0.121	0.664	0.662	0.710
GSR-REC	0.214	0.658	0.723	0.734	0.120	0.712	0.713	0.735
MFA	*0.237*	*0.676*	*0.728*	*0.757*	*0.136*	*0.742*	*0.715*	*0.763*
Gains	*8.2%*	*2.7%*	*0.7%*	*3.1%*	*10.6%*	*4.2%*	*0.3%*	*3.8%*

Table 4 Experimental results of different competing approaches under the settings of $K = 15$ and 20

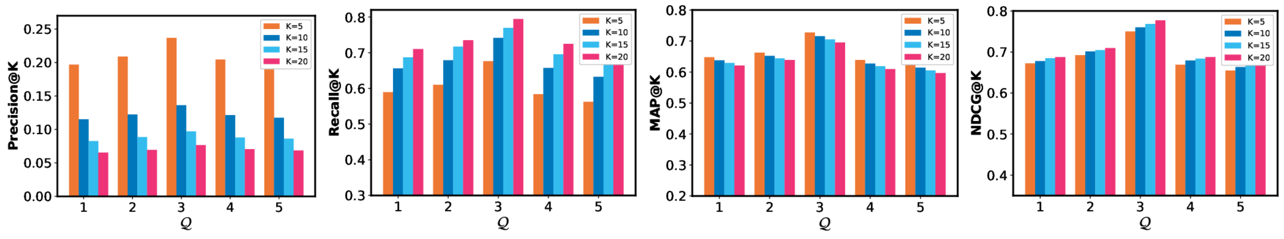
Methods	*K = 15*				*K = 20*			
	Precision	*Recall*	*MAP*	*NDCG*	*Precision*	*Recall*	*MAP*	*NDCG*
CF	0.076	0.619	0.261	0.392	0.061	0.650	0.259	0.396
NCF	0.068	0.562	0.423	0.502	0.055	0.590	0.421	0.506
AFUP	0.071	0.570	0.414	0.524	0.052	0.625	0.405	0.576
SFTN	0.082	0.611	0.489	0.573	0.070	0.653	0.478	0.598
SPR	0.090	0.716	0.530	0.621	0.072	0.784	0.518	0.625
RWR	0.088	0.729	0.567	0.653	0.070	0.768	0.555	0.652
MTFM	0.084	0.696	0.652	0.710	0.067	0.728	0.641	0.708
GSR-REC	0.089	0.734	0.706	0.747	0.071	0.766	0.700	0.757
MFA	*0.097*	*0.770*	*0.710*	*0.762*	*0.077*	*0.795*	*0.705*	*0.760*
Gains	*7.8%*	*4.9%*	*0.6%*	*2.0%*	*6.9%*	*1.4%*	*0.7%*	*0.4%*

Figure 5 Performance impact with the number of attention heads of multi-head attention, (a) precision (b) recall (c) MAP (d) NDCG (see online version for colours)



(a) (b) (c) (d)

Figure 6 Performance impact with the predicted number of service categories, (a) precision (b) recall (c) MAP (d) NDCG (see online version for colours)



(a) (b) (c) (d)

5.5.1 *Impact of number of attention heads of multi-head attention*

Figure 5 shows the line chart of evaluation metrics changing with $\mathcal{H}$ when the number of recommended Web API services is 5, 10, 15 and 20. $\mathcal{H}$ is a parameter of the neural network model, which mainly controls the model's ability to express service feature. As the number of attention heads of multi-head attention increases, the four evaluation metrics gradually increases. When the number of attention heads is 200, the values of the four evaluation metrics are the highest. After 200, the recommended performance begins to decline. This is because as the number of attention heads increases, the expressive ability of MFA gradually improves. More attention heads mean that the model can better capture different relationships and feature between services, thereby improving recommendation performance. The multi-head attention mechanism allows the model to focus on different parts of the input simultaneously, which helps to fuse information more comprehensively. In recommendation tasks, this may lead to more accurate and comprehensive Mashup service interest modelling, which is beneficial to improving recommendation performance. When the number of attention heads is 200, the values of the four evaluation metrics are the highest. This is because the model has been able to capture the complex relationships between services very well at this point, and continuing to increase the number of attention heads does not significantly improve model performance.

5.5.2 *Impact of predicted number of service categories*

Figure 6 shows the histogram of evaluation metrics changing with $\mathcal{Q}$ in scenarios where the number of recommended Web API services is 5, 10, 15 and 20. $\mathcal{Q}$ is a non-model parameter, similar to the number of Web API recommendation recommended by Web API. It controls the number of categories to which each service belongs. Since categories are introduced when building a graph network to enrich the relationship between services, Therefore, it can greatly affect the extraction of various heterogeneous features of services. It can be seen from the figure that when the number of predicted service categories $\mathcal{Q}$ increases from 1 to 3, the four evaluation metrics also increase, and when $\mathcal{Q}$ increases from 3 to 5, the four evaluation metrics become smaller and smaller. It can be analysed from the distribution of the dataset that each service belongs to three categories on average. Therefore, when the number of predicted service categories is less than 3, the accuracy of service category prediction is gradually improving, making the construction of graph networks more and more complex. It is close to reality, thus ensuring that the service multi-aggregation features obtained by the hierarchical feature aggregation algorithm of the graph convolution network are more accurate. The performance of service recommendation is also improved. When the number of predicted service categories exceeds 3, the accuracy of service category prediction decreases, there are more and more inaccurate edges in the graph network, and the Mashup service aggregates more irrelevant features, resulting in reduced recommendation performance. Therefore, setting the number of predicted service categories to 3 enables MFA to obtain the best Web API recommendation performance.

6 Related work

Recently, Web API recommendation has received much attention, existing approaches can be mainly divided into three categories, including collaborative filtering approaches, content-based approaches, and deep learning-based approaches.

6.1 CF-based Web API recommendation

Collaborative filtering technique has been widely applied in recommender systems that can be used to make Web API recommendation. Sang et al. (2023) propose a Bayesian matrix factorisation model with text similarity and adversarial training. By utilising natural language processing and collaborative filtering, the model effectively leverages contextual information and historical data, thereby improving recommendation accuracy. However, there is a sparsity issue in the interaction data between Mashup services and Web APIs, which may hinder the matrix factorisation model from accurately capturing implicit service features. Based on the neural graph collaborative filtering technique, Lian and Tang (2022) propose an API recommendation approach that exploits the high-order connectivity between Web APIs and their users. Ke et al. (2019) proposed a hybrid collaborative filtering and attention-based CNN model for web service recommendation, where the hybrid service invocation matrix and attention-based CNN are seamlessly integrated into a deep neural network for better capturing complex relationships among hybrid services.

While CF-based recommendation methodologies exhibit considerable adaptability in the realm of Web API recommendation, they are notably hindered by difficulties in extracting semantic information from Mashup services and Web APIs. This constraint underscores the necessity for future investigations to delve into hybrid recommendation systems. Such systems would amalgamate CF with content-based approaches or semantic technologies, thereby circumventing the limitations inherent in relying exclusively on CF for Web API recommendation. This proposed integrative approach promises to enhance the robustness and accuracy of Web API recommendations, opening new avenues for enriching user experiences.

6.2 Content-based Web API recommendation

This kind of recommendation approach necessitates the extraction of salient features through the comparison of similarities between Mashup services and Web APIs. These features typically encompass service descriptions, service tags, and the interrelationships between Mashup services and Web APIs. Text-matching-based approaches compute the correlation between Mashup requirements and Web API descriptions, generally employing TF-IDF statistical techniques to recommend pertinent Web APIs (He et al., 2017a). Nevertheless, these methods often fail to generate a satisfactory recommendation list, as they lack the capability to semantically extract the defining characteristics of both Mashup services and Web APIs.

In order to overcome the limitations of Web API recommendation based on text matching, semantic-based approaches have been investigated for Web API recommendation, which can be divided into two categories. One is based on knowledge mapping (Wang et al., 2019), which leverages entities, attributes, and relationships to link Mashup services with Web APIs. It builds a rich and accurate Mashup knowledge system, where logic reasoning is performed to calculate the semantic similarity between Mashup services and Web APIs. However, due to the lack of domain knowledge and the high cost of manual annotation, it is difficult to extend to large-scale datasets. The other is based on latent semantics, which usually exploits the topic model to extract semantic features, and measures the relevance between Mashup requirements and Web APIs by feature similarity. Shi et al. (2019) employed a probabilistic topic model to extend sentence-level service descriptions, aiming to optimise the original service descriptions. The primary objective is to eliminate redundant elements from the descriptions with the goal of enhancing the extraction of service features. However, content-based Web API recommendation approaches mainly extract shallow semantic information, which may lead to the loss of hidden feature among Mashup services and Web APIs, thus resulting in a decrease of recommendation performance.

6.3 Deep learning-based Web API recommendation

With the rapid development of deep learning, more recent investigations apply deep neural networks to learn complex interactions between Mashup services and Web APIs for better Web API recommendation. By combining multilayer perceptron and collaborative filtering algorithm, it can precisely learn the nonlinear relationships that can effectively recommend Web APIs. Ma et al. (2021) have introduced DNN for cold-start service recommendation by integrating multiple interactions between Mashup services and Web APIs and their content similarities.

In recent years, with the continuous developments of graph neural networks (Cai et al., 2023; Wang et al., 2023), they have been widely investigated and used to mine complex interactive features. Liu et al. (2023) have proposed a novel approach for creating a Mashup service by designing a service bundle recommendation model based on dynamic graph neural network to learn the interactions between Mashups and Web APIs. Yu et al. (2023) propose a web service recommendation approach called SRMG, which utilises Transformers and GraphGAN for intelligent Mashup creation recommendation by calculating functional similarities. The preferences of a new Mashup service are guided by its resemblance to existing ones, offering an effective strategy for Mashup creation. The model of graph convolution neural network (Hu et al., 2023) can be used to learn the relationship between Mashup services and Web APIs, and simultaneously attention mechanism can also be fused to more accurately calculate the similarity for promoting the performance of API recommendation.

However, Mashup developers or service vendors are not always able to provide accurate descriptions of aggregated demands or service functionalities. As a result, existing deep learning-based approaches still face challenges in effectively uncovering the latent features from service functionalities for Web API recommendation. To this end, we design a novel framework for Web API recommendation by Mashup description reduction and deep feature learning, which can optimise the representation of original Mashup services and extract the features from the reduced blocks and their corresponding positions.

7 Conclusions and future work

In this paper, we propose a new Web API recommendation framework called MFA. It uses an attention mechanism to extract and integrate features from similar service descriptions, addressing the issue of poor service description quality. It also predicts categories of service to reveal hidden service relationships, constructs a detailed graph network by interaction record of services, and uses graph convolutional networks for feature extraction. A hierarchical feature aggregation algorithm is used to combine these features, enhancing Web API recommendation performance and compatibility. The effectiveness of MFA was confirmed through experiments comparing it with other Web API recommendation methods on a large real dataset. The experimental results fully proved the superiority of MFA in Web API recommendation. At the same time, we apply this method to practical applications, which reduces developers' time and energy in finding and selecting APIs, thereby reducing development costs.

In the future, we plan to explore novel prompt engineering and investigate fine-tuning of LLMs with the consideration of embedding the invocation relationships among Mashup and Web APIs, further advancing the reduction of original Mashup description and hidden feature learning for better Web API recommendation.

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 62272290).

References

Ansuini, A., Laio, A., Macke, J.H. and Zoccolan, D. (2019) 'Intrinsic dimension of data representations in deep neural networks', in *Advances in Neural Information Processing Systems (NIPS)*, Vol. 32, pp.6111–6122.

Bag, S., Kumar, S.K. and Tiwari, M.K. (2019) 'An efficient recommendation generation using relevant Jaccard similarity', *Information Sciences*, Vol. 483, pp.53–64.

Benard Magara, M., Ojo, S.O. and Zuva, T. (2018) 'A comparative analysis of text similarity measures and algorithms in research paper recommender systems', in *Conference on Information Communications Technology and Society (ICTAS)*, pp.1–5.

Bianchini, D., De Antonellis, V. and Melchiori, M. (2017) 'WISeR: a multi-dimensional framework for searching and ranking Web APIs', *ACM Transactions on the Web*, Vol. 11, No. 3, pp.1–32.

Cai, J., Li, W., Li, X., Li, K., Gui, Z. and Khan, M.K. (2023) 'GTxChain: a secure IoT smart blockchain architecture based on graph neural network', *IEEE Internet of Things Journal*, Vol. 10, No. 24, pp.21502–21514.

Cao, B., Tang, M. and Xing, H. (2014) 'CSCF: a Mashup service recommendation approach based on content similarity and collaborative filtering', *International Journal of Grid and Distributed Computing*, Vol. 7, No. 2, pp.163–172.

Coates, A., Ng, A. and Lee, H. (2011) 'An analysis of single-layer networks in unsupervised feature learning', in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp.215–223.

Cremonesi, P., Picozzi, M. and Matera, M. (2012) 'A comparison of recommender systems for Mashup composition', in *International Workshop on Recommendation Systems for Software Engineering (RSSE)*, pp.54–58.

Fan, G., Chen, S., He, Q., Wu, H., Li, J., Xue, X. and Feng, Z. (2023) 'Service recommendations for Mashup based on generation model', *IEEE Transactions on Services Computing*, pp.1–14.

Han, K., Xiao, A., Wu, E., Guo, J., Xu, C. and Wang, Y. (2021) 'Transformer in transformer', in *Advances in Neural Information Processing Systems (NIPS)*, pp.15908–15919.

He, Q., Zhou, R., Zhang, X., Wang, Y., Ye, D., Chen, F. and Grundy, J.C. (2017a) 'Keyword search for building service-based systems', *IEEE Transactions on Software Engineering*, Vol. 43, No. 7, pp.658–674.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X. and Chua, T-S. (2017b) 'Neural collaborative filtering', in *Conference on World Wide Web (WWW)*, pp.173–182.

Hu, X., Lu, L. and Wu, H. (2020) 'A category aware non-negative matrix factorization approach for app permission recommendation', in *IEEE International Conference on Web Services (ICWS)*, pp.240–247.

Hu, N., Zhang, D., Xie, K., Liang, W., Li, K. and Zomaya, A. (2023) 'Multi-graph fusion based graph convolutional networks for traffic prediction', *Computer Communications*, Vol. 210, pp.194–204.

Jain, A., Liu, X. and Yu, Q. (2015) 'Aggregating functionality, use history, and popularity of APIs to recommend Mashup creation', in *International Conference on Service-Oriented Computing (ICSOC)*, pp.188–202.

Ke, J., Xu, J., Meng, X. and Huang, Q. (2019) 'Hybrid collaborative filtering with attention CNN for web service recommendation', in *2019 3rd International Conference on Data Science and Business Analytics (ICDSBA)*, pp.44–52.

Li, H., Liu, J., Cao, B., Tang, M., Liu, X. and Li, B. (2017) 'Integrating tag, topic, co-occurrence, and popularity to recommend Web APIs for Mashup creation', in *IEEE International Conference on Services Computing (SCC)*, pp.84–91.

Lian, S. and Tang, M. (2022) 'API recommendation for Mashup creation based on neural graph collaborative filtering', *Connection Science*, Vol. 34, No. 1, pp.124–138.

Liang, W., Li, Y., Xu, J., Qin, Z., Zhang, D. and Li, K. (2024) 'QoS prediction and adversarial attack protection for distributed services under DLaaS', *IEEE Transactions on Computers*, Vol. 73, No. 3, pp.669–682.

Liu, M., Tu, Z., Xu, H., Xu, X. and Wang, Z. (2023) 'DySR: a dynamic graph neural network based service bundle recommendation model for Mashup creation', *IEEE Transactions on Services Computing*, Vol. 15, No. 4, pp.2592–2605.

Ma, Y., Geng, X. and Wang, J. (2021) 'A deep neural network with multiplex interactions for cold-start service recommendation', *IEEE Transactions on Engineering Management*, Vol. 68, No. 1, pp.105–119.

Ruder, S. (2017) *An Overview of Multi-task Learning in Deep Neural Networks*, arXiv preprint arXiv:1706.05098.

Samanta, P. and Liu, X. (2017) 'Recommending services for new Mashups through service factors and top-K neighbors', in *IEEE International Conference on Web Services (ICWS)*, pp.381–388.

Sang, C., Deng, X. and Liao, S. (2023) 'Mashup-oriented web API recommendation via full-text semantic mining of developer requirements', *IEEE Transactions on Services Computing*, Vol. 16, No. 4, pp.2755–2768.

Shi, M., Tang, Y. and Liu, J. (2019) 'Functional and contextual attention-based LSTM for service recommendation in Mashup creation', *IEEE Transactions on Parallel and Distributed Systems*, Vol. 30, No. 5, pp.1077–1090.

Wang, X., Wang, D., Yu, D., Wu, R., Yang, Q. and Deng, S. (2023) 'Intent-aware graph neural network for point-of-interest embedding and recommendation', *Neurocomputing*, Vol. 557, p.126734.

Wang, X., Wu, H. and Hsu, C-H. (2019) 'Mashup-oriented API recommendation via random walk on knowledge graph', *IEEE Access*, Vol. 7, pp.7651–7662.

Wu, H., Duan, Y., Yue, K. and Zhang, L. (2022)' Mashup-oriented web API recommendation via multi-model fusion and multi-task learning', *IEEE Transactions on Services Computing*, Vol. 15, No. 6, pp.3330–3343.

Yu, T., Yu, D., Wang, D. and Hu, X. (2023) 'Web service recommendation for Mashup creation based on graph network', *The Journal of Supercomputing*, Vol. 79, No. 8, pp.8993–9020.

Zhong, Y., Fan, Y., Tan, W. and Zhang, J. (2018) 'Web service recommendation with reconstructed profile from Mashup descriptions', *IEEE Transactions on Automation Science and Engineering*, Vol. 15, No. 2, pp.468–478.