International Journal of Web Services Research
Volume 20 • Issue 1

A Service Recommendation Algorithm Based on Self-Attention Mechanism and DeepFM

Li Ping Deng, Shanxi Vocational University of Engineering Science and Technology, China & Taiyuan University of Technology, China

Bing Guo, Taiyuan Normal University, China

Wen Zheng, Taiyuan University of Technology, China*

https://orcid.org/0000-0002-6570-6245

ABSTRACT

This article proposes a recommendation model based on self-attention mechanism and DeepFM service, the model is SelfA-DeepFM. The method firstly constructs the service network with DTc-LDA model to mine the potential relationship between Mashup and API, which not only fully considers the text attributes but also combines the network structure information to effectively mitigate the sparsity of the service data. Secondly, service clustering to obtain numerical feature similarities. Finally, the self-attention mechanism is used to capture the different importance of feature interactions, and the DeepFM model is used to mine the complex interaction information between multidimensional features to predict and rank the quality score of API services to recommend suitable APIs. To verify the performance of the model, the authors use the real data crawled from the ProgrammableWeb platform to conduct multiple groups of experiments. The experimental results show that the model significantly improves the accuracy of service recommendation.

KEYWORDS

DeepFM, Self-Attention Mechanism, Service Recommendation, Web Services Network

INTRODUCTION

With the emergence of new software technologies such as cloud computing, mobile computing, and blockchain, APIs (Application Programming Interface) is playing an increasingly important role in software development (Liang et al.,2019). Driven by the API economy, many companies, such as Google, Amazon, and Microsoft, have released Web-based open APIs, often to allow third parties to access their critical resources in a programmable way, and as a result the number of open APIs on the Internet has grown rapidly (Lian et al.,2021). In fact, the number of API services is showing a trend of multiplication (Almarimi, N. et al.,2019), with as many as 20,373 API services in the

DOI: 10.4018/IJWSR.331691 *Corresponding Author

Programmable-Web website as of November 2019 (Deng et al., 2021). In this context, it has become an important problem in the field of service computing to quickly and effectively find API services to meet the Mashup needs of developer users from such a large-scale collection of services, and it is attempted to be solved by service recommendation (Kang et al.,2021; Jiang et al.,2021).

In the field of service computing, there have been many studies on recommending suitable APIs for Mashup creation, where collaborative filtering (CF) algorithms have played an important role (Cao et al.,2020). However, existing CF-based recommendation methods suffer from data sparsity and cold start, which lead to poor recommendation results. To alleviate the data sparsity problem, traditional recommendation methods often convert user attributes and item attributes into a generic feature vector and use supervised learning to predict recommendation scores, such as FM (Rendle et al.,2011), Wide& Deep (Cheng et al.,2016), DeepFM (Guo et al.,2017), xDeepFM (Lian et al.,2018), etc. Despite the superiority of deep learning-based FM, these approaches ignore the relationship between Web service structural properties (e.g., Mashup-API, Mashup-Mashup, API-API), which can affect the accuracy of recommendations (Cao&Liu et al.,2019). Deep learning-based recommendation methods are introduced into Web networks to solve the problem of data sparsity, and many studies have shown that the introduction of knowledge graphs in Web service recommendation can fully exploit the internal information of knowledge graphs to improve the recommendation effect (Jiang et al.,2021; Wang et al.,2017). However, these methods focus more on mining the structural attributes of the knowledge graph and ignore extracting the auxiliary information(e.g., tags) of the nodes, which affects the accuracy of recommendations (Cao et al.,2023). In response to the shortcomings of existing recommendation methods, this paper proposes a service recommendation SelfA-DeepFM model .

The main contributions of this paper are summarized as follows:

1. A very effective general framework for service recommendation is proposed to connect API and Mashup related information to improve the performance of service recommendation. Among others, constructing service networks and service clustering models can be done with the DTc-LDA model mentioned in this framework, or other models can be used.
2. The SelfA-DeepFM model is proposed. Using the Self-Attention Mechanism to capture the different importance of feature interactions. The DeepFM model is used to mine the complex interaction information between multi-dimensional features to predict and rank the quality score of API services to recommend suitable APIs, thus improving the recommendation effect.
3. In order to effectively alleviate the sparsity of services, the DTc-LDA model is first used to construct a service network, which provides a comprehensive portrayal of the relationships between Web services and explores the potential relationships between Mashup and APIs, which not only fully considers the text properties, but also combines the network structure information.
4. A series of experiments were conducted on a real Web dataset, and the experimental results show that the SelfA-DeepFM model effectively improves the effect of service recommendation .

The rest of this paper is organized as follows: we begin by deliberating the existing works in literature. Then, we proclaim the preparatory knowledge and present the details of our proposed model. After that, we set up the experiments, introducing the experimental data, the evaluation metric and the implementation details. The experimental results and analysis are illustrated to show the effectiveness of our methods. We finally concluded this manuscript and indicated future work at the end of this present paper.

RELATE WORK

This section first introduces the technical background and research overview of service recommendation methods.

In recent years, research on service recommendation has received much attention due to the information overload problem in the service computing domain. A lot of work has been carried out for traditional Web services. Currently, collaborative filtering is a common algorithm in recommendation systems, where collaborative filtering-based approaches use user activity and past interactions to understand user preferences and generate recommendations (Lian et al.,2021; Jiang et al.,2021). Although the above collaborative filtering-based approach achieves good recommendation results, it still has its limitations, such as when the data is very sparse, the recommendation performance tends to drop significantly. To alleviate data sparsity, Jiang et al. (2021) proposed a service recommendation algorithm based on knowledge graph and collaborative filtering, but collaborative filtering only considers interactions with low-dimensional features and ignores interactions with high-dimensional features, whereas interactions with higher-order features are usually required in complex user service environments (Guo et al.,2017).

The Factorization Machines (FM) has been widely used in the field of machine learning and data mining (Steffen 2010). Some recent works have shown that FM-based approaches have better performance than CF-based approaches in Web service recommendation (Steffen 2010). FM-based service recommendation methods (Lian et al.,2018; Xiao et al.,2017) can take multidimensional features (e.g., tags, topics, co-occurrence relationships, and popularity) as input and consider the combined interactions between the multidimensional features to improve recommendation effectiveness. Despite the obvious advantages of factorization machines, features in FM must be manually preprocessed and higher-order relationships between features are not considered, leading to a decrease in the accuracy of their predictions and recommendations (Kang et al.,2021).

To overcome this drawback in methods based-FM and to better utilize multi-dimensional higher-order features and their implicit cross-interactions, Guo et al. (2017) proposed the DeepFM (Deep Factorization Machines) model, which can combine both low-order and high-order features to better explore the relationship between various features. In 2019, Cao et al. (2019) proposed an API service recommendation method integrating SOM functional clustering and Mashup quality prediction, and used the DeepFM model to mine the implicit information and improve the recommendation effect. Although, Mashup utilizes multi-dimensional features in an implicit manner, their role on each feature lacks reasonable interpretability. Considering that the interaction with useless features may introduce noise and adversely affect the prediction performance, AFM (Attention Factorization Machines) was introduced into the service recommendation (Xiao et al., 2017). AFM learns the importance of each feature interaction from the data through the Neural Attention Network. Based on this, Kang et al. (2021) proposed a hybrid decomposition machine model with a novel neural network architecture, called NAFM, which integrates a deep neural network to capture non-linear and complex feature interactions and uses an attention mechanism to capture the different importance of feature interactions. However, the model does not consider the topology of the service network, which affects the service recommendation performance.

Considering the combination of explicit higher-order interaction modules with implicit interaction modules, implicit interaction modules and traditional FM modules, xDeepFM (Lian et al., 2018) was introduced into service recommendation and achieved good performance (Cao et al., 2023). In 2023, Cao et al. proposed a service recommendation method integrating a bilinear graph attention network (BGAT) service representation and xDeepFM for quality prediction. This method is based on content-oriented and structure-oriented service function classification, and is combined with service call prediction based on multi-dimensional quality attributes. Despite the superiority of deep learning-based FM, these approaches ignore the relationship between Web service structural properties (e.g., Mashup-API, Mashup-Mashup, API-API), which can affect the accuracy of recommendations (Cao&Liu et al., 2019).

ALGORITHM ANALYSIS

In this section, we first briefly describe the problem definitions for constructing Web service networks and service recommendations for Mashup creation, and list the symbols used in this article. Then, four functional modules are described in detail: data set processing, constructing Web service networks, Service clustering, SelfA-DeepFM-based service prediction and ranking. Finally, the relationship between SelfA-DeepFM and existing methods is compared, and the theoretical merits of the proposed approach are argued from different aspects for service recommendation.

Framework

Specifically, the general framework of the method proposed in this paper is shown in Figure 1. It mainly consists of four components: Data-set processing; Constructing Web service networks; Service clustering and SelfA-DeepFM -based service prediction and ranking.

(1) Data-set processing: After crawling the service dataset from the programmable Web (https://www.programmableWeb.com/), the API and Mashup description information from the service repository is subjected to data pre-processing, usually three steps of document pre-processing: normalization, word stemming, and tagging and deactivation word removal.
(2) Constructing Web service networks: The document-topic distribution derived from the training of DTc-LDA model can be used to calculate the similarity between Web services, and build the Web service networks to mine the potential relationships between Mashups and APIs, so as to reduce the impact of data sparsity.
(3) Service clustering: Clustering of the service network divides different APIs into different functional clusters and calculates numerical feature similarity, such as Similarity of Inter-Class, Similarity of Desc-Docs, Similarity of Tags, Similarity of Desc-Docs + Tags.
(4) SelfA-DeepFM-based service prediction and ranking: The self attention representation component learns the weights of latent features and their interactions in the Web service records and concatenates all latent features into attentive representations. The self-attention mechanism is used to capture the different importance of feature interactions. The DeepFM model is used to mine the complex interaction information between multidimensional features to predict and rank the quality score of API services to recommend suitable APIs.

The following describes in detail the models, algorithms and implementation processes included in each functional module of the method.

Problem Definition

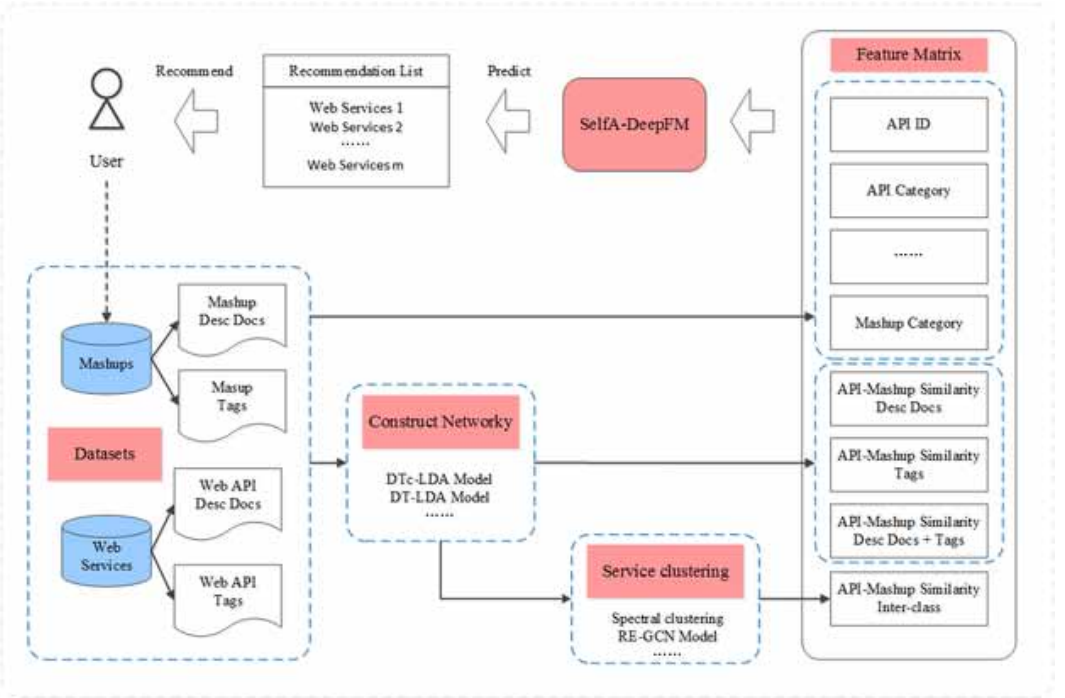
Definition 1 (Web Service Function Description Document)

A function description document of Web Service *M* can be represented as $D = \left\{d_1, d_2, \ldots, d_M\right\}$,*D* represents the set of Web service document description document. The d_i represents the *i-th* text of the description document, and *M* represents the total number of documents. The $d_i = \left\{w_1, w_2, \ldots, w_N\right\}$,where w_i represents the *i-th* word in the description text, *N* indicates the total number of words in the description text.

Definition 2 (The Tag of Web Service Document)

$T = \{t_{ij} \mid 1 <= i <= M$, indicating the serial number of the Web service document, *j* is the serial number of the tag of a document}, and t_{ij} is the *j-th* tag of the *i-th* document. *N* indicates the total number of words in the description documents.

Figure 1. The framework of web service recommendation



Definition 3 (Web Service Network)

$G = \{V, E\}$ $|V| = M$, V represents the vertex set of Web service network. Each vertex V represents a Web service, and E represents the set of connecting edges between vertices. $A = \left[a_{ij}\right]_{M \times M}$ represents adjacency matrix of G, a_{ij} represents the similarity between the *i-th* service and the *j-th* service.

Definition 4 (Service Recommendation)

In a traditional recommender system, for the set of users $U = \{u1, u2...\}$ and the set of items $I = \{i1, i2...\}$, the prediction function is expressed as follows:

$$y: U \times I \to R \tag{1}$$

Where $y(u, i)$ is the predicted rating value, which indicates the rating of user u for item i. In FMs, the target Mashup and the active service API can be considered as users and items, respectively. In the Web API recommendation scenario for Mashup, we use the IDs of the API and Mashup, the category information of the API and Mashup, and the numerical feature similarity between the API and Mashup to predict the recommendation probability. These features are used as the input feature vectors in DeepFM modeling. It should be noted that other relevant features can also be considered flexibly, while the focus of this paper is more on the improvement of the recommendation model. Therefore, the problem statement in Eq. (1) can be extended to the following function with the features of the five main aspects as inputs.

$$y: DA \times DM \times SIMs \times CA \times CM \to S \tag{2}$$

Table 1. Main symbols used

Notation	Description	Notation	Description
D	Represents a collection of web service document description texts	L	the number of layers of similar networks
d_i	denotes the description text of the *i-th* document	$z_{m,n}$	Implied theme
M	Indicates the total number of documents	θ_d^T	θ_d^T is the transpose of θ_d
T	Web service documentation tags	θ_t^T	θ_t^T is the transpose of θ_t
w_i	denotes the *i-th* word in the description text	G	Network weighting
N	indicates the total number of words contained in the description text	SIM	Similarity between the API and the Mashup
t_{ij}	denotes the *j-th* tag of the *i-th* document	γ	is the coefficient
a_{ij}	denotes the similarity of the *i-th* service to the *j-th* service	DA	Description information of the API
V	denotes the set of vertices of the web service network	DM	Description information of the Mashup
E	denotes the set of connected edges between vertices	θ_t	Tag-Topic Distribution
A	denotes the adjacency matrix of G	CA	Categories of API
θ_d	Calculation of document-topic distribution	CM	Categories of Mashup
θ_m	Indicates the topic distribution of the document	S	Prediction Score
φ_k	Thematic distribution of vocabulary under the theme	α,β	A priori parameters
$n_t^{(k)}$	indicates the number of topic z occurring in document tag t.	α_k	the prior Dirichlet of the topic z
$n_d^{(k)}$	denotes the number of topic z occurring in the document description document d	G	Web Services Network

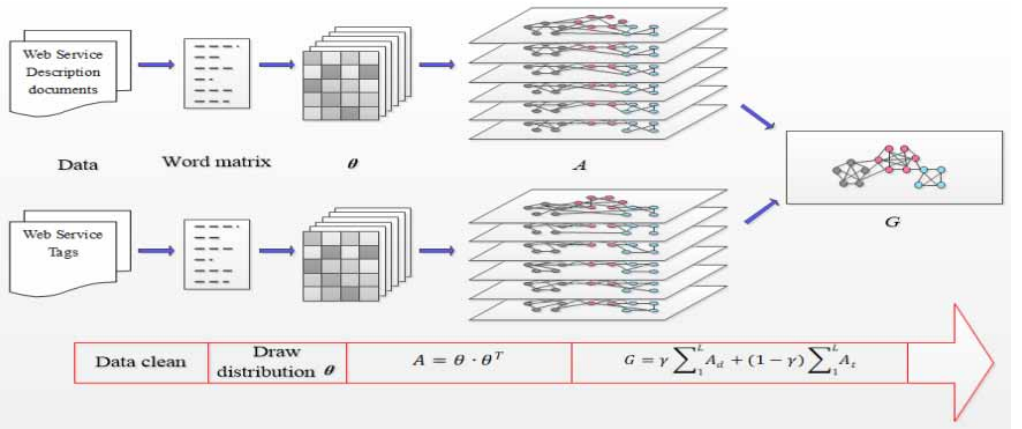
where *DA, DM, SIMs, CA, CM,* and *S* are the descriptive information of API, descriptive information of Mashup, numerical feature similarity between API and Mashup, category of API, category of Mashup, and prediction score, respectively. The purpose of service recommendation in this paper is to predict the probability of Web APIs invoked by Mashup. Table 1 lists the main notations used in this section.

Constructing a Service Network

Properly constructed Web service networks can effectively describe the relationships between Web services. In view of this feature, the DTc-LDA model (Deng et al.,2021) (Description and Tags with coefficient Latent Dirichlet Allocation) is used to construct the service network, and its working principle is shown in Figure 2.

The description of working process of DTc-LDA model: Detailed work has been published in the article (Deng et al.,2021). This DTc-LDA model constructs two Web service networks, which are

Figure 2. DTc-LDA model



modeled by document *D* and tags *T* respectively. Then, the two networks are superposed and fused by weight. Finally, the target network *G* is constructed.

The two types of similar networks are overlapped and fused by weight to generate the target network.

$$G = \gamma \sum_{i=1}^{L} A_{di} + \left(1-\gamma\right) \sum_{i=1}^{L} A_{ti} \tag{3}$$

L is the number of similar network layers and γ is the coefficient. $A_d = \theta_d \times \theta_d^T$, $A_t = \theta_t \times \theta_t^T$ The adjacency matrix of the document similar network is calculated. θ_d^T is the transpose of θ_d; θ_t^T is the transpose of θ_t.

Multi-Head Self-Attention Representation

In a recommendation system, different features in the service data will represent different contributions to the recommendation results, and thus need to be assigned reasonable weights to them. The attention mechanism (Vaswani et al., 2017) is a mechanism that extracts important information from a large amount of information by simulating the way the human brain processes information overload, which relies less on external information and focuses more on capturing the internal relevance of data or features. The use of the Self-Attentive Mechanism to dynamically adjust the weights of different features of the service data can enhance the extraction of features with high contribution to the service recommendation results and make the model more focused on the classification of features with high contribution (Tang et al., 2023).

Let $X = \left\{x_1, x_2 ... x_N\right\}$ be all Web service records and a record is denoted by features, $x_i = \left\{F_1, F_2, ..., F_K\right\}$. The embedding layer produces dense features like n_{ik}, which represents the *k-th* feature of the *i-th* record. The multi-head self-attention mechanism can be formulated as follows:

$$\alpha_{k_1,k_2}^{(h)} = \frac{exp\left(sim\left(n_{ik_1}, n_{ik_2}\right)\right)}{\sum_{\left\{k_2 \neq k_1 \middle| k_2 \in F\right\}} exp\left(sim\left(n_i k_1, n_i k_2\right)\right)}, sim^{(h)}\left(n_{ik_1}, n_{ik_2}\right) = \left\langle W_{query}^{(h)} n_{ik_1}, W_{key}^{(h)} n_{ik_2} \right\rangle \tag{4}$$

wherein, $h \in \left(1, H\right)$ is the head number, k_1 and k_2 are dense features, $\alpha_{k_i,k_j}^{(h)}$ represents the attention weight for generating the representation of k_i. The higher the value of $\alpha_{k_i,k_j}^{(h)}$, the more influence on the *ki*'s feature embedding imposed by the *k j*'s feature preference. The *k*-th dense feature can be calcaluated as follows:

$$\hat{n}_{ik}^{(h)} = \sum_{j=1}^{K} \alpha_{k_1,k_2}^{(h)} W_{value}^{(h)} n_{ij} \tag{5}$$

where $\hat{n}_i^{(h)} \in R^{1\times A}$ is the representation calculated by a single attention head. Multi-head attention allows the model to jointly attend to the information from different representation subspaces at different positions, and the computational formula is as follows:

$$\hat{n}_i = \hat{n}_i^{(1)} \oplus \hat{n}_i^{(2)} \oplus \ldots \oplus \hat{n}_i^{(H)} \tag{6}$$

SelfA-DeepFM-Based Service Prediction and Ranking

Usually, in a recommendation system, combining different features in the data positively affects the recommendation results. If we are only using a simple linear model, we cannot learn higher-order features or imperceptible latent features from the service data. DeepFM (Guo et al.,2017) is an effective way to extract complex features from ordinary data without manual feature extraction. This can significantly improve the accuracy of recommendations when DeepFM is used in Web API recommendations. The DeepFM model takes the IDs of Web APIs and Mashups, the category information of APIs and Mashups, and a one-hot encoding of numerical similarity features between APIs and Mashups as inputs to construct a multidimensional information matrix, as shown in Figure 3.

The DeepFM comprehensively learns low and high-order feature interactions, deeply mining potential relationships between various information (Cao&Liu&Wen et al.,2019). Moreover, the output of the self-attention representation module implies personalized traits and correlation among different features. Every element of this vector is informative. The DeepFM model automatically extracts valid feature combination relationships including high-order features and low-order feature combination relationships from Web data (such as documents semantic similarity, Tag semantic similarity, etc.),

Figure 3. Multi-dimensional information matrix

Box1 API ID	Box2 Mashup ID	Box3 API Category	Box4 Mashup Category	Box5 Similarity Feature1	Box6 Similarity Feature2	Box7 Similarity Feature3	Box8 Similarity Feature4	.....	Targe Y
A1	M1	Maping	Travel	0.5	0.4	0.4	0.3	.....	1
A1	M2	Maping	Music	0.2	0.1	0.2	0.4	.....	0
A2	M1	Sports	Travel	0.4	0.2	0.3	0.2	.....	1
A2	M2	Sports	Music	0.3	0.1	0.2	0.2	.....	0
.....	.....	.....	.....	.....	.....	.....	.....	.....	.....

predicts and ranks the quality score of each API service relative to the target Mashup application, and recommends the top N API services with the highest scores to developer users.

Among them, the detailed meanings of Box names are shown in Table 2:

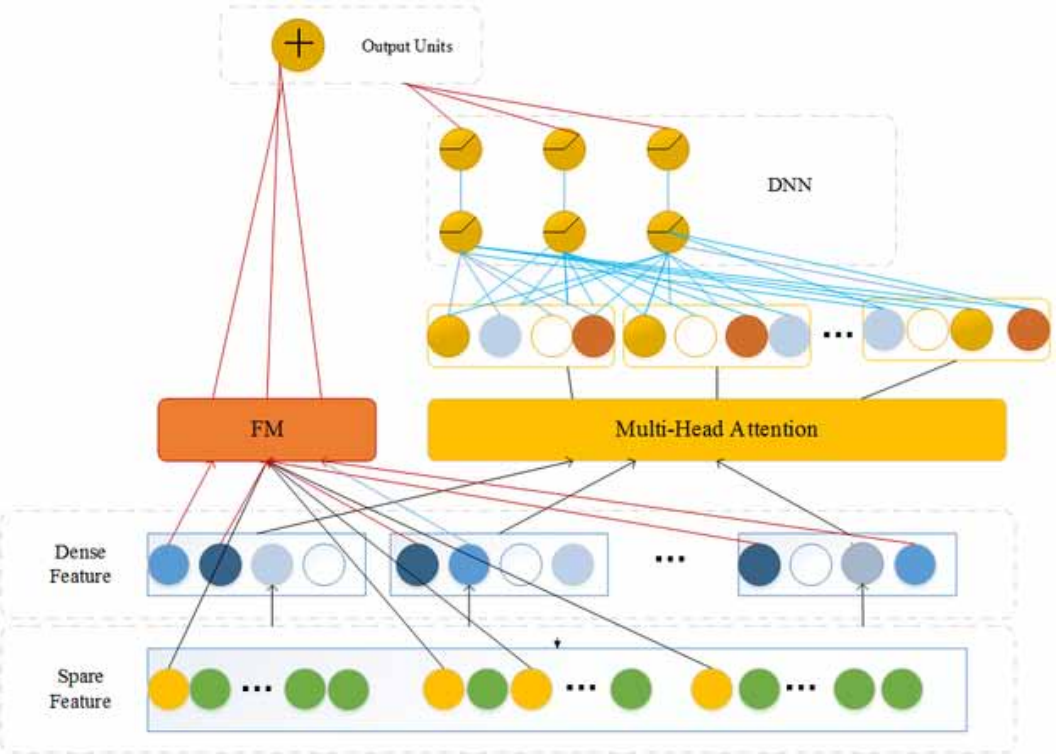
As shown in Figure 4, we combine DeepFM and self-attention representation to build a neural network SelfA-DeepFM . The Web services prediction procedure comprises the following steps:

1) *Feature Processing:* This step takes Web services records as input. The raw data need to be transformed into a standard format that can be easily encoded. Categorical features are mapped

Table 2. Detailed meaning of Box name

Box name	Implication
Box1	API ID
Box2	Mashup ID
Box3	API Category
Box4	Mashup Category
Box5-Box28	The similarity features between each API and Mashup, there are 4 major features in this category, which are description text similarity features, label similarity features, description text + label similarity features, and belonging class center similarity features. Finally, a coefficient is assigned to these 4 main features, which are fused into 24 features, and the coefficient values are determined by the attention mechanism.
Target Y	If the API is actually called by Mashup, it is 1 or 0.

Figure 4. The model SelfA-DeepFM based on self-attention representation and DeepFM



into one-hot vectors. As mentioned above, in a record $x_i = \left\{F_1, F_2, ..., F_f\right\}$, F_i contains the IDs of Web APIs and Mashups, the category information of APIs and Mashups, and a one-hot encoding of numerical feature similarities between APIs and Mashups as inputs to construct a multidimensional information matrix, as shown in Figure 3.

2) *Feature Embedding*: The sparse features output by the last step are high dimensional vectors. In this step, they are embedded into dense and low dimensional vectors. The embedding vector is calculated the same as the multi-head self-attention representation component, i.e.:

$$n_{if} = x_{if} M_f \tag{7}$$

where $M_f \in R^{F \times D}$ is the embedding matrix, *F* is the number of Feature *F's* category, and *D* is the embedding dimension.

3) *Low-Level Feature Interaction Capturing:* This step employs a FM to learn the first-order and second-order feature interactions. The following equation (8) is the representation of FM, where w, x corresponds to the first-order calculation in the Figure 4, and the right side of the formula corresponds to the second order calculation. The output of the FM is a weighted summation of the first-order embedding and inner product of embeddings, which can be denoted by:

$$y_{FM} = w, x + \sum_{j_1=1}^{f} \sum_{j_2=j_1+1}^{f} v_i, v_j x_{j1} x_{j2} \tag{8}$$

where *f* is the input feature size, *w* is the weight of the dimensional feature, and *x* is the feature after a one-hot encoding.

4) *High-Level Feature Interaction Capturing:* This step learns the complex interactions between features. As mentioned previously, the process of encoding is as follows:

$$\begin{aligned} a^{(0)} &= \hat{n}_i = \left[\hat{n}_{i1}, \hat{n}_{i2} ... \hat{n}_{iF}\right] \\ a^{(l+1)} &= \sigma\left(W^{(l)} a^{(l)} + b^{(l)}\right) \end{aligned} \tag{9}$$

wherein, $a^{(l)}$ represents the output of the *l-th* layer. $\hat{n}_{ij}$ indicates the embedding representation, $W^{(l)}$ and $b^{(l)}$ denote the weight and bias, respectively. The output of the attention module is:

$$y_{ANN} = \sigma\left(W^{(L)} a^{(L)} + b^{(L)}\right) \tag{10}$$

5) *Web Services Predicting:* The low-level and high-level feature interactions are jointly combined into the Web Services prediction model. Finally, the predicted Web Services value is calculated as:

$$\hat{y} = Relu\left(y_{FM} + y_{ANN}\right) \tag{11}$$

EXPERIMENT AND EVALUATION

Research Questions

In the following, we first introduce the experimental data, evaluation metrics, comparison methods, and then discuss the detailed experimental results. In this section, we validate the performance of the proposed service recommendation method on various real-world networks. We conduct extensive experiments with the aim of answering the following research questions (RQ):

RQ1: How does the proposed SelfA-DeepFM model perform compared to other methods?
RQ2: Are both the attention mechanism and the numerical characteristics essential for SelfA-DeepFM?

Dataset and Preparation

To ensure the authenticity of the experiment, we obtained 18,536 real APIs and 7,732 real Mashups from the Programmable Web platform (PWeb). API and Mashup have few features, with only a small number of category features such as ID, Category and Tags, etc. The number of samples eligible for the experiment is 387*6,206 = 2,401,722, of which the number of invoked relations is 8,224. 8,224/2,401,722=0.34%. That is, only 34 calling relationships out of 10,000 data, and to find them accurately is a challenge to the recommended method. In this paper, the top 20 categories with the most API services are selected as the experimental data set, which are showed in table 3.

Evaluation Index

According to the Mashup needs of developers and users, a set of experiments are set up based on the results of service clustering to compare the experimental results of different recommended methods.

Table 3. The number of top-20 categories API

Category	Number of API service	Category	Number of API service
Financial	902	Security	362
Tools	823	Telephony	333
Messaging	611	Email	317
Payments	604	Education	291
eCommerce	538	Reference	284
Enterprise	483	Data	279
Social	474	Video	279
Mapping	405	Transportation	277
Science	384	Games	268
Government	376	Sports	262

The effectiveness of the experiments is evaluated using metrics of precision@k, recall@k, MRR and $NDCG_u@k$.

(1) precision@k,reflects the proportion of relevant Web APIs in the set of recommended Web APIs, it is calculated as shown in the Eq(12):

$$precision@k = \frac{\left|Rel_u \cap Rec_u\right|}{\left|Rec_u\right|} \tag{12}$$

where, Rec_u denotes the set of goods (the test set) associated with user *u*; Rel_u indicates the first K lists recommended to the user; The intersection of the two is divided by the number of Rel_u elements of the set (actually K) to get Precision@K. Generally, Precision@K is calculated for each user and then averaged.

(2) *recall@k*,reflects the ratio of recommended relevant Web APIs to all relevant Web APIs and is calculated as shown in Eq(13):

$$recall@k = \frac{\left|Rel_u \cap Rec_u\right|}{\left|Rel_u\right|} \tag{13}$$

where, Rec_u denotes the set of goods (the test set) associated with user *u* ; Rel_u indicates the first K lists recommended to the user; The intersection of the two is divided by the number of set elements of $\left|Rel_u\right|$ to obtain recall@K.

(3) MRR, the performance of the retrieval system is evaluated by the ranking of the correct search result value in the search results.

$$MRR = \frac{1}{Q}\sum_{i=1}^{|Q|}\frac{1}{rank_i} \tag{14}$$

Where $\left|Q\right|$ is the number of users; $rank_i$ is for the i *-th* user, the position of the first item in the ground-truth result in the recommendation list.

(4) $NDCG_u@k$ (normalized discounted cumulative gain) In the field of information retrieval, this method is a popular metric for measuring ranking quality. It is used to measure the merit of the ranking of the recommended Web APIs in the recommendation list. The value of $NDCG_u@k$ is higher, the result of the recommendation list ranking of the Web APIs is better. That is

$$DCG_u@k = \sum_{i}^{k} \frac{2^{r(i)} - 1}{\log_2(i+1)} \tag{15}$$

$$IDCG_u = \sum_{i}^{k} \frac{1}{\log_2(i+1)} \tag{16}$$

$$NDCG_u@k = \frac{DCG_u@k}{IDCG_u} \tag{17}$$

k indicates the recommended number of Web APIs; $r(i)$ indicates the relevance score of the $i-th$ recommended Web API. If the $i-th$ Web API recommended is the Web APIs called by Mashup in the real dataset, then $r(i)=1$; otherwise, $r(i)=0$. $IDCG_u$ is the *ideal* $DCG_u@k$, which is the maximum value of $DCG_u@k$. $DCG_u@k$ can be calculated by Equation (15), that is, the optimal recommended case.

Method Comparison

In this paper, 7 methods are used as baseline system for evaluation and comparative analysis. As follows:

1. **AutoInt** (Song et al.,2019):The goal of this method is to map the original sparse high-dimensional feature vector to the low-dimensional space while building a high-order feature interaction model.
2. **DCN** (Wang et al.,2017):The Wide part is replaced by Cross implemented by a special network structure, which automatically constructs finite higher-order cross features and learns the corresponding weights, saying farewell to the tedious manual fork multiplication.
3. **DCNMix** (Wang&Rakesh et al.,2017):Efficiently learning the intersection of explicit and implicit features, the model is efficient and simple and being more expressive.
4. **FiBiNET** (Huang et al.,2019):Combining inner product and Hadamard product and introducing an additional parameter matrix W WW to learn the feature crossr.
5. **DeepFM** (Guo et al.,2017):Combining FM with neural networks to improve the ability of FM to capture multi-order interaction information among features.
6. **xDeepFM** (Lian et al.,2018):It jointly learns explicit and implicit high-order feature interactions effectively. xDeepFM (eXtreme Deep Factorization Machines) is a further optimization of DCN. The idea of FM vector level is introduced in the crossover section to form the CIN module, which can explicitly measure the higher-order feature interaction of services. Linear, CIN, and DNN are combined to form an xDeepFM.
7. **WSR-MGAT** (Li et al.,2022): The Web services recommendation based on Metapath-guided Graph Attention Network Model (WSR-MGAT) to fully exploit the structural information of the knowledge graph to improve the recommendation accuracy. It uses meta-paths to guide nodes to recursively aggregate higher-order neighbor information and use an attention mechanism to distinguish the importance of neighbors.

Experimental Result

Based on the Mashup needs of developers and users, based on the results of service clustering, we set up a set of experiments to compare the experimental results of different recommended methods.

Performance Comparison (RQ1)

To answer the question RQ1, the experiment was set up with six comparison methods. Web services contain basic features: API ID, Mashup ID, API Category and Mashup Category. The similarity features between each API and Mashup, there are 4 major features in this category, which are

description documents similarity features (Desc-Docs), tags similarity features (Tags), description documents + tags similarity features (Desc-Docs+Tags), and belonging class center similarity features (Similarity of Inter-Class).

In Table 4, it can be observed that the model SelfA-DeepFM proposed in this paper outperforms the other methods in the first top2 and top3 recommendation results. This demonstrates that the DTc-LDA module in the model SelfA-DeepFM connects API and Mashup related information to reasonably construct the Web service network and mine the potential relationship between Mashup and API, thus reducing the impact of data sparsity; Attention mechanisms capturing the differential importance of multi-dimensional feature interactions are important for improving the accuracy of prediction systems.

Ablation Study (RQ2)

To answer question RQ2, in this section we validate the effect of the number of Similarity Features (SF) and the self attention mechanism on the model's effectiveness, respectively, by setting up a comparison experiment: SF=0 means only basic features API ID, Mashup ID, API Category and Mashup Category and the number of similarity features is 0 ; SF=2 means that the number of similarity features randomly selected is 2; SF=4 means the number of similarity features is 4; SF=28 indicates that 4 similarity features are linearly fused into 28 features and the coefficients are set manually. SF=28-A indicates that 4 similarity features are linearly fused into 28 features and the coefficients are determined by the self attention mechanism. When topk=2 and topk=3, the experimental results are shown in Figure.5 and Figure.6.

In order to study the effect of different number of similarity features on the effectiveness of the model by comparing SF=0, SF=2, SF=4 and SF=28. It can be analyzed from the figures that the experimental results are all improved with the increase of the number of similarity features. The worst experimental results are obtained when SF=0, which also shows the importance of different similarity features on the model.

Table 4. Comparison of the results of various algorithms

Methods	topK	precision@K	recall@K	MRR	NDCG@K
AutoInt	2	0.4398	0.8041	0.8115	0.7822
DCN	2	0.4405	0.8048	0.8176	0.7812
DCNMix	2	0.4339	0.7947	0.8069	0.7692
DeepFM	2	0.4315	0.7902	0.8002	0.7623
xDeepFM	2	0.4326	0.7912	0.8024	0.7656
WSR-MGAT	2	0.4296	0.8044	0.8011	0.7679
SelfA-DeepFM	2	**0.4408**	**0.8050**	**0.8201**	**0.7825**
AutoInt	3	0.3139	0.8539	0.8215	0.8072
DCN	3	0.3119	0.8486	0.8176	0.8021
DCNMix	3	0.3096	0.8427	0.8069	0.7925
DeepFM	3	0.3076	0.8389	0.8014	0.7885
xDeepFM	3	0.3082	0.8405	0.8024	0.7892
WSR-MGAT	3	0.3145	0.8489	0.8184	0.8061
SelfA-DeepFM	3	**0.3153**	**0.8573**	**0.8287**	**0.8076**

Figure 5. Effect of different similarity feature numbers on the model, when topk=2

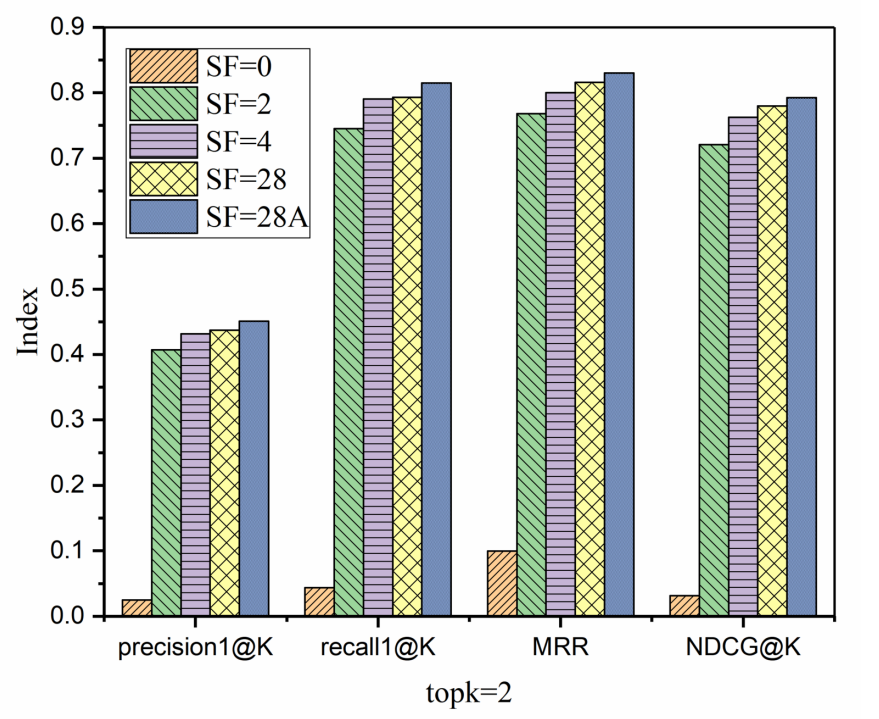
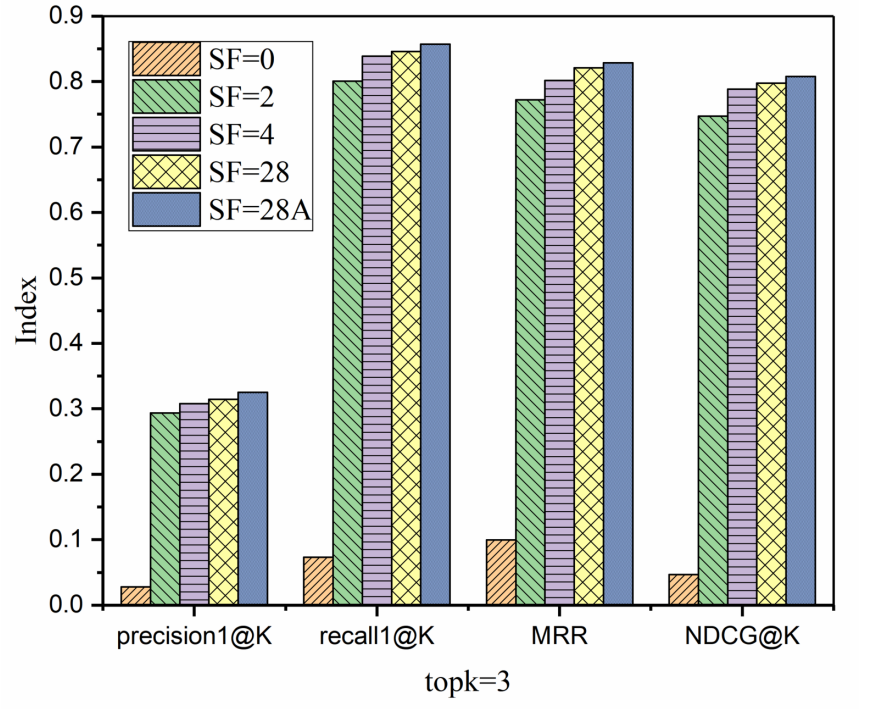


Figure 6. Effect of different similarity feature numbers on the model, when topk=3



To investigate the effect of self-attention mechanism in the model, the experiments compared the SF=28 and SF=28-A experimental results. It can be observed that the effect of SF=28-A is higher than that of SF=28, which indicates that the effect of the SealfA-DeepFM model is better than that of DeepFM, indicating that the self-attention mechanism is very necessary for service recommendation. Introducing an self-attention mechanism to assign different weights to different features, reflecting the different importance of the features, which can improve the effect of the model.

CONCLUSION AND FUTURE WORK

In the field of service computing, there are many studies on recommending suitable APIs for Mashup creation, in which CF algorithms play an important role. However, existing CF-based recommendation methods suffer from data sparsity and cold start, which lead to the poor recommendation results. To alleviate the above problems, this paper proposes the service recommendation method based on attention mechanism. The method first constructs the service network with DTc-LDA model to mine the potential relationship between Mashup and API, which not only fully considers the text attributes but also combines the network structure information to effectively mitigate the sparsity of the service. Since not all features are equally useful and predictive for service recommendations, modeling all features with the same weight may lead to sub-optimal results. Therefore, the model captures the different importance of feature interactions with an attention mechanism, and uses the DeepFM model to model and mine the complex interactions between its multidimensional features to predict and rank the quality scores of API services to recommend suitable APIs.

In the work of service recommendation, higher-order feature interactions have different importance. Therefore, in future work, we will investigate the effect of more finer-grained higher-order feature interactions to further improve the performance of service recommendations. In addition, users may have uncertain preferences for Web API. This was not taken into account in this work. Therefore, we can consider recommending diverse Mashup solutions to expand the range of options for users, so that the accuracy of service recommendations can be improved.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China Grant Nos. 11702289, the Key Core Technology and Generic Technology R&D Project of Shanxi Province (Grant No. 2020XXX013), the Scientific and Technological Innovation Programs of Higher Education Institutions in Shanxi (Grant No. 2020L0102), and the National Key R&D Program of China (2020 Special Project of Science and Technology for Assisting Economics).

REFERENCES

Almarimi, N., Ouni, A., Bouktif, S., Mkaouer, M. W., Kula, R. G., & Saied, M. A. (2019). Web service API recommendation for automated mashup creation using multi-objective evolutionary search. *Applied Soft Computing*, *85*, 105830. doi:10.1016/j.asoc.2019.105830

Cao, B., Liu, J., Wen, Y., Li, H., Xiao, Q., & Chen, J. (2019). QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications. *Journal of Parallel and Distributed Computing*, *132*, 177–189. doi:10.1016/j.jpdc.2018.04.002

Cao, B., Liu, X. F., Rahman, M. M., Li, B., Liu, J., & Tang, M. (2017). Integrated content and network-based service clustering and web apis recommendation for mashup development. *IEEE Transactions on Services Computing*, *13*(1), 99–113. doi:10.1109/TSC.2017.2686390

Cao, B., Zhang, L., Peng, M., Qing, Y., Kang, G., & Liu, J. (2023). Web Service Recommendation via Combining Bilinear Graph Representation and xDeepFM Quality Prediction. *IEEE Transactions on Network and Service Management*, *20*(2), 1078–1092. doi:10.1109/TNSM.2023.3234067

Cao, Y., Liu, J., Shi, M., Cao, B., Chen, T., & Wen, Y. (2019). Service recommendation based on attentional factorization machine. *In 2019 IEEE International Conference on Services Computing (SCC)*, (pp. 189-196). IEEE. doi:10.1109/SCC.2019.00040

Cheng, H. T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., & Shah, H. (2016). Wide & deep learning for recommender systems. *In Proceedings of the 1st workshop on deep learning for recommender systems,* (pp. 7-10). IEEE. doi:10.1145/2988450.2988454

Deng, L. P., Guo, B., & Wen, Z. (2021). Web Service Clustering Approach Based on Network and Fused Document-Based and Tag-Based Topics Similarity. [IJWSR]. *International Journal of Web Services Research*, *18*(3), 63–81. doi:10.4018/IJWSR.2021070104

Guo, H., Tang, R., Ye, Y., Li, Z., & He, X. (2017). DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv preprint a*rXiv:1703.04247. arxiv.org/abs/1703.04247 10.24963/ijcai.2017/239

Hu, T., Xu, C., Zhang, S., Tao, S., & Li, L. (2023). Cross-site scripting detection with two-channel feature fusion embedded in self-attention mechanism. *Computers & Security*, *124*(102990), 1–12. doi:10.1016/j.cose.2022.102990

Huang, T., Zhang, Z., & Zhang, J. (2019). FiBiNET: combining feature importance and bilinear feature interaction for click-through rate prediction. *In Proceedings of the 13th ACM Conference on Recommender Systems.* ACM. 169-177.doi.org/10.1145/3298689.3347043

Jiang, B., Yang, J., Qin, Y., Wang, T., Wang, M., & Pan, W. (2021). A service recommendation algorithm based on knowledge graph and collaborative filtering. *IEEE Access : Practical Innovations, Open Solutions*, *9*, 50880–50892. doi:10.1109/ACCESS.2021.3068570

Kang, G., Liu, J., Xiao, Y., Cao, B., Xu, Y., & Cao, M. (2021). Neural and attentional factorization machine-based Web API recommendation for mashup development. *IEEE Transactions on Network and Service Management*, *18*(4), 4183–4196. doi:10.1109/TNSM.2021.3125028

Li, X., Zhang, X., Wang, P., & Cao, Z. (2022). Web services recommendation based on Metapath-guided graph attention network. *The Journal of Supercomputing*, *78*(10), 12621–12647. doi:10.1007/s11227-022-04369-8

Lian, J., Zhou, X., Zhang, F., Chen, Z., Xie, X., & Sun, G. (2018). XDeepFM: Combining explicit and implicit feature interactions for recommender systems. *In Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, (pp. 1754-1763). ACM. doi:10.1145/3219819.3220023

Lian, S., & Tang, M. (2022). API recommendation for Mashup creation based on neural graph collaborative filtering. *Connection Science*, *34*(1), 124–138. doi:10.1080/09540091.2021.1974819

Liang, W., Tang, M., Long, J., Peng, X., Xu, J., & Li, K.-C. (2019). A secure fabric blockchain-based data transmission technique for industrial internet-of-things. *IEEE Transactions on Industrial Informatics*, *15*(6), 3582–3592. doi:10.1109/TII.2019.2907092

Rendle, S., Gantner, Z., Freudenthaler, C., & Schmidt-Thieme, L. (2011). Fast context-aware recommendations with factorization machines. *In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval,* (pp. 635-644). ACM. doi:10.1145/2009916.2010002

Song, W., Shi, C., Xiao, Z., Duan, Z., Xu, Y., Zhang, M., & Tang, J. (2019, November). Autoint: Automatic feature interaction learning via self-attentive neural networks. *In Proceedings of the 28th ACM international conference on information and knowledge management*, (pp. 1161-1170). ACM. doi:10.1145/3357384.3357925

Srivastava, P., Bej, S., Yordanova, K., & Wolkenhauer, O. (2021). Self-attention-based models for the extraction of molecular interactions from biological texts. *Biomolecules*, *11*(11), 1591. doi:10.3390/biom11111591 PMID:34827589

Steffen Rendle. (2010). Factorization machines. *In 2010 IEEE International conference on data mining*. IEEE. 995-1000.doi.org/10.1109/ICDM.2010.127

Tang, M., Tang, W., & Xie, F. (2023). Accurately Predicting Quality of Services in IoT via Using Self-Attention Representation and Deep Factorization Machines. *IEEE Transactions on Intelligent Transportation Systems*, 1558–0016. doi:10.1109/TITS.2023.3279412

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., & Polosukhin, I. (2017). *Attention is all you need* (Vol. 30). Advances in neural information processing systems, arxiv.org/abs/1706.03762

Wang, R., Fu, B., Fu, G., & Wang, M. (2017). Deep & cross network for ad click predictions. *In Proceedings of the ADKDD'17*, (pp. 1-7). IEEE. doi:10.1145/3124749.3124754

Xiao, J., Ye, H., He, X., Zhang, H., Wu, F., & Chua, T. S. (2017). Attentional factorization machines: Learning the weight of feature interactions via attention network, *International Joint Conference on Artificial Intelligence,* (pp. 3119-3125). IEEE. arxiv.org/abs/1708.04617 doi:10.24963/ijcai.2017/435

Liping Deng is currently a Ph.D. student at the College of Data Science, Taiyuan University of Technology. Her main research interests include social network, Service Computing, Artificial Intelligent (AI), Complex Networks, and Machine Learning. Bing Guo is currently a teacher at Department of Computer Science and Technology, Taiyuan Normal University. His main research interests include social network, Service Computing, Artificial Intelligent(AI), Complex Networks, and Machine Learning.

Wen Zheng is a Professor of the College of Data Science, Taiyuan University of Technology. He has been a director of the institute of big data for public security at Taiyuan University of Technology since 2018. He graduated from Modern Mechanics Department of University of Science and Technology of China with his bachelor's degree in 2008 and Ph.D.degree in 2013. From 2013 to 2018, he successively served as a post doctoral fellow at the national research center of microscale physical science in Hefei. And he began to work as an associate researcher at the College of Physics, University of Science and Technology of China. His current research interests include big data technology, industrial Internet of things, computational physics, and impact dynamics.