DINRec: Deep Interest Network Based API Recommendation Approach for Mashup Creation

Yong Xiao[1,2], Jianxun Liu[1,2](✉), Rong Hu[1,2], Buqing Cao[1,2], and Yingcheng Cao[1,2]

[1] Key Laboratory of Knowledge Processing and Networked Manufacturing, Hunan University of Science and Technology, Xiangtan 411201, Hunan, China
ljx529@gmail.com

[2] School of Computer Science and Engineering, Hunan University of Science and Technology, Xiangtan 411201, Hunan, China

Abstract. Recommending appropriate APIs for Mashup creation has become a challenge as the number of APIs from different sources grows fast. In order to understand the relationships among multiple ecosystem APIs, most existing API recommendation methods focus on semantic similarity relationships but underutilize the composition and cooperation relationships between APIs, which may lead to low recommendation precision. In view of this problem, a Deep Interest Network based API Recommendation approach (DINRec) for Mashup development is proposed in this paper. In this approach, APIs are chosen incrementally for compositing into a Mashup and in that process the embedding vector of the Mashup's existing composition features will be updated adaptively by using Deep Interest Network. Moreover, a Doc2simu model is used to help training industrial deep networks with relatively small amounts of dataset. Finally, some experiments on real-world dataset are implemented to verify the efficiency of our proposed approach.

Keywords: Deep Interest Network · Doc2simu model · API recommendation · Mashup

1 Introduction

Mashup technique has got a far-reaching impact in recent years which provides a flexible way for fulfilling dynamic and customized Web service developer requirements and tackles the functional limitations of individual Application programming interfaces (APIs). However, as the number of APIs grows rapidly, how to recommend appropriate ones for Mashup creation to satisfy users' requirements becomes a challenge. For example, as of May 16, 2019, the dominant website ProgrammableWeb has published 21,552 web APIs under 484 categories. If a developer wants to build a Mashup related with messaging, ProgrammableWeb search engine will return a list containing 1,576 Web APIs. It is a difficult task to go through these lists of results and select the desired APIs.

Some existing methods focus on keyword or semantic matching while others are based on Quality of Service (QoS) prediction [1] for service recommendation. However, keyword-based matching is usually imprecise while semantic-based matching is expensive to construct in practice. In addition, QoS is unstable and lagging that may affect the precision of real-time prediction. In view of these shortcomings, in recent years, some machine learning techniques such as Latent Dirichlet Allocation (LDA) [2, 3] or Relational Topic Model (RTM) [4, 5] were used to learn topic from the services' descriptions or the users' requirements. In some other studies, Word2vec and Doc2vec [6] were used to extract deep semantic features between words and vectorize descriptions of API and Mashups.

In this paper, we also apply machine learning techniques to recommend APIs to Mashup, while considering the cooperation and composition relationships between APIs simultaneously. Generally, the function of a Mashup is implemented by several APIs, or all APIs realize the complex function of the Mashup. Therefore, the prior chosen APIs may affect the selection strategy of the subsequent APIs while creating a Mashup. For example, a Mashup BBC Browser described as "Maps channel program information to relevant Twitter account" contains three APIs, i.e., BBC Nitro, Twitter and Facebook. Suppose that BBC Nitro API and Twitter API have been chosen for creating this Mashup, then the probability of Facebook API being recommended to the Mashup would increase. The first reason is that an API which is category-similar to the prior selected APIs would not likely to be recommended to the same Mashup. Second, it is more reasonable to recommend an API that can complete the function of the Mashup that have not been completed by the prior selected APIs. For these reasons, Facebook API will be recommended to BBC Browser Mashup according to the prior selected BBC Nitro API and Twitter API.

To realize this conception, we proposed a Deep Interest Network (DIN) [7] based API recommendation approach, called DINRec. By introducing a local activation unit, DINRec adaptively learn the representation vector of composition and cooperation relationships between selected APIs and candidate API, moreover, this representation vector varies over different candidate APIs, which makes it possible to update embedding vector of the prior selected APIs when gradually add APIs into the Mashup. Through this mechanism, the prior selected APIs with higher cooperation to the candidate API will get higher activated weights before they get into the multilayer perceptron. We conduct some experimental studies to gain insight about this phenomenon. The final results show that DINRec perfectly integrates the functional semantics and composition relationship of Mashups.

A recent check of ProgrammableWeb.com's statistics shows that the number of APIs used by Mashups only covers a quarter of the amounts of total APIs, and most Mashups only contain less than 3 Web APIs. However, training industrial deep networks with few features is prone to over-fitting. To solve this problem, in this paper, we present a Doc2simu model to train the text vectors of all Web APIs, and choose the ones with high similarities as the extended dataset help to support effective training in industrial networks. The process of the method proposed in this paper is illustrated in Fig. 1. The contributions of this paper are summarized as follows:

(1) We propose a recommendation method based on Deep interest Network (DINRec), which can improve the expressive ability of feature model and better capture the diversity characteristics related to functional semantics and composition relationships of Mashups.
(2) We present a Doc2simu model to help training industrial deep networks by extending dataset based on the Doc2vec model and cosine similarity.
(3) We conduct experiments on a real-world dataset crawled from ProgrammableWeb to evaluate the effectiveness of DINRec.

The rest of this paper is organized as follows. Section 2 presents the process and structure of DINRec. Section 3 discusses and analyzes the experimental results and variable parameters. Section 4 describes the related works and Section 5 draws a conclusion of the paper.

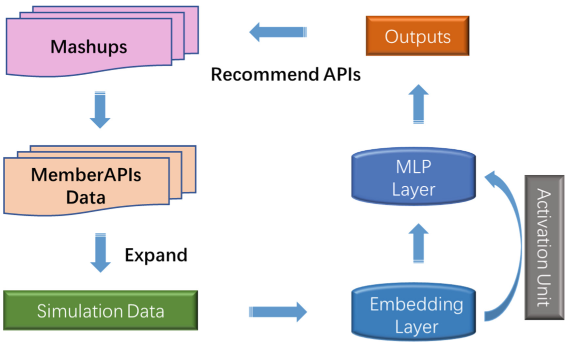


Fig. 1. The framework of DINRec.

2 Process of DINRec

2.1 Feature Representation

Feature Description. Describing the features of Mashups and their member APIs is the fundamental task to obtain the functional semantics and composition relationships of Mashups. Formally, the feature of a Mashup is defined as follow:

Definition 1 (Feature of a Mashup). The feature of a Mashup can be defined as a tuple $F = (F^M, F^A)$. . In this tuple, $F^M = (N^M, T^M, C^M, D^M)$, where N^M is the name of the Mashup, T^M is the tags of the Mashup, C^M is the category of the Mashup and D^M is the description text of the Mashup; $F^A = \{F_i^A | 0 \le i \le N_i\}$ (N^A, T^A, C^A, D^A), where $N^A = \{n_{A,i} | 1 \le i \le n\}$, $T^A = \{t_{A,i} | 1 \le i \le n\}$, $C^A = \{c_{A,i} | 1 \le i \le n\}$ and $D^A = \{d_{A,i} | 1 \le i \le n\}$, $n_{A,i}$, $t_{A,i}$, $c_{A,i}$, $d_{A,i}$ represent the i-th API's name, tag, category and description text feature, respectively, n is the number of member APIs for each Mashups.

Feature Representation. As the above shows, features in our recommendation tasks is mostly in a multi-group categorial form, based on this, we use one-hot encoding to represent features in this paper. One-hot encoding is simple to compute and understand, and employed frequently when it is necessary to represent a categorical variable in a neural network, which is normally transformed into high-dimensional sparse binary features [8, 9]. Mathematically, encoding vector of i-th feature group is formularized as $t_i \in R^{K_i}$. K_i denotes the dimensionality of feature group i, which means feature group i contains K_i unique APIs. $t_i[j]$ is the j-th element of t_i and $t_i[j] \in \{0, 1\}$, $\sum_{j=1}^{K_i} t_i[j] = k$. Vector t_i with $k = 1$ refers to one-hot encoding and $k > 1$ refers to multi-hot encoding. Then one instance can be represent as $\mathrm{x} = [t_1^T, t_2^T, \ldots t_M^T]^T$ in a group-wise manner, where M is number of feature groups, $\sum_{i=1}^{M} K_i = K$, K is dimensionality of the entire feature space. In this way, the aforementioned instance with one-hot encoding and multi-hot encoding of features are illustrated as:

$$\underbrace{[0, \ldots, 1, \ldots, 0]}_{N^M = \text{PropRover}} \quad \underbrace{[0, \ldots, 1, \ldots, 1, \ldots, 0]}_{N^A = \{\text{FeedBurner,GoogleMaps}\}}$$

The whole feature set used in our system is described in Table 1. It is composed of four categories, among which Mashup's MemberAPIs features are typically multi-hot encoding vectors and contain rich information of Mashup preferences. Note that in our setting, there are no combination features. We capture the interaction of features with deep neural network.

Table 1. Simple indications of the representation of feature representation.

Category	Feature Group GROUP	Dimension	Type	Ids per Instance
Mashup Features	Name	$\sim 10^4$	One-hot	1
	Tag	$\sim 10^3$	One-hot	1
	Cate	$\sim 10^3$	One-hot	1
Mashups' MemberAPIs features	APIs_Names	$\sim 10^2$	Multi-hot	$\sim 10^2$
	APIs_Tags	~ 10	Multi-hot	~ 10
	APIs_Cates	~ 10	Multi-hot	~ 10
Candidate API features	API_Name	$\sim 10^4$	One-hot	1
	API_Tag	$\sim 10^3$	One-hot	1
	API_Cate	$\sim 10^3$	One-hot	1
Content features	Map	$\sim 10^2$	One-hot	1
	Game	$\sim 10^2$	One-hot	1

2.2 Deep Interest Network

In this section, we will introduce the framework of Deep Interest Network (DIN). The architecture of it can be illustrated in the Fig. 2, which consists of several parts:

Embedding Layer. As the inputs are high dimensional binary vectors, embedding layer is used to transform them into low dimensional dense representations. For the i-th feature group of t_i, let $W^i = \left[w_1^i, \ldots, w_j^i, \ldots, w_{1K_t}^i\right] \in R^{D \times K_t}$ represent the i-th embedding dictionary, where $w_j^i \in R^D$ is an embedding vector with dimensionality of D. Embedding operation follows the table lookup mechanism, as illustrated in Fig. 2.

- If t_i is one-hot vector with j-th element $t_i[j] = 1$, the embedded representation of t_i is a single embedding vector $\mathrm{t}_i = w_j^i$.
- If t_i is multi-hot vector with $t_i[j] = 1$ for $j \in \{i_1, i_2, \ldots, i_k\}$, the embedded representation of t_i is a list of embedding vectors: $\{e_{i_1}, e_{i_2}, \ldots e_{i_k}\} = \left\{w_{i_1}^i, w_{i_2}^i, \ldots w_{i_k}^i\right\}$.

Pooling Layer and Concat Layer. Notice that different Mashups have different numbers of APIs. So that the number of non-zero values for multi-hot behavioral feature vector t_i varies across instances, causing the lengths of the corresponding list of embedding vectors to be variable. As fully connected networks can only handle fixed-length inputs, it is a common practice [8, 10] to transform the list of embedding vectors via a pooling layer to get a fixed-length vector:

$$e_i = pooling(e_{i_1}, e_{i_2}, \ldots e_{i_k}) \tag{1}$$

average pooling, which apply element-wise sum/average operations to the list of embedding vectors. Both embedding and pooling layers operate in a group-wise manner, mapping the original sparse features into multiple fixed length representation vectors. Then all the vectors are concatenated together to obtain the overall representation vector for the instance.

Activation Unit. From the above steps, we obtain a fixed-length representation vector of Mashup composition by pooling all the embedding vectors over the Mashup composition feature group, as Eq. (1). This representation vector stays the same for a given Mashup, in regardless of what candidate APIs are. In order to solve this problem, DIN pay attention to the representation of locally activated intentions to recommend APIs for Mashup. Instead of expressing all Mashup's diverse composition with the same vector, DIN adaptively calculate the representation vector of Mashup's composition by taking into consideration the relevance of existing composition to recommend candidate APIs for Mashup. And this representation vector varies over different candidate APIs, so that we can select novel APIs for Mashup incrementally based on the composition relationship information.

From the Fig. 2, we can observe that DIN introduces a novel designed local activation unit. Specifically, activation units are applied on the Mashup composition features, which performs as a weighted sum pooling to adaptively calculate Mashup representation v_U given a candidate API A, as shown in Eq. (2):

$$v_U(A) = f(v_A, e_1, e_2, \ldots, e_H) = \sum\nolimits_{j=1}^{H} a(e_j, v_A)e_j = \sum\nolimits_{j=1}^{H} w_j e_j \tag{2}$$

where $\{e_1, e_2, \ldots, e_H\}$ is the list of embedding vectors of composition of Mashup U with length of H, v_A is the embedding vector of API A. In this way, $v_U(A)$ varies over

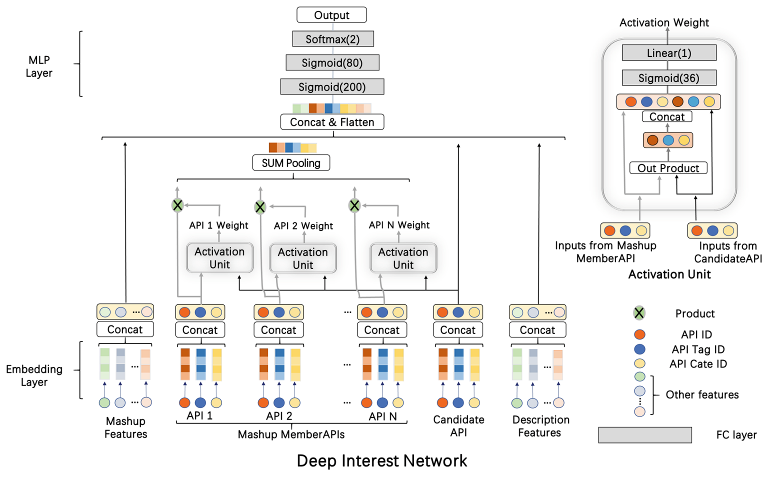


Fig. 2. DIN model structure.

different APIs. a(·) is a feed-forward network with output as the activation weight, as illustrated in Fig. 2. Apart from the two input embedding vectors, a(·) adds the out product of them to feed into the subsequent network, which is an explicit knowledge to help relevance modeling. Local activation unit of Eq. (2) shares similar ideas with attention methods which are developed in NMT task [11]. However different from traditional method, the constraint of $\sum_i w_i = 1$ is relaxed in Eq. (2), aiming to reserve the intensity of Mashup composition. That is, normalization with softmax on the output of a(·) is abandoned. Instead, value of $\sum_i w_i$ is treated as an approximation of the intensity of activated Mashup composition to some degree. For example, if a Google Maps API has been chosen for creating a Mashup of travel class. Given two candidate APIs of Bing Maps and World Weather Online, World Weather Online may get larger value of v_U (higher intensity of preference) than Bing Maps, because it complements the function of this Mashup and avoids choosing category-similar APIs for the Mashup. Traditional attention methods lose the resolution on the numerical scale of v_U by normalizing of the output of a(·). We have tried LSTM to model Mashup's invoked APIs dataset in the sequential manner. But it shows no improvement, we leave it for future research.

MLP. Given the concatenated dense representation vector, fully connected layers are used to learn the combination of features automatically. Recently developed methods [8, 12, 13] focus on designing structures of MLP for better information extraction.

Loss. The objective function used in base model is the negative log-likelihood function defined as:

$$L = -\frac{1}{N}\sum_{(x,y)\in S}\left(y\log p(x) + (1-y)\log(1-p(x))\right) \qquad (3)$$

where S is the training set of size N, with x as the input of the network and $y \in \{0, 1\}$ as the real label, $p(x)$ is the output of the network after the softmax layer, representing the predicted probability of sample x being recommended.

3 Experiment

3.1 Dataset Description and Doc2simu Preprocess

Dataset Description. To evaluate the performance of different APIs recommendation methods, we crawled 6415 real Mashups which invoke 1595 APIs from the ProgrammableWeb site and the overall statistics of our datasets is show in Table 2. For each Mashups or APIs, we firstly obtained their descriptive text and then performed a preprocessing process to get their standard description information. Figure 3 presents the statistics of APIs distribution in Mashups on the crawled dataset. From the Fig. 3, we can see that, 53.1%/25.1%/10.4% Mashups respectively invoke 1/2/3 APIs. Totally, more than 99% Mashups invoke 1–10 APIs. Therefore, we report experiment results obtained by recommending 1 to 10 APIs for target Mashup in this section.

Table 2. Statistic of our ProgrammableWeb dataset.

Projects	Mashup	API
Number of entities	6415	1595
Number of categories	375	127
Number of tags	996	964

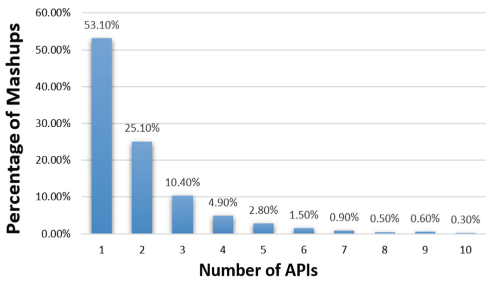


Fig. 3. Web APIs distribution of Mashups in the crawled dataset.

Doc2simu Preprocess. As the Fig. 3 shows, the dataset mentioned above is relatively small and most Mashups invoked only a small amount APIs which to a great extent will cause over-fitting in industrial depth network, and it may not be tolerated for our recommendation system. To remit this problem, we set up a Doc2simu model to expand our dataset. We followed several steps to clean and preprocess them.

First, we retrieved the three sections for each invoked APIs, including the name, primary category, primary tag and description. Then we processed the description document of all APIs by using tokenizer and stemming, meanwhile we removed those illegal characters including digits and special characters (e.g., &, % and $, etc.), and removed general or stop words in the end. The rest of the words were validated using dictionary.

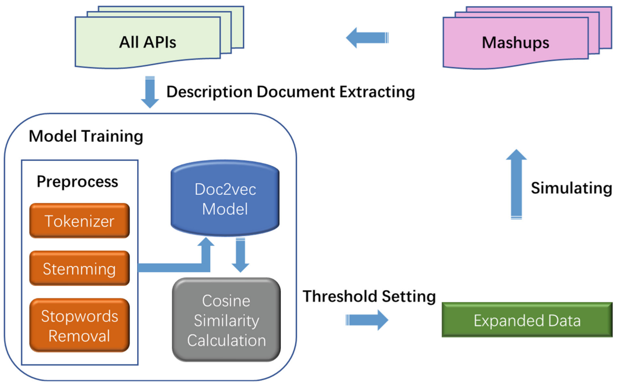


Fig. 4. Doc2simu preprocess.

Next, we put the ultima corresponding description document of each API into the Doc2vec [14] model for training and get the corresponding word vector, then the cosine similarity between each API and all other APIs word vectors is calculated, and the semantic similarity matrix is obtained. Finally, we select APIs, whose cosine similarity is more than 0.88 with the member APIs of each Mashups, as the extended simulation dataset of the corresponding Mashup. Cosine similarity is calculated as follows:

$$\cos(x, y) = \frac{\sum_{i=1}^{n}(x_i \times y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2} \times \sqrt{\sum_{i=1}^{n}(y_i)^2}} \tag{4}$$

where x_i and y_i represent the elements in the word vector between two different APIs x and y. Figure 4 gives a detailed introduction to the Doc2simu Preprocess. In addition, in order to ensure that the composition relationship of Mashups can be taken into account when training our model, we must filter out Mashups which contain three or more APIs.

After the above treatment, our dataset has become rich, and with more than 3 MemberAPIs for each Mashups. Features include API_id, cate_id, Mashup's invoked APIs_id_list and cate_id_list. Let all MemberAPIs of a Mashup be $(\mathrm{b}_1, \mathrm{b}_2, \ldots, \mathrm{b}_k, \ldots, \mathrm{b}_n)$, the task is to predict the $(k+1)$-th MemberAPIs by making use of the first k MemberAPIs. Training dataset is generated with k = 1, 2, …, n-2 for each Mashups. In the test dataset, we predict the last one given the first n-1 MemberAPIs. For all models, we use SGD as the optimizer with exponential decay, in which learning rate starts at 1 and decay rate is set to 0.1. The activation function is set to be sigmoid function.

3.2 Metrics

In recommendation field, Area Under Receiver Operator Characteristic Curve (**AUC**) is a widely used metric [15]. It measures the goodness of order by ranking all the APIs with recommendation, including intra-Mashups and inter-Mashups orders. A variation of Mashups weighted AUC is introduced in [16, 17] which measures the goodness of intra- Mashups order by averaging AUC over Mashups. We adapt this metric in our experiments. For simplicity, we still refer it as AUC. It is calculated as follows:

$$AUC = \frac{\sum_{i=1}^{n} \#impression_i \times AUC_i}{\sum_{i=1}^{n} \#impression_i} \tag{5}$$

where n is the number of Mashups, $\#impression_i$ and AUC_i are the number of impressions and AUC corresponding to the i-th Mashup.

Besides, we introduce Average Precision (**AP**) to evaluate the performance of all methods. AP is calculated as the area under the precision-recall curve. AP considers two measurements (i.e., precision and recall) simultaneously. It has been widely used in Information Retrieval [18] and Computer Vision [19]. Hence AP is defined as:

$$AP = \frac{1}{2}\sum_{i} (Pre(i) + Pre(i-1)) \times (Re(i) - Re(i-1)) \tag{6}$$

where $Pre(i)$ and $Re(i)$ are the precision and recall at the i-th threshold, respectively. Larger AUC and AP values indicate better performance.

3.3 Performance Comparison

We compare DINRec with the following strong baselines that are designed for Service Recommendation:

- LR [20]. Logistic regression (LR) is a widely used shallow model before deep networks for recommendation task. We implement it as a weak baseline.
- NMF (nonnegative matrix factorization) [21]. This approach employs matrix factorization to user-item matrix with a constraint that the factorized matrix is positive.
- FM [19]. This approach is the traditional factorization machine. It concatenates user id and item id as sparsity feature, and learns the interactions between users and items to complete the user-item matrix.
- Wide&Deep [8]. In real industrial applications, Wide&Deep model has been widely accepted. It consists of two parts: (i) wide model, which handles the manually designed cross product features, (ii) deep model, which automatically extracts nonlinear relations among features. Wide&Deep needs expertise feature engineering on the input of the “wide” module. We follow the practice in [13] to take cross-product of Mashups composition and candidates as wide inputs. For example, in our dataset, it refers to the cross-product of Mashup rated APIs and candidate APIs.
- DeepFM [13]. This approach combines the power of factorization machines for recommendation and deep learning for feature learning in a new neural network architecture.

In this part, we conduct experiments by randomly removing a part of Mashup-API pairs from the Mashup-API interaction matrix to make the matrix with different densities (i.e., 10% to 90%) in diverse models. For example, 10% denotes that we remove 90% entries on the Mashup-API matrix. And then we set the 10% entries as training set, the remaining 90% entries as testing set [22]. The results of performance comparison on our dataset is shown in Table 3. Obviously, all the deep networks beat LR model significantly, which indeed demonstrates the power of deep learning. DeepFM with specially designed structures preforms better than Wide&Deep. The performance of FM is better than NMF due to learning the interactions between Mashups and APIs. In addition, the performance of DeepFM is better than FM due to applying deep neural network to learn the high-dimensional interactions between Mashups and APIs. DINRec performs best among all the competitors. We owe this to the design of local activation unit structure in DIN. DIN pays attentions to the locally related Mashup composition relationship by soft-searching for parts of Mashup invoked APIs that are relevant to candidate API. With this mechanism, DIN obtains an adaptively varying representation of Mashup composition relationship, greatly improving the expressive ability of model compared with other deep networks. The table only presents the results of training data sparsity that is 10%, 20%, 80% and 90%, all results and impact of training data sparsity will be described and discussed in the next subsection.

3.4 Impact of Training Dataset Sparsity

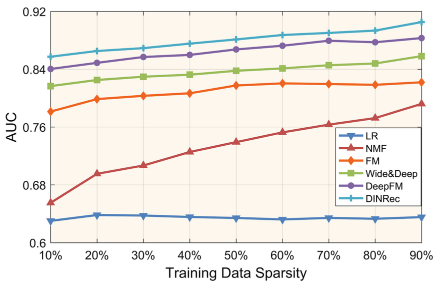
The training dataset sparsity is an important factor to impact recommendation performance. It represents how much information considered by Mashup on APIs we can utilize. To study impact of training data density, we set it from 10% to 90% with a step value of 10%. From Fig. 5, it can be found that our DINRec model achieves the best performance under all training data density. The performance of FM based approaches (e.g., FM and DeepFM) is better than LR and NMF. The performance of Wide&Deep is only worse than DINRec and DeepFM. Moreover, the AUC and AP values grow up with the increasing of training dataset sparsity. It is reasonable because when there is more training dataset, there is more information between Mashups and APIs will be collected, which is benefit for improving the recommendation accuracy.

3.5 Impact of Cosine Similarity Setting

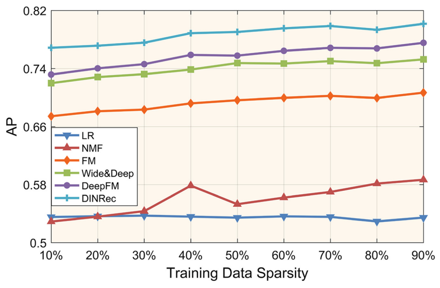
In this subsection, we performed an empirical study on the effect of different cosine similarity setting of DINRec on the results. As mentioned in Section 1, it is necessary to find a suitable cosine similarity threshold when we extend our datasets by using Doc2simu model. To investigate the effect of this threshold, we performed an experiment under different setting of the threshold including 0.80, 0.82, 0.84, 0.86, 0.88, 0.90, 0.92, 0.94, 0.96. The results are listed in Fig. 6. It can be observed that when increasing similarity setting value from 0.80 to 0.96, the AUC and AP value show an upward trend at first, and then a downward trend. Obviously, the best performance of AUC and AP value is achieved when the value is set as 0.88. This phenomenon demonstrates that larger similarity setting values bring better recommendation results. But why the AUC and AP values drop when the cosine similarity value is greater than

Table 3. Performance comparison.

Methods	Sparsity of training dataset							
	Training dataset = 10%		Training dataset = 20%		Training dataset = 80%		Training dataset = 90%	
	AUC	AP	AUC	AP	AUC	AP	AUC	AP
LR	0.6302	0.5353	0.6383	0.5364	0.6331	0.5293	0.6354	0.5345
NMF	0.6552	0.5291	0.6953	0.5358	0.7724	0.5814	0.7921	0.5867
FM	0.7814	0.6742	0.7987	0.6811	0.8184	0.6994	0.8219	0.7067
Wide&Deep	0.8166	0.7198	0.8251	0.7282	0.8479	0.7473	0.8581	0.7526
DeepFM	0.8403	0.7317	0.8488	0.7402	0.8773	0.7677	0.8831	0.7754
DINRec	**0.8573**	**0.7686**	**0.8652**	**0.7714**	**0.8934**	**0.7934**	**0.9052**	**0.8016**

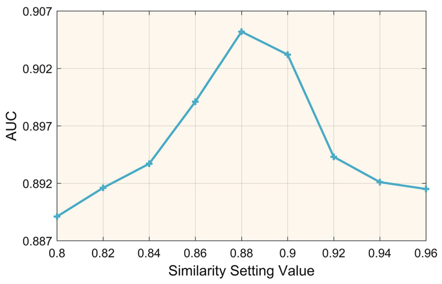


(a) AUC

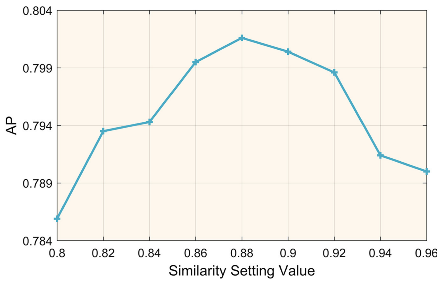


(b) AP

Fig. 5. Impact of training dataset sparsity.



(a) AUC



(b) AP

Fig. 6. Impact of cosine similarity setting.

0.88 is a question that worth thinking about. Indeed, in theory, the higher the similarity the higher the final result should be better, but it can' be ignored that the corresponding amount of training dataset will be less. In the depth learning model, the amount of training dataset is proportional to the good results. For example, when the similarity is set to 0.86, there are 199314 training datasets, but when the similarity is set to 0.96, the training dataset is only 43832.

4 Related Works

Web API recommendation technique plays an important role in service-oriented computing and effectively improves the quality of service discovery [23]. A number of research works have been done on Web API recommendation. They are mainly classified into three types: Collaborative filtering-based methods, Semantic based methods, and network based methods.

Collaborative filtering-based methods make use of user activities and past interactions to learn preferences and generate recommendations. [24] incorporated functional interest, QoS preference and diversity feature to recommend top-N diversified Web services to users. [25] proposed a collaborative filtering approach to predict missing QoS based on the information of similar Web users and services. [26] incorporate user, topic, and service-related latent factors into service discovery and recommendation.

The semantic based approaches aimed at finding the highest matching degree services via semantic similarity computation. [1] proposed a semantic content-based recommendation approach by analyzing the context of intended service. [26] considered simultaneously both rating dataset and semantic content dataset of Web services using a probabilistic generative model. [25] proposed a semantic-based service discovery framework, consisting of user model, context model, service model and a service discovery process. The similarity usually calculates from services' functionality description with some topic model, such as LDA topic model. [27] presented a recommendation system to design Mashup applications, relying on the multi-dimensional information, such as similar Mashups, similar Web APIs, cooccurrence and popularity of Web APIs. [28] advanced the current state of the art for Web API search and ranking from mashups developers' point of view, by addressing two key issues: multi-dimensional modeling and multi-dimensional framework for selection.

The network based approaches consist of two parts: social network and information network. The social network based approaches tend to apply user interest, social relationship and link prediction. [29] proposed a combined approach that improves description-based techniques with these social ranking measures. [30] proposed to combine current discovery techniques (exploration) with social information (exploitation). [31] proposed a social-aware service recommendation model by exploring multi-dimensional social relationships among potential users, topics, Mashups, and services. [27] presented an approach based on user interest from their Mashup usage history and social relationships information. [32] proposed a social network-based service recommendation method with trust enhancement by employing matrix factorization and random walk algorithm. The information network based approaches mainly employ different kinds of information and multiple semantic meanings of meta paths to recommend service. [33] proposed an efficient consistent regularization framework to enhance Mashup discovery by leveraging HIN between Mashups and their components. [34] proposed to recommend services for Mashup creation by exploiting different types of relationships in service related HIN. Inspired by the above approaches and in view of their shortcomings, we propose a novel recommendation approach that integrates Mashup functional semantics to composition structure approach.

5 Conclusion and Future Work

This paper introduces an effective service recommendation approach for Mashup creation based on DIN. Excessive reliance on functional semantic information in previous research work is a bottleneck for capturing the diversity of Mashups composition relationship. To improve the expressive ability of the traditional models, a novel approach named DINRec is proposed to activate related Mashup composition relationships and obtain an adaptive representation vector for prior selected APIs which varies over different candidate APIs. Besides, a novel technique is introduced to help training industrial deep networks with small-scale dataset and further improve the performance of DINRec. Our method was examined on extended ProgrammableWeb dataset. The results demonstrate that our method outperforms several state-of-the art methods. In future work, neoteric activation unit and textual features of APIs under our framework deserves further investigation.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China under Grant 61572187, Grant 61872139, Grant 61772193 and Grant 61702181, in part by the Natural Science Foundation of Hunan Province under Grant 2017JJ2098, Grant 2018JJ2136, Grant 2018JJ3190 and Grant 2018JJ2139, and in part by the Educational Commission of Hunan Province of China under Grant 17C0642.

References

1. Zhong, Y., Fan, Y., Huang, K., Tan, W., Zhang, J.: Time-aware service recommendation for mashup creation. IEEE Trans. Serv. Comput. **8**(3), 356–368 (2014)
2. Cao, B., Liu, X.F., Liu, J., Tang, M.: Domain-aware Mashup service clustering based on LDA topic model from multiple data sources. Inf. Softw. Technol. **90**, 40–54 (2017)
3. Bai, B., Fan, Y., Tan, W., Zhang, J.: SR-LDA: Mining effective representations for generating service ecosystem knowledge maps. In: 2017 IEEE International Conference on Services Computing (SCC), pp. 124–131 (2017)
4. Cao, B., Liu, J., Wen, Y., Li, H., Xiao, Q., Chen, J.: QoS-aware service recommendation based on relational topic model and factorization machines for IoT Mashup applications. J. Parallel Distrib. Comput. (2018)
5. Li, C., Zhang, R., Huai, J., Sun, H.: A novel approach for API recommendation in mashup development. In: 2014 IEEE International Conference on Web Services, pp. 289–296 (2014)
6. Chen, Q., Sokolova, M.: Word2Vec and Doc2Vec in unsupervised sentiment analysis of clinical discharge summaries. arXiv preprint arXiv:1805.00352 (2018)
7. Zhou, G., et al.: Deep interest network for click-through rate prediction. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1059–1068, July 2018
8. Cheng, H.T., et al.: Wide & deep learning for recommender systems. In: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (ACM), pp. 7–10 (2016)
9. Shan, Y., Hoens, T.R., Jiao, J., Wang, H., Yu, D., Mao, J.C.: Deep crossing: web-scale modeling without manually crafted combinatorial features. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 255–262 (2016)

10. Covington, P., Adams, J., Sargin, E.: Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM Conference on Recommender Systems, pp. 191–198 (2016)
11. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
12. Qu, Y., et al.: Product-based neural networks for user response prediction. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), pp. 1149–1154 (2016)
13. Guo, H., Tang, R., Ye, Y., Li, Z., He, X.: DeepFM: a factorization-machine based neural network for CTR prediction. arXiv preprint arXiv:1703.04247 (2017)
14. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning, pp. 1188–1196 (2014)
15. Fawcett, T.: An introduction to ROC analysis. Pattern Recogn. Lett. **27**(8), 861–887 (2006)
16. Zhu, H., et al.: Optimized cost per click in taobao display advertising. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 2191–2200 (2017)
17. He, R., McAuley, J.: Ups and downs: modeling the visual evolution of fashion trends with one-class collaborative filtering. In: Proceedings of the 25th International Conference on World Wide Web, International World Wide Web Conferences Steering Committee, pp. 507–517 (2016)
18. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 271–278 (2007)
19. Rendle, S.: Factorization machines. In: 2010 IEEE International Conference on Data Mining, pp. 995–1000 (2010)
20. Hosmer Jr., D.W., Lemeshow, S., Sturdivant, R.X.: Applied Logistic Regression, vol. 398. Wiley, Hoboken (2013)
21. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems, pp. 556–562 (2001)
22. Xie, F., Chen, L., Ye, Y., Zheng, Z., Lin, X.: Factorization machine based service recommendation on heterogeneous information networks. In: 2018 IEEE International Conference on Web Services (ICWS), pp. 115–122 (2018)
23. Xia, B., Fan, Y., Tan, W., Huang, K., Zhang, J., Wu, C.: Category-aware API clustering and distributed recommendation for automatic Mashup creation. IEEE Trans. Serv. Comput. **8**(5), 674–687 (2014)
24. Klusch, M., Fries, B., Sycara, K.: Automated semantic web service discovery with OWLS-MX. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 915–922 (2006)
25. Xu, S.Y., Raahemi, B.: A semantic-based service discovery framework for collaborative environments. Int. J. Simul. Modelling **15**(1), 83–96 (2016)
26. Yao, L., Sheng, Q.Z., Ngu, A.H., Yu, J., Segev, A.: Unified collaborative and content-based web service recommendation. IEEE Trans. Serv. Comput. **8**(3), 453–466 (2014)
27. Cao, B., Liu, J., Tang, M., Zheng, Z., Wang, G.: Mashup service recommendation based on user interest and social network. In: 2013 IEEE 20th International Conference on Web Services, pp. 99–106 (2013)
28. Bianchini, D., De Antonellis, V., Melchiori, M.: WISeR: a multi-dimensional framework for searching and ranking web APIs. TWEB **11**(3), 19:1–19:32 (2017)
29. Tapia, B., Torres, R., Astudillo, H.: Simplifying Mashup component selection with a combined similarity- and social-based technique. In: Proceedings of the 5th International Workshop on Web APIs and Service Mashups, p. 8 (2011)

30. Torres, R., Tapia, B., Astudillo, H.: Improving Web API discovery by leveraging social information. In: Proceedings of the IEEE International Conference on Web Services, pp. 744–745 (2011)
31. Xu, W., Cao, J., Hu, L., Wang, J., Li, M.: A social-aware service recommendation approach for Mashup creation. In: 2013 IEEE 20th International Conference on Web Services, pp. 107–114 (2013)
32. Deng, S., Huang, L., Xu, G.: Social network-based service recommendation with trust enhancement. Expert Syst. Appl. **41**(18), 8075–8084 (2014)
33. Wan, Y., Chen, L., Yu, Q., Liang, T., Wu, J.: Incorporating heterogeneous information for Mashup discovery with consistent regularization. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 436–448 (2016)
34. Liang, T., Chen, L., Wu, J., Dong, H., Bouguettaya, A.: Meta-path based service recommendation in heterogeneous information networks. In: Sheng, Q.Z., Stroulia, E., Tata, S., Bhiri, S. (eds.) ICSOC 2016. LNCS, vol. 9936, pp. 371–386. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46295-0_23