A Deep Recommendation Framework for Completely New Users in Mashup Creation

Yanmei Zhang[1(✉)], Jinglin Su[1], and Shiping Chen[2]

[1] Information School, Central University of Finance and Economics, Beijing 100081, China
Zhangym@cufe.edu.cn
[2] Software and Computational Systems CSIRO Data61, Sydney, Australia
shiping.chen@data61.csiro.au

Abstract. When service business is in evolution from B2B to B2C model, a cold-start problem raises for service composition due to the completely new clients with no historical records. Therefore, it is of great importance to solve the cold-start problem brought by completely new users. In this paper, we propose a recommendation framework for completely new users in Mashup creation based on deep-learning technology. Firstly, this framework extracts the mapping relationship between Mashup description and APIs offline by the deep neural network. Then, when the completely new users have the Mashup demands online, the matching APIs are recommended for them by using the mapping relationship. The experimental results with real-world datasets show that our proposed model outperforms the state-of-the-art ones in term of both accuracy and recall rate. The accuracy of the proposed method is 1.34 times higher than that of the state-of-the-art methods, and the recall rate is 1.55 times higher than that of the state-of-the-art methods. Moreover, considering that the new user history invocational data is very sparse, the performance of the proposed method can be greatly improved on the denser dataset.

Keywords: Service recommendation · Recommendation framework · Completely new user · Deep neural network · Mashup

1 Introduction

Web Mashups are Web applications developed using the contents and services available online [1]. Compared with traditional "developer-centric" composition technologies, such as BPEI and WSCI, Mashup provides a flexible and easy-of-use way for service composition on Web [2]. As the largest and most active collection of Web APIs and Mashups, the Programmable Web consists of more than 20,000 APIs until Jan-2020 [3]. Both the B2C business model and massive service information bring new challenges to service recommendation in the process of Mashup creation. First, there are lots of new users without any service invocation records. Second, some service Mashup platforms (e.g., Programmable Web) store composite service information (e.g., which APIs a Mashup is composed of), instead of any history of user invocation so that which the

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 550–566, 2021.
https://doi.org/10.1007/978-3-030-67537-0_33

corresponding historical data are quite scarce. Without any historical service invocation raises new challenges for service composition and recommendation.

The types of service recommendation models are increasingly diverse now. Firstly, demands matching methods [4–7] usually use Natural Language Processing (NLP) technology for demand matching. However, it cannot accurately describe the complex relationship between users and services. Secondly, some methods [8–10] mainly deal with the original matrix and expand the original scoring matrix with implicit feedback. However, for completely new users, the implicit feedback information is insufficient to effectively expand the original matrix. The third type of methods [11–14] focuses on the relationship of user-service, which mainly improves the process of user-service relationship modeling. But inadequate initial information makes it hard to build accurate models for new users. How to describe the complex relationship and make the model more suitable for new users becomes an intractable issue.

This paper proposes a novel deep recommendation framework for completely new users. The framework is divided into two parts: (a) offline learning and (b) online recommendation. In the offline part, the mapping relationship is extracted between Mashup description and APIs by deep learning techniques. In the online part, the User-API matrix is predicted for recommendation by the user demands and the mapping relationship obtained from the offline part. The contributions of this paper are highlighted as follows:

1. We propose a novel deep recommendation framework for completely new users, which combines the technologies of user feature extraction, matrix decomposition and learning mapping relationship by the neural network. The framework is low coupling so that each part can be updated or replaced independently.
2. Large experiments on the real data sets have been conducted and the experimental results prove that the accuracy of the proposed method has been improved, which is 1.34 times higher than the state-of-the-art methods, and the recall rate has been improved, which is 1.55 times higher than the state-of-the-art methods.

The rest of this paper is organized as follows: Sect. 2 presents the related work. Section 3 describes the proposed framework. Section 4 analyzes the experimental results on real data set. The last part draws a conclusion.

2 Related Work

According to the recommendation algorithm framework, the related work is divided into the following three categories:

2.1 Methods Focusing on Requirements Matching

Texts play an important role in the recommended system. It is one of the major directions of method improvement to accurately mine the user demands by converting the natural language into machine language.

Traditional NLP models such as LDA(Latent Dirichlet Allocation) [15], Word2Vec [16] and Doc2Vec [17], eventually extract long-text natural language as a word vector. But some fixed phrases have unique meanings of the words themselves. Gu et al. [4] proposed a method to decompose text into discourse units rather than word vectors in NLP so that the original meaning can be maintained for analysis. Then the relationship matrix of services can be generated based on discourse analysis and LSI (Latent Semantic Index) model for recommendation. But this method shows unclear relationship definitions. Some scholars have noticed the traditional NLP has a mediocre interpretability record. Xiong et al. [5] extract the semantic relationship of text as a "verb-noun-verb" triplet and generates a corresponding "object-attribute-value" triplet semantic map for recommendation. Although this method has better interpretability, it may lead to many complex sentences that are difficult to be parsed in the demand description of completely new users. Lin et al. [6] used a non-negative matrix decomposition method TNMF based on word correlation matrix. However, this method cannot provide effective query based on word vectors, and the recommendations could not accurately perform on small data sets. LIAN Tao et al. [7] proposed a recommendation algorithm based on the mixed LDA, which transports the historical message to pseudo-document information for analyzing. The method extracts detailed features by LDA and predicts the blank terms in the matrix by neighbor fields. Although this method defined the complex relationship between users and projects not so good, it provided important ideas for this paper. LDA is a topic model proposed by Blei et al. [15], that is divided into different levels according to documents, topics and words. After continuous improvement, LDA has become one of the most classical text analysis models. So, we choose the LDA model to extract more accurate features of the text.

2.2 Methods Focusing on Matrix

Most cold-start recommendation methods focus on how to effectively use the original matrix containing information about users and services. Song et al. [8] supplement the user feature matrix based on the user's click to learn his product preference online, build an online product cluster tree, and improved the accuracy by continuously collecting feedback. However, the life of service recommendation API is too short to seek the optimal solution through long-term exploration. Most methods use other sources of information to make up for the lack of rating data. Barjasteh et al. [9] proposed to use the similarity between users and items to alleviate the problem of cold start. This method generates the evaluation submatrix by excluding cold-start users and items from the original evaluation matrix, and then processes the cold-start matrix according to the similar matrix. But this method needs to have enough information about cold-start, and the recovery effect of the 0-1 matrix in the problem is unknown. Other scholars like Zhou et al. [10] proposed an iterative optimization method based on functional matrix factorization (FMF). The method iterates between the construction of the decision tree and the extraction of potential configuration files and considers the regularization scheme of tree structure. But this method pays more attention to matrix factorization technology, ignores the content features, as an important data source. Rendle et al. [11] combined the BPR(Bayesian Personalized Ranking) with matrix decomposition technique and KNN for recommending. This method optimized the original matrix by

increasing the probability of user preference and provided an important idea for 0-1 matrix decomposition in this paper.

2.3 Methods Focusing on Relationship

Data relationship mining is the main part of recommendation method. It is one of the key points of recommendation method, i.e. how to construct the mapping relationship between users and services.

Yuan et al. [12] proposed a method for pairing a cold start item with the original item. But this method involves less numerical operations and the accuracy of the method is mediocre. Obadic et al. [13] proposed a method based on DNN. The DNN learns the mapping relationship directly between the content of the item and the potential factors obtained by the UI matrix. He et al. [18] developed a general framework named NCF, short for Neural network based Collaborative Filtering, which can express and generalize matrix factorization. The method leverages a multi-layer perceptron to learn the user-item interaction function. Xiong et al. [19] proposed a hybrid method named DHSR which aims to capture underlying complex interactions between mashups and services according to their invocation history and the corresponding functionalities.

The above methods show that the neural network accurately captures the relationship between services and mashups, thus brings better performance in service recommendation. Based on this observation, we decide to use DNN to learn relational mapping to improve recommendation accuracy.

There is no uniform specification for service descriptions, which results in either no service description or unclear service description making it difficult for service recommend systems to deliver high quality of recommendation services. Hao et al. [14] proposed a method based on target reconstruction service description (TRSD) which can recommend service items that are more in line relevant application scenarios for users. But if the new user enters the service platform for the first time, the text of user demand may be blurred, and the effect of reconstructing the hidden valuable information in the text may not be ideal.

The common mashup data stored in the service recommendation platform is of high value. Gao et al. [20] proposed to model the user's historical preferences for mashups and APIs separately in a single framework. Shi et al. [21] proposed a mashup-API-Tag recommendation model, which shows the potential value of tags. Mbipom et al. [22] proposed that the method of query could also be improved based on academic terms. Although these algorithms have some shortcomings, they provide some valuable ideas for our framework design.

In conclusion, the methods focusing on requirements matching by NLP can better mine user demands. Focusing on the original matrix can effectively alleviate the cold-start problem, but this method could not accurately mine the demands of users. Focusing on mapping relationship can learn the relationship more accurately, but it is easy to fall into difficulties in sparse data. In a word, current work cannot effectively solve the cold-start problem, i.e. lack of user data, caused by completely new users.

3 The Deep Recommendation Framework

First of all, in the III-A part, we explain the framework proposed in this paper and propose a method based on clustering to solve the case of sparse data. Then, the last three parts describe each component of the LDB model presented in this article.

3.1 Overall Framework

The novel deep recommendation framework we propose combines user feature extraction, matrix factorization and neural network learning mapping. Based on the selection of specific application technology, we call this method LBD (LDA + BPR + DNN) for short (see Fig. 1)

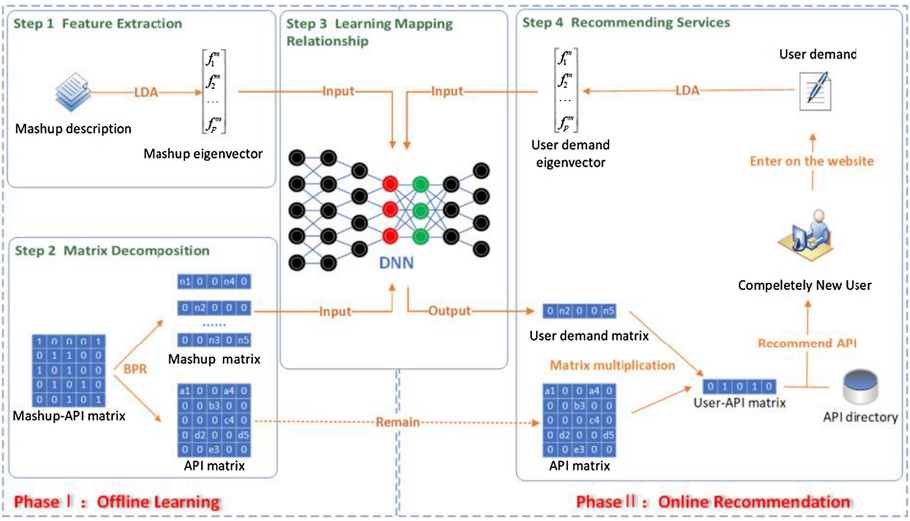

Fig. 1. LBD algorithm framework

LBD Method. As is shown in the figure, the method is divided into two parts: offline learning and online recommendation.

The offline part is shown in phase I. Mashup information is used to represent users' information in the context. The LDA model is used to process the limited service description text in the Mashup data and extract the Mashup description eigenvector. Then the Mashup-API invocation matrix is decomposed into the potential factor matrix of Mashup and API by using BPR (Bayesian personalized ranking). After getting data support from these two steps, the DNN was trained to learn the complex mapping of Mashup description eigenvectors and Mashup potential factor matrices offline.

The online part is as shown in phase II. When the new user inputs demand online which will be processed by the same LDA model to obtain user demand eigenvector. And the potential factor matrix of user demand is mapped by DNN, which has learned the

complex mapping relationship. Finally, a new User-API matrix is obtained for recommendation by multiplying the potential factor matrix of user demand and the potential factor matrix of API obtained by matrix decomposition in the offline phase.

LBD-Clustering Method. To effectively overcome the problem of high sparsity in datasets, we propose an LBD-Clustering method based on the improved LBD. The function of clustering is to divide the samples in the dataset into disjoint subsets. According to topic clustering method [23], the original data is compressed to reduce data sparsity, and the clustering results are used as the new input validation effect of the method.

For sparse data, we cluster the Mashup entries in the original data. In the feature extraction stage, LDA can directly give the topics to which each service description belongs. According to formula (1), the corresponding vectors of all descriptions belonging to the same topics are added together to get the eigenvectors of the topic. n is the total number of words in the document collection. t presents the topic collection.

$$\mathrm{V}(t) = \sum_{i=0}^{n} v(i)\{i \in t\} \tag{1}$$

3.2 Feature Extraction Based on Mashup Description

The service recommendation mainly relies on the service descriptions and user demand descriptions. Eigenvector extraction is the first step of the method to accurately define the relationship between user demand and service function. The extraction result has a great impact on the recommendation. Two important factors that determine the validity of extraction are data sources and vectorization. The following two aspects are introduced separately.

Data Sources Processing. Mashup classification and Mashup description are used in our method. The Mashup classification describes the application area and service type of the API. The Mashup description introduces the basic information of the service comprehensively such as the function and application scenario of the API and plays a major role in feature extraction. Since the completely new users only have natural language description of demand, we analyze the Mashup description as the text of user demand. When processing the Mashup description in a long text, we need to filter out the redundant information to improve the effect of feature extraction. As shown in Table 1, and we use the part-of-speech (POS) tag function to analyze part of speech, only retain vocabulary such as verbs, name forms, adjective adverbs and so on.

While processing the long text, we also obtained the Mashup classification tags. The classification tags are relatively independent, and there is no obvious relationship between the classification tags and the original long text of the Mashup description. We have integrated these tags with the long text as the service description of the API, and then extracted its eigenvector. In order to highlight the importance of the Mashup classification tags compared to general vocabularies, we have increased the number of times they appear in the text to improve their effect in feature extraction.

Text Vectorization for Demand Matching. The classification tags are relatively independent, and there is no obvious relationship between the classification tags and the

Table 1. Text preprocessing instructions.

Description	Example
Stop words	tool, solution, way, xml, platform, etc.
Frequency higher than 0.5	service, web, response, method, API, etc.
Frequency less than 0.01 and mistake	eventcategorie, myhurricane, phyloinformatic, hotukdeal, etc.
Meaningless word	Like prepositions, pronouns, articles etc.

original long text of Mashup description. Therefore, we use the LDA to extract the features of the Mashup description. The model of LDA is shown in Fig. 2 and the symbol convention is shown in Table 2. The feature extraction generates two files from the source document: the index table file and the vocabulary file. The Mashup description document is represented by multiple topics, which can not only solve the synonym problem but also effectively solve the problem of ambiguity.

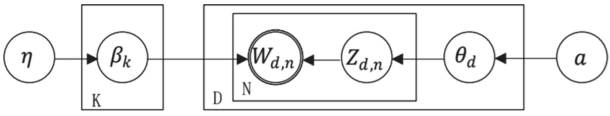

Fig. 2. LDA model diagram.

We extract text features according to the model shown in Fig. 2 α is the k-dimensional vector, which represents the proportion of the corresponding topic in the text. The text information is vectorized by using the formula (2) probability model:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta)p(w_n|z_n, \beta) \tag{2}$$

β, calculated by the formula, is the v-dimension vector, which represents the probability value of the words in each topic. The entire β is combined as a k × V matrix, feature extraction of the text is made based on the matrix. The feature extraction is based on the description information of Mashup in the training set, we integrate the description text information and establish the LDA model. Through continuous training, different topic-k in the description information of Mashup is extracted clearly, and finally we can obtain the description eigenvector η of Mashup.

We take the eigenvector of the Mashup service description as the input of the deep neural network, and then take the corresponding Mashup potential factor matrix as the specified output to train the neural network model. Therefore, the next main task is how to choose the method to get the better Mashup potential factor matrix.

3.3 Matrix Decomposition Based on BPR

The common matrix decomposition methods in the recommended system include SVD, NMF, PMF [24] and SVD++ [25]. However, completely new users without service

Table 2. Symbolic representation of LDA model diagram.

Symbol	Explain	Symbol	Explain
α	Dirichlet super parameters of text-topic distribution	W	Words in text
η	Dirichlet super parameters of topic-word distribution	N	Total words in text
θ	Text-topic distribution	D	Total text
β	Topic-word distribution	K	Total topics
Z	Topic of words in text	V	Total words in dictionary

invocation history can only be recommended by analyzing the Mashup data against the description of user demand. What's more, the Mashup API invocation matrix is a 0-1 matrix instead of the scoring matrix. However, NMF and SVD++ are more focused on matrix decomposition with implicit feedback such as click and browse. They are more suitable for the scoring matrix so that they are ineffective for this problem. In the process of research, we find that BPR can effectively solve the decomposition problem of the 0–1 matrix by sorting reconstruction during matrix decomposition. So, we choose BPR as the main technology of matrix decomposition.

BPR (Bayesian Personalized Ranking) is a sort method based on matrix decomposition. It processes the score of users (display feedback "1", implicit feedback "0") into a set of pair pairs <i, j>, where i is the API with a score of "1" and j is the API with a score of "0". Assuming that a user has L "1" scores and D "0" scores, the user has L × D pair pairs, so that the data set is represented by triple <u, i, j> and the physical meaning is: for user u, API interface I has higher priority than interface j. The training results are two decomposed matrices W and H. K is a smaller dimension that needs to be defined by itself. For any user u, we can calculate its ranking score for API interface i according to formula (3) as follows:

$$\bar{x}_{ui} = w_u * h_i = \sum_{f=1}^{k} w_{uf} h_{if} \tag{3}$$

From the ranking scores of user u for all APIs, we find the largest k-score, which is our real recommendation set top k API combination for user U. In the specific operation, we iterate the two submatrix W, h of the initial random according to the following formula (4, 5, 6) until the matrix converges to output the submatrix W, H:

$$w_{uf} = w_{uf} + \alpha\left(\sum_{(u,i,j)\in D} \frac{1}{1+e^{\bar{x}_{ui}-\bar{x}_{uj}}}\left(h_{if} - h_{jf}\right) + \lambda w_{uf}\right) \tag{4}$$

$$h_{if} = h_{if} + \alpha\left(\sum_{(u,i,j)\in D} \frac{1}{1+e^{\bar{x}_{ui}-\bar{x}_{uj}}} w_{uf} + \lambda h_{if}\right) \tag{5}$$

$$h_{if} = h_{if} + \alpha\left(\sum_{(u,i,j)\in D} \frac{1}{1+e^{\bar{x}_{ui}-\bar{x}_{uj}}} w_{uf} + \lambda h_{if}\right) \tag{6}$$

Where α is the gradient step, λ is the regularization parameter and K is the dimension of the decomposition matrix. We regard 1 in the 0–1 matrix as the items with the highest user preference ranking in the conventional U-I matrix, and the Mashup-API matrix is decomposed by using the derived formula. Experimental results show that due to the sparsity of data, the BPR model can effectively decompose the 0–1 matrix, thus improving the final effect of the method.

3.4 Learning Mapping Relationship by DNN

The DNN can be used to learn the mapping relationship between the eigenvector of Mashup description and the Mashup potential factor matrix, which is obtained by decomposing the Mashup-API matrix. Since the API potential factor matrix obtained by matrix decomposition is fixed, it is equivalent to learn the complex mapping relationship between Mashup description and API. The eigenvectors of service description extracted by LDA are simplified enough that DNN is suitable for most classification tasks and has excellent learning ability.

We decided to use the neural network with black box property to learn the relationship, and to accurately define its mapping relationship by fitting large-scale training data sets. The DNN is shown in Fig. 3.

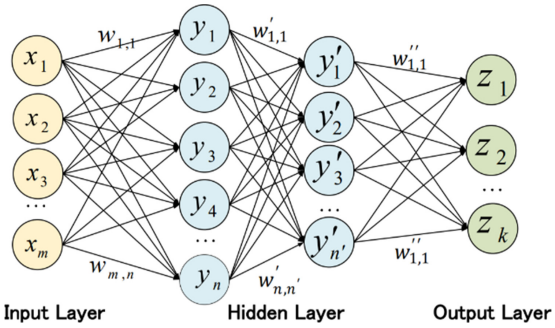

Fig. 3. DNN model.

The DNN is composed of many neurons, which are essentially the same as the perceptron. The neurons are trained by inputting x with weight W and activating function, that is, the linear relationship of multiple formulas (7) plus activating function δ (z), where B represents the threshold constant.

$$z = \sum w_i x_i + b \tag{7}$$

The weight w_i and the bias b are randomly initialized, then the training samples obtained in the earlier part are input into the neurons, the forward propagation has calculated the output. Then get the mean squared error by calculating between the target and the output. The error is propagated back to the hidden layers to update the parameters until the error to our expectation.

The number of layers in a neural network directly determines its performance ability. Although more layers may bring better effects, as the number of layers in a neural network deepens, the optimization function is more and more likely to fall into the local optimal solution, even deviate from the real global optimal solution. Sometimes, the performance of the deep network trained with limited data is not as good as that of the shallow network. Therefore, the layer number of DNN in our proposed model and the number of neurons in each layer will be the focus of the following discussion.

So far, the training stage of DNN has been completed, and the next step is to use the trained DNN to predict the services required by completely new users.

3.5 Prediction of Matching APIs for Completely New Users

After using LDA to extract the features of Mashup description, we can obtain the user demand eigenvector in the same vector space by using the same LDA to process the demanded text of a completely new user. Then, the Mashup demand eigenvector is input into the trained DNN, and the potential factor matrix of the user demand can be predicted. The latent factor matrix of Mashup demand is the same as the implicit space of API latent factor matrix obtained by BPR decomposition. The final User-API matrix can be obtained by multiplying the latent factor matrix of user demand and the latent factor matrix of API.

4 Experiment

The experiment will try to answer the following research questions:

RQ1: How to tune parameters to optimize the performance of the LBD framework?
RQ2: How does LBD perform in solving completely new user problems, compared with the state-of-the-art methods?
RQ3: How much does the performance of the LBD framework improve when the data set becomes denser?

The experiments are conducted on a 2.20 GHZ Core i7 processor and 16 GB RAM under Windows 10. All methods are implemented in Python Language.

4.1 Data Set Introduction

The real data set from Programmable Web is crawled for the experiment. The data set includes 6295 Mashup data where 1496 APIs are used. In addition to building the corresponding Mashup-API matrix through the data set, we also capture the corresponding description of Mashup, 4656 Mashup classification tags and five types of services. The statistical characteristics of the data are shown in Table 3.

4.2 Evaluation Metrics

To measure the recommended accuracy of the method we proposed, we adopt three classical metrics: precision, recall rate and F1-score. In the following comparison experiments, we used P, R and F1 respectively to represent them. The higher the calculation results of the three evaluation criteria, the better the recommendation effect of the method.

Table 3. Symbolic representation of LDA model diagram

Relations (X-Y)	Number of X	Number of Y	Number of (X-Y)
Mashup-API	6295	1496	13185
Mashup-Tag	6295	4656	6295
Mashup-Content	6295	6284	13185
Mashup-Type	6295	5	422

4.3 Baseline Approaches

To verify the superiority of our proposed method in solving completely new user problems, we select four state-of-the-art methods LDA-CF [7], BPR-KNN [11], NCF [18] and DHSR [19].

- BPR-KNN [11]: This method uses implicit feedback data with BPR to learn the k-nearest neighbor (KNN) model.
- LDA-CF [7]: This method uses LDA to analyze user text similarity and then makes recommendation based on collaborative filtering.
- NCF [18]: This method uses neural collaborative filtering(NCF) to learn the relationship between mashups and services, then recommends relevant service.
- DHSR [19]: This method uses a novel deep learning based hybrid approach to recommend Web service. The invocation interactions between mashups and services as well as their functionalities are seamlessly integrated into a deep neural network, which can be used to characterize the complex relations between mashups and services.

4.4 Experimental Results and Analysis

Parameter Tuning. *DNN structure optimization.* There are two main factors affecting the performance of the deep neural network model: the number of hidden layers, and the number of neurons contained in each hidden layer. So, we will try to find out the best DNN structure in this experiment. About the other parameters in the DNN models, we chose the same in each experiment. We choose the ReLU as the activating function and set the learning rate to 0.1.

To observe the influence of hidden layer number on model effect, we use neural network from 1 to 3 hidden layers to carry out experiments, and the results of neural network methods with different hidden layers are shown in Table 4.

Result: When the number of hidden layers is 2, the accuracy, recall rate and F1 value of the model are the highest. When the number of hidden layers increases to 3, the performance of the method decreases greatly.

Analysis: Because DNN has more hidden layers than the Perceptron, it can learn more complex non-linear relations. However, because the task of learning the relationship between Mashup description features and potential factors of Mashup is relatively simple for the neural network, two hidden layers are enough to learn the potential relationship, and more hidden layers will be used to fit the phenomenon.

Table 4. Performance of the number of hidden layers

Number of hidden layers	P	R	F1
1	0.481	0.352	0.406511
2	0.496	0.391	0.437285
3	0.463	0.325	0.381916

In addition, in view of the influence of the number of neurons in the hidden layer of the neural network on the effectiveness of the method, we have also carried out relevant parametric adjustment experiments. In order to facilitate the test, the number of neurons in the hidden layer is set to the same constant. On the premise of 2 hidden layers, the number of neurons in each hidden layer is set to 500, 1000 and 1500 respectively, and the performance is compared in Table 5.

Table 5. Performance of number of neurons

Number of neurons	P	R	F1
500	0.496	0.391	0.437285
1000	0.494	0.400	0.442058
1500	0.488	0.395	0.436602

Result: The accuracy decreases with the increase of the number of neurons in the hidden layer. When the number of neurons in the hidden layer is set to 1000, the method has the highest recall rate and F1 value.

Analysis: Because the complexity of describing text information is general, when the number of neurons in the hidden layer increases to 1500, the feature extraction process will have overfitting phenomenon. After comprehensive consideration, we take the DNN with two hidden layers and 1000 neurons in each layer as the final choice of LBD.

Parameter Optimization in Prediction Part. In the final recommendation prediction part, we search for some users being similar to new users through pre-processing and then recommend APIs. We compare the selection of similar users and the number of recommended APIs.

We use the control variable to optimize K and N. First, we ensure that the number K of recommended APIs is fixed, change the number of similar users N to carry out the control experiment. Second, select the best N value as the final choice, and then carry out the second round of control experiment for different API recommendation number K. The results are shown in Table 6 and Table 7.

Table 6. Performance of different number of similar users

With K = 20, the number of N	P	R	F1
10	0.183	0.253	0.212
20	0.191	0.292	0.231
30	0.230	0.345	0.276

Table 7. Performance of different recommended number of API

With N = 30, the number of K	P	R	F1
5	0.215	0.286	0.245
10	0.236	0.314	0.269
15	0.254	0.356	0.296
20	0.230	0.345	0.276
25	0.192	0.329	0.242

Data Set Partition. The data set partitioning will affect the result of the method, so we conducted a comparative experiment of data sets partitioning on the basic of DNN structure optimization. The data set is divided into the training set and test set according to the ratio of 8:2, 7:3 and 6:4 respectively. The experimental results are shown in Table 8.

Table 8. Performance of data set partition

Training set: Test set	P	R	F1
8:2	0.462	0.360	0.404672
7:3	0.478	0.392	0.430749
6:4	0.441	0.351	0.390886

Result: According to the ratio of training set and testing machine, the recall rate R reaches the highest when the experimental data set is divided into 7:3, and the method effect decreases when the ratio is too high or too low.

Analysis: When the partition ratio of training set and testing machine is 8:2, the method appears over fitting phenomenon, which is caused by the increase of the sparsity of training data set with the increase of training set. When the partition ratio of training set and testing machine is 6:4, the method appears under fitting phenomenon, which is because the training data is small, which is not enough to support the learning task of neural network. Therefore, in this method, the data set is divided into training set and test set in the ratio of 7:3.

Compared Experiment. In order to verify the performance superiority of our proposed method in predicting the demands of the completely new users, we select four baseline methods for comparison.

In this part, we choose the K = 15, N = 30 and take the average value of the code running results of 20 times as the last result of each method. For the baseline methods, we have also carried out parameter tuning experiments to get the best performance. The comparison results of LBD and other methods are shown in Fig. 4.

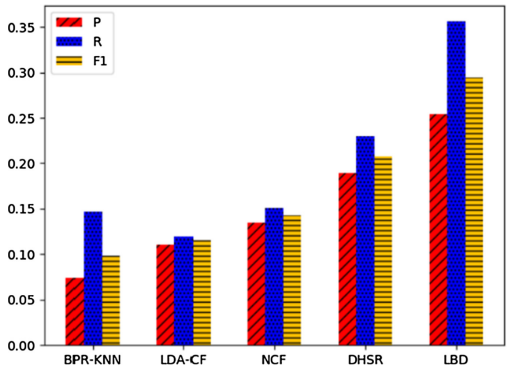

Fig. 4. Performance of different methods

Result: The performance of LBD is significantly improved compared with the other four methods in terms of precision and recall. The precision and recall rate of LBD is 1.34 times and 1.55 times respectively of best baseline method DHSR.

Analysis: BPR-KNN approach decomposes 0-1 matrix effectively and makes APIs recommendation using KNN method. LDA-CF method uses LDA model to analyze the similarity of service description text, and then recommends APIs based on collaborative filtering method. LDA-CF improves the performance better than the BPR-KNN, which proves the analysis of service description plays a key role for the content-based recommendation. However, these two methods fail to make use of the data comprehensively.

NCF approach captures the non-linear relationship between mashups and services based on a neural network framework. Compared with the BPR-KNN and LDA-CF, NCF can characterize the relations between mashups and services better thus it achieves better performance. DHSR approach adopts a hybrid deep learning based recommendation method. Though the DHSR shows that the hybrid recommend method has more advantages in service recommendation, our method LDB performs better than it.

Experiment on Data Sparsity. *Clustering of Data Sets.* In addition, the best clustering number of LBD-Clustering method is tested. We compared the Topic number selection method of clustering operations, and the results are shown in Table 9.

Table 9. Performance of different numbers of Topics

The number of Topic	P	R	F1
100	0.235	0.703	0.352
200	0.266	0.683	0.382
300	0.283	0.657	0.396
400	0.276	0.631	0.384
500	0.252	0.615	0.358

Result: When the original data set is clustered into 300 topics, the recall rate is the highest and the recommendation effect is the best.

Analysis: Due to the size of the data set and the average length of the description information, when the original data set is clustered into 300 topics, it has a better classification effect, so this optimal parameter is determined by the size of our data set.

Sparsity Experiment of LBD Method. The experimental results for data sparsity of LBD method are shown in Table 10, and the clustering method is referred to as LBD-Clustering.

Table 10. Performance of different method effects

Method	P	R	F1
LBD	0.254	0.356	0.295
LBD-clustering	0.283	0.657	0.396

Result: Compared with the original LBD method, LBD-Clustering has improved accuracy, recall rate and F1 score.

Analysis: From the comparison of the experimental results of LBD-Clustering and LBD, because LBD-Clustering carries out classification prediction based on the nature of the experimental data set, the recommendation effect has been greatly improved, which shows that the LBD method has certain expansion in classification prediction, it is also proved that with the increase of data density, the performance of our LBD method still has a lot of room to improve.

5 Conclusion

In order to solve the problem of insufficient information bring by the completely new users for recommendation, we propose a novel deep recommendation framework. At first, the deep neural network is used to extract the mapping relationship between Mashup description and APIs offline. Then, the mapping relationship between the Mashup

description extracted in advance can predict matching APIs for new users through their requirements. This framework can completely rely on the text of new users' demands to recommend services. The experimental results on real-world datasets show that the proposed LBD framework has greater improvement in terms of accuracy and recall than the state-of-the-art algorithms.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (Nos. 61602536, 61773415, 61672104).

References

1. Almarimi, N., Ouni, A., Bouktif, S., Mkaouer, M.W., Kula, R.G., Saied, M.A.: Web service API recommendation for automated mashup creation using multi-objective evolutionary search. Appl. Soft Comput. **85**, 105–830 (2019)
2. Liu, X., Hui, Y., Sun, W., et al.: Towards service composition based on mashup. In: IEEE Congress on Services. IEEE (2007)
3. Huang, K., Fan, Y., Tan, W.: An empirical study of programmable web: a network analysis on a service-mashup system. In: ICWS2012, pp. 552–559 (2012)
4. Gu, Q., Cao, J., Peng, Q.: Service package recommendation for mashup creation via mashup textual description mining. In: ICWS 2016, pp. 452–459 (2016)
5. Xiong, W., Wu, Z., Li, B., Gu, Q., Yuan, L., Hang, B.: Inferring service recommendation from natural language API descriptions. In: ICWS 2016, pp. 316–323 (2016)
6. Lin, C., Kalia, A.K., Xiao, J., Vukovic, M., Anerousis, N.: NL2API: a framework for bootstrapping service recommendation using natural language queries. In: ICWS 2018, pp. 235-242 (2018)
7. Lian, T., Ma, J., Wang, S., Cui, C.: LDA-CF: a mixture model for collaborative filtering. J. Chin. Inf. Process. **28**, 129–150 (2014)
8. Song, L., Tekin, C., Schaar, M.V.D.: Online learning in large-scale contextual recommender systems. IEEE Trans. Serv. Comput. **9**(3), 433–445 (2017)
9. Barjasteh, I., Forsati, R., Masrour, F., Esfahanian, A.-H., Radha, H.: Cold-start item and user recommendation with decoupled completion and transduction. In: Conference on Recommender Systems (2015)
10. Zhou, K., Yang, S.H., Zha, H.: Functional matrix factorizations for cold-start recommendation. In: Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, 25–29 July 2011. ACM (2011)
11. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. In: Conference on Uncertainty in Artificial Intelligence, pp. 452–461 (2009)
12. Yuan, J., Shalaby, W., Korayem, M., et al.: Solving cold-start problem in large-scale recommendation engines: a deep learning approach. In: IEEE International Conference on Big Data, pp. 1901–1910. IEEE (2017)
13. Obadić, I., Madjarov, G., Dimitrovski, I., Gjorgjevikj, D.: Addressing item-cold start problem in recommendation systems using model based approach and deep learning. In: Trajanov, D., Bakeva, V. (eds.) ICT Innovations 2017. CCIS, vol. 778, pp. 176–185. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67597-8_17
14. Hao, Y., Fan, Y., Tan, W., Zhang, J.: Service recommendation based on targeted reconstruction of service descriptions. In: ICWS 2017, pp. 285–292 (2017)

15. Blei, D.M., Ng, A.Y., Jordan, M.I., et al.: Latent dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: ICLR Workshop (2013)
17. Rong, X.: word2vec parameter learning explained. arXiv preprint arXiv:1411.2738 (2014)
18. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.-S.: Neural collaborativefiltering. In: Proceedings of the 26th International Conference on World Wide Web, pp. 173–182 (2017)
19. Xiong, R., Wang, J., Zhang, N., Ma, Y.: Deep hybrid collaborative filtering for web service recommendation. Expert Syst. Appl. **110**, 191–205 (2018)
20. Gao, W., Chen, L., Wu, J., Bouguettaya, A.: Joint modeling users, services, mashups and topics for service recommendation. In: ICWS 2016, pp. 260–267 (2016)
21. Shi, M., Liu, J., Zhou, D., Tang, M., Xie, F., Zhang, T.: A probabilistic topic model for mashup tag recommendation. In: ICWS 2016, pp. 444–451 (2016)
22. Mbipom, B., Massie, S., Craw, S.: An E-learning recommender that helps learners find the right materials. In: AAAI 2018, pp. 7928–7933 (2018)
23. Zhang, M.: Forward-stagewise clustering: an algorithm for convex clustering. Pattern Recognit. Lett. **128**, 283–289 (2019)
24. Mnih, A., Salakhutdinov, R.: Probabilistic matrix factorization. In: Neural Information Processing Systems, pp. 1257–1264 (2007)
25. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Knowledge Discovery and Data Mining, pp. 426–434 (2008)