Assessing User Interest in Web API Recommendation using Deep Learning Probabilistic Matrix Factorization

T. Ramathulasi[1], M. Rajasekhara Babu[2]*
Research Scholar, School of Computer Science & Engineering,Vellore Institute of Technology, India[1]
Professor, School of Computer Science & Engineering, Vellore Institute of Technology, India[2]

***Abstract*—Internet 2.0 Things connected to the Internet not only manage data supply through devices but also control the commands that flow through it. The communication technology created by the desired sensor is used by a new computing model so that the collected data appears in Web 2.0 for management. In addition to enhancing Sense efficiency through the simple IoT computing process, it is used in many cases for example video surveillance, and improved and intelligent manufacturing. Every fragment of the system is carefully continued and supervised in this process by software collection using a large number of recurs. An important process for this is to access web APIs from various public platforms in an efficient way. The use of different APIs by developers for the integration of different IoT devices and the deployment process required for this is unnecessary. Obtaining configured target APIs makes it easy to know where and how to get started with the workflow approach. Rapid industrial development can be achieved through this powerful API approach. But finding adequately powerful APIs from a large number of APIs has become a great challenge. However, due to the massive spike in the count of APIs, combining the two APIs has now become a major challenge. In this paper, for the time being, only the relationships between users and the API are considered. In this case, they had to face difficulties in extracting contextual value from their interpretation. So better accuracy could not be obtained due to this. The consequence of the user's time aspect on the cryptographic properties concerning the information collected from the API contextual description can be enhanced by the Deep Learning Probabilistic Matrix Factorization (DL-PMF) method, which improves the accuracy of the API recommendation in considering the cryptographic features of the user in the API recommendation. In this paper, we have used CNN (Convulsive Neural Network) for web elements such as APIs, and LSTM (Long-Term and Short-Term Memory) Network, which works with a diligent mechanism to find hidden features, to find hidden features that suit the tastes of the users. In conclusion, the combination of PMF (Probabilistic Matrix Factorization) evaluation of the recommended results was obtained as described above. The combination of DL-PMF method experimental results was found to be better than previous PMF, ConvMF, and other methods, thus improving the recommended accuracy.**

***Keywords*—Implicit feature; API's recommendation; IoT; collaborative filtering; matrix factorization**

I. Introduction

In this new era due to the huge explosion in data registration as an online platform, consumers have to face various big problems in big data usage. Retrieving large data and other necessary information by communicators using [1] technology in a very short period has not been very successful. In addition to providing services tailored to the personal preferences of different users, this method also answers the major problem of data overload. This order, which was recommended by a mature process developed by a highly researched cooperative filtering method, became highly publicized. This has been referred to as neighborhood and model-based practices [2]. Better results than neighborhood-based recommended policy outcomes can be obtained by recommending using PMF and SVD methods with model-based approaches [3]. The probability factor established by the model-based approaches PMF and SVD plays a major role in gradually improving the effectiveness of the recommended process. The PMF approach turned out to be effective in uncovering the hidden features of consumers by rating matrix but has failed to make effective and efficient utilization of this informational interpretation in terms of useful and vital customer information and services.

In this case, the recommendation system was applied through approaches undertaken by several researchers with the knowledge gained from the in-depth study. As a result, the recommended system has been improved by eliminating the flaws in the traditional recommendation method and utilizing information efficiently through complex approaches. There are still some errors that need to be corrected. Two CNNs (Conventional Neural Networks) were created in parallel to capture the underlying elements of Web API description information and the factor evaluation mechanism from the customer evaluation description [4]. The impact of consumer preferences on the recommendation system could not be ascertained due to the time factor not taking into account the consumer preferences in this model. For better recommendation, we have combined both the denoising autoencoder and the CDL model, which can effectively obtain encryption components through auxiliary information, and achieve better results [5]. The hierarchical Bayesian model approach is proposed later in this series.

The methods described by the above research have not been able to achieve better results in extracting relevant information from the content description. CNN and PMF have been used interchangeably, as described in the research papers [6] and [7] to identify latent properties as the basis of object content. But this does not allow us to find out the latent features of the user using the user information. The model [8], developed by combining CNN and latent factors, was successful in capturing users' object attributes but failed to achieve better-optimized hidden attributes. RNN (Repeated

*Corresponding Author.

Neural Network) assists in the detection of consumer data irregularities [9] and the output of received news messages for recommendation by taking what is collected and received as input for what was previously considered the Browsing History Helpline. Dimensionally we get output messages that have the same dimension size as the input in the feature learning process. This requires the use of optimized dimensionality. In addition to extracting latent features from the user information description, the time it takes to use them in the process is also calculated using the RNN-based model [10] proposed by the collaborative filtering algorithm. Failed to fully extract from the elements the comprehensive latent features based on the model described in this sequence.

It has now become a challenge to capture the latent features according to their real, precise needs according to the information in the description of the consumer items. Integrating the PMF model with Deep Learning has been proposed for its solution. The text descriptive for this system is given as input for recommendation and the whole process is done in three steps. The latent features were learned from the description of the elements in the first stage, the latent features of the users in the second stage, and finally, the total latent features were trained by merging the two latent features into the PMF. To this end, in the aforementioned paper, we have attached a technique with particular care to capture and better understand the exact latent nature of CNN from element description [11, 12]. Care is taken to remove useless information in the service description as well as to learn the latent features according to the tastes of the customers [13]. In this paper, LSTM has been adapted LSTM to find the latent features appropriate to the needs of real and accurate users [14, 15]. The impact of consumption time on consumer pre-history discussions and consumer tastes has been taken into account in the process of learning these hidden features. Finally, it was made possible by combining users' hidden features with PMF for content-based assessment of user preference using both elements and user description. The combination of ConvMF and PMF has been verified by proposing Deep Learning-based PMF for greatly improved accuracy in the recommendation and enhanced accuracy capability by experimenting on a data set that has been crawled from a website called Programmableweb.

II. Related Work

The section includes, the most widely used CF algorithms in service recommendation [16, 17, 18] are currently reviewed by Mac as algorithms relevant to service and API recommendations. In this paper, Mache proposes a hybrid CF algorithm to overcome the shortcomings of CF algorithms [17] following current traditional methods. For this purpose, the information of the web APIs was collected by the first collecting mechanism and a new method of weight calculation was used. This algorithm then improved the accuracy of the recommendation by combining the user and item-based CF model with different weights. The web API information description was collected by a mechanism proposed in [16]. Different weights were added to the user and item-based CF model through the new-weight calculation method.

The revised proposal has sparked renewed interest in APIs in recent years as a means of addressing the guideline's flaws. The algorithms in this sector are separated into API and Mashup suggestions. The authors of [19] advised mashups for consumers based on customer preferences and service social networks for Mashup Recommendation. Historical usage data from the mashup is used to determine user interest. For the mashup service, a social network of web services was developed using a description of the information included in the API and associated tags. The network has uncovered and integrated user interest in web services. As noted in [16], potential affiliates have been discovered as a set of API mashups produced by authors from among APIs. It also focuses on an excavation in the context of Mashup's proposal for a shared API invoice model, as implemented by the excavated association recommended model. The potential model here improves the streamlined latent correlation.

The framework based on the Knowledge Graph was submitted by the authors of [20] on the recommendation of the APIs. It first recognizes the invite relationship between APIs and also identifies key information related to categories and labels. This information is used to form the relationship component between entities. The input is part of the recommendation in the knowledge graph. The authors later [21] proposed the recommendation of the framework obtained by applying random walks on the Knowledge Graph. The information on the most useful APIs was first compiled by the authors using the knowledge graph and the nodes were filtered based on the semantic information contained. The relationship between the candidate and the nominee was re-evaluated by the authors using random walks. Different networks are used to resolve different relationships represented by multiple links and multiple types of objects. A diverse information network is built by the authors by combining the API information with the information contained in [22] about the mashup and its related features, and group preference can be obtained for the mashup by following a clustering approach to the information in the network. Fixed issues with authors [23] currently having problems getting permission to search for API representative services by search engines. In addition, the recovery process was proposed by the new authors with the help of a collaborative network of services and the visualization technique provided for easier browsing.

Recommendations for APIs are made by traditional CF algorithms. The CF algorithm was proposed by the author in a paper [24] due to an item-based API recommendation. Item similarity can be achieved by making the customer group based on customers who have similar preferences. Furthermore, the CF algorithm applies a specific item based on each user group. It has been confirmed that better performance than neighbor-based CF can be achieved with MF model-based CF, with the authors attempting to assess user preferences when presenting invitations in [25] and terms of APIs and mashups. This assessment was made possible by the authors incorporating user preferences into the MF model to create a new recommendation model.

Even though recent procedures have a high level of suggestion accuracy, the content of the data has not been accomplished. Because most consumers use different objects at

various times throughout the day, Chen et al. [26] argued that contextual factors are crucial in the brilliant object suggestion of the Internet of Things. The researchers projected a time-aware, bright object recommendation model integrating user interest across times and creative object resemblance. Duan et al. [27] suggested JointRec, a deep learning-based joint video streaming recommendation system for IoT, to deliver precise video ways for a better marginal of users. Item descriptor information in IoT is generally diverse and multimodal, according to Huang et al. [28]. The paper presents a new heterogeneous representation learning-based strategy to improve predictive performance in IoT. In [29], Hu et al. suggested automotive ad-hoc network representation training for IoT recommendations since trajectory data better knowledge of human movement patterns. In contrast to these approaches, we investigate the potential linkages between users and API attributes in this research. To uncover implicit information, we use collaborative learning. Finally, we propose multiple models for IIoT devices and API recommendations grounded on the association between users and APIs. A clustering technique called Word-Embedded Clustering (AUG-LDA) is suggested in [30] for improved outcomes with a significant influence on word vector quality. This method can offer more efficiency and accuracy in mashup service innovation. DMM-Gibbs (Dirichlet Multinomial Mixer with Gibbs) predicts better clustering performance by reducing the number of features represented [31].

III. Prior Knowledge

In this segment, we critically summarize the most popular collaborative filtering technique called “Matrix Factorization (MF)” and the “Convolutional Neural Network (CNN)”.

The MF method operates in a process-sharing cryptocurrency that calculates the strengths of the user-object relationship with the rating given by the consumer on the object, intending to collect hidden patterns of the object [3]. Let's assume we have N consumers and S web services and a ranking index for user service is $R \in \mathbb{R}^{N \times S}$. In MF, k-dimensional structures, $u_i \in \mathbb{R}^k$and $s_j \in \mathbb{R}^k$ are represented as the latent structures of consumer i and service j. A user i and service 's ranking r_{ij} is approximated by the inner-product of the related latent user i and service j models (i.e. $r_{ij} \approx \hat{r}_{ij} = u_i^T s_j$). One possible simple way to better minimize the damage function $\mathcal{L}$ value is to train Latin models. It has a policy of having total-squared-error terms between ratings and actual ratings, and L_2 regularized terms. To better avoid the major problem of over-fitting, try to improve it as shown below:

$$\mathcal{L} = \sum_i^N \sum_j^S I_{ij} \left(s_{ij} - u_i^T s_j\right)^2 + \lambda_u \sum_i^N \|u_i\|^2 + \lambda_v \sum_j^S \|s_j\|^2 \quad (1)$$

Where an indicator function I_{ij} exists such that, if the user is given a rating it is 1 otherwise 0.

A "Convolutional Neural Network" (CNN) is called feed-forward and is a type of neural network that varies with the following components: 1) To construct local attributes, the convolution layer is used; 2) The pooling layer is used to represent only the data as an abbreviated representation, called the sub-template. Via activation functions, all of the representatives with the highest score from the previous layer send local features to the next layer.

Although CNN was originally used to develop computer-vision-based applications, its main idea was to retrieve information and, in particular, to return the NLP search query [32], to model sentences [33], and additionally perform NLP tasks in traditional methods [34]. Here NLP has led to significant changes in the structure of CNN for the betterment of work and better results in the performance of the NLP's diverse work.

However, in reality, CNN is not yet well used in some areas, especially in the recommendation system. To the best of our knowledge, van den Ord et al. CNN [35] was first applied to music recommendation. There they analyzed the songs via CNN from a sound analysis perspective. And the desired pattern for calculating ratings is based on the latent pattern of the song obtained by sound CNN. The CNN model has not shown enough improvement in processing the documents and retrieving the sound signal. The reason for this is that the quality of the documents, the sound signals, and the features around them differ inherently with some variation. At a given time i.e. a signal with a short time difference is equal to the surrounding signals. And the word in a document at a certain point differs greatly from the meanings of the surrounding words. So, in the end, there is a great need for CNN to have such different variations and different structures to influence the quality of features locally. Also, the cooperative information of the model is not fully available. In particular, service latent features are determined by the analysis results obtained by CNN to convey the audio signal rather than the collaborative information. Therefore, "weighted matrix factorization (WMF)” [36] alone cannot improve the performance of the entire recommendation system, and it is one of the traditional collaborative filter MF-based methods that deals only with datasets with latent features.

IV. Methodology of DL-PMF Model

Matrix factors can be considered models by PMF as a potential solution. User preference in service selection is explained by the use of a potential combination of low-dimensional feature vectors, and potential properties are mapped by API and customer element information based on the potential angle to the low-dimensional mapping space. The process of identifying matrix factors as a potential solution was used as a model by the PMF. To assist in the dimensional mapping of a small area, it has used a low dimensional feature vector to highlight important information potential features in the API and user information. Specifically, this simple combination makes it possible to take customer preference into account in the process description recommended by the API. In this paper, the main purpose is to get better results using the proposed DL-PMF model. For this, the service / APIs and customer information through the continuous iteration process are used to find hidden vectors with the help of the PMF method. The performance of this model can be seen in a step-by-step sequence as shown in Table I below and Fig. 1 below.

TABLE I. THE ARRANGEMENT OF STEPS IN THE DEEP-LEARNING BASED PMF MODEL

Algorithm	:	The arrangement of steps in the Deep-Learning based PMF model
Step 1	:	Incidentally, both U and W_S variables are generated and configured as rating matrix R.
Step 2	:	Attention-CNN updates S depending on R and U.
Step 3	:	S is trained using the PMF method while U is trained using the LSTM algorithm using Attention-CNN and LSTM.
Step 4	:	Steps 2 and 3 should be repeated until the loop convergence is reached.

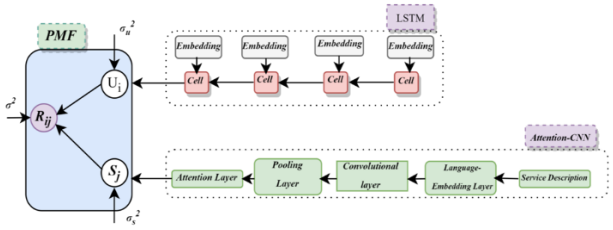

Fig. 1. DL-PMF model framework

The hidden feature vectors are detected by the Attention-CNN used from the service information. The DL-PMF model is part of a CNN-based network that is interested in this process. We were able to obtain accurate results based on the achievement of the natural language processing process for application programs to text emotion classification as well as analysis. Potential features can be easily extracted from global information using CNN [4, 6, 5, 7]. The focus is on each characteristic of the customer information description and the features of most of the API information description, in line with the visual approach and human information. These features and their meaning can also be obtained due to the in-depth study that is part of the attention process. The combination of CNN and the Focus Approach makes it possible to easily get better features as well as remove useless features from the Global Information filter. We will see in the proposed DL-PMF model how to retrieve hidden feature vectors using user information by LSTM with memory function. Current and past interests can be fulfilled based on history record information by remembering precise guarantees for customer perceptions and timing, and by remembering the implicit connection features between their respective APIs. The internal construction of Attention-CNN, as well as LSTM, is an overview as shown below.

A. The Structure of Attention-CNN Network

The Attention-CNN architecture is known as the network architecture seen in Fig. 2 and this architecture's main objective is to identify and acquire its tacit vectors based on the details contained in the document's service specification. Here, in the service text description, we convert the data into digital quantity and take it as input to the Attention-CNN model. Complex instances are correlated with word embedding and hence controllable measurements can be obtained. In this proposed model, we've introduced the embedding process.

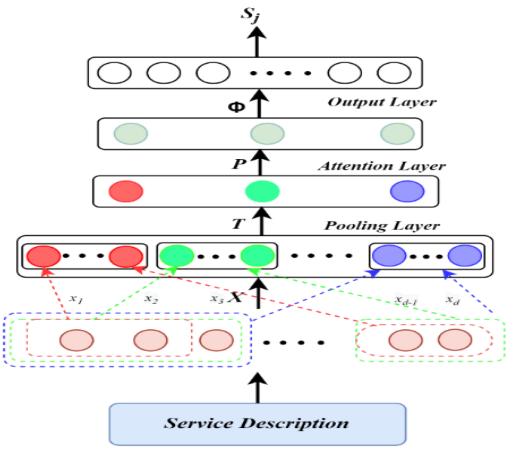

Fig. 2. Architecture of attention-CNN

1) Language embedding layer: The language embedding layer collects the raw material for the next level of the convolution layer and converts it into a dense numeric matrix and gives it as input. The word description of the service is used to input the words in the documents into d word vectors, converting the document into a matrix and input to the language embedding layer. Each word in the document is converted to a low-dimensional numeric vector using the pre-trained word2vec tool. We get the output of the layer as $X \in R^{k \times d}$, denoting the maximum length d of the vectors.

$$X = \begin{bmatrix} | & | & | & & | & | & | & & | \\ x_1 & x_2 & x_3 & \dots & x_{i-1} & x_i & x_{i+1} & \dots & x_d \\ | & | & | & & | & | & | & & | \end{bmatrix} \quad (2)$$

Where k is each word vector's dimension, $x_i \in R^k$.

2) Convolution layer: The key role of this layer has been described as gathering feature information related to the text. It is used to locate text attributes with different sharing weights using various convolution kernels. The length of the kernel window is then defined as l, and Eq. (3) is used to express the mutual weight matrix. The properties gained by the convolution kernel's layer j are expressed as follows.

$$t^j = [t_1^j, t_2^j, \dots., t_i^j, \dots, t_{d-l+1}^j] \quad (3)$$

The output eigenvalue obtained by a convolution kernel concerning region ix is denoted by t_i^j, and $t_i^j = f\left(W^j \otimes X_{(i:(i+l-1))} + b^j\right)$. The paper will specify the configuration operator as $\otimes$, the non-linear activation function as $f(\cdot)$, and the activation function as ReLU. From the i column of X to $i + l - 1$ column by $X_{(i:(i+l-1))}$, and we identified the mapping matrix W^j and the bias by b^j.

This layer's outcome is:

$$\mathrm{T} = [t^1, t^2, t^3, \dots, t^i, \dots, t^n] \quad (4)$$

Here, n is the kernel convolution number.

3) Pooling layer: This current layer captures the representative properties through the convolution layer on the

back, and constructs a fixed-length vector of equal length with the document variable following the pooling operation. The variable length obtained by the convolution layer represents a contextual feature vector with $d - l + 1$. This constant change in length makes it difficult to form subsequent layers. So we used max-pooling in this case. The maximum contextual characteristic of a piece of information is derived from each contextual feature vector that represents it and is represented by a shorter fixed-length vector. To substitute the output of that position in the network, this layer utilizes the general features of an adjacent output at a region. The layer's outcome is:

$$P = [p_1, p_2, \dots, p_i, \dots, p_n] \quad (5)$$

Where $P_i = \max(t^i)$, $max(\cdot)$ is the maximum in t^i.

4) Attention Layer: At this stage, the layer has been introduced with more attention-grabbing approaches. The service weight is calculated from the information weight in each term based on the attention matrix approach. This not only highlights some of the important features but also leaves out unnecessary information using weight value in terms of size. So the corresponding weight for each word is calculated according to (6).

$$\alpha_i = g(W^{attn} * (p_i^{attn})^T + b^{attn}) \quad (6)$$

Where $P_i^{attn} = \left[P_{i+\frac{-m+1}{2}}, P_{i+\frac{-m+3}{2}}, \dots, P_i, \dots, P_{i+\frac{m-1}{2}}, \right], m$ is the magnitude of the sliding window. $W^{attn} \in R^{l \times m}$, $b^{attn} \in R$. $g(\cdot)$ is a function of non-linear activation.

The outcome of this current layer is

$$\Phi = [\omega_1, \omega_2, \dots, \omega_i, \dots, \omega_n] \quad (7)$$

Where $\omega_i \in \alpha_i P_i$.

5) Output layer: After obtaining high-level attributes from the previous layer in this sequence, they must be changed gradually for specific work as soon as the output layer is reached. That is, high-dimensional attributes are converted into a specific size feature vector and a latent attribute representation for customer service after reaching the layer before the output layer. The output to the output layer can be specified as follows.

$$s_j = tanh(W_o \Phi + b_o) \quad (8)$$

From the above equation, we reference to $W_o \in R^{D \times n}$ as the mapping matrix, $b_o \in R^D$ as the bias, and $s_j \in R^D$ as the service's latent characteristic vector j.

B. LSTM Neural Network Structure

The correlation between customer and service features is very strong. This paper considers it a series of tasks to identify user characteristics for this correlation exploration. In this series of processes, we use the LSTM model to thoroughly search and dig into the correlations between expected features and to identify service users individually. Neural networks are specifically designed as "Recurrent Neural Networks (RNN)" and are used to process sequences of data. This allows each layer to be output to the next layer as well as its current state for processing and RNNs are good at mining the interrelationships between cryptographic patterns. RNN showed better results in gradient disappearance and gradient explosion. Developed an evolutionary version of RNN called "Long Short-Term Memory (LSTM)" [37].

Another special type of RNN, LSTM, replaces the complex neural network structure by internally filling cells with low-cell neurons [38] Here, the LSTM of RNN is used to explore the properties of services that are interrelated through this framework and to independently evaluate the features of the services. Also, in evaluating subsequent characteristics, the LSTM internal structure indicates the secret state in the pre-history information and works better in the sequence modeling process. As input information, it takes complete sequence information, models it, and processes it into a single-dimensional vector. The inner component structure of the contributing LSTM-Cell can be described as shown in Fig. 3.

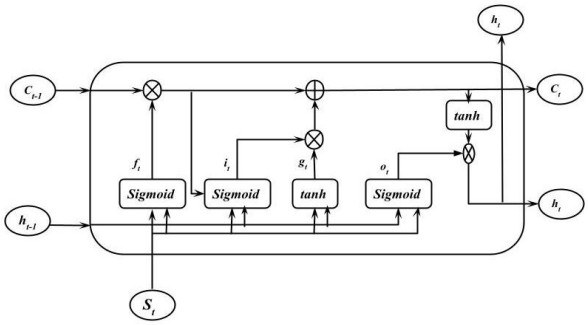

Fig. 3. Inner components of LSTM-Cell

Initially, LSTM, like RNN, considers service inputs to be interconnected and considered as a time series, and finally, its internal structure has become an excellent answer to the problem of bursting and disappearing gradient problems [16]. Below are the four significant aspects of the LSTM model. As shown in the figure: the input gate, forget gate, gate of the input module, and gate of the output. The gate and output gates of the module are used via the input gate to change the status of the cell, forget the maintenance gate, and control the deletion. The numerical method can be represented by the following equations for this procedure.

$$\left\{ \begin{array}{c} i_t = sigmiod(W_i * s_t + W_i * h_{t-1} + + b_i) \\ f_t = sigmiod(W_f * s_t + W_f * h_{t-1} + b_f) \\ o_t = sigmiod(W_o * s_t + W_o * h_{t-1} + b_o) \\ g_t = \tanh\left(W_g * s_t + W_g * h_{t-1} + b_g\right) \\ C_t = f_t \cdot C_{t-1} + i_t \cdot g_t \\ h_t = \tanh(g_t) \end{array} \right\} \quad (9)$$

Where the current cell status value, last time frame cell status value, and the update for the current cell status value are expressed by C_t, C_{t-1}, and g_t. Input gate, forget gate, and output gate, respectively, are the notations i_t, f_t, and o_t. With suitable input parameters, according to Eqs (9), the output value h_t is determined based on g_t and C_{t-1} values. Based on the difference between the output value and the actual value

after the back-propagation through time (BPTT) algorithm [39], all weights, including: W_i, W_f, W_o, and W_g, are modified.

This paper describes how the latent feature vector is derived from users' historical record information. The services used in the information in the usage history record over time need to be considered in the acquisition process and for this, the process of learning the latent feature vector can be considered as a time series approach. The answer to this approach is to use the LSTM architecture shown in Fig. 4 below to extract the latent feature vector from user information.

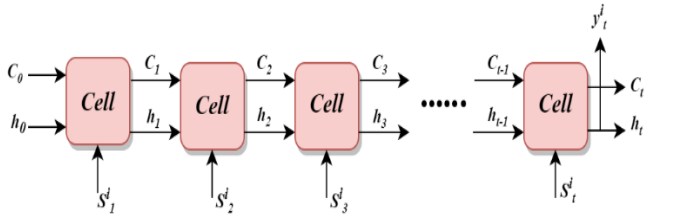

Fig. 4. LSTM External Architecture

The output and latent feature vectors of historical customer services can be obtained by input series, chronologically inserted into the LSTM model.

$$\begin{cases} C_t = \sigma\left(W_1 \cdot \left[h_{t-1}, s_t^i\right] + b_1\right) * \tanh\left(W_2 \cdot \left[h_{t-1}, s_t^i\right] + b_2\right) + \\ \sigma\left(W_3 \cdot \left[h_{t-1}, s_t^i\right] + b_3\right) * C_{t-1} \\ h_t = \sigma\left(W_4 \cdot \left[h_{t-1}, s_t^i\right] + b_4\right) * \tanh(C_t) \\ y_t^i = h_t \end{cases} \quad (10)$$

The mapping templates are denoted by W_1, W_2, W_3, and W_4 , the bias is denoted by b_1, b_2, b_3 and b_4 , and the cell state by C_t at t, the latent attribute vector relative to the user i by $y_t^i \in R^D$ and here $\sigma(\cdot)$ is the activation function related to the sigmoid.

C. Methodology for Proposed Model

In Fig. 1, the user i's latent feature vector is denoted as $U_i \in R^D$, the latent attribute vector of the service j as $S_j \in R^D$, and the user i rating of the service j as R_{ij}. The variation of the Gaussian distribution of U_i and S_j is denoted by σ_U^2 and σ_S^2, respectively.

The probability of a true conditional rating matrix [14] is determined as follows:

$$P(R|U,S,\sigma^2) = \prod_i^N \prod_j^M \left[N\left(R_{ij}\middle|U_i^T S_j, \sigma^2\right)\right]^{I_{ij}} \quad (11)$$

The probability density of the Gaussian distribution with the 's mean and σ^2 variance is given by $N(x|\mu,\sigma^2)$ suggested here. If the user i rates the service j, the index function I_{ij} value is set to 1, otherwise, its value is set to 0 will allocate. Assuming that the total number of users is N and the services are M, we identify the user's feature matrix by $U \in R^{D\times N}$ and the latent feature matrix of services by $S \in R^{D\times M}$.

PMF is a sequence of consumers' and services' latent feature vectors that are independent of each other. Also, before the Gaussian distribution, the likelihood assumes the formula shown below for a zero-mean value.

$$(U|\sigma_U^2) = \prod_{i=1}^N N(U_i|0,\sigma_U^2 I) \quad (12)$$

$$P(U|\sigma_S^2) = \prod_{j=1}^M N(S_i|0,\sigma_S^2 I) \quad (13)$$

It is possible to substitute the Attention-CNN and LSTM outputs with equations (12) and (13).

$$P(U|W_U, X_U, \sigma_U^2) = \prod_{i=1}^N N(U_i|y^i, \sigma_U^2 I) \quad (14)$$

$$P(U|W_S, X_S, \sigma_S^2) = \prod_{j=1}^M N\left(S_i|s_j, \sigma_S^2 I\right) \quad (15)$$

In LSTM, W_U is used to denote the whole mapping matrix and its biases, and W_S is used to denote both mapping matrices with their biases. We can understand that the latent potential of the latent feature matrix for customers and services can be satisfied using the following formula, according to the Bayesian formula.

$$P\left(U,S,W_U,W_V\middle|R,X,\sigma^2,\sigma_U^2,\sigma_S^2,\sigma_{W_U}^2,\sigma_{W_S}^2\right) \approx P(R|U,S,\sigma^2) \times P(U|W_U,X_U,\sigma_U^2) \times P(W_U|\sigma_U^2) \times P(S|W_S,X_S,\sigma_S^2) \times P(W_S|\sigma_S^2) \quad (16)$$

For the prior probability distribution $P\left(W_U\middle|\sigma_{W_U}^2\right) = \prod_\theta^\Omega N\left(0,w_\theta\middle|\sigma_{W_U}^2 I\right)$ relative to W_U , the total representation mapping matrix in the LSTM network can be observed as W_U. Priority probability distribution concerning W_S is $P\left(W_S\middle|\sigma_{W_S}^2\right) = \prod_\phi^\Re N\left(0,w_\phi\middle|\sigma_{W_S}^2 I\right)$, the total mapping matrix represented in Attention-CNN Can be observed as W_S. Also X_U in LSTM and X_S in Attention-CNN are referred to as the input set respectively.

In estimating the maximum posterior likelihood, the estimation matrix R found based on the DL-PMF model offered a nice solution.

$$max\left(P\left(U,S,W_U,W_V\middle|R,X,\sigma^2,\sigma_U^2,\sigma_S^2,\sigma_{W_U}^2,\sigma_{W_S}^2\right)\right) = max\left(P(R|U,S,\sigma^2) \times P(U|W_U,X_U,\sigma_U^2) \times P(W_U|\sigma_U^2) \times P(S|W_S,X_S,\sigma_S^2) \times P(W_S|\sigma_S^2)\right) \quad (17)$$

We consider the natural logarithm for the above equation for convenience. In

$$\left(\mathrm{P}\left(\mathrm{U},\mathrm{S},\mathrm{W_U},\mathrm{W_V}\middle|\mathrm{R},\mathrm{X},\sigma^2,\sigma_\mathrm{U}^2,\sigma_\mathrm{S}^2,\sigma_{\mathrm{W_U}}^2,\sigma_{\mathrm{W_S}}^2\right)\right) = -\frac{1}{2\sigma^2}\sum_{\mathrm{i=1}}^{\mathrm{N}}\sum_{\mathrm{j=1}}^{\mathrm{M}}\mathrm{I_{ij}}\left(\mathrm{R_{ij}}-\mathrm{U_i^T S_j}\right)^2 - \frac{1}{2\sigma_\mathrm{S}^2}\sum_{\mathrm{j=1}}^{\mathrm{M}}\left\|\mathrm{S_j}-\mathrm{s_j}\right\|^2 - \frac{1}{2\sigma_\mathrm{U}^2}\sum_{\mathrm{i=1}}^{\mathrm{N}}\left\|\mathrm{U_i}-\mathrm{y^i}\right\|^2 - \frac{1}{2\sigma_{\mathrm{W_U}}^2}\sum_\theta^\Omega\|\mathrm{w_\theta}\|^2 - \frac{1}{2\sigma_{\mathrm{W_S}}^2}\sum_\phi^\Re\left\|\mathrm{w_\phi}\right\| + \mathrm{C} \quad (18)$$

The LSTM, Attention CNN mapping matrix numbers, are denoted as Ω and ℜ respectively, and the matrix entries in the equation (16) can be omitted since the LSTM, Attention CNN mapping matrices, are constructed randomly according to the distribution basis of Gaussian.

To make the maximum value of equation (18) equal to the minimum, the following equation is used:

$$f = \sum_{i=1}^N\sum_{j=1}^M I_{ij}\left(R_{ij}-U_i^T S_j\right)^2 + \chi\sum_{j=1}^M\left\|S_j-s_j\right\|^2 + \beta\frac{1}{2\sigma_U^2}\sum_{i=1}^N\left\|U_i-y^i\right\|^2 \quad (19)$$

Where $\chi = \sigma^2/\sigma_S^2, \beta = \sigma^2/\sigma_U^2$.

The algorithm of coordinate descent is used in this paper to find and solve the minimum value of f. The U_i and S_j updates are expressed as follows:

$$U_i \leftarrow \left(2\left(2\beta I_k + \sum_{j=1}^{M} I_{ij} S_j S_j^T\right)^{-1} * \left(\beta * y^i + \sum_{j=1}^{M} R_{ij} I_{ij} S_j\right)\right) \quad (20)$$

$$S_j \leftarrow \left(2\left(2\chi I_k + \sum_{i=1}^{M} I_{ij} U_i U_i^T\right)^{-1} * \left(\beta * v_j + \sum_{i=1}^{N} R_{ij} I_{ij} U_i\right)\right) \quad (21)$$

V. Experimental Evaluation

Initially, each value of the matrix in the process results indicated here is denoted by 0 or 1 and equated with the performance of the four other methods. In this case, any API followed by the user is marked as '0' and otherwise as '1'.

A. Datasets

We used as a package the existing API data from the website called *programmableweb* and the API dataset was scrambled with comprehensive information about them. We scrambled a count of 17412 API data for this data set. It includes a list of followers for each API, as well as developer information, a quick summary, a range of names, and a post date. And here we have extracted the new follow feature from the web along with customer information. Consumer information description is considered equivalent to the performance of the goods. As followers of 17412 APIs, we have compiled the data of 140,000 users for this purpose and we have included the details in Table II.

TABLE II. Statistics from Crawled Dataset

Scrambled Dataset	*Total (#)*
User profiles	17412
API's	22032
Invocation files	248530

All the data in the scrambled dataset is split into training, validation, and test sets to evaluate the results of the experiments we perform. Random selection of training sets to 90%, 80%, 70%, 60%, and 10% is possible due to the arrangement of the data settings. In this experiment, 90 data sets for training, 5 for validation, and 5 percent for test density were analyzed first, and for the other four data settings for 10% validation of the data in the data set, the remaining data for the test set was composed. I.e. training, certification, and test sets with 10%, 10%, and 80% concentrations were taken. Thus experiments were performed ten times for each group. We analyzed all the results for each group experiment with MAE (Mean Absolute Error) and RMSE (Root-mean-square error) values.

B. Evaluation Metrics

There has been an introduction to a recommended model to estimate a type. The goal is to determine the prediction accuracy of the API suggestion estimates based on the trials completed. We used the MAE and RMSE as evaluation criteria.

$$MAE = \frac{1}{N}\sum_{i,j}\left|R_{ij} - \hat{R}_{ij}\right| \quad (22)$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i,j}\left(R_{ij} - \hat{R}_{ij}\right)^2} \quad (23)$$

N stands for the numeral of tests, R_{ij} stands for the user's rating of service j, and $\hat{R}_{ij}$ stands for the user's estimated rating of service j using the stated method.

C. Parameter Setting

Since the model, we proposed above depends on the latent factor method (LFM), this model is particularly affected by two parameters. The first is the total quantity of hidden features: though its value is too high it cannot be understood by the model. Yet if this value is too low, we will get negative effects from the model. So, we took this number to be 10. The second introduced a parameter called regularization λ to adjust the weight of the term. As part of this, λ is set to 0.1, as well as the regularization value λ_u, λ_a is set to 0.01 related to user terms and related API terms.

D. Comparative Result Evaluation of Performance

To evaluate the effectiveness of the proposed strategy, we used the following methods and the experimental findings are listed in below Table III.

TABLE III. Experiment Results

Method	The density of the Training set (T)									
	T=90%		T=80%		T=70%		T=60%		T = 10%	
	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
PNF	0.391	0.555	0.411	0.571	0.426	0.584	0.441	0.596	0.635	0.739
NMF	0.265	0.531	0.283	0.564	0.315	0.623	0.395	0.747	0.402	0.818
ConvMF	0.468	0.684	0.469	0.685	0.471	0.69	0.481	0.696	0.482	0.719
Autoencoder	0.299	0.378	0.332	0.378	0.344	0.387	0.377	0.377	0.417	0.45
DL-PMF	0.151	0.204	0.163	0.223	0.176	0.246	0.193	0.276	0.424	0.646

1) PMF [40]: For collaborative filtering, Probabilistic Matrix Factorization incorporates ratings.

2) NMF (Non-Negative MF) [41]: A Non-Negative matrix factorization is modeled as the product of the two non-negative matrices.

3) ConvMF [7]: CNN is used to extract item information in Convolutional Matrix Factorization.

4) Autoencoder [42]: A multilayer neural network that is commonly used in feature learning to reduce dimensionality.

5) DL-PMF. Our method combines the PMF with a CNN and an LSTM network to predict recommendation results.

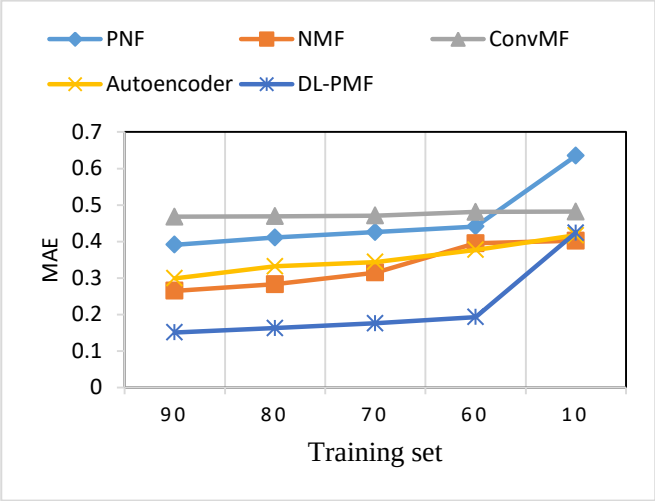

Fig. 5. Impact on MAE

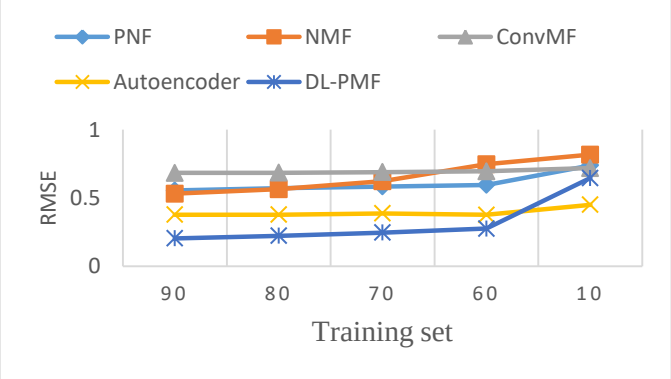

Fig. 6. Impact on RMSE

We have shown the line chart as shown in Fig. 5 and Fig. 6, clearly describing the model proposed by us by comparing the working methods of different methods with training sets of different densities and describing its typical performance. Fig. 5 and Fig. 6 show the MAE and RMSE values for training sets of different densities. The consistent performance of the test set with relatively different densities is maintained in the NMF approach. So, depending on this it is understood that the logistic regression here is relatively constant. The difference is huge. MAE and RMSE values are higher in the autoencoder model than in ConvMF when the concentration of the test set is 90%. In general, the value of this difference is much lower in the model proposed by us and works better than in any other algorithm. In IoT, our approach will deliver stronger API recommendations not only in terms of data efficiency but also in terms of perceiving features related to large data obtained in the IoT system.

VI. Conclusion

The DL-PMF algorithm purposed for mentioned paper enhances the efficiency of the recommendation model by perceiving the implicit nature of the users and API elements. For this purpose, the model of the API treats only relevant information by filtering non-interfering information in the description. Our launch results have demonstrated that obtaining the latent characteristics of users and API elements effectively through our proposed DL-PMF model can significantly raise the recommendation's accuracy. In this instance, the recommendation accuracy will not only be considered as contextual information but in future work, we will consider the collection of features in the description information by RCNN.

References

[1] T. Ramathulasi and M. R. Babu, "Comprehensive Survey of IoT Communication Technologies," In Emerging Research in Data Engineering Systems and Computer Communications, pp. 303-311, 2020.

[2] S. Zhang, L. Yao, A. Sun and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," ACM Computing Surveys (CSUR), vol. 52, no. 1, pp. 1-38, 2018.

[3] Y. Koren, "Factorization meets the neighborhood: a multifaceted collaborative filtering model.," in In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, August.

[4] L. Zheng, V. Noroozi and P. S. Yu, "Joint deep modeling of users and items using reviews for recommendation," in In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017.

[5] H. Wang, N. Wang and D. Y. Yeung, "Collaborative deep learning for recommender systems.," in In Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, 2015.

[6] D. Kim, C. Park, J. Oh, S. Lee and H. Yu, "Convolutional matrix factorization for document context-aware recommendation.," In Proceedings of the 10th ACM conference on recommender systems, pp. 233-240, 2016.

[7] D. Kim, C. Park, J. Oh and H. Yu, "Deep hybrid recommender systems via exploiting document context and statistics of items," Information Sciences, 417,, pp. 72-87, 2017.

[8] X. Shen, B. Yi, Z. Zhang, J. Shu and H. Liu, "Automatic recommendation technology for learning resources with convolutional neural network.," In 2016 International Symposium on Educational Technology (ISET), IEEE., pp. 30-34, 2016.

[9] S. Okura , Y. Tagami , S. Ono and A. Tajima , "Embedding-based news recommendation for millions of users," in the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.

[10] B. H. Devooght R, "Collaborative filtering with recurrent neural networks," arXiv preprint arXiv:1608.07400., 2016.

[11] S. Seo, J. Huang, H. Yang and Y. Liu, "Representation Learning of Users and Items for Review Rating," International Workshop on Machine Learning Methods for Recommender Systems (MLRec)(SDM'17)., 2017.

[12] J. Chen, H. Zhang and X. He, "Attentive collaborative filtering: Multimedia recommendation with," Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval. ACM, pp. 335-344, 2017.

[13] T. Ramathulasi and M. Rajasekhar Babu, "Enhanced PMF Model to Predict User Interest for Web API Recommendation," in In P. Krishna (Ed.), Handbook of Research on Advances in Data Analytics and Complex Communication Networks, 2022.

[14] B. Twardowski, "Modelling contextual information in session-aware recommender systems with neural networks.," in In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, 2016.

[15] H. Soh , S. Sanner , M. White and G. Jamieson, "Deep sequential recommendation for personalized adaptive user interfaces," in Proceedings of the 22nd International Conference on Intelligent User Interfaces, 2017.

[16] L. Yao, X. Wang, Q. Z, W. Shenh, Ruan and W. Zhang, "Service recomrecommendation for mashup composition with implicit correlation regularization," in Int. Conf. Web Serv. (ICWS), 2015.

[17] Z. Zheng, H. Ma, L. Lyu and I. King, "WSRec:Acollaborative filtering based web service recommender system," in IEEE Int. Conf. Web Serv., 2009.

[18] Z. Zheng, H. Ma, M. Lyu and I. King, "Collaborative web service QOS prediction via neighborhood integrated matrix factorization," IEEE Trans. Serv. Comput., vol. 6, p. 289–299, 2013.

[19] B. Cao, J. Liu, M. Tang, Z. Zheng and G. Wang, "Mashup service recommendation based on user interest and social network," in IEEE Int. Conf. Web Serv. (ICWS), 2013.

[20] H. Xue and D. Zhang, "A recommendation model based on content and social network," in In 2019 IEEE 8th Joint International Information Technology and Artificial Intelligence Conference (ITAIC), 2019, May.

[21] X. Wang, H. Wu and C. H. Hsu,, "Mashup-oriented api recommendation via random walk on knowledge graph," IEEE Access, vol. 7, pp. 7651-7662, 2018.

[22] F. Xie, L. Chen, D. Lin, C. Chen, Z. Zheng and X. Lin, "Group preference based API recommendation via heterogeneous information network," in IIn Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings, 2018.

[23] S. P. Ma, H. J. Lin, C. A. Yu and C. Y. Lee, "Web API recommendation based on service cooperative network," in In 2017 International Conference on Applied System Innovation (ICASI), 2017,May.

[24] H. Sun, Z. Zheng, J. Chen, W. Pan, C. Liu and W. Ma, "Personalized open API recommendation in clouds via item-based collaborative filtering," in In 2011 Fourth IEEE International Conference on Utility and Cloud Computing, 2011, December.

[25] K. Fletcher, "Regularizing matrix factorization with implicit user preference embeddings for web API recommendation," in In 2019 IEEE International Conference on Services Computing (SCC), 2019, July.

[26] Y. Chen, M. Zhou, Z. Zheng and D. Chen, "Time-aware smart object recommendation in social internet of things," IEEE Internet of Things Journal, vol. 7, no. 3, pp. 2014-2027, 2019.

[27] S. Duan, D. Zhang, Y. Wang, L. Li and V. Zhang, "JointRec: A deep-learning-based joint cloud video recommendation framework for mobile IoT.," IEEE Internet of Things Journal,, vol. 7, no. 3, pp. 1655-1666, 2019.

[28] Z. Huang, X. Xu, J. Ni, H. Zhu and C. Wang, "Multimodal representation learning for recommendation in Internet of Things," IEEE Internet of Things Journal, vol. 6, no. 6, pp. 10675-10685, 2019.

[29] H. X. Hu, B. Tang, Y. Zhang and W. Wang, "Vehicular ad hoc network representation learning for recommendations in internet of things," IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2583-2591, 2019.

[30] T. Ramathulasi and M. Rajasekharababu, "Augmented latent Dirichlet allocation model via word embedded clusters for mashup service clustering," Concurrency and Computation: Practice and Experience, vol. 34, no. 15, p. e6896, July 2022.

[31] T. Ramathulasi and M. R. Babu, "Enhancing Clustering Performance Using Topic Modeling-Based Dimensionality Reduction," International Journal of Open Source Software and Processes (IJOSSP), vol. 13, no. 1, pp. 1-16, 2022.

[32] Y. Shen, X. He, J. Gao, L. Deng and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," In Proceedings of the 23rd ACM international conference on conference on information and knowledge management, pp. 101-110, 2014.

[33] Y. Kim, "Convolutional neural networks for sentence classification," In Proceedings of the 2014 Empirical Methods in Natural Language Processing (EMNLP), p. 1746–1751, 2014.

[34] R. Collobert, J. Weston, M. Bottou and K. Karlen, "Natural language processing (almost) from scratch," Natural language processing (almost) from scratch. Journal of Machine Learning Research (JMLR), p. 2493–2537, 2011.

[35] A. Van Den Oord, S. Dieleman and B. Schrauwen, "Deep content-based music recommendation".

[36] Y. Hu, Y. Koren and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets".

[37] S. Hochreiter and J. Schmidhuber, "Long short-term memory. Neural computation, 9(8)," pp. 1735-1780., 1997.

[38] R. Jozefowicz, W. Zaremba and I. Sutskever, "An empirical exploration of recurrent network architectures," In International conference on machine learning, pp. 2342-2350, 2015, June.

[39] P. J. Werbos, "Backpropagation through time: what it does and how to do it.," Proceedings of the IEEE, 78(10), pp. 1550-1560, 1990.

[40] A. Mnih and R. R. Salakhutdinov, "Probabilistic matrix factorization," In Advances in neural information processing systems, pp. 1257-1264, 2008.

[41] M. N. Schmid, O. Winther and L. K. Hansen, "Bayesian non-negative matrix factorization," In International Conference on Independent Component Analysis and Signal Separation, pp. 540-547, 2009,March.

[42] J. Liu, Y. Qiu, Z. Ma and Z. Wu, "Autoencoder based API Recommendation System for Android Programming," In 2019 14th International Conference on Computer Science & Education (ICCSE), pp. 273-277, 2019, August.