A Multi-modal Feature Fusion-based Approach for Mobile Application Classification and Recommendation

Buqing Cao*, Weishi Zhong, Xiang Xie, Lulu Zhang, Yueying Qing

School of Computer Science and Engineering& Key Lab. of Service Computing and Software Service New Technology, Hunan University of Science and Technology, China
{buqingcao, luluzhang0727, qyyshhz}@gmail.com, {1151347656, 845455117}@qq.com

Abstract

With the rapid growth of the number and type of mobile applications, it becomes challenging to accurately classify and recommend mobile applications according to users' individual requirements. The existing mobile application classification and recommendation methods, for one thing, do not take into account the correlation between large-scale data and model. For another, they also do not fully exploit the multi-modal, fine-grained interaction features with high-order and low-order in mobile application. To tackle this problem, we propose a mobile application classification and recommendation method based on multi-modal feature fusion. The method firstly extracts the image and description features of the mobile application using an "involution residual network + pre-trained language representation" model (i.e. the TRedBert model). Afterwards, these features are fused by using the attention mechanism in the transformer model. Then, the method classifies the mobile applications based on the fused features through a softmax classifier. Finally, the method extracts the high-order and low-order embedding features of the mobile app with a bi-linear feature interaction model (FiBiNET) based on the classification results of the mobile app, by combining the Hadamard product and inner product to achieve fine-grained high-order and low-order feature interaction, to update the mobile app representation and complete the recommendation task. The multiple sets of comparison experiments are performed on Kaggle's real dataset, i.e., 365K IOS Apps Dataset. And the experimental results demonstrated that the proposed approach outperforms other methods in terms of Macro F1, Accuracy, AUC and Logloss.

Keywords: Mobile applications, Multi-modal feature fusion, Attention mechanism, Bi-linear feature interaction

1 Introduction

As of October 2021, there were nearly 3.99 million mobile applications in China, ranking the first one in the world. Mobile applications, such as e-commerce, online food delivery, games and we-media, are comprehensively affecting the daily life of people and changing their way of life. The number of mobile applications on the Internet has grown exponentially in recent years. In the face of these massive mobile applications, although there is already a large amount of sample data for training, when there is new data to be processed, it still faces some problems, such as cold start and data sparsity. The core problem of how to use existing large-scale classified data samples is to train models for selecting appropriate models. When new mobile applications appear, most of the information they contain is just pictures, descriptions, and publisher information. On the one hand, it is difficult for practitioners to conduct overall benchmark and relevant analysis of the mobile application market, so they need to accurately classify mobile apps to complete subsequent tasks such as risk control and data analysis. In addition, it is challenging for users to choose mobile applications suitable for their personalized preferences and needs. Therefore, it is necessary to provide high-quality mobile application recommendation mechanisms to improve user experience.

In traditional mobile application classification methods, such as multi-layer perceptron [1] and support vector machine [2], the performance of most classification models depends on the quality of annotation data set. As we know, obtaining high-quality annotated data requires a lot of labor costs. However, this method relies on manual design and is affected by human factors, so its improvement is poor. Features that perform well in one field may not perform well in other fields. Traditional mobile application recommendation methods, such as collaborative filtering [3] and matrix factorization [4], usually transform mobile application recommendation problems into supervised learning problems. In essence, such models first embed the user and the application separately, and then leverage the interaction information between them to optimize the model and perform recommendations. These methods work well in many recommendation and ranking tasks. However, they also have some shortcomings. For example, they are sensitive to sparse data, have limited predictive capability for new users, and only learn linear interactions between users and services.

With the growth of multi-modal data on the Web, content information from different modes (visual, auditory, etc.) has recently been used to provide complementary feature signals for traditional text features. Most existing research in this area has focused on categorizing emotions in conversation. Specifically, Poria et al. [5] proposed a multi-core learning method and an LSTM-based sequential architecture [6] in 2015 and 2017, respectively, to integrate text, visual and audio features. Based on this work, Zadeh et al. [7] further designed tensor aggregation networks and memory aggregation

*Corresponding Author: Buqing Cao; E-mail: buqingcao@gmail.com
DOI: 10.53106/160792642022112306023

networks [8] to better capture the interactions between different modes. However, these methods are designed for coarse-grained classification, which may not be very effective for our fine-grained, target-oriented mobile application classification. Moreover, some deep learning model are applied for recommendation, few-shot Segmentation [9], data communication [10], anomaly detection [11].

To solve the problems of too short-text description for mobile applications and insufficient consideration of multi-modal feature and its fine-grained high-low order feature interaction, a mobile application classification and recommendation method based on multi-modal feature interaction is proposed. The method can enhance the performance and interpretability of classification and recommendation. In the classification task, the method uses deep large-scale network to improve the learning ability of feature parameters, and uses attention mechanism to enhance the aggregation of mobile application description features and image features. In the recommendation task, SENET mechanism is used to dynamically learn the importance of mobile application features, and bi-linear feature interaction is used to better extract the semantics of dense features and sparse features during interaction. The main contributions of this paper are summarized as:

(1) To the best of our knowledge, this is the first time that the residual network supported by involution module [31] is exploited in image feature extraction for mobile applications. This helps to focus on local features in Logo images of mobile applications and improve image feature extraction performance.

(2) We propose a new mobile application classification and recommendation method based on multi-modal feature fusion. The proposed method uses attention mechanism to learn the importance of different modal feature dynamically, and learns feature interactions in a fine-grained way to improve the accuracy of service classification and recommendation.

(3) Based on Kaggle's real data set, we verify the validity and accuracy of multi-modal feature aggregation and bi-linear feature interaction in mobile application classification and recommendation tasks. Experimental results show that in most cases, the performance of the proposed method is better than that of Macro F1, Accuracy, AUC and Logloss.

The rest of this paper is organized as follows. In Section 2 we present work related to mobile app classification and recommendation. In Section 3, we describe specific methods. We report experimental result and analysis in Section 4, and we conclude the work of this paper and subsequent research in Section 5.

2 Related Work

With the development of mobile network terminals, it is difficult for users to quickly find their interested mobile applications from a large number of rich mobile applications. Therefore, relevant scholars have studied mobile application recommendation to improve the efficiency of mobile application search and the accuracy of recommendation results. Mobile application recommendation mainly includes content-based, user-item interaction diagram and hybrid mobile application recommendation. Existing studies have shown that mobile applications with similar functions are first divided into correct clusters, and the efficiency of mobile application discovery can be improved by reducing the search space of mobile applications, which can greatly improve the effect of mobile application recommendation.

2.1 Mobile Application Classification

Mobile application classification is mainly based on feature vectors extracted from mobile application documents, and then similarity measurement methods such as Euclidean distance and cosine similarity are used to calculate the similarity between services. Finally, service classification or clustering is carried out to improve the service recommendation effect. Woerndl et al. [12] introduced both social network information and context information into mobile application classification tasks, and proposed a hybrid mobile service classification approach. Cao et al. [13] integrated service content and service network to perform service clustering. Liu et al. [14] proposed a new structure selection model, which uses the hierarchical structure of application tree to capture the competition among mobile applications and learns fine-grained user preferences to achieve classification. More and more machine learning and deep learning have been applied into the field of mobile application classification to further exploit the hidden relationships between mobile applications with limited feature information. Among this, Cao et al. exploited LDA topic model based on multiple data sources [15], relational topic model [16], and graph attention network [17] to generate service representation for service classification and clustering.

In the above methods, the topic model or deep neural network is used to extract mobile application description features, and perform similarity calculation or fusion with the initial feature vector and complete mobile application classification. They take into account the problems of short description documents in mobile applications and limited corpus, and propose a method of mining the word order and context information of words in description documents or auxiliary information such as fusion tags, so as to better realize short text modeling. However, the discrete features used by these methods are generally associated, and the problem of sparse document semantics is not solved well. Therefore, the mobile application classification method which only considers a single modal feature is slightly lacking in accuracy and interpretability. To achieve fine-grained classification for mobile applications, different weights should be given to the features of different modes when they interact.

2.2 Mobile Application Recommendation

Mobile application recommendation is mainly to recommend mobile applications to users by mining and using the text description, label and other information of mobile applications. For example, based on the characteristics of Twitter followers of various mobile applications, Lin et al. [18] used LDA topic model to generate potential groups and predict the possibility of target users' preferences for the application. Chen et al. [19] analyzed content similarity of mobile applications according to the meaning relation and category among mobile applications, and then performed content-oriented mobile application recommendation. Cao et al. [20] proposed a mobile application recommendation framework focusing on application version information based on dual heterogeneous data and user application version update

information. Peng et al. [21] proposed a mobile application recommendation method based on license and function. Xu et al. [22] presented a mobile application recommendation method based on neural network, which makes use of semantic methods used by mobile applications to carry out effective mobile application recommendation. In recent years, recommendation system based on deep learning has become a hot research topic. MF (Matrix factorization) [23] is an enhanced collaborative filtering method that mines implicit semantics. It uses inner product to model the interaction between users and mobile applications, maps high-dimensional user mobile application scoring matrices into two low-dimensional user mobile application scoring matrices, and solves the problem of data sparsity. In addition, some extended models based on matrix factorization are proposed. For instance, NCF (Neural Collaborative Filtering) uses nonlinear neural network to replace the interaction function of inner product in matrix decomposition [24]. CDL (Collaborative Deep Learning) expands the embedding function of matrix decomposition by combining in-depth representation of rich feature information in mobile applications [25]. Recently, some researchers have proposed a fine-grained model of mobile application recommendation based on matrix decomposition to achieve accurate recommendation goals by distinguishing low-order features, high-order features and their importance. For example, NFM (Neural Factorization Machines) [26] and DeepFM [27] consider both low-order and high-order feature interactions. MLR (Multiple linear regression) [28] only uses high-order feature interaction. AFM (Attention Factorization Machines) [29] emphasizes the importance (weight) of feature interaction between users and mobile applications. Moreover, features learning [30] and cognitive knowledge [31] are exploited for recommendation.

These methods mainly focus on how to better mine and utilize the interaction between feature information, while ignoring the importance or weight of mobile application function attributes themselves. In fact, users have different preferences for different features in mobile applications. For example, most users care more about mobile application ratings than price and size. In other words, different features have different importance for mobile application recommendation tasks.

3 Proposed Method

The framework of the multi-modal feature fusion-based mobile application classification and recommendation method is shown in Figure 1. It consists of three key components: 1) The mobile application feature extraction, which performs the feature extraction of the images and description information of the mobile application nodes. 2) The mobile application classification, which uses the self-attention and multi-head attention mechanism in the Transformer to distinguish the feature importance of different modalities and fuse them, and uses the softmax classifier to classify the mobile application according to the fused feature information. 3) The mobile application recommendation, which inputs the classified data into the FiBiNet model according to its category, and dynamically learns the importance of features by fitting the relationship between features and samples through weights. For the more critical features, greater weight will be assigned, and the weight of non-critical features will be weakened. The importance of each dimension is considered simultaneously using bi-linear operations to complete the mobile app recommendation.

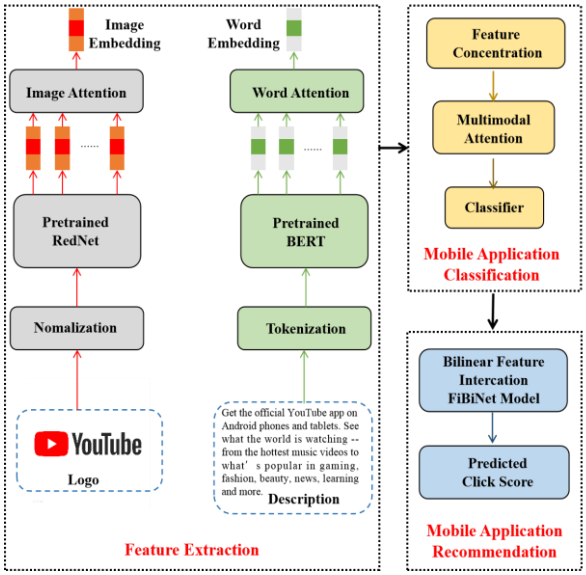


Figure 1. The framework of the proposed method

3.1 Mobile Application Feature Extraction

A set of multi-modal samples D is extracted from the mobile app datasets, for each sample $c \in D$, which contains a sentence S consisting of n words of mobile app description information (e.g., $w_1,...,w_n$) and an associated mobile app image I. Then, we use D as a training corpus to train and learn in a mobile app classifier, which is then able to predict the mobile app category labels among the unlearned samples to correctly predict the category labels of mobile apps. After the initial normalization and self-coding wording pre-processing, the mobile application description features are extracted in the feature extraction layer using the BERT model and the image features are extracted using the residual network (RedNet) of the involution module.

3.1.1 Descriptive Feature Extraction

The BERT model is investigated to be able to derive contextualized word representations from a large corpus, and has the ability to learn alignment between two random inputs. So, we apply BERT as a basic model to extract describing feature for mobile applications. In the experiment, the pre-trained dual prediction corpus is selected as the initial model, and its parameters are adjusted and learned by Fine-Tune operation. The BERT model is essentially a multi-layer bidirectional transformer encoder. A multi-head and self-attention layer will be used to convert each position in the input sequence into a weighted sum of the input layer to capture global information. Specifically, for the i-th head attention, the input layer $X \in R_{d \times N}$ is transformed based on the dot product attention mechanism:

$$ATT_i(X) = softmax\left(\frac{[W_{Q_i}X]\cdot[W_{K_i}X]}{\sqrt{d/m}}\right)[W_{V_i}X]. \quad \textbf{(1)}$$

Where $\{W_{Q_i}, W_{K_i}, W_{V_i}\} \in R_{(d/m)*d}$ are learnable parameters corresponding to query, key, and value

respectively. After that, the outputs of attention mechanisms are concatenated together for linear transformation.

We characterize each mobile application by self-coding its descriptive information and input it into the pre-trained BERT. To complete the specific classification task, in addition to the token of the word, the model also inserts a specific classification token (*[CLS]*) at the beginning of each input sequence, and the last transformer layer output corresponding to the classification token is used to play the role of aggregating the entire sequence characterization information. This paper retains the *[CLS]* vector and the extracted semantic vector as the output O to improve the model accuracy:

$$O = [H^0, H^{[CLS]}]. \quad \textbf{(2)}$$

After that, the output O is linearly varied by the softmax function and the final representation vector H_S of the *D*N*-dimensional text information of the mobile application I is obtained.

3.1.2 Image Feature Extraction

The pre-trained RESNET model learns the parameters well deep into the network. The main limitation of it is its visual performance is not sensitive to different targets, as the visual features of the same input sensor are always the same, regardless of the target it is considering. Intuitively, with the mobile application image as the input, only some areas of the relevant image are closely related to it, while other areas should be ignored to eliminate noise. To better capture the strong local features in the image information of mobile applications, which can obtain more accurate characterization of App's Logo, this paper chooses to use the involution residual network model in which the convolution kernel is replaced by the involution kernel and introduces the attention mechanism into the residual network to generate channel level attention factors to distinguish the weight differences between different local features, which is more conducive to multi-modal feature fusion.

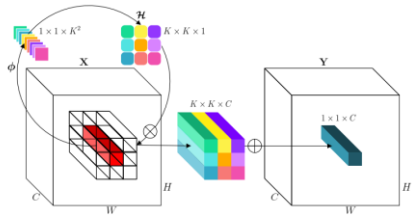


Figure 2. Involution module

As shown in Figure 2, the involution kernel $H_{i,j} \in R_v$ is produced by a function φ conditioned on a single-pixel at *(i, j)* followed by a channel-to-space rearrangement. The multiplication and addition operation of the involution is decomposed into two steps. The product operation is to multiply the tensors of the *C* channels by the involution kernel *H*, respectively, and the addition operation is to add the elements within the involution kernel to the involution kernel. In the design of the involution, the involution kernel is specially customized at the pixel $X_{i,j}$ corresponding to the coordinates *(i, j)*, but shared across channels, and *G* counts the number of groups each group shares the same involution kernel. The input is multiplied and added by the inner volume check, and the representation output of the inner volume module is obtained as follows:

$$Y_{i,j,k} = \sum_{(u,v)\in\Delta_K} H_{i,j,u+\lfloor\frac{K}{2}\rfloor,v+\lfloor\frac{K}{2}\rfloor,kG/C} X_{i+u,j+v,k}. \quad \textbf{(3)}$$

Unlike the convolution kernel, the shape of the Involution kernel *H* depends on the shape of the input feature map *X*. The design idea of it is to generate an Involution kernel conditioned on the original input tensor so that the output kernel is aligned with the input kernel. Here we denote the kernel generation function as $\emptyset$ and extract the function map for each position *(i, j)* as:

$$H_{i,j} = \emptyset(X_{\Psi_{i,j}}). \quad \textbf{(4)}$$

The ResNet model can solve the phenomenon that the training difficulty increases after the network is deepened. Its residual module contains two *3*3* convolution blocks and a short-cut connection. The short-cut connection can effectively alleviate the gradient disappearance caused by too deep depth during back-propagation, which makes the performance not worse after the network is deepened. Short-cut connection is another important idea of deep learning. In addition to computer vision, short-cut connection has also been used in the fields of machine translation and speech recognition. In addition, ResNet with short-cut connections can be seen as an ensemble of many networks of different depths sharing parameters, and the number of networks increases exponentially with the number of layers.

To sum up, we input the mobile application image I in the dataset into the visual model *RedNET-152* to obtain the output of the last layer of involution layer:

$$ResNet(I) = \{r_j | r_j \in R^{2048}, j = 1,2,\ldots,49\}. \quad \textbf{(5)}$$

The original mobile application image can be divided into *7×7=49* regions, *2048*-dimensional vector r_j represents each region. Next, we use the linear transformation function to project the visual features of mobile applications into the same space of text features $G=W_v RedNet(I)$, where $W_v \in R_d \times 2048$ is a learnable parameter. After that, we will output *RedNet(I)* and change it linearly through the softmax function to obtain the final characterization vector $G=W^T ResNet(I)$ of *D*2048*-dimensional mobile application image information.

3.2 Mobile Application Classification

After the feature extraction, the final representation vector H_S of *D*N* dimension of mobile application text information and the final representation vector $G=W^T ResNet(I)$ of *D*2048* dimension of image information can be obtained. Then the two representation vectors are fully connected to obtain the long vector $[G, H_S]$ and input it into the multi-modal encoder, i.e., the Bert layer composed of multiple transformers. The attention mechanism is used to self-encode and model the feature interaction between text and image modes:

$$ME(G, H_S) = BT_{L_m}([G, H_S]). \quad \textbf{(6)}$$

Where L_m is the number of layers of the multi-modal encoder, then the final hidden state marked by "*[CLS]*" can be used for mobile application classification tasks to effectively

capture the dynamic attention within and between modes of mobile applications.

The multi-modal fusion representation of mobile applications obtained from the model output is input to an FC layer, and the probability distribution of all candidate mobile application categories is output by using the softmax activation function. Softmax converts the output value of multiple categories into relative probability, representing the probability that the node belongs to a specific category. Its calculation is shown in formula (7), where N is the number of candidate categories.

$$softmax(\overrightarrow{f_t'}) = \frac{exp(\overrightarrow{f_t'})}{\sum_{i=1}^{N} exp(\overrightarrow{f_t'})}. \quad (7)$$

In the training process, the cross-entropy loss function is used to learn the parameters in the model, which is usually selected as the objective function of multi-classification problems. Cross-entropy describes the distance between the actual and expected output probability. The smaller its value is, the closer the two probability distributions are. The specific function calculation method is shown in formula (8):

$$Loss = -\sum_{i=1}^{K} y_i \, log \frac{1}{p_i}. \quad (8)$$

Where, K is the number of mobile application categories, y_i is the variable of the mobile application node, which is obtained by one hot coding. If the service node is consistent with the corresponding category i, it is 1, otherwise it is 0, p_i is the prediction probability that the mobile application belongs to the category.

3.3 Mobile Application Recommendation

After obtaining the classification results of multi-modal feature interactive mobile applications, the mobile application content information is embedded into the bi-linear feature interaction model according to its categories. The representation of mobile applications, the importance of different features and the fine-grained, high-order and low-order feature interactions are dynamically learned to improve the accuracy and diversity of recommendation.

As shown in Figure 3, the upper part of the bi-linear feature interaction model (i.e., FiBiNet) is the deep part, which is mainly the MLP network. It integrates the output connection of the bi-linear interaction layer into the dense vector through the connection layer, and then inputs the cross-combination feature into the neural network to obtain the prediction score in the prediction layer. The lower shallow part is the core of FiBiNet, which mainly processes the input features. Firstly, in the lower left part of the Figure 3, the high-dimensional sparse features of the APP are mapped into low-dimensional dense vector representation after passing through the initial embedding layer. The vector represents the importance of dynamic learning features embedded through the SENET layer to obtain SENET-Like embedding. Then, the initial characterization embedding and SENET-Like embedding are input into the bi-linear interaction layer, and the features are crossed. After the output cross features are concatenated, they are finally input into the MLP to complete the mobile application recommendation.

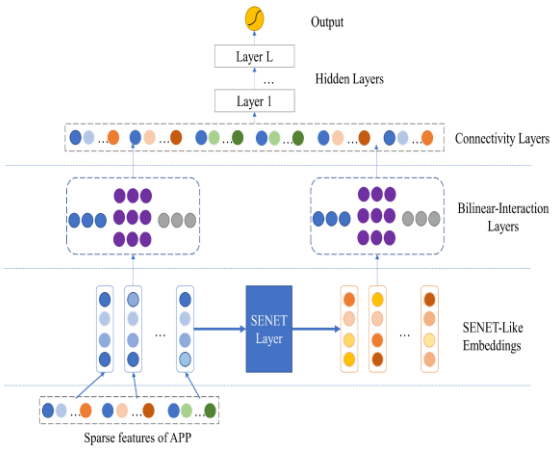


Figure 3. FiBiNet model

3.3.1 Shallow Part

Firstly, we input the classified mobile application into the initial embedding layer in FiBiNet according to the category, which can embed the sparse feature into the low dimensional, continuous, real value vector, convert the sparse matrix into the dense matrix through linear transformation, extract the hidden features of the matrix, and improve the generalization ability of the model. The output of the embedded layer is represented as follow:

$$E = [e_1, e_2, \dots, e_i, \dots, e_f]. \quad (9)$$

Then, for various downstream target tasks, different characteristics have different importance. For example, when the mobile application recommendation task needs to be completed, the user score of the mobile application is different from the applicable system version of the mobile application. This paper exploits SENET network for training and learning, obtains the embedding weight, and outputs the final embedding result. It includes: (1) dimension reduction, (2) excitation, (3) reweight. Firstly, the dimension reduction is performed on the embedded features to obtain the global features. Then, the sigmoid activation is carried out to learn the relationship between each embedding, and the embedding weights of different domains are obtained. Finally, the final embedding result is obtained by multiplying the original embedding result. After obtaining the mobile application representation of the initial embedding layer and SENET layer, it is necessary to carry out second-order and high-order feature interaction for sparse features and dense features. Classical feature interaction methods include inner product and Hadamard product. The inner product is widely used in shallow models such as FM, FFM, while the Hadamard product is usually used in deep model such as NFM. For example, the inner product is $[a_1, a_2, \dots, a_n] \cdot [b_1, b_2, \dots, b_n] = \sum N$, and the Hadamard product is $[a_1, a_2, \dots, a_n] \odot [b_1, b_2, \dots, b_n] = [a_1 b_1, a_2 b_2, \dots, a_n b_n]$. As the inner product and Hadamard product of the interaction layer are too simple, it is impossible to model the feature interaction of sparse data sets effectively. Therefore, this paper adopts a more fine-grained method, which combines inner product and Hadamard product to learn feature interaction with additional parameters. FiBiNet can integrate the advantages of two interactive products or use one alone. The interactive vectors p and q of output E in formula (9) of embedded layer and

output *V* of SENET layer can be calculated in the below three ways:

$$p_{ij} = v_i \cdot W_{ij} \odot v_j. \quad \textbf{(10)}$$

$$p = [p_1, \dots, p_i, \dots, p_n]. \quad \textbf{(11)}$$

$$q = [q_1, \dots, q_i, \dots, q_n]. \quad \textbf{(12)}$$

The above two interaction vectors are connected and inputted into the deep part.

3.3.2 Deep Part

To improve the accuracy of mobile application recommendation, after the shallow feature extraction and interaction are completed, a MLP composed of multi-layer DNN is added to capture high-order features. Combining shallow features and high-order features, the final recommendation result $\hat{y}$ can be obtained:

$$\hat{y} = \sigma(w_0 + \sum_{i=0}^{m} w_i x_i + y_d). \quad \textbf{(13)}$$

Where $\hat{y}$ is the prediction value for mobile application and $\hat{y} \in (0,1)$, σ is the sigmoid function, *m* is the characteristic size, and the rest is the linear regression.

In the overall training process of the model, *Logloss* is used as the optimization objective function:

$$Logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i \log(\hat{y}_i)) + (1 - y_i) * \log(1 - \hat{y}_i)). \quad \textbf{(14)}$$

Where y_i is the real label of the *i*-th mobile application, $\hat{y}_i$ is the prediction label of the *i*-th mobile application, *N* is the total number of mobile applications.

4 Experiment and Analysis

4.1 Dataset Information

Table 1. Kaggle dataset information

Name	Number	Name	Number
Business	95701	Education	94536
Games	58489	Entertainment	28554
Finance	9445	Productivity	9280
Music	8543	Utilities	7785
Book	7632	Lifestyle	7610
Reference	5927	Travel	5240
Photo & Video	4227	Medical	3872
Sports	2936	News	2797
Social Networking	2774	Stickers	2682
Food & Drink	2583	Shopping	2375

The experimental data is selected from Kaggle "365k-ios-apps-dataset", and its URL is *https://www.kaggle.com/fentyforte/365k-ios-apps-dataset*. This dataset includes information on 365K mobile apps and 21 different categories of mobile apps. To facilitate the validation test set, mobile applications lacking logo or descriptive information were eliminated during the experiment. Because the scale of the experimental dataset is too large, the top 5, 10, 15, and 20 categories with the largest number of mobile applications are selected as the experimental data, and the distribution of the top 20 categories mobile applications with the largest number is shown in Table 1. In addition, during the experiment, 60% of the experimental data is selected as the training set, 20% as the validation set, and 20% as the test set.

4.2 Mobile Application Classification Experiment and Analysis

4.2.1 Evaluation Metrics

To evaluate the effectiveness of mobile application classification, two commonly used evaluation metrics, Macro F1 and Accuracy, are used in the experiment. Among this, accuracy is the ratio of correct judgments to all judgments. The number of correct judgments is the sum of the true-positive case *TP* and the true-negative case *TN*, and the number of all judgments is the sum of the four judgment possibilities (false-positive case *FP*, false- negative case *FN*, true-positive case *TP*, true-negative case *TN*). Its calculation formula is as follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}. \quad \textbf{(15)}$$

By calculating the recall Rec_i and precision Pre_i of each category, the average recall Rec_{ma} and average precision Pre_{ma} of all *N* categories are obtained, and Macro F1 is finally obtained. Among them, the recall Rec_i describes the proportion of correctly classified mobile applications in all mobile applications of this category. The precision Pre_i describes the proportion of mobile applications that belong to this category in the final classification of the model. Macro F1 is the harmonic mean of recall and precision, which calculated as follows:

$$Macro - F1_{ma} = \frac{2 * Rec_{ma} * Pre_{ma}}{Rec_{ma} + Pre_{ma}}. \quad \textbf{(16)}$$

4.2.2 Baselines

- TResBert [32]: The characterization of the text part is the text features and position encoding extracted by BERT, and the input of the picture part is the image region features extracted by the original ResNet plus the corresponding position encoding. Two characterization vectors above are inputted into the encoder layer in transformer after vector splicing operation, and attention mechanism is used to dynamically assign the weights between multi-modalities and their respective modalities. The classification of mobile applications based on the final representation is performed through the softmax classifier.
- Res-bert [32]: The characterization of the text part and the input of the picture part is same to that of TResBert. Only subjected vector splicing operation is performed for two representation vectors above and softmax classifier is used to classify mobile applications.
- Red-bert [32]: The characterization of the text part is the text features and position encoding extracted by BERT, the input of the picture part uses the picture region features extracted by the in-convolution residual network RedNet and the corresponding position encoding. Only subjected vector splicing operation is performed for two representation vectors

above. Softmax classifier is used to classify mobile applications.

- Bert [33]: Only the description feature of mobile applications extracted by Bert is exploited to classify the mobile application.

4.2.3 Parameter Settings

In this part, the parameter settings mainly include the size of embedding, the number of layers of neural network and learning rate. In the experiment of mobile application classification, because the model is too large, we choose to use Bert with medium bilingual material library for pre- training and Rednet model with 50 layers. The model training uses the batch training mode, selects the BPR loss function, which is widely used in mobile application classification. The Adam is exploited as the mini batch optimizer. To ensure the authenticity and validity of the experimental results, we set the parameters of all models in the comparative experiment as shown in Table 2 and Table 3. Among this, in the RedNET, the parameters include *batch_size, pretrain, epoch, layers, Drop_out, Regularizations, Learning rate, and Warmup*. In the Bert, the parameters include *batch_size, pretrain, epoch, Embed_size, Drop_out, Regularizations, Learning rate, and WarmUp.*

Table 2. Parameter settings of RedNET

Parameter	Setting
batch_size	32
pretrain	1
epoch	1000
layers	50
Drop_out	0.1
Regularizations	1e-3
Learning rate	5e-5
WarmUp	0.1

Table 3. Parameter settings of BERT

Parameter	Setting
batch_size	32
pretrain	1
epoch	1000
Embed_size	32
Drop_out	0.1
Regularizations	1e-3
Learning rate	5e-5
WarmUp	0.1

4.2.4 Experimental Results and Analysis

The experimental results of all methods are shown in Figure 4 and Figure 5. It can be found that:

- During data pre-processing, no desensitization processing is performed on the mobile application text, and only the mobile application documents is subjected to tokenization and self-encoding processing, so the overall accuracy of the experiment is not high.
- Of all the compared methods, the performance of only Bert is the worst. That is to say, the accuracy of classifying mobile applications using text information alone is the worst. Thus, it is necessary that more detailed mining of relevant information between different modal data, such as image-text feature interaction.
- In most cases, the accuracy of using involution instead of CNN for image feature extraction of mobile applications is higher, which shows that the use of attention mechanisms in multi-modal feature fusion can better distinguish the importance of different features.
- Overall, TRedBert maintains good performance. In particular, when the number of categories is 20, TRedBert has 10.69%, 19.63%, 41.61%, and 48.58% improvement in Accuracy. It has 5.598%, 10.80%, 30.25%, and 37.42% improvement in Macro-F1, compared to TResbert, Redbert, Resbert, and Bert, respectively. The using of transformer for feature fusion is more accurate than the model utilizing only vector stitching. It can be seen that the attention mechanism in multi-modal feature fusion can better distinguish the importance of different features, and make the representation more suitable to downstream tasks.

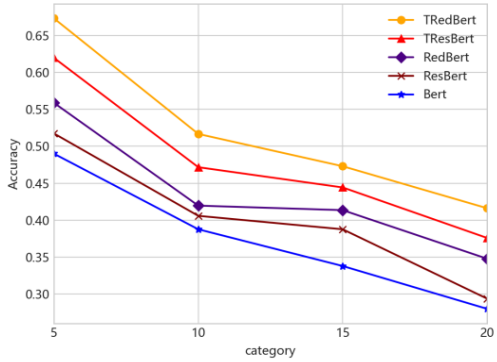


Figure 4. Accuracy of mobile application classification

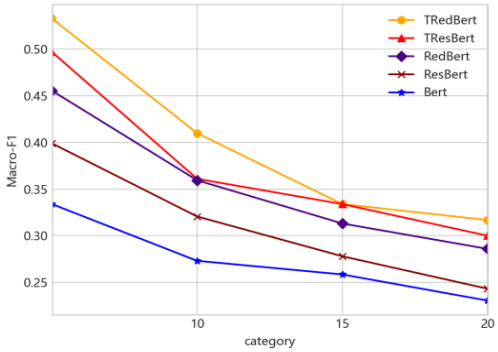


Figure 5. Macro-F1 of mobile application classification

4.3 Mobile Application Recommendation Experiment and Analysis

4.3.1 Evaluation Metrics

To evaluate the effectiveness of mobile app recommendations, two commonly used evaluation metrics, i.e., AUC and Logloss, are adopted. Usually, for binary classification problems, we can set a threshold to classify samples into positive and negative samples. According to

different thresholds, the corresponding coordinate points in the ROC are calculated to form the ROC curve. AUC is the area under the ROC curve. When *0.5<AUC<1*, the model outperforms a random classifier. In particular, the closer the AUC is to 1.0, the higher the authenticity, when it is equal to 0.5, the authenticity is the lowest, and its calculation formula is as follows:

$$AUC_i = \int_0^1 ROC_i(fpr)d(fpr). \quad \textbf{(17)}$$

Where *fpr* stands for false-positive rate and *tpr* stands for true-positive rate. In ROC space, coordinate points describe the trade-off between FP (false-positive cases) and TP (true-positive cases).

Logloss measures the accuracy of the classifier by penalizing wrong classifications. Minimizing the logloss is equivalent to maximizing the accuracy of the classifier. Logloss reflects the average deviation of the sample and is often used as the loss function of the model for optimization. Its calculation formula is as follows:

$$Logloss = -\frac{1}{N}\sum_{i=1}^{N}(y_i log(\hat{y_i})) + (1 - y_i) * log(1 - y_i). \quad \textbf{(18)}$$

4.3.2 Baselines

- MLR [28]: LR is a regression analysis that models the relationship between one or more independent and dependent variables using a least-squares function called a linear regression equation. LR cannot fit nonlinear data, MLR can fit nonlinear data by multivariate fitting.
- FNN [34]: The FNN model only includes the deep part for high-level feature extraction of mobile applications. Through the interaction between the splicing features, the low-level features cannot be fitted due to the lack of the shallow part (machine learning model), and a pre-trained model is required.
- AFM [29]: AFM introduces the attention mechanism into the factorization machine model, which can assign weights to different feature combinations. The general idea is to give different degrees of attention to different mobile application feature combinations and process cross-features more finely.
- NFM [26]: The neural factorization machine is an attempt to achieve the neural network of the FM model, which enhances the representation ability of the model by taking the second-order cross term of the FM as the input of the deep model.
- DeepFM [27]: DeepFM is divided into wide and deep parts. The wide part extracts low-order features by FM, and the deep part extracts high-order features by DNN. Both low-order or high-level features may have an impact on final recommendation. Therefore, the most important thing is to learn the implicit combination of features behind user's click behavior.

4.3.3 Parameter Settings

In the training process of mobile application recommendation, parameter settings mainly include the size of embedding, the number of neural network layers, learning rate and super parameters. The BPR loss function, which is widely used in mobile application recommendation, is selected, and Adam is utilized as the mini batch optimizer. In the feature modeling, there are more continuity features and less sparsity features. To ensure the authenticity and validity of the experimental results, we set the parameters of all models in comparative experiment as shown in the following Table 4, including *batch_size, pretrain, epoch, Mess_drop, Node_drop, Regularizations, and Learning rate*.

Table 4. Parameter settings in recommendation experiments

Parameter	Setting
Batch_size	32
Pretrain	0
Epoch	600
Mess_drop	0.1
Node_drop	0.1
Regularizations	1e-3
Learning rate	1e-5

4.3.4 Recommendation Experiment and Analysis

The experimental results of all methods are shown in Figure 6 and Figure 7. It can be found that:

- When the number of categories in the dataset increases and other experimental settings remain unchanged, the overall recommendation performance decreases with the increase of categories. Especially for the models like FM, the feature interaction performance of the factorization machine also decreases due to the increase in the sparsity of the feature matrix. However, due to the large dataset, the performance difference is not obvious when the number of categories reaches more than 15.
- Among all the compared methods, MLR and AFM performed poorly. This is because they cannot learn high-order interaction features, which leads to the performance of mobile app recommendation suffering. The overall performance of NFM and DeepFM is better, indicating that the low-order and high-order feature interactions they learned help to improve the recommendation quality.
- Deep models such as NFM and DeepFM outperform MLR. When using 20 categories as input, the performances of FNN and DeepFM are improved by 15.88% and 13.72%, respectively. The experimental results show that when the features are sparse, the deep model can better model and mine effective information.
- On the whole, TRedBert+FiBiNET maintains good performance. In particular, when the number of categories is 20, TRedBert+FiBiNET has 15.66%, 1.764%, 4.163%, 13.53%, and 30.43% improvement in AUC, and has 50.60%, 7.5%, 17.88%, 25.17%, and 52.42% improvement in Logloss compared to AFM, DeepFM, NFM, FNN and MLR, respectively. It can be seen that through the attention mechanism to distinguish the importance of multi-dimensional mobile application features and the use of fine-grained high- and low-order feature interaction learning, which enable the model to achieve better recommendation performance under the same experimental setting.

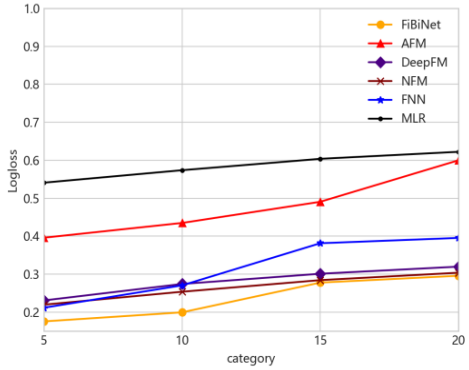


Figure 6. Logloss of mobile application recommendation

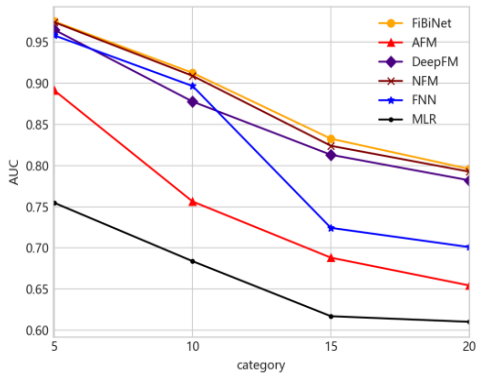


Figure 7. AUC of mobile application recommendation

5 Conclusion and Future Work

In this paper, we propose a multi-modal feature fusion-based approach for mobile application classification and recommendation. It first extracts textual and image features of mobile applications using two leading large-scale pre-training models. Then, it performs fine-grained fusion of the stitched multi-modal feature vectors and classifies the mobile applications by using an attention mechanism. Finally, the FiBiNET model is used to interact with the classified mobile applications in high and low order bi-linear features to complete the recommendation task. The multi-set comparison experiments on real Kaggle datasets demonstrate the effectiveness of the proposed method. In the future, we will consider using a single model to extract features from multi-modal data, and achieve easier training and more actionable multi-modal mobile application classification and recommendation.

Acknowledgements

The authors thank the anonymous reviewers for their valuable feedback and comments. The work is supported by the National Natural Science Foundation of China (No. 61873316, 61872139, 61832014, and 61702181), Hunan Provincial Natural Science Foundation of China under grant No. 2021JJ30274, and the Educational Commission of Hunan Province of China (No. 20B244). Buqing Cao is the corresponding author of this paper.

References

[1] Z. Zhang, M. Lyons, M. Schuster, S. Akamatsu, Comparison between Geometry-based and Gabor-wavelets-based Facial Expression Recognition Using Multi-Layer Perceptron, *Proceedings of Third IEEE International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, 1998, pp. 454-459.

[2] C. Saunders, M. Stitson, J. Weston, R. Holloway, L. Bottou, B. Scholkopf, A. Smola, Support Vector Machine, *Computer Science*, Vol. 4, No. 1, pp. 1-28, September, 2002.

[3] J. Breese, D. Heckerman, C. Kadie, Empirical Analysis of Predictive Algorithm for Collaborative Filtering, *In Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, Madison, Wisconsin, USA, 1998, pp. 43-52.

[4] Y. Koren, R. Bell, C. Volinsky, Matrix Factorization Techniques for Recommender Systems, *IEEE Computer Journal*, Vol. 42, No. 8, pp. 30-37, August, 2009.

[5] S. Poria, E. Cambria, D. Hazarika, N. Mazumder, A. Zadeh, L. Morency, Multi-level Multiple Attentions for Contextual Multimodal Sentiment Analysis, *2017 IEEE International Conference on Data Mining (ICDM)*, New Orleans, LA, USA, 2017, pp. 1033-1038.

[6] S. Poria, E. Ca, A. Gelbukh, Deep Convolutional Neural Network Textual Features and Multiple Kernel Learning for Utterance-level Multimodal Sentiment Analysis, *Proceeding of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, 2015, pp. 2539-2544.

[7] A. Zadeh, M. Chen, S. Poria, E. Cambria, L. Morency, Tensor Fusion Network for Multimodal Sentiment Analysis, *Proceedings of the 2017 Conference on Empirical Methods in Natural Language*, Copenhagen, Denmark, 2017, pp. 1103-1114.

[8] A. Zadeh, P. Liang, N. Mazumder, S. Poria, E. Cambria, L. Morency, Memory Fusion Network for Multi-view Sequential Learning, *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, New Orleans, Louisiana, USA, 2018, pp. 5634-5641.

[9] H. Gao, J. Xiao, Y. Yin, T. Liu, J. Shi, A Mutually Supervised Graph Attention Network for Few-shot Segmentation: The Perspective of Fully Utilizing Limited Samples, *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)*, pp. 1-13, March, 2022. DOI:10.1109/TNNLS.2022.3155486.

[10] H. Gao, C. Liu, Y. Yin, Y. Xu, Y. Li, A Hybrid Approach to Trust Node Assessment and Management for VANETs Cooperative Data Communication: Historical Interaction Perspective, *IEEE Transactions on Intelligent Transportation Systems*, Vol. 23, No. 9, pp. 16504-16513, September, 2022.

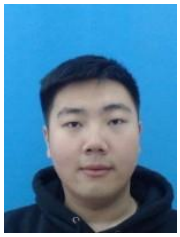
[11] H. Gao, B. Qiu, R. Barroso, W. Hussain, Y. Xu, X. Wang, TSMAE: A Novel Anomaly Detection Approach for Internet of Things Time Series Data Using Memory-Augmented Autoencoder, *IEEE Transactions on Network Science and Engineering (TNSE)*, pp. 1-14, March, 2022. DOI:10.1109/TNSE.2022.3163144.

[12] W. Woerndl, C. Schueller, R. Wojtech, A Hybrid Recommender System for Context-aware

Recommendations of Mobile Applications, *IEEE 23rd International Conference on Data Engineering Workshop*, Istanbul, Turkey, 2017, pp. 871-878.
[13] B. Cao, X. Liu, M. Rahman, B. Li, J. Liu, M. Tang, Integrated Content and Network-Based Service Clustering and Web APIs Recommendation for Mashup Development, *IEEE Transactions on Services Computing*, Vol. 13, No. 1, pp. 99-113, January-February, 2020.
[14] B. Liu, Y. Wu, N. Gong, J. Wu, H. Xiong, M. Ester, Structural Analysis of User Choices for Mobile App Recommendation, *ACM Transactions on Knowledge Discovery from Data*, Vol. 11, No. 2, pp. 1-23, May, 2017.
[15] B. Cao, X. Liu, J. Liu, M. Tang, Domain-aware Mashup Service Clustering based on LDA Topic Model from Multiple Data Sources, *Information and Software Technology*, Vol. 90, pp. 40-54, October, 2017.
[16] B. Cao, J. Liu, Y. Wen, H. Li, Q. Xiao, J. Chen, QoS-aware Service Recommendation Based on Relational Topic Model and Factorization Machines for IoT Mashup Applications, *Journal of Parallel and Distributed Computing*, Vol. 132, pp. 177-189, October, 2019.
[17] B. Cao, M. Peng, Y. Qing, J. Liu, G. Kang, B. Li, K. Fletcher, Web API Recommendation via Combining Graph Attention Representation and Deep Factorization Machines Quality Prediction, *Concurrency and Computation Practice and Experience*, Vol. 34, No. 21, Article No. e7069, September, 2022.
[18] J. Lin, K. Sugiyama, M. Kan, T. Chua, Addressing Cold-start in App Recommendation: Latent User Models Constructed from Twitter Followers, *The 36th International ACM SIGIR conference on Research and Development in Information Retrieval*, Dublin, Ireland, 2013, pp. 283-292.
[19] N. Chen, S. Hoi, S. Li, X. Xiao, SimApp: A Framework for Detecting Similar Mobile Applications by Online Kernel Learning, *The 8th ACM International Conference on Web Search and Data Mining (WSDM 2015)*, Shanghai, China, 2015, pp. 305-314.
[20] D. Cao, X. He, L. Nie, X. Wei, X. Hu, S. Wu, T. Chua, Cross-Platform App Recommendation by Jointly Modeling Ratings and Texts, *ACM Transactions on Information Systems*, Vol. 35, No. 4, pp. 1-27, October, 2017.
[21] M. Peng, G. Zeng, Z. Sun, J. Huang, H. Wang, G. Tian, Personalized App Recommendation based on App Permissions, *World Wide Web*, Vol. 21, No. 1, pp. 89-104, January, 2018.
[22] X. Xu, K. Dutta, A. Datta, C. Ge, Identifying Functional Aspects from User Reviews for Functionality-Based Mobile App Recommendation, *Journal of the Association for Information Science and Technology*, Vol. 69, No. 2, pp. 242-255, February, 2018.
[23] R. Salakhutdinov, A. Mnih, Probabilistic Matrix Factorization, *Advances in Neural Information Processing Systems*, Vancouver, British Columbia Canada, 2007, pp. 1257-1264.
[24] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T. Chua, Neural Collaborative Filtering, *Proceedings of the 26th International Conference on World Wide Web*, Perth, Australia, 2017, pp. 173-182.
[25] H. Wang, N. Wang, D. Yeung, Collaborative Deep Learning for Recommender Systems, *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Sydney, Australia, 2015, pp. 1235-1244.
[26] X. He, T. Chua, Neural Factorization Machines for Sparse Predictive Analytics, *ACM SIGIR Conference on Research and Development in Information Retrieval*, Shinjuku Tokyo Japan, 2017, pp. 355-364.
[27] H. Guo, R. Tang, Y. Ye, Z. Li, X. He, DeepFM: A Factorization-Machine Based Neural Network for CTR Prediction, *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 1725-1731.
[28] K. Gai, X. Zhu, H. Li, K. Liu, Z. Wang, Learning Piece-wise Linear Models from Large Scale Data for Ad Click Prediction, arXiv preprint arXiv: 1704.05194, April, 2017. https://arxiv.org/abs/1704.05194
[29] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu, T. Chua, Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks, *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3119-3125.
[30] Y. Yin, Z. Cao, Y. Xu, H. Gao, R. Li, Z. Mai, QoS Prediction for Service Recommendation with Features Learning in Mobile Edge Computing Environment, *IEEE Transactions on Cognitive Communications and Networking*, Vol. 6, No. 4, pp. 1136-1145, December, 2020.
[31] Y. Yin, Q. Huang, H. Gao, Y. Xu, Personalized APIs Recommendation with Cognitive Knowledge Mining for Industrial Systems, *IEEE Transactions on Industrial Informatics*, Vol. 17, No. 9, pp. 6153-6161, September, 2021.
[32] N. Xu, W. Mao, G. Chen, Multi-Interactive Memory Network for Aspect Based Multimodal Sentiment Analysis, *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, Honolulu, Hawaii, USA, 2019, pp. 371-378.
[33] J. Devlin, M. Chang, K. Lee, K. Toutanova, Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding, arXiv preprint arXiv: 1810.04805, October, 2018. https://arxiv.org/abs/1810.04805
[34] W. Zhang, T. Du, J. Wang, Deep Learning over Multi-field Categorical Data, *European Conference on Information Retrieval*, Padua, Italy, 2016, pp. 45-57.

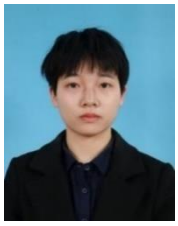
Biographies

Buqing Cao is now a professor in the School of Computer Science and Engineering and Hunan Key Laboratory for Service Computing and Novel Software Technology, at the Hunan University of Science and Technology, Xiangtan, China. His research interests include service computing and software engineering.



Weishi Zhong is an undergraduate student in the School of Computer Science and Engineering and Hunan Key Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology, Xiang-tan, China. Her research interests include mobile service classification and recommendation.

Xiang Xie is an undergraduate student in the School of Computer Science and Engineering and Hunan Key Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology, Xiangtan, China. Her research interests include service recommendation and graph neural networks.



Lulu Zhang is an undergraduate student in the School of Computer Science and Engineering and Hunan Key Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology, Xiang-tan, China. Her research interests include service classification and graph representation learning.

Yueying Qing is an undergraduate student in the School of Computer Science and Engineering and Hunan Key Laboratory for Service Computing and Novel Software Technology, Hunan University of Science and Technology, Xiangtan, China. Her research interests include service recommendation, and graph neural networks.