Decision Support Systems xxx (xxxx) xxx

Contents lists available at ScienceDirect

Decision Support Systems

journal homepage: www.elsevier.com/locate/dss

A deep multi-embedding model for mobile application recommendation

Yi-Hung Liu [a], Yen-Liang Chen [b,*], Po-Ya Chang [b]

[a] Department of Computer Science and Information Management, Soochow University, Taipei 100, Taiwan
[b] Department of Information Management, National Central University, Taiwan

ARTICLE INFO

Keywords:
Recommender system
Deep learning
Neural networks
Matrix factorization
Multi-embedding

ABSTRACT

With the explosion in the number of mobile applications, it becomes difficult and time-consuming for users to find the most suitable and interesting application from the millions of applications that exist today. Therefore, mobile app recommendation becomes a top priority to help people solve this problem. In previous studies, there are two common methods to solve this problem, one is to apply matrix factorization to the rating data to predict the target user's ratings for other applications, and the other is to predict ratings based on the user's review text. In addition to combining the above two traditional methods, this study also combines deep learning methods to generate better models. For rating data, we use a deep matrix factorization method to obtain latent factor vectors of users and items from the rating matrix to improve the embedding of ratings. For review text, we improve the text representation by using a multi-embedding approach. Most importantly, our model also incorporates deep neural networks with nonlinear transformation capabilities to extract more representative abstract features. We conduct experiments on datasets collected from the real world. Extensive experimental results show that our proposed method has better performance than other existing methods.

1. Introduction

With the popularity of the Internet, the problem of information overload has become increasingly prominent [1]. The emergence of recommender systems can help people filter complex information [2], further realizing a win-win situation for consumers and message senders. Therefore, recommender systems have been widely used and become one of the important components in the field of e-commerce, providing consumers with personalized information to help them make decisions. It can be applied in many contexts of e-commerce, such as movie recommendation [1], online news recommendation [3], paper recommendation [4], mobile marketing recommendation [5], etc.

Generally, there are two major kinds of studies in the recommendation system, namely, personalized ranking tasks [6] and rating prediction [7]. The task of personalized ranking is to provide people with a list of the top-N items in which they may be interested, while the latter approach, rating prediction, which predicts a particular user's rating of an item based on past ratings, is exactly the problem that we mainly address in this study. So far, researchers have explored various technologies to implement mobile application recommendation tasks from millions of applications. To address this problem, content-based filtering (CBF), collaborative filtering (CF), and hybrid filtering are three prevalent approaches commonly used by researchers [8–11]. A hybrid recommendation approach combines multiple techniques, leveraging their complementary advantages and overcoming their respective limitations while utilizing auxiliary information to alleviate data sparseness and cold-start [12,13]. Ling et al. [14] proposed a unified model that uses mobile app review text as auxiliary message and combined with rating data to alleviate the cold start problem, and achieved significant results. Inspired by this study, the methodology in this article uses a combination of mobile application ratings data and review text as inputs to the model.

In recent years, deep learning has emerged as a powerful machine learning technique [15]. Although the deep learning-based architecture can alleviate the data sparsity and cold-start problems of traditional recommendation techniques [16–18], it still has some challenges. According to Mikolov et al. [19], embedding is a crucial part of deep learning architecture since it converts messages into structured representations. Improving the encoding representation is vital for enhancing model performance as it can convey complex syntactic and semantic messages. Previous recommendation scenarios have used one-hot encoding as embedding technology. Nevertheless, it results in high-

* Corresponding author at: Department of Information Management, National Central University, No. 300, Zhongda Rd., Zhongli District, Taoyuan City 32001, Taiwan ROC
E-mail addresses: randyliu@scu.edu.tw (Y.-H. Liu), ylchen@mgt.ncu.edu.tw (Y.-L. Chen), 410535003@gms.ndhu.edu.tw (P.-Y. Chang).

https://doi.org/10.1016/j.dss.2023.114011
Received 6 July 2022; Received in revised form 19 April 2023; Accepted 14 May 2023
Available online 15 May 2023

dimensional and sparse vector representation, which is not conducive to deep learning architectures due to its large memory space and time consumption.

Considering the importance of embedding, we adopt different embedding techniques for review text and rating data in the first layer of the model to help convert high-dimensional sparse vectors into low-dimensional ones to represent certain characteristics of the corresponding item. However, popular text encoding models like word2vec [19] and Glove [20] are based on static word vectors, which are efficient but can't handle word context as only one vector can be learned for a word. Recently, dynamic word vector representations based on language models have received wide attention in the NLP research field, such as ELMo (Embeddings from Language Models) [21], BERT (Bidirectional Encoder Representations from Transformers) [22], etc. The major difference of pre-trained language model with static word embedding methods is that the word vectors generated by the dynamic word vector representation can be differentiated depending on different context. Among them, BERT is designed to pre-train large corpora using deep bidirectional transformers, which pre-trains unlabeled text representations by jointly adjusting left and right context across all layers, resulting in exceptional performance.

Whereas methods introduced by [23,24] combine different pre-trained encoders, resulting in further improvements to model performance. When multiple embedding representations are fused into one multi-embedding, there is an opportunity to enhance the representation of text as it can carry richer information. This augmentation is particularly effective in short text tasks [25,26]. Therefore, we aim to combine encoded representations with diverse strengths to enhance the representation of review texts in mobile applications.

As BERT [22] has received widespread attention, some scholars believe it suffers from undertraining because of its masking approach. Therefore, BERT-based variants have been proposed. Among them, Liu et al. found the under-training problem of BERT, so they improved the pre-training method based on BERT and proposed a robustly optimized scheme-RoBERTa [27], which considered the effect of key hyper-parameters and the size of training data. Finally, they prove that a large batch training size is more effective. New language models are emerging by continuously improving pre-training processes, emphasizing the importance of constructing better embeddings to enhance NLP tasks. To enhance review text representation in mobile applications, we propose combining embeddings generated from BERT and RoBERTa to implement multi-embeddings, integrating the advantages of both encoding methods.

On the other hand, since matrix factorization (MF) has good predictive power, it is mainly used for the final score prediction of the model, as Gan et al. [28]. Although we are also interested in its accuracy of predictions, the approach in this article is applying matrix factorization techniques for ratings' embedding. In practice, we use a general framework similar to that proposed by He et al. [29] - Using Neural Collaborative Filtering (NCF) and Matrix Factorization Embeddings as input to deep learning models to improve user vector representation and item vector representation, thereby increasing the accuracy of model accuracy. That is, for rating data processing, we use deep matrix factorization as the embedding method in order to enhance the representation and to learn more accurate recommendation results. Overall, the user rating data and unstructured text comments of the application are jointly considered in this article and applied to the recommendation task of the mobile application.

The contributions of this paper are as follows. First, the proposed method can enhance the accuracy of mobile app recommendations by analyzing both ratings and reviews. While prior efforts have largely disregarded the value of reviews, this method can capture more nuanced and diverse user feedback that is not reflected in rating data alone. This leads to more relevant and personalized app recommendations. Second, this study shows that extracting more representative and abstract features from user data and items through deep neural networks can provide app developers gain better understanding of user preferences and feedback, thereby enhancing app design, functionality, and user experience. Lastly, we contribute to the development of a versatile and flexible recommendation model that utilizes ratings and reviews data as input in diverse contexts, which allows results to be more generalizable. The proposed model can be a valuable resource for researchers and practitioners interested in solving recommendation-related issues.

The rest of this article is organized as follows. First, we present a review of related work in Section 2. Next, we present our proposed model in detail in Section 3. Then, Section 4 contains a series of experiments and extensive evaluations to prove the effectiveness of the model. Our discussion of this study is in Section 5. Finally, conclusions and future works are presented in Section 6.

2. Related work

In this section, we introduce the recommendation methods in the past. It will be divided into three parts. First, we introduce traditional mobile applications recommendation methods. In the second part, we will introduce deep learning mobile applications recommendation methods. Finally, we introduce BERT-based recommendation methods. Table 1 summarizes relevant traditional and deep learning mobile applications recommendation studies in recent literature.

2.1. App recommendation in traditional recommendation

In the following, we present three common traditional recommendation methods that are used for mobile application recommendations, including content-based, collaborative filter-based, and hybrid methods.

2.1.1. Content-based mobile applications recommendation

Cao & Lin [30] surveyed existing research on mining smartphone

Table 1
Previous studies on app recommendation.

Work	Data	Method	Outcome
Shi & Ali [8]	app log data	PCA (Principal Components Analysis) - based model	Top N items
Liu & Jiang [9]	users, items, ratings and log data	Hierarchical embedding model	Top N items
Yu et al. [10]	app log data	Collaborative filtering topic modeling	Top N items
Wu & Zhu [11]	app log data	Collaborative filtering RFD (Recency, Frequency, Duration) model Hybrid latent factor model	Top N items
Woerndl et al. [12]	users, items, and ratings	Hybrid model – Content-based filtering Collaborative filtering	Top N items
Costa-Montenegro et al. [13]	user's interaction data rating data	Hybrid model – Content-based filtering Collaborative filtering	Top N items
Guo et al. [16]	app metadata	Factoring machine Attention mechanism	Rating prediction
Li et al. [17]	users, items, and ratings	Gated Recurrent Unit (GRU) Multilayer Perceptron (MLP)	Rating prediction
Khan et al. [18]	Screenshots app metadata	Convolutional Neural Network (CNN)	Point of Interest prediction
He et al. [29]	users, items, and ratings	Neural network-based Collaborative Filtering (NCF)	Top N items
Xue et al. [31]	users, items, and ratings purchase history	Deep Matrix Factorization Model (DMF)	Top N items

data to uncover app usage patterns, including mobile app recommendations. Mobile application datasets tend to have more polarized data distribution than general scenarios such as movie recommendations, as noted by Shi & Ali [8]. To address this gap, they proposed a content-based model called Eigenapp to provide more attractive recommendations for application recommendation tasks.

Liu & Jiang [9] proposed a personalized application recommendation method based on hierarchical embedding. It analyzes the latent structure of data and constructs hierarchical embedded models to predict user interest in applications, achieved by matching weighted features in hierarchical associations with keyword queries or users' past records.

2.1.2. Collaborative filter-based mobile applications recommendation

Unlike content-based filtering methods mentioned above, which require additional content information as an auxiliary, collaborative filtering methods rely only on past user behavior (explicit feedback), such as browsing history or item rating, to implement recommendations. It is also one of the traditional recommendation methods that are widely used.

AppLibRec [10] combines collaborative filtering and topic modeling techniques for application recommendations. Wu & Zhu [11] marked users' preferences using the RFD (Recency, Frequency, Duration) model and proposed two hybrid models based on collaborative filtering: IIOCF and HLF. IIOCF (Item-Oriented Collaborative Filtering) improves item-oriented approaches, while HLF (Hybrid Latent Factor) integrates both latent factor and item-oriented predictions to capture the advantages of both, and combines them for mobile application recommendations.

2.1.3. Hybrid mobile applications recommendation

Although collaborative filtering-based approach avoids the possibility of incomplete content analysis by sharing the experience of others, and allows filtering based on complex, indescribable concepts, collaborative filtering approach is still limited in practice by severe data sparsity and cold start problems for new users or items. One possible solution is to blend recommendations by combining content-based and collaborative filtering methods.

Woerndl et al. [12] applied a hybrid recommendation system that recommends suitable apps based on user downloads in similar environments and uses randomness to provide diverse recommendations. A comprehensive solution [13] was proposed by Costa-Montenegro et al. who used five recommendation filtering techniques to obtain accurate and diverse recommendations. Among them, the content-based recommender effectively addresses cold start and data sparsity problems by recommending new applications based on users' previously downloaded app usage patterns. The authors showed that the results of the five filtering methods are complementary and effective.

The methods mentioned above involve using a user-item matrix and integrate additional information, related to users, products, and the interaction between them. However, they have limitations such as cold-start and data sparsity problems, the need for large datasets, and high computational complexity requirements, etc. To address these issues, we applied a deep matrix factorization model combined with a multi-embedding model to predict the target user's rating based on the large dataset of ratings and the user's reviews.

2.2. App recommendation in deep learning

Deep learning models have made significant progress in speech recognition, machine translation, and natural language processing by automatically learning high-level distributed representations, eliminating the need for costly manual feature extraction. A variety of deep learning frameworks are also widely used in mobile application recommendation tasks.

Guo et al. [16] proposed Knowledge-based Deep Factorization Machine (KDFM), a recommendation model based on factoring machine and attention deep learning techniques that exploit extensive categorization and textual knowledge in the market. Khan et al. [18] constructed a mobile application recommendation system using screenshots and user preferences through a CNN-based approach. They employed a pre-trained model and linear regression to improve the relevance of the recommended applications. Li et al. [17] introduced the NRT framework, which predicts application ratings and generates abstract tips. The abstract tips are produced using a Gated Recurrent Unit (GRU), while the neural scoring regression employs MLP. End-to-end trained multi-task learning methods are used to learn all parameters in both the GRU and MLP.

He et al. [29] proposed a general framework called Neural Collaborative Filtering (NCF), which is an abbreviation of neural network-based collaborative filtering, where we can express and generalize matrix factorization. Since NCF uses two paths to model users and items, it is intuitive to combine the features of the two paths by connecting them. However, the simple connection of vectors does not consider any interaction between users and items. It is not sufficient to model the collaborative filtering effect, so they add hidden layers to the concatenated vectors and then input them into the multi-layer neural architecture to learn the potential features between users and items interactions.

Our study combines review text and rating information. Unlike most deep learning-based recommendations for mobile applications, our work emphasizes the improvement of embedding. We use different coding techniques for review text and rating information to enhance their vector representation. Inspired by NCF [29], we concatenate vectors that represent users and items, respectively, after applying deep matrix factorization (DMF) [31]. Then, we input them into a multi-layer neural network architecture to learn the interaction of potential features between them. Regarding the review text, we input the concatenated review latent factors into a deep neural network (DNN) to learn the corresponding vectors. Finally, the training results from ratings and reviews are concatenated and passed through a final neural network to generate the final prediction ratings, which enhance mobile application recommendations.

2.3. BERT-based recommendation

Several BERT-based recommender systems exist for different fields, each with unique architectures, traits, and limitations. In citation recommendation, there are few recent works, such as the context-aware citation recommendation system [32]. This system uses graph convolutional networks and BERT on a modified PeerRead dataset containing cited references and paper metadata. In the news domain, Jia et al. [33] developed the Recurrent Reasoning Memory Network over BERT (RMBERT) recommendation system, which uses attention-based reasoning steps to dynamically learn user and news vectors and model their interaction. The HRM can track customer feedback and understand their preferences in e-commerce [34]. Cantiti et al. [35] combined BERT with a global expansion strategy for hashtag recommendation, and Che and Sun [36] developed a mobile app recommendation system for designers that uses Labeled-LDA text classification and BERT similarity matching to filter features for specific scenarios and user groups.

Despite the recent promising BERT-based recommender systems, we believe they have not yet incorporated a multi-embedded model that combines rating and review features to enhance recommendation accuracy for users. This study aims to predict a user's rating for items they have not yet rated. Predictions can come from explicit user feedback, like past ratings on items, or implicit feedback, such as user reviews that express opinions. Recent studies have shown that user reviews are strong predictors of implicit feedback or even ratings and should be incorporated into the recommendation system. Recent works on recommendation systems using reviews have focused on item recommendation or only used review content, ignoring user ratings. This motivated us to propose a rating prediction framework to help users automatically

discover mobile apps they may enjoy, without external assistance. By doing so, we believe that this study can provide a more intelligent and credible app rating recommendation framework.

3. Proposed approach

This section presents the proposed model, starting with main notations and problem formulation. Then, the model architecture, two modules, and final objective function are described in detail.

3.1. Task definition and notation

In this subsection, we first formalize the problem. Given a dataset $\mathscr{D}$ consisting of a user set $\boldsymbol{U}$ and item set $\boldsymbol{V}$, each sample $(u_i, v_j, r_{i,j}, w_{i,j})$ denotes a review $w_{i,j}$ written by the user u_i on the item v_j, and the corresponding rating $r_{i,j}$, where $\boldsymbol{U} = \{u_1, \ldots, u_{|U|}\}$, $\boldsymbol{V} = \{v_1, \ldots, v_{|V|}\}$, $|\boldsymbol{U}|$ denotes the number of users, $|\boldsymbol{V}|$ denotes the number of items, $r_{i,j} \epsilon \mathbf{R}$, $w_{i,j} \epsilon \mathscr{D}$.

Our goal is to train a model that is capable of predicting ratings based on the rating matrix $\mathbf{R} \epsilon \mathbb{R}^{|U|\times|V|}$, and the review $w_{i,j}$. The final goal is to find the predicted rating $\hat{r}_{i,j}$ that has the smallest error with the actual rating $r_{i,j}$.

3.2. Framework

This subsection presents our proposed model, which consists of two main sub-modules: the rating learning module and the review learning module. Fig. 1 shows an overview of the system architecture.

In the rating learning module, we use the rating matrix $\mathbf{R}$ as input and adopt the deep matrix factorization (DMF) method to extract the rating embedding vector p_i of user i and the rating embedding vector q_j of item j from the rating matrix $\mathbf{R}$, and then send it to the deep neural network to learn the corresponding rating prediction vector $\boldsymbol{rt}_{i,j}$. On the other hand, we represent the review text written by user i for item j as $w_{i,j}$ and input it into the review learning module. The latent representations of user reviews learned by the two embedding methods are concatenated and input into the deep neural network architecture to obtain the corresponding review prediction vector $\boldsymbol{rv}_{i,j}$.

After obtaining the final vectors of the two learning modules separately, we concatenate $\boldsymbol{rt}_{i,j}$ with $\boldsymbol{rv}_{i,j}$ to generate the final prediction rating $\hat{r}_{i,j}$ of user i for item j through the neural network.

3.3. Rating learning module

As mentioned in Section 2, Xue et al. [31] proposed a deep matrix factorization model (DMF) incorporating a neural network architecture. They take the user-item matrix with explicit ratings as input, and learns the common low-dimensional space to represent users and items through deep structure. Our rating learning module is inspired by their work and refers to their proposed architecture The left part of Fig. 1 shows the detailed structure of our rating learning module. First, it is known that we have $|\mathbf{U}|$ users and $|\mathbf{V}|$ items, and a rating matrix $\mathbf{R} \epsilon \mathbb{R}^{|U|\times|V|}$ constructed from the rating data, as in Eq. (1):

$$\mathbf{R} = \begin{cases} r_{i,j}, \text{if a rating of user } i \text{ on item } j \text{ is observed;} \\ 0, \text{otherwise} \end{cases} \tag{1}$$

In the preprocessing step, we first take the given user-item rating matrix $\mathbf{R} \epsilon \mathbb{R}^{|U|\times|V|}$ as input, which $\mathbf{R}_{i,*} \epsilon \mathbb{R}^{1\times|V|}$ represents the high-dimensional rating vector of user i for all items; similarly, $\mathbf{R}_{*,j} \epsilon \mathbb{R}^{|U|\times 1}$ represents the high-dimensional rating vector of item j. Then, in the deep matrix factorization with M-layer neural network, the input vector of each layer will be mapped to another new vector in the latent space. Assuming that the initial input vector is x, l_m $(m = 1, \ldots, M-1)$ represents the vector in the m-th hidden layer, W_m denotes the weight matrix of the m-th layer, and b_m is the bias of the m-th layer. We can obtain the latent representation of the m-th layer neural network output h by Eq. (2), where $(\cdot)$ denotes the nonlinear activation function.

$$\begin{aligned} l_1 &= W_1 x \\ l_m &= f(W_m l_{m-1} + b_m), m = 2, \ldots, M-1 \\ h &= f(W_M l_{M-1} + b_M) \end{aligned} \tag{2}$$

Same as [31], we use rectified linear units (ReLU) as the hidden layer activation function (as shown as Eq. (3)) of the $l_m, m = 2, \ldots, M-1$.

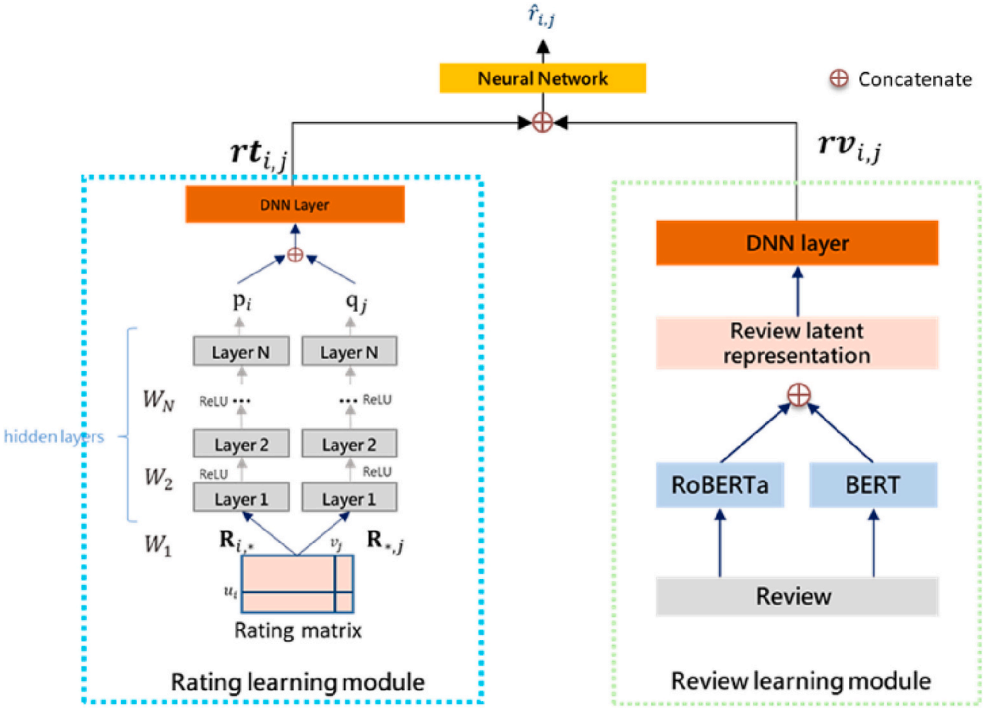

Fig. 1. Architecture of the deep learning model.

$$f(x) = max(0, x) \tag{3}$$

In our rating learning module, the deep matrix factorization framework consists of two parallel multi-layer networks, which are used to map the user rating vector $\mathbf{R}_{i,*}$ and the item rating vector $\mathbf{R}_{*,j}$ to a low-dimensional vector in the latent space, thus we have Eq. (4), where W_{U1} and W_{V1} denote the weight matrix of user set $\boldsymbol{U}$ and item set $\boldsymbol{V}$ at the first level, respectively; similarly, W_{U2} and W_{V2} are the weight matrices at the second level, and so on.

$$\begin{aligned} p_i &= f_{\theta_M^U}\left(\ldots f_{\theta_3^U}\left(W_{U2} f_{\theta_2^U}\left(W_{U1}\mathbf{R}_{i,*}\right)\right)\ldots\right) \\ q_j &= f_{\theta_M^I}\left(\ldots f_{\theta_3^I}\left(W_{V2} f_{\theta_2^I}\left(W_{V1}\mathbf{R}_{*,j}^T\right)\right)\ldots\right) \end{aligned} \tag{4}$$

Next, we concatenate p_i with q_j and learn the rating prediction vector $\boldsymbol{rt}_{i,j}$ of user i on item j in the rating learning module through a set of deep neural network (DNN) framework as in Eq. (5).

$$\begin{aligned} y &= p_i \oplus q_j, l_1 = W_1 y \\ l_n &= f'(W_n l_{n-1} + b_n), n = 2, \ldots, N-1 \\ \boldsymbol{rt}_{i,j} &= f'(W_N l_{N-1} + b_N) \end{aligned} \tag{5}$$

3.4. Review learning module

Previous approaches [37] used neural networks to learn users' latent representations from all review text. However, according to [38], the target user's review on the target item contains valuable information. Therefore, in the review learning module, we use the target user's review of the target item. The right part of Fig. 1 shows the details of the review learning module.

In the first stage of review learning, we input the user i's review $w_{i,j}$ on item j to both the two different embedding methods BERT and RoBERTa to generate the respective corresponding latent factor representations $o_{i,j}^{BERT}$ and $o_{i,j}^{RoBERTa}$.

3.4.1. BERT embedding

BERT [22] is a linguistic model trained by Google in 2018 using an unsupervised manner. We use BERT as an embedding method for review text because it captures the context and explicit differences between words, providing a more accurate representation than previous static word vectors.

Before we input the text data into BERT, it will have to go through three embedding layers. First, each review text $w_{i,j}$ is divided into words using WordPiece embeddings to obtain a list of words. Then, special tokens [CLS], [SEP], and [PAD] are added to them. The token [CLS] represents the classification task; [SEP] represents the interval between sequences, [PAD] is used to unify the length of a sequence. Then, all single words and special tokens are mapped to the corresponding word vectors.

After token embedding comes segment embedding, whose task is to distinguish between two given sentences, that is, to tell the model which words belong to the previous sentences and which belong to the later ones. This stage will output two kinds of embedding, $\boldsymbol{E_A}$ and $\boldsymbol{E_B}$ respectively. If there is only one sentence, only $\boldsymbol{E_A}$ is output.

In addition, adding a position embedding layer is necessary in BERT because its transformer encoder does not contain positional encoding. Then, the three layers of embedding are combined to obtain the final representation of the BERT model. When loading the pre-trained model, we choose the BERT-base-uncased version, which contains 12 layers, 768 hidden units, and 12 attention heads. The output of the BERT model is a vector of size 768 for each location, as shown in Fig. 2. According to [22], we can use the output of the first position (i.e., [CLS] token) vector of each sentence as the semantic representation of the sentence since [CLS] has a fully concatenated layer connected to all other tokens. Therefore, after the processing of BERT, we take out the final hidden state output corresponding to the [CLS] token in the text $w_{i,j}$ and represent it as $o_{i,j}^{BERT}$ to be the final BERT encoding representation, as in Eq. (6).

$$o_{i,j}^{BERT} = \text{BERT}\left(w_{i,j}\right) \tag{6}$$

3.4.2. RoBERTa embedding

In 2019, Facebook AI and Liu et al. proposed RoBERTa [27] as a better training method to optimize BERT after evaluating its pre-trained model and concluding that it was not sufficiently trained. RoBERTa achieved the most advanced results in GLUE and SQuAD at that time. Hence, we utilize RoBERTa as one of the embedding methods for review text.

RoBERTa, like BERT, has Transformer layers that can learn contextualized text representations through self-attention. The key difference is that RoBERTa uses longer and more sentences for training, eliminates the next sentence prediction task and employs a dynamic masking strategy. Additionally, RoBERTa is not case-sensitive during pre-training. We use RoBERTa-base, with a 12-layer transformer encoder and 12-head attention unit per layer, producing a 768-dimensional tensor at each position.

Similar to BERT, we first split the review text $w_{i,j}$ and get n single words $\{w_1, \cdots, w_n\}$ before input to RoBERTa model. Next, we add a special token, using the [CLS] token as the beginning of the text sequence, and use it as input to RoBERTa after implementing the padding operation. As with BERT, we take the [CLS] token vector of review $w_{i,j}$ as the final output of the sentence with dimension 768*1, which is denoted as $o_{i,j}^{RoBERTa}$, as shown in Eq. (7) and Fig. 3.

$$o_{i,j}^{RoBERTa} = \text{RoBERTa}\left(w_{i,j}\right) \tag{7}$$

3.4.3. Combine two kinds of embedding

Subsequently, we concatenate two latent representations from different embedding methods to obtain a comprehensive latent representation $o_{i,j}$ of the review $w_{i,j}$,(as)

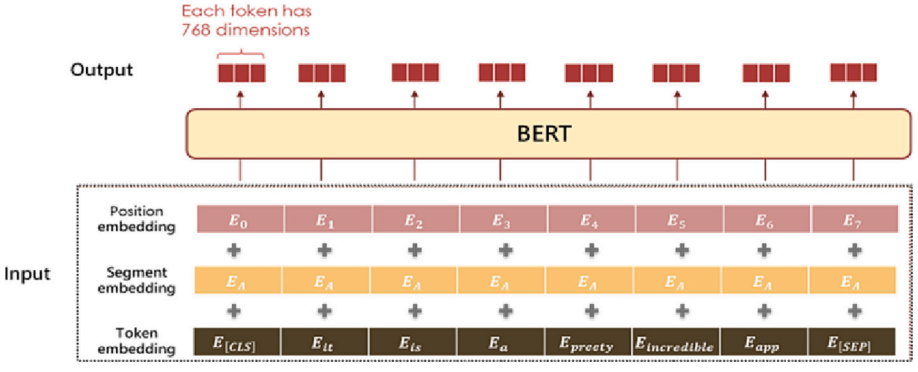

Fig. 2. The input and output of BERT model.

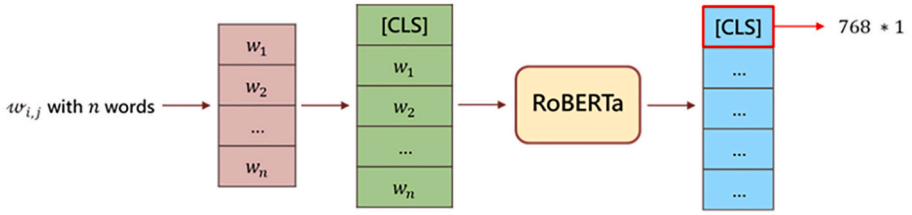

Fig. 3. The embedding process of RoBERTa model.

$$o_{i,j} = o_{i,j}^{BERT} \oplus o_{i,j}^{RoBERTa} \tag{8}$$

In order to learn deeper features and better capture dependencies between words, we input the previously obtained comprehensive review latent representation $o_{i,j}$ into the deep neural network architecture, where l_s (s $= 1,\ldots,S-1$) represents the s-th hidden layer, W_s denotes the weight matrix of the sth layer, and b_s is the bias of the s-th layer. We can obtain $\boldsymbol{rv}_{i,j}$, the latent representation of the output of the neural network at the S-th layer using Eq. (9).

$$\begin{aligned} l_1 &= W_1\, o_{i,j} \\ l_s &= f(W_s l_{s-1} + b_s), s = 2, \ldots, S-1 \\ \boldsymbol{rv}_{i,j} &= f(W_S l_{S-1} + b_S) \end{aligned} \tag{9}$$

3.5. Rating-review combined

Subsequently, we concatenate the final prediction vectors obtained from the two sub-modules, $\boldsymbol{rt}_{u,i} \in \mathbb{R}^{d_{rt} \times 1}$ and $\boldsymbol{rv}_{u,i} \in \mathbb{R}^{d_{rv} \times 1}$, and then input them to the last neural network to learn the final prediction ratings. In the experiment, we set both dimensions d_{rt} and d_{rv} to 64, where l_t ($t = 1, \ldots, T-1$) represents the t-th intermediate hidden layers in the neural network, $\mathbf{z}$ is the first input of the deep neural network, W_t represents the weight matrix of the t-th layer, and b_t is the bias of the t-th layer, as Eq. (10).

$$\begin{aligned} z &= \boldsymbol{rt}_{u,i} \oplus \boldsymbol{rv}_{u,i} \\ l_1 &= W_1\, z \\ l_t &= f(W_t l_{t-1} + b_t), t = 2, \ldots, T-1. \end{aligned} \tag{10}$$

Finally, we can obtain the output of the neural network at layer T as a latent representation of the predicted rating $\hat{r}_{i,j}$ as shown in Eq. (11)

$$\hat{r}_{i,j} = f(W_T l_{T-1} + b_T) \tag{11}$$

For the case that all latent vectors are real numbers, one typical objective function is the minimization of mean-square error (MSE). In order to optimize the model, we defined the objective function of the final DNN in the architecture as Eq. (12), where θ denotes all parameters of the model, $r_{i,j}$ is the actual rating and $\hat{r}_{i,j}$ is the final predicted rating of the model.

$$\min_{\theta} \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} \left(r_{i,j} - \hat{r}_{i,j}\right)^2 \tag{12}$$

4. Experiments and results

4.1. Datasets

In order to verify the effectiveness of our proposed model, we use a real-world dataset crawled from Google Play Store through a crawler. Before starting the experiment, we first pre-processed the dataset and used users' image links as the criteria for identifying the same user, and removed all user data that used Google's default image as the cast image since they corresponded to the same image link. Next, we remove repetitive columns, punctuation marks, single characters, and deactivated words. Then, we retained the data which has both rating and review text. Finally, we removed the data of users with less than six reviews of the application. Overall, the field information for each data entry in the dataset contains the following: User ID, Application ID, Rating, Review text.

Finally, the entire dataset we use is composed of 1,116,196 ratings and reviews provided by 135,331 users on 9095 applications, with the minimum rating of 1, the maximum rating of 5, and the rating scale is increased by 1. On average, each user included 8.2 ratings and reviews, and each application included an average of 122 ratings and reviews.

4.2. Experimental benchmark

We use nine common recommended algorithms as baselines: PMF [39], SVD++ [40], MF [41], NMF [42–44], DMF [33], DeepCoNN [37], NARRE [45], MPCN [46], and DAML [47].

1. **PMF**: Probabilistic matrix factorization introduces a Gaussian distribution and uses the rating matrix as an input to model the latent factors of the users and the items.
2. **SVD++**: SVD++ improves the accuracy of the model in terms of rating prediction by integrating implicit information into a model based on singular value decomposition.
3. **MF**: Matrix factorization, a widely used collaborative filtering method, predicts ratings using only the rating matrix as input and estimates two low-rank matrices.
4. **NMF**: Non-negative matrix factorization enforces non-negativity by adjusting the learning rate and rescaling it during parameter updates to preserve non-negativity.
5. **DMF**: Deep matrix factorization is a neural network model that learns a low-dimensional space for representing users and items from input user-item matrices. It is a leading representation learning matrix factorization technique.
6. **DeepCoNN**: Deep cooperative neural network is the first method for modeling user-item associations based on reviews using TextCNN. It employs a factorization machine to predict ratings and a pre-trained deep model for representing review text as single-word embeddings.
7. **NARRE**: Neural attentional rating regression adopts matrix factorization to obtain user/item representations from ratings and learns the importance of reviews for better results at the final prediction layer.
8. **MPCN**: Multi-pointer co-attention networks use multi-pointer learning and co-attention framework to extract critical information from user and item reviews, providing interpretability in this review-based recommendation system.
9. **DAML**: Dual attention mutual learning is a dual attention model implemented by using two convolutional structures with combined rating and review features. Finally, the prediction is performed by factorization machine.

4.3. Experimental Setting

Before starting the experiment, we divided the dataset into a training set, a validation set, and a test set in an 8:1:1 manner for performance evaluation. Details regarding the experimental data are shown in Table 2. A sensitivity analysis will be performed to determine the optimal values for the hyperparameters, which will then be used in the subsequent experiments. We optimized our neural network using the

Table 2
The amount of training, verification, and testing data used in this study.

	Training Set	Validation Set	Test Set	Total
#NUM	907,800	100,865	112,073	1,116,196

Adam optimizer [48], and determined the learning rate, batch size, and epoch through sensitivity analysis for the best mean squared error (MSE) performance. Additionally, the number of layers and neurons in the network were determined based on the optimal performance of different modules.

In the rating learning module, the user and item embedding layers are set up as a 2-layer neural network with 128 and 64 neurons in [31]. After concatenating the user and item vectors, we input them into a 2-layer neural network and obtained the best results when the numbers of neurons are 128 and 64 according to the sensitivity analysis.

For the review learning module, we performed sensitivity analysis on the numbers of layers and neurons of the neural network. When concatenating the two modules, we also conducted a sensitivity analysis for d_{rt} and d_{rv}, and set them to 64.

In regards to the baseline methods, we refer to the suggested parameters in the authors' original article for setting, and use the same loss function as our model. Some parameters not mentioned in the original article use the same settings as in this article. Detailed experimental descriptions and results are presented in the following.

4.4. Evaluation metric

To compare with baseline methods, we utilized four widely-used evaluation metrics for rating prediction tasks in recommendation systems: mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). They are defined in the following Eqs. (13)–(16), respectively, where r_{ij} represents the ground-truth rating of user i to item j, and $\hat{r}_{i,j}$ is corresponding predicted rating, and N indicates the number of ratings between users and items.

$$MSE = \frac{1}{N}\sum_{n=1}^{N}\left(r_{ij} - \hat{r}_{i,j}\right)^2 \tag{13}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(r_{ij} - \hat{r}_{i,j}\right)^2} \tag{14}$$

$$MAE = \frac{1}{N}\sum_{n=1}^{N}\left|r_{ij} - \hat{r}_{i,j}\right| \tag{15}$$

$$MAPE = \frac{100\%}{N}\sum_{n=1}^{N}\left|\frac{r_{ij} - \hat{r}_{i,j}}{r_{ij}}\right| \tag{16}$$

4.5. Experimental design

In order to verify the effectiveness of our method proposed in this article, we divided the experiment into three parts in this study.

1. The sensitivity analysis of hyperparameters, including learning rate, batch size, epoch, neural network layers of review learning module, the number of neurons in rating learning module and review learning module, and the dimension for concatenation of two modules, respectively.
2. In order to demonstrate the advantages of combining rating-review and multiple embedding technique, we conducted the following two set of ablation experiments, and measured the performance by calculating various metrics.
   - We will keep only the rating learning module in the full model, and only the review learning module in the full model, for comparison with the results of the full model.
   - We will remove the review learning module with the BERT embedding and the review learning module with the RoBERTa embedding for comparison with the full review learning module.
3. In order to demonstrate the performance of the full model for the mobile application recommendation task, we compared it with other baseline methods on the dataset crawled from Google Play Store.

4.6. Experimental platform

All of our experiments are implemented in Python using the Pytorch framework. The experiments were conducted on a computer equipped with an Intel Core i5–10,400 CPU @ 2.90GHz, 128GB RAM, and NVIDIA GeForce RTX 3090 GPU.

4.7. Experimental results and discussion

In this part, we utilized the dataset collected from Google Play Store to conduct our experiment, and present the results in the following tables. In total, five models were evaluated, including the results of the review learning module with only RoBERTa embedding (RV1-RoBERTa), the review learning module with only BERT embedding (RV1-BERT), a full review learning module (RV2), a full rating learning module (RA), and using the entire model for recommendation (RARV2). Tables 3-9 show the results of parameter sensitivity analysis for our proposed full model, while Tables 10-11 present the ablation study analysis for the entire proposed model. Finally, we compare the performance of our model with all baseline methods in Table 12.

4.7.1. The sensitivity analysis of parameters

Table 3 shows the selection of different learning rates of the whole model in this article (RARV2). According to the results of the sensitivity analysis, it is found that the best performances of the four metrics are achieved at the learning rate = 0.0001, hence, we set the learning rate of our model to 0.0001.

Table 4 shows the comparison of different batch size of the whole model in this article. According to the results of the sensitivity analysis, we can find that the best performances of the four metrics are achieved in the case of batch size = 32, hence, we set the batch size of the model in this article to 32.

Table 5 shows the comparison of different epoch of the whole model in this article. According to the results of the sensitivity analysis, we can find that the best performances of the five metrics are achieved in the case of epoch =50, hence, we set the epoch of the model in this article to 50.

Table 6 shows the comparison of different layers of the review learning module in our model. According to the results of the sensitivity analysis, it is found that with 2 layers, the best performance is achieved in all four metrics as compared to only 1 layer of the network, hence, we set the layers of the review learning module to 2.

Table 7 shows a comparison of the different neurons of the review learning module in our model. According to the results of sensitivity analysis, it is found that these four metrics perform best in the case of 2048 and 1024 neurons, so we use it as the setting for the review learning module in this paper.

Table 8 shows a comparison of the different neurons of the rating learning module in our model. According to the results of sensitivity analysis, it is found that the four metrics perform best in the case of 128 and 64 neurons, so we use it as the setting of the rating learning module

Table 3
The learning rate of our whole model.

RARV2	MSE	RMSE	MAE	MAPE
learning rate = 0.001	0.6991	0.8361	0.5886	27.8817
learning rate = 0.0001	0.6189	0.7867	0.5120	25.5996
learning rate = 0.00001	0.6316	0.7947	0.5258	27.101

Table 4
The batch size of our whole model.

RARV2	MSE	RMSE	MAE	MAPE
batch size = 128	0.6263	0.7914	0.5206	26.1706
batch size = 64	0.6225	0.7890	0.5189	25.9817
batch size = 32	0.6189	0.7867	0.5120	25.5996
batch size = 16	0.6217	0.7885	0.5178	25.8196

Table 5
The epoch of our whole model.

RARV2	MSE	RMSE	MAE	MAPE
epoch = 20	0.6368	0.7980	0.5358	27.364
epoch = 30	0.6248	0.7905	0.5269	27.5153
epoch = 40	0.6214	0.7883	0.5169	26.4192
epoch = 50	0.6189	0.7867	0.5120	25.5996
epoch = 60	0.6205	0.7877	0.5151	26.1241

Table 6
The layer of the review learning module in our model.

RARV2	MSE	RMSE	MAE	MAPE
layer = 1; neuron = 1024	0.6616	0.8134	0.5661	28.6275
layer = 1; neuron = 2048	0.6484	0.8052	0.5580	28.1315
layer = 2; neuron = 2048, 1024	0.6189	0.7867	0.5120	25.5996

Table 7
The neuron of the review learning module.

RARV2	MSE	RMSE	MAE	MAPE
layer = 2; neuron = 512,256	0.6351	0.7970	0.5338	26.1873
layer = 2; neuron = 1024, 512	0.6213	0.7882	0.5188	26.1460
layer = 2; neuron = 2048, 1024	0.6189	0.7867	0.5120	25.5996
layer = 2; neuron = 4096, 2048	0.6207	0.7878	0.5183	25.9716

Table 8
The neuron of the rating learning module in our model.

RARV2	MSE	RMSE	MAE	MAPE
layer = 2; neuron = 512, 256	0.6273	0.7920	0.5274	26.5673
layer = 2; neuron = 256, 128	0.6197	0.7872	0.5213	26.2012
layer = 2; neuron = 128, 64	0.6189	0.7867	0.5120	25.5996
layer = 2; neuron = 64, 32	0.6239	0.7899	0.5232	26.8849

Table 9
The neuron of the rating learning module in our model.

RARV2	MSE	RMSE	MAE	MAPE
$d_{rt} = d_{rv} = 32$	0.6210	0.7880	0.5211	26.9748
$d_{rt} = d_{rv} = 64$	0.6189	0.7867	0.5120	25.5996
$d_{rt} = d_{rv} = 128$	0.6227	0.7891	0.5284	26.3583

Table 10
The importance of combining ratings and reviews as inputs in our model.

Model	MSE	RMSE	MAE	MAPE
RARV2	0.6189	0.7867	0.5120	25.5996
RA	1.7511	1.3233	0.9855	50.8045
RV2	0.6766	0.8226	0.5302	26.6524

in this paper.

Table 9 shows the comparison of different dimension settings for concatenation of two modules. According to the results of the sensitivity analysis, when concatenating two modules, the best performances of the four metrics are achieved in the case of d_{rt}=d_{rv}= 64, hence, we set the dimension in this article to 64.

Table 11
The results of ablation study in our review learning modules.

Model	MSE	RMSE	MAE	MAPE
RARV2	0.6189	0.7867	0.5120	25.5996
RV2	0.6766	0.8226	0.5302	26.6524
RV1-RoBERTa	0.7360	0.8579	0.5726	29.2122
RV1-BERT	0.7310	0.8550	0.5593	28.3552

Table 12
The integration of experimental results on Google Play Store dataset.

	Model	MSE	RMSE	MAE	MAPE
Rating	MF	4.3233	2.0793	1.6235	53.7283
	SVD++	2.0103	1.4179	1.0997	59.1324
	NMF	2.0451	1.4301	1.0600	50.3649
	PMF	2.7524	1.6590	1.5205	80.5468
Rating + DL	DMF	1.7630	1.3278	0.9994	50.2670
Review + DL	DeepCoNN	2.8157	1.6780	1.5114	42.8426
	MPCN	2.7226	1.6500	1.5049	42.1549
Rating + Review + DL	NARRE	2.0691	1.4384	1.2014	35.3641
	DAML	2.5119	1.5849	1.4062	39.5853
	DistilBERT	2.3150	1.5215	1.3442	39.4577
	XLNet	2.0125	1.4186	1.1585	34.5967
Our approach	RARV2	0.6189	0.7867	0.5120	25.5996

4.7.2. Ablation analysis

The following table documents the ablation analysis of our model on real datasets collected from the Google Play Store. The experimental results in Table 10 show that the whole model we proposed performs better in all metrics compared to a single module using only ratings or reviews as input, which demonstrates that the combination of review information and rating data as input in our method can further improve the performance in mobile application recommendations.

In addition, we also conduct a detailed ablation study on the review learning module in this paper, as shown in Table 11. RV1-BERT and RV1-RoBERTa are models that use only BERT and RoBERTa as review embeddings, respectively, while RV2 represents a review learning model that combines BERT and RoBERTa into multiple embeddings. It is worth mentioning that we reach a conclusion consistent with the previous argument in this paper that using multiple embeddings does improve the recommendation performance of models using only a single embedding. Meanwhile, the results in this experiment also show that the combination of review and rating learning models can bring us more accurate recommendations than a single learning model.

4.7.3. Comparison with baseline method

In this part, we perform the rating prediction results of our method and baseline methods on the Google Play Store dataset, which are reported in Table 12. From these results, we have several observations as follows.

First, among the models based on rating only, methods considering deep neuron network (e.g. DMF, RA in Table 10) generally perform better than collaborative filtering models (e.g. MF, SVD++, NMF, PMF) which only consider the rating matrix as the input, with significant improvements in all metrics. The reason may be that deep learning can model the interaction of users and items in a non-liner manner, which is a serious limitation of traditional CF-based models. Furthermore, some tricks in deep learning, such as dropout, can be used to avoid overfitting. Hence, it could potentially improve performance.

Second, compared to methods that only use reviews as input (DeepCoNN, MPCN), methods that introduce both ratings and reviews as input (NARRE, DAML, DistilBERT, XLNet) have significant improvements in recommendation performance. Such a result is not surprising,

since the review information is used as a supplement to the rating, so the quality of representation of latent factors can be improved, and the learning accuracy of user preferences and item features can be increased.

Third, although a combined rating and review approach (NARRE, DAML, DistilBERT, XLNet) is superior to an approach that uses only reviews as model inputs, it is inferior to a deep learning approach based on ratings (DMF). The reason may be that although review information is useful in recommendations, its performance depends on how it is utilized [45]. Our results demonstrate that applying deep matrix factorization to rating data processing and converting review text into multi-embedding representations is a valuable design concept. Converting real-world data into BERT-based embeddings consistently leads to better performance in rating prediction work, thanks to the excellent performance of pre-trained BERT models in NLP.

Finally, thanks to deep learning's ability to jointly model and extract deep effective features, and to obtain the rich semantics present in the review text through two high-level language models, our proposed method outperforms all baseline methods in all four metrics and beats the best baseline obtained by obtaining significant improvements in the Google Play Store dataset, as shown in Table 12.

5. Discussion

The findings of the study have several theoretical and practical implications that are worthy of discussion. First, mobile app recommendations differ from other contexts in the type and amount of user data collected on preferences and interests, including ratings, reviews, features, brand, purchase history, and so on. Recommendation systems often prioritize rating and review attributes in practical applications, while also combining them with other attributes based on specific circumstances to improve the accuracy and effectiveness of their recommendations. The proposed multi-embedding model effectively processes ratings and reviews, resulting in improved mobile app recommendations. Researchers can apply our approach to their own contexts to see if it also leads to significant performance.

Second, platforms with insufficient ratings for an app to display to users can utilize the predicted ratings as an alternative to help users make informed decisions. Furthermore, predicted ratings can also help platforms identify apps with a higher likelihood of getting high ratings and promote them accordingly. Third, this study's applicability is not limited to mobile apps; it can be extended to other areas where ratings and reviews are collected. Platforms, such as e-commerce websites, where predicted ratings could be used to guide customers in their purchasing decisions, and restaurant review websites, where predicted ratings could help diners to choose the best places to eat based on their preferences. By implementing using our approach, businesses can improve customer satisfaction, increase revenue, and stay ahead of the competition.

Finally, the present study highlights the capability of generalizability. Our study showcases the ability of scalability in handling large datasets and enhancing the efficiency of recommendation systems, particularly for businesses with significant customer bases or data collection. By integrating multiple techniques to enhance accuracy, developers can explore various methods to discover the optimal combination for their specific application. A practical challenge faced by business platforms is determining which strategies to utilize in order to retain customers. Our study emphasizes the importance of customer feedback and provides a customer-centric approach to recommendation systems. By understanding customer preferences, businesses can make better recommendations and enhance the overall customer experience. More interestingly, the multi-embedding model for review analysis has broad applicability beyond mobile app recommendations; it can be employed in various domains. This finding highlights our method can be utilized for different products or services, including but not limited to movies, books, or restaurants. Furthermore, the multi-embedding model used in recommendation approaches is highly interpretable, facilitating developers' comprehension of how the model makes recommendations. Future algorithmic development should prioritize interpretability to increase the transparency of the recommendation system.

6. Conclusion and future research

In this paper, we have presented a recommendation model based on deep learning by combining ratings and reviews information from mobile application stores as input. In order to enhance the input information of the model, we mainly improve the embedding layer of the model.

We apply the concept of deep neural network into matrix factorization to enhance user and item representations in the rating module, instead of using one-hot encoding. In addition, we use a multi-embedding technique to convert review text into BERT and RoBERTa embeddings, resulting in more diverse information in synthetic vectors. By combining these embeddings with deep learning, we can capture complex relationships between variables. Our experiments on the Google Play Store dataset demonstrate the effectiveness of our approach.

Additionally, we can consider the extension of the proposed model in the following directions in the future. First, we integrate two different language models as review text embedding to implement multi-embedding in this article. One can move towards using other more advanced language models or combining more kinds of effective language models to find better matching methods with more semantically informative text representations, further improving the model's recommendation performance.

Second, with the rapid advancement of deep learning, the possibility of applying different deep learning models or frameworks to the proposed method can also be explored in the future. For example, the Transformer architecture [49], which has become popular in recent years, can be adopted, in which the self-attention mechanism is able to capture the complex relationships between elements in a sequence, which can provide interpretability to the model in addition to better modeling the user preferences.

Furthermore, in this study, only the review text is used in the language model as auxiliary information. It is possible to consider other attributes, such as the types or names of the applications, developer data (developer experience, number of apps developed, etc.), or user data (age, gender, location, etc.), to increase the diversity of the content and make the recommendation system perform more comprehensive analysis in order to capture the characteristics of user preferences more precisely. In addition, our proposed framework could have broader implications if it is proven to be generalizable. This means it could be applied to other datasets, contexts, and methods beyond the current study. Verifying the generalizability of our framework would not only increase its validity and reliability, but also open new avenues for future research. Additionally, it is also important to explore topics such as app download likelihood, app purchase likelihood, in-app purchases, and how to distinguish between users who have not yet discovered an app they may be interested in and those who are not interested in the app, as these have the potential to enhance mobile app recommendations and improve the accuracy and effectiveness of app recommendation systems.

CRediT authorship contribution statement

Yi-Hung Liu: Data curation, Software, Methodology, Validation, Writing – review & editing. **Yen-Liang Chen:** Conceptualization, Methodology, Supervision, Validation, Writing – original draft, Writing – review & editing. **Po-Ya Chang:** Conceptualization, Methodology, Data curation, Writing – original draft.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence

the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] S. Kumar, K. De, P.P. Roy, Movie recommendation system using sentiment analysis from microblogging data, IEEE Trans. Computat. Soc. Syst. 7 (2020) 915–923.
[2] Z. Lin, An empirical investigation of user and system recommendations in e-commerce, Decis. Support. Syst. 68 (2014) 111–124, https://doi.org/10.1016/j.dss.2014.10.003.
[3] J. Ren, J. Long, Z. Xu, Financial news recommendation based on graph embeddings, Decis. Support. Syst. 125 (2019), 113115, https://doi.org/10.1016/j.dss.2019.113115.
[4] Y. Li, R. Wang, G. Nan, D. Li, M. Li, A personalized paper recommendation method considering diverse user preferences, Decis. Support. Syst. 146 (2021), 113546, https://doi.org/10.1016/j.dss.2021.113546.
[5] C. Yin, S. Ding, J. Wang, Mobile marketing recommendation method based on user location feedback, Hum. Cent. Comput. Inf. Sci. 9 (2019). https://doi-org.etd.lib.npust.edu.tw:8443/10.1186/s13673-019-0177-6.
[6] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, in: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009, pp. 452–461.
[7] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Item-based collaborative filtering recommendation algorithms, in: Proceedings of the 10th International Conference on World Wide Web, Association for Computing Machinery, New York, NY, USA, 2001, pp. 285–295.
[8] K. Shi, K. Ali, GetJar mobile application recommendations with very sparse datasets, in: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '12, ACM, Beijing, China, 2012, pp. 204–212.
[9] D. Liu, W. Jiang, Personalized app recommendation based on hierarchical embedding, in: 2018 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation China, 2018, pp. 1323–1328.
[10] H. Yu, X. Xia, X. Zhao, W. Qiu, Combining collaborative filtering and topic modeling for more accurate android mobile app library recommendation, in: Proceedings of the 9th Asia-Pacific Symposium on Internetware, Association for Computing Machinery, New York, NY, USA, 2017, pp. 1–6.
[11] X. Wu, Y. Zhu, A hybrid approach based on collaborative filtering to recommending mobile apps, in: 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), 2016, pp. 8–15.
[12] W. Woerndl, C. Schueller, R. Wojtech, A hybrid recommender system for context-aware recommendations of mobile applications, in: 2007 IEEE 23rd International Conference on Data Engineering Workshop, 2007, pp. 871–878.
[13] E. Costa-Montenegro, A.B. Barragáns-Martínez, M. Rey-López, Which app? A recommender system of applications in markets: Implementation of the service for monitoring users' interaction, Expert Syst. Appl. 39 (2012) 9367–9375.
[14] G. Ling, M. Lyu, I. King, Ratings meet reviews, a combined approach to recommend, in: Proceedings of the 8th ACM Conference on Recommender Systems, 2014, pp. 105–112.
[15] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, Cambridge, MA, USA, 2016.
[16] C. Guo, Y. Xu, X. Hou, N. Dong, J. Xu, Q. Ye, Deep attentive factorization machine for app recommendation service, in: 2019 IEEE International Conference on Web Services (ICWS), 2019, pp. 134–138.
[17] P. Li, Z. Wang, Z. Ren, L. Bing, W. Lam, Neural rating regression with abstractive tips generation for recommendation, in: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017, pp. 345–354.
[18] M.U. Khan, R.M.U. Khalid, S.A. Burhan, M. Nauman, Machine learning Based recommendation system for android apps, in: 13th International Conference on Electronics, Computers and Artificial Intelligence (ECAI), Romania, 2021, pp. 1–4.
[19] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: Proceedings of the 26th International Conference on Neural Information Processing Systems, 2013, pp. 3111–3119.
[20] J. Pennington, R. Socher, C. Manning, GloVe: global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543.
[21] M.E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer, Deep contextualized word representations, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2018, pp. 2227–2237.
[22] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, BERT: pre-training of deep bidirectional transformers for language understanding, in: Proceedings of NAACL-HLT, 2019, pp. 4171–4186.
[23] W. Yin, H. Schütze, Learning word meta-embeddings, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Berlin, Germany, 2016, pp. 1351–1360.
[24] J. Turian, L. Ratinov, Y. Bengio, Word representations: a simple and general method for semi-supervised learning, in: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, USA, 2010, pp. 384–394.
[25] K. Zhang, Z. Lian, J. Li, H. Li, X. Hu, Short text clustering with a deep multi-embedded self-supervised model, in: International Conference on Artificial Neural Networks, 2021, pp. 150–161.
[26] V.Q. Nguyen, T.N. Anh, H.-J. Yang, Real-time event detection using recurrent neural network in social sensors, Int. J. Distribut. Sens. Network. 15 (2019), https://doi.org/10.1177/1550147719856492.
[27] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, J. Gao, Deep learning–based text classification: a comprehensive review, ACM Comput. Surv. 54 (3) (2022). https://dl.acm.org/doi/abs/10.1145/3439726.
[28] M. Gan, Y. Ma, K. Xiao, CDMF: a deep learning model based on convolutional and dense-layer matrix factorization for context-aware recommendation, in: Proceedings of the 52nd Hawaii International Conference on System Sciences, 2019, pp. 1126–1133.
[29] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, Australia, 2017, pp. 173–182.
[30] H. Cao, M. Lin, Mining smartphone data for app usage prediction and recommendations: a survey, Pervasive Mobile Comput. 37 (2017) 1–22.
[31] H.-J. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Australia, 2017, pp. 3203–3209.
[32] C. Jeong, S. Jang, E. Park, S. Choi, A context-aware citation recommendation model with BERT and graph convolutional networks, Scientometrics. 124 (2020) 1907–1922.
[33] Q. Jia, J. Li, Q. Zhang, X. He, J. Zhu, RMBERT: news recommendation via recurrent reasoning memory network over BERT, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021, pp. 1773–1777.
[34] A.L. Karn, R.K. Karna, B.R. Kondamudi, G. Bagale, D.A. Pustokhin, I.V. Pustokhina, S. Sengan, Customer centric hybrid recommendation system for E-commerce applications by integrating hybrid sentiment analysis, Electron. Commer. Res. 23 (2023) 279–314.
[35] R. Cantini, F. Marozzo, G. Bruno, P. Trunfio, Learning sentence-to-hashtags semantic mapping for hashtag recommendation on microblogs, ACM Trans. Knowl. Discov. Data 16 (2022) 1–26.
[36] X. Che, Q. Sun, A feature and deep learning model recommendation system for mobile application, in: IEEE 7th International Conference on Big Data Intelligence and Computing, 2021, pp. 35–40.
[37] L. Zheng, V. Noroozi, P.S. Yu, Joint deep modeling of users and items using reviews for recommendation, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 425–434.
[38] R. Catherine, W. Cohen, TransNets: learning to transform for recommendation, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 288–296.
[39] R.R. Salakhutdinov, A. Mnih, Probabilistic matrix factorization, in: Proceedings of the 20th International Conference on Neural Information Processing Systems, 2007, pp. 1257–1264.
[40] Y. Koren, Factorization meets the neighborhood: a multifaceted collaborative filtering model, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 2008, pp. 426–434.
[41] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (2009) 30–37.
[42] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature. 401 (1999) 788–791.
[43] G. Chen, F. Wang, C. Zhang, Collaborative filtering using orthogonal nonnegative matrix Tri-factorization, in: Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007), 2007, pp. 303–308.
[44] Q. Gu, J. Zhou, C. Ding, Collaborative filtering: weighted nonnegative matrix factorization incorporating user and item graphs, in: Proceedings of the 2010 SIAM International Conference on Data Mining, 2010, pp. 199–210.
[45] C. Chen, M. Zhang, Y. Liu, S. Ma, Neural attentional rating regression with review-level explanations, in: Proceedings of the 2018 World Wide Web Conference, 2018, pp. 1583–1592.
[46] Y. Tay, A. Luu, S. Hui, Multi-pointer co-attention networks for recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 2309–2318.
[47] D. Liu, J. Li, B. Du, J. Chang, R. Gao, DAML: dual attention mutual learning between ratings and reviews for item recommendation, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, USA, 2019, pp. 344–352.
[48] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, in: 3rd International Conference on Learning Representations ICLR, USA, 2015.
[49] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is All you Need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, pp. 6000–6010.

Yi-Hung Liu received his Ph.D. in Information Management from National Central University, Taiwan. Since 2020 he has been with the Department of Computer Science and Information Management, Soochow University, Taipei, Taiwan, where he is currently an Assistant Professor. His current research interests include Machine Learning, Social Networks, Health IT, and Smart Legal Analytics. He has published papers in Information Processing & Management, Journal of Information Science, Information Technology &

People, Journal of the Association for Information Science and Technology and many others.

Yen-Liang Chen is Professor of Information Management at National Central University of Taiwan. He received his Ph.D. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan. His current research interests include data mining, information retrieval, deep learning, and decision models. He has published papers in Decision Support Systems, Operations Research, Decision Sciences, Information & Management, IEEE Transactions on Software Engineering, IEEE Transactions on Knowledge and Data Engineering, IEEE Transactions on Cybernetics, IEEE Transactions on Systems, Man, Cybernetics: Systems, European Journal of Operational Research, Journal of the Association for Information Science and Technology, Information Processing & Management and many others. He is the former editor-in-chief of Journal of Information Management and Journal of e-Business.

Po-Ya Chang is a master student in Department of Information Management, National Central University, Taiwan. Her current research interests include data mining, deep learning, and EC technologies.