Journal of Database Management
Volume 34 • Issue 3

Completion of Parallel app Software User Operation Sequences Based on Temporal Context

Haiyi Liu, Kunming University of Science and Technology, China
Ying Jiang, Kunming University of Science and Technology, China*
Yongquan Chen, Kunming University of Science and Technology, China

ABSTRACT

While mobile application (app) software is becoming increasingly important in people's daily lives, researchers have the limitation of understanding the details of user operations inside the app. With the update of the Android system and application user interface, relying on manually defined user operation event templates or modifying the app source code can no longer meet the needs of fine-grained user operation analysis in multiparallel applications. In this article, a novel method is proposed for effectively analyzing user operations in parallel apps based on the temporal context of user operation sequences. The authors provide a general framework in the Android system to parse out fine-grained user operations. In addition, the authors build a deep learning model with LSTM-TextCNN to complete user operations in parallel app from global temporal context and app temporal context. The authors collected 240k operations of 12 users over a month. Comparative experiments with a baseline show that the proposed method can efficiently and accurately analyze parallel app user operations.

KEYWORDS

app Software, app Temporal Context, Global Temporal Context, LSTM, Parallel User Operation Sequence Completion, TextCNN

INTRODUCTION

In recent years, with the rapid development of the mobile application market, competition among mobile application (app) software has become increasingly fierce. In this context, user-driven software evolution has been the focus of researchers. However, most of the current research focuses on user reviews, software profiles and other display features that embody user experience. Keertipati et al. (2016) found that app reviews contain valuable feedback about what features should be fixed and improved. There is some research aimed at improving user experience by mining user reviews. For example, Guzman & Maalej (2014) used natural language processing techniques to identify fine-grained app features in the reviews. Wu et al. (2021) proposed a novel approach that leverages app software profiles and user reviews to identify key features of a given app.

DOI: 10.4018/JDM.321196 *Corresponding Author

In addition to user reviews and software profiles, Shu et al. (2018) pointed out that user operations such as viewing or downloading provide rich information about users' preferences and usage habits of app software, which have great potentials to advance app recommendations. However, the existed studies have limitations of processing and analyzing huge app software user operation data efficiently and accurately.

In earlier studies, some commercial frameworks were applied to evaluate the usability of mobile devices. For example, Flurry Analytics (2012) provided data and analytics tools that captured how users use their mobile app devices and how the app performed on different phones. Subsequently, Lettner and Holzmann (2012) pointed out that to use these frameworks, developers need to manually add source code for different applications, which means that developers have to spend a lot of time studying the structure of the framework and the target application. He also noted that these statistics reflect how the user generally uses the app software, such as how many times the user uses the app software in a week and how much traffic is consumed, but they cannot reflect the details of user operations of the app software, such as what functions users often use in the app software and what content they focus on and search for.

Balagtas and Hussmann (2009) proposed a method to simplify the analysis of a large number of logs with the help of visualization and third-party tools. Lettner and Holzmann (2012) proposed a method to automatically collect and analyze logs. This method further simplified the work of researchers and can effectively collect usability data. However, McMillan et al. (2015) argued that most logs were limited to recording coarse-grained user operations. For example, these log-based methods were able to answer which application was opened or closed, but specific details of the user operations inside the application were difficult to capture. Users can use WeChat to transfer files, update status, participate in group chats or any other functions that wechat contains.

Krieter and Breiter (2018) used a program based on Python script to convert high-quality screen recording videos into pictures. Then, they analyzed the user operations in the app through image processing technology and defined event picture templates. Based on this method, some studies on user operation data analysis are conducted. For example, Liu et al. (2019) defined app software functions through screen shots and established the connection between app software functions and user operations (A case study of snapchat). However, due to the large differences in the user interfaces of different apps, the manual definition of app software functions based on screen captures is not universal. Similarly, combined with screen recording, timed screen capture, image comparison technologies, Xin (2021) designed and developed the user operation capture function in the app, and selected NetEase Youdao Dictionary app as an application case to observe and analyze the user operation of 13 user samples for a month. Through semi-automated data analysis, they found that the user experience and requirements hidden in the user operation details, thus providing more powerful support for product design, development and maintenance. However, Xin (2021) also pointed out the disadvantages of this method. For example, high quality video recording on the screen will occupy a large amount of mobile storage space, and processing screen shots is time consuming. Xin (2021) tried to reduce the processing time by compressing the images, but he found that this reduced the accuracy of user operation recognition.

Convolutional neural network (CNN) has been used to extract text features and achieve impressive results on the task of sentence classification (Yoon, 2014; Dos et al., 2014). Yoon (2014) proposed a simple one-layer CNN that achieved comparable results across several datasets. Dos (2014) proposed a new deep convolutional neural network to extract character and sentence level information to perform sentiment analysis of short texts. In natural language processing, TextCNN model can be used to extract deeper semantic structure information (Zhang & Wallace, 2015) and performs well on text classification tasks (Zhang & You, 2021; Deng et al., 2020). Long Short-Term Memory (LSTM) is a special recurrent neural network (RNN) model. It is competent and capable of learning dependencies within time series without the necessity for substantial historical time series data (Karim et al., 2019; Greff et al., 2016). Furthermore, LSTM based networks are reported to outperform the state-of-the-

art techniques on user action recognition and prediction tasks (Wu & Tang, 2021; Muhammad et al., 2021). Lastly, joint LSTM TextCNN model has been used to extract richer text features (Wang, 2016; Liu & Guo, 2019; Li et al., 2020). Li et al. (2020) used TextCNN model to extract the local features of sentences, LSTM to extract the global features of sentences, and fused the local features and global features to obtain the embedded representation of sentences. Experiments have shown that joint LSTM TextCNN model is superior to the traditional single model TextCNN and LSTM in capturing text features.

In this paper, the authors propose a general framework to obtain fine-grained user operation content of app software. In addition, to obtain the rich temporal context of the user operations and further improve the processing efficiency, the authors establish a neural network model with LSTM-TextCNN. The model has the ability to process the sequence data of parallel app software user operations. The global temporal context and app temporal context are extracted from the model to complete the user operation sequences in different app. Experimental results show that the proposed method can effectively complete the parallel app user operation sequences, and outperform the baseline in accuracy and processing efficiency.

In summary, this article makes the following contributions: (1) The authors propose a general, unsupervised method for the acquisition of fine-grained app software user operation information on Android. (2) The authors propose a model based on LSTM-TextCNN to complete the information of parallel app software operation sequences from the context of the global operation sequence and the context of the parallel app software operation sequence, which enables the model to process multiple parallel app simultaneously. (3) The experimental results show that the method of modeling global temporal context and app temporal context provides significant performance improvement and accuracy guarantee for the task of completing parallel app software user operation sequences.

The organization of the paper is as follows: In section (Related work) the related research work of app software user operation analysis is introduced; In section (User operation parsing) the capturing and analyzing process of app user operation data is described; In Section (Parsing fine-grained user operations process), the usage scenarios of users in app are first analyzed, and then the completion method of parallel app user operation sequence is provided and introduced in detail. In section (Training data collection) the description of the training data is provided; In section (Extraction of parallel context) the concept and extraction process of parallel app user operation context is presented; In Section (Experiments), the experimental setup and results are provided and analyzed; The last section summarizes the conclusions and contributions of this paper and outlines the future work.

RELATED WORK

The use of screenshots or screen recordings to gain better insights has always been a common topic in human-computer interaction research, but such methods are limited to human-defined events and time-consuming to process screenshots and videos. In 2021, a V2S (video to scenarios) model was proposed (Havranek et al., 2021). It can convert video recordings of mobile application usage into replayable scenarios. Although this model uses a deep learning model to improve the accuracy of user operation restoration, the input videos must adhere to the specific constraints.

Window-based user interface systems generate user interface (UI) events as natural products of user's operation. These events have long been regarded as a potentially fruitful source of information regarding application usage and usability issues (Hilbert & Redmiles, 2000). A framework is presented by Hilbert and Redmiles (2000) to extract usability information from the user interface events. They attempted to establish links between user interface events and other forms of usability data. Inspired by this work, the authors try to extract app user operation based on Android UI framework. Ma et al. (2013) combined the Android application framework to define user operation events and developed a toolkit that can be embedded inside the application. They point out that the toolkit requires minimal modifications to the source program. One of the benefits of this approach highlighted by Ma et al.

(2013) is that it reduces the work of researchers and speeds up data processing. However, the method (Ma et al., 2013) still requires modification to the source code of the app software, and the user operations obtained by this method are still coarse-grained.

The Word2vec model has been widely used to obtain distributed word embeddings with semantic context (Mikolov et al., 2013). Wang et al. (2021) proposed treating the text content in app software documents as sequences and using Word2vec to obtain app software word embeddings. Similarly, Grbovic and Cheng (2018) defined user operations on Airbnb, which were clicks, queries, reservations and purchases, and used an improved Word2vec model to obtain word embeddings with contextual semantics. Based on these studies, the authors hope to establish the semantic context through fine-grained app software user operation content to obtain a better representation of user operations.

In the current research on user operation analysis of app software, both log-based analysis methods and image processing-based analysis methods ignored the temporal context of user operations, especially when multiple app software ran in parallel. Zhang et al. (2021) proposed using an LSTM-based neural network model to obtain the temporal process of user trajectories and thereby strengthen the temporal representation of user trajectories.

Inspired by the above methods, the authors propose a novel method to obtain and complete parallel app software user operation sequences. To make the model universal to all applications of the Android system, the authors no longer rely on any manually defined user operation event templates. Instead, the original app software user operations are captured according to the Android UI framework and application program interface (API) in Android system. In addition, the model can parse the raw information to obtain a fine-grained app software user operation sequence. Finally, the global temporal context and app temporal context are extracted by LSTM-TextCNN model and used to complete the software user operation sequence of the app.

USER OPERATION PARSING

Row Data Collection

The traditional user operation analysis method is to obtain system logs or generating custom logs and screen recording videos by embedding the program in the source code in app software. Krieter and Breiter (2018) defined 30 user operation event picture templates and combined image recognition technology to restore the user operations behavior in the video. Lettner and Holzmann (2012) used the logs output by the system to analyze the user hit rate in different functional activities of the app. In this paper, app software user operation events (AccessibilityEvent) and node information (AccessibilityNodeInfo) were captured based on the user interface framework in real time through the accessibility service interface developed by Google. Table 1 shows the difference between the data obtained by the proposed method and traditional methods that can be used to analyze the user operations of app software. The differences between different data types will be explained from 4

Table 1. Data used by different methods for app software user operation analysis

Data	Real-time	Generality	Granularity	Analysis Difficulty
System Log	√	√	Coarse-grained	Easy
Generative Log	√	×	Coarse-grained	Difficult
Video	×	×	Fine-grained	Difficult
AccessibilityEvent	√	√	Fine-grained	Easy
AccessibilityNodeInfo	√	√	Fine -grained	Easy

perspectives. They are real-time, generality, granularity and analysis difficulty. Real-time refers to whether the data can be obtained in real time. Generality refers to whether the method is common to other Android applications. Granularity refers to the degree to which data explain the app software user operation. Analysis difficulty refers to the difficulty of acquiring and processing data.

As shown in Table 1, the method of obtaining system logs is general and real-time, but previous studies have shown that system logs cannot support the analysis of fine-grained app software user operations. The generated logs improve the granularity of the data, which enriches the content of user operations. However, this method requires researchers to design embedding programs for different app software, which means that it is not general to other app software. In recent years, the method of analyzing video data, as an improvement to the method of generating logs, has the advantage of supporting the analysis of fine-grained user operation content of app software through manually defined user operation templates and image processing technology. However, the analysis process is divided into two steps of saving the video data and analyzing the video data, which results in the data not being analyzed in real time. In addition, researchers need to design different user operation event templates for different app software user interfaces, which greatly increases the difficulty of data processing. Based on these issues, the authors hope to find a general method that can obtain real-time data that supports fine-grained software user operation analysis through a simple process. In the proposed method, the authors can obtain user operation events (AccessbilityEvent) and user interface node information (AccessibilityNodeInfo) without manually defined templates and any source code modifications. Furthermore, AccessbilityEvent and AccessibilityNodeInfo can be used to parse fine-grained app software user operations. This article will discuss how to parse fine-grained app software user operations in Section Parsing fine-grained user operations process.

Fine-Grained Operation in App

Ideally, researchers would like to obtain semantic information of user operations that can be independently interpreted without user involvement. However, the reality is that, due to the different functions and depths of internal activities in different applications, the user operation inside the app cannot be well explained based only on the captured raw user operation information. Therefore, a more detailed description of the user operations inside the Android system application software is necessary. Ma et al. (2013) pointed out that the core content of the Android user interface framework is Activity (interface) and View (component). An Activity is an application component that provides a screen that the user can interact with to accomplish a certain task. A View can be regarded as the components that make up the screen, such as windows, menus, and buttons. However, relying only on activities and views, fine-grained user operations cannot be obtained.

In this paper, the authors further extend the content of user operations inside the app. The expanded user operation content is shown in Table 2. In addition to Activity and View, the authors also pay attention to more details.

Table 2. Fine-grained user operation and its meaning

Feature Name	Meaning
Package	Name of the app software
Activity	Functional interface of the application
Class	Interactive components on the user interface
Type	User operation type
Content	The text contained in the interactable component
Date	Date of user operation

Fine-grained operation content can help us better analyze app software user operations. In previous work, user operations were treated as mutually independent content, which means that the relationship between the content of operations cannot be established. In this section, the interpretability relationship within each user operation is established through fine-grained operation content, which consists of six main components. Table 3 is an example of a fine-grained user operation. An operation of the user in the app can be clearly interpreted from these components. The data in Table 3 indicate that at 10:13:27 am on December 17, 2021, the user edited a message with the content "Good morning" in the text component using the input method in WeChat.

Through the fine-grained operation content, the authors can establish the semantic relationship between user operation content. For example, the authors found that when a user performs a "Click" operation in the "Main Activity" of a music player application, the probability of the user interacting with "Button" with the content "My Favorite" is very high. Analyzing the semantic relationship between the operation contents can help us better explain the user operation in app.

Parsing Fine-Grained User Operations Process

Based on the fine-grained user operation content in an app, the authors propose an automatic parsing method. The Accessibility Services Interface was originally developed to help people with disabilities use smartphones more easily. Based on the accessibility interface, the authors have designed a general service that can capture the raw user operation event information triggered by user operations in real time, and this service is common to all Android apps. It is worth noting that the authors did not enumerate and define all user operation events, which is almost impossible. Instead, the authors proposed a general framework based on the Android interface framework and event distribution mechanism to parse the fine-grained user operation content from the original information. Figure 1 shows the process of automatically parsing fine-grained user operation content by the proposed method.

It is important to note that the authors have especially considered the privacy of users. Users can selectively turn the service on or off when using smartphones. In addition, the service runs in the background and has little impact on users' daily use.

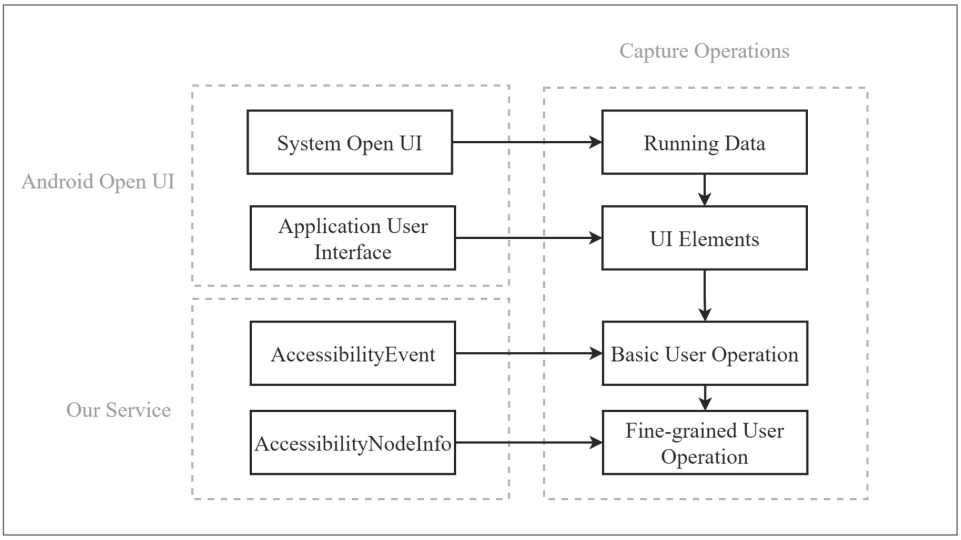
The running data obtained by the Android open UI include the relevant information of the currently running application, such as the name of the application and the size of the memory occupied. Additionally, according to the Android UI framework, the authors can record different interactable elements from the UI, such as Text Field and Button. It should be noted that since the combination of the above two types of information is not considered, the previous method cannot support obtaining fine-grained user operation content of app software. Therefore, additional data support is necessary.

The raw user operation information captured by the service is stored in AccessibilityEvent and AccessibilityNodeInfo. Raw information contains much information that the authors do not need and cannot interpret. According to the running data and user interface elements, the authors can filter out the invalid information in AccessibilityEvent so that the authors can obtain valid user

Table 3. An example of fine-grained user operation in app

Feature Name	Data
Package	com.tencent.mm
Activity	android.inputmethodservice.SoftInputWindow
Class	android.widget.EditText
Type	TextSelection
Content	'Good Morning'
Date	2021-12-17 10:13:27

Figure 1. The process of parsing fine-grained user operations



operations, such as Package, Activity, Class, Time, and Content. However, different app security settings and user interface designs can cause some data to be empty. He et al. (2021) emphasized the relationship between user operations and mobile application UI in their research and pointed out that the semantics of various attributes of UI components can be represented by a tree-based view hierarchy. To further obtain the fine-grained user operation content of the app software, the authors further parse the AccessibilityNodeInfo. The authors found that AccessibilityNodeInfo contains a similar tree structure to store the relationship of components in the view, which is a special layout of Android views. Based on this structure, the authors can obtain content descriptions, resource IDs, component classes, and component text fields, which are supplementary content for fine-grained app software user operations.

COMPLETION OF PARALLEL APP USER OPERATION SEQUENCE

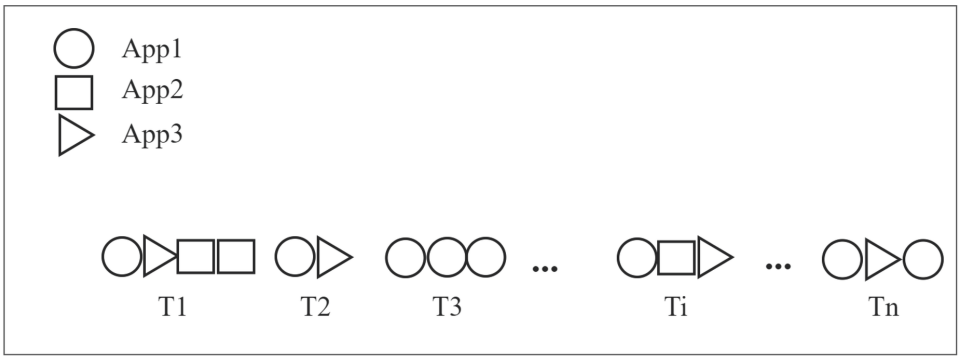
In Section User Operation Parsing, the fine-grained app software user operation content has been obtained. However, it is not enough to rely only on these contents to analyze the user operations of app software, the real usage scenarios have to be considered.

App Software Usage Scenario

Figure 2 shows a more realistic distribution of user operation sequences in a parallel app. A user may interact with multiple app software within a period of time. The authors collected the operation behaviors of 12 users over a month, and obtained 240,000 app user operations. Through the collected data, the authors found that users generally interacted with more than two apps every three minutes. In the example shown in Figure 2, "T" refers to a period of time, and "Ti" refers to a certain time period. The user interacts with app1, app2, and app3 successively at T1, while only interacts with app1 at T3.

Krieter and Breiter (2018) found that even though the definition of user operation events was manually defined in some applications, the model still produced some erroneous log information when analyzing the image when the application was switched and closed. Lettner and Holzmann (2012) pointed out that most of the current methods or frameworks for automatically obtaining user operation

Figure 2. An example of user operation sequences in multiple parallel running apps over a period of time



data in applications do not consider contextual information, which may directly affect the usability of the method. Therefore, the authors propose temporal contextual features of user operation sequences.

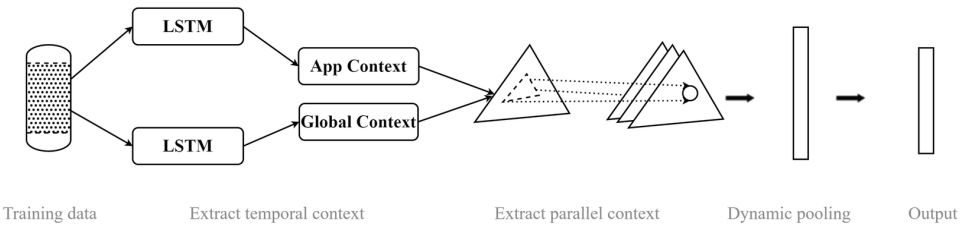
However, simply relying on a context is not enough. According to the user interface design of the Android system, the system only processes and responds to the top application, which may result in the lack of user operation data at different time periods for some parallel applications at the bottom layer. However, the missing part of the data should also be considered part of the sequence of user operations at the same time. For example, when a user uses a dictionary application to look up words, he accepts a video chat request initiated by a WeChat friend, and the dictionary application also runs in the background while the user is chatting in WeChat. However, while WeChat occupies the screen, the data of dictionary application will not be recorded. These missing data can cause errors in the analysis of the user operation of the app software, especially in the methods based on log and video analysis.

Parallel App Software User Operation Sequence Completion Process

To complete these missing data, the authors propose the method shown in Figure 3. The authors first obtain training data through preprocessing and pretraining. Then, the authors further propose that the app software user operation sequence should be completed from two temporal contexts, which are the global temporal context and the app temporal context. The authors use an LSTM-based deep learning model to extract these two temporal contexts and obtain parallel temporal context through the TextCNN model. The parallel temporal context will be used to complete the parallel app software user operation sequence.

This article has introduced the global distribution of parallel software user operation sequences. To fully consider the real parallel app software user operation scenarios, the authors further propose that the temporal context of user operations in each app software should also be considered. For example,

Figure 3. Parallel app software user operation sequence completion process



when a user gives comments on an update posted by a friend in WeChat, it always goes through a series of contextual operations (i.e., open WeChat, view the update, post a comment), regardless of whether other parallel apps occupy the process user interface. Figure 4 shows the sequence of user operations for different apps running in parallel over a period of time. The authors can see that users operate in different parallel app software at different times. For example, in Figure 4, at T3, the user performs three operations on app1, while app2 and app3 are still running in the background. To complete the missing user operations on app2 and app3 at T3, the authors need to consider not only the global temporal context of user operations but also the temporal context of each app.

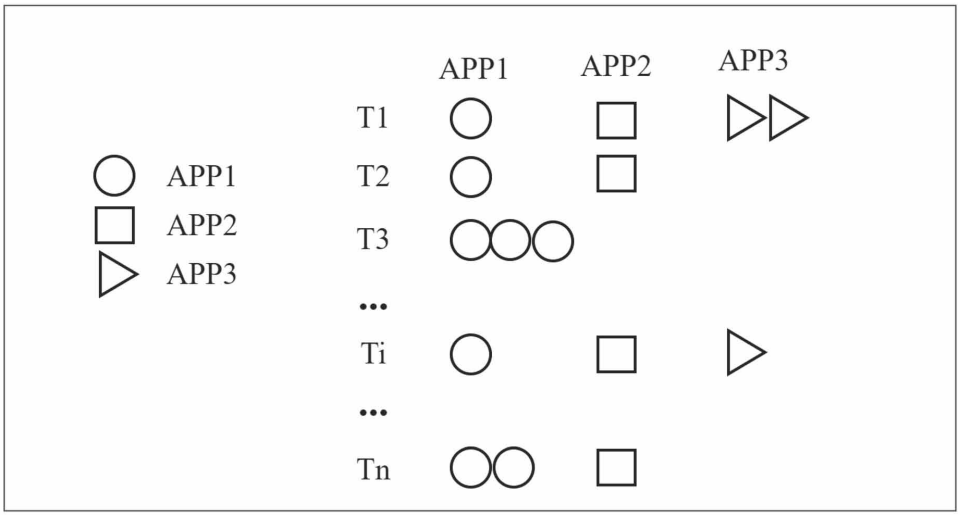
Considering the temporal context of app is a critical step for the proposed method. The authors found that regardless of whether a system log, screen recording or the proposed method is used, due to the particularity of the Android user interface and event response mechanism, some app user operation data will always be lost. When completing this part of the data, in addition to considering the global temporal context of user operations, the temporal context of user operations in each app is particularly important. However, previous work does not take contextual information into account, which also makes it difficult for previous methods to analyze the user operation sequences in multiple parallel apps. Furthermore, by considering the app temporal context, the proposed method can process parallel app user operation sequences simultaneously, which greatly improves the efficiency of the model.

Training Data Collection

Through the method implemented in section User operation parsing, the authors have obtained the fine-grained content of user operations. To consider both the global temporal context and the app temporal context, the authors need two copies of the data, one for global temporal context extraction and another for app temporal context extraction.

First, the authors use the open source word embedding corpora (Song et al. 2018) to transform user operation data into word embedding representation. Then, a pre-trained model based on Word2Vec is used to establish the relationship between user operations. The authors choose the skip-gram model for pretraining. It is worth noting that fine-grained user operation content can help us obtain better user operation representation. Next, the authors divide the user operation data by hours to obtain training data. In particular, as shown in Figure 4, the data copies used to extract the app temporal context will be grouped by app software.

Figure 4. An example of considering the user operations sequence from the perspective of app software

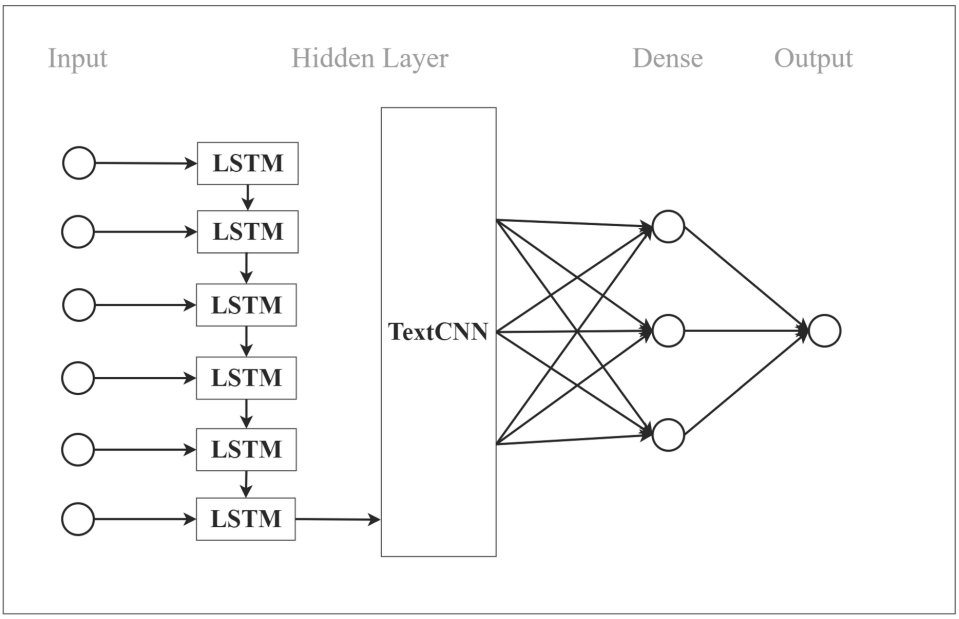


Extraction of Parallel Context

Studies have shown that significant events with gaps and delays in time series can be analyzed and predicted using the LSTM (long short-term memory) model. LSTM models have achieved superior performance in the task of extracting contextual semantics from temporal data (Li et al., 2015; Greff et al., 2016). Therefore, the authors use an LSTM-based neural network model to extract the global temporal context and app temporal context of user operation sequences from parallel app software user operation sequences. Since user operation sequences can be viewed as simple short text sequences, the authors do not discuss more complex variant LSTM models in this paper. Compared with a single app running environment, one of the characteristics of the data of parallel app software user operation is that users may interact with multiple apps to achieve their goals. Therefore, the weights of the two temporal contexts cannot simply be designed manually but should be based on data from user operations. Studies have proven that TextCNN has a strong ability to extract text shallow features (Zhang & Wallace, 2015), especially in the processing of short texts, which means that TextCNN is suitable for the completion of parallel app software user operation sequences. Compared to manually setting fixed weights, TextCNN can help us better extract more accurate semantic information from the data. Therefore, the authors propose a TextCNN-based neural network model for extracting parallel temporal context from global temporal context and app temporal context. In summary, as shown in Figure 5, the authors propose a neural network model based on LSTM-TextCNN to extract parallel temporal context and complete parallel app software operation sequences.

As shown in Figure 5, since the authors need to extract the global temporal context and the app temporal context through the LSTM layer, the authors use the two copies of the training data for extracting different temporal contexts as the input of the model. During training, each user operation sequence will be regarded as a short text of length 6. In the LSTM layer, the temporal context of the user operation sequences is stored in the last latent vector. Through the LSTM layer, the authors can obtain the global temporal context and the app temporal context. It is worth noting that in the training process, to avoid training too many neural network models, the authors did not set neural network models for all of the parallel apps to extract the app temporal contexts but concatenated different app

Figure 5. Parallel temporal context extraction based on the LSTM-TextCNN model



software operation sequences. In this way, the authors only need to train a neural network model to extract the app temporal context, which can further improve the efficiency. However, some errors will inevitably occur in this way. The authors try to reduce these errors by varying the training stride in the LSTM layers. The authors found that setting the stride of the LSTM layer to 3 had the lowest error. Then, the global temporal context and APP temporal context obtained through the LSTM layer will be regarded as the input of the TextCNN layer. The authors set the convolution window size to 2 and the convolution kernel to 128. Finally, through the fully connected layer, the authors can obtain the completed parallel app software user operation sequences.

EXPERIMENTS

Krieter and Breiter (2018) proposed a method called "Mobile screen recordings to log file (MSRTLF). In recent years, MSRTLF has proven to be a more effective way to analyze app user operation. They use screen recording software to record the user operations and process the video frame by frame into pictures through Python scripts. Then, these pictures are matched with predefined user operation event picture templates through image processing technology. However, MSRTLF also has some problems, which will be discussed in detail in the analysis below. In this section, the authors conduct comparative experiments from multiple perspectives to illustrate the superiority of the proposed method on the task of completing parallel app software user operation sequences.

Accuracy on User Operation Sequence Completion

First, the authors designed a method that can capture and parse fine-grained user operations of app software in real time, which has been introduced in Section User Operation Parsing. The authors collected the operation behaviors of 12 users over a month, and obtained 240,000 app user operations. Since the video data occupy a large amount of mobile phone memory, the authors recorded 20 times during this period, and the duration of each recording was 30 seconds to 10 minutes. The effectiveness of the method is evaluated by manually comparing the average precision of processing the corresponding user operation data over this time period.

Table 4 shows the accuracy of the proposed method and the MSRTLF method on the task of completing user operation sequences with different granularities. In the MSRTLF method, the log format set by the researchers has four properties consistent with the proposed method, namely, "Package", "Time", "Class" and "Type". The authors use "G" to denote the granularity of user operation sequences. For example, "GTC" represents a sequence of user operations considering "time" and "category". The authors set up two scenarios, a single app running scenario and a multiparallel app running scenario. Experiments show that the proposed method leads to huge accuracy improvements in both cases. Furthermore, the proposed method is more stable on the task of completing fine-grained user operation sequences.

Table 4. Average accuracy comparison on user operation sequence completion at different granularities

app(s)	Models	Average Accuracy (%)			
		GT	GTP	GTPO	GTPOC
With Single Running app	MSRTLF	45.9	34.9	30.5	28.5
	The proposed Method	96.2	95.8	94.6	92.4
With Parallel Running apps	MSRTLF	24.8	18.5	14.5	12.8
	The proposed Method	92.7	91.9	90.4	90.2

Note: "GT" represents a user operation sequence including the operation time; "GTP" represents a user operation sequence including the operation time and the name of the package; "GTPO" represents a user operation sequence including operation time, the name of the package and operation object; "GTPOC" represents a user operation sequence that includes operation time, the name of the package, operation object and the text of operation content.

Discussion

In fact, the experimental results of the MSRTLF method are consistent with the work in (Krieter & Breiter, 2018). They pointed out that although the method based on image processing technology can effectively analyze the user operations data in the app, this method has some limitations. First, due to the limitation of relying on manually defined event templates, the MSRTLF method is often only effective for user operation data analysis in a single app. When faced with more complex scenarios, when the user interacts with multiple applications at the same time, the researchers need to add or redefine the user operation event templates, which will greatly affect the efficiency of this method. Conversely, the proposed method does not require researchers to define user operation templates individually, which greatly simplifies the researchers' work. Second, the MSRTLF method restores the user operation after it has ended, which will inevitably cause data inconsistency. For example, when MSRTLF parses video into pictures, it will miss some millisecond-level user operations. The proposed method can capture the user operations in the app in real time, which fundamentally solves this problem. Finally, Krieter and Breiter (2018) pointed out that the MSRTLF method encounters difficulties as the Android system and application user interface are updated. Unless relying on manual updating of more detailed user operation event templates, the accuracy of the method will drop significantly when analyzing user operations in other updated apps. It is worth noting that the proposed method provides a general framework on the Android system, which guarantees accuracy and stability for researchers to analyze a wide range of app software user operations.

Performance on Processing Efficiency

Krieter and Breiter (2018) found that the number of event templates and the size of the original video are two important factors that affect the processing time. In addition, factors that affect the number of event templates are the number of applications running in parallel and the granularity of user operation. For example, if the user interacts with three app software in a period of time, the researchers must design user operation event templates for different app software when using the MSRTLF method. To compare the efficiency of the methods, as shown in Table 5, the authors compare the time consumed by MSRTLF and the proposed method in analyzing videos of different lengths and corresponding user operation data. Similarly, the authors also set up two scenarios: a single app running and multiple apps running in parallel. The authors set three different levels to mark the length of the video: short (30 seconds), normal (180 seconds), and long (600 seconds). The results show that the proposed method has a large improvement in efficiency compared to the MSRTL method, especially when multiple applications run in parallel.

Discussion

Krieter and Breiter (2018) tried to improve the processing efficiency of the MSRTLF method by reducing the video quality, but this would greatly reduce the recognition accuracy of user operation. Compared with MSRTLF, the proposed method uses a neural network model to process text data,

Table 5. Time consumption comparison of processing different sizes of data

app(s)	Models	Time Consumption(sec)		
		Short	Normal	Long
With Single Running app	MSRTLF	105	910	2946
	The proposed Method	10	12	13
With Parallel Running app	MSRTLF	196	1152	4200
	The proposed Method	11	12	13

which is more accurate and efficient. In addition, since there is no need to design user operation templates for each app software, the proposed method can efficiently process parallel app software user operations. Experiments show that compared with the MSRTLF method, the proposed method can quickly process large-scale parallel software user operation data. In particular, pretraining with Word2vec can make the model converge quickly, which can save approximately 0.3 seconds per 1000 user operation data.

Contribution of Temporal Context

The process of using app software for users is often more complicated. During a period of time, users may interact with multiple parallel app software packages in different forms, which makes it difficult for traditional methods to analyze parallel app user operations. Based on this fact, using multidimensional temporal context becomes especially important when analyzing parallel app software user operations.

Based on the data, the authors list four typical scenarios of user operations: frequent, sparse, concentrated, and scattered. Frequent and sparse are used to describe the quantitative feature, which is the number of user operations over a period of time. Concentrated and scatted are used to describe the distribution of user operations in different app software. The authors design an automatic tagging method. For example, when the number of user operations in 1 hour is greater than the average number of user operations, they are considered frequent; otherwise, they are considered sparse. The authors extracted 3 sets of data from 240,000 user operation data, each with a number of 30,000, and calculated the average number of user operations per hour, which is 106. Second, for the distribution characteristic, the average highest proportion of parallel app software within an hour is 50.66%. Similarly, when the proportion of user operations of a single app software within 1 hour is higher than the average highest proportion, the user operations are considered concentrated; otherwise, they are considered scattered.

In this subsection, to further illustrate the effectiveness of the multidimensional temporal context, the performance of the four contextual features in different user operation scenarios is shown in Table 6. The authors designed 4 user operation scenarios: ‘FC’, ‘FS’, ‘SC’, and ‘SS’.

Table 6. Completion of parallel software user operations based on different contexts and user operation scenarios

Context	User Operation Scenarios			
	FC	FS	SC	SS
1 Hour				
Only Semantical	59.1%	57.3%	60.0%	55.3%
Global	70.4%	68.1%	67.8%	59.6%
app	79.8%	77.5%	82.5%	54.7%
Parallel	86.4%	85.4%	**85.9%**	79.2%
2 Hours				
Only Semantical	59.2%	58.4%	74.7%	59.6%
Global	71.5%	71.5%	70.3%	63.7%
app	80.3%	77.8%	82.7%	57.2%
Parallel	86.1%	84.9%	**86.5%**	82.6%

Note: ‘FC’ stands for frequent, concentrated user operations. ‘FS’ stands for frequent, scatted user operations. ‘SC’ stands for sparse, concentrated user operations, ‘SS’ stands for sparse, scattered user operations.

Discussion

As shown in Table 6, first, the authors found that the overall effect of user operation completion for two hours is better than that within one hour. Based on data from parallel app software user operations, the authors find that it takes longer for users to interact with a parallel app than with a single app. Second, concentrated user operations are always easier to complete because they are closer to user operation in a single app, especially for sparse, concentrated user operations. The method that only considers the semantic context of user operations ignores the temporal feature of user operations, which leads to a generally low completion accuracy rate. After taking the global temporal context feature into account, the completion accuracy rate is improved by approximately 10%, especially when the user operates frequently. However, the global temporal context is not applicable for completing sparse and scattered user operations. In addition, the app temporal context is more suitable for concentrated user operation completion. Compared with the global temporal context, the app temporal context focuses on the user operation sequence of each app. However, for the completion of sparse user operations, the app temporal context does not bring an improvement in accuracy; in contrast, it drops by approximately 5% when user operations are sparse and scattered. The authors found that when user operations are sparse and scattered, users may open the app software abnormally or use the functions of the app software incorrectly, which reduces the usability of the app temporal context, because the app temporal context focuses more on the use of independent app software functions. Based on the above analysis, the authors propose a method that combines global temporal context and app temporal context for the completion of parallel app software user operations. The experimental results show that the proposed method can achieve high accuracy in different parallel app software user operation scenarios, which indicates that the parallel temporal context is suitable for the completion of parallel app software user operations.

CONCLUSION

In this paper, the authors build a general, powerful framework for analyzing user operations in apps. In analysis, the proposed method supports obtaining fine-grained sequence data of user operations without relying on any manual definition and modification of application source code. In addition, the authors propose that user operations should be pretrained as short texts to obtain better training data, which can improve the training speed and accuracy of the model. Furthermore, the authors divide the temporal context of the user operation sequence into the global temporal context and the app temporal context. The experimental results show that the global temporal context and the app temporal context of the user operation sequence can be effectively extracted through the LSTM model. In addition, with the TextCNN model, the authors can effectively combine the features of these two temporal contexts. According to the experimental results, compared with the baseline, the proposed method has greatly improved in terms of accuracy, stability and efficiency. In summary, the proposed method is general and efficient, which solves some difficult problems of analyzing user operations in the multi-parallel app environment and simplifies the work of researchers. The proposed method provides support for the large-scale analysis of app user operations. Therefore, the analysis of app software user operations no longer limited to few app software case studies. Based on the proposed method, future researchers can further explore the value of app user operations, such as analyzing the relationship between normal user operations and abnormal operations. It should be noted that the authors collected the operation behaviors of 12 users over a month, and obtained 240,000 app user operations which contains 82 app software most commonly used by users and more than 500 app software functions. These data are representative and can generally reflect the

details of user operation. However, the number of user samples is limited, which means that there are few user operations have not been considered and analyzed. Finally, the completed app software operation sequence is fine-grained and contains rich context information. The author is exploring and establishing the relationship between app user operations and other data reflecting usability.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China under Grants No. 62162038, 61462049, 60703116 and 61063006, the National Key Research and Development Program of China (2018YFB1003904), the Key Project of Yunnan applied Basic Research (2017FA033), and the Open Foundation of Yunnan Key Laboratory of Computer Technology application (2020101).

REFERENCES

Balagtas-Fernandez, F., & Hussmann, H. (2009, November). A methodology and framework to simplify usability analysis of mobile applications. *In 2009 IEEE/ACM international conference on automated software engineering* (pp. 520-524). IEEE.

Deng, D., Zhou, Y., Chi, J., & Li, J. (2020). Review of short text classification technology research. *Software*, (02), 141–144.

Dos Santos, C., & Gatti, M. (2014, August). Deep convolutional neural networks for sentiment analysis of short texts. *In Proceedings of COLING 2014, the 25th international conference on computational linguistics: technical papers* (pp. 69-78). IEEE.

Flurry Analytics. (2012). *Homepage.* Flurry Analytics. https://www.flurry.com

Grbovic, M., & Cheng, H. (2018, July). Real-time personalization using embeddings for search ranking at airbnb. *In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 311-320). ACM. doi:10.1145/3219819.3219885

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2016). LSTM: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(10), 2222–2232. doi:10.1109/TNNLS.2016.2582924 PMID:27411231

Guzman, E., & Maalej, W. (2014, August). How do users like this feature? a fine- grained sentiment analysis of app reviews. In *2014 IEEE 22nd international requirements engineering conference (RE)* (pp. 153-162). IEEE.

Havranek, M., Bernal-Cárdenas, C., Cooper, N., Chaparro, O., Poshyvanyk, D., & Moran, K. (2021, May). V2S: a tool for translating video recordings of mobile app usages into replayable scenarios. *In 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (pp. 65-68). IEEE.

He, Z., Sunkara, S., Zang, X., Xu, Y., Liu, L., Wichers, N., Schubiner, G., Lee, R., & Chen, J. (2021, May). Actionbert: Leveraging user actions for semantic understanding of user interfaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(7), 5931–5938. doi:10.1609/aaai.v35i7.16741

Hilbert, D. M., & Redmiles, D. F. (2000). Extracting usability information from user interface events. [CSUR]. *ACM Computing Surveys*, *32*(4), 384–421. doi:10.1145/371578.371593

Karim, F., Majumdar, S., & Darabi, H. (2019). Insights into LSTM fully convolutional networks for time series classification. *IEEE Access : Practical Innovations, Open Solutions*, *7*, 67718–67725. doi:10.1109/ACCESS.2019.2916828

Keertipati, S., Savarimuthu, B. T. R., & Licorish, S. A. (2016, June). approaches for prioritizing feature improvements extracted from app reviews. In *Proceedings of the 20th international conference on evaluation and assessment in software engineering* (pp. 1-6). IEEE. doi:10.1145/2915970.2916003

Krieter, P., & Breiter, A. (2018, September). Analyzing mobile application usage: generating log files from mobile screen recordings. *In Proceedings of the 20th international conference on human-computer interaction with mobile devices and services* (pp. 1-10). IEEE. doi:10.1145/3229434.3229450

Lettner, F., & Holzmann, C. (2012, December). Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. *In Proceedings of the 10th International Conference on Advances in Mobile Computing & Multimedia* (pp. 118-127). IEEE. doi:10.1145/2428955.2428983

Li, J., Chen, X., Hovy, E., & Jurafsky, D. (2015). Visualizing and understanding neural models in nlp. *arXiv preprint arXiv:1506.01066.*

Li, Z., Geng, C., & Peng, S. (2020). Short text classification based on LSTM-TextCNN joint model. *Journal of Xi'an University of Technology*, (03), 299–304. doi:10.16185/j.jxatu.edu.cn

Liu, G., & Guo, J. (2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing*, *337*, 325–338. doi:10.1016/j.neucom.2019.01.078

Liu, Y., Shi, X., Pierce, L., & Ren, X. (2019, July). Characterizing and forecasting user engagement with in-app action graph: A case study of snapchat. *In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 2023-2031). ACM. doi:10.1145/3292500.3330750

Ma, X., Yan, B., Chen, G., Zhang, C., Huang, K., Drury, J., & Wang, L. (2013). Design and implementation of a toolkit for usability testing of mobile apps. *Mobile Networks and applications*, *18*(1), 81–97. doi:10.1007/s11036-012-0421-z

McMillan, D., McGregor, M., & Brown, B. (2015, August). From in the wild to in vivo: Video Analysis of Mobile Device Use. *In Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services* (pp. 494-503). IEEE. doi:10.1145/2785830.2785883

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Muhammad, K., Ullah, A., Imran, A. S., Sajjad, M., Kiran, M. S., Sannino, G., & de Albuquerque, V. H. C. (2021). Human action recognition using attention based LSTM network with dilated CNN features. *Future Generation Computer Systems*, *125*, 820–830. doi:10.1016/j.future.2021.06.045

Shu, K., Wang, S., Liu, H., Tang, J., Chang, Y., & Luo, P. (2018, August). Exploiting user actions for app recommendations. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (pp. 139-142). IEEE.

Song, Y., Shi, S., Li, J., & Zhang, H. (2018, June). Directional skip-gram: Explicitly distinguishing left and right context for word embeddings. *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,* Volume 2 *(Short Papers)* (pp. 175-180). doi:10.18653/v1/N18-2028

Wang, J., Yu, L. C., Lai, K. R., & Zhang, X. (2016, August). Dimensional sentiment analysis using a regional CNN-LSTM model. *In Proceedings of the 54th annual meeting of the association for computational linguistics (*volume 2*: Short papers)* (pp. 225-230). ACM. doi:10.18653/v1/P16-2037

Wang, M., Peng, J., & Liu, J. (2021). A classification method of Android applications based on text features. *Modern Computer*, (15), 89–93.

Wei, Z., Yang, L., Ji, Z., & Wang, J. (2021). A User Trajectory Recognition Model Integrating Spatiotemporal Behavior and Social Relationships. *Chinese Journal of Computers*, *44*(11), 16.

Wu, H., Deng, W., Niu, X., & Nie, C. (2021, May). Identifying key features from app user reviews. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 922-932). IEEE. doi:10.1109/ICSE43902.2021.00088

Wu, M., & Tang, Z. (2021, June). Research on User Action Recognition Method Based on parallel CNN-BiLSTM neural network. *In 2021 IEEE 4th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)* (Vol. 4, pp. 2003-2009). IEEE.

Xin, X. (2020). *Research on Capture and Analysis of User Behaviors in Mobie applications* [Master dissertation, Dalian Maritime University]. https://kns.cnki.net/KCMS/detail/detail.aspx?dbname=CMFD202101&filename=1020078242.nh

Yoon, K. (2014). Convolutional Neural Networks for Sentence Classification [OL]. *arXiv Preprint*.

Zhang, T., & You, F. (2021). Research on short text classification based on textcnn. []. IOP Publishing.]. *Journal of Physics: Conference Series*, *1757*(1), 012092. doi:10.1088/1742-6596/1757/1/012092

Zhang, Y., & Wallace, B. (2015). A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. arXiv preprint arXiv:1510.03820.

Haiyi Liu, born in 1997, Postgraduate. His main research interests include intelligent software engineering and software quality assurance and testing.

Ying Jiang, born in 1974, PhD, Professor, PhD supervisor. Her main research interests include software quality assurance and testing, cloud computing, big data analysis and intelligent software engineering.

Yongquan Chen, born in 1998, Postgraduate. His main research interests include software engineering.