Analysis Method of App Software User Experience Based on Multisource Information Fusion

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

ABSTRACT

With the rapid development and popularization of intelligent terminals, app software has also developed rapidly. The research and practical value of mining user experience (UX) of app software form interaction information are becoming increasingly prominent. The interactive information of app software is multisource homogeneous and heterogeneous. In order to obtain more accurate and more comprehensive app software UX results, the fused multisource information should be analyzed. In this paper, the app software UX analysis method based on multisource information fusion is proposed. First, feature engineering is carried out to extract the features. Then, the feature combination tree is constructed after feature correlation mining. Finally, the multisource app software interactive data are fused, and the result is further analyzed to obtain the information of app software UX. The experiments clearly show that the method can effectively fuse multisource app software interaction data and help to comprehensively mine the app software UX embodied in the data.

KEYWORDS

App Software, Interactive Information of App Software, Multisource Information Fusion, User Experience

INTRODUCTION

With the growing popularity of intelligent terminals in recent years, software applications have developed rapidly. Numerous App software packages run on massive numbers of intelligent terminals (Ahmed, 2019; Moreira et al., 2020). However, App software has the characteristics of a short development cycle, fast iterative updating and fierce competition, and competition in the App software industry is becoming increasingly fierce (Zhang et al., 2020). It is necessary to develop App software that is more in line with users’ habits, interests, and preferences (Carlos et al., 2020). Therefore, how to mine user experience and the evolution of App software development driven by user requirements have attracted increasing attention (Sartori et al., 2022; Youm & Kim, 2022; Leem & Eum, 2021).
User experience (UX) is described as a personal, subjective psychological feeling established by users in the process of using a product or service (Jeff Sauro & James R. Lewis, 2014). Olsson (2013) pointed out that UX is associated with vague, dynamic, and hard-to-quantify concepts, such as ‘experience’, ‘perception’, ‘pleasure’, and ‘emotions’. Additionally, UX depends on very dissimilar constructs that all encompass an endless area of research: the users themselves, the technical systems and functionalities, products or services, and the contextual factors such as social setting, cultural layers, and the users’ other activities. Haaksma et al. (2018) pointed out that UX is a more recent and broader concept than usability. Users pay much attention to the way that the product works: Does it work as intended, as expected, without flaws or obstructions? They also found that UX seems to be influenced by the specific context in which the product is normally used. Many papers on UX considered that the content of UX involves three aspects: user, product, and interaction environment (Tang et al., 2022; Ding et al., 2014; Berni & Borgianni, 2021).

Many studies hold the view that the UX of App software is also related to the above aspects. For example, Hu and Jiang (2019) argued that there were a lot of valuable software usage feedbacks in user comments, which could reflect the user’s feeling in the process of using the software and the software characteristics concerned by the user. Jiang and Hu (2019) emphasized that software user comments were important information sources that helped users and software developers to evaluate software quality-in-use. Cao and Lin (2017) suggested that while apps were used, data logs were typically generated, and the ambience context was recorded, forming a rich data source of smartphone users’ behaviors that contained the user’s behavior patterns, usage habits, interests and other user personal behavior information and UX. Furthermore, the environment in which the App runs is also a key factor influencing the UX of App software. Therefore, the analysis of App software UX needs to integrate the multisource interactive information of App software to obtain more accurate and reliable results of App software UX.

The App software users have a wide range of communities, and a large number of user comments, App running process data, App running time data and other interactive information of App software are generated at any time. However, this information is multisource homogeneous and heterogeneous information obtained in different spaces or at different times, which together embodies the UX of App software. Therefore, to obtain homogeneous and complementary UX information of App software, the method of multisource information fusion is needed to fuse homogeneous parts in multisource and heterogeneous information (Huang et al., 2020). Multisource information fusion is “the technology of combining and merging information or data from multiple sources to form a unified result” (Chen et al., 2013). For Bostrm et al. (2007), information fusion is the study of efficient methods for automatically or semiautomatically transforming information from different sources and different points in time into a representation that provides effective support for human or automated decision making. Wei (2022) pointed out that multisource information fusion is a multilevel, multistep process, with the characteristics of availability, compatibility, and aggregation of multisource data, to regularly improve status and identity analysis. This paper proposes a methodology for analyzing UX of App software based on multisource information fusion. The methodology uses the technology of multisource information fusion to fuse multisource heterogeneous information reflecting the UX of App software and analyzes the UX of App software on the fused data to determine the embodied UX of App software, which is helpful for developers to maintain and improve App software in a targeted manner. For example, in the fusion results obtained by this method, a certain class of App software UX information mainly related to the usability and functionality of the App interface. If the UX information is not good, the developers need to pay attention to this type of Apps, and update the interface and functions to improve the UX of the related Apps.

The rest of the paper is organized as follow. The first section (Related Works) compares this research with related works and describes the contributions of this method. The next section (The Definition of App Software User Experience) elaborates the definition of App software user experience and its influencing factors. The next section (Method of App Software UX) presents the principle and
steps of App software user experience analysis method. The next section (Experiments and Analysis) depicts the experimental design conducted in this research and presents the results of the experiment. The final section (Conclusions) summarizes the method in this paper, describing the limits of the study and possible future directions.

RELATED WORKS

A number of models and theoretical approaches have been proposed by researchers to help analyze UX. Some scholars mined valuable software usage feedback and UX based on user comments. For example, Hu and Jiang (2019) aimed at different types of usage feedback for App software and proposed the extracting rules of evaluation object and evaluation opinion. Moreover, a semisupervised method based on review seeds was designed to mine usage feedback in user comments, and finally, the corresponding software usage feedback from App software user comments was mined. Maalej and Nabil (2015) introduced several techniques, such as text classification, natural language and sentiment analysis, to classify App reviews into four types: bug reports, feature requests, user experiences, and ratings. Lu et al. (2014) designed a web system that obtains users’ overall evaluation of the software, as well as the positive and negative rating of software on the granularity of features, through data mining methods such as feature extraction, emotional word extraction, polarity analysis of emotional words, and feature clustering. Guzman et al. (2015) applied multiple classifiers to classify user comments in more detail. They classified user comments into 7 categories: bug report, feature strength, feature shortcoming, user request, praise, complaint and usage scenario. Different categories of user comments could feed back users’ satisfaction with the software or problems existing in the software. Hussain et al. (2022) proposed a three-step UX quantification model from online reviews to understand customer satisfaction using the effect-based Kano model. They extracted the UX dimensions and mapped them on the customer satisfaction model to understand the user perspective about the systems, products, and services.

To analyze user behavior in a big data context, Chen (2016) designed a user behavior analysis system on the Hadoop big data platform. The system collected the daily log information of users through the SDK of the input method to obtain the user’s behavior data and carried out routine data statistical calculations on the user’s behavior data, including the user’s daily add, the user’s daily activeness, the analysis of user retention, channel analysis, custom events, and error analysis. Moreover, the paper also analyzed the trend of emotional color through the user’s input data. Ren (2020) proposed the mobile end-user path analysis model of the App. The data in the App page buried points were used to collect user data, session data, and page access data. By combining the depth traversal, the method improved the maximum forward algorithm to output the user access to the maximum forward path during each session. Finally, it calculated the data arrival rate of each functional module of the application in combination with the Apache Spark data processing platform. Toriumi et al. (2016) focused on private chat systems and analyzed the private chat log data provided by LINE PLAY to clarify the behavior of private chat users. They classified users into 15 clusters based on their communication behaviors. To improve the service quality of mobile application users and analyze the requirements and behavior characteristics of users, Wang et al. (2019) designed a mobile user behavior analysis system. At the bottom of the system, the SDK and database were used to collect and save user behavior data, and the Spark platform was used to quickly clean, extract, transform, merge and cluster the data. They also improved the Apriori algorithm, analyzed the correlation of user behavior, and conducted a verification test on the app.

The aforementioned studies did not give comprehensive analysis of the factors affecting the software UX and did not comprehensively analyze the data related to the software UX from various sources. They only used the data from one of the sources for analysis, and the obtained software user experience was not comprehensive enough. The works in (Hu & Jiang, 2019; Maalej & Nabil, 2015; Lu et al., 2014; Guzman et al., 2015; Hussain et al., 2022) mainly studied the software UX contained
in software user comments, excavated the software usage feedback hidden in comments and obtained the description related to software UX. Other studies (Chen, 2016; Ren, 2020; Toriumi et al., 2016; Wang et al., 2019) mainly analyzed the user’s interactive data and runtime data of software, mined the user behavior patterns, usage habits, and hobbies reflected in the data, and used the user’s behavior data to obtain the user’s software UX. However, many factors influence the UX of software. Therefore, this paper proposes the App software UX analysis method based on multisource information fusion technology. The contributions of this paper are as follows.

1. App software user experience is defined, and the influencing factors of App software user experience are analyzed.

2. An analysis method of App software user experience based on multisource information fusion is proposed, which integrates various factors affecting the UX of App software, such as users, software, and the environment. Through the steps of feature engineering, feature correlation mining, feature combination tree construction and multisource data fusion. The interactive information of App software from multiple sources is fused, and the heterogeneous and homogeneous information is extracted, which is helpful to obtain comprehensive analysis results of the App software UX.

3. In this paper, the App software user experience themes on the fused data are analyzed and the App software user experience reflected in them is explored. Experimental results show that this method can effectively and reasonably fuse multisource App software interaction data and extract comprehensive App software user experience information, which helps developers maintain and improve App software in a targeted way.

THE DEFINITION OF APP SOFTWARE USER EXPERIENCE

User experience is “a momentary, primarily evaluative feeling (good-bad) while interacting with a product or service” (Hassenzahl, 2008). By this definition, UX shifts attention from the product and materials (i.e., content, function, presentation, interaction) to humans and feelings – the subjective side of product use. Many researchers focusing on UX are based on the ISO 9241-11:2018(E) (2018), which defines UX as “person’s perceptions and responses resulting from the use and/or anticipated use of a product, system or service.” The definition of UX of ISO.9241-11:2018(E) pointed out that UX includes all the users’ emotions, beliefs, preferences, perception, physical and psychological responses, behaviors and accomplishments that occur before, during and after use. Moreover, UX is a consequence of brand image, presentation, functionality, system performance, interactive behavior and assistive capabilities of the interactive system, the user’s internal and physical state resulting from prior experiences, attitudes, skill and personality, and the context of use. For Forlizzi and Battarbee (2004), UX refers to all use experiences and interactive experiences in a period of time, which has a clear start time and end time. The focus of the definition is to associate the UX with the user’s use behavior and use experience to further clarify the specific content and source of the UX. While there are a few different conceptualizations of UX, researchers agree that UX is affected by user, product or service and the environment of use. In this paper, the UX of the App software is the feeling and satisfaction of the user in the process of interacting with the App software under a specific App software running environment. The UX of App software is a consequence of the user, App software, and App running environment. The influencing factors of the App software UX proposed in this paper are shown in Figure 1.

The user is the main body of App software UX and the embodiment of subjective factors of App software UX. App software and the App running environment are the objective factors that affect the UX of App software. The subjective factors combine with the objective factors to influence the UX of App software.
Furthermore, this study analyzed numerous interactive data of App software, and discovered that there are corresponding data to reflect the impact of different factors, such as user, App software, and App running environment, on App software UX. Therefore, by analyzing the corresponding data, the description result of App software UX can be obtained. The multisource data associated with App software UX are shown in Table 1.

The interactive data of App software from different sources are heterogeneous data that reflect the UX of App software, and the data from different sources are quite different in modality, calibration, granularity and connotation. Thus, the technologies of multisource information fusion are needed to fuse and comprehensively analyze the App software UX information commonly reflected in the multisource App software interaction data. The more comprehensive App software UX can be obtained.

**METHOD OF APP SOFTWARE UX**

In this section, the analysis method of App software UX is introduced in detail. The method comprehensively analyzes the multisource interaction data of App software, which includes user feedback data, App operation process data, and App runtime data. The multisource interaction data of App software are fused by specific rules to obtain the description result or explanation related to the App software and the UX of the App software. However, to obtain the information of App software UX, first, the features of the multisource interaction data of App are analyzed to extract the significant features of the multisource interaction data of App. Then, the correlations between extracted features are mined, and the combinations according to the feature correlation are extracted to build the feature combination tree with specific rules. Finally, according to the feature combination tree, the interactive data of App software from multiple sources are fused, and the information of

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Data Description in Detail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback data of users</td>
<td>The user’s comments of App software, especially the comments related to App software UX in App software user’s comments. User’s feeling and internal satisfaction are included in the feedback data, which is directly related to UX.</td>
</tr>
<tr>
<td>Operation process data of App</td>
<td>The operation process data of App are captured by the user in the process of using the App, which can reflect the functions and services that App software itself can provide to users. The interactive information between users and App software in a period of time can be obtained by analyzing this part of data.</td>
</tr>
<tr>
<td>Runtime data of App</td>
<td>The log information, memory information, network environment data and other data related to the running environment of the App during the running process.</td>
</tr>
</tbody>
</table>

Figure 1. The influencing factors of app software UX
App software UX is analyzed. This method will be helpful to obtain more complete, accurate, and reliable comprehensive analysis results of the App software UX more conveniently. The flow chart of the analysis method of App software UX is shown in Figure 2.

**Feature Extraction of Interactive Data From Multisource App Software**

The multisource interaction data of App are multisource homogeneous and heterogeneous information obtained in different spaces or at different times, which together reflect the UX of App software. Therefore, the heterogeneous information must be abstracted, merged or integrated to map to the same computing space. In the same computing space, the App software UX of homogeneous information is obtained more easily.

To map the multisource interaction data of the App to the same computing space, first, data preprocessing for multisource App interaction data is needed, which includes data cleaning, data conversion, and data filtering. Among them, data cleaning includes the removal of missing and invalid value data such as empty, all punctuation, special characters, web links, stop words, and advertisements from all source data. For example, users’ comments that are empty or full of punctuation marks will be cleaned. The data conversion consists of the conversion of capitalization and the conversion of traditional and simplified Chinese characters. The main purpose of data filtering is to remove nonnumeric, non-Chinese, nonalphabetic, nonpunctuation, and other data that cannot be accurately decoded and recognized.

Furthermore, the App interaction data from various sources have an emphasis on reflecting the UX of App software, and it is necessary to carry out feature engineering on the data to further abstract and extract the features related to the UX of App software. The heterogeneous data from different sources can be mapped to the same computing space. In this paper, the App software UX analysis method comprehensively analyzes the multisource App software interaction data mentioned above, and the corresponding models are used for feature engineering of the features of each source data.

![Figure 2. The flow chart of the analysis method of app software UX](image-url)
Feedback Data of Users. The data of this source are user comments. Reference (Hu & Jiang, 2019) argued that in the process of mining user comments that contain usage feedback, the software features concerned by users are the evaluation object and the evaluation opinion, and the evaluation opinions are views expressed by users. The object of evaluation is the target entity in which the viewpoint holder expresses emotion, usually consisting of one or more words. The opinion of evaluation is the word and expression with emotional tendency that can express the user’s own opinions. It is the fundamental basis for judging the user’s emotion to the evaluation object. Users express different opinions for different evaluation objects, and different types of usage feedback can be mined by classifying evaluation opinions. Therefore, for the user feedback data, the dependency syntactic structure and part-of-speech rules method are used in this paper to extract the evaluation objects and the evaluation opinions in App software user’s comments, and the word vector method is used to quantify them.

Operation Process Data of the App. The operation process data of the App are captured by the user during the process of using the App. The paper (Chen & Jiang, 2022) suggested that a key role in mining the behavior of App users in the process of App operation data is played by the features which are operation object and operation content. Among them, the operation object in the paper is the name of the App, and the operation content is the function of the App. Thus, the features obtained by analyzing the operation data of the App are shown in Table 2.

Runtime Data of App. The runtime data of App include log data, memory data, and network environment data when the App is running. Paper (Liu, 2020) believed that the log object, which is the App name in App log data, and the log occurrence time, play a critical role in mining user behavior in App log data. Furthermore, the error information in App log data is very important (Chen, 2016). Therefore, the features obtained by analyzing the runtime data of the App are shown in Table 3.

The features of data from different sources have different influences on the UX of App software. Therefore, it is necessary to further analyze the obtained features and rank the features according

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User ID</td>
<td>The unique identification of user</td>
</tr>
<tr>
<td>Operation object</td>
<td>The name of the App currently used by the user</td>
</tr>
<tr>
<td>App function</td>
<td>The activity started by the App currently used by the user and the functions provided by the App for user</td>
</tr>
<tr>
<td>User’s operation</td>
<td>The description of the specific operation that the user currently performs when using the App</td>
</tr>
<tr>
<td>Number of steps</td>
<td>Number of steps required by the user to perform the current operation</td>
</tr>
<tr>
<td>Usage times</td>
<td>The number of times the user performs this operation in a period of time</td>
</tr>
</tbody>
</table>

Table 3. The features of app runtime data

<table>
<thead>
<tr>
<th>Feature Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log object</td>
<td>The name of the App that currently generates the log record</td>
</tr>
<tr>
<td>Log time</td>
<td>The generation time of current log record</td>
</tr>
<tr>
<td>Log level</td>
<td>The level of current log record</td>
</tr>
<tr>
<td>Log error</td>
<td>Error description of current log record</td>
</tr>
<tr>
<td>Memory occupation</td>
<td>Memory size occupied by the currently used App</td>
</tr>
<tr>
<td>Network state</td>
<td>The network state where the App is currently running</td>
</tr>
</tbody>
</table>
to the influence degree and contribution rate of each feature on the UX of App. The importance of features is determined. Numerous data from various sources are analyzed, and the findings are summarized as follows.

- **Rule 1.** The length of specific data of the feature can represent the detailed degree of the feature’s description of software interaction information, and the longer the length, the more detailed the description, the greater its influence on the UX of App software;

- **Rule 2.** Nonduplicated data in specific data of features represent the diversity of data. The higher the diversity is, the richer the interaction information of the software, and the higher the importance and priority.

Therefore, according to the above analysis, the method of this paper formulates the rules for calculating feature weights. The feature weights of each source data are calculated separately, and the priority of the features is also determined separately.

The feature weight determined by rule 1 is calculated by Eq. (1) where $weight_{i1}$ is the weight of feature $i$ determined by rule 1 and $\text{len}(\text{feature}_j)$ is the lengths of the data $j$ of feature $i$.

$$weight_{i1} = \text{Mean}\{\text{len}(\text{feature}_{i1}), \ldots, \text{len}(\text{feature}_{in})\}$$  \hspace{1cm} (1)

The feature weight determined by rule 2 is calculated by Eq. (2) where $weight_{i2}$ is the weight of feature $i$ determined by rule 2 and $\text{num}(\text{feature}_i)$ is the number of nonduplicate data of feature $i$.

$$weight_{i2} = \text{num}(\text{feature}_i)$$  \hspace{1cm} (2)

Since the range of weights calculated by Eq. (1) and Eq. (2) is inconsistent, the weights need to be mapped to the same range of values to facilitate the subsequent calculation and measure the importance of features. The range is set as $[0, 1]$. The total feature weight is calculated by Eq. (3) where $weight_{i}$ is the weight of feature $i$ and $\text{normalization()}$ means normalizing the weight.

$$weight_{i} = \text{normalization}(weight_{i1}) \ast \text{normalization}(weight_{i2})$$  \hspace{1cm} (3)

Some of the features mentioned above are written in natural language, some are nominal or interval types, and some are numerical types. The data from different sources are quite different in modality, calibration and structure. To map the data with different structures and calibration into the same computing space, intermediate media such as word vectors and embedding are needed. For example, the features are represented by natural language, the meaning of data and the embodied UX are contained in the semantics of data, and it is necessary to analyze its semantics to obtain the information of App software UX. For the features described by a nonnatural language, the internal UX of App software is reflected in the specific values of its features and the semantics represented by the values. It is necessary to analyze the specific values and the semantics represented by the values together, and the information of App software UX can be analyzed. Therefore, different quantization methods for different data types are needed to map them into the same computing space.

For the features described by natural language, the word vector model is used to quantify them. The word vector model based on neural networks and deep learning has been widely used in the field of natural language processing (Jiang et al., 2017). Here, the corpus data of Wikipedia are used to train the Word2Vec model with the Python third-party open source gensim library, and the dimension
of the word vector is set to 400. Then, the word frequency of each word in this batch of data is counted to obtain the proportion of word frequency of each word in this batch of data. For the word vector \( P_i \) of data \( i \), it is calculated by Eq. (4), where \( \text{rate}_{ij} \) is the word frequency proportion of word \( j \) of data \( i \) in this batch of data and \( v_{ij} \) is the word vector of word \( j \) of data \( i \).

\[
P_i = \sum_{j=1}^{n} \text{rate}_{ij} \times v_{ij}
\]  

(4)

To quantify the features described by nonnatural language, the one-hot coding method is used to process the nominal or interval type data. Then, the embedding method is used to transform its shape and size into the same size as the word vector above. Moreover, it is also necessary to give semantic meaning to these features. The semantics of the feature are word vectorized and then spliced and combined with the vector obtained by one-hot coding to obtain the corresponding vector. For the numerical data described by a nonnatural language, the normalization method of Eq. (5) is used for processing.

\[
f'_i = \frac{f_i}{\left(\sum_{j=1}^{n} f_j\right)}
\]

(5)

Where \( f'_i \) represents the value of the data \( i \) after normalization, \( f_i \) is the value of the data \( i \) of numerical feature, \( f_j \) is the value of the data \( j \) of the numerical feature, and \( n \) represents the data quantity of the feature. The normalized value is also expressed in vector form. The semantic vector of the feature is fused with the vector of normalized values to obtain the quantitative vector of this type of feature. In summary, the method of quantifying the features of natural language types, such as the evaluation object, evaluation opinion, operation object, App function, user operation, log object and log error, is the word vector. The method of quantifying the features of nominal or interval types, such as log time, log level and network state, is one-hot coding + embedding + word vector. The quantification method of numerical features, such as the number of steps, usage times, and memory occupation, is normalization + word vector.

**Feature Correlation Mining and Feature Combination Tree Construction of Multisource App Software Interactive Data**

After feature extraction of multisource App software interaction data, the features of each source data can be extracted, and the data can be quantified and mapped to the same computing space. The data from various sources are heterogeneous data, and there are correlations between their features, but the correlations are more implicit and not significant enough. Therefore, it is necessary to mine the correlations of the features that have been subject to feature engineering and further clarify the correlations between the features to mine the homogeneous data in multisource data. Moreover, considering that there may be homogeneous and complementary information in different features, the integration of these features will help to analyze the App software UX information contained therein. Therefore, this method constructs the feature combination according to the corresponding rules to obtain the set of feature combinations.

The Euclidean distance can measure the distance between two points in m-dimensional space. The shorter the Euclidean distance is, the closer the two points are and the higher the similarity is. In contrast, the farther away they are, the lower the similarity is. Therefore, this paper uses Euclidean distance to measure the similarity between features. The similarity is calculated by Eq. (6), Eq. (7)
and Eq. (8), where $S_{(i,j)}$ is the similarity of feature $i$ and feature $j$, $S_{im}$ is the similarity between the data $m$ in feature $i$ and all data in feature $j$, $v_{im}$ is the vector of data $m$ of feature $i$, and $D_{(X,Y)}$ is the normalized Euclidean distance between vectors $X$ and $Y$.

$$D_{(X,Y)} = \frac{1}{1 + \left( \sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}}$$ (6)

$$S_{im} = \max\{D(v_{im}, v_{i1}), D(v_{im}, v_{i2}), \ldots, D(v_{im}, v_{in})\}$$ (7)

$$S_{(i,j)} = \operatorname{Mean}\{S_{i1}, S_{i2}, \ldots, S_{ik}, \ldots, S_{im}\}$$ (8)

After the similarity between two features is calculated by the above method, the feature combinations are constructed according to the rule that there is at least one feature in the data of each source. Then, the average similarity of this feature combination is calculated, which is the average value of the similarity of any two features in the feature combination. It is used as the similarity of this feature combination, and the feature combination set can be obtained. The algorithm of feature correlation mining and feature combination construction is shown in Algorithm 1.

**Algorithm 1**
The algorithm of feature correlation mining and feature combination construction

**Input:** FeatureList //List for storing features.
Multisource data set D

**Output:** Feature combination set F
F = Ø;
doubleFeatureList = [] //List for storing pairwise features and their similarity.
for i = 0 to FeatureList.size() do
    for j = i+1 to FeatureList.size() do
        tempFeature_1 = FeatureList.get(i);
        tempFeature_2 = FeatureList.get(j); //Get pairwise features.
        data_1 = getData(tempFeature_1, D); //Get specific data corresponding to the features.
        data_2 = getData(tempFeature_2, D);
        Similarity = CalculateSimilarity(data_1, data_2); //Calculate the similarity of two features.
        doubleFeatureList.add(tempFeature_1, tempFeature_2, Similarity); //Store pairwise features and their feature similarity to the list.
    end for
end for
Combinations = generateCombinations(FeatureList); //According to the rule to construct the feature combination.
for i = 0 to Combinations.size() do
    tempCombinations = Combinations.get(i);
    MS = Mean(doubleFeatureList.find(tempCombinations).getSimilarity()); //Read the similarity of pairwise features and calculate their average similarity.
    F.add(tempCombinations, MS); //The feature combination similarities are stored in F.
end for
return F;

Through the algorithm of feature correlation mining and feature combination construction, the relation between features of different data sources can be mined, and the feature combinations can be constructed to provide support for the construction of the feature combination tree.

In this paper, the authors draw lessons from the concept of a “binary tree” in a data structure to propose the concept of a “feature combination tree”. “Feature combination tree” refers to extracting feature combinations from the set of feature combinations with corresponding strategies to generate a “minimum spanning tree”. The nodes of each layer in the tree are a kind of feature combination. The “minimum spanning tree” is a tree that ensures the reasonable fusion of data with the minimum number of nodes. The construction strategy of the feature combination tree is to select the feature combination with the least number of features, the largest number of features that have not appeared in the previous nodes and the highest average feature similarity in the feature combination to join the tree, and the left child nodes are added first. The algorithm for constructing the feature tree is shown in algorithm 2.

Algorithm 2 The algorithm for constructing the feature combination tree
Input: Number of classifications of data fusion K
Feature combination set F
Output: Feature combination tree T
T = Ø;
leafNode = 0;
while leafNode < K:
    tempNode = getCorrectCombination(F);  //Get the correct feature combination according to the strategy.
    T.add(tempNode)
    F.remove(tempNode);
    leafNode = getLeafNode(T);
return T;

According to the feature combination tree algorithm, the minimum feature combination set meeting the fusion requirements can be constructed, which can guide the data to be fused purposefully on the basis of ensuring the data fusion to meet the fusion requirements, instead of fusing blindly, avoiding the waste of computational resources and computational time, and improving the performance of the multisource data fusion algorithm. A schematic diagram of the constructed feature combination tree is shown in Figure 3.

The feature combinations contained in the feature tree are the minimum feature combination set that meets the requirements of data fusion. The feature combination in each node in the tree can guide the data fusion of App interactive data from various sources and provide a theoretical basis for the next analysis of App software UX.

Analysis of App Software User Experience

The UX of App software is the comprehensive feeling and satisfaction of users, and the UX of App software obtained from one or two kinds of data analysis is incomplete and inaccurate. Therefore, fusing the interactive data of App software from various sources is needed. First, the data with strong correlation are aggregated, and then data clusters with strong correlation in the UX of App software are obtained. Finally, the data clusters are further analyzed to obtain comprehensive UX results of App software. The feature combination tree constructed in section “Feature Correlation Mining and Feature Combination Tree Construction of Multisource App Software Interactive Data” is the key basis to guide the interactive data of multisource App software, and data fusion based on the feature combination tree is the critical step of UX analysis of App software.
Considering the large amount of interactive data of App software from different sources, if the supervised methods are used for data fusion, a great quantity of data labels are needed, and a lot of time is taken to train the model. Furthermore, supervised methods are computationally intensive, inefficient in data fusion, and have poor performance. In contrary, unsupervised methods do not require training models and have better performance with less computational overhead. Therefore, an unsupervised method is used to fuse data in this paper. Clustering is a simple but effective unsupervised method that has significant advantages for the analysis of a large amount of data. The partitioning around medoids (PAM) algorithm belongs to the partitioning-based methods of clustering, which is an improvement of K-means. The largest difference between them is the iterative correction strategy of the clustering center. The clustering center selected by the K-means algorithm is the mean point of all points in the current cluster, while the clustering center selected by the PAM algorithm is the point with the smallest distance from other points in the current cluster. The K-means algorithm is greatly influenced by noise and edge values, and the cluster center may not be actual data in the data. However, the clustering center selected by the PAM algorithm each time is a piece of data in the data, and finally, several clusters are obtained by the PAM algorithm. The sum of the distances between each element in the cluster and the cluster center is the smallest. The PAM algorithm is less influenced by noise and edge value. Therefore, this paper chooses the PAM algorithm for multisource data fusion.

In the multisource data fusion section, the feature combination tree obtained in the feature correlation mining and feature combination tree construction section is the basis. Each layer of the feature combination tree is a fusion process, which includes one or more data fusion activities. At each node of the tree, one data fusion activity is performed. One data fusion activity can obtain two or several data fusion results. After multiple fusion processes according to the feature combination tree, all multisource homogeneous heterogeneous data will be reasonably fused into several classifications.

Moreover, here, only homogeneous or complementary multisource App software interactive information is obtained after fusion. To obtain the information of App software UX in the fused data,
extracting the topic keywords from the fused data to obtain the core information of App software UX is needed. LDA (Latent Dirichlet Allocation) topic models enable topic keyword extraction by representing topics as distributions over words to obtain the correlation between topics and words (Tian et al., 2017). However, the LDA topic model is a supervised method, which requires predefined topics and training the model using a large corpus. In this paper, it is not intended to predefine the topics of the fused data, but to obtain the topics of this class of fused data by analyzing the data; therefore, the supervised topic extraction method is not applicable to this paper. The unsupervised topic keyword extraction methods are TF-IDF, Text Rank, etc. Since TF-IDF only calculates word weights by statistical features, it ignores the consideration of low-frequency words and the contextual semantic information of the documents. In contrast, Text Rank can better extract the subject keywords in the data by using the position relationship of words in the document to construct a graph and then using the graph-based ranking algorithm to iteratively calculate until convergence through the corresponding iteration, which fully considers the contextual semantic information of words (Tian et al., 2017). Therefore, the keyword extraction algorithm based on Text Rank is used to extract the topic keywords of the fused multisource App software interaction data and to construct the App software UX topic keyword clusters. The multisource data fusion algorithm is shown in algorithm 3.

**Algorithm 3** The multisource data fusion algorithm

**Input:** Feature combination tree $T$

**Multisource data set $D$**

**Output:** result  //The result of data fusion.

result = Ø;

for level in $T$ do

levelNode = $T$.getLevelNode(level);  //Get the nodes contained in current layer of the feature combination tree.

featureList = Ø;

for node in levelNode do

featureList.add(levelNode.getFeature(node)); //Combining the featureList.

end for

featureData = Ø;

for feature in featureList do

tempFeatureData = getFeatureData(feature, $D$); //Query the data corresponding to the feature and store it in the data set.

featureData.add(tempFeatureData);

end for

tempResult = PAM(featureData);  //According to the feature combination, PAM algorithm is used to fuse.

$D$ = tempResult;  //Update the multisource data set to the data that has been fused once.

result.add(tempResult);  //Store the data fusion results, and save the results of each fusion.

end for

return result;

The combination of the multisource data fusion algorithm and Text Rank keyword extraction algorithm of the App software UX analysis step can automatically fuse multisource App software user experience information and extract comprehensive App software user experience information, finally achieving the goal of this paper’s method.
EXPERIMENTS AND ANALYSIS

Experimental Data and Sources

To more comprehensively verify the effectiveness of this method, the authors randomly crawled the user comments of various App software from the Android App market to build the user’s comments library of App software and developed an App named “App Operation Process Analysis Software (AOPAS)” using the Android Studio development platform to collect the operation process data of App and the runtime data of App. The AOPAS can monitor the user’s process information of operating Apps and the runtime information of Apps in real time, and it also allows user to view the obtained operation process information of Apps and the runtime information of Apps. The AOPAS can run as a service in the background of the user’s terminal. When the user operates the Apps installed on the terminal, AOPAS can capture the user’s operation content and operation trajectories. Furthermore, the AOPAS will request the user to give the corresponding permissions, which solicits the user’s consent to ensure legitimacy. The user can clearly know what information the AOPAS will obtain from the user. The AOPAS was used to collect the App operation process data and the App runtime data of various users over a period of time. Finally, 117,661 pieces of experimental data were acquired. After data preprocessing, 105,698 pieces of interactive experimental data were obtained, which involved more than 30 Apps. The number of each source of experimental data involved in the experiment was obtained according to the proportion of data actually presented, and the number of experimental data from three sources shows an imbalance in reality, with the most runtime data of App, the second most user’s comment data of App, and the least operation process data of App. For each App in the experiment, its data from three sources were analyzed. The details of the experimental data after data preprocessing are shown in Table 4.

1“1;com.sogou.activity.src;com.sogou.reader.RecommendActivity$1;2021-05-13;Loading;2;168” means that the user with user number 1 started the “Sogou search” App on 2021.5.13 and clicked the “Loading” button. The number of steps required by the user to perform the current operation was 2, and the number of times the user performs this operation in a period of time was 168. “2021-05-13 10:10:13.941 W/PackageManager(1431): Unknown permission com.android.launcher.permission. READ_SETTINGS in package com.sogou.activity.src,980,false” indicates that a warning of the “PackageManager” type was generated when the user used the “Sogou search” App because the “READ_SETTINGS” permission was unknown, and the log level was warning. The memory occupied by the App was 980kb, and its network status was connected. The example data in Table 4 contains portion fields of the data.

<table>
<thead>
<tr>
<th>id</th>
<th>Data Source</th>
<th>Data Quantity/Piece</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>User’s comment data of App software</td>
<td>31776</td>
<td>The loading interface design of the software is first-class, the style is my favorite.</td>
</tr>
<tr>
<td>2</td>
<td>Operation process data of App</td>
<td>12267</td>
<td>1;com.sogou.activity.src;com.sogou.reader.RecommendActivity$1;2021-05-13;Loading;2;168</td>
</tr>
</tbody>
</table>
Feature Correlation Mining Experiment

In this experiment, the preprocessed experimental data were used to verify the effectiveness of feature engineering and feature correlation mining algorithms. The word vectors of each feature data were obtained through feature engineering, and the correlation between features was mined by feature correlation mining algorithms. Here, the authors focus on the influence of the amount of data used to calculate feature similarity. To more comprehensively evaluate the influence of the amount of data on the calculated feature similarity, six different data quantities were used to conduct the feature correlation mining experiment (i.e., 10%, 20%, 40%, 60%, 80% and 100% of the total experimental data). The pairwise feature combinations with the top 10 feature similarities in the results of the experiment are shown in Table 5.

The “feature priority” column in the table refers to the priority order of the feature among the many features in its corresponding source data. The smaller the number of priorities, the higher the priority. Its priority is determined according to the weight calculated in the feature engineering. The greater the weight is, the higher the priority. Table 5 reveals that the feature correlation calculated by using 20% of the experimental data is very similar to that calculated by using more than 20% of the experimental data. However, the feature correlation calculated by using 10% of the experimental data is quite different from that calculated by using more than 10% of the experimental data. Therefore, using only 20% of the experimental data for feature correlation mining can mine the correlation between the corresponding features, which helps to reduce the amount of calculation and improve the efficiency of the algorithm. Moreover, according to the experimental results, the two feature combinations with the top 10 feature similarities include the top 3 features of each source data, and the similarities between features are high. These results show that the calculation method of the feature weight in feature engineering and the feature correlation mining algorithm are reasonable and effective.

App UX Analysis Experiment

In this section, the quantitative feature data after feature engineering and the constructed feature combination tree construction were used to carry out the data fusion experiments. Multisource data fusion was guided by the feature combination tree. Here, the authors focus on the influence of the classification number K of data fusion on the fusion results. The classification number K of data fusion was set to 2, 4, and 6 (the amount of data used to calculate feature similarity was set to 20% of the experimental data), and the corresponding feature combination tree was constructed.

Table 5. The pairwise feature combinations with the top 10 feature similarities

<table>
<thead>
<tr>
<th>id</th>
<th>Feature Name</th>
<th>Feature Priority</th>
<th>Feature Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>10%</td>
</tr>
<tr>
<td>1</td>
<td>(Operation object, Log object)</td>
<td>(3,2)</td>
<td>1.000</td>
</tr>
<tr>
<td>2</td>
<td>(Evaluation object, Operation object)</td>
<td>(1,3)</td>
<td>0.793</td>
</tr>
<tr>
<td>3</td>
<td>(Evaluation object, Log object)</td>
<td>(1,3)</td>
<td>0.793</td>
</tr>
<tr>
<td>4</td>
<td>(Evaluation object, App function)</td>
<td>(1,1)</td>
<td>0.796</td>
</tr>
<tr>
<td>5</td>
<td>(Evaluation object, User’s operation)</td>
<td>(1,2)</td>
<td>0.797</td>
</tr>
<tr>
<td>6</td>
<td>(Operation object, Log error)</td>
<td>(3,1)</td>
<td>0.792</td>
</tr>
<tr>
<td>7</td>
<td>(Log object, Log error)</td>
<td>(3,1)</td>
<td>0.792</td>
</tr>
<tr>
<td>8</td>
<td>(Evaluation object, Log error)</td>
<td>(1,1)</td>
<td>0.784</td>
</tr>
<tr>
<td>9</td>
<td>(User’s operation, Log error)</td>
<td>(2,1)</td>
<td>0.778</td>
</tr>
<tr>
<td>10</td>
<td>(App function, User’s operation)</td>
<td>(1,2)</td>
<td>0.561</td>
</tr>
</tbody>
</table>
In this experiment, 105,698 pieces of data were randomly divided into five groups, each containing interactive data from App software from three sources. Five data fusion experiments were conducted, and the order of all experimental data was random. To evaluate the effectiveness of this method for multisource data fusion of App software, 6 users were invited to manually check the results of each experimental data fusion to determine whether the data were classified correctly and whether the data in the classification truly had a strong relation. The experimental results of multisource data fusion with the classification number K=2 of data fusion are shown in Table 6. The experimental results of multisource data fusion with the classification number K=4 of data fusion are shown in Table 7. The experimental results of multisource data fusion with the classification number K=6 of data fusion are shown in Table 8.

The experimental results show that the average accuracy of data fusion is above 70%. Among them, the fusion categories with higher accuracy, especially class 1, indicate the emotional tendency of the interface aesthetics as good, and the data characteristics of this class are more obvious; therefore, the effect of clustering is better, and the accuracy is higher. In contrary, the characteristics of the data in the fusion categories with lower accuracy are more implicit and consistent with the characteristics of the data itself. Through in-depth analysis of the experimental results, it is found that there are two possible reasons for the lower accuracy of some fusion categories. On the one hand, only some features were used to fuse multisource data, and some information may be missing. On the other hand, the similarity calculation method currently used was relatively simple, which may lead to misjudgment during the fusion. These limitations of this method need to be improved in the future. Furthermore, the following situations were observed:

- When K = 2, the average accuracy of fusion is higher, but the fused classifications are less and the granularity is coarse; compared with K = 2, when K = 4, the average accuracy of fusion is lower, but the decrease is small; however, there are more classifications fused, and the granularity is in the middle; when K = 6, the classifications fused are the most, but the average accuracy is the lowest.
- With the increase in the K value, the average accuracy of data fusion decreases gradually, and the time spent on data fusion increases by degrees, but the granularity of data fusion becomes finer. Therefore, it is necessary to find a balance among granularity, accuracy and efficiency to

---

**Table 6. Experimental results of multisource data fusion with K = 2**

<table>
<thead>
<tr>
<th>id</th>
<th>Data Quantity/Piece</th>
<th>Data Fusion Classification</th>
<th>Classification Accuracy/%</th>
<th>Average Accuracy/%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21139</td>
<td>Class 1</td>
<td>85.3</td>
<td>84.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>83.2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>21139</td>
<td>Class 1</td>
<td>84.1</td>
<td>82.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>81.6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21139</td>
<td>Class 1</td>
<td>86.7</td>
<td>83.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>80.5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21139</td>
<td>Class 1</td>
<td>83.8</td>
<td>82.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>81.5</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>21142</td>
<td>Class 1</td>
<td>82.9</td>
<td>81.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>80.8</td>
<td></td>
</tr>
</tbody>
</table>
ensure that they are within an acceptable range. Through the experiment, K = 4 is suggest to set, which can make the granularity, accuracy and efficiency of data fusion more reasonable.

The authors also focus on the influence of the combination of the amount of data for calculating feature similarity and the classification number K of data fusion on the accuracy of data fusion. Therefore, the relevant experiments were carried out and the experimental results were recorded. The experimental results are shown in Figure 4.

From the experimental results, it can be identified that the accuracy of 20% of the experimental data used to calculate the feature similarity is higher than that of 10% of the experimental data used to calculate the feature similarity. Since the feature similarity calculated by taking 20% of the experimental data and the feature similarity calculated by taking 40% of the experimental data are basically the same, the difference in average accuracy is small. This also shows the feasibility and rationality of calculating feature similarity with partial data. To improve the computational efficiency of the algorithm, it is suggested that 20% of the data be used to calculate the feature similarity. Furthermore, when the amount of data for calculating feature similarity is the same, the greater the K value is, the lower the accuracy; however, the granularity of data fusion classification is finer. Through the experiments, the K value of 4 suggested is reasonable, which can take into account not only the accuracy but also the efficiency and the granularity of fusion. From the above experimental results and analysis, the conclusion is that the analysis method of App software UX based on multisource information fusion can reasonably and effectively fuse the interactive data of multisource App software.

<table>
<thead>
<tr>
<th>id</th>
<th>Data Quantity/Piece</th>
<th>Data Fusion Classification</th>
<th>Classification Accuracy/%</th>
<th>Average Accuracy/%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21139</td>
<td>Class 1</td>
<td>92.6</td>
<td>79.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>61.1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 3</td>
<td>90.3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 4</td>
<td>72.6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>21139</td>
<td>Class 1</td>
<td>90.9</td>
<td>77.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>60.2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 3</td>
<td>89.6</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 4</td>
<td>70.6</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21139</td>
<td>Class 1</td>
<td>91.3</td>
<td>78.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>60.8</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 3</td>
<td>89.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 4</td>
<td>71.2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21139</td>
<td>Class 1</td>
<td>90.1</td>
<td>76.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>60.4</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 3</td>
<td>88.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 4</td>
<td>68.1</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>21142</td>
<td>Class 1</td>
<td>91.7</td>
<td>78.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>61.9</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 3</td>
<td>89.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 4</td>
<td>70.6</td>
<td></td>
</tr>
</tbody>
</table>
Furthermore, in this section, the keyword extraction algorithm based on Text Rank was used to extract the topic keyword cluster of each classification of fusion data. The number of keywords in the topic keyword cluster was set to 20, which means that the top 20 topic keywords are extracted. The results of the topic keyword cluster extracted from the fusion data with experimental id “1” in Table 7 above are shown in Table 9.

Table 9 reveals that the UX of App software embodied in the class 1 topic keyword cluster is mainly related to the usability of the App interface, and the UX reflected is good. The UX of App software reflected in the class 2 topic keyword cluster is mainly related to the usability and functionality of the App interface,
and the UX is relatively poor. The UX of App software embodied in the class 3 topic keyword cluster is mainly related to the functional availability and experience of App, and the UX is also good. The UX of App software embodied in the class 4 topic keyword cluster is mainly related to the functional availability and experience of App, and the UX is relatively poor. The “Example” column in Table 9 is the user’s

Table 9. The results of topic keyword clusters extracted from the fusion data with experimental id “1” in Table 7

<table>
<thead>
<tr>
<th>Experiment id</th>
<th>Data Quantity/Piece</th>
<th>Data Fusion Classification</th>
<th>Topic Keyword Cluster</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21139</td>
<td>Class 1</td>
<td>interface, like, feel, software, news, speed, watch, mobile phone, function, download, payment, use, suitable, style, recommendation, time, operation, killing, fun, good</td>
<td>QQ: It is fast, easy to use, and the interface is simple and friendly.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 2</td>
<td>interface, speed, utility, function, software, garbage, setting, download, service, dislike, game, system, activity, inability, time, beeper, manager, feel, access denial, flash back</td>
<td>JingDong: Bad review. I bought something yesterday. The interface prompts that the payment is successful, however, there is no order information. I hope it can be improved.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 3</td>
<td>like, hope, recommend, utility, feel, member, software, things, feel, mobile phone, movie, everyone, effective, video, download, advertisement, content, continue, customer service, TV series</td>
<td>IQiYi: I feel this software is very good and comprehensive. You can see good movies in the software, and there are many resources, I like it.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Class 4</td>
<td>garbage, software, wechat, inoperative, driver, unusable, mobile phone, Didi, button, hope, problem, cancel, download, verify, login, friends, customer service, bili, function, no response.</td>
<td>Wechat: I hope we can change the theme in WeChat.</td>
</tr>
</tbody>
</table>
comment data of App software in the data fusion results after using the method in section “Analysis of App Software User Experience”. For example, the example data in the class 2 expresses user’s dissatisfaction with JingDong App. The reasons and specific functions of dissatisfaction are clearly pointed out, that is, “the payment page prompts successful payment but does not display order information”. Therefore, developers need to focus the “payment” and “display order information” functions of JingDong App, and check whether the bug really exists. If it does exist, it needs to be repaired. With this method, the efficiency of software maintenance and improvement can be effectively implemented, and the App can be more in line with users’ usage habits. Therefore, it is necessary to pay attention to the classes that reflect bad UX information which can help developers improve and upgrade related Apps. Furthermore, the classes that reflect good UX information can tell developers which functional modules and style designs of the Apps are outstanding, and help developers to accumulate good experience for App software design.

The experimental results show that the analysis method of App software UX can effectively extract the information of App software UX from the multisource App software interaction data and obtain the comprehensive analysis results of App software UX.

CONCLUSION

In this study, the authors focused on the App software interaction information of multiple sources that contained the UX of App software, and proposed the analysis method of App software UX based on multisource information fusion. First, through feature engineering, feature correlation mining, feature combination tree construction and UX analysis of App software, the interactive data of App software from different sources were processed. Then, the homogeneous and heterogeneous information from different sources was reasonably integrated. Finally, the topic keyword clusters of the fusion data to obtain more accurate and reasonable comprehensive analysis results of App software UX were extracted, which could provide services for further mining App software UX.

Extensive experiments well demonstrated the feasibility and effectiveness of the analysis method of App software UX based on multisource information fusion. However, only some features of the data were used to guide the fusion of multisource data, some information may be lost; Moreover, the similarity calculation method currently used was relatively simple, which may lead to misjudgment during the fusion. The above two aspects lead to poor effect for complex multisource data fusion. Therefore, the next step will improve the methods of feature extraction and association mining to further improve the methods proposed in this paper. Furthermore, this paper did not further analyze the App software UX of the extracted topic keyword clusters and did not further mine the results of App software UX. Therefore, the deep meaning of the fused multisource homogeneous heterogeneous interactive information of App software will be study in the future. Mining the UX pattern and user behavior pattern of App software in the interactive information is also one critical future work of the study.

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).

CONFLICT OF INTEREST

The authors declare that there are no conflicts of interest regarding the publication of this paper.
REFERENCES


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

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.

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