Real-time On-Device Troubleshooting Recommendation for Smartphones

Keiichi Ochiai*
NTT DOCOMO, INC.

Kohei Senkawa†
NTT DOCOMO, INC.

Naoki Yamamoto
NTT DOCOMO, INC.

Yuya Tanaka
NTT DOCOMO, INC.

Yusuke Fukazawa
NTT DOCOMO, INC.

ABSTRACT

Billions of people are using smartphones everyday and they often face problems and troubles with both the hardware as well as the software. Such problems lead to frustrated users and low customer satisfaction. Developing an automatic machine learning-based solution that would detect that the user has a problem and would engage in troubleshooting has the potential to significantly improve customer satisfaction and retention. Here, we design and implement a system that based on the user's smartphone activity detects that the user has a problem and requires help. Our system automatically detects a user has a problem and then helps with the troubleshooting by recommending possible solutions to the identified problem. We train our system based on large-scale customer support center data and show that it can both detect that a user has a problem as well as predict the category of the problem (89.7% accuracy) and quickly provide a solution (in 10.4ms). Our system has been deployed in commercial service since January, 2019. Online evaluation result showed that machine learning based approach outperforms the existing method by approximately 30% regarding the user problem solving rate.

CCS CONCEPTS

• **Information systems** → **Data mining**; • **Human-centered computing** → **Mobile computing**;

KEYWORDS

Smartphone Usage Log, Customer Support Log, User Assistant, Deep Learning

ACM Reference Format:
Keiichi Ochiai, Kohei Senkawa, Naoki Yamamoto, Yuya Tanaka, and Yusuke Fukazawa. 2019. Real-time On-Device Troubleshooting Recommendation, for Smartphones. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3292500.3330669

*Contact author: ochiaike@nttdocomo.com
†Currently, Recruit Sumai Company Ltd.

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330669

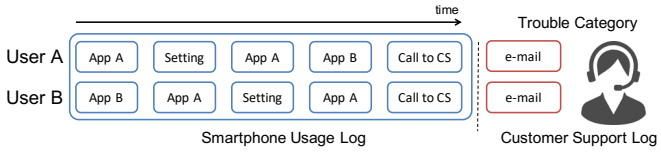

Figure 1: Assumed behavior of a user. If a user has a problem, the user is moving between different apps to try to fix the problem. Then, if the user cannot solve the problem, the user will call the customer support. Thus, we can exploit the customer support log as ground truth for the smartphone usage log.

1 INTRODUCTION

As smartphones become more sophisticated, the number of users who have a problem with smartphone operation is increasing. Thus, call center operations become considerably busy. The numbers of inquiries are more than a few millions per month in the case of Japanese cellular operators[1]. Call center staff guides users in a step-by-step procedure of resolving problems based on the categories of inquiry. Therefore, it is important and useful to automatically detect whether a user has problems or not based on the operation history of the user's smartphone and provide operation troubleshooting according to the content of each problem.

Developing an automatic user problem detection and troubleshooting recommendation is beneficial for both users and mobile operators. In the case of users, a user can conveniently use a smartphone by themself by requesting for automatic user support. In the case of mobile operators, automatic user assistant can reduce the responding operation of inquiries. To this end, the solution has two requirements for the solution, namely, **(R1)** real-time detection (e.g., within several hundreds of millisecond), and **(R2)** accurate detection of user's problem (e.g., more than 80%).

Detecting user problem accurately in real-time is a non-trivial and difficult problem. One simple method is rule-based approach which extracts a transition of an operation supposed to be in problem. However, the coverage of rule-based user problem detection is limited, and managing such rule set is costly with the increasing number of rules because smartphone usage is diverse [31].

Another method is machine learning approach. A machine learning model, particularly deep learning, generally requires a significant amount of annotated data. Moreover, the quality of the ground truth data should be sufficiently high. However, manually labeling

[1]http://www.soumu.go.jp/main_content/000007902.pdf (Japanese Only)

timestamp	class name
2016-08-24 09:16:21	com.nttdocomo.android.dhome.HomeActivity
2016-08-24 09:16:27	com.google.android.finsky.activities.MainActivity
2016-08-24 09:16:32	com.google.android.finsky.activities.MultiInstallActivity
2016-08-24 09:16:45	com.google.android.finsky.activities.MultiInstallActivity

Table 1: Example of smartphone operation logs. Timestamp indicates the time of the event occurred, and class name (also referred to as "activity name" in Android community) is an identifier of a screen of an app.

smartphone operation logs regarding user's problem category is difficult and time consuming because we need to understand the situation of the user from only operation logs. Table 1 shows an example of operation logs. In this example, the logs indicate that the user started from the home screen (`com.nttdocomo.android.dhome.HomeActivity`), then tapped "Google Play[2]" (`com.google.android.finsky.activities.MainActivity`), and finally downloaded and installed some apps in Google Play (`com.google.android.finsky.activities.MultiInstallActivity`). Interpreting each log as described previously when we assume that a user has a problem and the problem category for smartphone usage logs is necessary but difficult.

In this paper, we focus on the behavior of a user who has a problem to address the problem of high-quality ground truth labeling. Figure 1 shows an example of an assumed user behavior. First, we assume that if a user has a problem with how to use an app or set a function, a user explores the functions and setting of the specific app. Second, if the user cannot solve the problem, the user will ask help from a customer support. Finally, the customer support staff records who, when and what category is the user problem regarding user's inquiry. Thus, we can leverage the customer support log as high quality ground truth for smartphone app usage log. Fortunately, we at NTT DOCOMO, which is one of the largest mobile operator in Japan, possess the advantage because we have a customer support center for our customers. Hence, we can collect the previously mentioned data. We also collected the data on smartphone usage logs with user's consent. These data provide us a unique opportunity to tackle the problem of building large-scale high quality labeled data regarding user's problem.

We design, implement and deploy a real-time on-device user assistant application for smartphone usage using machine learning-based user problem detection. Our application recommends a smartphone usage troubleshooting for a user based on the detected problem. Figure 2 shows screenshots of our troubleshooting recommendation. If our application recognizes that a user has a problem, the "Hint" icon will pop up (Figure 2 (a)). The application recommends first a higher-level category such as e-mail and account setting (Figure 2 (b)), and, then a more detailed procedure for a specific setting (Figure 2 (c) and (d)). We formulate the user problem detection problem as a classification problem based on user's smartphone usage logs. Specifically, we predict that a user will inquire or not within T hours. If a user will inquire, we determine the category of inquiry based on a user's smartphone usage logs.

The contributions of this study are the following:

[2]Google and Google Play are trademarks of Google LLC.

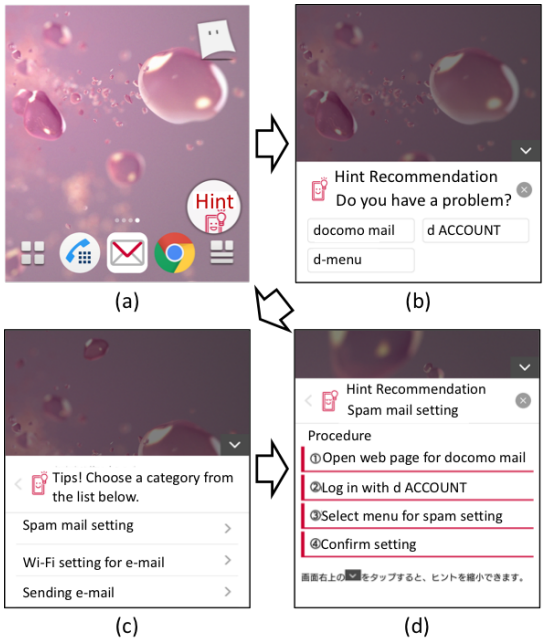

Figure 2: Screenshots of our troubleshooting recommendation service. While a user is using the smartphone, (a) our application automatically detects that the user has a problem, (b, c) and then also identifies what kind of a problem the user has, (d) so that the user can solve the user's problem.

- We formulated a user problem detection problem using smartphone usage history as a classification problem.
- We proposed a model to detect a user problem category from smartphone usage logs. To establish a large-scale high quality labeled data, we exploited the characteristics of call center logs.
- We evaluated the proposed model based on smartphone usage logs of more than 7 months as offline evaluation. The results showed that our system accurately detects a user's problem category (89.7%) and rapidly provide an operation troubleshooting (in 10.4ms).
- We deployed the proposed application in our commercial service since January, 2019. We also conducted an online evaluation in our production environment. The online evaluation result showed that our system improved the user problem solving rate by approximately 30%.

The rest of this paper is organized as follows. The next section offers a problem formulation that the user problem detection problem as a classification problem. We describe our approach in Section 3. We explain an evaluation to validate our approach in Section 4. In Section 5, we summarize our lessons learned from this project. In Section 6, we review related work on the analysis of smartphone application usage and user state estimation from smartphone usage history. Finally, we conclude this study and discuss future work.

2 PROBLEM FORMULATION

In this section, first, we provide the preliminary definition and then formulate the user problem detection problem.

2.1 Preliminary

Definition 1 (Smartphone Usage Log) Each smartphone usage log l consists of a user $u \in U$, timestamp t, and class name (activity name) a, which is the screen identifier of an app. That is, $l = (u, t, a)$.
Definition 2 (Smartphone Usage History) A user's smartphone usage history is a sequence of smartphone usage logs l sorted by timestamp t within specific time duration (e.g., recent T hours). The smartphone usage history of a user is defined as $x_i = \{l_1, l_2, \cdots, l_s\}$ where i is the number of history (sequence) and s is the number of smartphone usage in a sequence.
Definition 3 (User Problem Category) User problem category is defined as a two-level hierarchical category. The top level category c_1 consists of *e-mail*, *Wi-Fi*, *account setting* etc., whereas the lower level category c_2 is composed of *Spam mail setting*, *Sending and receiving mail*, *Attachments* etc. for the *e-mail* category. The number of top level categories is 20, whereas that of lower level categories is 68 in our setting.
Definition 4 (Call Center Log) Each call center log y consists of user u, timestamp t, and problem categories c_1 and c_2. That is, $y_i = (u, t, c_1, c_2)$.

2.2 Problem Formulation

We now formulate our user problem detection problem as follows. The user problem detection problem involves a two-stage problem.
Definition 5 (User Problem Detection Problem) Given the smartphone usage history x_i, our first goal is to estimate the top level category of the user problem c_1 wherein the user will inquire within T hours. Then, considering the smartphone usage history x_i and the user problem category c_1, our second goal is to estimate the lower level category c_2.

A set of smartphone usage history and call center log can be combined based on user u and timestamp t as keys, and, then we can create a dataset $D = \{x_i, y_i\}_{i=1}^{N} = \{x_i, c_{1,i}, c_{2,i}\}_{i=1}^{N}$ for training a machine learning model. In our problem setting, a user problem category and a troubleshooting have one to one correspondence. Thus, if we can detect a user problem category, we can recommend a troubleshooting for the user.

3 APPROACH

Here, we introduce our approach for user problem detection based on smartphone usage. Based on R1 and because our service is provided for Android smartphones, we selected TensorFlow Lite[3] as the framework for machine leaning. As of December, 2018, TensorFlow Lite only supports a limited number of deep learning architectures[4] such as multi-layer perceptron (MLP). Therefore, we considered mainly the MLP model but considered long short-term memory (LSTM) [11] model for performance reference.

[3]https://www.tensorflow.org/lite
[4]https://www.tensorflow.org/lite/tf_ops_compatibility

3.1 MLP Model

The first approach is the MLP model with bag-of-words features, which is often used in natural language processing [19]. We treat each class name a as word, and each sequence x_i as document in natural language processing. In the first stage estimation, where the top level category c_1 is estimated, the frequency of each class name is used as bag-of-words features for classification. In the second stage estimation, where the lower level category c_2 is estimated, we use the top level category selected by the user as input apart from the frequency of each class name because first the user interface of our application provides a candidate of the top level problem category and then the user selects the top level category, as shown in Figure 2. Thus we can use the selected category information in the second stage estimation. We treat the user selected category as one-hot encoded feature. Therefore, the input of the second stage estimation is the frequency of each class name and one-hot encoded user selected category.

Bag-of-words features are generated according to the following procedure using x_i. First, each class name involved in x_i is filtered by a predefined class name dictionary, and then filtered x_i (denoted by x_i') is the output. Here, a predefined class name dictionary is created based on $TF \cdot IDF$ of class name, where $TF(a_j, x_i)$ is the class name frequency used in x_i; and $IDF(a_j)$ is the inverse of the number of sequences, which contain the class name a_j. By using all data D, we calculate $TF \cdot IDF$ for each class name, and use the class names of top-k scored by $TF \cdot IDF$ is defined as class name dictionary. Then, the frequency of each class name is calculated based on x_i'. The purpose of filtering x_i using the predefined class name dictionary is to reduce real-time computational cost and mobile app size on the smartphones because the computational resources and storage capacity are limited on the smartphone.

A Rectified Linear Unit (ReLU) [10] is used as an activation function for all layers except the final fully-connected layer, which uses softmax function as an activation function. The weight of the neural network is learned using Stochastic Gradient Descent (SGD) [6] by setting cross-entropy as the loss function. To avoid overfitting, dropout [26] is used in the fully-connected layer.

3.2 Sequential Feature Model

The second approach is the sequential feature model. One of the limitations of the bag-of-words features is its lack of class name order. To overcome this limitation, we use the LSTM to exploit sequential information. Although the current version of TensorFlow Lite does not support the LSTM model (as mentioned in the beginning of this section), we still investigate the performance of LSTM model for reference.

3.2.1 Simple LSTM Model. Figure 3 shows our simple LSTM model to detect a user problem. Each class name, which is filtered by predefined class name dictionary similar to the bag-of-words model, is fed to LSTM model in the first stage estimation. Similar to the case of the bag-of-words features, the user selected category is used in the second stage estimation and is employed as one-hot encoded feature (Figure 4). Then, the hidden layer and one-hot encoded feature are concatenated, and fed to fully-connected layer. Finally, the softmax activation function is applied to classify the user problem.

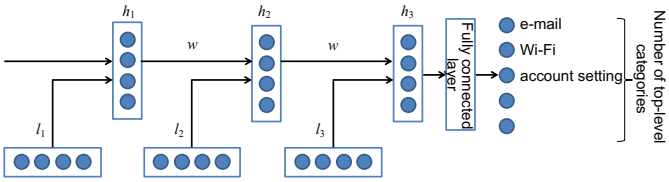

Figure 3: LSTM model for the first stage estimation c_1. The smartphone usage log l is the input for each layer.

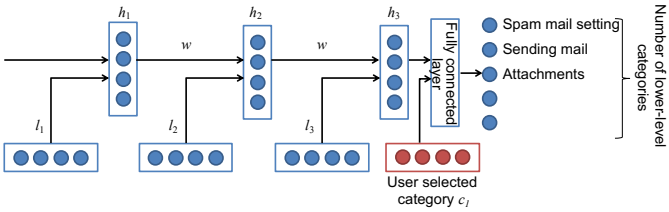

Figure 4: LSTM model for the second stage estimation c_2. We leverage the user selected category as one-hot vector.

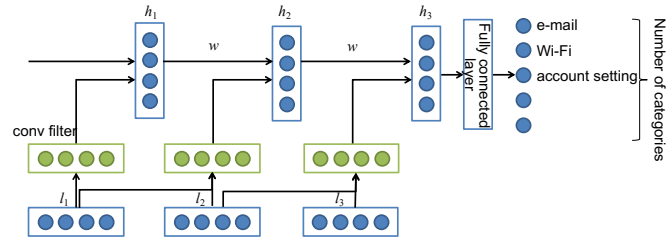

Figure 5: CNN-LSTM model. The input is first convolved, and fed to LSTM part. In the case of the second stage estimation, the user selected category is also used as input similar to LSTM model.

3.2.2 CNN-LSTM Model. In the past few years, deep convolutional neural network (CNN) has demonstrated state-of-the-art results for natural language processing tasks, such as text classification [14, 17] and sentiment analysis [24]. In these studies, word embedding is first applied, and then an embedding vector is used for the CNN. Inspired by these studies, we combine the LSTM model with CNN (CNN-LSTM) to achieve a remarkably accurate classification. Figure 5 shows our CNN-LSTM model architecture. First, the filtered class name sequence x'_i is fed as input similar to the simple LSTM. Then, each class name is converted into dense vector by embedding layer, and convolutional filters are applied. Thereafter, the feature maps are calculated and utilized as input to LSTM part. Here, the ReLU is used as an activation function for CNN layer. Finally, the fully connected layer with softmax activation function is employed to classify the problem category. The weight of the neural network is learned using Adam [15] by setting cross-entropy as the loss function.

3.3 Ground-Truth Label Acquisition

We use our call center logs to acquire the ground-truth label for each smartphone usage history. If users encounter a problem, several portion of them often call the customer support center. Then, the customer support staff records the user ID, time and problem category based on user's inquiry. By exploiting these data, we label the smartphone usage log recorded T hours prior to the inquiry. Among these data, we label the inquiry regarding a specific category as positive sample, and other categories as negative sample.

3.4 System Overview

Figure 6 shows our system overview. Our system consists of offline and online processing. For offline processing, first, smartphone usage and customer support call logs are aggregated on the basis of user ID u and timestamp t to construct a dataset of a supervised machine learning model. Second, the trained model is downloaded to each user's smartphone. For online processing, the usage log of smartphone is stored in each user's smartphone, and the features for machine learning are generated on basis of the smartphone usage log as background process. Third, the troubleshooting recommendation is automatically calculated using the features at a predefined time interval. Finally, if the probability exceeds the threshold, a troubleshooting is provided to a user.

4 EVALUATION

4.1 Offline Evaluation

To demonstrate the effectiveness of our application, we have conducted extensive offline evaluations. We also evaluated our application in the commercial service as online evaluation.

4.1.1 Dataset. We collected the smartphone usage and customer support call logs with user's consent from April to October, 2017. The number of records was several millions. First, we integrated the smartphone usage and customer support call logs based on user ID and timestamp. Then, we divided the data into two groups, that is, the data from April to September (dataset 1) and of October (dataset 2). Dataset 1 is further divided into training and validation data. The validation data are randomly sampled from 25% of dataset 1 and the remaining 75% is used as the training data. Meanwhile, dataset 2 is used as hold-out test data. The total number of records in dataset 1 and 2 is almost 50,000.

The number of top level categories is 20, whereas that of lower level categories is 68. The top level categories include *docomo mail*[5], *d ACCOUNT*[6], *Wi-Fi*, and *Google account setting* etc. Meanwhile, the lower level category consists of *Spam mail setting*, *Sending and receiving mail*, *Attachments* etc. for *docomo mail* category. More comprehensive category list is shown in Appendix A (Table 8).

4.1.2 Evaluation Setting. The parameter k for a predefined class name dictionary based on $TF \cdot IDF$ is set to 1,000. The number of layers for MLP varies from 2 to 4. The number of neurons for each MLP layer is set to (1,000, 500, 20) for two layers, (1,000, 500, 250, 20) for three layers and (1,000, 500, 250, 125, 20) for four layers

[5]This is our mobile e-mail service. See https://www.nttdocomo.co.jp/english/iphone/service/mail/

[6]This is our user registration program. See https://www.nttdocomo.co.jp/english/support/procedure/change_release/user_info/

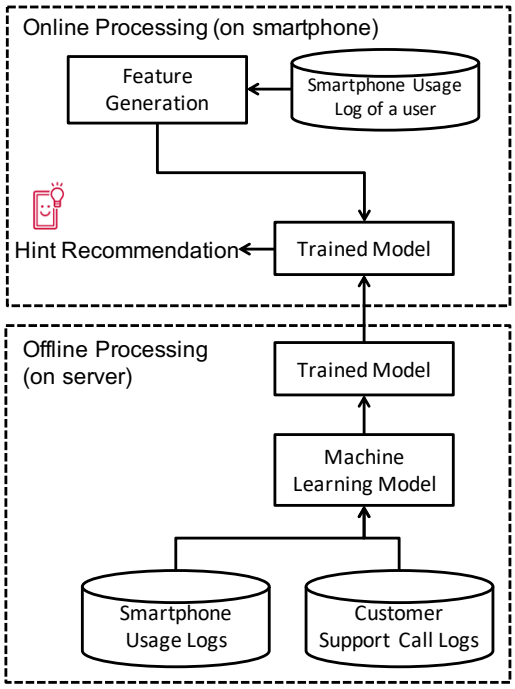

Figure 6: System overview. Our system consists of two parts: offline processing and online processing. In offline processing, a machine learning model is trained using smartphone usage and customer support call log. In online processing, the automatic user problem detection is executed at a predefined interval as background process. If the system detects a user has a problem, the system provides a troubleshooting for the user.

regarding the first stage estimation. Moreover, we evaluated the bag-of-words features with normalization, set the time parameter $T = 1$ hour. In the LSTM model, the hidden layer dimension is set to 100. In the CNN-LSTM model, the number of filters is 32 and kernel size is 3. In the case of the second stage estimation, the input size is set to 1,020 and the output size is set to 68.

4.1.3 Baseline Method. XGBoost [8] is selected as the baseline classifier because the classifier is extensively used in many classification problems [29, 34], although XGBoost is not supported in TensorFlow Lite. The features for XGBoost are similar to that of the MLP model described in Section 3.1.

4.1.4 Classification Performance. We used *accuracy@n* as the offline evaluation metric. The metric is widely used in the existing studies regarding recommendation [28, 30, 35]. The evaluation procedure is as follows:

(1) We calculate the probability of each category for all test data using each model.
(2) We form a ranked list by sorting all of these categories according to their probabilities, and top-n scored categories are recommended.
(3) If the ground-truth category appears in the top-n recommendation list, the test case is counted as correct.

Model	Number of layers of MLP	acc@3
XGBoost	-	0.590
MLP(w/o dropout)	2	0.591
MLP(w/o dropout)	3	0.589
MLP(w/o dropout)	4	0.584
MLP(w/dropout)	2	0.598
MLP(w/dropout)	3	0.593
MLP(w/dropout)	4	0.585
LSTM	-	0.605
CNN-LSTM	-	0.609

Table 2: Classification performance results for the first stage estimation c_1. The best performance model is CNN-LSTM, and the two-layer MLP with dropout is the best among MLP models which can be implemented on Android.

Model	Number of layers	acc@3
XGBoost	-	0.87
MLP	2	0.897
MLP	3	0.896
MLP	4	0.897
CNN-LSTM	-	0.892

Table 3: Classification performance results for the second stage estimation c_2. We only evaluated the MLP models without dropout because the performance of MLP with and without dropout is almost the same in the first stage estimation.

We define *accuracy@n* as the percentage of users who had a category in their ground truth of the top-n recommended categories. In our commercial application, the maximum number of recommended categories is three, therefore we used *accuracy*@3 for our evaluation.

In December 2018, TensorFlow Lite only supports a fully connected layer in our proposed models. Therefore, the purpose of the evaluation is to compare the performance of MLP with other models. Table 2 and 3 list the *accuracy*@3 for the first and second stage estimation, respectively. As presented in Table 2, the best performance model is CNN-LSTM. In the MLP models which can be implemented by TensorFlow Lite, the two-layer MLP with dropout exhibits the best performance. Dropout is slightly effective for improving the performance. In our case, a slight tendency may exist where the more the number of layers of MLP is, the lesser its performance will be. For the second stage estimation, all deep learning models achieved almost the same performance which was better than the XGBoost model (See Table 3). Because the performance of MLP with and without dropout is almost the same in the first stage estimation, we only evaluate the MLP without dropout in the second stage estimation. Moreover, the MLP models achieve an accuracy of more than 80% which is the requirement.

In addition, we evaluated the effect of time width T for classification performance and compared the cases of $T = 1, 3, 6$. Table 4 shows the results. A very slight tendency may exist where the more the time width is, the more its performance will be for the

T	Number of layers	acc@3 (first stage)	acc@3 (second stage)
1	2	0.591	0.897
	3	0.589	0.896
	4	0.584	0.897
3	2	0.595	0.903
	3	0.587	0.901
	4	0.580	0.897
6	2	0.597	0.899
	3	0.590	0.897
	4	0.583	0.902

Table 4: Classification performance results with different T. A very slight tendency may exist where the more the time width is, the more its performance will be for the first stage estimation.

Model	Inference Time (ms)
CNN-LSTM	374.1
MLP	10.4

Table 5: Measurement results of inference time. The MLP model is much faster than the CNN-LSTM model.

first stage estimation, whereas we cannot find a tendency for the second stage estimation. From these results, we selected $T = 1$ for production setting to reduce the computing resource usage.

4.1.5 Inference Performance on Smartphone. In real-world application, the inference time must be relatively short because our application needs instant troubleshooting display to customers who have a problem and provides a troubleshooting before a user calls a customer support. Thus, we evaluated the inference time on the smartphone. In addition, if the resources usage of smartphone such as battery consumption, CPU and memory usage is high, it cannot be acceptable for commercial service if its inference time is relatively short. Therefore, we investigated the resources usage as inference performance on a smartphone.

We used Nexus 5X (Android 8.1.0) and MLP model implemented using TensorFlow Lite 0.1.7. CNN-LSTM model has been also implemented by using the TensorFlow Mobile[7] 1.8.0 for reference. The smartphone usage log is stored in the database of Android. Each inference was executed at an interval of 10 minutes, the total number of execution was 18 and the number of MLP layers was 3.

We measured the time for preprocessing (feature generation) and the time for inference using a machine learning model, respectively. The mean preprocessing time is 24 ms. Table 5 presents the mean inference time for each model. Because the MLP model is more simple than the LSTM model, the inference time of the MLP model is shorter than the CNN-LSTM model. Moreover, the inference time of the MLP model is 10.4 ms, thus, this model is acceptable in the real-world application.

We evaluated the resources usage regarding battery consumption, CPU and memory usage. Table 6 shows the measurement

Battery Consumption (mAh/times)	CPU Usage Rate (%)	Memory Usage Rate (%)
0.02205	5.63	1.04

Table 6: Measurement results of resource usage. These results indicated that the resources usage is enough low.

rank	class name
1	jp.buffalo.aoss.activity.AossStartActivity
2	com.android.settings.wifi.KlausAdvancedWifiSettings
3	com.android.captiveportallogin.CativePortalLoginActivity
4	com.android.settings.wifi.RakurakuWifiSettings.RakurakuWifiKlausW\Settings
5	com.android.settings.wifi.KausAdvancedWifiSettings3
6	com.lge.wifisettings.WifiSetttings
7	com.lge.wifisettings.activity.WifiSettingsActivity
8	com.android.settings.Settings$WifiSettingsActivity
9	com.lge.launcher3.LauncherExtension
10	com.android.settings.SubSettings$SaveAccessPointSubSettings

Table 7: Example of the $TF \cdot IDF$ ranking regarding the Wi-Fi setting category. The ranking clearly reflects the problem category.

results. The average battery consumption for each inference was 0.02205 mAh. The total battery of Nexus 5X is 2,700 mAh. If our application can use 10% of the total battery, the total number of execution is $2,700 \times 0.1/0.02205 = 12,244.89$. Thus, if the inference is executed every 10 seconds, we can use the smartphone for $12,244.89 \times 10 = 122,448.9 seconds = 34.01 hours$. The average CPU usage was 5.63% and average memory usage was 1.04%. Note that we measured the performance when only our application was executed. Thus, a difference from the same performance is possible in real-world use. However, from these results, we confirmed that the resources usage is low.

4.1.6 Qualitative Evaluation. Apart from the quantitative evaluation of classification performance and inference time, we also conducted a qualitative evaluation to verify the effectiveness of applying $TF \cdot IDF$ to the class name dataset. We calculated $TF \cdot IDF$ for the data annotated as *Wi-Fi* category by call center logs as an example. Table 7 shows the result of the $TF \cdot IDF$ ranking of the *Wi-Fi* category data. In the ranking, many class names contain the word *wifi*. The top-ranked class name is regarding AOSS which is a Wi-Fi connectivity support system used in Japan[8]. In addition, third-ranked class name is related to captive portal, which is used for a web portal to log in the Wi-Fi service[9]. From these examples, we can see that the $TF \cdot IDF$ can capture the characteristics of smartphone usage category.

4.2 Online Evaluation

Our rule-based troubleshooting recommendation has been launched since 2016[10], and installed by over a few million users in Japan. After that, machine learning-based troubleshooting recommendation, which is proposed in this paper, has been released in January, 2019. In the actual service, a feedback, *Did this troubleshooting solve your*

[7] TensorFlow Mobile supports LSTM architecture. However, Google announced that TensorFlow Mobile will be deprecated in early 2019. See https://www.tensorflow.org/lite/tfmobile/

[8] https://en.wikipedia.org/wiki/AOSS

[9] https://en.wikipedia.org/wiki/Captive_portal

[10] https://www.nttdocomo.co.jp/service/osusume_hint/ (Japanese Only)

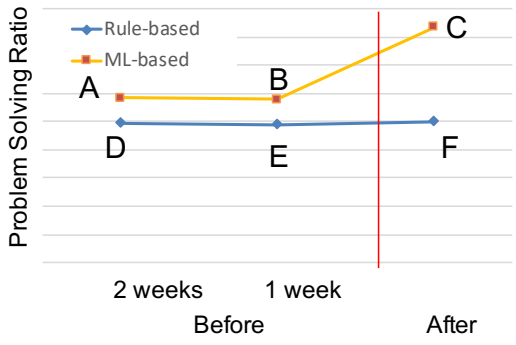

Figure 7: Result of the online evaluation. PSR of ML-based approach is higher than that of rule-based.

problem?, will be displayed after the troubleshooting is finished, and then a user can select *Yes* or *No*. We use the ratio of *Yes*, referred to as problem solving ratio (PSR), to measure the effectiveness of the system in online evaluation. We compared machine learning approach with rule-based approach as an online evaluation. In the work of Althoff et al. [2], they exploited that a user uses a specific function (e.g., use social network connection) or not to identify the effect of the specific function using difference-in-difference analysis [16]. Inspired by the work in [2], we use the difference-in-difference analysis to verify the effect of the proposed system in online evaluation.

Figure 7 shows the result of PSR for rule-based and machine learning(ML)-based approach. Due to the sensitivity of the data, we did not indicate the absolute PSR in Figure 7. From the data before the ML-based application released (i.e., the data points of A, B, D and E), we can confirm that the common trends assumption is met because line AB and DE in Figure 7 are approximately parallel. Then, we can calculate the effect of ML-based approach by $(C-B)-(F-E)$. The relative improvement of PSR is approximately 30% by ML-based approach.

5 LESSONS LEARNED AND DISCUSSION

In this section, we summarize our lessons learned from this project.
Heterogeneous Data Integration: Labeling the ground truth to raw class name sequence is very difficult. To overcome this problem, we focused on the characteristics of customer support call logs. Similar strategy is used in the work of Pan et al. [20]. From our evaluation, our strategy is validated using a large amount of smartphone usage and customer support logs. Thus, integrating heterogeneous logs can be an effective method for labeling the raw data that humans cannot interpret. Meanwhile, the performance of the first stage estimation, which is approximately 60% with all models, needs improvement. One possible solution is improving the ground truth labeling. If a user has a problem but cannot immediately call a customer support, our ground truth labeling strategy will fail. For example, if a user, who is on the way to work, has a problem with smartphone usage just before arriving at the office, the user cannot immediately call a customer support but may call during his or her break time. To solve this problem, it may be effective to utilize the user's context (e.g. location, time of the day) for ground truth labeling and inferenece.
Combination of Machine Learning and User Interface: In our preliminary experiment, we directly estimated the lower level category c_2 from the smartphone usage sequence x_i. This approach failed and the maximum performance ($accuracy@3$) was approximately 30% only. Therefore, we defined the user problem detection problem as the two-stage classification problem defined in Section 2. From this result, we designed the user interface shown in Figure 2 and implemented a machine learning model that exploits the user's category selection information. One of the most important lessons learned from this project is that the combination of machine learning algorithm and user interface is an effective technique for real-world applications because these machine learning algorithm and user interface are strongly bound in the recommender system [3]. In some cases, user interface relax the difficulties of machine learning problem.

6 RELATED WORK

In this section, we review related work on the analysis of smartphone application usage. Related work can be categorized into two groups, namely, (1) next app prediction and app recommendation from smartphone usage logs and (2) user state estimation from smartphone usage logs.

6.1 Next App Prediction and App Recommendation

The next app prediction refers to the task of predicting the subsequent app that will be used, whereas the app recommendations is the task to recommend a user to install new apps. In the next app prediction task, the app usage history (e.g., app usage sequence, latest used app and category etc.) is the basic feature [4, 5, 25, 32], and the installed app or impression [9] in the app market is used as a fundamental feature for app recommendation. In both tasks, various types of additional features are used (refer to survey [7]). These features are (1) location features [4, 5, 36], (2) temporal features [4, 5], and (3) smartphone status (battery level, accelerometer and proximity sensor etc.) [25]. Böhmer et al. [5] proposed a mobile app recommendation framework referred to as *AppFunnel*. They investigated the effect of app popularity, time and location and exploited the results for app recommendation. Shin et al. [25] collected smartphone usage logs including app use, accelerometer, environment-related sensors and GPS etc. from smartphones. They model the context of mobile app use and exploit the results for app usage prediction. Xu et al. [32] used a considerably abstract feature, that is, user preference, to achieve high prediction accuracy.

6.2 User State Estimation from Smartphone Usage Log

Another related work is user state estimation such as demographic [18], user trait [23], mental health [22, 27, 33], and cognitive alertness [1, 21]. For example, Pielot et al. [21] proposed a method of detecting boredom from demographics and mobile phone usage. They used self-reported boredom feelings as ground truth, and mobile usage pattern, such as SMS usage and battery status, is used as

features for machine learning model. In these studies, the smartphone app usage history is used as features for machine learning model, and the target for each study varies.

To the best of our knowledge, our study differs from existing works owing to the following reasons:

- The prediction target, user problem, and method of the ground truth labeling in our work are unique.
- No existing study implemented an on-device inference for a real-time prediction.

7 CONCLUSION

In this paper, we described the challenges encountered in the real-time on-device smartphone operation troubleshooting recommendation. First, we formulated a smartphone operation troubleshooting recommendation as the classification problem. Then, we presented the MLP model to be implemented on an Android smartphone and evaluated the performance of the proposed model, in comparison with the extensively used machine learning model (i.e., XGBoost) and advanced deep learning model (i.e., CNN-LSTM). We adopted the MLP model because the current version of the TensorFlow library only supports a limited number of deep learning architectures including MLP. The offline evaluation results showed that the MLP model meets the two requirements for commercial application. Hence, we launched our application in our commercial service since January, 2019. After the service released, we also conducted an online evaluation in the commercial service. Online evaluation result showed that machine learning based approach outperforms the rule-based approach by approximately 30% regarding PSR. To achieve a more accurate troubleshooting recommendation in the future, we may adopt the CNN-LSTM model to predict the user problem category if the LSTM layer is supported by TensorFlow Lite.

According to the recent survey of on-device machine learning inference [12] and official web page of TensorFlow Lite, the current applications are mainly image recognition, natural language processing such as Smart Reply [13], and audio recognition. We believe that our project opens the door to novel application domain for on-device machine learning.

ACKNOWLEDGMENTS

We would like to thank Koichi Moriyama who mainly supported our R&D project, Satoru Morota, Nozomi Matsumoto, Ryo Babazono, Yuma Kuribayashi, Teppei Koga, Ryoko Hayashi and Shodai Muta for leading this R&D project to commercial service from business side. We would also like to thank Yoshitaka Inoue, Noriaki Hirokawa and Shigeki Tanaka for proofreading. Finally, we would like to thank Prof. Jure Leskovec at Stanford University for improving the paper structure.

REFERENCES

[1] Saeed Abdullah, Elizabeth L. Murnane, Mark Matthews, Matthew Kay, Julie A. Kientz, Geri Gay, and Tanzeem Choudhury. 2016. Cognitive Rhythms: Unobtrusive and Continuous Sensing of Alertness Using a Mobile Phone. In *Proceedings of UbiComp '16*. 178–189.

[2] Tim Althoff, Pranav Jindal, and Jure Leskovec. 2017. Online Actions with Offline Impact: How Online Social Networks Influence Online and Offline User Behavior. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM '17)*. ACM, New York, NY, USA, 537–546. https://doi.org/10.1145/3018661.3018672

[3] Xavier Amatriain and Justin Basilico. 2016. Past, Present, and Future of Recommender Systems: An Industry Perspective. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM, New York, NY, USA, 211–214. https://doi.org/10.1145/2959100.2959144

[4] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting The Next App That You Are Going To Use. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*. ACM, New York, NY, USA, 285–294. https://doi.org/10.1145/2684822.2685302

[5] Matthias Böhmer, Lyubomir Ganev, and Antonio Krüger. 2013. AppFunnel: A Framework for Usage-centric Evaluation of Recommender Systems That Suggest Mobile Applications. In *Proceedings of the 2013 International Conference on Intelligent User Interfaces (IUI '13)*. ACM, New York, NY, USA, 267–276. https://doi.org/10.1145/2449396.2449431

[6] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*. Springer, 177–186.

[7] Hong Cao and Miao Lin. 2017. Mining smartphone data for app usage prediction and recommendations: A survey. *Pervasive and Mobile Computing* 37 (2017), 1–22.

[8] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2939672.2939785

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 7–10. https://doi.org/10.1145/2988450.2988454

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep Sparse Rectifier Neural Networks.. In *Aistats*, Vol. 15. 275.

[11] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[12] Andrey Ignatov, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. 2019. AI Benchmark: Running Deep Neural Networks on Android Smartphones. In *Computer Vision – ECCV 2018 Workshops*, Laura Leal-Taixé and Stefan Roth (Eds.). Springer International Publishing, Cham, 288–314.

[13] Anjuli Kannan, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado, Laszlo Lukacs, Marina Ganea, Peter Young, and Vivek Ramavajjala. 2016. Smart Reply: Automated Response Suggestion for Email. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 955–964. https://doi.org/10.1145/2939672.2939801

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of EMNLP*. Association for Computational Linguistics, 1746–1751.

[15] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[16] Michael Lechner et al. 2011. The estimation of causal effects by difference-in-difference methods. *Foundations and Trends® in Econometrics* 4, 3 (2011), 165–224.

[17] Kathy Lee, Ashequl Qadir, Sadid A. Hasan, Vivek Datla, Aaditya Prakash, Joey Liu, and Oladimeji Farri. 2017. Adverse Drug Event Detection in Tweets with Semi-Supervised Convolutional Neural Networks. In *Proceedings of the 26th International Conference on World Wide Web (WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 705–714. https://doi.org/10.1145/3038912.3052671

[18] Eric Malmi and Ingmar Weber. 2016. You Are What Apps You Use: Demographic Prediction Based on User's Apps.. In *ICWSM*. 635–638.

[19] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Vol. 39. Cambridge University Press.

[20] Lujia Pan, Jianfeng Zhang, Patrick P.C. Lee, Hong Cheng, Cheng He, Caifeng He, and Keli Zhang. 2017. An Intelligent Customer Care Assistant System for Large-Scale Cellular Network Diagnosis. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17)*. ACM, New York, NY, USA, 1951–1959. https://doi.org/10.1145/3097983.3098120

[21] Martin Pielot, Tilman Dingler, Jose San Pedro, and Nuria Oliver. 2015. When Attention is Not Scarce - Detecting Boredom from Mobile Phone Usage. In *Proceedings of UbiComp '15*. 825–836.

[22] Akane Sano and Rosalind W Picard. 2013. Stress recognition using wearable sensors and mobile phones. In *Affective Computing and Intelligent Interaction (ACII), 2013 Humaine Association Conference on*. IEEE, 671–676.

[23] Suranga Seneviratne, Aruna Seneviratne, Prasant Mohapatra, and Anirban Mahanti. 2014. Predicting User Traits from a Snapshot of Apps Installed on a Smartphone. *SIGMOBILE Mob. Comput. Commun. Rev.* 18, 2 (June 2014), 1–8. https://doi.org/10.1145/2636242.2636244

[24] Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*

(SIGIR '15). ACM, New York, NY, USA, 959–962. https://doi.org/10.1145/2766462.2767830

[25] Choonsung Shin, Jin-Hyuk Hong, and Anind K. Dey. 2012. Understanding and Prediction of Mobile Application Usage for Smart Phones. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing (UbiComp '12)*. ACM, New York, NY, USA, 173–182. https://doi.org/10.1145/2370216.2370243

[26] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[27] Rui Wang, Min S. H. Aung, Saeed Abdullah, Rachel Brian, Andrew T. Campbell, Tanzeem Choudhury, Marta Hauser, John Kane, Michael Merrill, Emily A. Scherer, Vincent W. S. Tseng, and Dror Ben-Zeev. 2016. CrossCheck: Toward Passive Sensing and Detection of Mental Health Changes in People with Schizophrenia. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '16)*. ACM, New York, NY, USA, 886–897. https://doi.org/10.1145/2971648.2971740

[28] Weiqing Wang, Hongzhi Yin, Ling Chen, Yizhou Sun, Shazia Sadiq, and Xiaofang Zhou. 2015. Geo-SAGE: A Geographical Sparse Additive Generative Model for Spatial Item Recommendation. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1255–1264. https://doi.org/10.1145/2783258.2783335

[29] Zheng Wang, Kun Fu, and Jieping Ye. 2018. Learning to Estimate the Travel Time. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. ACM, New York, NY, USA, 858–866. https://doi.org/10.1145/3219819.3219900

[30] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. 2016. Learning Graph-based POI Embedding for Location-based Recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, New York, NY, USA, 15–24. https://doi.org/10.1145/2983323.2983711

[31] Qiang Xu, Jeffrey Erman, Alexandre Gerber, Zhuoqing Mao, Jeffrey Pang, and Shobha Venkataraman. 2011. Identifying Diverse Usage Behaviors of Smartphone Apps. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference (IMC '11)*. ACM, New York, NY, USA, 329–344. https://doi.org/10.1145/2068816.2068847

[32] Ye Xu, Mu Lin, Hong Lu, Giuseppe Cardone, Nicholas Lane, Zhenyu Chen, Andrew Campbell, and Tanzeem Choudhury. 2013. Preference, Context and Communities: A Multi-faceted Approach to Predicting Smartphone App Usage Patterns. In *Proceedings of the 2013 International Symposium on Wearable Computers (ISWC '13)*. ACM, New York, NY, USA, 69–76. https://doi.org/10.1145/2493988.2494333

[33] Naoki Yamamoto, Keiichi Ochiai, Akiya Inagaki, Yusuke Fukazawa, Masatoshi Kimoto, Kazuki Kiriu, Kouhei Kaminishi, Jun Ota, Tsukasa Okimura, Yuri Terasawa, and Takaki Maeda. 2018. Physiological Stress Level Estimation Based on Smartphone Logs. In *2018 Eleventh International Conference on Mobile Computing and Ubiquitous Network (ICMU)*. IEEE, 1–6.

[34] Peng Ye, Julian Qian, Jieying Chen, Chen-hung Wu, Yitong Zhou, Spencer De Mars, Frank Yang, and Li Zhang. 2018. Customized Regression Model for Airbnb Dynamic Pricing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '18)*. ACM, New York, NY, USA, 932–940. https://doi.org/10.1145/3219819.3219830

[35] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. 2013. LCARS: A Location-content-aware Recommender System. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '13)*. ACM, New York, NY, USA, 221–229. https://doi.org/10.1145/2487575.2487608

[36] Donghan Yu, Yong Li, Fengli Xu, Pengyu Zhang, and Vassilis Kostakos. 2018. Smartphone App Usage Prediction Using Points of Interest. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 174 (Jan. 2018), 21 pages. https://doi.org/10.1145/3161413

A CATEGORY LIST

Figure 8 shows comprehensive categories used in our evaluation.

Top Level Category	Lower Level Category
docomo mail	Send / Receive message
	Spam filter setting
	Silent mode setting
	Send / Receive attachments
	Wi-Fi setting
	Manage contacts
	Create new message
	Auto sorting
	Delete message
Phone	Make / Receive call
	Silent mode setting
	Block phone numbers
	Voice mail
	Call History
	Volume control
dmenu	Customer Support
	My menu
	Application and procedures
	Usage history
	Content payment services
SMS	Create / Send new message
	Block numbers
	Delete message
Contacts	Add contacts
	docomo Cloud
	Delete contancts
Wi-Fi	Connect home Wi-Fi
	docomo Wi-Fi
d ACCOUNT	Issue/Confirmation
	Change
Play store	App download
	How to use
	App uninstall
Camera	Take photo
	Photo storage
	Record video
Screen Lock	Settings
	Unlock
Character Input	Keyboard setting
Google Account	Account setting
Internet	Data usage
	Unable to connect
Browser	Display
	Bookmarks

Table 8: List of target categories. Due to business constraints, several categories are not indicated in this table.