Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 2798 – 2816

doi:10.3906/elk-1811-196

Cloud-supported machine learning system for context-aware adaptive M-learning

Muhammad ADNAN*, Asad HABIB, Jawad ASHRAF, Shafaq MUSSADIQ
Institute of Computing, Faculty of Physical & Numerical Sciences,
Kohat University of Science and Technology, Kohat, Pakistan

Received: 30.11.2018 • **Accepted/Published Online:** 29.03.2019 • **Final Version:** 26.07.2019

Abstract: It is a knotty task to amicably identify the sporadically changing real-world context information of a learner during M-learning processes. Contextual information varies greatly during the learning process. Contextual information that affects the learner during a learning process includes background knowledge, learning time, learning location, and environmental situation. The computer programming skills of learners improve rapidly if they are encouraged to solve real-world programming problems. It is important to guide learners based on their contextual information in order to maximize their learning performance. In this paper, we proposed a cloud-supported machine learning system (CSMLS), which assists learners in learning practical and applied computer programming based on their contextual information. Learners' contextual information is extracted from their mobile devices and is processed by an unsupervised machine learning algorithm called density-based spatial clustering of applications with noise (DBSCAN) with a rule-based inference engine running on a back-end cloud. CSMLS is able to provide real-time, adaptive, and active learning support to students based on their contextual information characteristics. A total of 150 students evaluated the performance and acceptance of CSMLS for a complete academic semester, i.e. 6 months. Experimental results revealed the threefold success of CSMLS: extraction of students' context information, supporting them in appropriate decision-making, and subsequently increasing their computer programming skills.

Key words: Artificial intelligence, machine learning, DBSCAN, intelligent system, mobile learning, cloud computing, adaptive learning, computer programming

1. Introduction

It has been recognized that the computer programming abilities of learners increase if they are encouraged to solve problems that they encounter in the real world [1]. In order to provide suitable and tailored programming exercises, detailed learning context data must be first extracted from learners' mobile devices. In comparison to other subjects, computer programming is more difficult and challenging for learners [2]. Factors that make computer programming difficult to learn include the inherent sensitivity of programming languages, students' perceptions and motivations for learning programming, creativity, tenacity, and specialized domain-specific concepts [3]. Moreover, for a better understanding of computer programming, its practical applications in real-life scenarios must be imparted to learners. Keeping the above-mentioned facts in mind, we cannot use context data to recommend appropriate computer programming learning content on learners' mobile devices in the same way that learning content is delivered while learning English, history, medicine, etc. Environmental context data are useful in framing real-world programming exercises for learners and then can be delivered to them on their mobile devices at appropriate learning times. Learning computer programming is in stark contrast

*Correspondence: adnan@kust.edu.pk

to other learning fields where learning is possible at any time and anywhere. It requires thinking processes, silence, analytical abilities, patience, and persistence [4]. In learning computer programming, understanding and remembering concepts, terminologies, and language constructs is necessary, and applying the acquired knowledge in writing computer programs that solve real-world problems is challenging and needs a significant amount of practice [5].

In this research, we propose a novel cloud-supported machine learning system (CSMLS) for context-aware and adaptive learning of computer programming. CSMLS generates differential learning paths for individual learners from context data gathered through their respective mobile devices, which are then processed by a machine learning algorithm on the back-end cloud. These learning paths subsequently recommend solving real-world computer programming problems at an appropriate time based on performance states of the learners.

Supporting learners in the learning process based on proper contextual information is a challenging issue [6]. Contextual information includes the learner's background knowledge, learning content, location, and time and duration of learning process. A context-aware learning system facilitates students in adaptive learning unrestricted by temporal or spatial factors. With the invention of various types of mobile devices, it is now possible to reveal the proper context of learners and subsequently support them in adaptive learning (u-learning, m-learning) processes [7, 8]. A number of research studies have investigated technologies such as GPS [9, 10], RFID [11], mobile devices [12], QR code interfaces [13], NFC [14, 15], Bluetooth [16], and virtual keyboards [17] to explore the characteristics of students' learning contexts.

For the last decade, research studies have been conducted that first explore the proper context of learners and then guide them accordingly through Internet-connected mobile devices (e.g., smartphones and tablets). For example, Jorge et al. developed a context-aware system that simulated students' active learning [18]. This context-aware system used five context dimensions, namely students, learning objects, time, location, and learning activity. To deliver effective learning resources, the system used reasoning capability based on context-based ontology. Smartphone-enabled technologies such as NFC (near field communication), QRCODE (quick response code), GPS (Global Positioning System), and BLE (Bluetooth Low Energy) were used to discover learning objects and learner locations in a learning context. The entire system was based on client-server architecture.

Likewise, many other studies have investigated methods for providing relevant information to mobile users taking into account their contextual information. The architectures used in these studies were mainly based on a client-server architecture, where most of the processing is done at the web-server side [19, 20].

The drawback of the web-server approach is that separate processing and memory resources are dedicated to compute the learning path of each learner based on his or her learning context information [21]. Moreover, web servers need continuous hardware and software installation, configuration, monitoring, and upgrading [22]. Compared to computer clouds, web servers perform poorly in scalability, power consumption, and resource utilization [23]. In this paper, we present a cloud-supported mobile learning system (CSMLS) that uses a machine learning algorithm for exploring different learning paths of learners having similar learning behaviors, context information, and learning preferences. The main objective of CSMLS is to encourage learners in solving those programming problems that they encounter in the real world, thus making their learning riveting and engaging.

Mobile devices (smartphones and tablets), when connected to the cloud, can upsurge their processing power and storage capacity [24]. The continued improvement of cloud computing technologies has enabled ubiquitous, real-time, and on-demand access to computing resources (e.g., applications, platforms, infrastructure)

with nominal management efforts [25]. Currently, many smartphone applications related to health, entertainment, communication, education, and safety are using complementary features of clouds to mix and match their needs [26, 27].

The prevalent M-learning studies have investigated methods for knowing learners' learning behaviors and preferences by analyzing learning records at the back-end web servers and then facilitating them accordingly. Some M-learning studies target the enhancement of students' English listening, speaking, reading, and writing skills [28]. M-learning systems like those in [29, 30] facilitate students to learn on the move, e.g., at zoos, museums, parks, colleges, and universities campuses. These types of M-learning systems first consider the learner's location, time, preferred learning contents, and learning behavior and then provide relevant and adaptive learning contents.

For the purpose of optimizing the programming language skills of learners, this study was exercised to establish learning analytics based on environmental context data and students' knowledge and behavior to provide applied and personalized computer programming learning material. This adaptive learning mechanism is distinctive from previous context-aware recommendation systems that store and use limited context information to recommend appropriate learning resources to individual students. For a comprehensive learning mechanism, a complete statistical model of data about learners and their context should be built. This statistical model includes detailed information about learners including learners' learning behavior; the number of places they visit (location context) on a daily, weekly, or monthly basis; learning time and duration of learning time (time context); learning preferences; learning performance; learning support; and preferred learning content.

This study executed a clustered-based machine learning algorithm called DBSCAN on learners' context data stored in the Google Firebase database cloud to discover the relationships among learning context variables. Based on context variables' characteristics explored by DBSCAN, the system generates different learning patterns for individual students to enhance their computer programming efficiency. For exploratory analysis of students' context data, the system relied on large-scale fine-grained data generated by learners during the learning process. The context data are procured from learners' mobile devices and stored in the Firebase cloud. The proposed system can be used to guide learners in learning computer programming subjects that need ample practice and are practical in nature. The system relies on large-scale granular context data of learners; for this reason, it is able to represent a more accurate and reliable picture of a learner's learning context and provide appropriate learning content and guidance.

2. Literature review

The purpose of this study was to enhance the computer programming skills of learners by encouraging them to solve the programming problems that they encounter in real life. For this purpose, we established a learning environment where learners could practice and solve programming problems using their mobile devices such as smartphones and tablets.

The following sections present generic context-aware studies, applications, and theories that are explored by researchers in various branches of learning.

2.1. Context-aware learning

The growing use of mobile devices and their sister technologies have made it possible to discover learning styles and behaviors of students by mining their learning context information [31, 32]. The M-learning context is where learning takes place, with temporal constraints, graphical user interface (GUI) interaction details, and

the learner's current knowledge, preferences, and performance states [33]. The results of experimental studies show that M-learning assists students in various learning settings by providing them suitable content at the right time and the right place.

Increasingly, English, Japanese, and Chinese vocabulary has been examined by a few context-aware ubiquitous M-learning systems [34, 35]. Context-aware learning has also been applied in the spheres of road safety, health services, education, physical exercise, etc. [36]. Hwang et al. explored the learning context through RFID-enabled technologies for providing appropriate learning assistance to learners in completing their learning activities [37].

2.2. Machine learning in context-aware M-learning

Machine learning (ML), which evolved from artificial intelligence and computational learning theory, gives computers the ability to learn without explicitly being programmed [38]. ML algorithms highly rely on data patterns to make clusters, classifications, and data-driven decisions. With the advancements in wireless communication technologies, mobile devices, and localizing/tracking sensors it is now possible to mine and store large-scale information generated from learning settings in the back-end cloud [39]. This information could be used by ML algorithms to discover new insights about how a learner learns in particular learning settings.

Very few studies have used ML techniques to create a meaningful learning context model of students and to provide context-aware learning material. Krause et al. used wearable sensors to gather and estimate students' context qualities such as location, learning activities, learning schedule, and ambient context information to determine a student's current learning state in order to configure learning settings on mobile devices proactively [40]. They reported that predefined learning paths and context-sensitive responses are not appropriate in mobile learning settings and ML techniques can successfully reveal context-aware preferences of the individual learner.

A predictive modeling approach called decision tree (DT) learning was used in [41] to predict the performance of students based on information collected from an online learning management system (LMS) at the end of the semester. The learning context information gathered from the LMS included students' attendance, quiz results, assignments, submissions, etc. The study suggested that ML techniques such as decision trees can be used to predict students' learning performances in the future and could help reduce dropout rates of weaker students.

ML techniques are successfully applied in other fields, including stock market prediction, retail sales, bioinformatics, clinical medicine, and counter-terrorism. Webb et al. pointed out key challenges in modeling students' behavior in modern educational settings and how these challenges have hindered ML techniques in modeling the learning behavior of students [42]. This study brought to the surface four critical issues that are limiting the real-world application of modeling students' educational settings and how corresponding attempts are trying to overcome them. The four issues addressed were the lack of a large dataset representing students' learning behavior, the lack of labeled data, concept drift, and computational complexity. The authors optimistically concluded that advancement in modern technologies such as the World Wide Web (WWW), mobile devices, mature users' feedback systems, and learning management systems will be able to provide fine-grained users' context data.

2.3. Mobile cloud computing

The purpose of mobile cloud computing (MCC) is to enable users to execute resource-hungry rich mobile applications on a plethora of mobile devices. MCC provides unrestricted processing, functionality, and storage

to a multitude of mobile devices based on a pay-as-you-use principle. Cloud resources are offered to users based on multiple categories including infrastructure as a service (IaaS) [43], e.g., Amazon Web Services, Microsoft Azure, and Google Compute Engine; platform as a service (PaaS) [44], e.g., Heroku, Google ML Engine, IBM Bluemix, and SaleForce; software as a service (SaaS) [45], e.g., Evernote, Gmail, Facebook, Office 365, and Google Apps; and back-end as a service (BaaS) [46], e.g., Parse and Google Firebase. Our proposed system uses Google Firebase for real-time database support, user authentication, remote configuration, performance monitoring, and users' analytics. Google Firebase is further integrated with Google Cloud ML Engine. Students' learning context data are stored at the Firebase back-end whereas machine learning processing is performed by Google Cloud ML Engine.

To enable and facilitate learners in learning computer programming language using their mobile devices and to overcome the limitations and disadvantages of previous studies, we developed a cloud-supported machine learning system (CSMLS) for enhancing real-world programming skills of students. CSMLS comprises three operational layers for its working, namely a context acquisition layer, context analysis layer, and learning path generation layer. In summary, the purpose of this study was manifold: to assist learners in solving the real-world programming problems that they encounter on a daily basis; to provide personalized programming learning contents to learners based on their performances and preferences; to explore location- and time-based learning characteristics of learners and provide them with relevant programming problems and material; to maintain and update learning analytics of learners used by the ML algorithm; to discover learners' learning behavior patterns and split students into a set of learning categories; to explore learning patterns of learners using a rule-based inference engine; and to provide suitable learning support that makes learning programming engaging and interesting.

3. Architecture of cloud-supported machine learning system

This study used the Google Firebase cloud and Google Cloud ML Engine for providing data storage and data mining services. These clouds were integrated with learners' smartphones for retrieving and processing their contextual information. The operational mechanism of the CSMLS architecture comprises three layers: a context acquisition layer, context analysis layer, and learning path generation layer, as shown in Figure 1.

The context acquisition layer uses an Android client app for determining and capturing the learner's location, preferences, and performance. In order to recommend suitable programming content and exercises to different learners that encourage understanding and solving real-world programming problems, geographical information generated by learners' smartphones was first excavated by the DBSCAN ML algorithm running on the Google Cloud ML Engine. The output data generated by the DBSCAN algorithm form different clusters according to learners' geophysical information. The excavated data in the form of clusters are further integrated with other contextual information (preferences, performance, learning time, etc.) and processed by the rule-based inference engine in the learning path generation layer for generating learning paths for individual learners. These learning paths recommend adaptive and real-world programming content to learners on their smartphones, thus making learning programming more stimulating and exciting. A brief introduction to CSMLS working layers is given in the following subsections.

3.1. Context acquisition layer

The context acquisition layer uses GPS and geofencing techniques for location services facilitated by satellite signals, Wi-Fi hotspot IDs, and cell tower IDs. The learner's learning behavior data such as learning preferences,

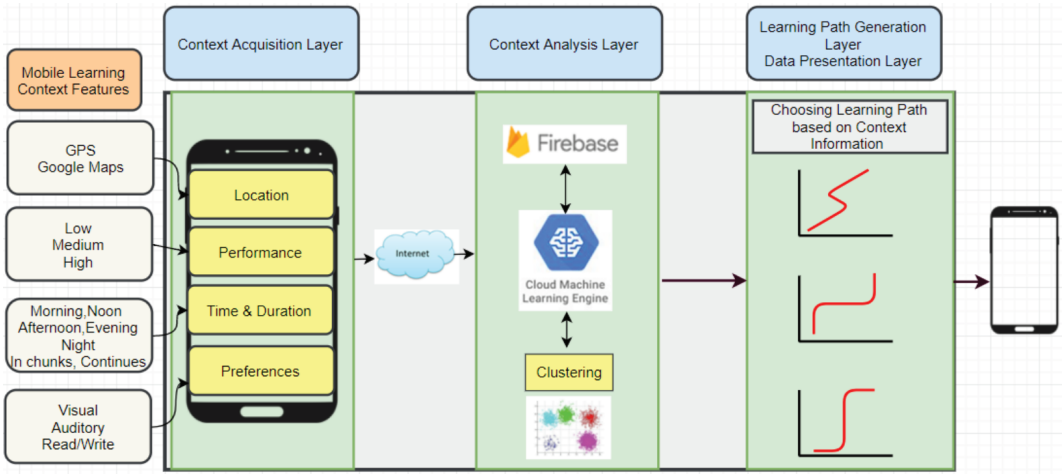


Figure 1. Architecture of the cloud-supported machine learning system (CSMLS).

learning time, learning duration, and learning performance are logged and transferred to a persistent Firebase database by the Android client app running on the learner's smartphone. The client app was developed for Android Lollipop 5.0 and later mobile operating systems to gather learning behavior information of learners. For providing real-world programming exercises to learners based on their learning behavior, it was important to first gather detailed fine-grain learning behavior information from learners' smartphones. For our experimental study, selected locations for learners included real-world places, e.g., classrooms, libraries, recreational playgrounds, and marketplaces, where the learners made frequent visits during their daily life activities. Performance information includes their current enactment in learning programming, which logically falls into low, medium, and high categories. Along with performance information, the learner's learning preferences in the form of visual, auditory, or read/write learning and learning time constraints including learning time and duration are logged on their smartphones and transferred to the Firebase cloud for learning context analysis. The following subsection explains the mobile learning attributes that form the learner's mobile learning model.

3.1.1. Mobile learning model

The mobile learning model embodies learner learning behavior while using a smartphone. In the mobile learning model, learners' learning attributes are represented by different variables formed during their learning activities. From time to time, the values of these learning variables are periodically synchronized between the learners' smartphones' client app and the Firebase cloud. The learning time variable is represented by timestamp T (T1, T2, T3…, Tn), where 'n' denotes the total number of learning activities in chunks in 24 h of time, i.e. morning, noon, afternoon, evening, and night. The values of the learning time variable represent the optimum time the learner is interested in learning computer programming. Connected with the learning time variable is learning time duration, represented by D (D1, D2, D3,…, Dm). The values of learning duration variable D represent learning duration time in increasing order. Real-world programming exercises are arranged according to a physical location and their difficulty levels. For example, (PL1, PL2, PL3,…, PLv) denotes programming exercises arranged in learning modules according to physical location and difficulty level. 'v' denotes the total number of programming modules as represented in Table 1.

Recommended learning contents are the programming exercises to be rendered on the learner's smart-

Table 1. Real-world programming exercises according to difficulty levels.

S.No	Real-world programming exercises	Related physical location	Difficulty level
1	PL1	Recreation ground	Difficulty low (DL)
2	PL1	Recreation ground	Difficulty high (DH)
3	PL2	Marketplace	Difficulty low (DL)
4	PL2	Marketplace	Difficulty medium (DM)
5	PL2	Marketplace	Difficulty high (DH)
6	PL3	Library	Difficulty low (DL)
7	PL3	Library	Difficulty medium (DM)
8	PL3	Library	Difficulty high (DH)
.....	.....	.....	.....

phone in the form of visual, auditory, and read/write material based on learners' preferences that were elicited from learners' smartphones during learning activities. In the mobile learning model, preferred learning contents are represented by a set denoted as LCPR (LP1, LP2, LP3,..., LPc), where 'c' denotes the total number of learning content types. In our study, we have limited the content types to visual, auditory, and read/write types. Background knowledge represents a learner's prior knowledge or prior performance for a particular programming exercise. For brevity, background knowledge is categorized into low, medium, and high levels. The background knowledge changes dynamically according to learner performance in daily, weekly, and monthly programming quizzes.

Our proposed system encourages learners to learn programming by undertaking real-world programming exercises based on their learning context information so that the learners are able to realize that unlike other subjects, programming needs active interaction, attention, and practice rather than learning in the passive mode.

3.2. Context analysis layer

The purpose of the context analysis layer is to develop the mobile learning model. As discussed in the previous section, the mobile learning model consists of learning attributes of the learner while learning with the help of a smartphone. DBSCAN relies on learners' location attributes (latitude and longitude coordinates) to cluster learners into different groups. Table 2 shows generic and specific learning attributes of learners related to programming levels/exercises. Instructors can specify generic learning attributes of programming levels in the XML-based attributes description documents.

The context analysis layer integrates the Google API client library in Python code running on the Firebase cloud to make calls to Google Cloud ML Engine REST APIs. Location information in the Firebase cloud is sent to Google Cloud ML Engine as a job along with the required parameters. Job parameters include training application name, job name, module name, machine learning algorithm, and job directory. For quick and easy sharing of location data between Firebase and Google Cloud ML Engine, we used the Firebase SDK (software development kit), which uses a default storage bucket in Google Cloud ML Engine. The DBSCAN Python code running on Google Cloud ML Engine determines which places are commonly visited by learners in the form of clusters.

Table 2. Generic and specific learning attributes of learners.

Generic learning attribute of programming levels				
Learning time & duration	Difficulty level	Real-world programming examples (location-wise)	Generic learning contents	Background knowledge
Learner learning attribute of programming levels				
Learning time & duration	Performance in learning level	Preferred real-world programming exercises	Adaptive learning contents	Background knowledge

3.2.1. Geofencing learners' locations

For our experimental setup, four types of monitored location zones were defined, namely recreation grounds, library, classrooms, and marketplace. The Android client app was responsible for sending the learner's location statistics to the Firebase database every time a learner entered or left a particular Google maps geofenced zone. Four monitored locations of interest were specified using their latitude and longitude data. Due to the difference in the proximity of location zones, different radii were defined. The latitude, longitude, and radius defined the geofence for four different monitored locations. The following Android code snippet shows how geofencing was defined.

Listing 1. Geofencing learner location

```
// Create a Geofence
private Geofence createGeofence(
LatLng latLng, float radius ) {
Log.d(TAG, "createGeofence");
    return new Geofence.Builder()
.setRequestId(GEOFENCE_REQ_ID)
.setCircularRegion
(latLng.latitude, latLng.longitude, radius)
.setExpirationDuration( GEO_DURATION )
.setTransitionTypes
(Geofence.GEOFENCE_TRANSITION_ENTER
| Geofence.GEOFENCE_TRANSITION_EXIT )
            .build();
}
```

3.2.2. Applying DBSCAN clustering on learners' geofenced data

We used the density-based spatial clustering of applications with noise (DBSCAN) clustering algorithm to determine which monitored locations were visited by learners and which were not. DBSCAN clustering enabled location-based programming recommendations to learners on their smartphones. We used the DBSCAN clustering algorithm as it works well with latitude-longitude geographical data and arbitrary distances. In DBSCAN algorithm clusters, a geographical dataset is based on two parameters: epsilon and minimum number of points (min-points). Epsilon determines physical distances to search for points near a given point, whereas

min-points determines the minimum cluster size, i.e. how many points should be present adjacent to an assumed given point in order to keep growing a given cluster. Eq. 1 shows how DBSCAN defines a spatial cluster.

$$N\epsilon(P) : \{q|d(p,q) \leqslant \varepsilon\} \tag{1}$$

In the above equation, 'high density' regions are those where the ϵ neighborhood of an object contains at least min-points of objects. The (p,q) pair represents the location objects (students' location points) in the DBSCAN cluster. The Google Cloud ML Engine performed learners' DBSCAN location-clustering tasks as it scales well with millions of users and billions of events [47]. Google Cloud ML Engine carried out location-clustering using key-value pairs. We choose a key to be a learner's unique identifier and a value to be the geographical latitude and longitude data of a learner. An example of a key-value pair is shown below in a tuple extracted from the cloud database.

Listing 2. Geofencing key-value pair

```
(19987, DBSCANMatrix (33.588570, 71.439529
                      33.586997, 71.441717
                      33.587641, 71.443091
                      33.585388, 71.440773))
```

For demarcation of geographical places into different clusters based on the learner's visits, Google Cloud ML Engine uses the following DBSCAN algorithm implemented in Python. In this code, Google Cloud ML Engine looks for the learner's entrance or exit in the range of 100–350 m (about 0.001 in degrees) and it starts clustering if there are at least five points close to each other.

Listing 3. Partitioning of geographical places into clusters by DBSCAN algorithm

```
import pandas as pd, numpy as np,
matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from geopy.distance import great_circle
from shapely.geometry import MultiPoint
df = pd('DBSCANMatrix')
coords = df.as_matrix(columns=['latitude',
'longitude'])
DBSCAN(epilson=0.001, metric = 'euclidean',
min-points=5)
Dbscan.fit(coords)
clusters = pd.Series
([coords, for n [labels == n]
in range (num_clusters)])
```

Figure 2 shows an example of four clusters showing that a particular learner visited locations during a period of two months. Learner visits are clustered according to their geographical trace. For instance, classrooms visits are represented by dark green dots, library stays by dark orange dots, recreation area entry by dark red dots, and marketplace visits by blue dots. The information related to geographical locations of every learner is further transferred to the learning path generation layer for development and dissemination of appropriate learning material to learners.

Figure 2. Clustering of learner visits according to geographical location by DBSCAN.

3.3. Learning path generation layer

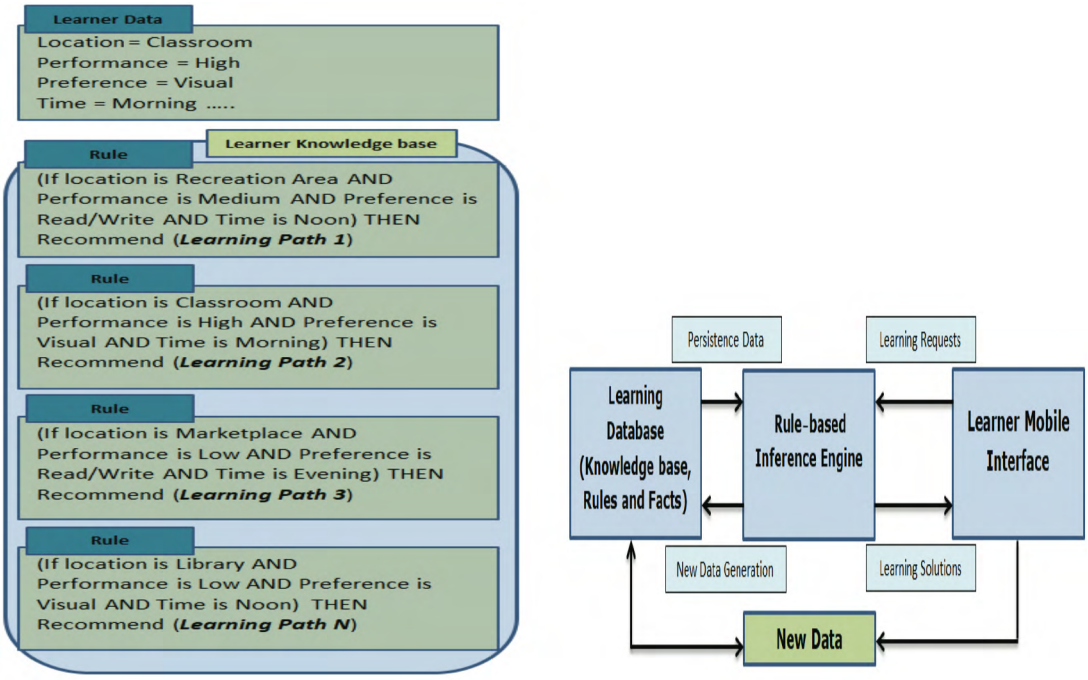
In the learning path generation layer, CSMLS used a rule-based inference engine on learning context data to recommend suitable programming exercises to learners [48]. The input for the learning path generation layer is learners' geographical clusters generated by DBSCAN and learning context data stored in the Firebase database. Using the rule-based inference engine, the learning path generation layer makes appropriate recommendations to learners based on their learning attributes, which together with the passage of time from the learners' knowledge base that subsequently leads to the deductive derivation of new learning facts.

The rule-based inference engine module represents a model of actual learner behavior. The working of the rule-based inference engine is based on condition-action pairs as shown in Figure **??**. A rule is made of an IF (condition) part and THEN (action) part. If the IF part of the rule is satisfied, consequently, the THEN part is executed; otherwise, the flow of control goes to the next IF/THEN rule. The rule-based inference module learns from experience and generates new learning knowledge for the learning database module. The learner mobile interface module is responsible for delivering adaptive real-world programming content to learners on their smartphones. Any new learning knowledge gathered by the learner's smartphone is communicated to the learning database module by the learner mobile interface, where, based on this knowledge, new rules are established and existing rules are reorganized.

The learning path generation layer uses a forward chaining approach of the rule-based inference engine to generate dynamic learning paths to learners based on their contextual data. As shown in Figure **??**, the rule-based inference engine uses three modules to generate appropriate learning paths to the learners. In the learning database module, the knowledge acquisition activity obtains the learner's location and learning behavior information from the context analysis layer. With the passage of time, the learning database module gets more mature and complete as it gets information from the context analysis layer and new data generated by the rule-based inference engine. Depending on the learning behavior information, new learning recommendation rules are recognized, formulated, and restructured in the learning database module.

4. Experimental procedure

In order to evaluate the effectiveness of CSMLS, we carried out our experimental study at Kohat University of Science and Technology, Pakistan. Three monitored physical locations, i.e. classrooms, a recreation area,



(a) Execution of Rules using IF THEN logic

(b) Working of Rule based inference engine

Figure 3. Students' IEPLS interaction activities.

and a library, were defined and selected inside the university while one monitored location, the marketplace, was monitored and geofenced outside of the campus. The main objective of this experimental study was to reveal whether or not the adaptive and real-world computer programming makes the learning of computer programming exciting and engaging to learners.

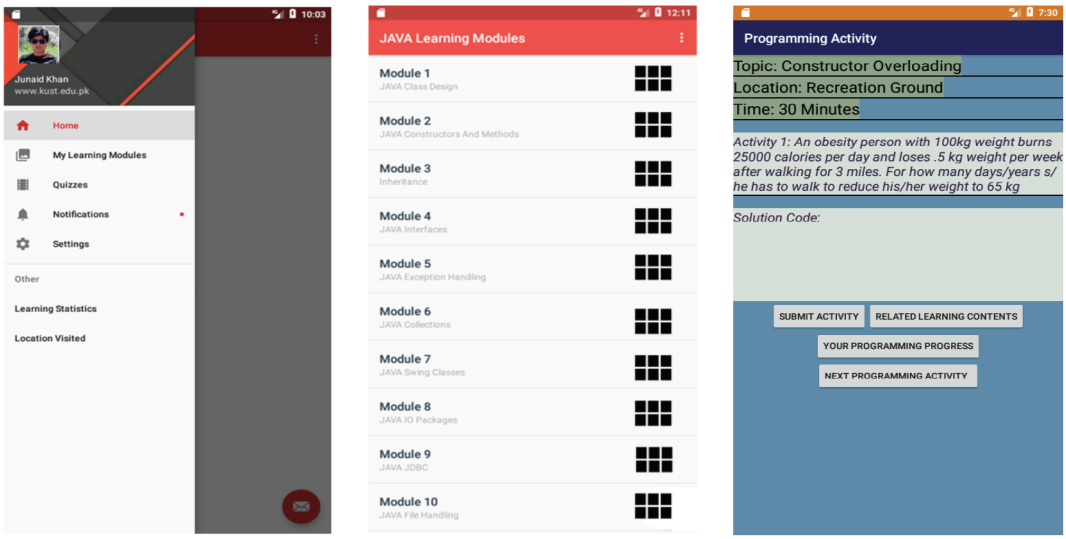
4.1. Participants

A total of 150 undergraduate students participated in our experimental study. We acquired a large number of participants because the DBSCAN machine learning algorithm generates better results for larger datasets. We randomly divided the participating students into two independent groups of the same size (75 participants in each group). A control (C) group received normal programming contents and exercises independently of considering their physical locations and learning behavior. The experimental (E) group received real-world programming content and exercises considering their physical locations and learning behavior preferences. While selecting participants, we made sure that all participants had earlier programming learning experience and had Android smartphones running Android KitKat 4.4.4 or a later version of OS. We also ascertained that the Android client app was installed and running properly on the smartphones of all the participants. A brief 30-min training session was given to all participants regarding Android client app functionality, modules, customization, and connectivity to the Firebase cloud. For detailed evaluation, both the C and E groups used CSMLS for one complete semester, i.e. 6 months. An exhaustive experiment was needed as DBSCAN generates more accurate results on a large dataset. Routine visits to monitored geographical locations and learning behavior (preferences, performances, times) of both groups were recorded in the Firebase persistent database storage on a regular basis. With the passage of time, the E group participants received real-world

programming content and exercises according to the places they visited and learning behavior patterns while C group participants received normal programming content independent of their physical location visits and learning behavior patterns on their smartphones.

4.2. Android client app interaction

Figure 4 shows Android client app interaction interfaces for the E group participants. Figure **??** shows a learner's personal profile, from which he or she can navigate to learning modules, quiz attempts, notifications from the Firebase cloud, options to customize settings, learning statistics, and history of visited locations. Figure **??** illustrates the Java programming modules in chronological order, which a learner has to cover in a 6-month period. For E group participants, the content of these programming modules becomes adaptive with respect to the locations visited and learning behavior. This content adaptiveness is further presented in Figure **??**, where a learner who often visits the recreational area (playgrounds, physical training zone) is presented with a programming task that is related to his or her visited location.



(a) Learner profile (b) Adaptive programming modules (c) Adaptive programming exercise

Figure 4. Learner interaction with CSMLS Android client app.

4.3. CSMLS operational analysis

In this section, we discuss the influence of CSMLS in improving the computer programming skills of learners. At the start of the CSMLS experimental phase, a pretest was conducted to gather initial programming capabilities of both groups. The programming language selected for the pretest was Java. The pretest content included basic Java concepts, which are also applied in other object-oriented programming languages, such as conditional statements, loops, variables, arrays, and classes. The pretest lasted for 60 min and students were informed to submit the solutions online through their smartphones. The analysis results of both groups are shown in Table 3, where it can be concluded that the initial programming capabilities of both groups were almost the same with mean values of 52.48 and 51.85, respectively.

From Table 4, we also notice that the P-value of Levene's test is 0.498 (>0.05), which implies that the variances of both groups are the same. Similarly, the significance P-value is 0.277 (>0.05), so we conclude that there is no significant difference between the mean score of the E group and C group at 5% significance level.

Table 3. Pretest group statistics.

Participants		N	Mean	Std. deviation	Std. error mean
Programming score	Experimental group	75	52.48	3.60	0.4164
score	Control group	75	51.85	3.41	0.3948

Table 4. Pretest independent sample t-test.

	Variances	Levene's test for equality of variances	t-Test for equality of means			
		F	Sig.	t	df	Sig. (2-tailed)
Programming score	Equal variances assumed	0.461	0.498	1.092	148	0.277
	Equal variances not assumed			1.092	147.58	0.277

After making sure in the pretest that both groups had the same computer programming skills, we conducted six posttests after every one month in a 6-month period to identify the difference between the two groups after using our proposed CSMLS learning system. With the passage of time, the E group received adaptive programming content that encouraged learning and solving real-world programming problems, whereas the C group received normal programming content taught in textbooks where learners have to solve general programming problems. We were interested in finding the difference in programming skills between the two groups after using CSMLS and finding how much CSMLS is engaging and helpful during the learning process. In this experiment the target learning language was Java and its learning contents were divided into 10 modules.

In order to discern the effect of the learning guidance of CSMLS, we piloted six posttests after every one month for six months. The posttests were an integral part of the CSMLS system and every participant used his or her smartphone to take the posttest and submit the results on the Firebase cloud. Every posttest lasted for 60 min and for E group participants the posttest exercise contents were according to the participant's past learning behavior (location, performance, preferences, and learning time). On the other hand, C group participants' posttest exercise contents were general, irrespective of their learning behavior. Table 5 shows a group analysis for six posttests with mean and standard deviation. The result show that during the first two months, the performance of both groups was almost parallel with mean values of 55.10 and 54.88 for the first month and mean values of 57.80 and 54.90 for the second month, respectively. The significant (2-tailed) P-values for the first and second posttest are 0.819 and 0.024, both >0.05, implying that statistically there is no significant difference between E group performance and C group performance. It was after the third test result when we observed the influence of CSMLS in improving programming skills of experimental group participants. In Table 6, the significant (2-tailed) P-value for the third posttest is 0.000120 (<0.05), which shows a noticeable difference between the mean scores of the E group and C group at 5% significance level. Similarly, the significant (2-tailed) P-values for the fourth, fifth, and sixth posttest scores are 0.000, 0.001, and 0.000, all less than a 0.05 significance level, demonstrating that the programming performance of E group participants was better than that of C group participants.

Table 5. Posttest group statistics.

Groups		N	Mean	Std. deviation	Std. error mean
Posttest, first month	Experimental group	75	55.106	6.421	0.741
	Control group	75	54.880	5.687	0.656
Posttest, second month	Experimental group	75	57.800	7.289	0.841
	Control group	75	54.906	8.226	0.949
Posttest, third month	Experimental group	75	60.280	7.045	0.813
	Control group	75	55.120	8.850	1.021
Posttest, fourth month	Experimental group	75	61.493	7.831	0.904
	Control group	75	54.040	10.047	1.160
Posttest, fifth month	Experimental group	75	62.880	8.196	0.946
	Control group	75	58.320	7.788	0.899
Posttest, sixth month	Experimental group	75	63.160	8.147	0.940
	Control group	75	57.253	8.243	0.951

Table 6. Posttest independent sample t-test.

Tests	Variances	Levene's test for equality of variances		t-Test for equality of means		
		F	Sig.	t	df	Sig. (2-tailed)
Posttest, first month	Equal variances assumed	2.201	0.140	0.229	148	0.819
	Equal variances not assumed			0.229	145.875	0.819
Posttest, second month	Equal variances assumed	2.437	0.121	2.280	148	0.024
	Equal variances not assumed			2.280	145.885	0.024
Posttest, third month	Equal variances assumed	4.348	0.039	3.950	148	0.000
	Equal variances not assumed			3.950	140.918	0.000
Posttest, fourth month	Equal variances assumed	10.180	0.002	5.067	148	0.000
	Equal variances not assumed			5.067	139.674	0.000
Posttest, fifth month	Equal variances assumed	0.391	0.533	3.493	148	0.001
	Equal variances not assumed			3.493	147.614	0.001
Posttest, sixth month	Equal variances assumed	0.009	0.926	4.414	148	0.000
	Equal variances not assumed			4.414	147.980	0.000

Later on, we analyzed the Firebase cloud database to find the prime reasons why the learning performance of both groups was the same during the first two months and we came up with a few reasons. Initially, the DBSCAN running on Google Cloud ML Engine was not able to adapt to the learning behavior of learners because the learners' dataset (learning model) contained inadequate information. Due to this, the E group participants were having problems in getting adaptive programming content. With the passage of time and regular use, the data model got mature, containing ample information. Subsequently, CSMLS started to recognize the differences in learning behavior between the participants of the E and C groups. As a result, CSMLS started to recommend adaptive and real-world programming content to E group participants, which resulted in improving

their learning performance as compared to C group participants. One of the reasons for testing CSMLS for 6 months was to make sure that the DBSCAN algorithm had enough data points to generate proper location clusters and to recognize individual learners based on their visited locations history.

Table 7. End user computing satisfaction (EUCS) questionnaire results.

	Dimension	Question	Mean
1	Usefulness	I think the use of CSMLS improved my programming skills	4.67
2	Usefulness	I think CSMLS contributed a lot in my understanding of computer programming	4.55
3	Engaging	I was curious about learning to program using CSMLS	4.78
4	Engaging	CSMLS was able to engage me in learning programming on a daily basis	4.84
5	Ease of use	I think using CSMLS is easy	4.53
6	Ease of use	I think it is possible to use CSMLS without expert help	4.44
7	Timeliness	The programming contents were provided on time	4.60
8	Timeliness	CSMLS knew when to send programming contents	4.67
9	Adaptive	The programming contents provided were tailored and personalized	4.34
10	Adaptive	The programming contents provided were according to my performance state	4.55
11	Attitude towards CSMLS	I will use a similar type of system in the future	4.87
12	Attitude towards CSMLS	I will continue to use CSMLS in order to increase my programming skills	4.76

4.4. Analysis of EUCS model questionnaire

We used a modified version of the end user computing satisfaction (EUCS) model [49] to elicit learners' satisfaction and experiences of using CSMLS. We conducted an online EUCS survey with 75 E group participants. The EUCS survey questionnaire covered six dimensions of CSMLS, namely usefulness, engagingness, ease of use, timeliness, adaptiveness, and attitude towards CSMLS. The online survey comprised 12 questions covering the six dimensions of CSMLS on a five-point Likert-scale where 1 represents "strongly disagree" and 5 represents "strongly agree". The mean user satisfaction ranking is greater than 4 (agree), which shows that, overall, the participants were satisfied with using the CSMLS system. Table 7 displays CSMLS learning dimensions, corresponding questions, and mean scores. The responses to dimensions 1 and 2 indicate that CSMLS was able to increase the programming skills of learners (m = 4.67, m = 4.55). The responses to dimensions 3 and 4 show that CSMLS was able to attract and engage participants in making time to learn computer programming (m = 4.78, m = 4.84). The responses to dimensions 5 and 6 show that the user interface of CSMLS was easy to use and user-friendly (m = 4.53, m = 4.44). The responses to dimensions 7 and 8 indicate that CSMLS considered learners' preferred learning time and sent programming contents accordingly (m = 4.60, m = 4.67). The answers to dimensions 9 and 10 indicate that personalized and tailored programming contents were delivered to learners according to their learning performance and learning behavior (m = 4.34, m =4.55). The responses to dimensions 11 and 12 reflect that learners agreed to use CSMLS or a similar kind of software system in the future to increase their programming skills (m = 4.87, m = 4.76).

5. Conclusions and future work

In this study, we proposed and developed a cloud-supported ML system, CSMLS, to guide learners in learning real-world and applied computer programming considering their location and other learning behavior parameters. The density-based spatial clustering of applications with noise (DBSCAN) algorithm together with a rule-based inference engine was administered on Firebase and Google Cloud ML Engine to recommend appropriate programming exercises and content to learners on their smartphones. For this study, four types of geographical locations were geofenced to record learners' physical visits and subsequently, based on these visits, to recommend real-world programming exercises. CSMLS comprises three layers, namely a context acquisition layer, context analysis layer, and learning path generation layer. The context acquisition layer uses GPS and a geofencing method for acquiring location information facilitated by satellite signals, Wi-Fi hotspot IDs, the Android client app, and cell tower IDs. The purpose of the context analysis layer was to develop a mobile learning model for individual learners. The context analysis layer used the DBSCAN algorithm to cluster the learners based on their geographical information (latitude and longitude coordinates). The information processed by the context analysis layer is further transferred to the learning path generation layer to generate appropriate and suitable learning guidance and learning content to learners. The learning path generation layer used the rule-based inference engine (IF/THEN rule) to recommend personalized programming exercises to learners. Moreover, we also observed that experimental group participants receiving adaptive and applied computer programming content found CSMLS more useful, engaging, adaptive, easy to use, and productive. In the first two months of CSMLS testing, the experimental and control groups' programming performances were found to be the same, which was due to the fact that initially experimental group participants were considering it as a normal programming learning system and were not able to interpret its help and support. Furthermore, the DBSCAN learning model in the Google Cloud ML Engine was not fully trained in the first two-month duration, due to which CSMLS was not able to differentiate among the learners based on their geographical information database. After two months, the DBSCAN training model was complete, and as the geographical dataset matured, CSMLS was able to cluster learners based on their physical locations and subsequently was able to recommend real-world adaptive programming content to experimental group participants.

To the best of our knowledge, this is the first study where we premeditated the application of machine learning on information generated from educational settings. By using mobile learning environment characteristics to provide personalized programming material to learners, we developed a learning model via an unsupervised machine learning clustering method, i.e. DBSCAN geographical clustering. This learning model was used by the rule-based inference engine to generate personalized learning paths to diverse learners. In the future, we would like to expand the geographical information database by including more physical locations such as home, regular travels, banks, museums, industry, hospitals, etc. We expect that a larger geographical information database will help us know more comprehensive information about different learners at detailed and granular levels. We would also like to study the effect of supervised ML algorithms (supervised learning) that deal with classification and prediction of a learner's performance.

References

[1] Robins A, Rountree J, Rountree N. Learning and teaching programming: a review and discussion. Computer Science Education 2003; 13 (2): 137-172. doi: 10.1076/csed.13.2.137.14200

[2] Jenkins T. On the difficulty of learning to program. In: Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences; Loughborough, UK; 2002. pp. 53-58.

[3] Watson C, Li FW. Failure rates in introductory programming revisited. In: Proceedings of the 2014 Conference on Innovation & Technology in Computer Science Education; Uppsala, Sweden; 2014. pp. 39-44. doi: 10.1145/2591708.2591749

[4] Tillmann N, Moskal M, Halleux J, Fahndrich M, Bishop J et al. The future of teaching programming is on mobile devices. In: Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education; Haifa, Israel; 2012. pp. 156-161. doi: 10.1145/2325296.2325336

[5] Psycharis S, Kallia M. The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. Instructional Science 2017; 45 (5): 583-602. doi: 10.1007/s11251-017-9421-5

[6] Verbert K, Manouselis N, Ochoa X, Wolpers M, Drachsler H et al. Context-aware recommender systems for learning: a survey and future challenges. IEEE Transactions on Learning Technologies 2012; 5 (4): 318-335. doi: 10.1109/TLT.2012.11

[7] Hwang GJ, Tsai CC, Yang SJ. Criteria, strategies and research issues of context-aware ubiquitous learning. Journal of Educational Technology & Society 2008; 11 (2): 81-91.

[8] Chen GD, Chang CK, Wang CY. Ubiquitous learning website: scaffold learners by mobile devices with information-aware techniques. Computers & Education 2008; 50 (1): 77-90. doi: 10.1016/j.compedu.2006.03.004

[9] Ogata H, Yano Y. Context-aware support for computer-supported ubiquitous learning. In: 2nd IEEE International Workshop on Wireless and Mobile Technologies in Education; Jung Li, Taiwan; 2004. pp. 27-34. doi: 10.1109/WMTE.2004.1281330

[10] Chen CM, Li YL. Personalised context-aware ubiquitous learning system for supporting effective English vocabulary learning. Interactive Learning Environments 2010; 18 (4): 341-364. doi: 10.1080/10494820802602329

[11] Chen CC, Huang TC. Learning in a u-museum: developing a context-aware ubiquitous learning environment. Computers & Education 2012; 59 (3): 873-883. doi: 10.1016/j.compedu.2012.04.003

[12] Valenzuela JF, Pardo PJ, Padilla P, Lozano AJ. Low cost ubiquitous context-aware wireless communications laboratory for undergraduate students. IEEE Transactions on Learning Technologies 2016; 9 (1): 31-36. doi: 10.1109/TLT.2015.2438864

[13] Hsu TY, Chiou CK, Tseng JC, Hwang GJ. Development and evaluation of an active learning support system for context-aware ubiquitous learning. IEEE Transactions on Learning Technologies 2016; 9 (1): 37-45. doi: 10.1109/TLT.2015.2439683

[14] Mandula K, Meda SR, Jain DK, Kambham R. Implementation of ubiquitous learning system using sensor technologies. In: IEEE International Conference on Technology for Education; Chennai, India; 2011. pp. 142-148. doi: 10.1109/T4E.2011.30

[15] Borrego F, García GC, Ruiz IL, Gómez MÁ. An NFC based context-aware solution for access to bibliographic sources in university environments. Journal of Ambient Intelligence and Smart Environments 2013; 5 (1): 105-118.

[16] Wu PH, Hwang GJ, Su LH, Huang YM. A context-aware mobile learning system for supporting cognitive apprenticeships in nursing skills training. Journal of Educational Technology & Society 2012; 15 (1): 223-236.

[17] Godwin-Jones R. Mobile apps for language learning. Language Learning & Technology 2011; 15 (2): 2-11.

[18] Gómez JE, Huete JF, Hernandez VL. A contextualized system for supporting active learning. IEEE Transactions on Learning Technologies 2016; 9 (2): 196-202. doi: 10.1109/TLT.2016.2531685

[19] Yao CB. Constructing a user-friendly and smart ubiquitous personalized learning environment by using a context-aware mechanism. IEEE Transactions on Learning Technologies 2017; 10 (1): 104-114. doi: 10.1109/TLT.2015.2487977

[20] Hwang WY, Wang CY, Sharples M. A study of multimedia annotation of Web-based materials. Computers & Education 2007; 48 (4): 680-699. doi: 10.1016/j.compedu.2005.04.020

[21] Menzel M, Ranjan R. CloudGenius: decision support for web server cloud migration. In: Proceedings of the 21st International Conference on World Wide Web; Lyon, France; 2012. pp. 979-988. doi: 10.1145/2187836.2187967

[22] Barish G, Obraczke K. World wide web caching: trends and techniques. IEEE Communications Magazine 2000; 38 (5): 178-184. doi: 10.1109/35.841844

[23] Truong D. How cloud computing enhances competitive advantages: a research model for small businesses. The Business Review, Cambridge 2010; 15 (1): 59-65.

[24] Kumar K, Lu YH. Cloud computing for mobile users: can offloading computation save energy?. Computer 2010; 1 (4): 51-56. doi: 10.1109/MC.2010.98

[25] Stergiou C, Psannis KE, Kim BG, Gupta B. Secure integration of IoT and cloud computing. Future Generation Computer Systems 2018; 78: 964-975. doi: 10.1109/MC.2010.98

[26] Chang V, Bacigalupo D, Wills G, Roure D. A categorisation of cloud computing business models. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing; Melbourne, Australia; 2010. pp. 509-512. doi: 10.1109/CCGRID.2010.132

[27] Sultan N. Cloud computing for education: A new dawn? International Journal of Information Management 2010; 30 (2): 109-116. doi: 10.1016/j.ijinfomgt.2009.09.004

[28] Nguyen VA, Pham VC, Ho SD. A context-aware mobile learning adaptive system for supporting foreigner learning English. In: 2010 IEEE RIVF International Conference on Computing & Communication Technologies, Research, Innovation, and Vision for the Future; Hanoi, Vietnam; 2010. pp. 1-6. doi: 10.1109/RIVF.2010.5632316

[29] Dao TH, Jeong SR, Ahn H. A novel recommendation model of location-based advertising: context-aware collaborative filtering using GA approach. Expert Systems with Applications 2012; 39 (3): 3731-3739. doi: 10.1016/j.eswa.2011.09.070

[30] Chu HC, Hwang GJ, Tsai CC, Tseng JC. A two-tier test approach to developing location-aware mobile learning systems for natural science courses. Computers & Education 2010; 55 (4): 1618-1627. doi: 10.1016/j.compedu.2010.07.004

[31] Yılmaz Ö. Matching points of interest with user context: an ANN approach. Turkish Journal of Electrical Engineering & Computer Sciences 2017; 25 (4): 2784-2795. doi: 10.3906/elk-1503-61

[32] Lan YF, Tsai PW, Yang SH, Hung CL. Comparing the social knowledge construction behavioral patterns of problem-based online asynchronous discussion in e/m-learning environments. Computers & Education 2012; 59 (4): 1122-1135. doi: 10.1016/j.compedu.2012.05.004

[33] Trifonova A, Ronchetti M. Hoarding content in m-learning context. In: Third IEEE International Conference on Pervasive Computing and Communications Workshops; Kauai Island, HI, USA; 2005. pp. 327-331. doi: 10.1109/PERCOMW.2005.39

[34] Hsieh HC, Chen CM, Hong CM. Context-aware ubiquitous English learning in a campus environment. In: Seventh IEEE International Conference on Advanced Learning Technologies; Niigata, Japan; 2007. pp. 351-353. doi: 10.1109/ICALT.2007.106

[35] Yau JY, Joy M. A context-aware personalized m-learning application based on m-learning preferences. In: 6th IEEE International Conference; Washington, DC, USA; 2010. pp. 11-18. doi: 10.1109/WMUTE.2010.15

[36] Reychav I, Wu D. Exploring mobile tablet training for road safety: a uses and gratifications perspective. Computers & Education 2014; 71: 43-55. doi: 10.1016/j.compedu.2013.09.005

[37] Hwang GJ, Yang TC, Tsai CC, Yang SJ. A context-aware ubiquitous learning environment for conducting complex science experiments. Computers & Education 2009; 53 (2): 402-413. doi: 10.1016/j.compedu.2009.02.016

[38] Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing 2006; 70 (1-3): 489-501. doi: 10.1016/j.neucom.2005.12.126

[39] Dinh HT, Lee C, Niyato D, Wang P. A survey of mobile cloud computing: architecture, applications, and approaches. Wireless Communications and Mobile Computing 2013; 13 (18): 1587-1611. doi: 10.1002/wcm.1203

[40] Krause A, Smailagic A, Siewiorek DP. Context-aware mobile computing: learning context-dependent personal preferences from a wearable sensor array. IEEE Transactions on Mobile Computing 2006; 5 (2): 113-127. doi: 10.1109/TMC.2006.18

[41] Pazzani MJ, Billsus D. Content-based recommendation systems. In: Brusilovsky P, Kobsa A, Nejdl W (editors). The Adaptive Web: Lecture Notes in Computer Science, Vol. 4321. Berlin, Germany: Springer, 2007, pp. 325-341. doi: 10.1007/978-3-540-72079-9_10

[42] Webb GI, Pazzani MJ, Billsus D. Machine learning for user modeling. User Modeling and User-Adapted Interaction 2001; 11 (1-2): 19-29. doi: 10.1023/A:1011117102175

[43] Dawoud W, Takouna I, Meinel C. Infrastructure as a service security: challenges and solutions. In: 7th International Conference on Informatics and Systems; Cairo, Egypt; 2010. pp. 1-8.

[44] Lawton G. Developing software online with platform-as-a-service technology. Computer 2008; 41 (6): 13-15. doi: 10.1109/MC.2008.185

[45] Buxmann P, Hess T, Lehmann S. Software as a service. Wirtschaftsinformatik 2008; 50 (6): 500-503. doi: 10.1007/s11576-008-0095-0

[46] Sareen P. Cloud computing: types, architecture, applications, concerns, virtualization and role of it governance in cloud. International Journal of Advanced Research in Computer Science and Software Engineering 2013; 3 (3): 533-538.

[47] Fernando N, Loke SW, Rahayu W. Mobile cloud computing: a survey. Future Generation Computer Systems 2013; 29 (1): 84-106. doi: 10.1016/j.future.2012.05.023

[48] Mitra S, Mitra M, Chaudhuri BB. A rough-set-based inference engine for ECG classification. IEEE Transactions on Instrumentation and Measurement 2006; 55 (6): 2198-2206. doi: doi.org/10.1109/TIM.2006.884279

[49] Etezadi J, Farhoomand AF. A structural model of end user computing satisfaction and user performance. Information & Management 1996; 30 (2): 65-73. doi: 10.1016/0378-7206(95)00052-6