Agent-based adaptive e-learning model for any learning management system

Authors:
George W. Musumba[1]
Robert O. Oboko[2]
Henry O. Nyongesa[3]

Affiliations:
[1]Department of Computer Science, Dedan Kimathi University of Technology, Kenya

[2]School of Computing and Informatics, University of Nairobi

[3]Department of Computer Science, University of the Western Cape

Correspondence to:
George Musumba

Email:
musumbaggw@gmail.com

Postal address:
P.O. Box 657-10100, Nyeri, Kenya

Dates:
Received: 29 Jan.2013
Accepted: 28 Apr.2013
Published: 06 June 2013

How to cite this article:
Musumba GW, Oboko RO, Nyongesa HO. Agent Based Adaptive E-Learning Model for Any Learning Management System. Int J Machine Learn Appl. 2013;2(1),Art. #6, 9 pages. http://dx.doi.org/10.4102/ijmla.v2i1.6

Many scholars are interested in improving e-learning in order to provide easy access to educational materials. There is, however, the need to incorporate the ability to classify learners into these learning systems. Learner classification is used adaptively to provide relevant information for the various categories of learners. There is also a need for learning to continue, whether learners are on- or off-line. In many parts of the world, especially in the developing world, most people do not have reliable continuous internet connections. We tested an Adaptive e-Learning Model prototype that implements an adaptive presentation of course content under conditions of intermittent Internet connections. This prototype was tested in February 2011 on undergraduate students studying a database systems course. This study found out that it is possible to have models that can adapt to characteristics such as the learner's level of knowledge and that it is possible for learners to be able to study under both on- and off-line modes through adaptation.

Introduction

Problem statement

The e-learning paradigm should capitalise on two aspects,[1] (1) the elimination of the barriers of time and distance and (2) the personalisation of the learners' experience. The current trends in education and training should put emphasis on identifying methods and tools for delivering just-in-time, on-demand knowledge experiences tailored to individual learners, taking into consideration their differences in skills level, perspectives, culture and other educational contexts. However, most of these systems are still static and inflexible, being developed on the basis of 'One size fits all'.[2] As a result, they do not present only relevant information to learners, but instead present all information (learning materials) from basic to advanced concepts at the same time. This gamut of information given to learners without considering their specific requirements makes them experience difficulties in perceiving and applying the information when answering quizzes and solving problems. This is because learners get all the information they are supposed to get, but in bits and pieces and in different learning stages. Different stages have specific information and quizzes are based on the information per stage, hence accessing all the information at the same time makes some of the information irrelevant.

These systems are also developed to work under constant internet connections (web applications that must be accessed online in order to function). In these instances, the learners can only learn when online. These concepts are found in nearly all e-learning systems which have been deployed by learning institutions and even in organisations that train their staff and clients using Learning Management Systems (LMS). For example, nearly all university e-learning portals are only functional online, meaning that learners must be connected to the internet in order to be able to carry out learning activities. The same case applies to most online training systems for organisations.

In the modern world, although internet accessibility has increased significantly, many areas still do not have access. This is seen especially in rural areas of the developing world where most Internet Service Providers (ISPs) have not yet invested in infrastructure to facilitate internet connectivity. This limitation locks out many people who would otherwise have wanted to participate in this learning process. This disadvantaged category of people can only learn by being present at a learning centre. Moreover, even in areas where internet is available, the connection does not always have 100% uptime. During the internet downtime, learning does not take place unless there is a physical instructor and a learner at the same venue or the learner has to wait for the internet connection to be re-established.

Furthermore, the LMSs that have been developed and distributed for use are installed as applications that must be in place for implementation and subsequent usage by the target groups.

It is necessary to have some modules which can be plugged into a system to provide additional functionality as and when necessary, such as synchronisation of course content and profiles to the off-line models from the online model after loss of connection.

The service developed in this research can integrate with any LMS and has the advantage of being deployed for both on- and off-line learning (under intermittent internet connection conditions). A framework is designed in the form of an Application Programming Interface (API)[3] which can be integrated into any LMS and can be used to classify learners dynamically into various categories as defined by information in the learner model.

The K-Nearest Neighbour algorithm (KNN) was used to classify new learners.[4] K-Nearest Neighbour algorithm is a component of supervised learning that has been used in many applications in the field of data mining. It is a method for classifying objects based on the closest training example in the feature space. An object is classified by a majority vote of its K neighbours and K is always an integer. The neighbours are taken from a set of pre-existing training examples for which the correct classification is known.[5]

Key focus

This article outlines how to overcome the above challenges by developing an agent-based personalised adaptive learning model. This model is deployed as a service using agent technology and not just as an application as is the case with all other available LMS. The service, or agent, is packaged as a Dynamic-Link Library (DLL).

In summary, this research is intended to meet the following objectives:

- To develop an adaptive learning model to support learning under conditions of intermittent internet connections.
- To classify learners correctly using the KNN classification algorithm which considers learners' features as values.
- To update the learners' profiles depending on their acquired knowledge, performance in quizzes and classification in order to ensure that they are able to avail themselves of the relevant learning materials.

Related work

E-learning has long been recognised as the new wave in education. It allows learners to study without the limitations of time and space. Despite the advantages of e-learning, most systems have not been designed well enough to respond adaptively to the individual learners' characteristics and needs. Granular information is essential for the delivery of the right information, to the right learner, in the right amounts.[6] The development of these systems enables just-in-time learning and the convergence of e-learning with Knowledge Management.

The ideal system should classify learners and provide appropriate learning materials customised for the individual. The 'one size fits all' philosophy results in too much information for users and lacks personalisation.[2] Today, many vendors offer products called Learning Management Systems (LMS), which they claim provide a complete e-learning solution.[6] However, products in this category do not address the need to develop and manage increasing volumes of content in smaller chunks by a larger group of content providers such as learning institutions. Nor do they provide adequate mechanisms for maintaining consistent instructional presentation or adapting that content to the needs of learners. It is thus important for organisations embarking on an e-learning track as a mode of training, to request that the vendor have in place a framework that allows for personalised training.

A survey conducted by Sun et al.,[7] which set out to investigate the critical factors affecting learners' satisfaction in e-learning, revealed that learner computer anxiety, instructor attitude toward e-learning, e-learning course flexibility, e-learning course quality, perceived usefulness, perceived ease of use and diversity in assessments are the critical factors affecting learners' perceived satisfaction. We focus here on learner course flexibility and e-learning course quality in order to develop the Adaptive e-Learning Model (AEM).

Adaptive e-learning systems

Traditional Technology-Enhanced Learning (TEL) systems offer very few strategies for the personalisation of educational offerings. This limits the scope for providing tailored, effective TEL experiences to learners.[8] However, adaptive educational hypermedia systems (AEHS) have been developed in order to address learner dissatisfaction by attempting to personalise the learning experience. Recent research in TEL has focused on the provision of adaptive educational experiences that are tailored to the particular needs of a learner. This adaptivity can be based upon various characteristics of the learner, including knowledge level, goals or motivation. The purpose of such adaptive educational offerings is to maximise learner satisfaction, learning speed (efficiency) and educational effectiveness.[9]

Blochl et al.[2] proposed an adaptive e-learning system. The system applied a user-centric approach so as to improve its usability and acceptance by users. E-learner requirements, including user skills, learning styles, learning strategy and other user profile information, were introduced into the system.[2] In this system, the user learning activities are observed and are used to update the user profile. The e-learning system is adjusted according to a dynamic user profile.

Saleh *et al.*[10] presented an adaptive active e-learning framework which consists of self-learning material, visualisation in an interesting way and self-testing. The framework was implemented using simple tools to support adaptive e-learning systems for numbering educational

material and it could be also used in other courses such as logic design, image processing, computational models, information theory, information engineering and digital communications. The framework finds a better way to engage learners in the learning process. Through their experimental results, it was shown that their model improved the learning process and affected the students in a positive way.

Component technologies and artificial intelligence are used to deliver e-learning. These components include: pedagogy agents, interactivity level, quality of feedback, control strategies, tutorial remediation and student models.

Pedagogy agents are used for integrating the behaviour of users and e-learning components of the system. They can be used to check student participation, track student progress through task procedures and address students' errors. Other agents can be used as tools for feedback. User performance during instruction should be analysed in order to monitor learning. Control strategies, planning for content and delivery strategies should be based on learner knowledge and concept structures such as curricula. Tutorial remediation is the component responsible for selecting appropriate actions to be performed by the learner in order to accomplish a pedagogy task. Student models can be used to render individualised instruction in the system. Students' instructional activities can be filtered, analysed and sorted based on individual profiles. This kind of system adapts to the changing knowledge requirements of the learner, is interactive and provides regular access to resource materials.[11]

Web-mining techniques have been used to build recommender agent-based e-learning systems. An agent recommends activities to a learner based on his access history. The recommendation should be an on-line activity including doing an exercise, providing messages on conferencing systems, running an on-line simulation, or web resources. This agent is claimed to improve course material navigation and assist the on-line learning process.[12] By observing user typing events, behaviours on studying lessons on web browser, tasks and examples, errors made by users and debugging events on the editor, the agent learns to understand user behaviour.[13]

Agent technology

A software agent is a computer programme that is capable of autonomous (or at least semi-autonomous) actions in pursuit of a specific goal.[14] The autonomy characteristic of a software agent distinguishes it from general software programmes. Autonomy in agents implies that the software agent has the ability to perform its tasks without direct control, or at least with minimum supervision, in which case it will be a semi-autonomous software agent. Software agents can be grouped, according to specific characteristics, into different software agent classes.[15] The available literature does not agree on the different types or classes of software agents. As software agents are classified according to a common set of characteristics, different classes of software agents often overlap, implying that a software agent might belong to more than one class at a time. For the purposes of this research, we distinguished between two simple classes of software agents, namely stationary agents and mobile agents. Agents in both these classes might, or might not, have any or a combination of the following characteristics: a user interface, intelligence, adaptivity, flexibility and collaborative properties.[16]

Whether or not an agent has a user interface, depends on whether it collaborates with humans, other agents or hosts. User interfaces are commonly only found where agents interact with humans. According to Wooldridge,[17] intelligence implies the inclusion of at least three distinct properties, namely reactivity, proactiveness and social ability. *Reactivity* refers to the agent's ability to perceive its environment and respond to changes that occur in order to achieve its design goals; *proactiveness* is the agent's ability to take the initiative in its environment in order to achieve its design goals; and *social ability* alludes to the collaborative nature of the agent.

There are different ways to define the collaborative nature of software agents. For the purposes of this paper we use Nienaber's[18] definition in which the collaborative nature of a software agent refers to the agent's ability to share information or barter for specialised services so as to cause a deliberate synergism amongst agents. It is expected that most agents should have a strong collaborative nature without necessarily implying other intelligence properties. *Adaptivity* is a characteristic that can also be regarded as an intelligence property, although it is not counted as being a prerequisite for identifying an agent as intelligent. Adaptivity refers to an agent's ability to customise itself on the basis of previous experiences. An agent is considered flexible when it can choose dynamically which actions to invoke, and in what sequence, in response to the state of its external environment.[3]

A stationary agent can be seen as a piece of autonomous (or semi-autonomous) software that resides permanently on a particular host. An example of such an agent is one that performs tasks on its host machine such as accepting mobile agents, allocating resources, performing specific computing tasks, enforcing security policies and so forth.

A mobile agent is a software agent that has the ability to transport itself from one host to another in a network. The ability to travel allows a mobile agent to move to a host that contains an object with which the agent wants to interact, and then to take advantage of the computing resources of the object's host in order to interact with that object. An example of a mobile agent is provided by a flight-booking system where a logged request is transferred to a mobile agent that then traverses the web seeking suitable flight-information quotations as well as itineraries.

We considered only stationary agents in this research. Agents reside on host devices and only interact with others through the implemented functions.

Research method and design

The model was set up in a computer laboratory and 30 students were allowed to use it for six hours a day for five days. An instructor introduced the system to the students, explaining the objectives of the model, how it works and the expected outcomes of the experiment.

The instructor the demonstrated the use the system for the learners: how to register, how to answer questions, the user interface changes as a result of adaptation, subsequent processes and, finally, filling out a questionnaire about the system. The students were then shown how to learn when online and also when off-line, they were given an explanation of similarities of individual profile status in both remote and local models, how to make a change on a local model whilst off-line, connecting to the remote model and how to check the similarities of the profiles both locally and remotely.

Two sets of learners were used. One group connected to both the intranet and internet, downloaded the information to the database in the local module and used it off-line to learn. The other group did the learning online only. The two groups were swapped around half-way through the course and the process was repeated. The internet was disconnected and reconnected five times for the online group. The results of the test (the test scores) were investigated to find out if there was a correlation between the treatments (i.e. learning online and learning under intermittent conditions but supported by the DLL).

For the second part of the evaluation, the learners were required to assess the system. A questionnaire was provided to be answered by the students after the learning process. The system provided the questionnaires online once all the requirements had been satisfied. The questions were designed to capture data related to the research objectives. Aspects considered included model usability, challenges in using the system and recommendations for improving the system.

The learning process

The following is a detailed description of the learning process as designed in this research.

Step 1: Registration of new learners

This was the first step, where details of a new learner were captured and the user name and password were created for subsequent logins and use of the system. An existing learner could also login and continue with the learning process. The learner's updated profile would determine the information that would be available to him.

Step 2: Prerequisite questions

These questions were designed to be able to test if the new learner met the prerequisite conditions so that he could be allowed to study the course. A combination of a number of the questions showed whether the new learner qualified to proceed with the course or not.

Step 3: Initial classification questions

At this stage, questions were designed that covered all sections of the course, beginning with the basic level through to the expert level. Basic level contained the introductory concepts of the course and the expert level had the most advanced concepts of the course. Questions were designed in such a way that those presented at the beginning tested the basics of the course whilst the questions presented at the end tested the complex concepts of the course. Each question was given a weight. The weights also reflected the level of the course being tested by the question, hence weights increased from first question to the last question. If a learner failed the first questions and subsequent ones he would be classified as a basic learner. Depending on how the learner performed in each section, together with other learning attributes, the learner was classified into an appropriate class level.

Step 4: Pointer to the appropriate level of notes and questions

Once a learner's class level was determined, the relevant learning information and subsequent section questions were highlighted. Reading time for the section notes was calculated. The learner had an option either to read the section notes or choose to answer questions only. In the former case, the learner was provided with one section quiz and classification attributes such as quiz time and scores were determined. For the latter case, two sets of section quizzes were provided, with the second quiz being more detailed.

Step 5: Determine new class level

Subsequent classification was carried out so as to determine the new class level for the learner and the relevant information was relayed to the learner.

Step 6

Steps 4 and 5 were repeated until the expert level was reached.

Step 7: Course evaluation

Upon fulfilling all the requirements for the expert level, both soft and hard copies of the evaluation questionnaires were provided and the learners' assessment of the system was captured.

System architecture

The model had four modules, namely the (1) learner module, (2) classifier agent module, (3) synchroniser agent module and (4) data storage module. All the modules were linked to work as one module as shown in Figure 1. The modules could also function independently as long as the database (data store) was available.

The learner agent module

The main function of the learner module was to facilitate the learning process both on- and off-line. This module made possible the interaction between the learner and the system

and was where the learner could register or login, access his or her profile details, get learning materials, read the notes, answer section quizzes and view all changes as they occurred. This module was connected to the data store and displayed information from the data store to both the learner and instructor.

There were two versions of the learner model namely, (1) the client model (off-line or stand-alone model) and (2) the server model (client-server or server-centric model). The client model was installed on the client machine (local machine) which was used whilst the learning was off-line. The server model was installed on the server machine (remote) and was used for online learning. It also facilitated profile updates by updating the off-line model whenever the internet connection was re-established.

The classifier agent module

The classifier

The classifier module or classifier DLL was a stationary agent that used the KNN algorithm to classify new learners for the first time and to do all other subsequent classifications of existing learners. The parameter K is an integer parameter representing the number of nearest neighbours to a new learner and whose most common class becomes the new learner's class. The default value of K can be fixed to an odd number such as 3. However, a low value of K restricts the classification of the new learner to classes of only a few neighbours. The best choice of K depends on the data and, in general, larger values of K reduce the effect of noise on classification but make boundaries between the classes less distinct. A good K is chosen using heuristic techniques such as cross-validation.

The parameter K should also be an odd integer number so that the majority vote is always attained. Even numbers for K can result in a tying vote that can hamper correct classification. For this research, K was 9. This figure was arrived at after considering that learners would be increasing with time and also to avoid restricting classification to a few training examples.

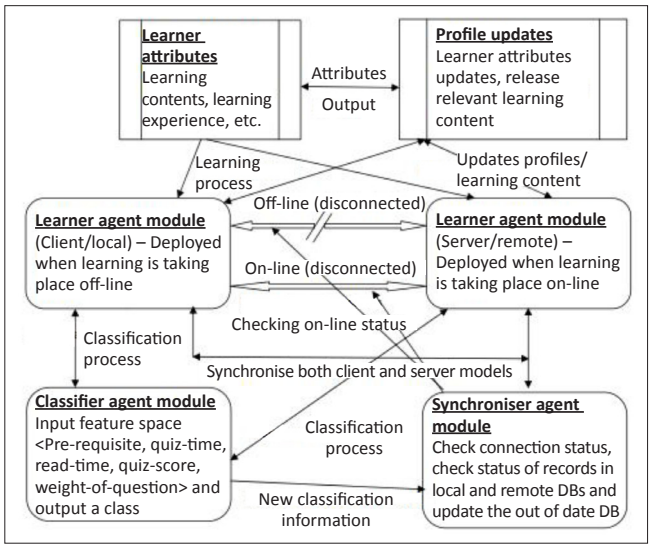
For the feature vector representing a learner, the KNN training data had features whose values could be combined in order to determine the class of the learner. If the attributes were too many, say more than 20 but where only two of them were relevant in the determination of the class of a particular query instance, the problem commonly referred to as the 'curse of dimensionality' is experienced.[4] The distance between training examples and the query instance is dominated by the large number of irrelevant attributes. To avoid this scenario, this research used five attributes namely, (1) score, (2) quiz time, (3) reading time, (4) prerequisite score and (5) weight of questions.

Determining factors in settling for these attributes

1. Score: Normally, students' performances are determined by their scores in examinations. All learning institutions use the scores as a major factor in grading their students. The higher the score a student attains, the higher the grade the student is awarded. In this research, we designed score ranges for all the classes, guided by the grading system used for undergraduate computing courses in the university where the study took place. The ranges and their classes are shown in Table 1.

2. Quizz time: This refers to the time the learner uses whilst taking a test. Normal practice is that examinations have a specific time allocated for their completion. When the allocated time is over, the learner sitting for the examination stops or, in the case of an online examination, is timed out. In this research, the system did not time out a learner if the set time was exceeded. Instead, the more time spent after the set time, the less the performance for the learner and the lower the class assigned. On the other hand, if the learner took less than the set time to sit for the examination, a higher the level of classification was assigned. A combination of both performance and time or other resources spent in achieving the learning is called learning efficiency[19] and is a measure of a learner's expertise. Expertise is higher for students who achieve a score with minimum effort compared with those who achieve the same performance after expending more resources.

3. Reading time: This is the time taken by the learner to read the learning material for the level or section. Normally, there is no limit in time for reading notes in preparation for an examination. A learner can take as long as possible to read the notes. In this research, a reasonable time threshold was set for reading the notes to enable both slow readers and fast readers to complete a topic. It was adjusted during testing of the course to make it appropriate.



Source: Authors' own construction

FIGURE 1: System architecture.

TABLE 1: Student classes and their corresponding score ranges.

Score Range	Class
0–39	Beginner
40–59	Intermediate
60–79	Advanced
80–100	Expert

Source: Authors' own construction

In considering this attribute, it was assumed that fast readers also perform better than slow readers, so the less time a learner took to read the notes, the better the performance that was achieved, leading to a higher class being assigned. A learner who took too much time beyond the threshold was assumed to be a slow learner and was assigned a basic class level.

4. Prerequisites: These were the conditions that a learner must satisfy before being allowed to proceed with the learning process. They included interest in studying the course. Whilst considering this attribute, the learner may either meet or not meet the conditions. This attribute cannot be used independently to classify a learner, but must applied in combination with other defined attributes for proper classification.

5. Weight of questions: The questions were weighted in an increasing manner from the first question to the last question. In addition, the questions were designed such that basic questions come first and complex ones come toward the end. Considering the design of the questions, it was prudent that basic questions were assigned less weight compared with complex questions. If a learner failed to answer basic questions correctly, it was assumed that he was still a beginner and was assigned to the beginner level class.

Course level

The class level was based on the experience of the learner. It was assumed for simplicity's sake that learners can be categorised into four levels, with the novices in the course being studied being referred to as 'Beginner'. The second category, we thought, should have a bit more experience, hence the label 'Intermediate'. 'Advanced' learners were the ones who had vast knowledge of the subject matter and 'Expert' learners were those who had the ability to apply the knowledge from the study. There were no particular criteria considered in coming up with these learner classes.

Choice of the model training data

The training data had two sections: the feature values and the target function values (i.e. the associated class) which was represented as a vector in the form $<a_1, a_2, a_3,\ldots_n> <T_a>$ where $<a_1, a_2, a_3,\ldots,a_n>$ was the feature vector and $<T_a>$ represented the target function value.

In this research, the training example vectors were defined as <prerequisite, score, readTime, quizTime, weight> <course level>. Table 2 has training examples demonstrating how the 2 vectors were populated.

TABLE 2: Sample training data based on the feature vector format.

Prerequisite	Score	ReadTime	QuizTime	Weight	Course level
1	91	40	15	8	Expert
1	75	47	10	9	Advanced
1	30	65	35	4	Beginner
1	50	50	20	7	Intermediate

Source: Authors' own construction

Given the training data, when a new learner joins with feature vector values such as <1, 45, 43, 10, 9> <?>, the KNN algorithm takes the new instance and compares it with the training data. The distances between the new instance attributes and the training data attributes are calculated.

The total distance of each new training example from the query instance is determined by summing all the attribute distances for the particular example. The closest nine neighbours were identified and the most popular class amongst these examples was assigned to the new instance. The assigned class was used to point to the relevant notes in the notes index and the notes were then displayed to the learner.

It is important to note that after classification of the new instance is carried out, the instance becomes part of the training data. The classifier agent receives data from the environment and after applying the KNN algorithm, classifies the learner and updates his or her profile dynamically. This agent is autonomous as it does not require any supervision and makes decisions depending on the prevailing information. This agent trains the model so that, based on the experience the model has with existing training data, it can classify new instances correctly.

The synchroniser agent module

The synchroniser agent or synchroniser DLL was also a static autonomous agent that synchronised the learner model contents for both the local and remote database. It collaborated with the classifier agent and learner module so that after the classifier agent had made changes with regard to the learner status, it made sure that learner's profile matched both locally and remotely.

The connection status of the models (client and server) was checked by the agent. The agent tried to establish a connection to the URL of the online application by using the public Internet Protocol address. Depending on whether the application was accessible or not, internet connection establishment was confirmed or failed. If connection establishment was confirmed then the remote version was used; if not, the local version was used. The connection of the model to the local and remote databases was checked. After establishing the connection status, status of the contents was compared. The status was determined by examining which database had more records and/or latest records. If the local copy was the latest, then the remote copy was updated and vice versa.

The module also displayed a message to a learner if there was no connection to the remote server, but allowed the learner to continue learning with a local copy which was later synchronised with the remote copy when the connection was reestablished. For synchronisation of both databases to take place, the synchroniser agent in the client machine located the domain address for the remote server and then connected to the database in the remote server. All records

were compared. The records of the side with more or the latest records were copied to the side with the missing data. This update was made per profile so that only the affected profile(s) were updated.

The following is the connection status testing algorithm:

Start
Client computer pings the address of the remote computer.
If there is a reply from remote computer, then connection status true,
Learning done using remote computer / server or online version,
Update the obsolete version of the database
Else connection status false,
Learning done using client computer / offline version
End

Results

The results of the research are explained in terms of the research objectives.

The first objective was to design a learner model that would use the KNN learning algorithm to get trained and to classify new learners.

Table 3 has 10 out of 30 new query data items that were classified using KNN.

The experiment was carried out with 30 students and the results were captured as shown in Table 3. Zero was used to indicate a query data that was classified incorrectly whilst 1 was used to show query data that was classified correctly. Of the 30 learners studied, only 5 learners were classified incorrectly. The percentage accuracy was:

$$^{25}/_{30}\ x\ 100 = 83.3\% \qquad \text{[Eqn 1]}$$

In this model, attributes were defined that were used to train the classifier so that when the classifier was presented with a new learner represented by a vector of attribute values, the classifier classified him intelligently. In most existing LMSs, learners just read the learning materials at their own pace. There are no checks put in place to determine if the reading is taking place or not. In this model, however, a note of reading time for notes and time taken by the learner to do the quiz is taken and these times contribute in both classification and subsequent profile updates.

The second objective was to make sure that learner profiles were updated as the learners continued with the learning process and relevant learning information was displayed to them, based on their profiles.

After the experiment was carried out, the learners were given hard-copy questionnaires. Table 4 shows a summary of the answers provided by the learners. The survey collected information related to the first and second objectives to double-check results from the logging of the learner activities.

The results show that an overwhelming majority of the students indicated that they were able to learn both on- and off-line. A great majority of the students also indicated that it was easy to learn with the system, that they could recommend the system to others, that they were able to get appropriate notes and that they were classified fairly. The questions and prerequisite questions were also well designed. The students also indicated that their profiles were updated and that the timings for the course were appropriate. The highest percentage of agreement was 100% and the lowest affirmative percentage was 60%.

Discussions and Conclusions

In considering the research objectives and other issues of interest, together with the results from the study, a number of conclusions were made.

The first objective was regarding developing the classifier module to classify learners appropriately. From the percentages of the learners that were classified correctly, it could be concluded that the model was accurate in classifying learners, with an accuracy of 83.3%. Likewise, from the survey results, 27 out of 30 (90%) learners said that they were classified as per their expectations.

The second objective was to ensure that the learners' profiles were updated. From the questionnaire results, this objective was achieved since 25 out of 30 (83%) learners stated that their profiles were updated. This is also seen from the log of updates of the learner class from 100 (beginner) through 200 (intermediate) to 300 (advanced) and 400 (expert). A sample of the logs is shown in Table 5.

TABLE 3: Sample results showing how learners were classified.

Student ID	Prerequisite Score	Read Time	Quiz Time	Weight	Output (Class)	Correctly classified
10 011	10	50	31	3	Beginner	1
10 020	40	60	30	4	Beginner	1
10 031	90	43	19	9	Expert	1
10 040	47	65	28	5	Beginner	0
10 051	39	54	26	7	Intermediate	1
10 061	65	45	15	9	Advanced	1
10 071	100	30	8	8	Expert	1
10 081	90	32	10	10	Advanced	0
10 091	95	40	20	8	Expert	1
10 100	100	29	6	9	Expert	1

Source: Authors' own construction
Note: The column 'correctly classified' has value 0 for NO and 1 for YES

TABLE 4: Summary of responses to questionnaires after the learning process.

Question	Question content	YES	NO
1	Would you recommend this learning model to someone else?	27	3
2	Is this learning model easy to use?	28	2
3	Were you able to learn online?	30	0
4	Were you able to learn off-line?	22	8
5	Do you think you were classified fairly?	27	3
6	Were you able to get appropriate notes?	28	2
7	Were the questions well designed?	26	4
8	Were the prerequisite questions appropriate?	30	0
9	Were the timings (time allocated) appropriate for all sections?	18	12
10	Did your profile get updated?	25	5
11	Did you face any challenges that relate to the research objectives?	3	27
12	Do you have recommendations that relate to the research objectives?	6	24

Source: Authors' own construction

TABLE 5: Learners' History Report for one course unit.

Student ID	Classified to	Date
8888	100	Wed Jan 19 13:28:52 EAT 2011
8899	400	Wed Jan 19 13:28:52 EAT 2011
8877	400	Wed Jan 19 13:28:52 EAT 2011
8866	100	Wed Jan 19 13:28:52 EAT 2011
8855	300	Wed Jan 19 13:28:53 EAT 2011
8833	300	Wed Jan 19 13:28:53 EAT 2011
8822	100	Wed Jan 19 13:28:53 EAT 2011
8811	200	Wed Jan 19 13:28:53 EAT 2011

Source: Authors' own construction

The third objective of this research was to enable the learning process to take place both on- and off-line. From what other scholars have performed, it is evident that all learning systems and/or models are developed as applications which are deployed to be used online by end users. The end users must always be online to do the learning.

In this research, a model was developed which is an API, packaged as a DLL and which can be used as a service by other LMS developers and users. It can be used either independently or integrated with other LMS. A master API was developed which was installed in a server to be accessed by the clients' APIs which were located remotely. The client APIs had two versions of API: the DLLs for off-line learning and HTTP protocol for online learning.

The model used for this research had a functionality that could detect internet connection and then connect to the server version of the model. The user first logged into the model on the client machine. He was then able to continue learning even if there was no internet connection. A replication of the database took place whenever there was an internet connection. Whichever of the databases had the more updated information was then replicated in the database along with an older version of the information. This property enabled the learners to be able to do their learning seamlessly whether an internet connection was established or not. This has been shown in the summary of their responses to the post-experiment survey.

User profiles were also updated, so when the internet connection timed out, both databases were on a par in terms of learning and learner information. The learners could thus go on learning off-line but with the most up to date profile and learning information. This contributed to the continued learning of the students even under conditions of intermittent internet connections.

Limitations and Challenges

- Classification was only based on the training data attributes and how close the attributes were to those of already classified learners. Other learners' characteristics are not considered. Bearing in mind that the KNN algorithm usually classifies data using up to 20 attributes, the five attributes as used in this research were possibly not enough to give conclusive results.
- The study was conducted over six hours a day for five days. This was not a long time, but was dictated by other factors, such as the available duration of time for the research. The amount of data available for use during classification was therefore also limited. It would have also been ideal to observe changes in learner knowledge levels over an entire semester instead of over just five days of intensive work.
- Only the KNN algorithm was used for the experiment. It would have been better to have been able to compare its performance against that of other algorithms.
- During the testing stage of the model, not all stakeholders were involved due to time constraints. It would possibly have been a good idea to hold off and let them participate later so that the model could gain a wider audience acceptance.

Future work

The authors would like to note the following for the intended users. As it is, this model is installed both on the server and client machine as separate entities after which the client version accesses the server version for updates. In case the learner does not have the application he will only be able to use the online version. An online downloadable version should be available for installation by any interested learner from anywhere in the world. This way, the learner can download the system and continue learning from any machine, especially in the case of travelling from one place to another. This is because whenever there is internet connection, updates to the learner profile and learning activities are logged into the online server and the server will therefore always contain the most up-to-date information about the student's learning.

This application has been developed using Java language, which is a resource-intensive language requiring higher specification computers for efficient running. This, however, may not be possible with everyone who might be interested in learning in this model. Therefore, it is recommended that research should be carried out to enable the development of the application with a lighter programming language.

Acknowledgements

The authors thank the editors and reviewers of the International Journal of Machine Learning and Applications for their constructive comments on this article.

Competing interests

The authors declare that they have no financial or personal relationship(s) that may have inappropriately influenced them in writing this article.

Authors' contributions

M.G.W. (Dedan Kimathi University of Technology) was the main researcher and O.O.R. (University of Nairobi) and N.O.H. (University of the Western Cape) provided guidelines during the research process

References

1. Sampson D, Karagiannidis C, Kinshuk D. Personalised learning: educational, technological and standarisation perspective. Digital Education Review. 2002;4:24–39.
2. Blochl M, Rumetshofer H, Wob W. Individualized e-learning systems enabled by a semantically determined adaptation of learning fragments. Paper presented at DEXA 2003. Proceedings of the 14th International Workshop on Database and Expert Systems Applications; 2003 Sep 1–5.
3. Pai WC, Wang CC, Jiang DR. A software development model based on quality measurement. Paper presented at ICSA 2000. Proceedings of the ICSA 13th International Conference; 2000. p. 40–43.
4. Wang Y, Wang ZO. Paper presentation. A fast KNN algorithm for text categorization. Proceedings of 2007 International Conference on Machine Learning and Cybernetics; 2007. Vol. 6, p. 3436–3441. http://dx.doi.org/10.1109/ICMLC.2007.4370742
5. Srihivok A, Intrapairote A. A conceptual framework for e-learning in the tertiary education in Thailand: Report of the National Council Research of Thailand; 2003.
6. Ismail J. The design of an e-learning system: Beyond the hype. The Internet and Higher Education. 2001;4(3–4):329–336. http://dx.doi.org/10.1016/S1096-7516(01)00069-0
7. Sun PC, Tsai RJ, Finger G, et al. What drives a successful e-learning? An empirical investigation of the critical factors influencing learner satisfaction. Compu Educ. 2008;50(4):1183–1202. http://dx.doi.org/10.1016/j.compedu.2006.11.007
8. Mulwa C, Lawless S, Sharp M, et al. Adaptive educational hypermedia systems in technology enhanced learning: a literature review. Paper presented at ACM 2010. Proceedings of the 2010 ACM Conference on Information Technology Education; 2010. p. 73–84.
9. Popescu E, Trigano P, Badica C. Towards a unified learning style model in adaptive educational systems. Paper presented at ICALT 2007. Proceedings of the 7th IEEE International Conference on Advanced Learning Technologies; 2007, July 18-20. p. 804–808.
10. Saleh AA, El-Bakry HM, Asfour TT, et al. Adaptive e-learning tools for numbering systems. Paper presented at ACE 2010. Proceedings of the 9th WSEAS International Conference on Telecommunications and Informatics; 2010 May 29–31. p. 293–298.
11. Atolagbe TA. E-learning: the use of components technologies and artificial intelligence for management and delivery of instruction. Paper presented at ITI 2002. Proceedings of the 24th International Conference on Information Technology Interfaces; 2002. p. 121–128.
12. Zaíane OR. Building a recommender agent for e-learning systems. In. Paper presentation. Proceedings of International Conference on Computers in Education; 2002 Dec 3–6. p. 55–59.
13. Mungunsukh H, Cheng Z. An agent based programming language learning support system. Paper presentation. Proceedings of the International Conference on Computers in Education; 2002 Dec 3–6. p. 148–152.
14. Lima RM, Sousa RM, Martins PJ. Distributed production planning and control agent-based system. Int J Prod Res. 2006;44(18-19), 3693–3709. http://dx.doi.org/10.1080/00207540600788992
15. D'Inverno M, Luck M. Understanding agent systems. Berlin: Springer-Verlag, 2001.
16. Pacheco O, Carmo J. A role based model for normative specification of organized collective agency and agents interaction. Auton Agent Multi-Ag. 2003;6(2):145–184. http://dx.doi.org/10.1023/A:1021884118023
17. Wooldridge M. An introduction to multiagent systems. Cambridge: John Wiley & Sons, 2009.
18. Nienaber R, Cloete E. A software agent framework for the support of software project management. Paper presented at SAICSIT 2003. Proceedings of the 2003 annual research conference of the South African institute of South African Institute for Computer Scientists and Information Technologists on Enablement through Technology; 2003. p. 16–23.
19. Van Merriënboer JJG, Sweller J. Cognitive load theory and complex learning: Recent developments and future directions. Educ Psychol Rev. 2005;17(2):147–177. http://dx.doi.org/10.1007/s10648-005-3951-0