What Will Be in Those Lap Tops: Empowering Students and Teachers to Add Content to an Educational Chatbot's Knowledge Base

Patrick K. Bii1,*, J. K. Too2

1Department of Mathematics and Computer Science, University of Kabianga, Kenya
2Department of Curriculum Instruction and Educational Media, School of Education, Moi University, Kenya

Abstract The Government of Kenya has in the recent past been on a serious footing to put in place ICT initiatives and projects aimed at realizing the potential within its recognition of ICT as a foundation for a knowledge economy as per vision 2030. The tempo has risen several notches higher with the current intention and initiative of making available laptops to pupils and students in primary schools in the years to come. However, once computer hardware is available to schools as PC’s, Laptops and other mobile computing devices, a critical issue which must be adequately addressed is the educational content within. This paper outlines a possible approach using open source software development concepts and the Participatory Action Research (PAR) model as a means of empowering students and teachers to become active developers of the content they consume.

Keywords ICT, Educational Software, Participatory Action Research, Chatbot

1. Continuing Efforts to Supply Computers for Educational Purposes

The Government of Kenya has in the recent past been on a serious footing to put in place ICT initiatives and projects aimed at realizing the potential within its recognition of ICT as a foundation for a knowledge economy as per vision 2030 [1, 2, 3]. The tempo has risen several notches higher with the current intention and initiative of making available laptops to pupils and students in primary schools in the years to come [4, 5, 6, 7]. Realistic cognizance of possible challenges to this initiative (and hence possible ways of surmounting the challenges) is pertinent, for as Anne [8, p1] has noted, “ever since the new government announced in their manifesto that they will be deploying laptops to standard 1 children in schools across Kenya, the big question on everyone’s mind is how will this program work? Can the country afford it? What would it take to make the idea work?” Some of the challenges include cost of hardware, cost of software, connectivity, infrastructure, pupil competencies, teacher competencies, school policies, administrative structures and cultures, hardware and software support systems, success metrics, security considerations, educational content, acceptance by teachers and society, danger of misappropriations, sustainability and business and commercial interests [4, 6, 8, 9, 10, 11]. This paper focuses on the critical issue of availing what will be in the laptops and other computers meant for educational use – educational software and content.

2. The Challenge of Availing Educational Software

Cautions to be considered in attempts to avail educational software to schools include not seeing the issue in terms of availing hardware alone, what software will be availed, the educational content of the software, scarcity of digitized content relevant to approved curriculum, and development of software that can extract content from school textbooks [6, 8]. Further, Hepburn [10, p1] stated that

An unavoidable part of making ICT available in schools is obtaining and maintaining the software that is necessary to allow school computers to function. Most software that schools use is produced by proprietary software companies that normally charge considerable sums of money for their products. The cost and usage restrictions that characterize proprietary software place an enormous stress on cash strapped schools. As a result of this situation,
What Will Be in Those Lap Tops: Empowering Students and Teachers to Add Content to an Educational Chatbot's Knowledge Base

schools are left with a serious problem: they clearly need to integrate ICT into teaching and learning but doing so requires large, ongoing expenditures to purchase and maintain ICT resources.

Hepburn [10] enumerates challenges of using proprietary software as high cost, restrictions on flexibility of use due to licensing constraints, and ethical and social issues including equity and the moral one of exposing students to and training them on particular companies’ software while the students pay the proprietary companies to do so.

3. A Suggested Approach to Availing Educational Software Content

A possible way out for schools and other institutions of learning to the software issue is the use of open source software, software distributed with a license granting access to source code, distribution, modification and free use [12, 13, 14, 10, 11]. Compared to proprietary software, open source software is less costly, offers greater flexibility of use, and is in a position to address the social issues of equity and corporate involvement [10]. This approach has borne fruit in the case of a number of open source software initiatives including the Linux operating system, OpenOffice and LibreOffice, and chatbot ALICE [15, 16, 17, 18]. If a similar approach is made possible for educational software content development, the pool of developers, development, and continuity potentially is unlimited since this approach opens up the software to a large community to become involved in the development effort, allowing rapid bug fixes and enhancements to occur. The potential developers of interest here are students, teachers, lecturers, educational resource persons and other interested developers. The suggested specific content to be developed by this pool of developers is the educational chatbot’s knowledge base. This open source approach coupled with the Participatory Action Research model, illustrated in Figure 1, has a possible chance of bearing fruit.

In Participatory Action Research, there is an explicit intention to educate and co-produce change with the collaboration of those affected by the issue being studied [19, 20]. The PAR process involves participants in ‘planning action (on the basis of reflection); in implementing these plans in their own action; in observing systematically this process; and in evaluating their actions in the light of evidence as a basis for further planning and action, and so on through a self-reflective spiral’ [21, p317]. The steps in practice may overlap and can begin at any point.

4. Adding Content to An Educational Chatbot’s Knowledge Base

A chatbot is a computer program that is created to simulate intelligent human language interaction through text or speech and whose purpose is to engage in conversation or to emulate informal chat communication between a human user and a computer using natural language [23, 24, 25]. They can be created using various computer programming languages including PHP, XML, JAVA, C++, Python and AIML. Two approaches can be used in developing a chatbot. One approach is to start with an empty database to which content is automatically added as it is used while the other approach is to have the chatterbot creator program the database so that it has pre-programmed questions, phrases or words and how it is to respond to each question, phrase or word [26]. Chat-logs created during interaction sessions additionally serve as sources for chatbot response improvement [27, 17, 25, 28]. In AIML, the pattern is the user's expected or assumed question (the matching part) while the template is the chatbot's prepared or programmed answer (the returning part). The AIML pattern syntax was designed as a very simple pattern language, but greater complex response can be achieved through use of a few more AIML tags which can, as Wallace [34, p1] states, “transform the reply into a mini-computer program which can save data, activate other programs, give conditional responses, and recursively call the pattern matcher to insert the responses from other categories.” A further simplification to AIML file creation is possible through use of MakeAiml Editor, created by Dryden [23] and which enables a programmer to reduce the amount of work required to write AIML files through providing a shorthand way to represent AIML tags and their contents [23]. For example, the AIML file

```xml
<aiml>
  <category>
    <pattern>Hello</pattern>
    <template>Hi! How are you?</template>
  </category>
</aiml>
```
typed in notepad can be typed in more simply as

\[ \text{p Hello} \]
\[ \text{t Hi! How are you?} \]

in MakeAiml. In other words, one need only specify the pattern (p) and template (t) pairs as illustrated above in creating an AIML file.

In order to add content to an educational chatbot’s knowledge base, the community of students, teachers, resource persons and interested developers can be tasked to program the database of the educational chatbot with educational content, including questions and answers on school subject content, teaching-learning notes typed into the chatbot directly by students and teachers or sourced from open source e-books, and digital content availed to schools. This has the potential of enhancing interaction and collaboration between students, teachers, resource persons and learning institutions, key requirements of an educational environment that can enable students acquire 21st century skills [29, 30, 31]. Such an approach could be used to develop individual, class, school, regional, national and even international knowledge bases for educational chatbots.

In a specific school, for example, the teacher when teaching a given topic may request students to group themselves according to the number of computers available in the school’s computer laboratory or their class, and then program their group’s chatbot with content, questions, answers, keywords and definitions concerning the topic being covered. Ideally, the chatbot starts blank without appropriate content, keywords, and responses to questions pertaining to the topic under study. The expectation is that by the end of the programming sessions, the chatbot should be able to respond intelligibly to the teaching-learning topic. The programming sessions could be undertaken at arranged times as per the school’s time tabling requirements. Upon completion of chatbot programming, the student group chatbots are rated on comprehensiveness of content and responses made to questions it is asked pertaining to the topic.

5. Knowie: A Concrete Example

A chatbot named Knowie was developed by the present authors with such use in view. The bot’s knowledge comes from AIML files. AIML consists of the elements, categories, patterns, and templates. A category is a basic unit of knowledge in AIML. It consists of an input question (pattern or stimulus), an output answer (category or response), and an optional context (topic, that). Implementation software for the technology is open source: Ubuntu Linux, Python, JDK, PyAIML, and MakeAiml. The chatbot Knowie was derived from the open source chatbot Howie, originally created by Stratton [32]. Knowie’s basic interface and sample interaction are presented in Figures 2 and 3 respectively. When the user launches the chatbot, it presents clickable options (Figure 2) including Directions (user help feature to assist the user to get started in interacting with the chatbot), New User (for a first-time user), Continuing User (for a user who has earlier interacted with the chatbot and given the chatbot their name), Teach Me (request by the chatbot for the user to teach it), Type Notes/Input Content (for the user to type in learning notes or input content from e-books), View Notes (for the chatbot to display content notes that has been input by the user into it), View Question-Answer pairs (for the chatbot to display the question-answer pairs it has and which correspond to AIML pattern-template pairs), and a Quit option. The buttons on the lower section are Edit Notes (to enable user to make changes to the notes the chatbot has), Edit p-t pairs (for question-answer pair editing), View Currently Unanswered Questions (to enable the user see the questions that the chatbot has been asked during interaction sessions and for which it has no answer i.e. corresponding template), Edit Currently Unanswered Questions Log (make changes to the log of questions that the chatbot does not have answers to). The grayed out buttons (Question, Answer, Keyword, Definition, Submit) become active once the user begins interacting with the chatbot. Figure 3 shows the chatbot’s interface during an interaction session. The user is a New User, the user types in their part of the interaction for the bot to accept inside the YOU text field, and the chatbot displays the interaction session exchange (what the user says and what it says in response) on the upper and lower display areas.
What Will Be in Those Lap Tops: Empowering Students and Teachers to Add Content to an Educational Chatbot's Knowledge Base

Figure 2. Knowie Basic User Interface
If the user selects Teach Me, the Question, Answer, Keyword, Definition and Submit buttons become active. The user can then program the chatbot by typing in a question (pattern) and the corresponding answer (template) and then submit them to the chatbot to incorporate as part of its knowledge base. The Keyword and Definition buttons serve the same purpose, but in this case, the user types in a keyword (for example Computer) and the appropriate definition of the keyword. Upon submission of the Keyword-Definition pair, the chatbot will proceed to form pattern-template pair combinations from the two using as many as possible ways of asking questions about the submitted keyword, enabling it to generate an answer based on the definition of the keyword and whose answer is the definition supplied by the user. The search (Knoogle) button allows the user to type in a keyword or phrase and submit to the chatbot to retrieve and display any content material that it has about the search term. If the keyword being searched for is found to have a definition, then the appropriate pattern-template pairs for it are also generated, enabling the chatbot to latter respond to a question asked about the keyword by the user.

The chatbot’s underlying structure and architecture is illustrated in Figure 4. The GUI Chat Interface registers user
input and displays user response. The user interface acts as
the front-end, allowing the student to communicate with the
chatbot. It also enables the student to input notes, save notes,
view notes, edit notes, search notes, input question (pattern)–answer (template) pairs, view and edit pattern-template
pairs, input keyword-definition pairs, and edit keyword-definition pairs. The GUI also displays the
questions that the chatbot currently does not have an answer
to as an aid for further targeted response improvement. The
AIML Interpreter performs user text input processing and
generates the bot’s response. The interpreter acts as a link
between the user and the knowledge base of the bot. It takes
the user input, passes it through a normalization process,
looks for matching patterns in the database, extracts a
suitable response from the database and displays it to the user.
The AIML Knowledge Base acts as the back-end which
stores the bot’s knowledge. Knowledge Base Sources are the
various ways in which one can add knowledge into the
chatbot’s knowledge base and the various online resources
the chatbot can access in order to add to its knowledge.

Figure 4. Chatbot Knowie’s underlying structure and architecture
The foregoing discussion indicates that it is possible to transfer chatbot technology use to students, teachers, resource persons and other interested developers because of the inherent simplicity involved in AIML design and use. This means that the power of Artificial Intelligence is within reach of the normal classroom environment and such use by teachers and students can be a way to begin involving the key players in the teaching and learning environment - students and teachers - in active development of the content they use in schools. Searches similar to that offered by internet search services is possible, and with further development efforts, can extend to live comprehensive online internet searches.

6. Conclusions

Once computer hardware is availed to schools as PC’s, Laptops and other mobile computing devices, a critical issue which must adequately be addressed is the educational software content within. This paper has outlined a possible approach using open source software development concepts and the PAR model as a means of empowering students and teachers to become active developers of the content they consume.

REFERENCES


