The Mobile Fact and Concept Training System (MoFaCTS)

Philip I. Pavlik Jr., Craig Kelly, Jaclyn K. Maass

Institute for Intelligent Systems, University of Memphis, Memphis, TN 38152, USA
ppavlik@memphis.edu, cnkelly@memphis.edu, jkmaass@memphis.edu

Abstract. The effectiveness of Intelligent Tutoring Systems (ITS) research is enhanced by tools that allow researchers to quickly bridge the divide between theoretical and applied work. By providing a common infrastructure to test cognitive and learning science theories in authentic contexts with real students, the Mobile Fact and Concept Training System (MoFaCTS) can aid in accelerating ITS research and real world implementation. MoFaCTS is run from a web browser and allows the teacher or administrator to set up a sequence of units of content. Because the "optimal practice" module is interchangeable, the system allows for the comparison of alternative methods of adaptive practice. To foster faster research progress, data export supports the DataShop transaction format, which allows quick analysis of data using the DataShop tools. Integration with Amazon Turk allows quick and efficient data collection from this source.

Keywords: intelligent tutoring systems, e-learning, instructional design

1 Introduction

MoFaCTS was based on the FaCT system, which was created to make faster progress on laboratory research and its translation to the classroom [1]. MoFaCTS is the latest implementation of the FaCT system, which has new features in addition to running in HTML5, which provides mobility to any common web browser. The framework of MoFaCTS is based on an implicit theory of "chunk" learning [2] which assumes that learning of chunks occurs through discrete "trials" (e.g. a single step problem or fill-in-in the blank sentence). As such it departs from the tradition of model tracing tutors [3], which focus on multistep problems of greater complexity, where the student is learning a sequence of rule applications. The simplified chunk-based approach in MoFaCTS allows the system to focus more clearly on the problem selection aspect of tutoring, and how the selected sequence can be improved. Moreover, from the beginning the system was designed without strong assumptions about the optimal schedule. Because of this the system is easy to adapt to the needs of specific projects. Screenshots of the system in action, see Figure 1, show a variety of functions, including multiple choice responding, fill-in-the-blank responding with branched feedback, and image-stimuli items.

adfa, p. 1, 2011.

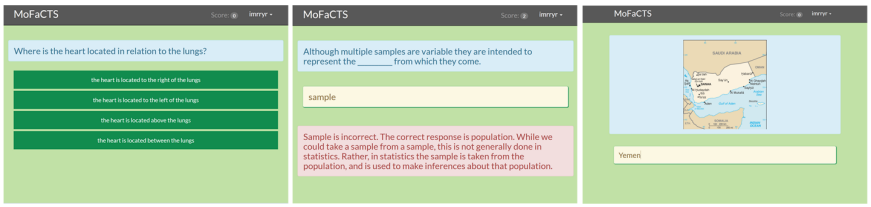

Fig. 1. Example screenshots.

2 Client/ Server Architecture

MoFaCTS was built using Meteor, a framework based on Node.js which uses a single programming language (JavaScript) for both the client and server logic [4-6]. Communication between the two sides of the architecture is handled transparently by the framework. This architecture conveniently off-loads any complex computations needed to compute practice schedules to the client machine, which allows much larger numbers of users to interact with the system simultaneously.

Since the web is a popular platform for content and application delivery, MoFaCTS is able to leverage a vast body of open source software. Currently this includes the Bootstrap CSS framework developed by Twitter [7]. Bootstrap provides MoFaCTS with a typographic and layout framework. Most importantly, Bootstrap contributes responsive web design, so content appears correctly on both desktop web browsers and mobile devices [8]. The HMTL displayed to the user is generated via Meteor's templating system. This system uses a style known as "reactive programming" [9]. A piece of code can change data on the server and trigger a change on the client. This style of programming works well with the frequent switches in display necessary when sequencing multiple educational content objects.

The server portion of MoFaCTS runs on the Node.js JavaScript server, which gives it access to all of the libraries and asynchronous communication abilities of Node.js. In addition, the Meteor framework simplifies the server code required for an application of this type. When the client contacts the server, Meteor transparently provides data transport, call batching, and automated retry. This functionality leads to excellent performance without the need for more complex software. This architecture is particularly useful given the need for frequent logging to maintain the database of users' learning histories. This MoFaCTS data is stored in a document-oriented database system named MongoDB [10]. In addition, MongoDB is schema-free, which allows for rapid iteration when designing new features or upgrades, which is often a great advantage in research.

3 Functionality

MoFaCTS has two primary unit types, learning and assessment, which define its two main modes of application. Both kinds of units are specified in the control file for each "tutor", which is called the tutor definition file (TDF). Each tutor definition file begins with a number of preliminaries, including the initial randomization commands. These randomizations allow the specification of shuffle and swap commands. The shuffle

command generates random orders within groups of specified sequences (e.g. "<shuffle> 0-5 6-11" shuffles the first 6 items among themselves and then shuffle the next six items among themselves). The swap command randomizes the order of those specified sequences (e.g. "<swap> 0-5 6-11" randomizes the order of the groups of sequences, so in this simple case it would be either 0-5 6-11 still or 6-11 0-5, while retaining the order of the subsequences). By running a shuffle and then a swap command, complex distributions of stimuli across conditions can be achieved. To enable comparisons of different assessment or learning conditions, the system also automatically randomizes into any number of between-subjects conditions. This choice is recorded in the data for each subject, and reinstated when they begin new sessions from the same root TDF, so multi-session between-subjects comparisons with counter-balancing are easily enabled.

3.1 Units

The first main type of unit is the assessment unit, which allows for complex schedules of content, where the TDF author has specified the number of repetitions and the location in the sequence for each repetition of each item. Each repetition may be a test with or without feedback or a passive study opportunity. Assessment units may be used for quizzes in a classroom setting or for experiments looking at practice, forgetting, learning, and/or recall. In an experimental context, the system allows additional sequence level randomization, to make sure blocks of the same items are individually randomized, so that spacing conditions are not predictable. Any number of assessment units can be strung together, which allows pretest, practice and posttest portions to be organized individually to compose a larger experiment. Because the data architecture (described below) saves the state of the learner at all times, assessment sessions are automatically resumed where they were previously stopped, allowing for multiple experimental sessions for the same experiment over days or weeks.

The system also allows the specification of units with dynamic scheduling based on a select function. These units sequence the items according to a mechanism in the select function. This select function could be based on any sort of model of the learning and/or pedagogical rules. Typically, the adaptive learning module would be some version of Pavlik's optimal learning method [11], which uses a computational model of memory to infer the best item to practice next.

Although assessment and learning units can both provide brief instructional screens prior to practice, an "instructions unit" presents only instructions, with a continue button to move to the next unit. These instructional units can also be configured with a between-subjects randomization into a "lockout condition" where the instruction screen has an active timer that only allows continuation after a specified amount of time.

3.2 Supported Practice Types

The system supports two forms of test items: the multiple choice items (which appear in button form for touchscreen responsiveness) and the short-answer items. For multiple choice items, the system allows the researcher to randomly display the order of two or more answer options. These answer options may be specified for each individual question, or be randomly selected from a larger "answer bank." Feedback can be displayed after incorrect responses. This feedback displays the correct answer for a fixed period

of time or until the user hits the spacebar, as specified in the TDF. If the trial is a short answer item (multiple choice branching is in development), more complex branching feedback is enabled which compares the response with a number of wrong responses, each of which has specific feedback text in the stimulus file. This allows the system to provide feedback tailored to particular response errors, hopefully promoting conceptual learning by directly challenging misconceptions in the student's model of the domain.

Since both the system and the user may be frustrated and deterred in their goals by an incorrectly marked response, the system provides a few ways to identify correct responses with some flaws or ambiguity. These include partial matching using regular expressions, simple Levenshtein proportion errors, or Levenshtein proportion for multiple synonyms. Each of these methods offers different advantages depending on the test type. Regular expressions allow answer specification to pick up the presence of key words for short answer responses, to automatically score relatively complex responses (see the Circulatory System example below). Levenshtein proportion marks an item correct if some proportion of the letters are correct (e.g. 75%).

Finally, a passive viewing trial type simply presents the stimulus (text, audio, or image) for a fixed number of seconds or until the user hits the spacebar. Normally, a fixed time is used, since unless the user population is intrinsically motivated, the students or participants may truncate these study trials, reducing (possible) learning.

3.3 Datashop Export and Amazon Turk Integration

The system provides native export to the PSLC DataShop tab-delimited format style with several custom fields. This functionality means that data collected in the system can be immediately imported into DataShop for analysis, storage, and/or presentation [12]. As part of the new LearnSphere project the DataShop is being expanded to include a graphical workflow analysis tool with multiple methods (http://learnsphere.org/). MoFaCTS users will be able to take advantage of these resources immediately. Further, there is a library of prior analyses already shared within the community for DataShop formatted files (https://pslcdatashop.web.cmu.edu/ExternalTools).

The system provides integration with Amazon's Mechanical Turk (MTurk) service. This integration was added to ease the administrative burden often encountered when running experiments with large numbers participants recruited via MTurk. A researcher can oversee the experiment via a management screen within MoFaCTS that shows the current progress of all participants. From the same screen, the researcher may approve payment for a participant's work and/or pay a post-payment bonus. If using the "lockout conditions" discussed previously, researchers may craft an automated message that the system will send to Mechanical Turk users when their lockout expires (e.g., email a reminder after a one-week retention interval).

4 Research using MoFaCTS

Described here are three recent experiments (one published as a dissertation, one in preparation, and one submitted for publication, respectively), using MoFaCTS with different experimental designs and stimuli. These large complex experiments demonstrate how flexible the system is for different tasks and goals.

This study assessed the effects of spaced practice on the ability to identify musical intervals. A total of 187 individuals from both a psychology subject pool and MTurk completed a pretest and then practiced identifying six musical intervals, with two musical intervals each randomly assigned to narrow, medium, and wide spacing for each individual. During this practice, the musical intervals were presented at two tone levels and were played as either harmonies or melodies. Participants were randomly assigned to return for a posttest 2 min, 1 day, or 7 days later. All individuals received a posttest of the same six musical intervals from practice at the same tone levels as practice and at a transfer tone level. The posttest also contained both harmonic and melodic trials.

A second experiment which utilized several features of MoFaCTS, presented participants with retrieval practice on questions about the circulatory system. A total of 178 participants, recruited through MTurk, completed the experiment producing valid data. Participants read a text (about the circulatory system), completed a retrieval practice session, and took a posttest. They were randomly assigned to practice retrieval in one of four conditions from a 2 (question depth: factual, applied) x 2 (answer format: multiple choice, short answer) between-subjects design. Practice consisted of a total of 32 trials (eight questions repeated four times each), followed immediately by 16 posttest trials (16 questions, not repeated). Each practice trial received immediate corrective feedback. MoFaCTS was able to score the short answer responses immediately by matching user type-ins to key words specified via regular expressions. This method was flexible enough to allow us to account for common synonyms and misspellings discovered through pilot testing. After practice, a posttest assessed repetition performance and transfer to a different format, a different depth, and previously unpracticed concepts.

A third experiment involved an arguably even more complex design, which replicated and extended prior work [13], in addition to testing refutation and long-term retention. In this experiment approximately 450 MTurk users filled-in blanks for a collection of 18 fill-in-the blank sentences about statistics to produce complete data. The experiment used a 2x3 between-subjects design with 3 levels of retention interval (either 2 minutes, 1 day or 3 days between 2 sessions of practice) and with 2 levels of feedback (either simple feedback of the correct fill-in or refutational feedback for a portion of the wrong answers). The within-subject design for the experiment crossed 3 levels of spacing (narrow, medium, or wide) with 3 levels of practice repetition (either 2, 4 or 8 repetitions) with 2 levels of fill-in variability during practice (same or random fill repetitions) and 2 levels of fill-in variability during posttest (same or random fill-in repetitions for each of the 9 items in each condition). Order of introduction (random or fill-in first) for each scheduling condition was counterbalanced in addition to using two different schedule orders, either starting in order from the beginning or from the end (i.e. in reverse) of the schedule. Additional sub-sequence randomization was used to prevent exact repetitions of the spacing of conditions from cueing recall. Posttest practice order tested each of the 18 items in random order with their respective response variability condition for 3 rounds of testing.

5 Conclusions

MoFaCTS was create as a research tool to investigate the effect of instructional sequence manipulations. The system is released on bitbucket.org as open source software

(https://bitbucket.org/ppavlik/mofacts/overview). As development continues we welcome collaborators in building this research accelerator of research. With this in mind, continued development will focus on not only increasing the capabilities in regard to different and more complex types of trials or problems, but also on making the process of creating a student model more streamlined so as to encourage the development of multiple options for student models to explain and control practice in the system.

6 Acknowledgements

This work is supported by the National Science Foundation Data Infrastructure Building Blocks program under Grant No. (ACI-1443068) and the University of Memphis Institute for Intelligent Systems.

7 References

1. Pavlik Jr., P.I., Presson, N., Dozzi, G., Wu, S.-m., MacWhinney, B., Koedinger, K.R.: The Fact (Fact and Concept Training) System: A New Tool Linking Cognitive Science with Educators. In: McNamara, D., Trafton, G. (eds.): Proceedings of the Twenty-Ninth Annual Conference of the Cognitive Science Society, 1379–1384. Lawrence Erlbaum, Mahwah, NJ (2007)
2. Johnson, N.F.: The Role of Chunking and Organization in the Process of Recall. The psychology of learning and motivation: 4**,** 171-247. (1970)
3. Anderson, J.R., Pelletier, R.: A Development System for Model-Tracing Tutors. Proceedings of the International Conference of the Learning Sciences, 1-8. Evanston, IL (1991)
4. Meteor Development Group: Meteor. (2015), https://www.meteor.com/
5. Node.js Foundation: Node.Js. (2015), https://nodejs.org/
6. Hickson, I., Berjon, R., Faulkner, S., Leithead, T., Navara, E.D., O'Connor, E., Pfeiffer, S.: Html5. (2014), http://www.w3.org/TR/html5/
7. @mdo, @fat: Bootstrap. (2015), http://getbootstrap.com/
8. Mohorovicic, S.: Implementing Responsive Web Design for Enhanced Web Presence. Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on, 1206-1210 (2013)
9. Bainomugisha, E., Carreton, A.L., Cutsem, T.v., Mostinckx, S., Meuter, W.d.: A Survey on Reactive Programming. ACM Computing Surveys (CSUR): 45**,** 52. (2013)
10. MongoDB Inc.: Mongodb. (2015), https://www.mongodb.org/
11. Pavlik Jr., P.I., Anderson, J.R.: Using a Model to Compute the Optimal Schedule of Practice. Journal of Experimental Psychology: Applied: 14**,** 101–117. (2008)
12. Koedinger, K.R., Baker, R.S., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J.: A Data Repository for the Edm Community: The Pslc Datashop. In: Romero , C., Ventura, S., Pechenizkiy, M. (eds.): Handbook of Educational Data Mining, Vol. 43. CRC Press, Boca Raton (2010)
13. Maass, J.K., Pavlik Jr., P.I., Hua, H.: How Spacing and Variable Retrieval Practice Affect the Learning of Statistics Concepts. In: Conati, C., Heffernan, N., Mitrovic, A., Verdejo, M.F. (eds.): 17th International Conference on Artificial Intelligence in Education, Vol. 9112, 247-256. Springer International Publishing (2015)