Rodrigo, M. M. T., Baker, R.S.j.D., McLaren, B.M., Jayme, A. & Dy, T.T. (2012). Development of a workbench to address the educational data mining bottleneck. In: K. Yacef, O. Zaïane, H. Hershkovitz, M. Yudelson, & J. Stamper, J. (Eds.) *Proceedings of the 5th International Conference on Educational Data Mining* (EDM 2012). (pp. 152-155).

Development of a Workbench to Address the Educational Data Mining Bottleneck

Ma. Mercedes T. Rodrigo
Ateneo de Manila University
Loyola Heights, Quezon City, Philippines
mrodrigo@ateneo.edu

Ryan S. J. d. Baker
Worcester Polytechnic Institute
Worcester, MA USA
rsbaker@wpi.edu

Bruce M. McLaren
Carnegie Mellon University
Pittsburgh, PA USA
bmclaren@cmu.edu

Alejandra Jayme
Ateneo de Manila University
Loyola Heights, Quezon City, Philippines
alejandra.jayme@gmail.com

Thomas T. Dy
Ateneo de Manila University
Loyola Heights, Quezon City, Philippines
thatsmydoing@gmail.com

ABSTRACT

In recent years, machine-learning software packages have made it easier for educational data mining researchers to create real-time detectors of cognitive skill as well as of metacognitive and motivational behavior that can be used to improve student learning. However, there remain challenges to overcome for these methods to become available to the wider educational research and practice communities, including developing the labels that support supervised learning, distilling relevant and appropriate data features, and setting up appropriate cross-validation and configuration algorithms. We discuss the development of an Educational Data Mining (EDM) Workbench designed to address these challenges.

Keywords

Educational data mining workbench

1. INTRODUCTION

In recent years, educational data mining methods have afforded the development of detectors of a range of constructs of educational importance, from gaming the system [5] to off-task behavior [3] to motivation [8] to collaboration and argumentation moves [11]. The development of these detectors has been supported by the availability of machine learning packages such as RapidMiner [12], WEKA [15], and KEEL [1]. These packages provide large numbers of algorithms of general use, reducing the need for implementing algorithms locally, however they do not provide algorithms specialized for educational data mining, such as the widely used Bayesian Knowledge-Tracing [7]. Furthermore, effective use of these packages by the educational research and practice communities presumes that key steps in the educational data mining process have already been completed. For example, many of these detectors have been developed using supervised learning methods, which require that labeled instances, indicative of the categories of interest, be provided. Typically, many labeled instances – on the order of hundreds, if not thousands – are required to create a reliable behavior detector. Labeling data is a time consuming and laborious task, made even more difficult by the lack of tools available to support it.

A second challenge is the engineering and distillation of relevant and appropriate data features for use in detector development [15]. The data that is directly available from log files typically lacks key information needed for optimal machine-learned models. For instance, the gaming detectors of both [5] and [14] rely upon assessments of how much faster or slower a specific action is than the average across all students on a problem step, as well as assessments of the probability that the student knew the cognitive skills used in the current problem step. This information can be distilled and/or calculated by processing data across an entire log file corpus, but there are currently no standard tools to accomplish this. Feature distillation is time-consuming, and many times a research group re-uses the same feature set and feature distillation software across several projects (the second author, for instance, has been using variants of the same feature set within Cognitive Tutors for nine years). Developing appropriate features can be a major challenge to new entrants in this research area. To address this "data labeling bottleneck" and the difficulty in distilling relevant features for machine learning, we are working to develop an *Educational Data Mining (EDM) Workbench*. A beta version of this Workbench, now available online at http://penoy.admu.edu.ph/~alls/downloads, is described in this paper. The workbench currently allows learning scientists to

1) label previously collected educational log data with behavior categories of interest (e.g. gaming the system, help avoidance), considerably faster than is possible through previous live observation or existing data labeling methods.
2) collaborate with others in labeling data.
3) automatically distill additional information from log files for use in machine learning, such as estimates of student knowledge and context about student response time (i.e. how much faster or slower was the student's action than the average for that problem step).

Through the use of this tool, we hope that the process of developing a detector of relevant metacognitive, motivational, engagement, or collaborative behaviors can eventually be sped up. Just the use of "text replays", on previously collected log data has been shown to speed a key phase of detector development by about 40 times, with no reduction in detector goodness [5].

2. EDM WORKBENCH

Version 1.0 of the EDM Workbench interfaces with some of the tools discussed in Section 1, filling some of the functional gaps that, without the Workbench, require manual intervention or require hand-coding of custom tools and cumbersome and

complex actions by the user in packages such as Excel. Version 1.0 of the Workbench has five functionalities: Log import, feature distillation, data sampling, data clipping and labeling, and data export. We discuss each of these functions in turn.

2.1 Log import

The EDM Workbench allows users to import logs in DataShop text format [9] and CSV. The data is assumed to be stored in a flat file, organized in rows and columns. The first row of the import file is assumed to contain each column's name. Each succeeding row represents one logged transaction, usually between the student and tutor but possibly between two or more students as in the case of collaborative learning scenarios. If the user specifies that the imported data is in DataShop text format, the Workbench will check whether the table contains the columns it requires to distill 26 pre-defined features (discussed in 2.2). The successfully-imported logs may be saved in the Workbench's format for work files—a compressed file containing the data in CSV format plus metadata specific to the EDM Workbench.

The Workbench can also import nested folders of data, where each folder level represents a meaningful subset of the data. For example, if data from a section of students is collected several times over a school year, the researcher may have one folder for the school year, one subfolder for each section within the school year, one subfolder for a session within each section, and finally one file or folder for each student within a session. The Workbench allows users to label each level of subfolder, creating new columns for these labels, appending them to the data tables during importation process.

2.2 Feature distillation

Assuming the necessary columns exist in the imported file, the Workbench can automatically distill 26 features from the data. The Workbench also has capacity for defining new features for future analyses. The 26 features distilled come from features used in past automated behavior detectors using DataShop data and related intelligent tutoring system data [2, 5, 13, 14]. The features include (but are not limited to) estimates of the student's knowledge of the current skill [7]; the time the student spent on the problem (both in absolute and relative terms); and the types, number and proportion of correct, wrong, or help actions for the current skill for the last n steps, for the skill, or for the student.

The current EDM Workbench uses 21 generic functions to compute the 26 automatically distilled features. Some functions correspond directly to a single feature while others are reusable, i.e. users can vary input parameters to compute for different features. Figure 1 is an excerpt of the EDM Workbench configuration file that specifies the features to be distilled and the functions used to distill them. It shows the specification of two features: `timeSD` and `timelastnSD`. These features have been used in several behavior detectors [2, 3].

The first example in the excerpt is the specification for the feature `timeSD`, which makes use of a generic function also named `timeSD`. `<group_col>` refers to a sub-grouping criterion. In this case, the data is grouped by type of step, as specified in the `Step Name` column. The `<range_col>` is the column that contains the duration values that will be used to compute `timeSD`. Finally, `<out>` specifies the feature and output column name.

The second example in the excerpt is the specification for `timelastnSD`. It uses the function `sumLastN`. `<sort_col>` refers to the column by which the data should be sorted before computing the feature. The two sets of `<group_col>`s imply that data sub-grouping in this case is based on two criteria, the `Anon Student Id` and the `Problem Name`. The `<range_col>` refers to the `timeSD` column, computed earlier. The `<n>` refers to the number of steps to be used in the computation. As with the first example, `<out>` specifies the feature and output column name.

Figure 1. Excerpt from the EDM Workbench configuration file.

```
<feature_set>
 <timeSD>
  <group_col>Step Name</group_col>
  <range_col>Duration</range_col>
  <out>timeSD</out>
 </timeSD>
 <sumLastN>
  <sort_col>Row</sort_col>
  <group_col>Anon Student Id</group_col>
  <group_col>Problem Name</group_col>
  <range_col>timeSD</range_col>
  <n>3</n>
  <out>timelastnSD</out>
 </sumLastN>
</feature_set>
```

Feature `timeSD`

Feature `timelastnSD`

At the moment, adding new features for distillation requires some programming: If the feature can be computed by using one of the 21 existing functions, the user can modify the EDM configuration file to define the new function and how it is derived. If a feature requires a new function, the user can add the new function to the EDM Workbench's source code, after which the new feature can be defined in the configuration file. It is our long-term objective to foster a user community that will eventually make new features available for others to use, similar to the open source software community, increasing the EDM Workbench's usefulness to the broader research community.

2.3 Clip generation

In different projects, text replays have been implemented in several different ways [5, 10, 13]. Two of the key ways that text replays have differed has been in terms of the information and grain-size of the data presented to the coder. For coding, data is subdivided into smaller units, termed *clips* — subsets of student-tutor transactions defined based on criteria for when they begin and end, and what information is included. For example, in various projects, clips have been defined as 20-second intervals [5], segments of 5 or 8 actions [10], and in terms of defined "begin" and "end" events in the learning software [13].

The EDM Workbench allows the user to define the set of features by which the data should be grouped, so that clips do not contain rows from different groups. For example, if the data should be grouped by student, a single clip will contain data from only one student and not multiple students. The workbench also specifies the clip size, either by time or by number of transactions. Delineation of clips by beginning and ending events is not yet possible, but is a feature planned for future implementation. The

Workbench then generates the clips for analysis, according to a sampling scheme discussed in the next section.

2.4 Data sampling

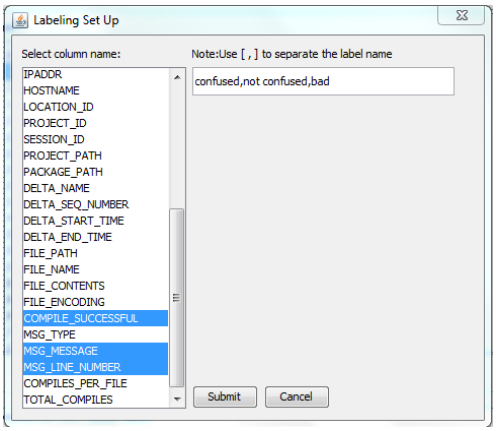
The data sampling feature of the Workbench allows the user to specify how clips are sampled from the data set. (It can also be used to sample at the action/transaction level). The user can specify the sample size, and whether the Workbench will randomly take the sample across the entire population or whether the workbench will stratify the sampling based on one or more variables.

Note that the Workbench allows the user to sample the data at any point of the process — after importing, after clipping, or after labeling – depending on the user's analytical goals.

2.5 Labeling

Once the sample has been taken, the user must then specify a subset of the clip columns that should be displayed in the text replay. It is possible that the user does not want all the clip columns displayed in the text replay. In the example shown in Figure 2, the user specified that only three columns will be displayed: COMPILE_SUCCESSFUL, MSG_MESSAGE and MSG_LINE_NUMBER. The user also specifies the labels that the observer or expert will use to characterize each clip. Figure 3 (bottom) shows that expert or observer will have three labels to choose from: Confused, Not Confused or Bad Clip – the coding scheme from [10]. The circumstances under which an expert or observer labels a clip as "bad" changes depending on the data set, but typically indicate cases that should not be coded. For example, in the case of [10], a clip was labeled "bad" if the transactions contained instructor-supplied programming examples rather than programs that the students had written themselves.

Figure 2. Specification of clip columns and labels.



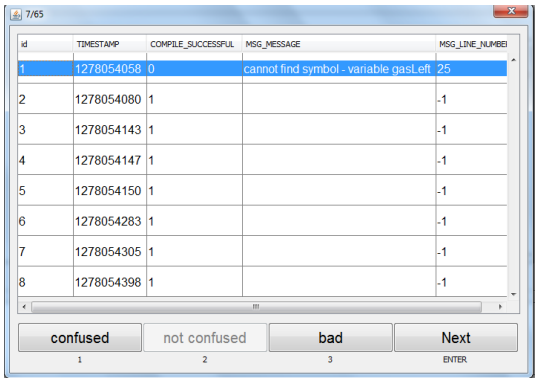
The Workbench then displays text replays of the clips together with the labeling options (Figure 3). A coder reads through the text replay and selects the label that best describes the clip. The labels are saved under a new column in the data set.

Because a coder may have to label tens of thousands of clips [5], the coder may save his or her work and can continue the labeling process in a later session.

2.6 Feature distillation and export

Once data labeling is complete, the user can create clip-level data features to associate with the clips, facilitating later development of detectors. The user first selects the feature or column of interest. The user then specifies whether he/she would like the Workbench to compute for the minimum, maximum, average or standard deviation of that feature [13]. The Workbench will add the new column and corresponding computed value results to the clip dataset

Figure 3. Text replay and label options.



Finally, once processing is complete, the Workbench allows the user to save the logs in CSV format, for re-importation into an appropriate data mining tool, such as RapidMiner or WEKA. The user is then able to use that tool to build a detector of the construct they labeled, using the features they distilled.

3. FUTURE WORK

In this paper, we have presented the Educational Data Mining Workbench, a tool that researchers can use to facilitate the development of detectors of varying forms of student behavior. Version 1.0 of the Workbench supports two key steps of the detector development process that are relatively difficult and time-consuming to do with existing tools: data labeling and feature distillation. By scaffolding users in conducting either or both of these steps, the tool may make it easier and quicker for a wider range of learning scientists and educational software developers to develop and use automated detectors of student behavior.

It is worth noting that the current version of the Workbench is still limited. Each of the limitations discussed here are scheduled for implementation in the coming months. (1) The automatically-distilled features are hard-coded; future releases will make it easier to alter the feature list. (2) The process of amending XML to create new features will be made more user-friendly. (3) The coders cannot change the way in which the text replays are displayed; future releases will support configuration of different ways to pretty print the text replays, towards highlighting the most important information for the coder's specific current purpose. (4) Users can currently only sample data and assemble it into clips in a limited number of fashions; we intend to implement more sophisticated sampling and clip-creation strategies [13].

A final direction for future work is to add support for researchers creating and validating models appropriately. Within the educational data mining community, there has emerged considerable know-how about how to set up tools such as RapidMiner to afford appropriate validation. (For example, batching data in order to support k-fold student-level cross-validation, and then using a BatchXValidation operator in RapidMiner to implement it). We plan to add support for automatically creating appropriately stratified batches to realize several common cross-validation strategies, and automatically export RapidMiner code that is set up to read in the correct data

and use appropriate cross-validation to build a detector of the construct that was labeled.

Development of the EDM Workbench remains ongoing, and we look forward to collaborating with a range of EDM researchers and learning scientists in making this tool as useful as possible for the EDM community. We welcome comments and suggestions – as well as contributions – from any interested colleague.

4. ACKNOWLEDGMENTS

We thank Jessica Sugay, Alipio Gabriel, and John Paul Contillo. We also thank John Stamper, Alida Skogsholm, and Ken Koedinger for helpful comments and suggestions. This research project was made possible through a grant from the Philippines Department of Science and Technology's Engineering Research and Technology for Development program entitled "Development of an Educational Data Mining Workbench."

5. REFERENCES

[1] Alcala-Fdez, J., Sanchez, L., Garcia, S., de Jesus, M.J., Ventura, S., Garrell, J. M., Otero, J., Romero, C., Bacardit, J. & Rivas, V.M. (2009). KEEL: A software tool to assess evolutionary algorithms for data mining problems. *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, *13*(3), 307-318.

[2] Baker, R.S.J.d., Corbett, A.T., Aleven, V. (2008). More Accurate Student Modeling Through Contextual Estimation of Slip and Guess Probabilities in Bayesian Knowledge Tracing. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, 406-415.

[3] Baker, R.S.J.d. (2007). Modeling and Understanding Students' Off-Task Behavior in Intelligent Tutoring Systems. *Proceedings of ACM CHI 2007: Computer-Human Interaction*, 1059-1068.

[4] Baker, R.S.J.d., Corbett, A.T., Koedinger, K.R., & Wagner, A.Z. (2004). Off-Task Behavior in the Cognitive Tutor Classroom: When Students "Game the System." *Proceedings of ACM CHI: Computer-Human Interaction*383-390.

[5] Baker, R.S.J.d. & de Carvalho (2008). Labeling Student Behavior Faster and More Precisely with Text Replays. *1st International Conference on Educational Data Mining*, 38-47.

[6] Cetintas, S., Luo, S., Yan Ping Xin, & Hord, C. (2010). Automatic detection of off-task behaviors in intelligent tutoring systems with machine learning techniques. *IEEE Transactions on Leanring Technologies*, *3*(3), 228-2*36.*

[7] Corbett, A.T., & Anderson, J.R. (1995). Knowledge Tracing: Modeling the Acquisition of Procedural Knowledge. *User Modeling and User-Adapted Interaction, 4*, 253-278.

[8] de Vicente, A., Pain, H. (2002). Informing the detection of the students' motivational state: an empirical study. *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, 933-943.

[9] Koedinger, K., Cunningham, K., Skogsholm, A., Leber, B. (2011) An data repository for the EDM community: The PSLC DataShop. In c. Romero, S. Ventura, M. Pechenizkiy and R. S. J. d. Baker, *Handbook of Educational Data Mining*. Boca Raton, FL: CRC Press, 43-55.

[10] Lee, D. M., Rodrigo, M. M. T. R., Baker, Ryan S. J. D., Sugay, J. O., & Coronel, A. (2011). Exploring the relationship between novice programmer confusion and achievement. In S. D'Mello & A Graesser (Eds.): *ACII 2011, Part I, LNCS 6974*, (pp. 175-184), Berlin Heidelberg: Springer-Verlag.

[11] McLaren, B.M., Scheuer, O., & Mikšátko, J. (2010). Supporting collaborative learning and e-Discussions using artificial intelligence techniques. *International Journal of Artificial Intelligence in Education (IJAIED) 20*(1), 1-46.

[12] Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M. & Euler, T. (2006). YALE: Rapid Prototyping for Complex Data Mining Tasks. In *Proc. of the 12th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining (KDD 2006)*, (pp. 935-940), ACM Press.

[13] Sao Pedro, M., Baker, R., Gobert, J., Montalvo, O., & Nakama, A. (in press). Leveraging Machine-Learned Detectors of Systematic Inquiry Behavior to Estimate and Predict Transfer of Inquiry Skill. *User Modeling and User-Adapted Interaction.*

[14] Walonoski, J. & Heffernan, N.T. (2006). Detection and Analysis of Off-Task Gaming Behavior in Intelligent Tutoring Systems. In Ikeda, Ashley & Chan (Eds.). *Proceedings of the 8th International Conference on Intelligent Tutoring Systems*. Springer-Verlag: Berlin. pp. 382-391.

[15] Witten, I. H. & Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques*, Second Edition, Morgan Kaufmann.