Automatic Action Analysis in an Interactive Learning Environment

Martin Muehlenbrock

German Research Center for Artificial Intelligence DFKI
Martin.Muehlenbrock@dfki.de

Abstract. Recently, there is a growing interest in the automatic analysis of learner interaction data with web-based learning environments. The approach presented in this paper aims at helping to establish a basis for the automatic analysis of interaction data by developing a data logging and analysis system based on a standard data base server and standard machine learning techniques. The analysis system has been connected to a web-based interactive learning environment for mathematics teaching, but is designed to allow for interfacing also to other web based learning environments. The system has been tested in a five-month experiment in which four classes of a secondary school participated throughout a complete school term on a weekly basis.

1. Introduction

Recently, there is a growing interest in the automatic analysis of learner interaction data with web-based learning environments. This is largely due to the increasing availability of log data from learning environments and in particular from web-based ones. The objectives include the detection of regularities and deviations in the learners’ or teachers’ actions among others, and to support teachers and learners by providing them with additional information to manage their learning and teaching, respectively, and possibly suggest remedial actions. Commercial systems such as WebCT, Blackboard, and LearningSpace already give access to some information related to the activity of the learners including some statistical analyses, and provide teachers with information on course attendance and exam results. With this information already being useful, it only represents the tip of iceberg of what might be possible by using advanced technologies.

This upcoming field, i.e., addressing the automatic analysis of learner interaction data, is related to several well-established areas of research including intelligent tutoring systems, web mining, and machine learning, and can build upon results from these fields for achieving its objectives. In contrast to intelligent tutoring systems, learner interaction analysis does not rely on models of learner or domain knowledge since these are heavy to build and maintain. In this regards, learner interaction analysis is comparable to website data mining, but with a specific perspective on learning settings and with the availability of pedagogical data that usually are not available in web mining applications that are mostly based on click through data. Click through data streams only allow for a rather shallow analysis, but with the inclusion of pedagogical data also more advanced techniques can be adopted from the field of machine learning.

Although a number of open questions have already been tackled [1][2][5][6][7][10][12], there is not yet a systematic approach in analysis interaction data from huge learner action logs. The
approach presented in this paper aims at helping to establish a basis for the automatic analysis of interaction data by developing a data logging and analysis system based on a standard database server and standard machine learning techniques. The analysis system has been connected to a web-based interactive learning environment for mathematics teaching, but is designed to allow for interfacing also to other web based learning environments. The system has been tested with a medium scale experiment in which four classes of a secondary school participated throughout a school term of five months on a weekly basis.

2. System

ActiveMath is a web-based learning environment that dynamically generates interactive courses adapted to the student's goals, preferences, capabilities, and prior knowledge \([3], [4], [9]\). The content is represented in a reusable XML-knowledge representation specifically designed for an educational context. ActiveMath supports individualized learning material in a user-adaptive environment, active and exploratory learning by using (mathematics) service tools and with feedback, better reusability and interoperability of the encoded content and exercises. For different purposes and for different users the learning material and its presentation can be adapted: the selection of the content, its organization, the means for supporting the user have to be different for a novice and an expert user, for an engineer and a mathematician, for different learning situations such as a quick review and a full study. Since there is no way of knowing in advance the goals, the profile, and the preferences of any user when designing the system, ActiveMath builds on adaptive course generation.

For each learner, the ActiveMath environment generates an online log that lists all user actions in the learning environment in terms of general information such as time, type of action, user name, and session number, as well as specific information including which page has been presented to the user, which item has been seen by the user, which exercise has been tackled and solved or not solved. A recent implementation of the learning environment also provides information on user actions in terms of events that other system components can subscribe to. The analysis system is comprised of three major components, i.e., the log database, the updater, and the analyzer.

- The **log database** is at the center of the action analysis system. It contains not only representations of the raw data in the user logs, but also has tables that hold the results of the analysis as well as tables for additional background knowledge concerning users or courses among others (see in figure 1, which will be detailed below).
- The **updater** receives event information on the users’ actions from the learning environment, and transforms every user event into one or more corresponding database tables. Usually, the updater receives the information online from the event queue, but it can also read in files with log data that have been generated in an offline mode. In addition to updating the event information in the database, the updater also enhances and extends the event data, as will be described below.
- The **analyzer** performs data aggregation and evaluation in terms of queries to the log database as well as incorporates a number of learning methods and takes the data from the log data base as an input. If needed, adjustments and preferences are input by the user that is running the analysis. An example will be given in section 4.

The analysis system has been implemented by using standard technology such as Java and mySQL, which are available for a number of platforms and operating systems, together with the suitable drivers for database connectivity. In addition, the analyzer is based on the Weka
toolkit [11], which provides tools for visualizing and exploring data as well as means for integrating machine learning functionality into applications.

Figure 1. The schema of the log database is comprised of tables for the raw data (subdivided into general event data, specific event data, and enhanced and extended event data), for the derived data, and for background data.
The database is organized in four areas as illustrated in figure 1. The schema depicts a table for generic event information (in the center), tables for event-specific data (at the bottom right), tables for enhanced and extended event-specific data (at the top right), tables for derived data and views (at the top left), and tables for background information (at the bottom left). The event table contains information that is present in every event. It represents the backbone of the data schema related to the raw data. Event-specific tables incorporate information that is provided only by individual events. The structure of these tables has been designed closely to the events specification, since this allows for simpler updating operations when the event subsystem is changed or replaced by another system.

In order to comply with this principle, information that is derived by the activity analysis and is closely related to the underlying events is not incorporated in these tables, but further tables are created, e.g. an extension of the event table that incorporates date and time information in a readable format instead of the timestamp, and an extension of the login table that derives information on a user’s location in terms of being in school/work or at home from the network address of the user’s computer. In addition, frequently for some tables the information is not complete. For instance, most users do not log out of the learning system explicitly but simply close the browser or shut down their computer. In this case no event is generated concerning the logout. The corresponding logout table is enhanced by information that is derived from the other events the user created and on heuristics concerning pauses and open hours among others. This information is automatically added to the login table, but is marked as derived information in an additional table concerning the source of information.

3. Experiment

The analysis system has been tested in an experiment in a secondary school with about 70 students from three different classes that used the learning environment for a period of five months. The subject area was fractions and divisibility, and material had been prepared in terms of reading material, exercises, and dictionary entries for the ActiveMath learning environment (see figure 2). In addition, a further course of about 25 students were taught the same subject, but in the traditional classroom manner. The other three courses used the ActiveMath learning environment on a weekly basis in two-hour lessons. During the online course each class was split into two subgroups using different computer rooms. Many students already were familiar with computers, but a considerable number needed further instruction even for basic operations such as login.

A preliminary evaluation of the logged data after a first couple of sessions showed some problems in the quality of the data. For instance, instead of registering with the ActiveMath system only in the very first session and using the created user account in the sequel, a large number of students created a new account including a new user name for each session, which makes difficult the longitudinal analysis of the data. The problem was resolved by having the students create only one account and making the registration procedure inaccessible for them after that. Figure 3 provides a view on the data that shows the number of events related to the hours of the day. Clearly, the major amount of events was created during lesson hours between 9 am (9h) and 2 pm (14 h), but some events were generated earlier or later in the day. Some of them are due to a small number of students using the system off time, though most of these are due to teachers and system administrators preparing or evaluating the system.
At the end of the term, a written post test has been done with the students to assess what they had learnt. The results were added to the database manually as well as some further information on gender, teacher, etc. Further information was automatically generated by the analysis system, or more specifically by the updater component, from the log data and added to the database. For each student the information in table 1 has been gathered for further analysis. For anonymity reasons the students used arbitrary user names in the learning environments, and they were to give these user names also in the post test. However, in one course the students put down their real names on the test sheets, a fact which makes the linking to their log data impossible. Finally, 25 student records were complete and clean enough for being used in the further analysis.

Figure 2. Sample course content on fractions and divisibility (in German).

Figure 3. Number of user actions in relation to hours of the day.
<table>
<thead>
<tr>
<th>Attribute</th>
<th>Generation</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Manual</td>
<td>User name (not used for the decision tree learning)</td>
</tr>
<tr>
<td>Class</td>
<td>Manual</td>
<td>Course, each comprised of about 20 students (not used for the decision tree learning)</td>
</tr>
<tr>
<td>Teacher</td>
<td>Manual</td>
<td>Each class has been split into two subgroups, with each being taught by another teacher (not used for the decision tree learning)</td>
</tr>
<tr>
<td>Gender</td>
<td>Manual</td>
<td>Male or female</td>
</tr>
<tr>
<td>Integration pupil</td>
<td>Manual</td>
<td>Whether the student is handicapped</td>
</tr>
<tr>
<td>Post test result</td>
<td>Manual</td>
<td>Results in the post test done in writing (binned into low, medium, and high for the decision tree learning)</td>
</tr>
<tr>
<td>Ex_started</td>
<td>Automatic</td>
<td>Number of exercises started</td>
</tr>
<tr>
<td>Ex_finished</td>
<td>Automatic</td>
<td>Number of exercises finished</td>
</tr>
<tr>
<td>Num_successes</td>
<td>Automatic</td>
<td>Number of successful exercises</td>
</tr>
<tr>
<td>Avg_reading</td>
<td>Automatic</td>
<td>Average number of reading actions in a session</td>
</tr>
<tr>
<td>Avg_solving</td>
<td>Automatic</td>
<td>Average number of exercise solving actions in a session</td>
</tr>
<tr>
<td>DictUsed</td>
<td>Automatic</td>
<td>Whether the student used the dictionary for searching information</td>
</tr>
<tr>
<td>WorkedOffTime</td>
<td>Automatic</td>
<td>Whether the student accessed the learning environment beyond lesson hours, e.g. from home or during free periods</td>
</tr>
<tr>
<td>Ex_finished_rate</td>
<td>Automatic</td>
<td>Rate of finished exercises to all started exercises</td>
</tr>
<tr>
<td>Ex_success_rate</td>
<td>Automatic</td>
<td>Rate of successful exercises to all finished exercises</td>
</tr>
</tbody>
</table>

Table 1. Data automatically gathered and updated for each user, plus some background information added manually.

4. Analysis

The analyzer component incorporates a number of machine learning methods for automatically analyzing the data in the log data base. In addition to getting a better insight into the underlying relationships in the data, this also allows for prediction and classification of future sessions. Many machine learning methods provide their output in an intelligible, human readable form. For instance, methods for generating decision tress from data, such as C4.5 [8], allow for a tree-shaped representation of the learning results. A decision tree is constructed by the algorithm first selecting an attribute to place at the root node of the tree and make one branch for each possible value. This splits up the example set into subsets, one for every value of the attribute. The attribute is selected in a way that maximizes the information gain by the chosen attribute. This process is repeated recursively for each branch, using only those instances that actually reach the branch. If at any time all instances at a node have the same classification, the developing of that part of the tree is stopped.

Figure 4 shows the decision tree that was generated by the system for characterizing the attribute post test, after the values of the post test results had been binned into the three categories low, medium, and high with nearly equal distribution as well as nearly equal intervals. According to this decision tree, the attributes exercise success rate and exercise finished rate are the most important and second most important features to classify the post test results, respectively. On the third and forth level also the number of finished exercises and the average time spend on reading are relevant. This decision tree is based on the data from the experiments, but it can also be used to make predictions in future usages, i.e., by estimating the effect of using the learning environment on new users in terms of a potential post test.
Figure 4. Decision tree for characterizing post test result.

Table 2. Confusion matrix with tenfold cross validation for the decision tree in figure 4.

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>11</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>Medium</td>
<td>9</td>
<td>1</td>
<td>13</td>
</tr>
<tr>
<td>High</td>
<td>5</td>
<td>5</td>
<td>11</td>
</tr>
</tbody>
</table>

The quality of the decision tree is presented in table 2 as a confusion matrix. A confusion matrix displays the result of testing the decision tree with data as a two-dimensional matrix with a row and a column for each class. Each matrix element shows the number of test examples for which the actual class is the row and the predicted class is the column. Good results correspond to large numbers down the main diagonal and small, ideally zero, off-diagonal elements. Stratifies tenfold cross-validation has been used to produce this table, which means that the data is divided randomly into ten parts, in each of which the class is represented in approximately the same proportions as in the full dataset. Each part is held out in turn and the learning scheme is trained on the remaining nine-tenth, and finally the error estimates are averaged.

The confusion matrix indicates that only for slightly more than one third of the examples the post test result is predicted correctly, for less than one half of the examples a near miss (medium instead of high etc.) is indicated, and for one fifth of the examples the classification is completely wrong.
5. Summary and further work

In this paper a system is presented for the automatic analysis of user actions in web-based learning environments. It has been tested in a school experiment with about 70 students over a couple of months. The automatic analysis of the data already produced a number of interesting results including decision trees that could also be used for prediction in further experiments as well as normal usages. The analysis components have been implemented by using Java and mySQL, which are available for a number of platforms and operating systems. This paper described work in progress, hence further experiments on school and university level will be conducted as well as further analysis methods and machine learning techniques will be investigated.

Acknowledgements

I am grateful to the members of the ActiveMath group for a number of stimulating discussions. A part of the implementation of the action analysis system has been done by Oliver Scheuer. The work presented in this paper is partially supported by European Community under the Information Society Technologies (IST) programme of the 6th FP for RTD - project iClass contract IST-507922.

References