Educational data mining for analysis of students' solutions

Karel Vaculík, Leona Nezvalová, and Luboš Popelínský

KD Lab, FI MU Brno
xvaculi4@fi.muni.cz, xnezva36@mail.muni.cz, popel@fi.muni.cz

Abstract. We introduce novel methods for analysis of logic proofs constructed by undergraduate students. The methods employ sequence mining for manipulation with temporal information about all actions that a student performed, and also graph mining for finding frequent subgraphs on different levels of generalization. We showed in [8–11] that these representations allow us to find interesting subgroups of similar solutions and also to detect outlying solutions. Specifically, distribution of errors is not independent on behavioral patterns and we were able to find clusters of erroneous solutions. We also observed a significant dependence between time duration of solving the task and an appearance of the most serious error. This text brings a brief summary of four contributions [8–11] presented for presentation in the period from October 2013 until September 2014. In the second part, based on [11] we focus on a newly developed outlier detection method that helps to find unusual solutions, both correct and erroneous.

Keywords: educational data mining, logic proofs, clustering, outlier detection, sequence mining

1 Data mining in teaching logic

Teaching constructive tasks, i.e. tasks that a student has to build in several steps – like tasks in descriptive geometry or logic and math proofs, requires advanced evaluation techniques. For example, in the case of resolution proofs in logic, it is not sufficient to assign the mark based only on the conclusion that the student reached. To evaluate a student solution properly, a teacher needs not only to check the final result of a solution (the set of clauses is or is not contradictory) but also to analyze the sequence of steps that a student performed, with respect to correctness of each step and with respect to correctness of that sequence.

We show that novel machine learning methods, such as graph mining and sequence mining, can be very helpful in that situation because of their capability to process structural and temporal information. Specifically, graph mining methods work with data represented as graphs, in our case one graph for each instance, and take into account the structural information of the graphs. An overview of graph mining methods can be found in [1]. Sequence mining is another topic of data mining oriented to structured data. In comparison to graph mining, data

are arranged in sequences, usually ordered by time, and they are assumed to be discrete. More information on sequence mining can be found in [3].

In the first two works [8, 9] we presented a method for analysis of students' solutions of resolution proofs that employed graph mining. It used frequent subgraph mining algorithm Sleuth [12] for finding frequently occurring subgraphs. These subgraphs were then generalized and used as new features for classification to correct and incorrect solutions. In [10] we extended this generalization procedure and left out the frequent subgraph mining algorithm. In [11] we propose a novel method that is much more robust and is actually independent on a particular student task. In addition, it processes also information about the sequence of steps that a student performed, like adding/deleting a node or edge in the proof and also about text (i.e. a formula) modification. It also exploits information about the time a particular operation was performed and uses temporal information for finding outlying solutions. We use three different feature extraction methods and show that by using those new temporal and structural features we are able to find clusters of erroneous solutions. We also show that there is a significant dependence between time duration and appearance of errors in a student solution.

Moreover, by means of class outlier detection we are able to find solutions that are anomalous and for that reason difficult to detect automatically. In the following text we summarize that result.

2 Outlier detection in resolution graphs

Data. The data, 873 solutions altogether, was obtained in the course on Introduction to logic. Via a web-based tool, each of the 351 students solved at least three tasks randomly chosen from 19 exercises. We selected only those solutions that employed linear or linear input resolution. The data set contained the resolution tree and also dynamics of the solutions, i.e. all the actions performed together with temporal information. Among these 873 different students' solutions of resolution proofs in propositional calculus, 101 of them were classified as incorrect and 772 as correct.

The most serious error in resolution is resolving on two literals. In this text we denote this error as E3. Other common errors in resolution proofs are the following: repetition of the same literal in the clause, incorrect resolution – the literal is missing in the resolved clause, resolving on the same literals (not on one positive and one negative), resolving within one clause, resolved literal is not removed, the clause is incorrectly copied, switching the order of literals in the clause, proof is not finished, intentional negation of literals in a clause. Information about the error that appeared in the logic proof is also part of the data. All actions that a student performed, such as adding/deleting a node, drawing/removing an edge, writing/deleting a text into a node, were saved into a database. The collected data contains also a timestamp for each action performed by a student. The timestamps also allow us to get the order of actions and use techniques for sequence mining as discussed in the following sections.

Generalized Subgraphs. As in the case of sequences, graph features may be constructed from graph substructures found by an algorithm for frequent subgraph mining [1]. The task of frequent subgraph mining result in the the frequently occurring subgraphs in a set of graphs. Next, Boolean features (a feature correspond to a appearance of a subgraph) are used and the value of a feature depends on whether the corresponding substructure occurs in the given instance or not. Because the node labels of incorrect solutions have very small support and frequent subgraph mining can be inefficient for small values of minimum support, we created a new method for finding interesting patterns. The other reason was that similar subgraphs differed only in alphabet letters and in the order of literals in text labels or in the order of parent nodes. The main idea used here is to generalize and unify subgraphs consisting of two parent nodes and a resolvent. Such generalized subgraphs may be expressed shortly in the following form: $parent1; parent2 - - > resolvent$. An example of generalized subgraph is $\{\neg Y, Z\}; \{\neg Y, \neg Z\} - - > \{\neg Y\}$, where Y, Z are variables that can match any propositional letter. We also created a new, higher-level, generalization [10], which further generalizes the patterns found earlier. As a result, new patterns in $added; dropped$ form are created. The *added* component simply denotes literals which were added erroneously to the resolvent and the *dropped* component denotes literals from parents which participated in the resolution process. Continuing with the example, the higher-level pattern will be $\{\}; \{\neg Z, Z\}$. The detailed description can be found in [10].

Method. The data we process has been labeled. We focused only on the E3 error as it was the most common and the most serious error. Specifically, there were two values of the class attribute, corresponding to the occurrence or concurrence of the error. Unlike in common outlier detection, where we look for outliers that differ from the rest of "normal" data, we need to exploit information about a class. That is why we used weka-peka [7] that looks for class outliers [6] using Random Forests (RF) [2]. It extends the outlier detection method in RF implemented in Weka [4] that actually works for classical settings – normal vs. anomalous data to manage class information. The main idea lies in different computation of the proximity matrix that exploits also information about a class label [7].

Results. When analyzing the strongest outliers that weka-peka found, we found three groups according to the outlier score. The two most outlying examples, with outlier factor overcoming 130, significantly differ from the others. The second cluster consists of four examples with the outlier score between 50 and 100, and the last group is comprised of instances with the lowest score of 15.91. Analysis of individual outliers let us draw several conclusions. Two most outlying instances contain one specific pattern, *looping*. This pattern represents the ellipsis in a resolution tree, which is used for tree termination if the tree cannot lead to a refutation. Both instances contain this pattern, but neither of them contains the pattern of correct usage of the resolution rule. The important thing is that these two instances contain neither the E3 error nor other errors. This

shows that it is not sufficient to find all errors and check the termination of proofs, but we should also check whether the student performed at least few steps by using the resolution rule. Otherwise we are not able to evaluate the student's skills. Instances with the outlier score less than 100 are less different from other instances.

Discussion. The other method that we used for comparison was CODB [5]. However, when compared with weka-peka, CODB returned much worse results mainly because of using density and distances (to nearest neighbors and to all members of the class) for outlier detection. Such poor results may be caused also by the fact that those metrics are too rough for our task. Moreover, it is much more difficult to obtain a comprehensive explanation of why a particular solution of CODB is an outlier.

It need to be mentioned that this method has not been developed for recognition of correct or incorrect solutions. However, to verify that the feature construction is appropriate, we also learned various classifiers of that kind. Best result was achieved by SMO (SVM implementation in Weka) – 96.9% accuracy. Similar results were obtained when only the higher level of subgraph generalization was used, again with SMO.

References

1. Cook, D. J., Holder, L. B.: Mining graph data. Hoboken, N.J.: Wiley-Interscience (2007)
2. Breiman, L.: Random Forests. In: Machine Learning, vol. 45, no. 1. pp. 5–32 (2001)
3. Dong, G., Pei, J.: Sequence data mining. Springer, New York (2007)
4. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. In: SIGKDD Explor. Newsl., vol. 11, no. 1. pp. 10–18 (2009)
5. Hewahi, N., Saad, M.: Class outliers mining: Distance-based approach. In: International Journal of Intelligent Technology, vol. 2, no. 1. pp. 55–68 (2007)
6. Papadimitriou, S., Faloutsos, C.: Cross-outlier detection. In: Proceedings of SSTD. pp. 199–213 (2003)
7. Pekarčíková, Z.: Supervised outlier detection. Master's thesis (in Czech). Masaryk University (2013)
8. Vaculik, K., Popelinsky, L., Mrakova, E., Jurco, J.: Tutoring and Automatic Evaluation of Logic Proofs. In: Proceedings of the 12th European Conference on e-Learning ECEL (2013)
9. Vaculík, K., Popelínský, L.: Graph Mining for Automatic Classification of Logical Proofs. In: CSEDU (2014)
10. Vaculík, K., Popelínský, L., Nezvalová, L.: Graph mining and outlier detection meet logic proof tutoring. Proceedings of G-EDM Ws, EDM London 2014.
11. Vaculík, K., Popelínský, L., Nezvalová, L.: Educational data mining for analysis of students' solutions. Proceedings of AIMSA, LNCS vol. 8722, Springer Verlag 2014.
12. Zaki, M.J.: Sequences Mining in Categorical Domains: Incorporating Constraints. In: 9th ACM CIKM pp. 422–429 (2000)