

Argument Graph Classification via Genetic Programming and C4.5

Collin Lynch¹, Kevin D. Ashley^{1,2}, Niels Pinkwart³, and Vincent Aleven⁴
collinl@cs.pitt.edu, ashley@pitt.edu, niels.pinkwart@tu-clausthal.de, aleven@cs.cmu.edu

¹ Intelligent Systems Program, University of Pittsburgh.

² School of Law, University of Pittsburgh.

³ Department of Informatics, Clausthal University of Technology.

⁴ HCII, School of Computer Science, Carnegie Mellon University.

Abstract. In well-defined domains there exist well-accepted criteria for detecting good and bad student solutions. Many ITS implement these criteria characterize solutions and to give immediate feedback. While this has been shown to promote learning, it is not always possible in ill-defined domains that typically lack well-accepted criteria. In this paper we report on the induction of classification rules for student solutions in an ill-defined domain.¹ We compare the viability of classifications using statistical measures with classification trees induced via C4.5 and Genetic Programming.

1 Introduction

Much of the success of Intelligent Tutoring Systems stems from their ability to give on-line assessment and feedback. This assessment is usually based upon the implementation of *a-priori* domain or tutoring knowledge. Such implementation works in a well-defined domain where there is widespread agreement about the structure of domain knowledge, relevant solution characteristics and acceptable solution processes.

In ill-defined domains [9], while it is possible to identify individual characteristics that are endorsed or proscribed by some domain experts, it is difficult to find widespread agreement about atomic solution actions let alone whole solutions or problem solving behaviors. This is a fundamental challenge for automating effective instruction in ill-defined domains.

In recent years the Educational Data-Mining (EDM) community has sought to augment the *a-priori* knowledge in existing tutoring systems with more automatically derived knowledge. Work by Baker et al. on gaming detection [1], Cen et al. on over-training [3], and Feng et al. on student modeling [5] has shown the utility of EDM to induce new pedagogical information. However, this work has been based upon statistical learning methods which, while they yield successful predictors, do not permit detailed examination of their internal logic to induce domain knowledge. In a word, they are not easily “inspectable”.

Work by Harrer et al. [6] and Miksatko and McLaren [11] has shown the utility of user-guided extraction of inspectable solution patterns. This work, has focused on a manual approach or a model-driven “guided-discovery” approach where user specified patterns are explored in the dataset to highlight interactions.

¹This research is supported by NSF Award IIS-0412830.

In this paper we describe our work on the automatic extraction of informative behavior patterns from student solutions in the ill-defined domain of legal argumentation. This work is based upon our existing tutoring system for law and legal argumentation called LARGO. As described in the next section, LARGO guides students through the process of analyzing and annotating Supreme Court oral arguments using *a-priori* structural hints. While these hints have been shown to be beneficial, they are restricted to relatively small-scale solution characteristics and, due to the limits of our domain knowledge, do not enable us to efficiently characterize the student solutions as a whole.

Our goal in this work is to induce inspectable classifiers capable of predicting a student's post-test performance based upon their solution characteristics. We will then analyze these classifiers comparing them to our domain model, checking our expectations of good student behavior and identifying similar groups of poor performers. For the present we chose to induce binary decision trees via C4.5 and Genetic Programming. We felt that these structures were expressive enough to cover a range of student behaviors while inspectable enough to be analyzed by domain experts and rendered into instructional explanations. We begin by providing the necessary background on LARGO, C4.5, and Genetic Programming. We follow this with a discussion of the rules that were induced and their relative successes and failures. We then conclude with a discussion of the suitability of these techniques for future work.

2 Background

LARGO, an Intelligent Tutoring System for legal argumentation [15], supports students in the reading, annotation, and analysis of oral argument transcripts taken from the U.S. Supreme Court. These transcripts cover cases argued before the court and contain complex real-world examples of legal argument as performed by practicing experts.

These arguments are characterized by the use of proposed tests and hypothetical cases. Advocates before the court propose a test or legal rule about how the case should be decided. Not coincidentally the outcome of this rule favors their client. Judges in turn probe the test by posing hypothetical cases, variations on the specific facts of the case that stress one part of the test or another to probe for weaknesses. The advocates respond by reformulating the test to take into account the hypotheticals or distinguishing the hypothetical from the case at hand, citing relevant facts and legal principles with which to justify their argument.

LARGO is designed to facilitate students' understanding of this form of argument by supporting their analysis of the examples. Students using LARGO are presented with a series of case problems comprising oral arguments to annotate using a graphical markup language. This markup is composed of test, hypothetical and fact nodes, as well as arcs connecting them. Students may add descriptive labels to any of the nodes and arcs, and may link the Test and Hypo nodes to selected portions of the argument transcript. The same formalism and the characteristics described below can also be used for the production of novel arguments in a manner similar to [4]. A sample diagram is shown in Figure 1.

LARGO analyzes the student diagrams for structural, contextual and content-related "characteristics" which we use as the basic features of our current analysis. Each characteristic is defined by a particular graphical pattern that, if it matches some portion of a student's diagram, identifies a possible structural weakness or opportunity for reflection. The characteristics were developed

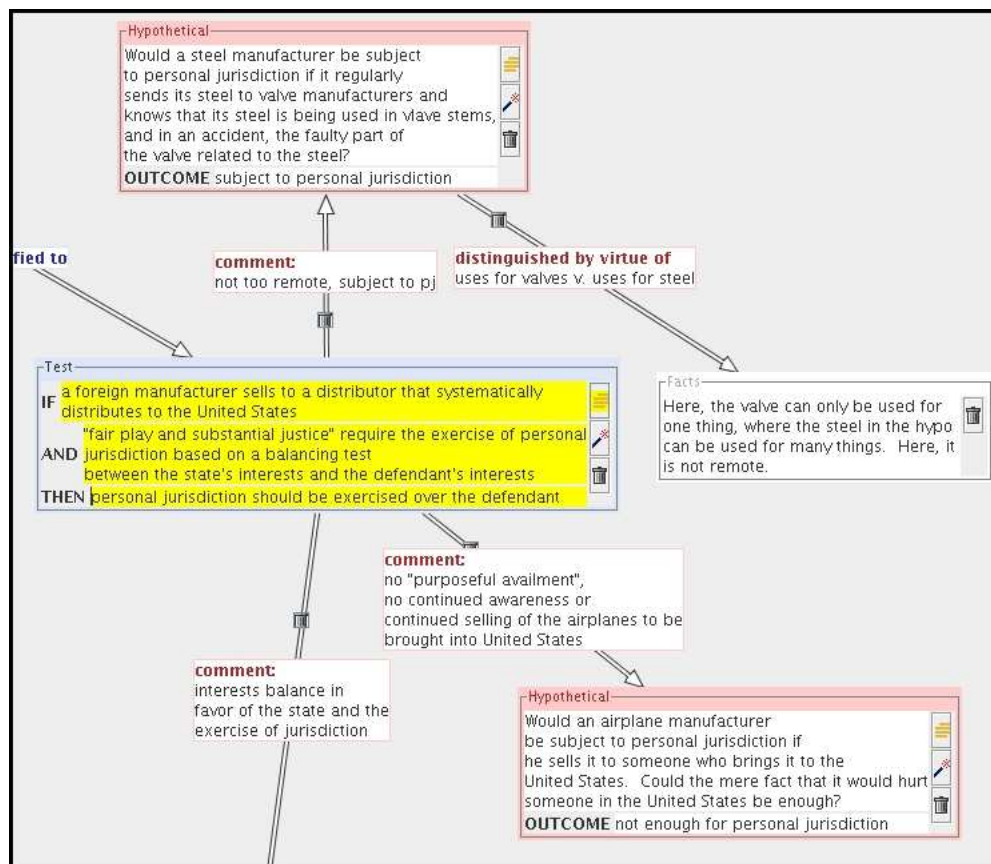


Figure 1. Sample LARGO Graph

with the help of an experienced legal instructor. For example *UNLINKED_TEST*, a context characteristic is active when the student has formed a test box in the graph but has not linked that box to the transcript. Such linking is a necessary part of good note-taking as it enables the students to reconnect their diagrams to the relevant parts of the arguments. The structural characteristic *FACTS_ISOLATED_FROM_HYPOS* is active when the student has produced a fact node but not linked it to the relevant hypothetical nodes.

These diagram characteristics are associated with phases of graph production (1=orientation, 2=transcript markup, 3=diagram creation, 4=analysis, and 5=reflection). Characteristics of phases 1-3 can be thought of as indicating basic “weaknesses” of a diagram (e.g. *UNLINKED_TEST*), while characteristics of phases 4 and 5 stand for opportunities for reflection contained in a diagram. The system provides feedback on diagrams in the form of self-explanation prompts triggered by the characteristics. In the earlier phases these prompts inform the student about how to fix up the diagrams. In the later phases, the prompts encourage reflection on the diagram and argument representation. These hints are provided upon request.

LARGO also contains a facility for *collaborative feedback*. For each case in the system we have identified two *target* test statements in the transcript. These are test statements that our domain expert considered to be particularly crucial for the analysis process. Students who link a test node to one of these statements are given the opportunity to rate other students’ statements

of the same test and to reconsider their own. Students who go through the process and whose tests are rated poorly by their peers are given the opportunity to change their own test in response. This characteristic is *TEST_REVISION_SUGGESTED*. It is active for students whose test has been rated poorly but have not changed the test statement. See [14] for a more detailed analysis of the help system and an argument example.

We have completed three studies of the LARGO system. In the Fall of 2006 we conducted a study with paid volunteers taken from the first year class at the University of Pittsburgh's School of Law (*Novice-2006*). Students were randomly assigned to analyze a pair of cases using LARGO or a text-based notepad tool with no feedback. We compared test scores between the groups and analyzed student interactions with the system. We found no overriding difference between the conditions, and close examination of the questions showed that some were too easy causing a ceiling effect. However on other question types lower aptitude students, as measured by their Law School Admission Test (LSAT) score (a frequently used predictor for success at law schools) in the LARGO condition, showed higher learning gains on some question types than their low-LSAT text peers. Also, the use of the help features was strongly correlated with learning [15].

In the Fall of 2007 we performed a follow-up study as part of the first year legal process course (*Novice-2007*). The study was mandatory for all 85 class members. As before students were assigned randomly to text or graph conditions. However the study included one additional case and students answered case-specific essay questions after each session. We also replaced some questions from the pre- and post-tests that had produced a ceiling effect with more challenging alternatives. We again found no significant differences between conditions. A post-hoc analysis revealed that the students in the first study made far more use of the advice functions than the students in the second study, which may explain the difference between the study outcomes.

We are presently conducting a follow-up study with LARGO among third-year law students (*Expert-2008*). All participants in this study used LARGO and performed the same set of tasks as those in *Expert-2007*. The purpose of this study is to examine novice-expert differences in the context of LARGO. At the time of this paper a total of 17 third-year students have completed the study and their data, along with data from the *Novice-2007* study, are employed below.

C4.5 is a decision tree induction algorithm [16]. When presented with data it induces an n -ary decision tree that acts as a functional classifier. Each interior node of the tree represents a logical test that branches to one child or another based upon the outcome of the test. Leaf nodes represent predictions or decisions made. Decision trees are traversed from root to leaf. Each decision path p from the root to a leaf node defines a class of cases based upon the relevant features and a classification tag to be assigned to those cases. Each tree therefore represents a hypothesis or test for carving up the space of diagrams according to the factors involved in each.

One such decision tree, and its representation in pseudocode are shown in Figure 2. As we shall discuss below this tree predicts student scores as at or below the mean (0) or above the mean (1) where the inner nodes represent tests for the presence or absence of graph characteristics and the leaves, classifications. Each unique path from root to leaf in a decision tree defines a distinct class of objects. For our purposes these denote unique classes of student solutions.

Intriguingly *TEST_REVISION_SUGGESTED* is taken as a sign of high performance. As we noted above, this is a later phase characteristic and will not be active unless the students have successfully marked up a target region of the diagram with a test node, and then made use of the

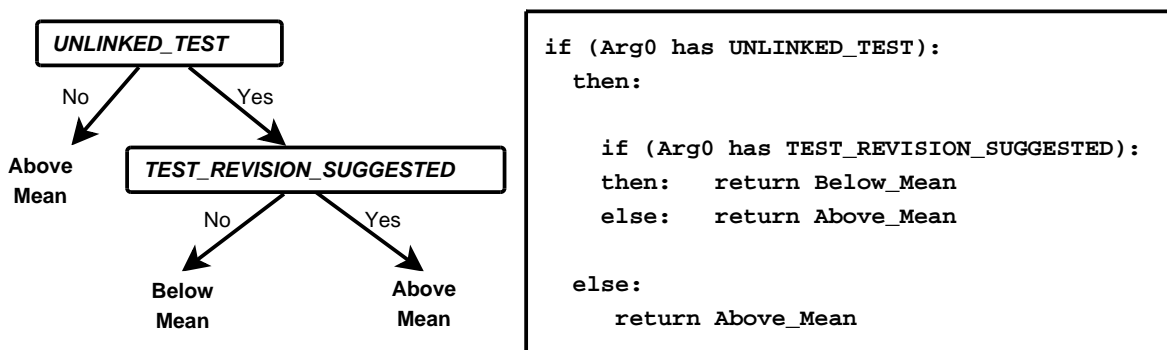


Figure 2. Sample Decision Tree with Pseudocode.

collaborative feedback with their test summary having been given a poor rating by their peers.

Genetic Programming (GP) is a type of Evolutionary Computation (EC) [12, 2]. In EC algorithms a population of candidate problem solutions is evolved over time via selection mutation, multiple-parent crossover and other operations inspired by biological reproduction. The field of EC arose initially out of work in artificial life [13] and has since been applied in a number of domains including design and decision making.

EC is a stepwise algorithm that starts with a population of randomly generated or externally defined candidate solutions. The fitness of each individual is assessed either by comparison to a gold standard or a competitive “tournament” selection. Based upon this fitness individuals are then permitted to pass their genetic code to the next by means of cloning, combination of genetic material with other fit members, or random mutation. The algorithm as a whole continues until an absolute fitness threshold is reached or a maximum number of generations has passed.

In genetic programming the individual members of the population are interpreted as function code with their performance compared against a target function or task. For the purposes of our experiments the target function was the mapping $\phi : f \rightarrow b_m$ of graph features f to the mean-score bin b_m we discussed above. In this case the raw fitness is defined by the ratio of correct classifications to total classifications in the set.

One disadvantage of GP is the tendency of systems using unconstrained representations (like ours) to select for code that is not just successful but genetically *robust*. Such code is characterized by *introns* or redundant code elements that protect the core function from destructive crossover and slow the discovery of new solutions. This necessitates the use of *parsimony pressure* to control code growth. In this project we applied a scaled penalty based upon size. Higher performing trees were assessed a larger penalty than lower performers.

In this experiment we made use of two reproduction operators: mutation and crossover. Under mutation an individual is copied directly into the next generation with a sub-tree being replaced by a new, randomly generated sub-tree. Under crossover, two parents exchange randomly selected sub-trees with the children being passed to the next generation.

Members of the population at time t are selected for reproduction based upon their fitness. Some forms of GP select individuals proportionally according to their absolute fitness. However this often results in extreme genetic drift toward initially fit individuals and reduces the selection pressure as the $\sigma_{f,t}$ goes down. We therefore employed sigma scaling to assign each individual a

reproductive fitness value of:

$$Exp(i,t) = \begin{cases} 1 + \frac{f_i - \bar{f}_t}{2\sigma_t} & : \sigma_t \neq 0 \\ 1.0 & : \sigma_t = 0 \end{cases} \quad (1)$$

We then select individuals using Stochastic Universal Sampling [12] which ensures that each individual reproduces at least $\lfloor Exp(i,t) \rfloor$ but no more than $\lceil Exp(i,t) \rceil$ times. Taken together these measures prevent genetic drift by ensuring that selection pressure is still high even as the absolute fitness increases. As a machine learning algorithm, GP has a number of advantages. It is well suited to the evolution of arbitrary structures ranging from neural networks to object-oriented programs. This makes it attractive for our present purposes. However, GP also has a number of disadvantages. As a non-deterministic algorithm it makes fewer guarantees about its performance than a more bounded, biased and specialized algorithm such as C4.5. And, while the statistical behavior of the system and use of proper tuning work to prevent random drift, it cannot be completely eliminated. It is also computationally costly; each of our runs required 12 hours of operation on a modern PC.

3 Results and Analysis

As stated in the introduction, our goal in this study is to examine the prospects for automatically inducing higher-order pedagogical knowledge from subject graphs. By analyzing subject graphs using machine learning methods we seek to identify potential target rules for the classification of successful and unsuccessful learners and to explore the interaction of the graph characteristics.

For purposes of this analysis we made use of the final graphs and post-test scores taken from the Novice-2007 and Expert-2008 studies. In both studies the subjects followed the same procedure and took the same tests. The graphs we analyzed were produced for two competing arguments in the *Burnham v. Superior Court* (495 U.S. 604) case, each of which was represented as a single unified set of graph characteristics as interpreted by LARGO. The post-test score was a single value representing their overall score in the absolute range. We elected to use the final graph students generated in the course of the study as it was created as the culmination of the students' training and thus, was most likely to be correlated with their final performance.

We were forced to remove some of the Novice-2007 subjects from our analysis as they took too little time on the post-test or too little time to read the cases (both indicating a lack of serious effort) or, in the case of four, because they candidly informed us that they were not trying to answer the questions. This left us with 34 students from the Novice-2007 study and 17 from the Expert-2008 study giving us a total of 51 graph/test pairs.

We binned the graph/test pairs according to their post-test score. We then binned the subjects by mean score (0.63) into two groups, those above the mean, and those at or below it. Of the 51 students 22 were below the mean while 29 were above it. Two of the Expert-2008 subjects fell below the mean score. Our χ^2 , C4.5 and GP analyses below are based upon this grouping.

Statistical Comparisons: As we report in [10], simple statistical analyses of the graph features such as the number of nodes or relations do not correlate highly with the students' learning outcomes. This was true both for the full set of 51 subjects as well as the study subgroups. While some of the measures *do* correlate with group membership (i.e., expert students produce more interconnected graphs than non-experts) they do not correlate with students' ultimate performance.

Our analysis showed no overall correlation between the phase groups and student performance. Again while there was some difference between the study groups those differences were not significant. However, once we binned the full set of graph results by mean score a distinction emerged. In particular two of the characteristics were significantly correlated with bin membership. *UNLINKED_HYPO* was significantly correlated with having a less than average score ($c^2(16.16, N = 51) = 1.00, p < 0.001$) as was *UNLINKED_TEST* ($c^2(18.27, N = 51) = 1.00, p < 0.001$). This highlights the importance of students' linking of tests and hypos in their diagrams to the argument transcript. In addition, *TEST_REVISION_SUGGESTED* was marginally significantly correlated with high performance ($c^2(4.07, N = 51) = 1.00, p < 0.05$). As the reader will recall this is a 'late phase' characteristic and requires some successful problem solving steps to occur before it is active. This strong correlation of the linking features with student performance fits our domain model. The connection of diagram elements to the argument transcript enables students to retain the context for each note and helps them to develop a "respect for the text" that is a goal of legal instruction.

The significance of *TEST_REVISION_SUGGESTED* prompted us to examine the student help behavior more closely. As you will recall this characteristic is activated when a student has marked up the target region, completed collaborative feedback, but not changed his test. From that we determined that very few of the students modified their test statements in response to this characteristic. Thus students who reached this point are not sufficiently differentiated. This led us to conclude that additional effort must be made to motivate student's help usage, particularly in the later "reflective" phases of work.

C4.5: We split the individual datapoints evenly into a 90/10 Test-train split with 45 Training cases (19 at or below the mean and 26 above) and 6 test cases (3 at or below and 3 above). We did not perform an iterative cross-validation as our goal was to induce information from known algorithms, not to validate the algorithms on existing data. C4.5 produces the pruned tree that is shown in Figure 2. This tree successfully classifies 82.2% of the training cases and 100% of the test cases. Interestingly the only graph features employed within it are *UNLINKED_TEST* and *TEST_REVISION_SUGGESTED*.

In many respects the tree supports the χ^2 analysis in highlighting the importance of the transcript linking, particularly for test nodes. Students who link their nodes to the transcripts do well while those who do not are split between students who receive a *TEST_REVISION_SUGGESTED* and are above the mean and those who are not. Thus students who perform well in other respects by highlighting the key transcript region, summarizing it, and partially completing collaborative filtering have been able to avoid linking all of their test nodes.

GP: For the Genetic Programming experiments we employed the same test/train split over all as with C4.5. On both the Mean classification task the evolutionary algorithm showed early successes. As of generation 93 the system produced the Mean classification tree shown in Figure 3. This tree correctly classified 87% of training cases successfully and 100% of test cases. Subsequent generations showed some improvements with the system achieving 89% correct classification of training instances as of generation 659. However the resulting trees were quite large suggesting a problem of overfitting the data. Introns were already present at generation 93 as shown by the useless appearances of *NO_ELEMENTS* and *ISOLATED_HYPO_DISCUSS* and the frequency of such code only increased as the process went on. Note also that both the *UNLINKED_TEST* and

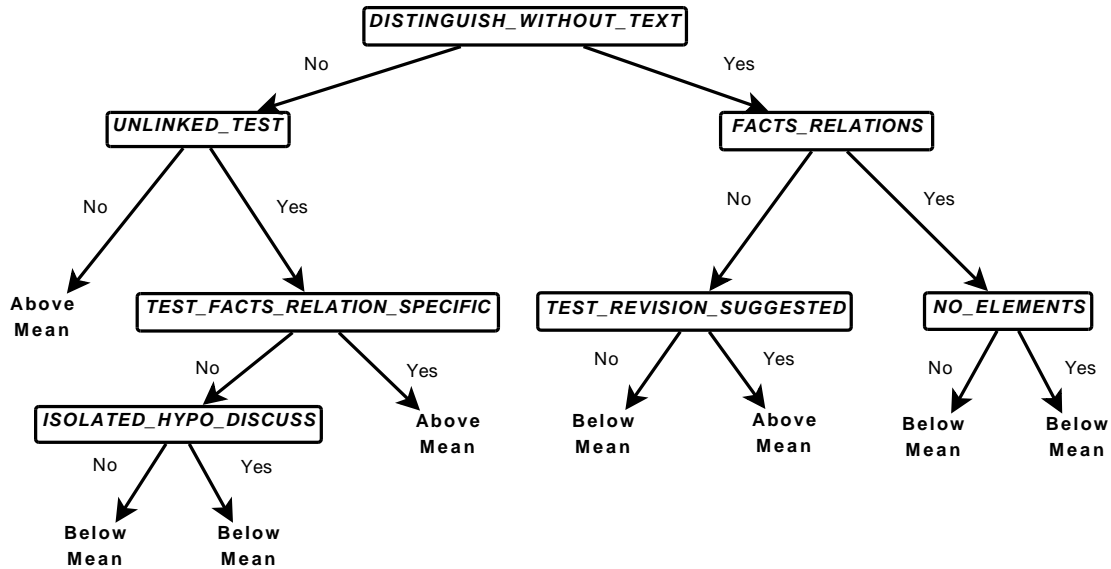


Figure 3. GP Mean Decision Tree.

TEST_REVISION_SUGGESTED rules are present in this tree but not *UNLINKED_HYPO*.

In analyzing the tree shown in Figure 3 we will focus on the three classes of poor performing students that it defines. The root node of the tree is *DISTINGUISH_WITHOUT_TEXT*. This characteristic is active when a student has noted that one node is distinguished from another (irrespective of node type) without giving a justification. As we noted above distinctions, according to our model, are always motivated by some principled or factual justification with which the student should annotate the arc.

As with the C4.5 tree students who exhibit this characteristic alone are rated low unless they also exhibit *TEST_REVISION_SUGGESTED*. The significance of this characteristics prompted us to examine the student help behavior more closely. Recall that this characteristic is activated when a student has marked up the target region, completed collaborative feedback, but not changed his test. From that we determined that very few of the students modified their test statements in response to this characteristic.

Students who exhibit both *DISTINGUISH_WITHOUT_TEXT* and the *FACTS_RELATIONS* characteristic are classified as below the mean. This latter characteristic indicates that the students added arcs relating fact nodes to one another. Again this is a violation of our model. Taken together these characteristics indicate a misunderstanding of the role of facts in the domain model both in terms of how distinctions are drawn using facts and how nodes may be interrelated.

The third class is the set of students who do not exhibit *DISTINGUISH_WITHOUT_TEXT* but do exhibit *UNLINKED_TEST* and *TEST_FACTS_RELATION_SPECIFIC*. This latter characteristic is active when the student has constructed a specific relation (e.g., “Modified To”) to the facts of the case. This is an example of the system’s providing more novel pedagogical information. While our domain model endorses the use of general relationships between the test and fact nodes, it is clear that some good students, who leave tests unlinked, also choose to use specific relations for test and fact nodes. This may signal a valid alternative to our model that bears further exploration.

4 Conclusions

In this paper we assessed the potential of inspectable machine learning methods to induce useful domain information from student work. Our goal was to demonstrate the potential of these methods to yield useful insights into the quality of student solutions, the tutoring system's behavior, and the domain itself. The results we describe above have led us to conclude that these methods do hold potential for domain exploration but not without some measure of guidance.

Our statistical analysis highlighted three salient graph characteristics one of which demonstrated how the use of the system by well-performing students was at odds with our desires. However apart from that it validated our domain model. The use of C4.5 further confirmed the above results and helped to validate some of our domain assumptions but otherwise did not yield much in the way of new information. GP by contrast yielded a set of classification trees one of which we presented here. Examination of this tree yielded useful information both about student misconceptions and student divergence from our domain model. This information has led us to consider alterations to the advice system and a reconsideration of some aspects of our domain model.

These results lead us to conclude that the use of GP to induce "inspectable" classifiers is a fruitful method of data extraction both for behavioral and pedagogical information. We believe that this process is especially useful in ill-defined domains where the relationship among individually detectable solution characteristics is not clear and the means for assessing them, open for debate. In cases such as these the use of inspectable post-hoc classification has been shown to reveal useful insights.

We plan to expand upon this work by moving from the present high-level graph classification into the induction of both lower level graph characteristics and student classifiers that track performance over the course of the study, again with the goal of identifying useful pedagogical and performance information. At the same time we plan to combine these automatic insights with expert human grading. This summer we will engage the services of law school instructors to grade the student graphs. We will then use this data to compare our assessment of good and poor students with theirs and make use of their data to train further classifiers. This will enable us to check the value of our present grading mechanism and to provide expert-level analysis to augment our existing classifications.

Bibliography

- [1] Ryan S.J.d. Baker, Albert T. Corbett, Ido Roll, and Kenneth R. Koedinger. Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction*. (to appear).
- [2] Wolfgang Banzhaf, Peter Nordin, Robert E. Keller, and Frank D. Francone. *Genetic Programming; an Introduction*. Morgan Kaufmann Publishers; San Francisco, 1998.
- [3] Hao Cen, Kenneth R. Koedinger, and Brian Junker. Is over practice necessary? - improving learning efficiency with the cognitive tutor through educational data mining. In Luckin et al. [8], pages 511–518.
- [4] Matthew W. Easterday, Vincent Aleven, and Richard Scheines. 'tis better to construct than to receive? the effects of diagram tools on causal reasoning. In Luckin et al. [8], pages 93–100.

- [5] Mingyu Feng, Niel T. Heffernan, and Kenneth R. Koedinger. Predicting state test scores better with intelligent tutoring systems: Developing metrics to measure assistance required. In Ikeda et al. [7], pages 31–40.
- [6] Andreas Harrer, Rakheli Hever, and Sabrina Ziebarth. Empowering researchers to detect interaction patterns in e-collaboration. In Luckin et al. [8], pages 503–510.
- [7] Mitsuru Ikeda, Kevin Ashley, and Tak Wai Chan, editors. *Proc. of the 8th International Conference on Intelligent Tutoring Systems. Jhongli Taiwan*. Springer-Verlag Berlin, 2006.
- [8] Rosemary Luckin, Kenneth R. Koedinger, and Jim Greer, editors. *Proc. of the 13th International Conference on Artificial Intelligence in Education (AIED2007)*. Amsterdam (Niederlande), IOS Press., 2007.
- [9] Collin Lynch, Kevin Ashley, Vincent Aleven, and Niels Pinkwart. Defining ill-defined domains; a literature survey. In Vincent Aleven, Kevin Ashley, Collin Lynch, and Niels Pinkwart, editors, *Proc. of the Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 8th International Conference on Intelligent Tutoring Systems. Jhongli (Taiwan), National Central University.*, pages 1–10, 2006.
- [10] Collin Lynch, Niels Pinkwart, Kevin Ashley, and Vincent Aleven. What do argument diagrams tell us about students’ aptitude or experience? a statistical analysis in an ill-defined domain. In Vincent Aleven, Kevin Ashley, Collin Lynch, and Niels Pinkwart, editors, *Proc. of the Third International Workshop on Intelligent Tutoring Systems for Ill-Defined Domains at the 9th International Conference on Intelligent Tutoring Systems. UQUAM Montreal, Canada.*, pages 1–9, 2008. (to appear).
- [11] Jan Miksatko and Bruce M. McLaren. What’s in a cluster? automatically detecting interesting interactions in student e-discussions. In *Proc. of the 9th International Conference on Intelligent Tutoring Systems. UQUAM Montreal, Canada.*, 2008. (to appear).
- [12] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge Massachusetts, 1999.
- [13] Melanie Mitchell and Stephanie Forrest. Genetic algorithms and artificial life. In Christopher G. Langton, editor, *Artificial Life; An Overview*, pages 267–289. MIT Press, Cambridge Massachusetts, 1999.
- [14] Niels Pinkwart, Vincent Aleven, Kevin Ashley, and Collin Lynch. Toward legal argument instruction with graph grammars and collaborative filtering techniques. In Ikeda et al. [7], pages 227–236.
- [15] Niels Pinkwart, Vincent Aleven, Kevin Ashley, and Collin Lynch. Evaluating legal argument instruction with graphical representations using largo. In Luckin et al. [8], pages 101–108.
- [16] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers; San Francisco, 1993.