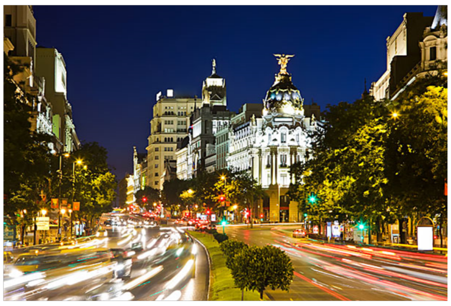
EDM 2015

The 8th International Conference on Educational Data Mining

26-29 June 2015
Madrid - Spain



EDM 2015 Workshops Proceedings

8th International Conference on Educational Data Mining

(EDM 2015)

Madrid, Spain – June 26-29 2015

Edited by

Kaśka Porayska-Pomsta
UCL Institute of Education, UK

Katrien Vebert
KU Leuven, Belgium

EDM 2015 Workshops Proceedings

Preface

This volume compiles the papers accepted for publication in the two workshops taking place during the 8th International Conference in Educational Data Mining (EDM 2015). The third workshop is collocated with the 17th International Conference on Artificial Intelligence in Education. This is the second year in which call for workshops has been issued under the auspices of the Educational Data Mining conference. The fact that workshops submitted to EDM and AIEd conferences are collocated is not incidental, demonstrating a complementarity and common roots of the two research disciplines.

The purpose of the EDM workshops is to provide targeted forum for academics, industry experts, and other interested parties to present and critique existing state of the art in EDM and to generating novel ideas of relevance to Educational Data Mining. EDM requires adapting existing approaches or developing new approaches that build upon techniques from a combination of areas, including but not limited to statistics, psychometrics, machine learning, information retrieval, recommender systems, and scientific computing. Given its focus on Education EDM represents a unique data science in that first and foremost its concerned is interpreting data for the specific purpose of informing education be it technology enhanced or otherwise.

The workshops held in conjunction with the 8th EDM and 17th AIEd conferences include:

Graph-based Educational Data Mining
Organisers: Collin F. Lynch, Tiffany Barnes, Jennifer Albert, Michael Eagle

Tools and Technologies in Statistics, Machine Learning and Information Retrieval for Educational Data Mining
Organisers: Marian Cristian Mihăescu, Mihai Mocanu, Costel Ionascu, Mirel Cosulschi

International Workshop on Affect, Meta-Affect, Data and Learning
Organisers: Genaro Rebolledo-Mendez, Manolis Mavrikis, Olga C. Santos, Benedict du Boulay, Beate Grawemeyer and Rafael Rojano-Cáceres

Two tutorials are also held at the EDM and these include:

Using Natural Language Processing Tools in Educational Data Mining
Organisers: Scott Crossley, Laura Allen, Danielle McNamara

Student Modeling Applications, Recent Developments & Toolkits
Organisers: José González-Brenes, Kai-Min Chang, Michael Yudelson, Yoav Bergner, Yun Huang, University Of Pittsburgh

We would like to thank all workshop organisers for their involvement in the organisation of the workshops and their cooperation, as well as their efforts in attracting contributions and participants to their workshops.

Kaśka Porayska-Pomsta & Katrien Verbert

Workshop

on

Graph-based Educational Data Mining

Graph-based Educational Data Mining (G-EDM 2015)

Collin F. Lynch
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
cflynch@ncsu.edu

Dr. Tiffany Barnes
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
tmbarnes@ncsu.edu

Dr. Jennifer Albert
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
jlsharp@ncsu.edu

Michael Eagle
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
mjeagle@ncsu.edu

1. INTRODUCTION

Fundamentally, a graph is a simple concept. At a basic level a graph is a set of relationships $\{e(n_0,n_2),e(n_0,n_j),...,e(n_{j-1},n_j)\}$ between elements. This simple concept, however, has afforded the development of a complex theory of graphs [1] and rich algorithms for combinatorics [7] and clustering [4]. This has, in turn, made graphs a fundamental part of educational data mining.

Many types of data can be naturally represented as graphs such as social network data, user-system interaction logs, argument diagrams, logical proofs, and forum discussions. Such data has grown exponentially in volume as courses have moved online and educational technology has been incorporated into the traditional classroom. Analyzing it can help to answer a range of important questions such as:

- What path(s) do high-performing students take through online educational materials?
- What social networks can foster or inhibit learning?
- Do users of online learning tools behave as the system designers expect?
- What diagnostic substructures are commonly found in student-produced diagrams?
- Can we use prior student data to identify students' solution plan, if any?
- Can we use prior student data to provide meaningful hints in complex domains?
- Can we identify students who are particularly helpful based upon their social interactions?

Thus, graphs are simple in concept, general in structure, and have wide applications for Educational Data Mining (EDM). Despite the importance of graphs to data mining and data analysis there exists no strong community of researchers focused on Graph-Based Educational Data Mining. Such a community is important to foster useful interactions, share tools and techniques, and to explore common problems.

2. GEDM 2014

This is the second workshop on Graph-Based Educational Data Mining. The first was held in conjunction with EDM 2014 in London [17]. The focus of that workshop was on seeding an initial community of researchers, and on identifying shared problems, and avenues for research. The papers presented covered a range of topics including unique visualizations [13], social capital in educational networks [8], graph mining [19, 11], and tutor construction [9].

The group discussion sections at that workshop focused on the distinct uses of graph data. Some of the work presented focused on student-produced graphs as solution representations (e.g. [14, 3]) while others focused more on the use of graphs for large-scale analysis to support instructors or administrators (e.g. [18, 13]). These differing uses motivate different analytical techniques and, as participants noted, change our underlying assumptions about the graph structures in important ways.

3. GEDM 2015

Our goal in this second workshop was to build upon this nascent community structure and to explore the following questions:

1. What common goals exist for graph analysis in EDM?
2. What shared resources such as tools and repositories are required to support the community?
3. How do the structures of the graphs and the analytical methods change with the applications?

The papers that we include here fall into four broad categories: interaction, induction, assessment, and MOOCs.

Work by Poulovassilis et al. [15] and Lynch et al. [12] focuses on analyzing user-system interactions in state based learning environments. Poulovassilis et al. focuses on the analyses of individual users' solution paths and presents a novel mechanism to query solution paths and identify general solution strategies. Lynch et al. by contrast, examined user-system interactions from existing model-based tutors to examine the impact of specific design decisions on student performance.

Price & Barnes [16] and Hicks et al. [6] focus on applying these same analyses in the open-ended domain of programming. Unlike more discrete tutoring domains where users enter single equations or select actions, programming tutors allow users to make drastic changes to their code on each step. This can pose challenges for data-driven methods as the student states are frequently unique and admit no easy single-step advice. Price and Barnes present a novel method for addressing the data sparsity problem by focusing on minimal-distance changes between users [16] while in related work Hicks et al. focuses on the use of path weighting to select actionable advice in a complex state space [6].

The goal in much of this work is to identify rules that can be used to characterize good and poor interactions or good and poor graphs. Xue at al. sought address this challenge in part via the automatic induction of graph rules for student-produced diagrams [22]. In their ongoing work they are applying evolutionary computation to the induction of Augmented Graph Grammars, a graph-based formalism for rules about graphs.

The work described by Leo-John et al. [10], Guerra [5] and Weber & Vas [21], takes a different tack and focuses not on graphs representing solutions or interactions but on relationships. Leo-John et al. present a novel approach for identifying closely-related word problems via semantic networks. This work is designed to support content developers and educators in examining a set of questions and in giving appropriate assignments. Guerra takes a similar approach to the assessment of users' conceptual changes when learning programming. He argues that the conceptual relationship graph affords a better mechanism for automatic assessment than individual component models. This approach is also taken up by Weber and Vas who present a toolkit for graph-based self-assessment that is designed to bring these conceptual structures under students' direct control.

And finally, Vigentini & Clayphan [20], and Brown et al. [2] focus on the unique problems posed by MOOCs. Vigentini and Clayphan present work on the use of graph-based metrics to assess students' on-line behaviors. Brown et al., by contrast, focus not on local behaviors but on social networks with the goal of identifying stable sub-communities of users and of assessing the impact of social relationships on users' class performance.

4. REFERENCES

[1] B. Bollobás. *Modern Graph Theory.* Springer Science+Business Media Inc. New York, New York, U.S.A., 1998.

[2] R. Brown, C. F. Lynch, Y. Wang, M. Eagle, J. Albert, T. Barnes, R. Baker, Y. Bergner, and D. McNamara. Communities of performance & communities of preference. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015).* CEUR-WS, June 2015. (in press).

[3] R. Dekel and K. Gal. On-line plan recognition in exploratory learning environments. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2014.

[4] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[5] J. Guerra. Graph analysis of student model networks. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015).* CEUR-WS, June 2015. (in press).

[6] A. Hicks, V. Catete, R. Zhi, Y. Dong, and T. Barnes. Bots: Selecting next-steps from player traces in a puzzle game. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015).* CEUR-WS, June 2015. (in press).

[7] D. E. Knuth. *The Art of Computer Programming: Combinatorial Algorithms, Part 1*, volume 4A. Addison-Wesley, 1^{st} edition, 2011.

[8] V. Kovanovic, S. Joksimovic, D. Gasevic, and M. Hatala. What is the source of social capital? the association between social network position and social presence in communities of inquiry. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2014.

[9] R. Kumar. Cross-domain performance of automatic tutor modeling algorithms. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2014.

[10] R.-J. Leo-John, T. McTavish, and R. Passonneau. Semantic graphs for mathematics word problems based on mathematics terminology. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015).* CEUR-WS, June 2015. (in press).

[11] C. F. Lynch. AGG: augmented graph grammars for complex heterogeneous data. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2014.

[12] C. F. Lynch, T. W. Price, M. Chi, and T. Barnes. Using the hint factory to analyze model-based tutoring systems. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015).* CEUR-WS, June 2015. (in press).

[13] T. McTavish. Facilitating graph interpretation via interactive hierarchical edges. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings.* CEUR-WS.org, 2014.

[14] B. Mostafavi and T. Barnes. Evaluation of logic proof problem difficulty through student performance data. In S. G. Santos

and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[15] A. Poulovassilis, S. G. Santos, and M. Mavrikis. Graph-based modelling of students' interaction data from exploratory learning environments. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[16] T. Price and T. Barnes. An exploration of data-driven hint generation in an open-ended programming problem. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[17] S. G. Santos and O. C. Santos, editors. *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[18] V. Sheshadri, C. Lynch, and T. Barnes. Invis: An EDM tool for graphical rendering and analysis of student interaction data. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[19] K. Vaculík, L. Nezvalová, and L. Popelínsky. Graph mining and outlier detection meet logic proof tutoring. In S. G. Santos and O. C. Santos, editors, *Proceedings of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2014.

[20] L. Vigentini and A. Clayphan. Exploring the function of discussion forums in moocs: comparing data mining and graph-based approaches. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[21] C. Weber and R. Vas. Studio: Ontology-based educational self-assessment. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

[22] L. Xue, C. F. Lynch, and M. Chi. Graph grammar induction by genetic programming. In C. F. Lynch, T. Barnes, J. Albert, and M. Eagle, editors, *Proceedings of the Second International Workshop on Graph-Based Educational Data Mining (GEDM 2015)*. CEUR-WS, June 2015. (in press).

Graph Grammar Induction via Evolutionary Computation

Linting Xue
Electrical Engineering Department
North Carolina State University
Raleigh, North Carolina, U.S.A.
lxue3@ncsu.edu

Collin F.Lynch
Computer Science Department
North Carolina State University
Raleigh, North Carolina, U.S.A.
cflynch@ncsu.edu

Min Chi
Computer Science Department
North Carolina State University
Raleigh, North Carolina, U.S.A.
mchi@ncsu.edu

ABSTRACT

Augmented Graph Grammars provide a robust formalism for representing and evaluating graph structures. With the advent of robust graph libraries such as AGG, it has become possible to use graph grammars to analyze realistic data. Prior studies have shown that graph rules can be used to evaluate student work and to identify empirically-valid substructures using hand-authored rules. In this paper we describe proposed work on the automatic induction of graph grammars for student data using evolutionary computation via the pyEC system.

Keywords

Augmented Graph Grammars, Graph Data, Evolutionary Computation

1. INTRODUCTION

Graph Grammars are logical rule representations for graph structures. They can be designed to encode classes of suitable graphs to recognize complex sub-features. They were introduced by Rosenfeld and Pfaltz in 1969 as "Context-free web grammars" [1]. Since then Graph grammars have been applied to a wide range of areas, including pattern recognition [2, 3, 4]; visual programming languages [5]; biological development [6]; classification of chemical compounds [7, 8]; and social network analysis [9, 10, 11]. Simple graph grammars are, like string grammars, composed of a set of production rules that map from one structure to another. In this case the rules map from a simple subgraph, typically a single node or arc, to a more complex structure. As with string grammars the node and arc types are drawn from finite alphabets. Despite their utility, however, graph grammars are very difficult to construct. The data structures required are complex [5]. Moreover, development of suitable graph grammars generally requires considerable domain expertise. Most existing uses of graph grammars have relied on hand-authored rules.

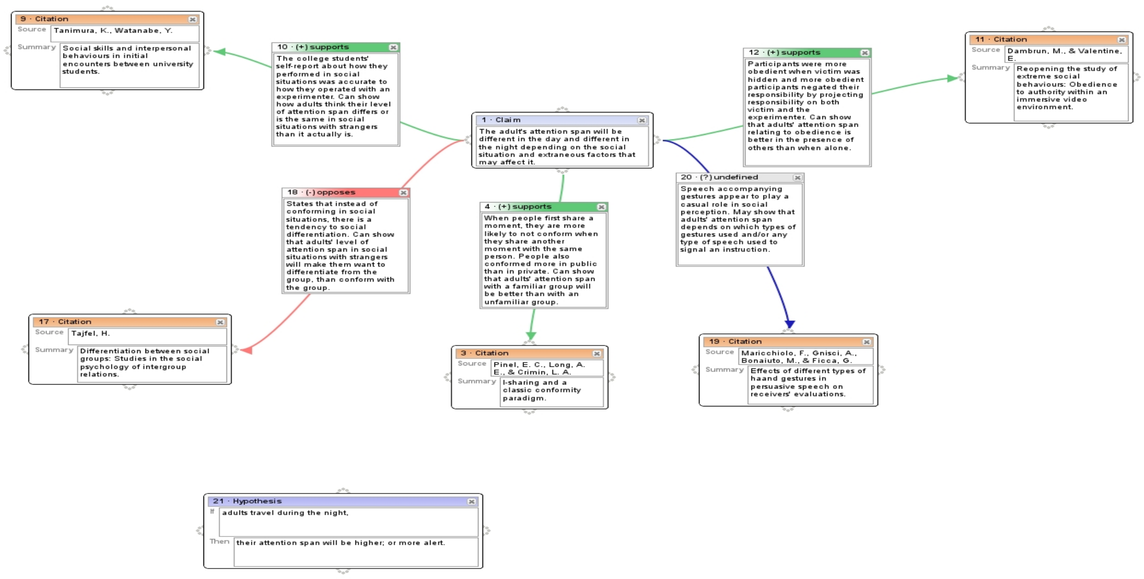
In this paper we describe our ongoing work on the automatic induction of Augmented Graph Grammars via Evolutionary Computation (EC). Our long-term goal in this work is to develop automated techniques that can extract empirically-valid graph rules which can, in turn, be used to classify student-produced argument diagrams and to provide the basis for automated student guidance and evaluation. This will build upon our prior on the evaluation of *a-priori* rules for student arguments. We will begin with background material on Augmented Graph Grammars and discuss prior work on grammar induction. We will then present an overview of our planned work.

2. AUGMENTED GRAPH GRAMMARS & ARGUMENT DIAGRAMS

Classical graph grammars are designed to deal with fixed graphs that are composed from a finite set of static node and arc types. Augmented graph grammars are an extension of simple graph grammars that allow for complex node and arc types, optional substructures, and complex rule expressions [12]. Rather than using a fixed alphabet of graph components they are defined by a complex ontology that allows for subsidiary types such as textual fields, access functions, and directional information. They can also be used to evaluate negated elements as well as quantified expressions. As such they are better suited to rich graph data such as user-system interaction logs and student-produced argument diagrams.

Augmented Graph Grammars have previously been used for the detection of empirically-valid substructures in student-produced argument diagrams [13, 14]. In that work *a-priori* rules were used to represent key discussion features and argumentative flaws. Argument diagrams are graphical argument representations that reify key features of arguments such as hypothesis statements, claims, and citations as nodes and the supporting, opposing, and informational relationships as arcs between them.

A sample diagram collected in that work is shown in Figure 1 The diagram includes a central research *claim* node, which has a single text field indicating the content of the research claim. A set of *citation* nodes are connected to the *claim* node via a set of *supporting, opposing* and *undefined* arcs colored with green, red and blue respectively. Each citation node contains two fields: one for the citation information, and the other for a summary of the cited work; each arc has a single text field explaining why the relationship holds. At



9 · Citation
Source: Tanimura, K., Watanabe, Y.
Summary: Social skills and interpersonal behaviours in initial encounters between university students.

10 · (+) supports
The college students' self-report about how they performed in social situations was accurate to how they operated with an experimenter. Can show how adults think their level of attention span differs or is the same in social situations with strangers than it actually is.

1 · Claim
The adult's attention span will be different in the day and different in the night depending on the social situation and extraneous factors that may affect it.

12 · (+) supports
Participants were more obedient when victim was hidden and more obedient participants negated their responsibility by projecting responsibility on both victim and the experimenter. Can show that adults' attention span relating to obedience is better in the presence of others than when alone.

11 · Citation
Source: Dambrun, M., & Valentine, E.
Summary: Reopening the study of extreme social behaviours: Obedience to authority within an immersive video environment.

18 · (-) opposes
States that instead of conforming in social situations, there is a tendency to social differentiation. Can show that adults' level of attention span in social situations with strangers will make them want to differentiate from the group, than conform with the group.

4 · (+) supports
When people first share a moment, they are more likely to not conform when they share another moment with the same person. People also conformed more in public than in private. Can show that adults' attention span with a familiar group will be better than with an unfamiliar group.

20 · (?) undefined
Speech accompanying gestures appear to play a casual role in social perception. May show that adults' attention span depends on which types of gestures used and/or any type of speech used to signal an instruction.

17 · Citation
Source: Tajfel, H.
Summary: Differentiation between social groups: Studies in the social psychology of intergroup relations.

3 · Citation
Source: Pinel, E. C., Long, A. E., & Crimin, L. A.
Summary: I-sharing and a classic conformity paradigm.

19 · Citation
Source: Maricchiolo, F., Gnisci, A., Bonaiuto, M., & Ficca, G.
Summary: Effects of different types of haand gestures in persuasive speech on receivers' evaluations.

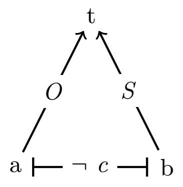
21 · Hypothesis
If: adults travel during the night,
Then: their attention span will be higher; or more alert.

Figure 1: A student-produced LASAD argument diagram representing an introductory argument.

the bottom of the diagram, there is a single isolated *hypothesis* node that contains two text fields, one for a *conditional* or IF field, and the other for *conditional* or THEN field. We expect the induced graph grammars from a set of argument diagrams can be used to evaluate the student thesis work.

Figure 2 shows an *a-priori* rule that was defined as part of that work. This rule is designed to identify a subgraph where a single target node t is connected to two separate citation nodes a and b such that: a is connected to t via an opposing path; b is connected via a supporting path; and there exists no comparison arc between a and b. The rules in that study were implemented using AGG an augmented graph grammar library built in Python [12]. AGG matches the graphs using recursive stack-based algorithm. The code first matches the ground nodes at the top-level of the class (t, a, & b). It then tests for the recursive productions O, and S, before finally testing for the negated comparison arc c. This rule does not make use of the full range of potential capacity for Augmented Graph Grammars. However it is illustrative of the type of rules we plan to induce here, rules that generalize beyond basic types and draw on existing production classes but not, at least in the immediate term, use complex textual elements or functional features.

$(ParedWcomp)$



$$\begin{Bmatrix} t.Tpye = \text{“}claim'' or \text{“}hypothesis'' \\ a.Type = \text{“}Citation'' \\ b.Type = \text{“}Citation'' \\ c.Type = \text{“}Comparison'' \end{Bmatrix}$$

Figure 2: A simple augmented graph grammar rule that detects compared counterarguments. The rule shows a two citation nodes (a, & b) that have opposing relationships with a shared claim node (t) and do not have a comparison arc (c) drawn between them. The arcs S and O represent recursive supporting and opposing path.

3. GRAMMAR INDUCTION

Graph and relational data has grown increasingly prevalent and graph analysis algorithms have been applied in a wide range of domains from social network analysis [15] to bioinformatics [16]. Most of this work falls into one of two categories of algorithms: frequent subgraph matching, and graph compression.

A number of algorithms have been developed to discover frequent subgraphs. These include the gSpan algorithm developed by Yan and Han [17]; the AGM algorithm developed by Inokuchi et al [18]; and the FSG algorithm developed by Kuramochi and Karypis which is based on the previous Apriori algorithm [19]. They are based upon controlled graph walks coupled with indexing. While these algorithms are effective, particularly on smaller graphs, with low vertex degree they can also overfit simpler graph structures and they do not scale well to larger, denser graph data [20].

The SUBDUE system takes a greedy-compression approach to graph mining. SUBDUE searches for candidate subgraphs that can best compress the input graphs by replacing a candidate subgraph with a single vertex. Then nodes and arcs are added to the vertices to form new candidate subgraphs. The process is recursive and relies on the Minimum-Description-Length (MDL) principle to evaluate the candidates. SUBDUE has been applied successfully to extract structure from visual programming [5], web search [21], and analyzing user behaviors in games [22].

While these methods are successful they have practical and theoretical limitations. Both classes of approaches are limited to static graphs composed from a finite alphabet of node and arc types. The frequent subgraph approaches are based upon iterative graph walks and can be computationally expensive and are limited to finding exact matches. They do not generalize beyond the exact graphs matched nor do they allow for recursive typing. SUBDUE, by contrast is a greedy algorithm that finds the single most descriptive grammar and does not allow for weighted matches.

For our present purposes, however, our goal is to identify multiple heirarchical classes of the type shown in Figure 2 that can: generalize beyond exact node and arc types; can draw on recursive rule productions; and can be weighted based upon the graph quality. Moreover our long-term goal with this work is to explore graph rule induction mechanisms that can be expanded to include textual rules and complex constraints. For that reason we have elected to apply evolutionary computation. This is a general-purpose machine learning mechanism that can be tunes to explore a range of possible induction mechanisms.

4. METHODS

4.1 Evolutionary Computation

Evolutionary Computation (EC) is a general class of machine learning and optimization methods that are inspired by the process of Darwinian evolution through natural selection [23] such as Genetic Algorithms [24] or Genetic Programming [25]. EC algorithms begin with a population of randomly generated candidate solutions such as snippets of random code, strings representing a target function, or formal rules. Each of these solutions is ranked by a *fitness function* that is used to evaluate the quality of the individuals. These functions can be defined by absolute measures of success such as a suite of test cases, or by relative measures such as a competition between chess-playing systems.

Once the individuals have been ranked a new generation of individuals is produced through a combination of crossover and mutation operations. *Crossover* operations combine two or more parents to produce one or more candidate children. In Genetic Algorithms where the candidate solutions are represented as strings this can be accomplished by splitting two parents at a given index and then exchanging the substrings to produce novel children. In Genetic Programming the parents exchange blocks of code, functions, or subtrees. *Mutation* operations alter randomly-selected parts of a candidate solution by swapping out one symbol or instruction for another, adding new sub-solutions, or deleting components. This process of ranking and regeneration will iterate until a target performance threshold is reached or a maximum number of generations has passed.

EC methods are highly general algorithms that can be readily adapted to novel domains by selecting an appropriate solution representation and modification operations. Thus, in contrast to more specific methods such as SUBDUE, the EC algorithm allows us to tune the inductive bias of our search and to explore alternative ways of traversing the solution space. Therefore it is well suited to our present needs. This flexibility is costly, however, as EC is far more computationally expensive than more specialized algorithms, and applications of EC can require a great deal of tuning for each use. In the subsections below we will describe the fitness function and the operators that we will use in this work. For this work we will rely on *pyEC* a general purpose evolutionary computation engine that we have developed [26].

4.2 Dataset

Our initial analysis will be based upon a corpus of expert graded student produced argument diagrams and essays previously described in [13, 14]. That dataset was collected as part of a study on students' use of argument diagrams for writing that was conducted at the University of Pittsburgh in 2011. For that study we selected a set of students in an undergraduate-level course on Psychological Research Methods. As part of the course the students were tasked with planning and executing an empirical research study and then drafting a written report. The students were permitted to work individually or in teams. This report was structured as a standard empirical workshop paper. Prior to drafting the report the students were tasked with diagramming the argument that they planned to make using LASAD an online tool for argument diagramming and annotation.

Subsequent to this data collection process the diagrams and essays were graded by an experienced TA using a set of parallel grading rubrics. These rubrics focused on the quality of the arguments in the diagrams and essays and were used to demonstrate that the structure and quality of the diagrams can be used to predict the students' subsequent essay performance. These grades will be used as the weighting metric for the diagrams and will be correlated with performance as part of the fitness function we describe below. After completion of the data collection, grading, and testing phases and accounting for student dropout and incomplete assignments we collected 105 graded diagram-essay pairs 74 of which were authored by teams.

4.3 Solution Representation

For the purposes of our present experiments we will use a restricted solution representation that relies on a subset of the augmented graph grammar formalism exemplified by the rule shown in Figure 2. This will include only element types and recursive productions. In future work we plan to support the induction of more complex rules defined by multiple graph classes, novel productions, and expressions. However for the present study we will focus on the simple case of individual classes coupled with predefined productions.

4.4 Fitness Function

We plan to use the frequency correlation metric previously employed in [13, 14]. In that study the authors assessed the *empirical validity* of a set of *a-priori* diagram rules. The validity of each individual rule was assessed by testing the correlation between the frequency of the class in the existing graph and the graph grade. The strength of that correlation was estimated using Spearman's ρ a non-parametic measure of correlation [27]. In that work the authors demonstrated that the *a-priori* rule frequency was correlated with students' subsequent essay grades and showed that the frequencies could be used to predict students' future performance.

4.5 Mutation

Our mutation operator will draw on the predefined graph ontology to make atomic changes to an existing graph class. The change will be one of the following operations:

Change Node change an existing node's type.

Change Arc Change an existing arc's type or orientation.

Delete Node Delete a node and its associated arcs.

Delete Arc Delete an existing arc.

Add Node Add a novel node with a specified type.

Add Arc Add an arc between existing nodes or add with new nodes.

4.6 Crossover

By design the crossover operation should, like genetic crossover, be conservative. Two very similar parents should produce similar offspring. Crossover operations should therefore preserve good building blocks and sub-solutions or *introns* through random behavior [25]. Arbitrary graph alignment and crossover is a challenging problem that risks causing unsustainable changes on each iteration. We therefore treat graph crossover as a matrix problem.

For each pair of parent classes we will define a pair of diagonal matricies of the type illustrated in Figure 3. The letter indicies on the top and right indicate nodes while the numerical indicies internally indicate arcs, and the ∅ symbol indicates that no arc is present. The matricies are generated in a canonical order based upon the order in which the nodes were added to the class. Thus on each iteration of the crossover process the corresponding elements will obtain the same index. As a consequence good subsolutions will obtain the same location and will tend to be preserved over time.

Once a set of parent matricies has been generated we then generate two child matricies of the same size as the parents and then randomly select the node and arc members. In the example shown in figures 3 and 4 the parents have nodes $\{A,B,C,D\}$ and $\{E,F,G\}$ while the children have $\{E,B,G,D\}$ and $\{A,F,C\}$. Thus we align the nodes in canonical order and, for each node pair, we flip a coin to decide where they are copied. If one parent is larger than the other than any additional nodes, in this case D, will be copied to the larger child. We then perform a comparable exchange process for the arcs. Each arc or potential arc is defined uniquely in the matricies by its endpoints. We thus align the lists of arcs in a comparable manner and then decide randomly which arc, or empty arc, to copy. As with the nodes, extra arcs from the larger parent, in this case *3,5,* and one ∅ are copied directly into the larger of the two children.

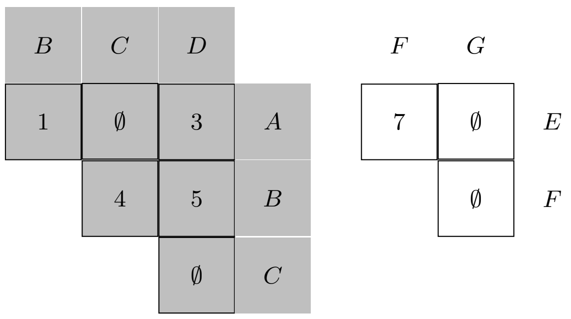


Figure 3: Canonical matricies for crossover parents.

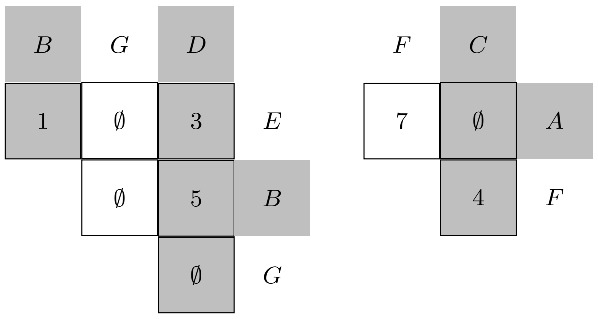


Figure 4: Canonical matricies for crossover children.

5. FUTURE WORK

In this paper we presented a method for the induction of augmented graph grammars through evolutionary computation. We are presently applying this work to the automatic induction of empirically-valid rules for student-produced argument diagrams. This work will serve to extend our prior efforts on the use of augmented graph grammars for student grading and feedback. This work represents an improvement over prior graph grammar induction algorithms which are limited to classical graph grammars and greedy extraction. This work also represents an extension for evolutionary computation by shifting it into a new domain. As part of this work we also plan to explore additional extensions to the standard evolutionary computation algorithm to address problems of over-fitting such as χ^2 reduction.

6. REFERENCES

[1] John L Pfaltz and Azriel Rosenfeld. Web grammars. In *Proceedings of the 1st international joint conference on Artificial intelligence*, pages 609–619. Morgan Kaufmann Publishers Inc., 1969.

[2] John L Pfaltz. Web grammars and picture description. *Computer Graphics and Image Processing*, 1(2):193–220, 1972.

[3] Horst Bunke. Attributed programmed graph grammars and their application to schematic diagram interpretation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4(6):574–582, 1982.

[4] Michihiro Kuramochi and George Karypis. Finding frequent patterns in a large sparse graph*. *Data mining and knowledge discovery*, 11(3):243–271, 2005.

[5] Keven Ates, Jacek Kukluk, Lawrence Holder, Diane Cook, and Kang Zhang. Graph grammar induction on structural data for visual programming. In *Tools with Artificial Intelligence, 2006. ICTAI'06. 18th IEEE International Conference on*, pages 232–242. IEEE, 2006.

[6] Francesc Rosselló and Gabriel Valiente. Graph transformation in molecular biology. In *Formal Methods in Software and Systems Modeling*, pages 116–133. Springer, 2005.

[7] Luc Dehaspe, Hannu Toivonen, and Ross D King. Finding frequent substructures in chemical compounds. In *KDD*, volume 98, page 1998, 1998.

[8] Stefan Kramer, Luc De Raedt, and Christoph Helma. Molecular feature mining in hiv data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 136–143. ACM, 2001.

[9] Wenke Lee and Salvatore J Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM transactions on Information and system security (TiSSEC)*, 3(4):227–261, 2000.

[10] Calvin Ko. Logic induction of valid behavior specifications for intrusion detection. In *Proceedings of the IEEE Symposium on Security and Privacy. (S&P 2000)*, pages 142–153. IEEE, 2000.

[11] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proceedings of the 20th International Conference on very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[12] Collin F Lynch. Agg: Augmented graph grammars for complex heterogeneous data. In *Proceedings of the first international workshop on Graph-Based Educational Data Mining (GEDM 2014)*.

[13] Collin F. Lynch and Kevin D. Ashley. Empirically valid rules for ill-defined domains. In John Stamper and Zachary Pardos, editors, *Proceedings of The* 7^{th} *International Conference on Educational Data Mining (EDM 2014)*. International Educational Datamining Society IEDMS, 2014.

[14] Collin F. Lynch, Kevin D. Ashley, and Min Chi. Can diagrams predict essay grades? In Stefan Trausan-Matu, Kristy Elizabeth Boyer, Martha E. Crosby, and Kitty Panourgia, editors, *Intelligent Tutoring Systems*, Lecture Notes in Computer Science, pages 260–265. Springer, 2014.

[15] Sherry E. Marcus, Melanie Moy, and Thayne Coffman. Social network analysis. In Diane J. Cook and Lawrence B. Holder, editors, *Mining Graph Data*, chapter 17, pages 443–468. John Wiley & Sons, 2006.

[16] Chang Hun You, Lawrence B. Holder, and Diane J. Cook. Dynamic graph-based relational learning of temporal patterns in biological networks changing over time. In Hamid R. Arabnia, Mary Qu Yang, and Jack Y. Yang, editors, *BIOCOMP*, pages 984–990. CSREA Press, 2008.

[17] Xifeng Yan and Jiawei Han. gspan: Graph-based substructure pattern mining. In *Proceedings of the IEEE International Conference on Data Mining (ICDM 2002)*, pages 721–724. IEEE, 2002.

[18] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In *Principles of Data Mining and Knowledge Discovery*, pages 13–23. Springer, 2000.

[19] Michihiro Kuramochi and George Karypis. Frequent subgraph discovery. In *Proceedings IEEE International Conference on Data Mining. (ICDM 2001)*, pages 313–320. IEEE, 2001.

[20] MICHIHIRO Kuramochi and George Karypis. Finding topological frequent patterns from graph datasets. *Mining Graph Data*, pages 117–158, 2006.

[21] Nitish Manocha, Diane J Cook, and Lawrence B Holder. Cover story: structural web search using a graph-based discovery system. *Intelligence*, 12(1):20–29, 2001.

[22] Diane J. Cook, Lawrence B. Holder, and G. Michael Youngblood. Graph-based analysis of human transfer learning using a game testbed. *IEEE Trans. on Knowl. and Data Eng.*, 19:1465–1478, November 2007.

[23] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* John Murray: Albermarle Street: London, United Kingdom, 6 edition, 1872.

[24] Melanie Mitchell. *An Introduction to Genetic Algorithms.* MIT Press: Cambridge, Massachusetts, 1999.

[25] Wolfgang Banzhaf. *Genetic programming: an introduction on the automatic evolution of computer programs and its applications.* Morgan Kaufmann Publishers ; Heidelburg : Dpunkt-verlag; San Francisco, California, 1998.

[26] Collin F. Lynch, Kevin D. Ashley, Niels Pinkwart, and Vincent Aleven. Argument graph classification with genetic programming and c4.5. In Ryan Shaun Joazeiro de Baker, Tiffany Barnes, and Joseph E. Beck, editors, *The 1st International Conference on Educational Data Mining, Montreal, Québec, Canada, June 20-21, 2008. Proceedings*, pages 137–146, 2008.

[27] Wikipedia. Spearman's rank correlation coefficient — wikipedia, the free encyclopedia, 2013. [Online; accessed 27-February-2013].

Communities of Performance & Communities of Preference

Rebecca Brown
North Carolina State University
Raleigh, NC
rabrown7@ncsu.edu

Collin Lynch
North Carolina State University
Raleigh, NC
cflynch@ncsu.edu

Yuan Wang
Teachers College, Columbia University
New York, NY
elle.wang@columbia.edu

Michael Eagle
North Carolina State University
Raleigh, NC
mjeagle@ncsu.edu

Jennifer Albert
North Carolina State University
Raleigh, NC
jennifer_albert@ncsu.edu

Tiffany Barnes
North Carolina State University
Raleigh, NC
tmbarnes@ncsu.edu

Ryan Baker
Teachers College, Columbia University
New York, NY
ryanshaunbaker@gmail.com

Yoav Bergner
Educational Testing Service
Princeton, NJ
ybergner@gmail.com

Danielle McNamara
Arizona State University
Phoenix, AZ
dsmcnamara1@gmail.com

ABSTRACT

The current generation of Massive Open Online Courses (MOOCs) operate under the assumption that good students will help poor students, thus alleviating the burden on instructors and Teaching Assistants (TAs) of having thousands of students to teach. In practice, this may not be the case. In this paper, we examine social network graphs drawn from forum interactions in a MOOC to identify natural student communities and characterize them based on student performance and stated preferences. We examine the community structure of the entire course, students only, and students minus low performers and hubs. The presence of these communities and the fact that they are homogeneous with respect to grade but not motivations has important implications for planning in MOOCs.

Keywords

MOOC, social network, online forum, community detection

1. INTRODUCTION

The current generation of Massive Open Online Courses (MOOCs) is designed to leverage student interactions to augment instructor guidance. The activity in courses on sites such as Coursera and edX is centered around user forums that, while curated and updated by instructors and TAs, are primarily constructed by students. When planning and building these courses, it is hoped that students will help one another through the course and that interacting with stronger students will help to improve the performance of weaker ones. It has not yet been shown, however, that this type of support occurs in practice.

Prior research on social networks has shown that social groups, even those that gather face-to-face, can fragment into disjoint sub-communities [37]. This small-group separation, if it takes place in an online course, can be considered negative or positive, depending on one's perspective. If poor students communicate only with similarly-floundering peers, then they run the risk of perpetuating misunderstandings and of missing insights discussed by better-performing peers and teaching staff. An instructor may wish to avoid this fragmentation to encourage poor students to connect with better ones.

These enduring subgroups may be beneficial, however, by helping students to form enduring supportive relationships. Research by Li et al. has shown that such enduring relationships can enhance students' social commitment to a course [18]. We believe that this social commitment will in turn help to reduce feelings of isolation and alienation among students in a course. Eckles and Stradley [9] have shown that such isolation is a key predictor of student dropout.

We have previously shown that students can form stable communities and that those communities are homogeneous with respect to performance [3]. However that work did not: show whether these results are consistent with prior work on immediate peer relationships; address the impact of hub students on these results; or discuss whether students' varying goals and preferences motivate the community structure. Our goal in this paper is to build upon our prior work by addressing these issues. In the remainder of this paper we will survey prior educational literature on community formation in traditional and online classrooms. We will then build upon our prior work by examining the impact of hub users. And we will look at the impact of user motivations on community formation.

2. RELATED WORK

2.1 MOOCs, Forums, & Student Performance

A survey of the literature on MOOCs shows the beginnings of a research base generating an abundance of data that has not yet been completely analyzed [19]. According to Seaton et al. [29], most of the time students spend on a MOOC is spent in discussion forums, making them a rich and important data source. Stahl et al. [30] illustrates how through this online interaction students collaborate to create knowledge. Thus students' forum activity is good not only for the individual student posting content or receiving answers, but for the class as a whole. Huang et al. [14] investigated the behavior of the highest-volume posters in 44 MOOC-related forums. These "superposters" tended to enroll in more courses and do better in those courses than the average. Their activity also added to the overall volume of forum content and they left fewer questions unanswered in the forums. Huang et al. also found that these superposters did not suppress the activity of less-active users. Rienties et al. [25] examined the way in which user interaction in MOOCs is structured. They found that allowing students to self-select collaborators is more conducive to learning than randomly assigning partners. Further, Van Dijk et al. [31] found that simple peer instruction is significantly less effective in the absence of a group discussion step, pointing again to the importance of a class discussion forum.

More recently Rosé et al. [27] examined students' evolving interactions in MOOCs using a Mixed-Membership Stochastic Block model which seeks to detect partially overlapping communities. They found that the likelihood that students would drop out of the course is strongly correlated with their community membership. Students who actively participated in forums early in the course were less likely to drop out later. Furthermore, they found one forum sub-community that was much more prone to dropout than the rest of the class, suggesting that MOOC communities are made up of students who behave in similar ways. This community can in turn reflect or impact a student's level of motivation and their overall experience in a course much like the "emotional contagion" model used in the Facebook mood manipulation study by Kramer, Guillroy, and Hancock [16].

Yang et al. [36] also notes that unlike traditional courses students can join MOOCs at different times and observed that students who join a course early are more likely to be active and connected in the forums, and less likely to drop out, than those who join later. MOOCs also attract users with a range of individual motivations. In a standard classroom setting students are constrained by availability, convention, and goals. Few students enroll in a traditional course without seeking to complete it and to get formal credit for doing so. MOOCs by virtue of their openness and flexibility attract a wide range of students with unique personal motivations [10]. Some join the course with the intent of completing it. Others may seek only to brush up on existing knowledge, obtain specific skills, or just watch the videos. These distinct motivations in turn lend themselves to different in-class behaviors including assignment viewing and forum access. The impact of user motivations in online courses has been previously discussed by Wang et al. [32, 33]; we will build upon that work here. Thus it is an open question whether these motivations affect students' community behaviors or not.

2.2 Communities, Hubs, & Peers

Kovanovic et al. [15] examined the relationship between social network position or centrality, and social capital formation in courses. Their work is specifically informed by the Community of Inquiry (COI) framework. the COI framework is focused on distance education and is particularly suited to online courses of the type that we study here. The model views course behavior through three *presences* which mediate performance: cognitive, teaching, and social.

This *social presence* considers the nature and persistence of student interactions and the extent to which they reinforce students' behaviors. In their analysis, the authors sought to test whether network relationships, specifically students' centrality in their social graph, is related to their social performance as measured by the nature and type of their interactions. To that end, they examined a set of course logs taken from a series of online courses offered within a public university. They found that students' position within their social graph was positively correlated with the nature and type of their interactions, thus indicating that central players also engaged in more useful social interactions. They did not extend this work to groups, however, focusing solely on individual hub students.

Other authors have also examined the relationship between network centrality, neighbor relationships, network density, and student performance factors. Eckles and Stradley [9] applied network analysis to student attrition, finding that students with strong social relationships with other students who drop out are significantly more likely to drop out themselves. Rizzuto et al. [26] studied the impact of social network density on student performance. Network density is defined as the fraction of possible edges that are present in a given graph. Thus it is a measure of how "clique-like" the graph is. The authors examined self-reported social networks for students in a large traditional undergraduate psychology course. They found that denser social networks were significantly correlated with performance. However, a dominance analysis [1] showed that this factor was less predictive than pure academic ability. These results serve to motivate a focus on the role of social relationships in student behavior. Their analysis is complicated, however, by their reliance on self-report data which will skew the strength and recency of the reported relationships.

Fire et al. [11] studied student interaction in traditional classrooms, constructing a social network based on cooperation on class assignments. Students were linked based on partnership on group work as well as inferred cooperation based on assignment submission times and IP addresses. The authors found that a student's grade was significantly correlated with the grade of the student with the strongest links to that student in the social network. We perform similar analysis in this paper to examine whether the same correlation exists in MOOCs.

Online student interaction in blended courses has also been linked to course performance. Dawson [8] extracted student and instructor social networks from a blended course's online discussion forums and found that students in the 90th grade percentile had larger social networks than those in the 10th percentile. The study also found that high-performing students primarily associated with other high-performing students and were more likely to be connected to the course instructor, while low-performing students tended to associate with other low-performers. In a blended course, this effect may be offset by face-to-face interaction not captured in the online social network, but if the same separation happens in MOOC communities, low-performing students are less likely to have other chances to learn from high-performing ones.

2.3 Community Detection

One of the primary activities students engage in on forums is question answering. Zhang et al. [38] conducted a social network analysis on an online question-and-answer forum about Java programming. Using vertex in-degree and out-degree, they were able to identify a relatively small number of active users who answered many questions. This allowed the researchers to develop various algorithms for calculating a user's Java expertise. Dedicated question-and-answer forums are more structured than MOOC forums, with question and answer posts identified, but a similar approach might help identify which students in a MOOC ask or answer the most questions.

Choo et al. [5] studied community detection in Amazon product-review forums. Based on which users replied to each other most often, they found communities of book and movie reviewers who had similar tastes in these products. As in MOOC forums, users did not declare any explicit social relationships represented in the system, but they could still be grouped by implicit connections.

In the context of complex networks, a community structure is a subgraph which is more densely connected internally than it is to the rest of the network. We chose to apply the Girvan-Newman edge-betweenness algorithm (GN) [13]. This algorithm takes as input a weighted graph and a target number of communities. It then ranks the edges in the graph by their edge-betweenness value and removes the highest ranking edge. To calculate Edge-betweenness we identify the shortest path $p(a,b)$ between each pair of nodes a and b in the graph. The edge-betweenness of an arc is defined as the number of shortest paths that it participates in. This is one of the centrality measures explored by Kovanovic et al. above [15]. The algorithm then recalculates the edge-betweenness values and iterates until the desired number of disjoint community subgraphs has been produced. Thus the algorithm operates by iteratively finding and removing the highest-value communications channel between communities until the graph is fully segmented. For this analysis, we used the iGraph library [7] implementation of G-N within R [24].

The strength of a candidate community can be estimated by modularity. The *modularity score* of a given subgraph is defined as a ratio of its intra-connectedness (edges within the subgraph) to the inter-connectedness with the rest of the graph minus the fraction of such edges expected if they were distributed at random [13, 35]. A graph with a high modularity score represents a dense sub-community within the graph.

3. DATA SET

This study used data collected from the "Big Data in Education" MOOC hosted on the Coursera platform as one of the inaugural courses offered by Columbia University [32]. It was created in response to the increasing interest in the learning sciences and educational technology communities in using EDM methods with fine-grained log data. The overall goal of this course was to enable students to apply each method to answer education research questions and to drive intervention and improvement in educational software and systems. The course covered roughly the same material as a graduate-level course, Core Methods in Educational Data Mining, at Teachers College Columbia University. The MOOC spanned from October 24, 2013 to December 26, 2013. The weekly course was composed of lecture videos and 8 weekly assignments. Most of the videos contained in-video quizzes (that did not count toward the final grade).

All of the weekly assignments were structured as numeric input or multiple-choice questions. The assignments were graded automatically. In each assignment, students were asked to conduct analyses on a data set provided to them and answer questions about it. In order to receive a grade, students had to complete this assignment within two weeks of its release with up to three attempts for each assignment, and the best score out of the three attempts was counted. The course had a total enrollment of over 48,000, but a much smaller number actively participated. 13,314 students watched at least one video, 1,242 students watched all the videos, 1,380 students completed at least one assignment,and 778 made a post or comment in the weekly discussion sections. Of those with posts, 426 completed at least one class assignment. 638 students completed the online course and received a certificate (meaning that some students could earn a certificate without participating in forums at all).

In addition to the weekly assignments the students were sent a survey that was designed to assess their personal motivations for enrolling in the course. This survey consisted of 3 sets of questions: MOOC-specific motivational items; two PALS (Patterns of Adaptive Learning Survey) sub-scales [21], Academic Efficacy and Mastery-Goal Orientation; and an item focused on confidence in course completion. It was distributed to students through the course's E-mail messaging system to students who enrolled in the course prior to the official start date. Data on whether participants successfully completed the course was downloaded from the same course system after the course concluded. The survey received 2,792 responses; 38% of the participants were female and 62% of the participants were male. All of the respondents were over 18 years of age.

The MOOC-specific items consisted of 10 questions drawn from previous MOOC research studies (cf. [2, 22]) asking respondents to rate their reasons for enrollment. These 10 items address traits of MOOCs as a novel online learning platform. Specifically, these 10 items included questions on both the learning content and features of MOOCs as a new platform. Two PALS Survey scales [21] measuring mastery-goal orientation and academic efficacy were used to study standard motivational constructs. PALS scales have been widely used to investigate the relation between a learning environment and a student's motivation (cf. [6, 20, 28]). Altogether ten items with five under each scale were included. The participants were asked to select a number from 1 to 5 with 1 meaning least relevant and 5 most relevant. Respondents were also asked to self-rate their confidence on a scale of 1 to 10 as to whether they could complete the course according to the pace set by the course instructor. All three groups of items were domain-general.

4. METHODS

For our analysis, we extracted a social network from the online forum associated with the course. We assigned a node to each student, instructor, or TA in the course who added to it. Nodes representing students were labeled with their final course grade out of 100 points. The Coursera forums operate as standard threaded forums. Course participants could start a new thread with an initial post, add a post to an existing thread, and add a comment or child element below an existing post. We added a directed edge from the author of each post or comment to the parent post and to all posts or comments that preceded it on the thread based upon their timestamp. We made a conscious decision to omit the textual content of the replies with the goal of isolating the impact of the structure alone.

We thus treat each reply or followup in the graph as an implicit social connection and thus a possible relationship. Such implicit social relationships have been explored in the context of recommender systems to detect strong communities of researchers [5]. This is, by design, a permissive definition that is based upon the assumption that individuals generally add to a thread after viewing the prior content within it and that individual threads can be treated as group conversations with each reply being a conscious statement for everyone who has already spoken. The resulting network forms a multigraph with each edge representing a single implicit social interaction. We removed self loops from this graph as they indicate general forum activity but not any meaningful interaction with another person. We also removed vertices with a degree of 0, and collapsed the parallel edges to form a simple weighted graph for analysis.

In the analyses below we will focus on isolating student performance and assessing the impact of the faculty and hub students. We will therefore consider four classes of graphs: *ALL* the complete graph; *Student* the graph with the instructor and TAs removed; *NoHub* the graph with the instructor and hub users removed; and *Survey* which includes only students who completed the motivation survey. We will also consider versions of the above graphs without students who obtained a score of 0, and without the isolated individuals who connect with at most one other person. As we will discuss below, a number of students received a zero grade in the course. Because this is an at-will course, however, we cannot readily determine why these scores were obtained. They may reflect a lack of engagement with the course, differential motivations for taking the course, a desire to see the course materials without assignments, or genuinely poor performance.

4.1 Best-Friend Regression & Assortativity

Fire et al. [11] applied a similar social network approach to traditional classrooms and found a correlation between a student's most highly connected neighbor ("best friend") and the student's grade. The links in that graph included cooperation on assignments as well as partnership on group assignments. To examine whether the same correlation existed in a massive online course in which students were less likely to know each other beforehand and there were no group assignments, we calculated each student's best friend in the same manner and performed a similar correlation.

The simple best friends analysis gives a straightforward mechanism for correlating individual students. However it is also worthwhile to ask about students who are one-step removed from their peers. Therefore we will also calculate the grade assortativity (r_G) of the graphs. Assortativity describes the correlation of values between vertices and their neighbors [23]. The assortativity metric r ranges between -1 and 1, and is essentially the Pearson correlation between vertex and their neighbors [23]. A network with $r = 1$ would have each vertex only sharing edges with vertices of the same score. Likewise, if $r = -1$ vertices in the network would only share edges with vertices of different scores. Thus grade assortativity allows us to measure whether individuals are not just connected directly to individuals with similar scores but whether they correlate with individuals who are one step removed.

Several commonly studied classes of networks tend to have patterns in their assortativity. Social networks tend to have high assortativity, while biological and technological networks tend to have negative values (dissortativity) [23]. In a homogeneous course or one where students only form stratified communities we would expect the assortativity to be very high while in a heterogeneous class with no distinct communities we would expect it to be quite low.

4.2 Community Detection

The process of community detection we employed is briefly described here [3]. As noted there we elected to ignore the edge direction when making our graph. Our goal in doing so was to focus on communities of learners who shared the same threads, even when they were not directly replying to one-another. We believe this to be a reasonable assumption given the role of class forums as a knowledge-building environment in which students exchange information with the group. Individuals who participate in a thread generally review prior posts before submitting their contribution and are likely to return to view the followups. Homogeneity in this context would mean that students gathered and communicated primarily with equally-performing peers and thus that they did not consistently draw from better-performing classmates and help lower-performing ones *or* that the at-will communities served to homogenize performance, with the students in a given cluster evening out over time.

While algorithms such as GN are useful for finding clusters they do not, in and of themselves, determine the *right* number of communities. Rather, when given a target number they will seek to identify the *best* possible set of communities. In some implementations the algorithm can be applied to iteratively select the maximum modularity value over a possible range. Determining the correct number of communities to detect, however, is a non-trivial task especially in large and densely connected graphs where changes to smaller communities will have comparatively small effects on the global modularity score. As a consequence we cannot simply optimize for the best modularity score as we would risk missing small but important communities [12].

Therefore, rather than select the clusterings based solely on the highest modularity, we have opted to estimate the correct number of clusters visually. To that end we plotted a series of modularity curves over the set of graphs. For each graph G we applied the GN algorithm iteratively to produce all clusters in the range $(2,|G_N|)$. For each clustering, we then calculated the global modularity score. We examined the resulting scores to identify a *crest* where the modularity gain leveled off or began to decrease thus indicating that future subdivisions added no meaningful information or created schisms in existing high-quality communities. This is a necessarily heuristic process that is similar to the use of Scree plots in Exploratory Factor Analysis [4]. We define the number identified as the *natural* cluster number.

5. RESULTS AND DISCUSSION

Before removing self-loops and collapsing the edges, the network contained 754 nodes and 49,896 edges. The final social network contained 754 nodes and 17,004 edges. 751 of the participants were students, with 1 instructor and 2 TAs. One individual was incorrectly labeled as a student when they were acting as the Chief Community TA. Since this person's posts clearly indicated that he or she was acting in a TA capacity with regard to the forums, we relabeled him/her as a TA. Of the 751 students 304 obtained a zero grade in the course leaving 447 nonzero students. 215 of the 751 students responded to the motivation survey.

There were a total of 55,179 registered users, so the set of 754 forum participants is a small fraction of the entire course audience. However, forum users are not necessarily those who will make an effort or succeed in the course. Forum users did not all participate in the course, and some students who participated in the course did not use the forums: 1,381 students in the course got a grade greater than 0, and 934 of those did not post or comment on the forums, while 304 of the 751 students who did participate in the forums received a grade of 0. Clearly students who go to the trouble of posting forum content are in some respect making an effort in the course beyond those who don't, but this does not necessarily correspond to course success.

5.1 Best-Friend Regression & Assortativity

We followed Fire et al.'s methodology for identifying Best Friends in a weighted graph and calculated a simple linear regression over the pairs. This correlation did not include the instructor or TAs in the analysis. We calculated the correlation between the students' grades to their best friends' grades in the set using Spearman's Rank Correlation Coefficient (ρ) [34]. The two variables were strongly correlated, $\rho(748) = 0.44$, $p < 0.001$. However, the correlation was also affected by the dense clusters of students with 0 grades. After removing the 0 grade students we found an additional moderate correlation, $\rho(444) = 0.29$, $p < 0.001$.

Thus the significant correlation between best-friend grade and grade holds over the transition from the traditional classroom to a MOOC. This suggests that students in a MOOC, excluding the many who drop out or do not submit assignments, behave similarly to those in a traditional classroom in this respect. These results are also consistent with our calculations for assortativity. There we found a small assortative trend for the grades as shown in Table 1. These values reflect that a student was frequently communicating with students who in turn communicated with students at a similar performance level. This in turn supports our belief that homogeneous communities may be found. As Table 1 also illustrates, the zero-score students contribute substantially to the assortativity correlation as well with the correlation dropping by as much as a third when they were removed.

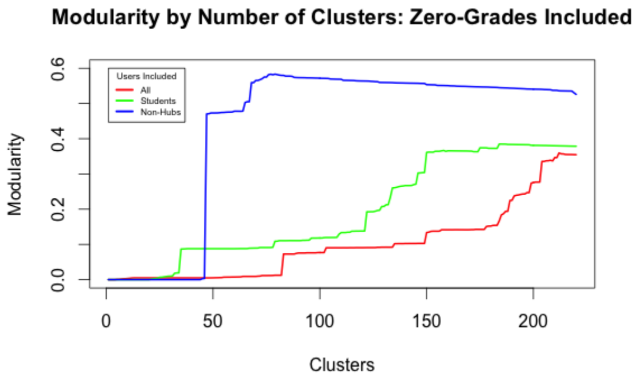


Figure 1: Modularity for each number of clusters, including students with zeros.

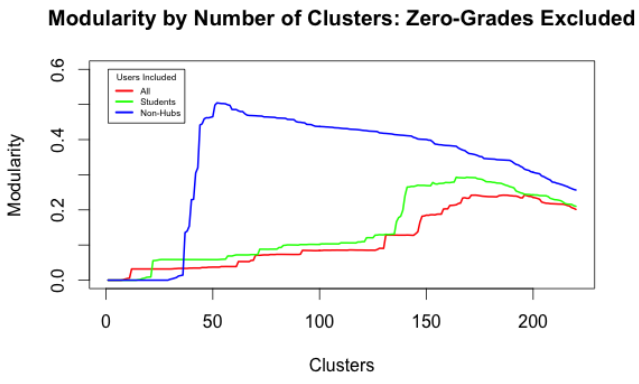


Figure 2: Modularity for each number of clusters, excluding students with zeros.

5.2 Community Structure

The modularity curves for the graphs both with and without zero-score students are shown in Figures 1 and 2. We examined these plots to select the natural cluster numbers which are shown in Table 2. As the values illustrate the instructor, TAs, and hub students have a disproportionate impact on the graph structure. The largest hub student in our graph connects to 444 out of 447 students in the network. The graph with all users had lower modularity and required more clusters than the graphs with only students or only non-hubs (see Table 2), with

Table 1: The grade assortativity for each network.

Users	Zeros	V	E	r_G
All	Yes	754	17004	0.29
All	No	447	5678	0.20
Students	Yes	751	15989	0.32
Students	No	447	5678	0.20
Non-Hub	Yes	716	9441	0.37
Non-Hub	No	422	3119	0.24

Table 2: Graph sizes and natural number of clusters for each graph.

Users	Zeros	V	E	Clusters
All	Yes	754	17004	212
All	No	447	5678	173
Students	Yes	751	15989	184
Students	No	447	5678	169
Non-Hub	Yes	716	9441	79
Non-Hub	No	422	3119	52
Survey	Yes	215	1679	58

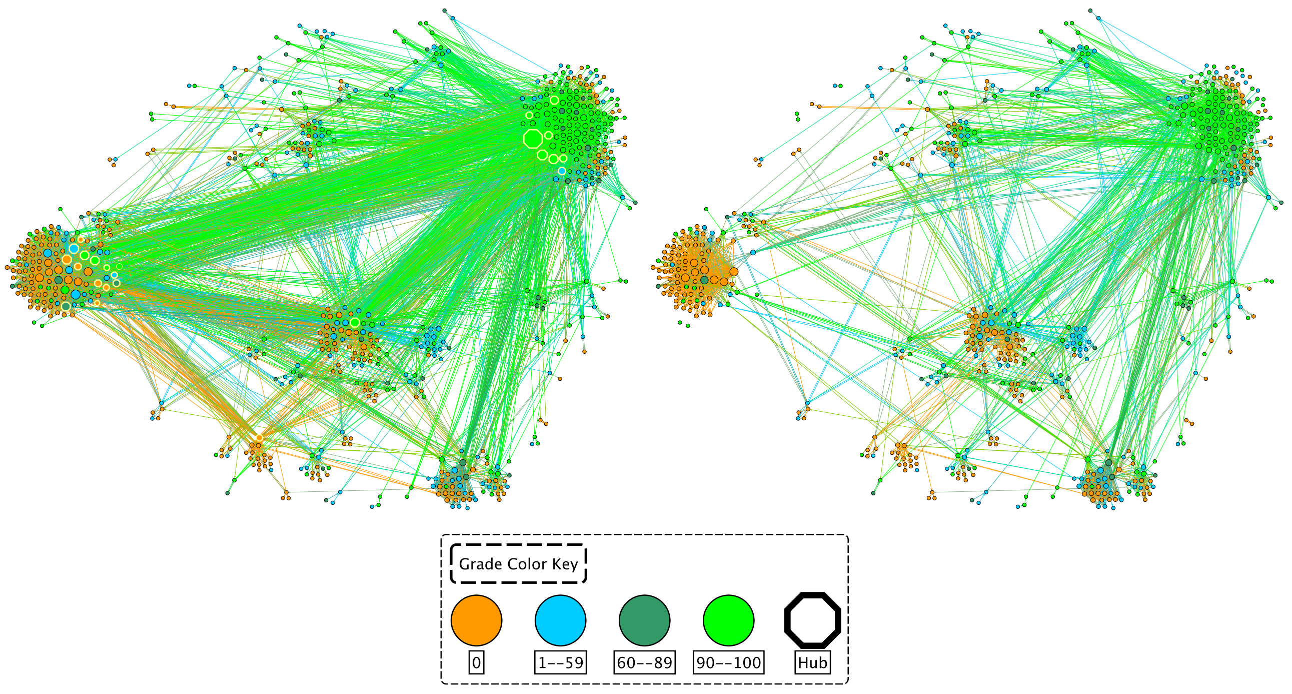


Figure 3: View of the student communities with edges of frequency <2 removed. The Student network with (left) and without (right) hub-students, with each vertex representing a student and grade represented as color.

the non-hub graph having the highest modularity. This suggests that non-hub students formed more isolated communities, while teaching staff and hubs communicated across these communities and connected them.

This largely consistent with the intent of the forums and the active role played by the instructor and TAs in monitoring and replying to all relevant posts in the forums. It is particularly interesting how closely the curves for the ALL and Student graphs mirror one another. This may indicate that the hub students are also those that followed the instructor and TAs closely, thus giving them isomorphic relationships, or it may indicate that they are more connected than even the instructors and thus came to bind the forums together on their own. This impact is further illustrated by the cluster plots shown in Figure 3. Here the absence of the hub students results in a noticeable thinning of the graph which in turn highlights the frequency of communication that can be attributed to this, comparatively small, group.

The difference between the full plots and those with zero values are also notable as the zero grade students were clearly a major factor in community formation. A direct examination of the user graph showed that many of the zero students were only connected to other zero students or were not connected at all. This is also highlighted in Figure 3. In both graphs the bulk of the zero score students are clustered in a tight network of communities on the left-hand side. That super-community consists primarily of zero score students communicating with other zero-score students, a structure we have nick-named the 'deathball.'

5.3 Student Performance & Motivation

As the color coding in Figure 3 illustrates, the students did cluster by performance. Table 3 shows the average grade and standard deviation for a small selection of the communities in the ALL reply network including zero-grades, hub students, and teaching staff. Several of the communities, particularly the larger ones, do show a blend of good and poor students, with a high standard deviation. However many if not most of the communities are more homogeneous with good and poor students sharing a community with similarly-performing peers. These clusters have markedly lower standard deviation.

Table 3: Grade statistics by community, selected to show examples of more and less homogeneous communities.

Members	Average Grade	Standard Deviation
118	21.62	36.58
41	22.00	32.45
34	25.41	40.44
31	56.13	47.69
20	49.05	45.64
16	12.44	31.13
14	88.43	22.47
12	96.08	6.36
11	96.45	7.38
4	3.00	6.00
4	8.50	9.81
4	4.25	8.50
4	96.25	3.50

An examination of the grade distribution for each of the clusters showed that the scores within each cluster were non-normal. Therefore we opted to apply the Kruskal-Wallis (KW) test to assess the correlation between cluster membership and perfor-

Table 4: Kruskal-Wallis test of student grade by community, for each graph.

Users	Zeros	Chi-Squared	df	p-value
All	Yes	349.0273	211	< 0.005
All	No	216.1534	172	< 0.02
Students	Yes	202.0814	78	< 0.005
Students	No	80.93076	51	< 0.005
Non-Hub	Yes	309.8525	183	< 0.005
Non-Hub	No	218.9603	168	< 0.01
Survey	Yes	99.99840	577	< 0.005

mance. The KW test is a nonparametric rank-based analogue to the common Analysis of Variance [17]. Here we tested grade by community number with the community being treated as a categorical variable. The results of this comparison are shown in Table 4. As that illustrates, cluster membership was a significant predictor of student performance for all of the graphs with the non-zero graphs having markedly lower p-values than those with zero students included. These results are consistent with our hypothesis that students would form clusters of equal-performers and we find that those results hold even when the highly-connected instructors, TAs and hub students are included.

We performed a similar KW analysis for the questions on the motivation survey and for a binary variable indicating whether or not the student completed the survey at all. For this analysis we evaluated the clusters on all of the graphs. We found no significant relationship between the community structure on any of the graphs and the survey question results or the survey completion variable. Thus while the clusters may be driven by separate factors they are not reflected in the survey content.

6. CONCLUSIONS AND FUTURE WORK

Our goal in this paper was to expand upon our prior community detection work with the goal of aligning that work with prior research on peer impacts, notably the work of Fire et al. [11]. We also sought to examine the impact of hub students and student motivations on our prior results.

To that end we performed a novel community clustering analysis of student performance data and forum communications taken from a single well-structured MOOC. As part of this analysis we described a novel heuristic method for selecting natural numbers of clusters, and replicated the results of prior studies of both immediate neighbors and second-order assortativity.

Consistent with prior work, we found that students' grades were significantly correlated with their most closely associated peers in the new networks. We also found that this correlation extended out to their second-order neighborhood. This is consistent with our prior work showing that students form stable user communities that are homogeneous by performance. We found that those results were stable even if instructors, hub players, students with 0 scores, and students who did not fill out the survey were removed from consideration. This suggests that either the students are forming communities that are homogeneous or that the effect of those individual and network features on the communities and on performance is minimal.

We also found that community membership was not a significant predictor of whether students would complete the motivation survey or of students' motivations. We were surprised by the fact that even when we focused solely on individuals who had completed the survey, the students did not connect by stated goals. This suggests to us that the students are more likely coalescing around the pragmatic needs of the class or conceptual challenges rather than on the winding paths that brought them there. One limitation of this work is that by relying on the forum data we were focused solely on the comparatively small proportion of enrolled students (6%) who actively participated in the forums. This group is, by definition a smaller set of more actively-involved participants.

In addition to addressing our primary questions this study also raised a number of open issues for further exploration. Firstly, this work focused solely on the final course structure, grades, and motivations. We have not yet addressed whether these communities are stable over time or how they might change as students drop in our out. Secondly, while we ruled out motivations as a basis for the community this work we were not able to identify what mechanisms do support the communities. And finally this study raises the question of generality and whether or not these results can be applied to MOOCs offered on different topics or whether the results apply to traditional and blended courses.

In subsequent studies we plan to examine both the evolution of the networks over time as well as additional demographic data with the goal of assessing both the stability of these networks and the role of other potential latent factors. We will also examine other potential clustering mechanisms that control for other user features such as frequency of involvement and thread structure. We also plan to examine other similar datasets to determine if these features transition across classes and class types. We believe that these results may change somewhat once students can coordinate face to face far more easily than online.

7. ACKNOWLEDGMENTS

This work was supported by NSF grant #1418269: "Modeling Social Interaction & Performance in STEM Learning" Yoav Bergner, Ryan Baker, Danielle S. McNamara, & Tiffany Barnes Co-PIs.

8. REFERENCES

[1] R. Azen and D. Budescu. The dominance analysis approach for comparing predictors in multiple regression. *Psychological Methods*, 8(2):129–48, 2003.

[2] Y. Belanger and J. Thornton. Bioelectricity: A quantitative approach Duke University's first MOOC. *Journal of Learning Analytics*, 2013.

[3] R. Brown, C. F. Lynch, M. Eagle, J. Albert, T. Barnes, R. Baker, Y. Bergner, and D. McNamara. Good communities and bad communities: Does membership affect performance? In C. Romero and M. Pechenizkiy, editors, *Proceedings of the 8^{th} International Conference on Educational Data Mining*, 2015. submitted.

[4] R. B. Cattell. The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276, 1966.

[5] E. Choo, T. Yu, M. Chi, and Y. Sun. Revealing and incorporating implicit communities to improve recommender systems. In M. Babaioff, V. Conitzer, and D. Easley, editors, *ACM Conference*

on Economics and Computation, EC '14, Stanford, CA, USA, June 8-12, 2014, pages 489–506. ACM, 2014.

[6] K. Clayton, F. Blumberg, and D. P. Auld. The relationship between motivation learning strategies and choice of environment whether traditional or including an online component. *British Journal of Educational Technology*, 41(3):349–364, 2010.

[7] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006.

[8] S. Dawson. 'Seeing' the learning community: An exploration of the development of a resource for monitoring online student networking. *British Journal of Educational Technology*, 41(5):736–752, 2010.

[9] J. Eckles and E. Stradley. A social network analysis of student retention using archival data. *Social Psychology of Education*, 15(2):165–180, 2012.

[10] A. Fini. The technological dimension of a massive open online course: The case of the CCK08 course tools. *The International Review Of Research In Open And Distance Learning*, 10(5), 2009.

[11] M. Fire, G. Katz, Y. Elovici, B. Shapira, and L. Rokach. Predicting student exam's scores by analyzing social network data. In *Active Media Technology*, pages 584–595. Springer, 2012.

[12] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proc. of the National Academy of Sciences*, 104(1):36–41, 2007.

[13] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. of the National Academy of Sciences*, 99(12):7821–7826, June 2002.

[14] J. Huang, A. Dasgupta, A. Ghosh, J. Manning, and M. Sanders. Superposter behavior in MOOC forums. In *Proc. of the first ACM conference on Learning@ scale conference*, pages 117–126. ACM, 2014.

[15] V. Kovanovic, S. Joksimovic, D. Gasevic, and M. Hatala. What is the source of social capital? the association between social network position and social presence in communities of inquiry. In S. G. Santos and O. C. Santos, editors, *Proc. of the Workshops held at Educational Data Mining 2014, co-located with 7th International Conference on Educational Data Mining (EDM 2014), London, United Kingdom, July 4-7, 2014.*, volume 1183 of *CEUR Workshop Proc.* CEUR-WS.org, 2014.

[16] A. D. I. Kramer, J. E. Guillory, and J. T. Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proc. of the National Academy of Sciences*, 111(24):8788–8790, 2014.

[17] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American statistical Association*, 47(260):583–621, 1952.

[18] N. Li, H. Verma, A. Skevi, G. Zufferey, J. Blom, and P. Dillenbourg. Watching MOOCs together: investigating co-located MOOC study groups. *Distance Education*, 35(2):217–233, 2014.

[19] T. R. Liyanagunawardena, A. A. Adams, and S. A. Williams. MOOCs: A systematic study of the published literature 2008-2012. *The International Review of Research in Open and Distributed Learning*, 14(3):202–227, 2013.

[20] J. L. Meece, E. M. Anderman, and L. H. Anderman. Classroom goal structure, student motivation, and academic achievement. *Annual Review of Psychology*, 57:487–503, 2006.

[21] C. Midgley, M. L. Maehr, L. Hruda, E. Anderinan, L. Anderman, and K. E. Freeman. *Manual for the Patterns of Adaptive Learning Scales (PALS).* University of Michigan, Ann Arbor, 2000.

[22] MOOC @ Edinburgh 2013. MOOC @ Edinburgh 2013 - report #1. *Journal of Learning Analytics*, 2013.

[23] M. E. Newman. Assortative Mixing in Networks. *Physical Review Letters*, 89(20):208701, Oct. 2002.

[24] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2012.

[25] B. Rienties, P. Alcott, and D. Jindal-Snape. To let students self-select or not: That is the question for teachers of culturally diverse groups. *Journal of Studies in International Education*, 18(1):64–83, 2014.

[26] T. Rizzuto, J. LeDoux, and J. Hatala. It's not just what you know, it's who you know: Testing a model of the relative importance of social networks to academic performance. *Social Psychology of Education*, 12(2):175–189, 2009.

[27] C. P. Rosé, R. Carlson, D. Yang, M. Wen, L. Resnick, P. Goldman, and J. Sherer. Social factors that contribute to attrition in MOOCs. In *Proc. of the first ACM conference on Learning@ scale conference*, pages 197–198. ACM, 2014.

[28] A. M. Ryan and H. Patrick. The classroom social environment and changes in adolescents' motivation and engagement during middle school. *American Educational Research Journal*, 38(2):437–460, 2001.

[29] D. Seaton, Y. Bergner, I. Chuang, P. Mitros, and D. Pritchard. Who does what in a massive open online course? *Communications of the ACM*, 57(4):58–65, 2014.

[30] G. Stahl, T. Koschmann, and D. Suthers. Computer-supported collaborative learning: An historical perspective. *Cambridge handbook of the learning sciences*, 2006:409–426, 2006.

[31] L. Van Dijk, G. Van Der Berg, and H. Van Keulen. Interactive lectures in engineering education. *European Journal of Engineering Education*, 26(1):15–28, 2001.

[32] Y. Wang and R. Baker. Content or platform: Why do students complete MOOCs? *MERLOT Journal of Online Learning and Teaching*, 11(1):191–218, 2015.

[33] Y. Wang, L. Paquette, and R. Baker. A longitudinal study on learner career advancement in MOOCs. *Journal of Learning Analytics*, 1(3), 2014.

[34] Wikipedia. Spearman's rank correlation coefficient — Wikipedia, the free encyclopedia, 2013. [Online; accessed 27-February-2013].

[35] Wikipedia. Modularity (networks) — Wikipedia, the free encyclopedia, 2014. [Online; accessed 5-February-2015].

[36] D. Yang, T. Sinha, D. Adamson, and C. P. Rose. Turn on, tune in, drop out: Anticipating student dropouts in massive open online courses. In *Proc. of the 2013 NIPS Data-Driven Education Workshop*, volume 10, page 13, 2013.

[37] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[38] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *Proc. of the 16th international conference on World Wide Web*, pages 221–230. ACM, 2007.

Using the Hint Factory to Compare Model-Based Tutoring Systems

Collin Lynch
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
cflynch@ncsu.edu

Thomas W. Price
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
twprice@ncsu.edu

Min Chi
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
mchi@ncsu.edu

Tiffany Barnes
North Carolina State University
890 Oval Drive
Raleigh, NC 27695
tmbarnes@ncsu.edu

ABSTRACT

Model-based tutoring systems are driven by an abstract domain model and solver that is used for solution validation and student guidance. Such models are robust but costly to produce and are not always adaptive to specific students' needs. Data-driven methods such as the Hint Factory are comparatively cheaper and can be used to generate individualized hints without a complete domain model. In this paper we explore the application of data-driven hint analysis of the type used in the Hint Factory to existing model-based systems. We present an analysis of two probability tutors Andes and Pyrenees. The former allows for flexible problem-solving while the latter scaffolds students' solution path. We argue that the state-space analysis can be used to better understand students' problem-solving strategies and can be used to highlight the impact of different design decisions. We also demonstrate the potential for data-driven hint generation across systems.

1. INTRODUCTION

Developers of model-based tutoring systems draw on domain experts to develop ideal models for student guidance. Studies of such systems have traditionally been focused on their overall impact on students' performance and not on the students' user-system interaction. The Hint Factory, by contrast, takes a data-driven approach to extract advice based upon students' problem solving paths. In this paper we will apply the Hint Factory analytically to evaluate the impact of user interface changes and solution constraints between two closely-related tutoring systems for probability.

Model-based tutoring systems are based upon classical expert systems, which represent relevant domain knowledge via static rule bases or sets of constraints [9]. These knowledge bases are generally designed by domain experts or with their active involvement. They are then paired with classical search algorithms or heuristic satisfaction algorithms to automatically solve domain problems, identify errors in student solutions, and to provide pedagogical guidance. The goal of the design process is to produce expert models that give the same procedural advice as a human expert. Classical model-based tutors have been quite successful in field trials, with systems such as the ACT Programming Tutor helping students achieve almost two standard deviations higher than those receiving conventional instruction [5].

Data-driven hint generation methods such as those used in Hint Factory [17] take a different approach. Rather than using a strong domain model to generate *a-priori* advice, data-driven systems examine prior student solution attempts to identify likely paths and common errors. This prior data can then be used to provide guidance by directing students towards successful paths and away from likely pitfalls. In contrast to the expert systems approach, these models are primarily guided not by what experts consider to be *ideal* but by what students *do*.

Model-based systems such as Andes [18] are advantageous as they can provide appropriate procedural guidance to students *at any point* in the process. Such models can also be designed to reinforce key meta-cognitive concepts and explicit solution strategies [4]. They can also scale up rapidly to include new problems or even new domain concepts which can be incorporated into the existing system and will be available to all future users. Rich domain models, however, are comparatively expensive to construct and require the long-term involvement of domain experts to design and evaluate them.

Data-driven methods for generating feedback, by contrast, require much lower initial investment and can readily adapt to individual student behaviors. Systems such as the Hint Factory are designed to extract solutions from prior student data, to evaluate the quality of those solutions, and to compile solution-specific hints [17]. While this avoids the need for a strong domain model, it is limited to the space of solutions explored by prior students. In order to incorporate new problems or concepts it is necessary to collect additional data. Additionally, such methods are not generally designed to incorporate or reinforce higher-level solution strategies.

We believe that both of these approaches have inherent advantages and are not necessarily mutually exclusive. Our goal in this paper is to explore what potential data-driven methods have to inform and augment model-based systems. We argue that data-driven methods can be used to: (1) evaluate the differences between closely-related systems; (2) assess the impact of specific design decisions made in those systems for user behaviors; and (3) evaluate the potential application of data-driven hint generation across systems. To that end we will survey relevant prior work on model-based and data-driven tutoring. We will describe two closely-related tutoring systems and data collected from them. We will then present a series of analyses using state-based methods and discuss the conclusions that we drew from them.

2. BACKGROUND

2.1 Model-Based Tutoring

Model-based tutoring systems take a classical expert-systems approach to tutoring. They are typically based upon a strong domain model composed of declarative rules and facts representing domain principles and problem-solving actions coupled with an automatic problem solver. This knowledge base is used to structure domain knowledge, define individual problems, evaluate candidate solutions, and to provide student guidance. Novices typically interact with the system through problem solving with the system providing solution validation, automatic feedback, pedagogical guidance, and additional problem-solving tasks. The Sherlock 2 system, for example, was designed to teach avionics technicians about appropriate diagnostic procedures [11]. The system relies on a domain model that represents the avionics devices being tested, the behavior of the test equipment, and rules about expert diagnostic methods. Sherlock 2 uses these models to pose dynamic challenges to problem solvers, to simulate responses to their actions, and to provide solution guidance.

Andes [19, 18, 20] and Pyrenees [4] are closely-related model-driven ITSs in the domains of physics and probability. They were originally developed at the University of Pittsburgh under the Direction of Dr. Kurt VanLehn. Like other model-based systems, they rely on a rule-based domain model and automatic problem solvers that treat the domain rules as problem-solving steps. They distinguish between higher-level domain concepts such as Bayes' Rule, and atomic steps such as variable definitions. Principles are defined by a central equation (e.g. $p(A|B) = (p(B|A) * p(A))/p(B)$) and encapsulate a set of atomic problem-solving steps such as writing the equation and defining the variables within it.

The systems are designed to function as homework-helpers, with students logging into the system and being assigned or selecting one of a set of predefined problems. Each problem

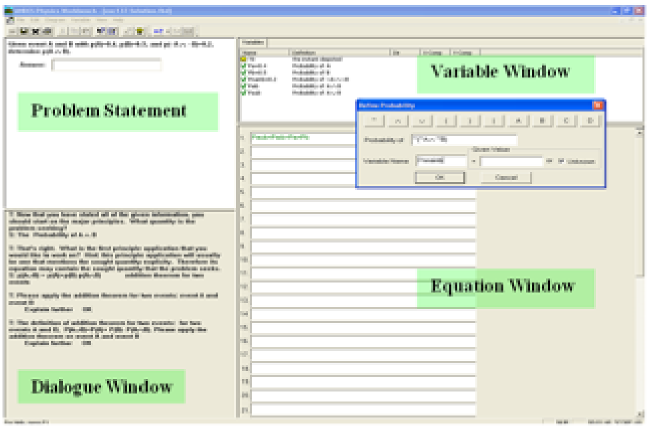
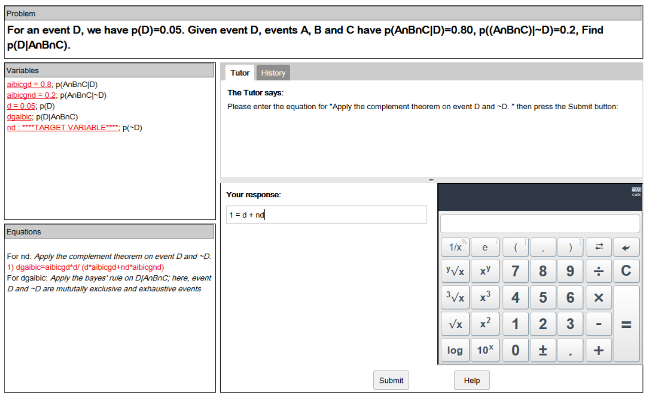


Figure 1: The Andes user interface showing the problem statement window with workspace on the upper left hand side, the variable and equation windows on the right hand side, and the dialogue window on the lower left.

is associated with a pre-compiled solution graph that defines the set of possible solutions and problem-solving steps. The system uses a principle-driven automated problem solver to compile these graphs and to identify the complete solution paths. The solver is designed to implement the Target Variable Strategy (TVS), a backward-chaining problem solving strategy that proceeds from a goal variable (in this case the answer to the problem) via principle applications to the given information. The TVS was designed with the help of domain experts and guides solvers to define basic solution information (e.g. given variables) and then to proceed from the goal variable and use principles to define it in terms of the given variables.

Students working with Andes use a multi-modal user interface to write equations, define variables and engage in other atomic problem-solving steps. A screenshot of the Andes UI can be seen in Figure 1. Andes allows students to solve problems flexibly, completing steps in any order so long as they are valid [20]. A step is considered to be valid if it matches one or more entries in the saved solution paths and all necessary prerequisites have been completed. Invalid steps are marked in red, but no other immediate feedback is given. Andes does not force students to delete or fix incorrect entries as they do not affect the solution process. In addition to validating entries, the Andes system also uses the precompiled solution graphs to provide procedural guidance (*next-step-help*). When students request help, the system will map their work to the saved solution paths. It will then select the most complete solution and prompt them to work on the next available step.

One of the original goals of the Andes system was to develop a tutor that operated as an "intelligent worksheet." The system was designed to give students the freedom to solve problems in any order and to apply their preferred solution strategy. The system extends this freedom by allowing invalid steps in an otherwise valid solution and by allowing students to make additional correct steps that do not advance the solution state or are drawn from multiple


Problem

For an event D, we have p(D)=0.05. Given event D, events A, B and C have p(AnBnC|D)=0.80, p((AnBnC)|~D)=0.2, Find p(D|AnBnC).

Variables

aibicgd = 0.8; p(AnBnC|D)
aibicgnd = 0.2; p(AnBnC|~D)
d = 0.05; p(D)
dgaibic; p(D|AnBnC)
nd : ****TARGET VARIABLE****; p(~D)

Equations

For nd: *Apply the complement theorem on event D and ~D.*
1) dgaibic=aibicgd*d/ (d*aibicgd+nd*aibicgnd)
For dgaibic: *Apply the bayes' rule on D|AnBnC; here, event D and ~D are mututally exclusive and exhaustive events*

The Tutor says:

Please enter the equation for "Apply the complement theorem on event D and ~D. " then press the Submit button:

Your response:

1 = d + nd

Figure 2: The Pyrenees user interface showing the problem statement at the top, the variable and equation lists on the left, and the tutor interaction window with calculator on the lower right.

solution paths. This was motivated in part by a desire to make the system work in many different educational contexts where instructors have their own preferred methods [20]. The designers of Andes also consciously chose only to provide advice upon demand when the students would be most willing to accept it. For the students however, particularly those with poor problem-solving skills, this passive guidance and comparative freedom can be problematic as it does not force them to adhere to a strategy.

This problem motivated the development of Pyrenees. Pyrenees, like Andes acts as a homework helper and supports students with on-demand procedural and remediation help. It uses an isomorphic domain model with the same principles, basic steps, problems, and solution paths. Unlike Andes, however, Pyrenees forces students to applying the target-variable-strategy during problem solving. It also requires them to repair incorrect entries immediately before moving on. Students are guided through the solution process with a menu-driven interface, shown in Figure 2. At each step, the system asks students what they want to work on next and permits them to make any valid step that is consistent with the TVS. Chi and VanLehn [3] conducted a study of the two systems and found that scaffolding the TVS in Pyrenees helped to eliminate the gap between high and low learners. This effect was observed both in the original domain where it was taught (in their case probability) and it transferred to a new domain (physics), where students used Andes alone.

2.2 Data-Extraction and Data-Driven Tutoring.

One of the longstanding goals of educational data-miners is to support the development of data-driven tutoring systems. Such systems use past student data to structure pedagogical and domain knowledge, administer conceptual and pedagogical advice, or evaluate student performance and needs. A number of attempts have been made to address these goals. One of the most successful data-driven systems is the Hint Factory [1, 2, 17]. The Hint Factory takes an MDP-based approach to hint generation. It takes as input a set of prior student logs for a given problem, represented as a network of interactions [6, 7]. Each vertex in this network represents the state of a student's partial solution at some point during the problem solving process, and each edge represents an action that takes the student from one state to another. A complete solution is represented as a path from the initial state to a goal state. Each state in the interaction network is assigned a weight via a value-iteration algorithm. A new student requesting a hint is matched to a previously observed state and given context-sensitive advice. If, for example, the student is working on a problem that requires Bayes' Rule and has already defined $p(A)$, $p(B)$, and $p(B|A)$ then the Hint Factory would first prompt them to consider defining $p(A|B)$, then it would point them to Bayes Rule, before finally showing them the equation $p(A|B) = (p(B|A) * p(A))/p(B)$.

These hints are incorporated into existing tutoring systems in the form of a lookup table that provides state-specific advice. When a user asks for help the tutor will match their current state to an index state in the lookup table and will prompt them to take the action that will lead them to the highest value neighboring state. If their current state is not found then the tutor will look for a known prior state or will give up. The Hint Factory has been applied successfully in a number of domains including logic proofs [17], data structures [8], and programming [15, 10, 13]. Researchers have also explored other related methods for providing data-driven hints. These include alternative state representations [13], path construction algorithms [16, 14], and example-based model-construction [12].

The primary goal of the Hint Factory is to leverage prior data to provide optimal state-specific advice. By calculating advice on a per-state basis, the system is able to adapt to students' specific needs by taking into account both their current state and the paths that they can take to reach the goal. As a consequence the authors of the Hint Factory argue that this advice is more likely to be in the students' Zone of Proximal Development and thus more responsive to their needs than a less-sensitive algorithm.

3. METHODS

In order to investigate the application of data-driven methods to model-based tutoring systems, we collected data from two studies conducted with Andes and Pyrenees in the domain of probability. We then transformed these datasets into interaction networks, consisting of states linked with actions. We used this representation to perform a variety of quantitative and qualitative analyses with the goal of evaluating the differences between the two systems and the impact of the specific design decisions that were made in each.

3.1 The Andes and Pyrenees Datasets

The Andes dataset was drawn from an experiment conducted at the University of Pittsburgh [3]. This study was designed to assess the differential impact of instruction in Andes and Pyrenees on students' meta-cognitive and problem-solving skills. Participants in this study were college undergraduates who were required to have taken high-school level algebra and physics but not to have taken a course in probability or statistics. The participants were volunteers and were paid by time not performance.

Forty-four students completed the entire study. However for the purposes of the present analysis, we drew on all

66 students who completed at least one problem in Andes-Probability. This is consistent with prior uses of the Hint Factory which draw from all students including those who did not complete the problem. The Pyrenees-Probability logs from this study were not used due to problems with the data format that prevented us from completing our analysis. From this dataset we drew 394 problem attempts covering 11 problems. The average number of steps required to solve the problems was 17.6. For each problem we analyzed between 25 and 72 problem attempts, with an average of 35.8 attempts per problem. Some attempts were from the same student, with at most two successful attempts per student. Over all problems, 81.7% of the attempts were successful, with the remainder being incomplete attempts.

The Pyrenees dataset was drawn from a study of 137 students conducted in the 200-level Discrete Mathematics course in the Department of Computer Science at North Carolina State University. This study used the same probability textbook and pre-training materials as those used in the Andes study. The students used Pyrenees as part of a homework assignment, in which they completed 12 problems using the tutoring system. One of these problems was not represented in the Andes dataset. We therefore excluded it from our analysis, leaving 11 shared problems.

Unlike the Andes students, however, the Pyrenees students were not always required to solve every problem. In this study the system was configured to randomly select some problems or problem steps to present as worked examples rather than as steps to be completed. In order to ensure that the results were equivalent we excluded the problem-level worked examples and any attempt with a step-level worked example from our analysis. As a consequence, each problem included a different subset of these students. For each problem we analyzed between 83 and 102 problem attempts, with an average of 90.8 attempts per problem. Some attempts were from the same student, with at most one successful attempt per student. Over all problems, 83.4% of the attempts were successful.

3.2 State and Action Representations

In order to compare the data from both tutors, we represented each problem as an interaction network, a representation used originally in the Hint Factory [7]. In the network a vertex, or state, represents the sum total of a students' current problem solving steps at a given time during a problem-solving attempt. Because Andes permits flexible step ordering while Pyrenees does not, we chose to represent the problem solving state s_t as the set of valid variables and equations defined by the student at time t.

A variable is a probabilistic expression, such as $P(A \cup B)$, that the student has identified as important to solving the problem, for which the probability is known or sought. An equation represents the application of a principle of probability, which relates the values of defined variables, such as the Complement Theorem, $P(A) + P(\neg A) = 1$. Because such equations can be written in many algebraically equivalent ways, we represent each equation as a 2-tuple, consisting of the set of variables included in the equation (e.g. $\{P(A), P(\neg A)\}$) and the principle being applied (e.g. Complement Theorem). Because we only represent valid equations, this representation uniquely identifies any equation for a given problem. Because we used the same state representation for both tutors, we were able to compare states directly across tutors.

Additionally, we opted to ignore incorrect entries. Pyrenees prevents students from applying the principles of probability improperly and forces them to correct any mistakes made immediately therefore any errors in the student logs are immediately removed making the paths uninformative. Andes, by contrast, gives students free reign when writing equations and making other entries. This freedom resulted in syntactic errors and improper rule application errors arising in our dataset. The meaning of these invalid equations is inherently ambiguous and therefore difficult to incorporate into a state definition. However such errors are immediately flagged by the system and may be ignored by the student without consequence as they do not affect the answer validity therefore they may be safely ignored as well.

An edge, or action, in our network represents the correct application of a rule or a correct variable definition and leads to a transition from one state to another. For the present dataset and state representation, the possible actions were the definition or deletion of variables or equations. Each of these actions was possible in both tutors.

4. ANALYSIS

In order to develop a broader understanding of our datasets, we first visualized the interaction network for each problem as a weighted, directed graph. We included attempts from both Andes and Pyrenees in the network, and weighted the edges and verticies by the frequency with which it appeared in the logs. We annotated each state and edge with the weight contributed by each tutor. Two examples of these graphs are given in Figure 3.

Throughout this section, we will use these graphs to address the points we outlined at the end of Section 1. We begin with a case study from one problem and will explore the student problem solving strategies using our graph representation. We will then compare the Andes and Pyrenees Systems with a variety of metrics based on this representation. We will relate our observations back to the design decisions of each system and identify evidence that may support or question these decisions. Finally, we will show how the analysis methods associated with data-driven hint generation can be used to validate some of these findings.

4.1 Case Study: Problem Ex242

The graphical representation of a problem is very helpful for giving a high-level overview of a problem and performing qualitative analysis. Problem *Ex242*, shown in Figure 3, presents an interesting scenario for a number of reasons. The problem was the 10th in a series of 12 practice problems, and asked the following:

> Events A, B and C are mutually exclusive and exhaustive events with $p(A) = 0.2$ and $p(B) = 0.3$. For an event D, we know $p(D|A) = 0.04$, $p(D|B) = 0.03$, and $p(C|D) = 0.3$. Determine $p(B|D)$.

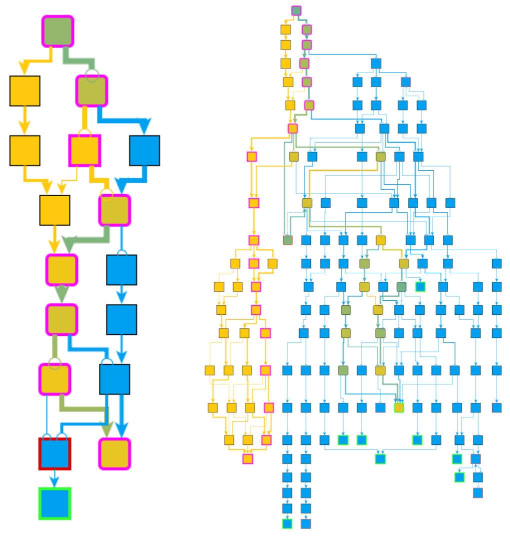


Figure 3: A graph representation of problems Ex252a, left, and Ex242, right. States and edges are colored on a gradient from blue to yellow, indicating the number of students who reached that state in the Andes and Pyrenees tutors respectively. Rounded edges indicate that at least one student from both tutors is present in a state. A green border indicates a solution state, and a pink border indicates that a state is contained in the pedagogically "ideal" solution. Edge thickness corresponds to the natural log of the number of problem attempts which included the given edge.

The problem is notable in the Pyrenees dataset because it was the only one in which the students were split almost evenly among two solution paths. For most of the problems in the dataset the vast majority of students followed the optimal solution path with only a few finding alternatives. The ideal solution path, as suggested by Pyrenees' domain model, employed repeated applications of the Conditional Probability Theorem: $P(A \cap B) = P(A|B)P(B)$, which the problem was designed to teach. The students had been previously exposed to Bayes' Theorem however, and over half of them chose to apply it instead. This allowed them to circumvent one variable definition and two applications of the Conditional Probability Theorem, achieving a slightly shorter solution path. We make no argument here which path the tutor should encourage students to take, but it is worth noting that the Hint Factory, when trained on the Pyrenees data for this problem, recommends the shorter, more popular path.

The Andes dataset gives us a very different set of insights into this problem. Because Andes lacks the strong process scaffolding of Pyrenees, students were able to make a wider variety of choices, leading to a graph with many more, less populous states. While almost every state and edge in the Pyrenees graph represents multiple students, the Andes graph contains a number of paths, including solutions, that were reached by only one student. In some state-based analyses the authors choose to omit these singleton states, for instance when generating hints. We have chosen to include them as they represent a fair proportion of the Andes data. For instance, 62 of the 126 Andes states for Ex242 were singleton states.

Interestingly, while there were small variations among their solutions, all of the Andes students choose to apply Bayes' Rule rather than relying solely on the Conditional Probability Theorem as suggested by Pyrenees. This, coupled with the strong proportion of Pyrenees students who also chose the Bayes' Rule solution, indicates that the solution offered by Pyrenees may be unintuitive for students, especially if they have recently learned Bayes' Rule. Again, this can be interpreted as evidence that Pyrenees' strong guidance did have an impact on students' problem solving strategies, but it also raises concerns about how reasonable this guidance will appear to the students. Regardless of one's interpretation, an awareness of a trend like this can help inform the evolution of model-based tutors like Andes and Pyrenees.

4.2 Comparing datasets

We now turn to qualitatively comparing the datasets. While it is not a common practice to directly compare data from different tutors, we argue that it is appropriate, especially in this context. In longstanding tutoring projects it is common for developers and researchers to make many substantive changes. The Andes system itself has undergone substantial interface changes over the course of its development [20]. These changes can alter student behavior in substantial ways, and it is important for researchers to consider how they affect not just learning outcomes but also problem solving strategies, as was investigated by Chi et al. [4].

In many respects the close relationship between Andes and Pyrenees makes them analogous to different versions of the same tutor and the presence of an isomorphic knowledge base and problem set makes it possible for us to draw meaningful comparisons between students. In this section we will inspect how the scaffolding design decisions made when constructing the tutors affected the problem solving strategies exhibited by the students.

A visual inspection of the state graphs for each problem revealed significant portions of each graph were shared between the two datasets and portions that were represented in only one of the two. Despite the fact that students using Andes were capable of reaching any of the states available to students in Pyrenees, many Pyrenees states were never discovered by Andes students. As noted in Section 4.1, this suggests that guidance from the Pyrenees tutor is successful in leading students down solution paths that they would not otherwise have discovered, possibly applying skills that they would otherwise not have used.

To quantify these findings, we calculated the relative similarity of students in each tutor. For a given problem, we defined the state-similarity between datasets A and B as the probability that a randomly selected state from a student in A will be passed through by a randomly selected student in B. Recall from Section 3.2 that our state representation allows us to directly compare states across tutors. By this definition, the self-similarity of a dataset is a measure of how closely its students overlap each other while the cross-similarity is a measure of how closely its students overlap

States	Andes	Pyrenees	Solution
Andes	0.551 (0.134)	0.494 (0.153)	0.478 (0.186)
Pyrenees	0.460 (0.141)	0.688 (0.106)	0.669 (0.146)
Actions	Andes	Pyrenees	Solution
Andes	0.878 (0.085)	0.874 (0.118)	0.851 (0.140)
Pyrenees	0.828 (0.117)	0.936 (0.021)	0.923 (0.036)

Table 1: Pairwise similarity across tutors and the ideal solution path. Similarities were calculated for each problem, and each cell lists the mean (and standard deviation) over all problems. The top half covers the state similarity metrics while the bottom half of each table covers action similarity.

with the other dataset. Similarity measures for the datasets can be found at the top of Table 1.

Predictably, both datasets have higher self-similarity than cross-similarity, with Pyrenees showing higher self-similarity than Andes. This indicates that Pyrenees students chose more homogeneous paths to the goal. This is reasonable and consistent with the heavy scaffolding that is built into the system. It is important to note that our similarity metrics are not symmetric. The cross-similarity of Pyrenees with Andes is higher than the reverse. This indicates that the path taken by Pyrenees students are more likely to have been observed by Andes students than vice-versa. This has important implications for designers who are interested in collecting data from a system that is undergoing modifications. If a system becomes increasingly scaffolded and restrictive over time, past data will remain more relevant than in a system that is relaxed. In many ways this simply reflects the intuition that allowing students to explore a state space more fully will produce more broadly useful data, and restricting students will produce data that is more narrowly useful. Note that here we are only observing trends, and we make no claims of statistical significance.

In our analysis, we found that, within both datasets, many solution paths or sub-paths differed only in the order that actions were performed. In our domain, many actions do not have ordering constraints. It is possible, for example, to define the variables A and B in either order, and the resulting solution paths would deviate from one another. We thus sought to determine how much of the observed difference between our two datasets was due to these ordering effects. To that end we define the *action-similarity* between datasets A and P as the probability that a randomly selected action performed by a student in A will have been performed by a by a randomly selected student in B. These values are shown in the bottom of Table 1, and each of the trends observed for state-similarity hold, with predictably higher similarity values.

It is notable that the similarity between Pyrenees and Andes is almost as high as Andes' self-similarity, indicating that the actions taken by Pyrenees students are almost as likely to be observed in Andes students as Pyrenees students. This suggests that, for the most part, the Andes students performed a superset of the actions performed by the Pyrenees students. Thus the impact of Pyrenees is most visible in the *order* of execution, not the actions chosen. This is consistent with the design goals of Pyrenees which was set up to guide students along the otherwise unfamiliar path of the TVS.

States	Andes	Pyrenees	Solution
Andes	0.419 (0.150)	0.327 (0.139)	0.253 (0.172)
Pyrenees	0.372 (0.193)	0.709 (0.145)	0.601 (0.214)
Actions	Andes	Pyrenees	Solution
Andes	0.818 (0.151)	0.582 (0.168)	0.636 (0.205)
Pyrenees	0.678 (0.212)	0.899 (0.038)	0.879 (0.055)

Table 2: Pairwise similarity across tutors and the ideal solution path calculated using a variable-free state representation. Rows and columns are the same as in Table 1.

Problem	**Andes**	**Pyrenees**
ex132	26 (47.5%)	2 (98.7%)
ex132a	13 (33.3%)	1 (100.0%)
ex144	2 (96.6%)	1 (100.0%)
ex152	11 (0.0%)	4 (0.0%)
ex152a	8 (59.0%)	3 (97.4%)
ex152b	12 (0.0%)	1 (100.0%)
ex212	8 (71.4%)	1 (100.0%)
ex242	9 (0.0%)	2 (49.38%)
ex252	7 (76.9%)	2 (98.4%)
ex252a	4 (81.8%)	2 (98.6%)
exc137	19 (0.0%)	2 (98.75%)

Table 3: For each problem, the tables gives the number of unique solution states represented in each tutor's dataset. Note that there may exist many solution paths which reach a given solution. The following percent (in parentheses) represents the percent solution paths that ended in the pedagogically ideal solution.

We also opted to examine the impact of the variable definitions on our evaluation. As noted above, variable definitions are an atomic action. They do not depend upon any event assertion and thus have no ordering constraints unlike the principles. We did so with the hypothesis that this would increase the similarity metrics for the datasets by eliminating the least constrained decisions from consideration. Our results are shown in Table 2 below. Contrary to our expectations, this actually reduced the similarity both within and across the datasets, with the exception of Pyrenees' self-similarity. Thus the unconstrained variable definitions did not substantially contribute to the dissimilarity. Rather, most of the variation lay in the order of principle applications.

4.3 Similarity to an "ideal" solution

We now turn to exploring how well the ideal solution was represented in the datasets. For both tutors the ideal solution is the pedagogically-desirable path constructed via the TVS. Our measure of cross-similarity between two datasets can also be applied between a single solution path and a dataset by treating the single solution as a set of one. We can thus measure the average likelihood of an ideal solution state appearing in a student's solution from each dataset. The results of this calculation are shown in tables 1 and 2, using both the state- and action-similarities explained earlier. Predictably, the solution has a high similarity with Pyrenees students, as these students are scaffolded tightly and offered few chances to deviate from the path.

As Table 3 shows, Pyrenees students were funneled almost exclusively to the ideal solution on the majority of problems, even if their paths to the solution were variable. We found only one problem, Ex152, where the Pyrenees students missed the ideal path. That was traced to a programming error that forced students along a similar path. Otherwise, there was only one problem, Ex242 (discussed in Section 4.1), where a meaningful percentage of students chose a different solution. The Andes students, by contrast, were much less likely to finish in the ideal solution state, but this was also problem-dependent.

4.4 Applications of the Hint Factory

Finally, having shown that the datasets differ, and that these differences are consistent with the differing design choices of the two tutors, we sought to determine what effect those differences would have on data-driven hint generation. Our goal was to determine how applicable a hint model of the type produced by the Hint Factory would be for one dataset if it was trained on another. To that end we performed a modified version of the Cold Start Experiment [1], which is designed to measure the number of state-specific hints that Hint Factory can provide given a randomly selected dataset. The Cold Start experiment functions like leave-one-out cross-validation for state-based hint generation. In the original Cold Start experiment, one student was selected at random and removed from the dataset, to represent a "new" student using the tutor. Each remaining student in the dataset was then added, one at a time, in a random order to the Hint Factory's model. On each iteration, the model is updated and the percentage of states on the 'new' student's path for which a hint is available is calculated. This is repeated a desired number of times with new students to account for ordering effects.

For the present study we calculated cold-start curves for both the Pyrenees and Andes datasets. We also calculated curves using the opposing dataset to illustrate the growth rate for cross-tutor hints. For these modified curves we selected the hint-generating students from the opposing dataset. All four curves are shown in Figure 4. Here *AvA* and *PvP* designate the within tutor curves for Andes and Pyrenees respectively while *PvA* and *AvP* designate the cross-tutor curves for hints from Pyrenees provided to Andes users and vice-versa. Figure 4 represents an average over all problems, and therefore the x-axis extends only as far as the minimum number of students to complete a problem. As the curves illustrate, the within-tutor curves reach high rates of coverage relatively quickly with *PvP* reaching a plateau above 95% after 21 students and *AvA* reaching 85%.

The cross-tutor curves, by contrast, reach much lower limits. *AvP* reaches a plateau of over 75% coverage, while *PvA* reaches a plateau of 60% coverage. This reflects the same trends observed in Tables 1 and 2, where Andes better explains the Pyrenees data than vice-versa; however, neither dataset completely covers the other. On the one hand this result is somewhat problematic as it indicates that prior data has a limited threshold for novel tutors or novel versions of a system in the same domain. Clearly a substantive interface and scaffolding change of the type made in Pyrenees can change the state space sufficiently that we cannot trivially rely on our prior data. On the other hand, while the cross-application of data does have upper limits, those limits are comparatively high. Clearly data from a prior system can be reused and can serve as a reliable baseline for novel system, with the caveat that additional exploratory data is required.

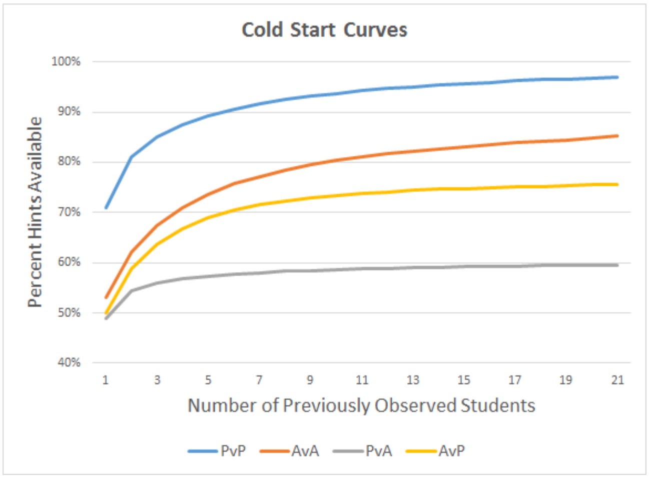


Figure 4: The four Cold Start curves, averaged across all problems. The x-axis shows the number of students used to train the model, and the y-access shows the percentage of a new student's path that has available hints. The curve labeled "XvY" indicates training on the X dataset and selecting a new student from the Y dataset (A = Andes; P = Pyrenees).

5. DISCUSSION AND CONCLUSION

Our goal in this paper was to evaluate the application of data-driven methods such as the Hint Factory to model-based tutoring systems. To that end we analyzed and compared datasets collected from two closely-related tutoring systems: Andes and Pyrenees. Through our analysis we sought to: (1) evaluate the differences between closely-related systems; (2) assess the impact of specific design decisions made in those systems for user behaviors; and (3) evaluate the potential application of data-driven hint generation across systems.

We found that, while the systems shared isomorphic domain models, problems, and ideal solutions, the observed user behaviors differed substantially. Students using the Andes system explored the space more widely, were more prone to identify novel solutions, and rarely followed the ideal solution path. Students in Pyrenees, by contrast, were far more homogeneous in their solution process and were limited in the alternative routes they explored. Contrary to our expectations, we found that this variation was not due to simple ordering variations in the simplest of steps but of alternative strategy selection for the higher-level domain principles. This is largely consistent with the design decisions that motivated both systems and with the results of prior studies.

We also found that the state-based hint generation method used in the Hint Factory can be applied to the Andes and Pyrenees data given a suitable state representation. For this analysis we opted for a set-based representation given the absence of strong ordering constraints across the principles. We then completed a cold-start analysis to show that the cross-tutor data could be used to bootstrap the construction of hints for a novel system but does not provide for complete coverage.

Ultimately we believe that the techniques used for data-driven hint generation have direct application to model-based systems. Data-driven analysis can be used to identify the behavioral differences between closely related systems and, we would argue, changes from one version of a system to another. We also found that these changes can be connected to the specific design decisions made during development. Further, we found that data-driven methods can be applied to model-based tutoring data to generate state-based hints. We believe that hint information of this type may be used to supplement the existing domain models to produce more user-adaptive systems. In future work we plan to apply these analyses to other appropriate datasets and to test the incorporation of state-driven hints or hint refinement to existing domain models.

6. ACKNOWLEDGMENTS

Work supported by NSF Grant #1432156 "Educational Data Mining for Individualized Instruction in STEM Learning Environments" Min Chi & Tiffany Barnes Co-PIs.

7. REFERENCES

[1] T. Barnes and J. Stamper. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In *Intelligent Tutoring Systems (ITS)*, pages 373–382, 2008.

[2] T. Barnes and J. C. Stamper. Automatic hint generation for logic proof tutoring using historical data. *Educational Technology & Society*, 13(1):3–12, 2010.

[3] M. Chi and K. VanLehn. Eliminating the gap between the high and low students through meta-cognitive strategy instruction. In *Intelligent Tutoring Systems (ITS)*, volume 5091, pages 603–613, 2008.

[4] M. Chi and K. VanLehn. Meta-Cognitive Strategy Instruction in Intelligent Tutoring Systems: How, When, and Why. *Educational Technology & Society*, 3(1):25–39, 2010.

[5] A. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *User Modeling 2001*, pages 137–147, 2001.

[6] M. Eagle and T. Barnes. Exploring Differences in Problem Solving with Data-Driven Approach Maps. In *Educational Data Mining (EDM)*, 2014.

[7] M. Eagle and T. Barnes. Exploring Networks of Problem-Solving Interactions. In *Learning Analytics (LAK)*, 2015.

[8] D. Fossati, B. D. Eugenio, and S. Ohlsson. I learn from you, you learn from me: How to make iList learn from students. In *Artificial Intelligence in Education (AIED)*, 2009.

[9] F. Hayes-Roth, D. A. Waterman, and D. B. Lenat. *Building Expert Systems*. Addison-Wesley Publishing Company Inc., Reading, Massachusetts, U.S.A., 1983.

[10] A. Hicks, B. Peddycord III, and T. Barnes. Building Games to Learn from Their Players: Generating Hints in a Serious Game. In *Intelligent Tutoring Systems (ITS)*, pages 312–317, 2014.

[11] S. Katz, A. Lesgold, E. Hughes, D. Peters, G. Eggan, M. Gordin, and L. Greenberg. Sherlock 2: An intelligent tutoring system built on the lrdc framework. In C. P. Bloom and R. B. Loftin, editors, *Facilitating the Development and Use of Interactive Learning Environments*, Computers, Cognition, and Work, chapter 10, pages 227 – 258. Lawrence Erlbaum Associates, Mawah New Jersey, 1998.

[12] R. Kumar, M. E. Roy, B. Roberts, and J. I. Makhoul. Toward automatically building tutor models using multiple behavior demonstrations. In S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, editors, *Proceedings of the* 12^{th} *International Conference on Intelligent Tutoring Systems*, LNCS 8474, pages 535 – 544. Springer Verlag, 2014.

[13] B. Peddycord III, A. Hicks, and T. Barnes. Generating Hints for Programming Problems Using Intermediate Output. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*, pages 92–98, 2014.

[14] C. Piech, M. Sahami, J. Huang, and L. Guibas. Autonomously Generating Hints by Inferring Problem Solving Policies. In *Learning at Scale (LAS)*, 2015.

[15] K. Rivers and K. Koedinger. Automatic generation of programming feedback: A data-driven approach. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013.

[16] K. Rivers and K. Koedinger. Automating Hint Generation with Solution Space Path Construction. In *Intelligent Tutoring Systems (ITS)*, 2014.

[17] J. C. Stamper, M. Eagle, T. Barnes, and M. J. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *I. J. Artificial Intelligence in Education*, 22(1-2):3–17, 2013.

[18] K. VanLehn, C. Lynch, K. G. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Five years of evaluations. In C. Looi, G. I. McCalla, B. Bredeweg, and J. Breuker, editors, *Artificial Intelligence in Education - Supporting Learning through Intelligent and Socially Informed Technology, Proceedings of the 12th International Conference on Artificial Intelligence in Education, AIED 2005, July 18-22, 2005, Amsterdam, The Netherlands*, volume 125 of *Frontiers in Artificial Intelligence and Applications*, pages 678–685. IOS Press, 2005.

[19] K. VanLehn, C. Lynch, K. G. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Lessons learned. *I. J. Artificial Intelligence in Education*, 15(3):147–204, 2005.

[20] K. VanLehn and B. van de Sande. The Andes physics tutoring system: An experiment in freedom. *Advances in intelligent tutoring systems.*, pages 421–443, 2010.

An Exploration of Data-Driven Hint Generation in an Open-Ended Programming Problem

Thomas W. Price
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
twprice@ncsu.edu

Tiffany Barnes
North Carolina State University
890 Oval Drive
Raleigh, NC 27606
tmbarnes@ncsu.edu

ABSTRACT

Data-driven systems can provide automated feedback in the form of hints to students working in problem solving environments. Programming problems present a unique challenge to these systems, in part because of the many ways in which a single program can be written. This paper reviews current strategies for generating data-driven hints for programming problems and examines their applicability to larger, more open-ended problems, with multiple, loosely ordered goals. We use this analysis to suggest directions for future work to generate hints for these problems.

1. INTRODUCTION

Adaptive feedback is one of the hallmarks of an Intelligent Tutoring System. This feedback often takes the form of hints, pointing a student to the next step in solving a problem. While hints can be authored by experts or generated by a solver, more recent data-driven approaches have shown that this feedback can be automatically generated from previous students' solutions to a problem. The Hint Factory [18] has successfully generated data-driven hints in a number of problem solving domains, including logic proofs [2], linked list problems [5] and a programming game [12]. The Hint Factory operates on a representation of a problem called an interaction network [4], a directed graph where each vertex represents a student's state at some point in the problem solving process, and each edge represents a student's action that alters that state. A solution is represented as a path from the initial state to a goal state. A student requesting a hint is matched to a previously observed state and directed on a path to a goal state. The Hint Factory takes advantage of the intuition that students with the same initial state and objective will follow similar paths, producing a well-connected interaction network. When this occurs, even a relatively small sample of student solutions can be enough to provide hints to the majority of new students [1].

While problems in many domains result in well-connected networks, this is not always the case. Programming problems have a large, often infinite, space of possible states. Even a relatively simple programming problem may have many unique goal states, each with multiple possible solution paths, leaving little overlap among student solutions. Despite this challenge, a number of attempts have been made to adapt the Hint Factory to programming problems [9, 12, 16]. While these approaches have been generally successful, they are most effective on small, well structured programming problems, where the state space cannot grow too large. This paper explores the opposite type of problem: one that has a large state space, multiple loosely ordered goals, unstructured output and involves creative design. Each of these attributes poses a challenge to current data-driven hint generation techniques, but they are also the attributes that make such problems interesting, useful and realistic. In this paper, we will refer to these as open-ended programming problems. We investigate the applicability of current techniques to these problems and suggest areas for future research.

The primary contributions of this paper are 1) a review of current data-driven hint generation methods for programming problems, 2) an analysis of those methods' applicability to an open-ended problem and 3) a discussion of the challenges that need to be addressed before we can expect to generate hints for similar problems.

2. CURRENT APPROACHES

Current approaches to generating data-driven programming hints, including alternatives to the Hint Factory [10, 13, 17], can be broken down into three primary components:

1. A representation of a student's state and a method for determining when one state can be reached from another, meaning they are connected in the network
2. An algorithm that, given a student's current state, constructs an optimal path to a goal state. Often we simplify this to the problem of picking the first state on that path
3. A method to present this path, or next state, to the student in the form of a hint

For the purposes of this paper, we limit our discussion to the first step in this process. (For a good discussion of the second step, see an analysis by Piech et al. [13], comparing path selection algorithms.) While in some domains this first step is straightforward, in programming tasks, especially open-ended problems, it is likely the most challenging. The simplest approach is to take periodic snapshots of a student's code and treat these as states, connecting consecutive snapshots in the network. However, because two students' programs are unlikely to match *exactly*, this approach is likely to produce a very sparse, poorly connected network, making it difficult to match new students to prior solution attempts. A variety of techniques have been presented to address this problem, which can be grouped into three main strategies: canonicalization, connecting states and alternative state definitions.

2.1 Canonicalization

Canonicalization is the process of putting code states into a standardized form, often by removing semantically unimportant information, so that trivial difference do not prevent two states from matching. Rivers and Koedinger [15] present a method for canonicalizing student code by first representing it as an Abstract Syntax Tree (AST). Once in this form, they apply a number of functions to canonicalize the code, including normalizing arithmetic and boolean operators, removing unreachable and unused code, and inlining helper functions. After performing this canonicalization on a set of introductory programming problems, they found that a median 70% of states had at least one match in the network. Jin et al. [9] represent a program's state as a Linkage Graph, where each vertex is a code statement, and each directed edge represents an ordering dependency, determined by which variables are read and assigned to in each statement. This state representation allows the Hint Factory to ignore statement orderings which are not important to the execution of the program. Lazar and Bratko [10] use the actual text of Prolog code to represent a student's state, and then canonicalize the code by removing whitespace and normalizing variable names.

2.2 Connecting States

Even with canonicalization, a student requesting a hint may not match any existing state in the network. In this case, we can look for a similar state and create a connection between them. Rivers and Koedinger [16] use normalized string edit distance as a similarity metric between two program states. They connect any two states in the network which have at least 90% similarity, even if no historical data connect these states. Additionally, they use a technique called path construction to generate new solution paths from a given state to a nearby, unconnected goal state by searching for a series of insertions, deletions and edits to their AST that will transform it into the goal state [17]. They also use this method to discover new goal states which may be closer to the student's current state. Jin et al. [9] use a similar technique to transform their Linkage Graphs to better match the current state of a student when no direct matches can be found in the interaction network. Piech et al. [13] use path construction to interpolate between two consecutive states on a solution path which differ by more than one edit. This is useful to smooth data when student code is recorded in shapshots that are too far apart.

2.3 Alternate State Definitions

Another approach is to forego the traditional code-based representation of a student's state, and use an alternate definition. Hicks and Peddycord [7, 12] used the Hint Factory to generate hints for a programming game called Bots, in which the player writes a program to direct a robot through various tasks in a 3D level. They chose to represent the state of a player's program as the final state of the game world after the program was executed. They compared the availability of hints when using this "world state" model with a traditional "code state" model, and found that using world states significantly reduced the total number of states and increased the availability of hints. The challenge with this approach, as noted by the authors, is the generation of actionable hints. A student may be more capable of making a specific change to her code than determining how to effect a specific change in the code's output.

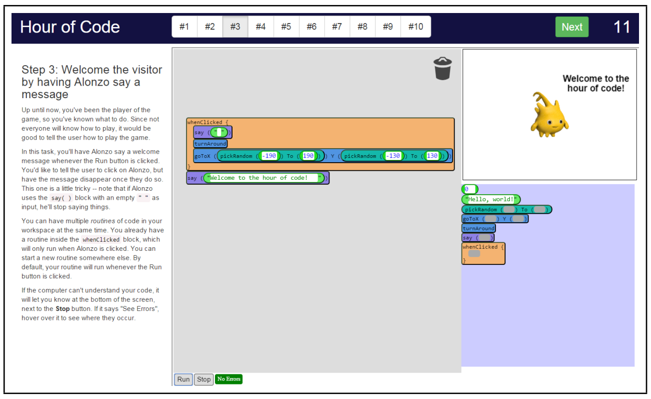
3. AN OPEN-ENDED PROBLEM

The above techniques have all shown success on smaller, well-structured problems, with ample data. We want to investigate their applicability to an open-ended problem, as described in Section 1, where this is not the case. The purpose of this paper is not to create actionable hints, nor are we attempting to show the failures of current methods by applying them to an overly challenging task. Rather, our purpose is exploratory, using a small dataset to identify areas of possible future work, and challenges of which to be mindful when moving forward with hint generation research.

We collected data from a programming activity completed by 6th grade students in a STEM outreach program called SPARCS [3]. The program, which meets for half-day sessions approximately once a month during the school year, consists of lessons designed and taught by undergraduate and graduate students to promote technical literacy. The class consisted of 17 students, 12 male and 5 female.

The activity was a programming exercise based on an Hour of Code activity from the Beauty and Joy of Computing curriculum [6]. It was a tutorial designed to introduce novices to programming for the first time. The exercise had users create a simple web-based game, similar to whack-a-mole, in which players attempt to click on a sprite as it jumps around the screen to win points. The exercise was split into 9 objectives, with tutorial text at each stage. Students were not required to finish an objective before proceeding. A finished project required the use of various programming concepts, including events, loops, variables and conditionals. The students used a drag-and-drop, block-based programming language called Tiled Grace [8], which is similar to Scratch [14]. The user writes a program by arranging code blocks, which correspond directly to constructs in the Grace programming language. The editor also supports switching to textual coding, but this feature was disabled. A screenshot of the activity can be seen in Figure 1.

During the activity, the students were allowed to go through the exercise at their own pace. If they had questions, the students were allowed to ask for help from the student volunteers. Students were stopped after 45 minutes of work. Snapshots of a student's code were saved each time it was run and periodically throughout the session. Occasional technical issues did occur in both groups. One student had severe technical issues, and this student's data was not analyzed (and is not reflected in the counts above). Students



Step 3: Welcome the visitor by having Alonzo say a message

Up until now, you've been the player of the game, so you've known what to do. Since not everyone will know how to play, it would be good to tell the user how to play the game.

In this task, you'll have Alonzo say a welcome message whenever the Run button is clicked. You'd like to tell the user to click on Alonzo, but have the message disappear once they do so. This one is a little tricky -- note that if Alonzo uses the `say( )` block with an empty `" "` as input, he'll stop saying things.

You can have multiple *routines* of code in your workspace at the same time. You already have a routine inside the `whenClicked` block, which will only run when Alonzo is clicked. You can start a new routine somewhere else. By default, your routine will run whenever the Run button is clicked.

If the computer can't understand your code, it will let you know at the bottom of the screen, next to the **Stop** button. If it says "See Errors", hover over it to see where they occur.

```
whenClicked {
  say ("")
  turnAround
  goToX (pickRandom (-190) To (190)) Y (pickRandom (-130) To (130))
}
say ("Welcome to the hour of code! ")
```

```
0
"Hello, world!"
pickRandom ( ) To ( )
goToX ( ) Y ( )
turnAround
say ( )
whenClicked {
}
```

Figure 1: A screenshot of the Hour of Code activity that students completed. Students received instructions on the left panel. The center panel was a work area, and students could drag blocks in from the bottom-right panel. The top-right panel allowed students to test their games.

produced on average 148.5 unique code states and accomplished between 1 and 6 of the activity's objectives, averaging 3.2 objectives per student.

4. ANALYSIS

We attempted to understand the applicability of each of the techniques discussed in Section 2 to our dataset. However, since the output of our program was a game that involved nondeterminism, we felt it would be inappropriate to attempt to represent a program's state as the result of its execution. We therefore focused on the first two strategies, canonicalization and connecting states.

4.1 Canonicalization

Our initial representation of a student's code state was a tree, where each code block was a node, and its children included any blocks that were nested inside of it. In this way, our representation was similar to Rivers and Koedinger's ASTs. To get a baseline for the sparsity of our dataset, we first analyzed the code states without performing any canonicalization. We calculated the total number of states in the interaction network and the percentage which were only reached by one student. Of those states reached by multiple students, we calculated the mean and median number of students who reached them. We also calculated the percentage of each student's states that were unreached by any other student in the dataset.

We then canonicalized the data by removing variable names and the values of number and string literals. Our problem featured very few arithmetic or logical operators, and these were generally not nested, so we did not normalize them, as suggested by Rivers and Koedinger [15]. We reran our analyses on the canonicalized data. To ensure that we had effectively removed all unimportant ordering information, we recursively sorted the children of each node in the tree. This effectively removed any ordering information from a student's code state, and kept only hierarchical information. This is somewhat more extreme than the Linkage Graphs of Jin et al. [9], and it does allow two meaningfully different code states to be merged in the process. We therefore see this as an upper bound on the value of removing unimportant orderings from a code state. We recomputed our metrics for the ordered-canonicalized interaction network as well. The results can be seen in Table 1. Our later analyses use the unsorted, canonicalized code representation.

	Raw	Canonical	Ordered
Total States	2380	1781	1656
% Unique	97.5%	94.8%	92.8%
Mean NU Count	3.44	3.95	2.82
Median NU Count	2	2	2
Mean % Path Unique Standard Deviation	89.9% (6.67)	83.0% (10.5)	78.9% (13.3)

Table 1: Various measures of the sparseness of the interaction network for the raw, canonicalized, and ordered-canonicalized state representations. Mean and median NU counts refer to the number of students who reached each non-unique state.

These results indicate that canonicalization does little to reduce the sparsity of the state space, with students spending most of their time in states that no other student has seen. For comparison, recall Rivers and Koedinger found 70% of states in a simple programming problem had a match after canonicalization [15], though they were using a much larger dataset. In our dataset, it is unlikely that we would be able to find a direct path from a new student's state to a goal state in order to suggest a hint.

4.2 Connecting States

To address this, we explored the feasibility of connecting a new student's state to a similar, existing state in the network. It is unclear how close two code states should be before it is appropriate to connect them as in [16], or to generate a path between them as in [17]. It certainly depends on the state representation and distance metric used. Rather than identifying a cutoff and measuring how often these techniques could be applied, we chose to visualize the distance between two students and make qualitative observations. Because our code states were already represented as trees, we used Tree Edit Distance (TED) as a distance metric. While Rivers and Koedinger reported better success with Levenshtein distance [16], we believe that TED is the most appropriate distance metric for block code, where tree edit operations correspond directly to user actions.

For each pair of students, A and B, we created an N by M distance matrix, D, where N is the number of states in A's solution path, and M is the number of states in B's solution path. $D_{i,j} = d(A_i, B_j)$, where d is the TED distance function, A_i is the ith state of A and B_j is the jth state of B. We used the RTED algorithm [11] to calculate the distance function, putting a weight of 1.0 on insertions, deletions and replacements. We also omitted any state which was identical to its predecessor state. We normalized these values by dividing by the maximum value of $D_{i,j}$, and plotted the result as an image. Three such images can be seen in Figure 2.

We also calculated the "path" through this matrix that passes through the least total distance. This does not represent a

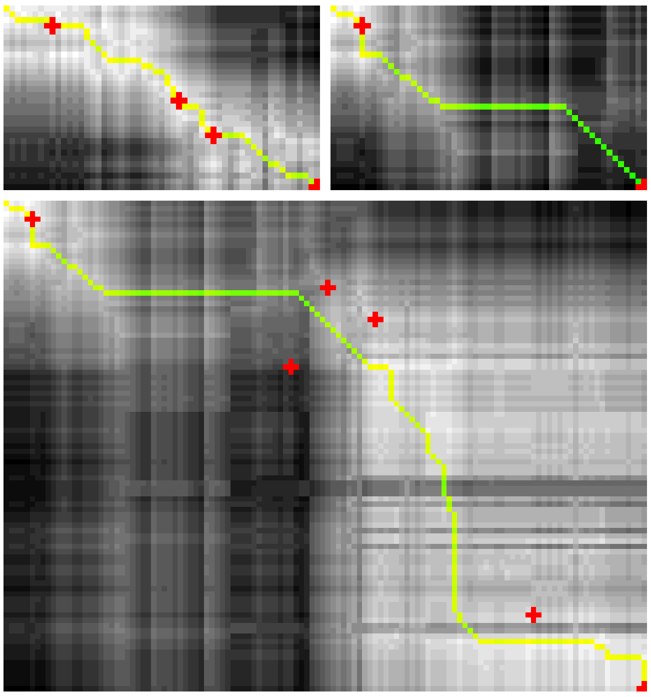


Figure 2: Three distance matrices, each comparing two students, where each pixel represents the TED between two states. Lighter shades of gray indicate smaller distances. The green/yellow line shows the path through the matrix with minimized total distance, with yellow shades indicating smaller distances. Red crosses indicate where both students met an objective. The top-left figure compares two students as they completed objectives 1-4 in the exercise. In this example, the minimum-distance line crosses each objective. The top right figure also depicts two students completing objective 4, but the darker colors and straighter line indicate less alignment. The bottom figure depicts two students completing objectives 1-6, with high alignment.

path through the interaction network, but rather an alignment between the states of student A and those of student B. Each pixel of the line represents a pairing of a state from A with a state from B, such that these pairings are contiguous and represent the smallest total distance. While we do not suggest applying this directly as a strategy for hint generation, it serves an a useful visual indicator of the compatibility of two students for hinting purposes.

An alternate approach would have been to pair each state in the interaction network with its closest pair from any other student, and use this as a measure of how sparse the network was. We chose to compare whole students, rather than individual states, because we felt that the former could lead to strange hinting behavior. Imagine, for instance, that a student requests a hint, which initially points to a state from student B, but at the very next step requests a hint that points instead to student C. Perhaps the attributes that make the student's state similar to that of B are different from those that make the state similar to C. The resulting hints would be at best confusing, and at worst conflicting.

	Mean	Median	Max	Farthest
1	0.25 (0.27)	0.15 (0.29)	0.76 (0.56)	2.23 (0.75)
2	4.88 (3.93)	4.95 (4.34)	9.18 (5.74)	12.73 (6.10)
4	4.92 (2.77)	4.83 (2.78)	10.11 (3.69)	14.67 (4.77)
5	7.79 (1.32)	7.75 (1.41)	13.17 (1.72)	18.17 (1.72)
6	7.49 (1.11)	7.76 (1.37)	13.17 (0.98)	18.67 (1.75)

Table 2: For each objective, average distances (and standard deviations) of minimum-distance student pairs, using the mean, median and max metrics. For reference, the final column represents the average maximum distance each student moved from the start state while completing the objective.

A visual inspection of these matrices reveals that while many student pairs are quite divergent, some show a notable closeness throughout the exercise. In order to quantify these results, we developed a set of distance metrics between *students*. First, the distance matrix and the minimum-distance "path" were calculated for the two students. The path is comprised of pairs of states, and for each pair, we recorded the tree edit distance between the states. From this list of distances, we calculated the mean, median and maximum distances between the two students. We looked at each objective in the exercise, and isolated the relevant subpath of each student who completed that objective. We paired each of these subpaths with the most similar subpath in the set, using the mean, median and max distance metrics. Table 2 shows the average values of these minimized pairs of students, using each metric. Objective 3, and objectives 7-9 were omitted, as too few students completed them.

5. DISCUSSION

We have attempted to apply a meaningful canonicalization to our state space, which did serve to reduce the number of states by 30.4%. However, as seen in Table 1, even after the strongest canonicalization, over 90% of the states in the interaction network had only been reached by one student, with an average 78.9% of the states in each student's solution path being unique to that student. It seems that our approach to canonicalization is insufficient to produce a meaningful reduction of the state space, though it is possible a more stringent canonicalization would be more effective.

Connecting existing states seems to be a more promising approach, giving us the ability to link new states to previously observed states, even when they do not match exactly. Our distance matrices indicate that some students take parallel, or slowly diverging solution paths, which suggests that they may be useful to each other in the context of hinting. As shown in Figure 2, students are often closest together when completing the same objective. This may seem self-evident, but it does indicate that our distance metric is meaningful. It is more difficult to put the actual TED values into context. Students get, on average, farther away from their closest paired student as they complete more objectives, but this average distance does not exceed 8 tree edits during the first 6 objectives. To put that number into context, during this same time students do not, on average, get more than 19 tree edits away from the start state. This suggest that there is certainly hint-relevant knowledge in these paired students, but that it may be difficult to harness this knowledge to generate a hint.

5.1 Limitations

It is important to note that this analysis is an exploratory case study, and makes no strong claims, only observations. We studied data from only 17 novice programmers, and the problem we analyzed was highly complex, involving multiple control structures, loosely ordered objectives, and unstructured output. This makes the problem quite dissimilar from previous problems that have been been studied in the context of hint generation, making it difficult to determine what observations should be generalized.

5.2 Future Work

Rivers and Koedinger [16], as well as Jin et al. [9] note the limitations of their methods for larger problems, and each suggest that breaking a problem down into subproblems would help to address this. Lazar and Bratko [10] attempted this in their Prolog tutor by constructing hints for individual lines of code, which were treated as independent subproblems. Similarly, we may be able to isolate the subsection of a student's code that is currently relevant, and treat this as an independent problem. The hierarchical nature of block-based coding environments lends itself to this practice, making it an appealing direction for future work.

Our work makes the simplifying assumption that unweighted TED is a reliable distance metric for code, but future work should investigate alternative metrics. This might include a weighted TED metric, which assigns different costs to insertions, deletions and replacements, or even to different types of nodes (e.g. deleting a for-loop node might cost more than a function call node). Regardless of the metric used, once two proximate states are identified, it is still an open question how this information can be best used for hint generation. It is possible to construct a path between the states and direct a student along this path. However, future work might also investigate how to extract hint-relevant information from one state and apply it to a similar state directly.

Because of the nature of our problem's output, we did not explore non-code-based state representations, as described in Section 2.3. It would still be worth investigating how this might be applied to open-ended problems. For instance, a code state could be represented as a boolean vector, indicating whether the code has passed a series of Unit Tests, and hints could direct the student to the current flaw in their program. However, creating actionable hints from this information would pose a significant challenge.

6. REFERENCES

[1] T. Barnes and J. Stamper. Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In *Intelligent Tutoring Systems (ITS)*, pages 373–382, 2008.

[2] T. Barnes, J. Stamper, L. Lehman, and M. Croy. A pilot study on logic proof tutoring using hints generated from historical student data. In *Proceedings of the 1st Annual International Conference on Educational Data Mining (EDM)*, pages 1–5, 2008.

[3] V. Cateté, K. Wassell, and T. Barnes. Use and development of entertainment technologies in after school STEM program. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 163–168, 2014.

[4] M. Eagle, M. Johnson, and T. Barnes. Interaction Networks: Generating High Level Hints Based on Network Community Clustering. In *International Educational Data Mining Society*, pages 164–167, 2012.

[5] D. Fossati, B. D. Eugenio, and S. Ohlsson. I learn from you, you learn from me: How to make iList learn from students. In *Artificial Intelligence in Education (AIED)*, 2009.

[6] D. Garcia, B. Harvey, L. Segars, and C. How. AP CS Principles Pilot at University of California, Berkeley. *ACM Inroads*, 3(2), 2012.

[7] A. Hicks, B. Peddycord III, and T. Barnes. Building Games to Learn from Their Players: Generating Hints in a Serious Game. In *Intelligent Tutoring Systems (ITS)*, pages 312–317, 2014.

[8] M. Homer and J. Noble. Combining Tiled and Textual Views of Code. In *Proceedings of 2nd IEEE Working Conference on Software Visualization*, 2014.

[9] W. Jin, T. Barnes, and J. Stamper. Program representation for automatic hint generation for a data-driven novice programming tutor. In *Intelligent Tutoring Systems (ITS)*, 2012.

[10] T. Lazar and I. Bratko. Data-Driven Program Synthesis for Hint Generation in Programming Tutors. In *Intelligent Tutoring Systems (ITS)*. Springer, 2014.

[11] M. Pawlik and N. Augsten. RTED: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345, 2011.

[12] B. Peddycord III, A. Hicks, and T. Barnes. Generating Hints for Programming Problems Using Intermediate Output. In *Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014)*, pages 92–98, 2014.

[13] C. Piech, M. Sahami, J. Huang, and L. Guibas. Autonomously Generating Hints by Inferring Problem Solving Policies. In *Learning at Scale (LAS)*, 2015.

[14] M. Resnick, J. Maloney, H. Andrés, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai. Scratch: programming for all. *Communications of the ACM*, 52(11):60–67, 2009.

[15] K. Rivers and K. Koedinger. A canonicalizing model for building programming tutors. In *Intelligent Tutoring Systems (ITS)*, 2012.

[16] K. Rivers and K. Koedinger. Automatic generation of programming feedback: A data-driven approach. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, 2013.

[17] K. Rivers and K. Koedinger. Automating Hint Generation with Solution Space Path Construction. In *Intelligent Tutoring Systems (ITS)*, pages 329–339, 2014.

[18] J. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *Artificial Intelligence in Education (AIED)*, 22(1):3–17, 2013.

Studio: Ontology-Based Educational Self-Assessment

Christian Weber
Corvinno Technology
Transfer Center
Budapest, Hungary
cweber@corvinno.com

Réka Vas
Corvinus University
of Budapest
Budapest, Hungary
reka.vas@uni-corvinus.hu

ABSTRACT

Students, through all stages of education, grasp new knowledge in the context of knowledge memorized all through their previous education. To self-predict personal proficiency in education, self-assessment acts as an important learning feedback. The in-house developed Studio suit for educational self-assessment enables to model the educational domain as an ontology-based knowledge structure, connecting assessment questions and learning material to each element in the ontology. Self-assessment tests are then created by utilizing a sub-ontology, which frames a tailored testing environment fitting to the targeted educational field. In this paper we give an overview of how the educational data is modeled as a domain ontology and present the concepts of different relations used in the Studio system. We will deduct how the presented self-assessment makes use of the knowledge structure for online testing and how it adapts the test to the performance of the student. Further we highlight where potentials are for the next stages of development.

Keywords

Education, adaptive test, self-assessment, educational ontology

1. INTRODUCTION

Students exploring new fields of education are always confronted with questions regarding their individual progress: how much do they know after iterations of learning, in which directions should they progress to fill the field most effectively, how to grasp the outline and details of the field and how much of their time should they invest in learning? Especially in higher education, where learning becomes a self-moderated, personalized process, students are in need of continuous self-assessment to capture their current state of proficiency. At the same time, the unframed, informal self-prediction of students regarding their personal skills is often substantive and systematically flawed [1]. Here a systematic and objective solution for self-assessment is substantial to prevent a wrong or biased self-evaluation and to support the self-prediction of the personal proficiency.

Following Jonassen, knowledge in education could be split into nine types across three categories to capture the human's cognitive behavior. In his discussion, eight out of nine knowledge types underline that knowledge in the scope of learning is interrelated and strongly associated with previous experiences [2]. As such, a supporting solution for self-assessment should grasp and formalize the knowledge to assess in the context of related knowledge.

The Studio suit for educational self-assessment, presented in this paper, provides here a software solution for testing the personal proficiency in the context of related knowledge. It enables to model areas of education as a substantial source for assessment and narrows the gap between a potentially flawed self-prediction and the real proficiency, by offering an objective and adaptive online knowledge-test. To follow the natural learning process and enable an easy extension, the software embeds the assessed knowledge into a network of contextual knowledge, which enables to adapt the assessment to the responses of the students.

This paper will give an overview of the Studio educational domain ontology and the aspects of the system supporting personalized self-assessment. Further it will highlight potentials for data mining on the gathered educational data with an outlook on the next stages of evaluation.

2. THE STUDIO APPROACH FOR SELF-ASSESSMENT

The basic concept of Studio is to model the focused education as an interrelated knowledge structure, which divides the education into sub-areas and knowledge items to know. The managed structure formalizes the relation between knowledge areas as a learning context and models the requirements to master specific parts of the education. This structure is used to create and support knowledge tests for students. Through this combination of assessment and knowledge structure, the student gains the freedom to explore not only single knowledge items but the education in the context of related knowledge areas, while the embedded requirements are used to map the modeled knowledge against the expected educational outcome.

The assessment-system is designed to be accompanied by phases of learning within the system, where the student gets access to learning material, based on and supported by the test feedback. This combined approach offers a unique self-assessment to the students, where the backing knowledge context is used to adapt the assessment in dependency of the test performance of the student.

Before any regular examination students may use Studio to assess their knowledge on their own. It is the tutor's responsibility to set the course of self-assessment test in Studio system by selecting knowledge areas and sub-knowledge areas which are relevant for the target education from the domain ontology. Then the frame will be automatically completed with elements from the ontology which detail the selected knowledge areas and are modeled as required for this part of the education.

As the system stores assessment questions for each knowledge element, Studio will then automatically prepare an assessment test, based on the defined selection and the domain ontology. The resulting knowledge-test is then accessible as a self-assessment test for the student, who explores the backed knowledge structure, which pictures the expected learning outcome, in cycles of testing, reflection and learning. The process of test definition and assessment is shown in Figure 1, while the result preparation for reflection and learning is discussed in section 2.5.

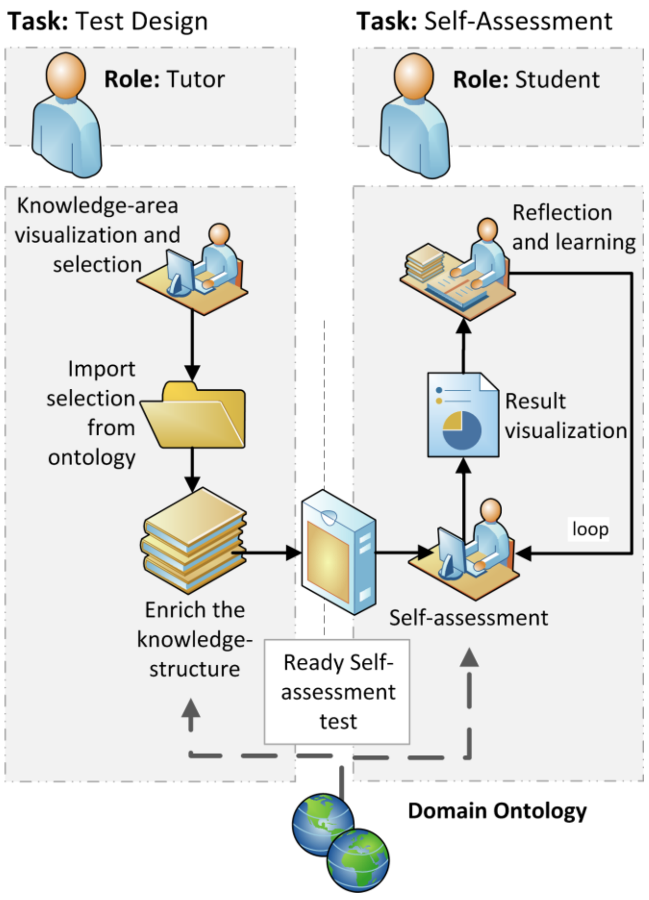


Figure 1: The overall design, assess and reflection cyle of the system.

2.1 The Educational Domain Ontology

The Studio system is based on a predesigned educational ontology, explained in detail by Vas in [3]. Domain ontology is a frequently used term in the field of semantic technologies and underlines the storage and conceptualization of domain knowledge and is often used in a number of projects and solutions [4][5][6] and could address a variety of domains with different characteristics in their creation, structure and granularity, depending on the aim and the modeling person [7]. A specialization in terms of the field is the educational domain ontology which is a domain ontology adapted to the area and concepts of education. They could target to model different aspects of education as the curriculum or aspects relevant for the task of learning and course creation [8][9][10] or describe the design, use and retrieval of learning materials till creating courses [11], as well as directly the learner within the education [12].

Within the area of educational ontologies, domain ontologies tend to model too specific details of the education, in an attempt to model the specific field as complete as possible. This enables a comprehensive view on the field but it comes at the cost of generality, with the potential to be inflexible to handle changes. Other concepts model the education across different ontologies, matching concepts like the learner, the education and the course description, introducing a broad horizon but with additional overhead to combine modelled insights and reason on new instances.

The appeal of the Studio educational ontology is the size and focus of the main classes and their relationships between each other. The knowledge to learn is the main connecting concept in the core of education. It enables a great flexibility to be resourceful for different education related questions. An example is here the business process management extension PROKEX, which maps process requirements against knowledge areas to create assessment test, reflecting the requirements of attached processes [13].

An important factor in learning is the distance between the expectation of the tutor and the learning performance of the student. Here a short cycle of repeated assessment and learning is a major factor for a better personal learning performance [14]. This aspect directly benefits from the focused concentration on knowledge-areas as the main exchange concept between students and tutors. As even further the close connections between learners and educators via direct tutoring is one major enabler for computer aided systems [15], each step towards a more direct interaction through focused concepts is an additional supporter.

The class structure fuses the idea of interrelated knowledge with a model of the basic types of educational concepts, involved in situations of individual learning. Figure 2 visualizes the class concepts as knowledge elements, together with the relation types, used to model the dependencies between different aspects of knowledge and learning within the educational ontology.

The Knowledge Area is the super-class and core-concept of the ontology. The ontology defines two qualities of main relations between knowledge areas: Knowledge areas could be a sub-knowledge area of other knowledge areas with the "has_sub-knowledge_area" relation or be required for another knowledge area with the "requires_knowledge_of" relation. A knowledge area may have multiple connected knowledge areas, linked as a requirement or sub-area. The "requires_knowledge_of" relation defines that a node is required to complete the knowledge of a parent knowledge area. This strict concept models a requirement dependency between fields of knowledge in education and yields the potential to assess perquisites of learning, analog to the basic idea of perquisites within knowledge spaces, developed by Falmagne [16].

Education is a structured process which splits the knowledge to learn into different sub-aspects of learning. Knowledge areas in the ontology are extended by an additional sub-layer of knowledge elements in order to effectively support educational and testing requirements. Figure 2 visualizes the sub-elements and their relations. By splitting the assessed knowledge into sub-concepts, the coherence and correlation of self-assessment questions could be expressed more efficiently and with the potential of a more detailed educational feedback.

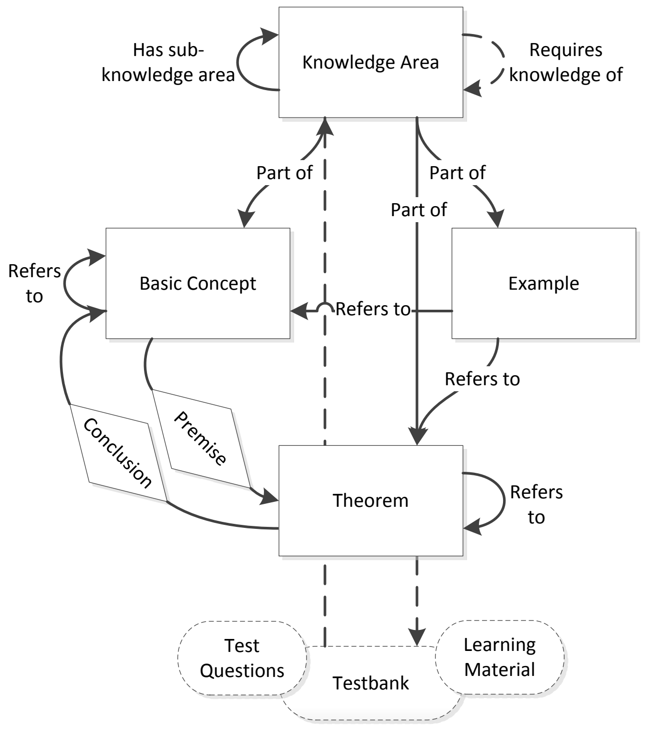


Figure 2: Model of the educational ontology.

Theorems express in a condensed and structured way the fundamental insights within knowledge areas. They fuse and explain the basic concepts of the depicted knowledge and set them in relation to the environment of learning with examples. Multiple theorems could be "part_of" a knowledge area. Each theorem may define multiple Basic Concepts as a "premise" or "conclusion", to structure how the parts of the knowledge area are related. Examples enhance this parts as a strong anchor for self-assessment questions and "refer_to" the theorems and basic concepts as a "part_of" one or more knowledge areas.

2.2 The Testbank

In order to connect the task of self-assessment with the model of the educational domain, the system integrates a repository of assessment questions. Each question addresses one element of the overall knowledge and is directly associated with one knowledge area or knowledge element instance within the ontology. The domain ontology provides here the structure for the online self-assessment while the repository of questions supplements the areas as a test bank. The target of the self-assessment is to continuously improve the personal knowledge within the assessed educational areas, by providing feedback on the performance after each phase of testing. To do so, the Studio system includes Learning Material connected to the test bank and the knowledge areas, analog to the test questions. The learning material is organized into sections as a structured text with mixed media, as pictures and videos, and is based on a wiki-engine to maintain the content, including external links.

2.3 Creating and Maintaining Tests

The creation and continues maintenance of the domain ontology is a task of ontology engineering. The ontology engineer (the ontologist), creates, uses and evaluates the ontology [17], with a strong focus on maintaining the structure and content. Within Studio, this process is guided and supported by a specialized administration workflow and splits in three consecutive task areas, in line with decreasing access rights:

- **Ontology engineering (instance level):** The creation and linking of instances of the existing knowledge-area classes into the overall domain ontology.
- **Test definition:** Knowledge areas, which are relevant to a target self-assessment test, are selected and grouped into specialized containers called Concept Groups (CG). These concept groups are organized into a tree of groups, in line with the target of the assessment. The final tree in this regards captures a sub-ontology. Concept groups are internally organized based on the overall ontology and include all relations between knowledge elements, as defined within the domain ontology.
- **Question and learning material creation:** Questions and learning materials alike are directly connected to single knowledge areas within the designed test frame and get imported, if already existing, from the domain ontology. More questions and learning materials are defined now, in line with the additional need of the targeted education and are available for future tests.

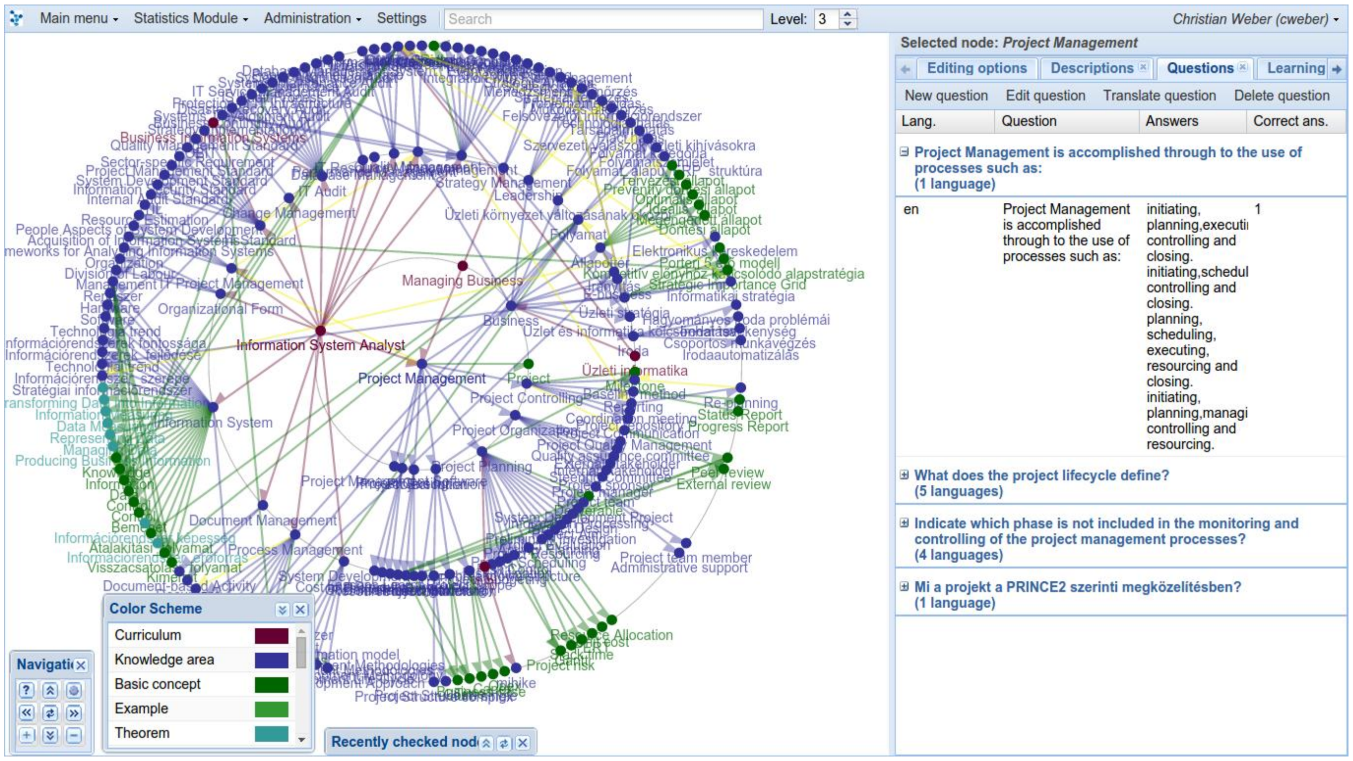
The pre-developed structure of classes and relations is fixed as the central and integral design of the system. A view of the system interface for administration is provided in Figure 3. The left area shows the visualization of the current ontology section in revision and the right area shows the question overview with editing options. Tabs give access to additional editing views, including the learning material management and interfaces to modify relations between nodes and node descriptions.

2.4 Adaptive Self-Assessment

To prepare an online self-assessment test, the system has to load the relevant educational areas from the domain ontology and extract the questions and relations of the filtered knowledge areas.

The internal test algorithm makes use of two assumptions:

- **Knowledge-area ordering:** As the main knowledge areas are connected through "requires_knowledge_of" and "part_of" relations, every path, starting with the start-element, will develop on average from general concepts to detailed concepts - given that the concept groups in the test definition are also selected and ordered to lead from general to more detailed groups.
- **Knowledge evaluation dependency:** If a person, taking the test, fails on general concepts he or she will potentially also fail on more detailed concepts. Further, if a high number of detailed concepts are failed, the parent knowledge isn't sufficiently covered and will be derived as failed, too.



Christian Weber (cweber)

Selected node: *Project Management*

Editing options | Descriptions | Questions | Learning

New question | Edit question | Translate question | Delete question

Lang.	Question	Answers	Correct ans.
Project Management is accomplished through to the use of processes such as: (1 language)			
en	Project Management is accomplished through to the use of processes such as:	initiating, planning,executi controlling and closing. initiating,schedul controlling and closing. planning, scheduling, executing, resourcing and closing. initiating, planning,managi controlling and resourcing.	1
What does the project lifecycle define? (5 languages)			
Indicate which phase is not included in the monitoring and controlling of the project management processes? (4 languages)			
Mi a projekt a PRINCE2 szerinti megközelítésben? (1 language)			

Figure 3: The main ontology maintenance and administration interface, showing a part of the domain ontology.

The filtering is done based on the selection of a tutor, acting as an expert for the target educational area. The tutor chooses related areas, which are then created as a Test Definition, containing Concept Groups, as described in section 2.3. The system then uses the test definition as a filtering list to extract knowledge areas. After the extraction, the structure is cached as a directed graph, while the top element of the initial concept group is set as a start element. Beginning with the start-element, the test will move then through the graph, while administering the questions connected to knowledge areas and knowledge elements.

The loading of knowledge-elements follows three steps:

1. Each type of relation between two knowledge-elements implements a direction for the connection. Assuming the system loads all relations, starting with the start-element and ending on a knowledge-element, this creates a two level structure where the start-node is a parent-element and all related, loaded elements are child-elements, as seen below in Figure 4.
2. The loading algorithm then selects one child-element and assumes it as a start-element and repeat the loading process of knowledge-elements.
3. When no knowledge-elements for a parent-element could be loaded, the sub-process stops. When all sub-processes have stopped, the knowledge structure is fully covered.

The test algorithm will now activate the child knowledge areas of the start element and select the first knowledge area to the left and draw a random question from the selected knowledge area. If the learner fails the question, the algorithm will mark the element as failed and selects the next knowledge area from the same level. If the learner's answer is correct, the system will activate the child elements of the current node and draw a random question from the first left child.

Based on the tree shaped knowledge structure, the assessment now follows these steps to run the self-assessment, supported by the extracted knowledge structure:

1. Starting from the start-element, the test algorithm will activate the child knowledge-areas of the start element.
2. The algorithm now selects the first child-knowledge area and draws a random question out of the pool of available questions for this specific knowledge-element from the test bank.
3. If the learner fails the question, the algorithm will mark the element as failed and select the next knowledge area from the same level. If the learner's answer is correct, the system will activate the child elements of the current node and trigger the process for each child-element.

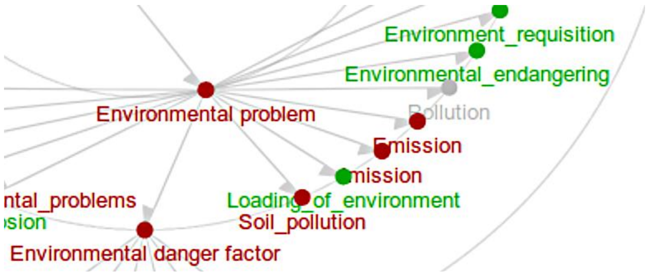
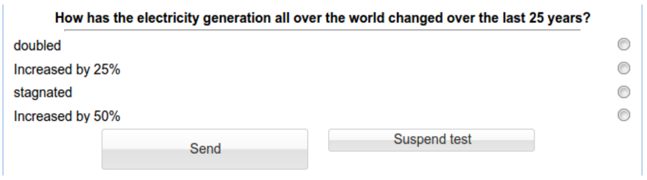


Figure 4: Excerpt from the sub-ontology visualization, with the visible parent-child relationship, as used in the data-loading for preparing the self-assessment.

An example question is shown below in Figure 5. Further following the testing algorithm, the system dives down within the domain ontology and triggers questions depending on the learner's answers and the extracted model of the relevant education. In this regards the Studio system adapts the test on the fly to the performance of the learner. Correlating to the idea of adaptation, the learner will later gain access to learning material for each mastered knowledge area. As the learner continues to use the self assessment to evaluate the personal knowledge, he or she will thus explore different areas of the target education, following their individual pace of learning.



How has the electricity generation all over the world changed over the last 25 years?

doubled ○
Increased by 25% ○
stagnated ○
Increased by 50% ○

Figure 5: Test interface with a random drawn test question.

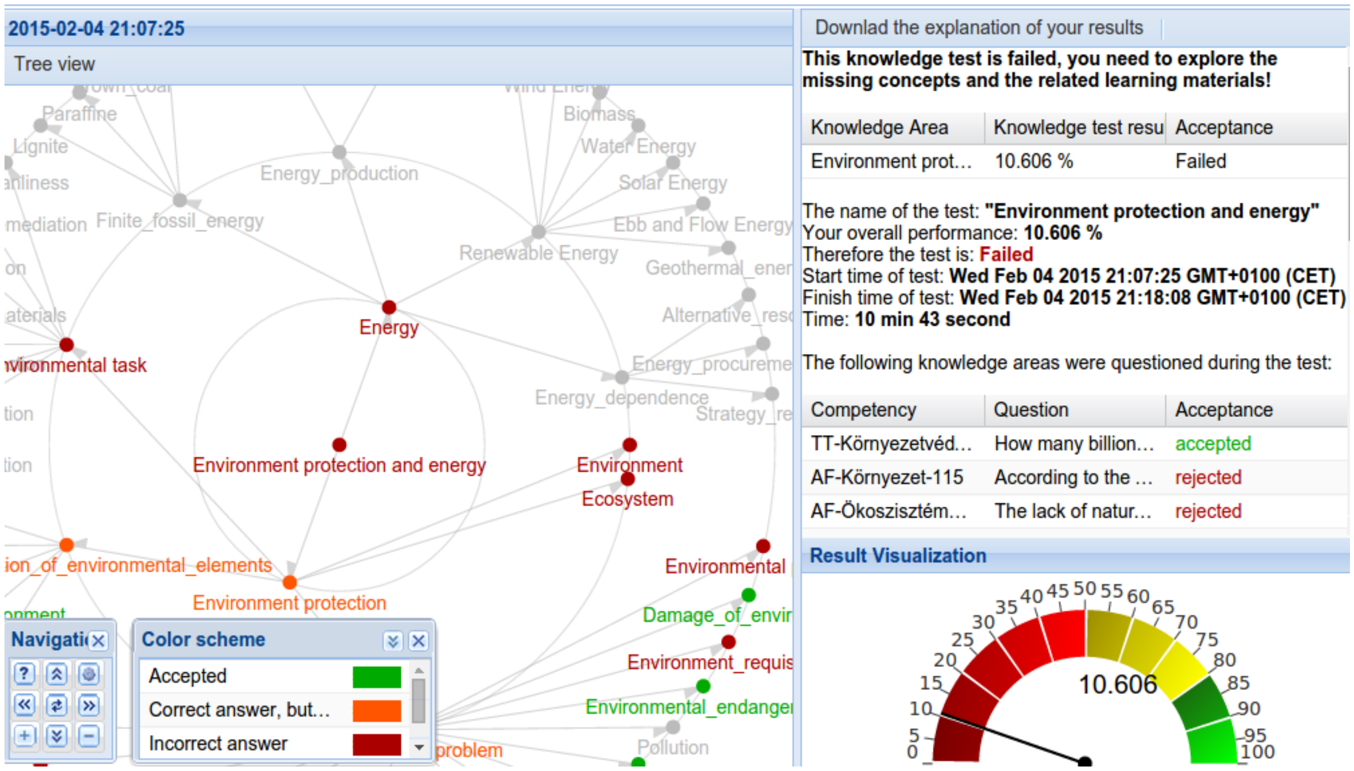
2.5 Test Feedback and Result Visualization

An important aspect of the system is the test feedback and evaluation interface. The educational feedback is one of the main enabler for the student to grasp the current state and extend of the personal education. The domain ontology models the structure and the dependencies of the educational domain, and the grouped test definition extracts the relevant knowledge for the target area or education. As such, the visualization of the ontology structure extracted for the test, together with the indication of correct and incorrect answers, represents a map of the knowledge of the learner.

Throughout each view onto the ontology, the system uses the same basic visualization, making use of the Sencha Ext JS JavaScript framework [18]. The visualization itself is a custom build, similar to the Ext JS graph function "Radar" and based on the idea of Ka-Ping, Fisher, Dhamija and Hearst [19]. All views are able to zoom in and out of the graph, move the current excerpt and offer a color code legend, explaining the meaning of the colored nodes. In comparison with state of the art, the interface offers no special grouping or additional visualization features like coding information into the size of nodes. Each interface offers an additional textual tree view to explore the knowledge-elements or concept groups in a hierarchical listing. This simple, straightforward approach for visualization correlates with the goal of a direct and easy to grasp feedback through interfaces which have a flat learning curve and enable to catch the functionality in a small amount of time.

While this simple visualization is sufficient for the reasonable amount of knowledge-elments within the result view, this alone is not suitable for the domain ontology administration interface, as seen in Figure 3. Here Studio realizes methodologies to filter and transform the data to visualize. To do so it makes use of two supporting mechanisms:

- The **maximum-level-selector** defines the maximum level the system extracts from the domain ontology for full screen visualization.
- In combination with the maximum level, the ontologist could select single elements within the domain ontology. This triggers an **on-demand re-extraction of the visualized data**, setting the selected knowledge-element as the centre element. The system then loads the connected nodes, based on their relations into the orientation circles till the maximum defined level is reached. More details about the transformation are in [19].



2015-02-04 21:07:25

Tree view

Color scheme

Accepted	🟩
Correct answer, but...	🟧
Incorrect answer	🟥

Download the explanation of your results

This knowledge test is failed, you need to explore the missing concepts and the related learning materials!

Knowledge Area	Knowledge test resu	Acceptance
Environment prot...	10.606 %	Failed

The name of the test: **"Environment protection and energy"**
Your overall performance: **10.606 %**
Therefore the test is: **Failed**
Start time of test: **Wed Feb 04 2015 21:07:25 GMT+0100 (CET)**
Finish time of test: **Wed Feb 04 2015 21:18:08 GMT+0100 (CET)**
Time: **10 min 43 second**

The following knowledge areas were questioned during the test:

Competency	Question	Acceptance
TT-Környezetvéd...	How many billion...	accepted
AF-Környezet-115	According to the ...	rejected
AF-Ökoszisztém...	The lack of natur...	rejected

Result Visualization

Figure 6: Result visualization as educational feedback for the learner.

Together, this selection and transformation mechanism enables the fluent navigation within the complete domain ontology structure, while re-using the same visualization interface.

Figure 6 shows the main view of the result interface. The left area shows the sub-ontology extracted for the test, while the colored nodes represent the answers to the administered questions. A red node visualizes wrong answers, while orange nodes are rejected nodes with correct answers but with an insufficient number of correctly answered child nodes, indicating a lack of the underlying knowledge. Green nodes represent accepted nodes with correct answers and a sufficient amount of correctly answered questions for child nodes. Grey nodes are not administered nodes, which were not yet reached by the learner, as higher order nodes had no adequate acceptance.

Even though the target of the system is not a strict evaluation in number, the evaluation of the percentage of solved and accepted knowledge elements helps the learner to track the personal progress and could additionally be saved as a report for further consultation. Besides providing an overview of the self-assessment result, the result interface gives access to the integrated learning material. For every passed node, the learner can now open the correlated material and intensify the knowledge for successful tested areas.

Retaking the test in cycles of testing and learning, while adapting the educational interaction, is the central concept of the Studio approach for self-assessment. As a consequence the system will not disclose the right answers to questions or learning material for not yet administered knowledge areas, to promote an individual reflection on the educational content outside of a flat memorization of content.

3. SYSTEM EVALUATION

The system has been used, extended and evaluated in a number of European and nationally funded research projects, including applications in business process management and innovation-transfer [20], medical education [21] and job market competency matching [22].

Currently the system is being evaluated based on a running study with 200 university students in the field of business informatics. The study will conclude on two current research streams which are improving the systems testing and analysis capability. The first direction looks into potentials for the integration of learning styles into adaptive learning systems to offer valuable advice and instructions to teachers and students [23]. Within the second direction the question is challenged on how to adapt the presented self-assessment further towards the performance of the students, based on extracting assessment paths from the knowledge structure [24].

For each running test, Studio collects basic quantitative data about the number of assigned questions, how often tests are taken and how many students open which test and when. This is completed by qualitative measures, collecting which questions and knowledge elements the students passed or failed. To conclude further on the mechanisms and impacts of Studio within the current study, a new logging system was developed, collecting the interaction with the system and detailed information about the feedback as detailed events.

Each event stores information about the system in 7 dimensions, as described in Table 1 below:

Table 1: Event blueprint to store events concerning system interaction.

Attribute	Description
Event description code	Which type of event and what factors are relevant.
Location code	On which part of the assessment-process or interface the event has occurred.
Session identifier	Each access of the system is one session for one user.
Numerical value storage	Multi-purpose field, filled depending on the event type.
String value storage	Multi-purpose field, filled depending on the event type.
Event-time	The time of the start of the event.
Item reference	A unique reference code, identifying the correlated item within the ontology. E.g. a question or a knowledge-element ID.

All events are stored in order of their occurrence, so if no explicit end event is defined, the next event for the same session and user is acting as the implicit end date. Extending the existing storage of information within Studio, the new logging system stores the additional events, as shown in Table 2 below:

Table 2: Assessment events and descriptions.

Event type	Description
START_TEST	Marks the start of a test.
END_TEST	Marks the end of a test.
OPEN_WELCOME_LM	The user opened the welcome page.
OPEN_LM_BLOCK	The student opened a learning material block on the test interface.
OPEN_LM	The student opened the learning material tab on the test interface.
RATE_LM	The student rated the learning material.
CHECK_RESULT	The student opened a result page.
CONTINUE_TEST	The student submitted an answer.
FINISH_TEST	The test has been finished.
SUSPEND_TEST	The user suspended the test.
RESUME_TEST	The user has restarted a previously suspended test.
SELECT_TEST_ALGO-RITHM	The algorithm used to actually test the student is selected.

TEST_ALGORITHM_-EVENT	The behavior of the current test algorithm changes, e.g. entering another stage of testing.
ASK_TESTQUESTION	Sends out a test question to the user to answer.
STUDIO_LOGOUT	The user logs out of the Studio system.

To store the events, the system implements an additional logging database, splitting the concepts of the logging to a star-schema for efficient extraction, transformation and loading. The logging system is modular and easy to extend with new concepts and easy to attach to potential event positions within the Studio runtime. Together with the existing logging of the assessment evaluation feedback, this new extension tracks the exploration of the sub-ontology within the assessment and enriches the feedback data with context information of the students behavior on the system.

4. NEXT STEPS

The domain ontology offers a functional and semantically rich core for supporting learning and education. Yet not all the semantic potentials are fully leveraged to support and test the learner's progress. The "requires_knowledge_of" relation-requirement is a potential start-concept to model sub-areas as groups which together compose the dependency. This could act as an additional input for the assessment, where the system derives more complex decision how to further explore the related parts of the structure [25]. This could also be visualized, enabling the learner to grasp the personal knowledge as a visible group of concepts.

Besides giving colors to the different types of relations, the visualizing of edges between knowledge areas is yet unfiltered, offering no further support for navigation. A next stage of implementation could be the introduction of a visual ordering and grouping of knowledge areas and relations. Underlying relations of sub-nodes could be interpreted visually through the thickness of relations between nodes, easing the perception of complex parts of the domain ontology, especially within administration and maintenance tasks.

The feedback of the current evaluation study of Studio will provide additional insights into the usage of the system by the students. Based on this new data it is possible to mine profiles over time on the knowledge structure. One major application is here the creation of behavior profiles, as proposed in [23].

5. REFERENCES

[1] D. Dunning, C. Heath, and J. M. Suls, "Flawed self-assessment implications for health, education, and the workplace," *Psychological science in the public interest*, vol. 5, no. 3, pp. 69–106, 2004.

[2] D. Jonassen, "Reconciling a Human Cognitive Architecture," in *Constructivist Instruction: Success Or Failure?*, S. Tobias and T. M. Duffy, Eds. Routledge, 2009, pp. 13–33.

[3] R. Vas, "Educational Ontology and Knowledge Testing," *Electronic Journal of Knowledge Management*, vol. 5, no. 1, pp. 123 – 130, 2007.

[4] M. Y. Dahab, H. A. Hassan, and A. Rafea, "TextOntoEx: Automatic ontology construction from natural English text," *Expert Systems with Applications*, vol. 34, no. 2, pp. 1474–1480, Feb. 2008.

[5] S.-H. Wu and W.-L. Hsu, "SOAT: a semi-automatic domain ontology acquisition tool from Chinese corpus," in *Proceedings of the 19th international conference on Computational linguistics-Volume 2*, 2002, pp. 1–5.

[6] M. Missikoff, R. Navigli, and P. Velardi, "The Usable Ontology: An Environment for Building and Assessing a Domain Ontology," in *The Semantic Web — ISWC 2002*, I. Horrocks and J. Hendler, Eds. Springer Berlin Heidelberg, 2002, pp. 39–53.

[7] T. A. Gavrilova and I. A. Leshcheva, "Ontology design and individual cognitive peculiarities: A pilot study," *Expert Systems with Applications*, vol. 42, no. 8, pp. 3883–3892, May 2015.

[8] S. Sosnovsky and T. Gavrilova, "Development of Educational Ontology for C-programming," 2006.

[9] V. Psyché, J. Bourdeau, R. Nkambou, and R. Mizoguchi, "Making learning design standards work with an ontology of educational theories," in *12th Artificial Intelligence in Education (AIED2005)*, 2005, pp. 539–546.

[10] T. Nodenot, C. Marquesuzaá, P. Laforcade, and C. Sallaberry, "Model Based Engineering of Learning Situations for Adaptive Web Based Educational Systems," in *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, New York, NY, USA, 2004, pp. 94–103.

[11] A. Bouzeghoub, C. Carpentier, B. Defude, and F. Duitama, "A model of reusable educational components for the generation of adaptive courses," in *Proc. First International Workshop on Semantic Web for Web-Based Learning in conjunction with CAISE*, 2003, vol. 3.

[12] W. Chen and R. Mizoguchi, "Learner model ontology and learner model agent," *Cognitive Support for Learning-Imagining the Unknown*, pp. 189–200, 2004.

[13] G. Neusch and A. Gábor, "PROKEX – INTEGRATED PLATFORM FOR PROCESS-BASED KNOWLEDGE EXTRACTION," *ICERI2014 Proceedings*, pp. 3972–3977, 2014.

[14] H. L. Roediger III, "Relativity of Remembering: Why the Laws of Memory Vanished," *Annual Review of Psychology*, vol. 59, no. 1, pp. 225–254, 2008.

[15] J. D. Fletcher, "Evidence for learning from technology-assisted instruction," *Technology applications in education: A learning view*, pp. 79–99, 2003.

[16] J.-C. Falmagne, M. Koppen, M. Villano, J.-P. Doignon, and L. Johannesen, "Introduction to knowledge spaces: How to build, test, and search them," *Psychological Review*, vol. 97(2), pp. 201–224, 1990.

[17] F. Neuhaus, E. Florescu, A. Galton, M. Gruninger, N. Guarino, L. Obrst, A. Sanchez, A. Vizedom, P. Yim, and B. Smith, "Creating the ontologists of the future," *Applied Ontology*, vol. 6, no. 1, pp. 91–98, 2011.

[18] "Ext JS API." [Online]. Available: http://www.objis.com/formationextjs/lib/extjs-4.0.0/docs/index.html. [Accessed: 04-May-2015].

[19] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst, "Animated exploration of dynamic graphs with radial layout," in *IEEE Symposium on Information Visualization, 2001. INFOVIS 2001*, 2001, pp. 43–50.

[20] M. Arru, “Application of Process Ontology to improve the funding allocation process at the European Institute of Innovation and Technology.,” in *3rd International Conference on Electronic Government and the Information Systems Perspective (EGOVIS).*, Munich, Germany, 2014.

[21] M., M., Ansari, F., Dornhöfer Khobreh and M. Fathi, “An ontology-based Recommender System to Support Nursing Education and Training,” in *LWA 2013*, 2013.

[22] V. Castello, L. Mahajan, E. Flores, M. Gabor, G. Neusch, I. B. Szabó, J. G. Caballero, L. Vettraino, J. M. Luna, C. Blackburn, and F. J. Ramos, “THE SKILL MATCH CHALLENGE. EVIDENCES FROM THE SMART PROJECT,” *ICERI2014 Proceedings*, pp. 1182–1189, 2014.

[23] H. M. Truong, “Integrating learning styles and adaptive e-learning system: Current developments, problems and opportunities,” *Computers in Human Behavior*, 2015.

[24] C. Weber, “Enabling a Context Aware Knowledge-Intense Computerized Adaptive Test through Complex Event Processing,” *Journal of the Sientific and Educational on Forum on Business Information Systems*, vol. 9, no. 9, pp. 66–74, 2014.

[25] C. Weber and R. Vas, “Extending Computerized Adaptive Testing to Multiple Objectives: Envisioned on a Case from the Health Care,” in *Electronic Government and the Information Systems Perspective*, vol. 8650, A. Kő and E. Francesconi, Eds. Springer International Publishing, 2014, pp. 148–162.

Graph Analysis of Student Model Networks

Julio Guerra
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA, USA
jdg60@pitt.edu

Yun Huang
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA, USA
yuh43@pitt.edu

Roya Hosseini
Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA, USA
roh38@pitt.edu

Peter Brusilovsky
School of Information Sciences
University of Pittsburgh
Pittsburgh, PA, USA
peterb@pitt.edu

ABSTRACT
This paper explores the feasibility of a graph-based approach to model student knowledge in the domain of programming. The key idea of this approach is that programming concepts are truly learned not in isolation, but rather in combination with other concepts. Following this idea, we represent a student model as a graph where links are gradually added when the student's ability to work with connected pairs of concepts in the same context is confirmed. We also hypothesize that with this graph-based approach a number of traditional graph metrics could be used to better measure student knowledge than using more traditional scalar models of student knowledge. To collect some early evidence in favor of this idea, we used data from several classroom studies to correlate graph metrics with various performance and motivation metrics.

1. INTRODUCTION
Student modeling is widely used in adaptive educational systems and tutoring systems to keep track of student knowledge, detect misconceptions, provide targeted support and give feedback to the student [2]. The most typical *overlay* student model dynamically represents the inferred knowledge level of the student for each *knowledge element* (KE) (also called *knowledge component* or KC) defined in a domain model. These knowledge levels are computed as the student answers questions or solves problems that are mapped to the domain KEs. Student models are frequently built over *networked* domain models where KEs are connected by prerequisite, is-a, and other ontological relationships that are used to propagate the knowledge levels and produce a more accurate representation of the knowledge of the learner. Since these connections belong to domain models, they stay the same for all students and at all times. In this work we explore the idea that it might be beneficial for a student model to include connections between domain KEs that represent some aspects of individual student knowledge rather than domain knowledge. This idea is motivated by the recognition that the mastery in many domains is reached as the student practices connecting different KEs, i.e., each KE is practiced in conjunction with other KEs. To address this, we build a model represented as a network of KEs that get progressively connected as the student successfully works with problems and assessment items containing multiple KEs. As the student succeeds in more diverse items mapped to different KEs, her model gets better connected.

To explore the value of this graph-based representation of student knowledge, we compute different graph metrics (e.g., density, diameter) for each student and analyze them in relation to student performance metrics and attitudinal orientations drawn from a motivational theory. This analysis was performed using data collected from 3 cohorts of a Java programming course using the same system and the same content materials. In the remaining part of the paper, we describe related work, introduce and illustrate the suggested approach, describe graph and performance metrics, and report the results of the correlation analysis.

2. RELATED WORK
Graph representation of student activity is not new. The 2014 version of the Graph-Based Educational Data Mining Workshop [1] contains two broad types of related work: the analysis of the networking interaction among students, for example work on social capital [14] and social networking in MOOCs [3, 12]; and analyses of learning paths over graph representation of student traces while performing activities in the system [1, 5]. Our work fits in the second type since we model traces of each student interacting with the system. However, our approach is different as it attempts to combine an underlying conceptual model with the traces of the student learning.

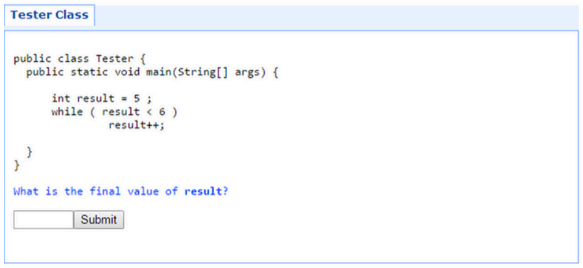
[1] `http://ceur-ws.org/Vol-1183/gedm2014_proceedings.pdf`

A considerable amount of work focused on graph representation of domain models that serve as a basis for overlay student models. The majority of this work focused on constructing the prerequisite relationships between domain knowledge components (concept, skills) [6, 13]. In this case links established between a pair of concepts represent prerequisite - outcome relationship. Another considerable stream of work explored the use of formal ontologies with such relationships as is-a and part-of for connecting domain knowledge components [7]. Ontological representation, in turn, relates to another stream of work that applies graph techniques to structural knowledge representation, for example by analyzing the network properties of ontologies [9].

The research on graph-based domain models also leads to a stream of work on using Bayesian networks to model the relationships between domain concepts for knowledge propagation in the process of student modeling [15, 4]. Yet, in both cases mentioned above links between knowledge components were not parts of individual student model, but either parts of the domain model or student modeling process and thus remain the same for all students. In contrast, the approach suggested in this paper adds links between knowledge components to *individual student models* to express combinations of knowledge components that the given student explored in a problem solving or assessment process. This approach is motivated by our belief that in the programming domain, student knowledge is more effectively modeled by capturing student progress when students needed to apply multiple concepts at the same time.

3. THE APPROACH

The idea behind our approach is that knowledge is likely to be stronger for concepts which are practiced together with a larger variety of other concepts. We hypothesize, for example, that a student who solves exercises, in which the concept *for-loop* is used with *post-incremental operator* and *post-decremental operator* will have a better understanding of *for-loop* than another student who practices (even the same amount of times) the *for loops* concept in a more narrow context, i.e., only with *post-incremental operator*. To represent our approach, for each student we build a graph-based student model as a network of concepts where the edges are created as the student succeeds in exercises containing both concepts to be connected. The Domain Model defining the concept space and the mapping between the concepts and programming exercises is explained in the next section. The weight of the edges in the graph is computed as the overall success rate on exercises performed by the student which contain the pair of concepts. Pairs of concepts that do not co-occur in exercises succeeded by the student are not connected in her graph. In this representation, highly connected nodes are concepts successfully practiced with different other concepts. We also compute a measure of *weight* for each node by taking the average weight among edges connecting the node. This measure of the success rate on concepts favors exercises that connect more concepts because each exercise containing n concepts produce or affects $n(n-1)/2$ edges. For example, a success on an exercise having 10 concepts contributes to 45 edges, but a successful attempt to an exercise connecting 5 concepts only contributes to 10 edges. We hypothesize that in a graph built following this approach, metrics like average degree, density, average path length, and average node weight can be good indicators of student knowledge compared to the amount of activities done or overall measures of assessment like success rate on exercises. We further explore these graph metrics in relation with motivational factors drawn from a learning motivation theory.



```
public class Tester {
  public static void main(String[] args) {

      int result = 5 ;
      while ( result < 6 )
              result++;

  }
}
```

What is the final value of result?

Figure 1: Exercise jwhile1

3.1 Domain Model and Content

Our content corpus is composed by a set of 112 interactive parameterized exercises (i.e., questions or problems) in the domain of Java programming from our system QuizJet [11]. Parameterized exercises are generated from a template by substituting a parameter variable with a randomly generated value. As a result each exercise can be attempted multiple times. To answer the exercise the student has to mentally execute a fragment of Java code to determine the value of a specific variable or the content printed on a console. When the student answers, the system evaluates the correctness, reports to the student whether the answer was correct or wrong, shows the correct response, and invites the student to "try again". As a result, students may still try the same exercises even after several correct attempts. An example of parameterized java exercise can be seen in Figure 1.

In order to find the concepts inside all of the exercises, we used a parser [10] that extracts concepts from the exercise's template code, analyzes its abstract syntax tree (AST), and maps the nodes of the AST (concepts extracted) to the nodes in a Java ontology [2]. This ontology is a hierarchy of programming concepts in the java domain and the parser uses only the concepts in the leaf nodes of the hierarchy.

In total there are 138 concepts extracted and mapped to QuizJet exercises. Examples of concepts are: *Int Data Type*, *Less Expression*, *Return Statement*, *For Statement*, *Subtract Expression*, *Constant*, *Constant Initialization Statement*, *If Statement*, *Array Data Type*, *Constructor Definition*, etc. We excluded 8 concepts which appear in all exercise templates (for example "Class Definition" or "Public Class Specifier" appear in the first line of all exercises). Each concept appears in one or more Java exercises. Each of the 112 exercises maps to 2 to 47 Java concepts. For example, the exercise "jwhile1", shown in Figure 1, is mapped to 5 concepts: *Int Data Type*, *Simple Assignment Expression*, *Less Expression*, *While Statement*, *Post Increment Expression*.

[2]`http://www.sis.pitt.edu/~paws/ont/java.owl`

3.2 Graph Metrics

To characterize the student knowledge graph we computed standard graph metrics listed below.

- **Graph Density** (density): the ratio of the number of edges and the number of possible edges.
- **Graph Diameter** (diameter): length of the longest shortest path between every pair of nodes.
- **Average Path Length** (avg.path.len): average among the shortest paths between all pairs of nodes.
- **Average Degree** (avg.degree): average among the degree of all nodes in an undirected graph.
- **Average Node Weight** (avg.node.weight): the weight of a node is the average of the weight of its edges. We then average the weights of all nodes in the graph.

3.3 Measures of Activity

To measure student activity so that it could be correlated with the graph metrics we collected and calculated the following success measures:

- **Correct Attempts to Exercises** (correct.attempts): total number of correct attempts to exercises. It includes repetition of exercises as well.
- **Distinct Correct Exercises** (dist.correct.attempts): number of distinct exercise attempted successfully.
- **Overall Success Rate** (success.rate): the number of correct attempts to exercises divided by the total number of attempts.
- **Average Success Rate on Concepts** (avg.concept.succ.rate): we compute the success rate of each concept as the average success rate of the exercises containing the concept. Then we average this among all concepts in the domain model.

3.4 Motivational Factors

We use the revised Achievement-Goal Orientation questionnaire [8] which contains 12 questions in a 7-point Likert scale. There are 3 questions for each of the 4 factors of the *Achievement-Goal Orientation framework*: Mastery - Approach, Mastery-Avoidance, Performance-Approach and Performance-Avoidance. **Mastery-Approach** goal orientation relates to intrinsic motivation: "I want to learn this because it is interesting for me", "I want to master this subject"; **Mastery-Avoidance** relates to the attitude of avoid to fail or avoid learning less than the minimum; **Performance -Approach** goal orientation stresses the idea of having a good performance and relates well with social comparison: "I want to perform good in this subject", "I want to be better than others here"; and **Performance-Avoidance** oriented students avoid to get lower grades or avoid to perform worse than other students. The goal orientation of a student helps to explain the behavior that the student exposes when facing difficulty, but does not label the final achievement of the student. For example, if a student is Mastery-Approach oriented, it does not necessarily mean that the student reached the mastery level of the skill or knowledge. In our case, we believe the achievement-goal orientation of the student can convey the tendency to pursue (or avoid) to solve more diverse (and more difficult) exercises, which contain more heterogeneous space of concepts, thus contribute to form better connected graphs.

Table 1: Correlation between activity measures and grade and between graph metrics and grade. * significance at 0.1, ** significance at 0.05.

Measure of Activity	Corr. Coeff.	Sig. (p)
Correct Attempts to Exercises	.046	.553
Distinct Correct Exercises	.114	.147
Overall Success Rate	.298	.000**
Avg. Success Rate on Concepts	.188	.016**

Graph Metric	Corr. Coeff.	Sig. (p)
Average Degree	.150	.055*
Graph Density	.102	.190
Graph Diameter	.147	.081
Average Path Length	.152	.052*
Average Node Weight	.201	.010**

4. EXPERIMENTS AND RESULTS

4.1 Dataset

We collected student data over three terms of a Java Programming course using the system: Fall 2013, Spring 2014, and Fall 2014. Since the system usage was not mandatory, we want to exclude students who just tried the system while likely using other activities (not captured by the system) for practicing Java programming. For this we looked at the distribution of distinct exercises attempted and we exclude all student below the 1st quartile (14.5 distinct exercises attempted). This left 83 students for our analysis. In total these students made 8,915 attempts to exercises. On average, students have attempted about 55 (Standard Deviation=22) distinct exercises while performing an average of 107 (SD=92) exercises attempts. On average, students have *covered* about 63 concepts with SD=25 (i.e., succeeded in at least one exercise containing the concept), and have covered about 773 concept pairs with SD=772 (i.e., succeeded in at least one exercise covering the concept pair.) The average success rate ($\frac{\#\text{correct attempts}}{\#\text{total attempts}}$) across students is about 69% (SD=11%).

4.2 Graph Metrics and Learning

We compare graph metrics (Avg. Degree, Graph Density, Graph Diameter, Avg. Path Length and Avg Node Weight) and measures of activity (Correct Attempts to Exercises, Distinct Correct Exercises, Overall Success Rate and Avg. Success Rate on Concepts) by computing the Kendall's τ_B correlation of these metrics with respect to the students' grade on the programming course. Results are displayed in Table 1.

Surprisingly, the plain **Overall Success Rate** (which does not consider concepts disaggregation, nor graph information) is better correlated with course grade than any other measure. Students who succeed more frequently, get in general better grades. Interestingly, both the **Average Success Rate on Concepts** and the **Average Node Weight** are both significantly correlated with grade. This last measure uses the graph information and presents a slightly better correlation than the former, which does not consider the graph information.

Among the other graph metrics, **Average Degree** and **Average Path Length** are marginally correlated with course

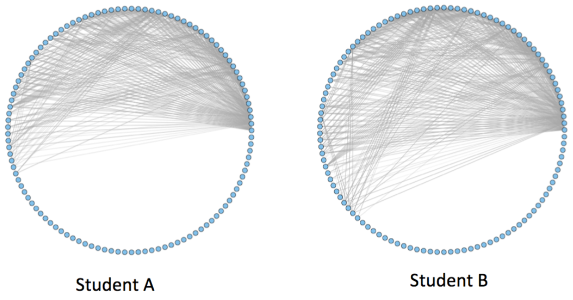


Figure 2: Graph representation of two students.

Table 2: Graph metrics, measures of activity and motivational scores of 2 students.

	Student A	Student B
Graph Density	0.077	0.094
Graph Diameter	2.85	2.00
Avg. Path Length	1.77	1.78
Avg. Degree	8.64	10.55
Avg. Node Weight	0.49	0.51
Correct Attempts	71	83
Dist. Correct Exercises	66	61
Overall Succ. Rate	0.82	0.76
Avg. Succ.Rate on Concepts	0.50	0.53
Mastery-Approach	0.83	0.78
Mastery-Avoidance	0.83	0.56
Performance-Approach	1.0	0.17
Performance-Avoidance	1.0	0.0
Grade (%)	100	97

grade (p values less than 0.1). Although this is a weak evidence, we believe that we are in the good track. A higher **Average Degree** means a better connected graph, thus it follows our idea that highly connected nodes signal more knowledge. **Average Path Length** is more difficult to interpret. A higher **Average Path Length** means a less connected graph (which contradicts our assumption), but also, it can express students reaching more "rear" concepts which appear in few more-difficult-exercises and generally have longer shortest paths. We think that further exploration of metrics among sub-graphs (e.g. a graph for an specific topic), and further refinement of the approach to build edges (e.g. connecting concepts that co-occur close to each other in the exercise) could help to clarify these results

Figure 2 shows the graphs of 2 students who have similar amount of distinct exercises solved correctly but present different graph metrics and motivational profile. See metrics in Table 2. Student B has more edges, lower diameter, higher density, higher degree, solved less questions more times. Student A presents a less connected graph although she he/she solved more distinct questions (66 compared to 61 on Student B). Student B has lower Mastery-Avoidance orientation score and lower Performance orientation scores than Student A, which could explain why Student B work result in a better connected graph. Analyses of Motivational factors are described in the following section.

4.3 Metrics and Motivation

We now explore the relationship between motivational factors and the graphs of the students. The idea is to see to which extent the motivational profile of the student explains the graph's shape. Step-wise regression models were used where the dependent variables are the graph metrics and the independent variables are the motivational factors. We found a significant model of the diameter of the graph ($R_2 = 0.161$, $F = 6.523$, $p = 0.006$) with the factors *Mastery-Avoidance* ($B = 0.952$, $p = 0.001$) and *Mastery-Approach* ($B = -0.938$, $p = 0.006$). Note the negative coefficient for Mastery-Approach and the positive coefficient for Mastery-Avoidance. As the Achievement-Goal Orientation framework suggests, Mastery-Approach oriented students are motivated to learn more, tend to explore more content and do not give up easily when facing difficulties; Mastery-Avoidance students, in the other hand, do not cope well with difficulties and tend to give up. Then, a possible explanation of the results is that, in one hand, students with higher Mastery-Approach orientation are more likely to solve difficult questions which connects more and more distant concepts which decreases the graph diameter; and on the other hand, Mastery-Avoidance students avoid difficult exercises containing many concepts, thus making less connections and producing graphs with higher diameters. Correlations between graph metrics and motivational factors confirmed the relation between Mastery-Avoidance and Graph Diameter (Kendall's $\tau_B = 0.197$, $p = 0.030$). Although these results are encouraging, they are not conclusive. For example, Mastery-Approach students might just do more work, not necessarily targeting difficult questions. More analysis is needed to deeply explore these issues.

5. DISCUSSIONS AND CONCLUSIONS

In this paper we proposed a novel approach to represent student model in the form of a dynamic graph of concepts that become connected when the student succeed in assessment item containing a pair of concepts to be connected. The idea behind this approach is to strengthen the model for those concepts that are applied in more different contexts, i.e., in assessment items containing other different concepts. We applied this approach to data of assessment items answered by real students and analyzed the graph properties comparing them to several performance measures such as course grade as well as motivational factors. Results showed that this idea is potentially a good indicator of knowledge of the students, but further refinement of the approach is needed. We used several measures of the built graphs as descriptors of student knowledge level, and we found that a metric aggregating the success rates of the edges to the level of concepts (nodes) is highly correlated to course grade, although it does not beat the plain overall success rate of the student in assessment items.

In the future work, we plan to repeat our analysis using more reliable approaches to construct the knowledge graph. One idea is to use rich information provided by the parser (mapping between exercises and concepts) to ensure that each new link connects concepts that interact considerably in the program code. This could be done by controlling the concepts proximity in the question code (e.g. only consider co-occurrence when concepts are close to each other in the parser tree.) Another approach to keep more reliable edges is to consider only a subset of important concepts for each problem using feature selection techniques. Also we plan to perform analyses of sub-graphs targeting specific "zones" of knowledge. For example, a partial graph with only concepts that belongs to a specific topic, or concepts that are prerequisites of a specific concept. Another interesting idea relates to recommendation of content: guide the student to questions that will connect the isolated parts of the knowledge graph or minimize the average path length of the graph. Along the same lines, the analysis of the graph shortest paths and overall connectivity can help in designing assessment items that better connect distant concepts.

6. REFERENCES

[1] N. Belacel, G. Durand, and F. Laplante. A binary integer programming model for global optimization of learning path discovery. In *Workshop on Graph-Based Educational Data Mining*.

[2] P. Brusilovsky and E. Millán. User models for adaptive hypermedia and adaptive educational systems. In P. Brusilovsky, A. Kobsa, and W. Nejdl, editors, *The Adaptive Web*, volume 4321 of *Lecture Notes in Computer Science*, chapter 1, pages 3–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[3] V. Catelé, D. Hicks, C. Lynch, and T. Barnes. Snag'em: Graph data mining for a social networking game. In *Workshop on Graph-Based Educational Data Mining*, volume 6, page 10.

[4] C. Conati, A. Gertner, and K. Vanlehn. Using bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction*, 12(4):371–417, 2002.

[5] R. Dekel and Y. Gal. On-line plan recognition in exploratory learning environments. In *Workshop on Graph-Based Educational Data Mining*.

[6] M. C. Desmarais and M. Gagnon. *Bayesian student models based on item to item knowledge structures*. Springer, 2006.

[7] P. Dolog, N. Henze, W. Nejdl, and M. Sintek. The personal reader: Personalizing and enriching learning resources using semantic web technologies. In P. De Bra and W. Nejdl, editors, *Adaptive Hypermedia and Adaptive Web-Based Systems*, volume 3137 of *Lecture Notes in Computer Science*, pages 85–94. Springer Berlin Heidelberg, 2004.

[8] A. J. Elliot and K. Murayama. On the measurement of achievement goals: Critique, illustration, and application. *Journal of Educational Psychology*, 100(3):613, 2008.

[9] B. Hoser, A. Hotho, R. Jäschke, C. Schmitz, and G. Stumme. *Semantic network analysis of ontologies*. Springer, 2006.

[10] R. Hosseini and P. Brusilovsky. Javaparser: A fine-grain concept indexing tool for java exercises. In *The First Workshop on AI-supported Education for Computer Science (AIEDCS 2013)*, pages 60–63, 2013.

[11] I.-H. Hsiao, S. Sosnovsky, and P. Brusilovsky. Guiding students to the right questions: adaptive navigation support in an e-learning system for java programming. *Journal of Computer Assisted Learning*, 26(4):270–283, 2010.

[12] S. Jiang, S. M. Fitzhugh, and M. Warschauer. Social positioning and performance in moocs. In *Workshop on Graph-Based Educational Data Mining*, page 14.

[13] T. Käser, S. Klingler, A. G. Schwing, and M. Gross. Beyond knowledge tracing: Modeling skill topologies with bayesian networks. In *Intelligent Tutoring Systems*, pages 188–198. Springer, 2014.

[14] V. Kovanovic, S. Joksimovic, D. Gasevic, and M. Hatala. What is the source of social capital? the association between social network position and social presence in communities of inquiry. 2014.

[15] E. Millán, T. Loboda, and J. L. Pérez-de-la Cruz. Bayesian networks for student model engineering. *Computers & Education*, 55(4):1663–1683, 2010.

Graph-based Modelling of Students' Interaction Data from Exploratory Learning Environments

Alexandra Poulovassilis
London Knowledge Lab
Birkbeck, Univ. of London
ap@dcs.bbk.ac.uk

Sergio Gutierrez-Santos
London Knowledge Lab
Birkbeck, Univ. of London
sergut@dcs.bbk.ac.uk

Manolis Mavrikis
London Knowledge Lab
UCL Institute of Education
m.mavrikis@lkl.ac.uk

ABSTRACT

Students' interaction data from learning environments has an inherent temporal dimension, with successive events being related through the "next event" relationship. Exploratory learning environments (ELEs), in particular, can generate very large volumes of such data, making their interpretation a challenging task. Using two mathematical microworlds as exemplars, we illustrate how modelling students' event-based interaction data as a graph can open up new querying and analysis opportunities. We demonstrate the possibilities that graph-based modelling can provide for querying and analysing the data, enabling investigation of student-system interactions and leading to the improvement of future versions of the ELEs under investigation.

Keywords

Exploratory Learning Environments, Interaction Data, Graph Modelling

1. INTRODUCTION

Much recent research has focussed on *Exploratory Learning Environments* (ELEs) which encourage students' open-ended interaction within a knowledge domain, coupled with intelligent techniques that aim to provide pedagogical support to ensure students' productive interaction [9]. The data gathered from students' interactions with such ELEs provides a rich source of information for both pedagogical and technical research, to help understand how students are using the ELE and how the intelligent support that it provides may be enhanced to better support students' learning.

In this paper, we consider how modelling students' event-based interaction data as a *graph* makes possible graph-based queries and analyses that can provide insights into the ways that students are using the affordances of the system and the effects of system interventions on students' behaviour. Our case studies are two intelligent ELEs: the MiGen system, that aims to foster 11-14 year old students' learning of algebraic generalisation [15]; and the iTalk2Learn system that aims to support 8-10 year old students' learning of fractions [7]. Both systems provide students with mathematical microworlds in which they undertake construction tasks: in MiGen creating 2-dimensional tiled models using a tool called *eXpresser* and in iTalk2learn creating fractions using the *FractionsLab* tool. In eXpresser, tasks typically require the construction of several models, moving from specific models involving specific numeric values to a general model involving the use of one or more variables; in parallel, students are asked to formulate algebraic rules specifying the number of tiles of each colour that are needed to fully colour their models. In FractionsLab, tasks require the construction, comparison and manipulation of fractions, and students are encouraged to talk aloud about aspects of their constructions, such as whether two fractions are equivalent.

Both systems include intelligent components that provide different levels of feedback to students, ranging from unsolicited prompts and nudges, to low-interruption feedback that students can choose to view if they wish. The aim of this feedback is to balance students' freedom to explore while at the same time providing sufficient support to ensure that learning is being achieved [9]. The intelligent support is designed through detailed cognitive task analysis and Wizard-of-Oz studies [13], and it relies on meaningful *indicators* being detected as students are undertaking construction tasks. Examples of such indicators in MiGen are 'student has made a building block' (part of a model), 'student has unlocked a number' (i.e. has created a variable), 'student has unlocked too many numbers for this task'; while examples of such indicators in FractionsLab are 'student has created a fraction', 'student has changed a fraction' (numerator or denominator), 'student has released a fraction' (i.e. has finished changing it).

Teacher Assistance tools can subscribe to receive real-time information relating to occurrences of indicators for each student, and can present aspects of this information visually to the teacher [8]. Indicators are either *task independent* (TI) or *task dependent* (TD). The former refer to aspects of the student's interaction that are related to the microworld itself and do not depend on the specific task the student is working on, while the latter require knowledge of the task the student is working on, may relate to combinations of student actions, and their detection requires intelligent reasoning to be applied (a mixture of case-based, rule-based and probablistic techniques). Detailed discussions of MiGen's TI and TD indicators and how the latter are inferred may be found in [8].

In this paper we explore how graph-based representation of event-based interaction data arising from ELEs such as MiGen and FractionsLab can aid in the querying and analysis of such data, with the aim of exploring both the behaviours of the students in undertaking the exploratory learning tasks set and the effectiveness of the intelligent support being provided by the system to the students. Data relating to learning environments has often been modelled as a graph in previous work, for example in [10] for providing support to moderators in e-discussion environments; in [16, 18] for supporting learning of argumentation; in [17] for modelling data and metadata relating to episodes of work and learning in a lifelong learning setting; in [1] for learning path discovery as students "navigate" through learning objects; in [3] for recognising students' activity planning in ELEs; and in [23] for gaining better understanding of learners' interactions and ties in professional networks.

Previous work that is close to ours is the work on interaction networks and hint generation [6, 21, 20, 4, 5], in which the graphs used consist of nodes representing states within a problem-solving space and edges representing students' actions in transitioning between states. This approach targets learning environments where students are required to select and apply rules, and the interaction network aims to represent concisely information relating to students' problem-solving sequences in moving from state to state. Our focus here differs from this in that we are using graphs to model fine-grained event-based interaction data arising from ELEs. In our graphs, nodes are used to represent indicator occurrences (i.e. events, not problem states) and edges between such nodes represent the "next event" relationship. Also, rather than using the information derived from querying and analysing this data to automatically generate hints, our focus is on investigating how students are using the system and the effects of the system's interventions in order to understand how students interact with the ELEs and improve their future versions.

2. GRAPH-BASED MODELLING

Figure 1 illustrates our Graph Data Model for ELE interaction data. We see two classes of nodes: Event — representing indicator occurrences; and EventType — representing different indicator types. The instances of the Event class are occurrences of indicators that are detected or generated by the system as each student undertakes a task. We see that instances of Event have several attributes: dateTime: the date and time of the indicator occurrence; userID: the student it relates to; sessionID: the class session that the student was participating in at the time; taskID: the taskID that the student was working on; and constrID: the construction that the student was working on[1].

[1]The model in Fig. 1 focusses on the interaction data. The full data relating to ELEs such as eXpresser and FractionsLab would also include classes relating to users, tasks, sessions and constructions; and attributes describing instances of these classes, such as a user's name and year-group, a task's name and description, a construction's content and description, and a session's description and duration.

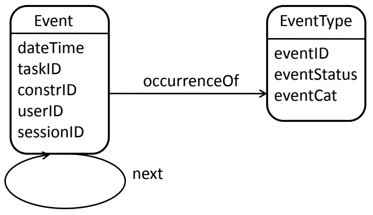


Figure 1: Core Graph Data Model

There is a relationship 'next' linking an instance of Event to the next Event that occurs for the same user, task and session. There is a relationship 'occurrenceOf' linking each instance of Event to an instance of the EventType class.

The instances of the EventType class include: startTask, endTask, numberCreated, numberUnlocked, unlockedNumberChanged, buildingBlockMade, correctModelRuleCreated, incorrectModelRuleCreated, interventionGenerated, interventionShown, in the case of eXpresser (see [8] for the full list); and startTask, endTask, fractionCreated, fractionChanged, fractionReleased, inverventionShown, in the case of FractionsLab.

We see that instances of the EventType class have several attributes, including:

- eventID: a unique numerical identifier for each type of indicator;
- eventStatus: this may be -1, 0 or 1, respectively stating that an occurrence of this type of indicator shows that the student is making negative, neutral or positive progress towards achieving the task goals; an additional status 2 is used for indicators relating to system interventions;
- eventCat: the category into which this indicator type falls; for example, startTask and endTask are task-related indicators; interventionGenerated and interventionShown are system-related ones; numberCreated, numberUnlocked, unlockedNumberChanged are number-related; and fractionCreated, fractionChanged, fractionReleased are fraction-related.

Figure 2 shows a fragment of MiGen interaction data conforming to this graph data model. Specifically, it relates to the interactions of user 5 as he/she is working on task 2 during session 9. The user makes three constructions during this task (with constrIDs 1, 2 and 3). The start and end of the task are delimited by an occurrence of the startTask and endTask indicator type, respectively — events 23041 and 33154. We see that the two events following 23041 relate to an intervention being generated and being shown to the student (this is likely to be because the student was inactive for over a minute after starting the task); following which, the student creates a number — event 24115.

There are additional attributes relating to events, not shown here for simplicity, capturing values relating to the student's

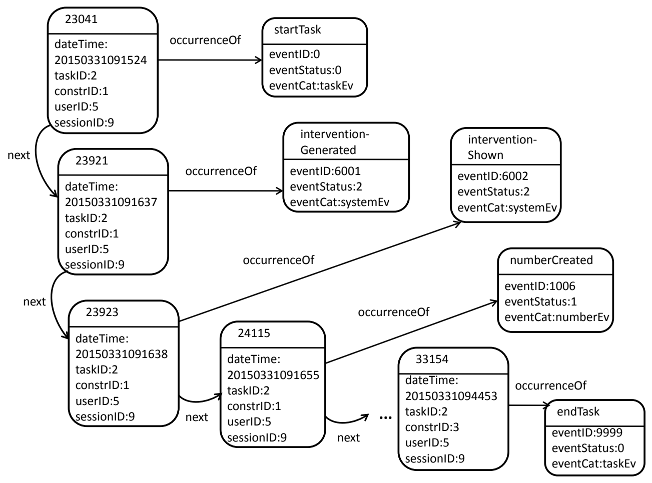


Figure 2: Fragment of Graph Data

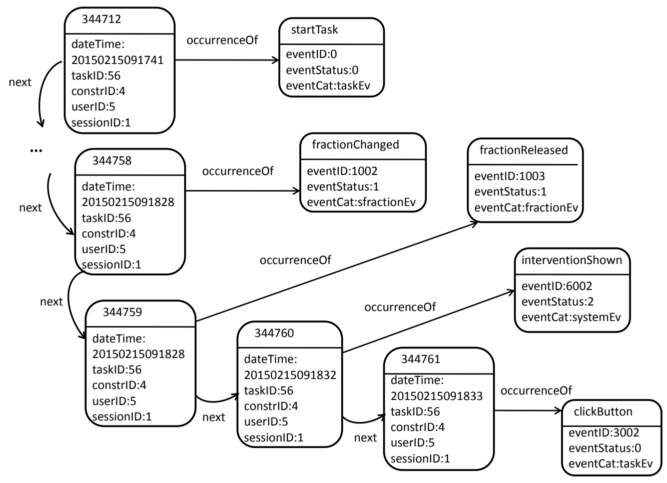


Figure 3: Fragment of Graph Data

constructions and information relating to the system's interventions. For example, for event 24115, the value of the number created, say 5; for event 23921, the feedback strategy used by the system to generate this intervention, say strategy 8; and for event 23923, the content of the message displayed to the user, say "How many green tiles do you need to make your pattern?" and whether this is a high-level interruption by the system or a low-level interruption that the student can choose to view or not. Such information can be captured through additional edges outgoing from an event instance to a literal-valued node: 24115 $\xrightarrow{value}$ 5, 23921 $\xrightarrow{strategy}$ 8, 23932 $\xrightarrow{message}$ "How many green tiles do you need to make your pattern?", 23932 $\xrightarrow{level}$ "high". Since graph data models are semi-structured (and graph data therefore does not need to strictly conform to a single schema), this kind of heterogeneity in the data is readily accommodated.

Figure 3 similarly shows a fragment of FractionsLab interaction data, relating to the interactions of user 5 working on task 56 during session 1. The user makes one construction during this task. We see events relating to the student changing and 'releasing' a fraction. Following which the system displays a message (in this case, it was a high-interruption message of encouragement "Great! Well Done").

We see from Figures 2 and 3 that the sub-graph induced by edges labelled 'next' consists of a set of paths, one path for each task undertaken by a specific user in a specific session. The entire graph is a DAG (directed acyclic graph): there are no cycles induced by the edges labelled 'next' since each links an earlier indicator occurrence to a later one; while the instances of EventType and other literal-valued nodes can have only incoming edges. The entire graph is also a bipartite graph, with the two parts comprising (i) the instances of Event, and (ii) the instances of EventType and the literal-valued nodes.

As a final observation, we note that Figures 1 – 3 adopt a "property graph" notation (e.g. as used in the Neo4J graph database, neo4j.com) in which nodes may have attributes. In a "classical" graph data model, each attribute of a node would be represented by an edge and its value by a literal-valued node. So, for example, the information that the taskID of event 23041 is 2 would be represented by an edge 23041 $\xrightarrow{taskID}$ 2. The query examples in the next section assume this "classical" graph representation.

3. GRAPH QUERIES AND ANALYSES

Because the sub-graph induced by edges labelled 'next' consists of a set of paths, the data readily lends itself to exploration using *conjunctive regular path (CRP) queries* [2]. A CRP query, Q, consisting of n conjuncts is of the form

$$(Z_1, \ldots, Z_m) \leftarrow (X_1, R_1, Y_1), \ldots, (X_n, R_n, Y_n)$$

where each X_i and Y_i is a variable or a constant, each Z_i is a variable that appears also in the right hand side of Q, and each R_i is a regular expression over the set of edge labels. In this context, a regular expression, R, has the following syntax:

$$R := \epsilon \mid a \mid \_ \mid (R_1.R_2) \mid (R_1|R_2) \mid R^* \mid R^+$$

where ϵ denotes the empty string, a denotes an edge label, $\_$ denotes the disjunction of all edge labels, and the operators have their usual meaning. The answer to a CRP query on a graph G is obtained by finding for each $1 \leq i \leq n$ a binary relation r_i over the scheme (X_i, Y_i), where there is a tuple (x, y) in r_i if and only if there is a path from x to y in G such that: $x = X_i$ if X_i is a constant; $y = Y_i$ if Y_i is a constant; and the concatenation of the edge labels in the path satisfies the regular expression R_i. The answer is then given by forming the natural join of the binary relations $r_1, \ldots, r_n$ and finally projecting on $Z_1, \ldots, Z_m$.

To illustrate, the following CRP query returns pairs of events x, y such that x is an intervention message shown to the user by the system and y indicates that the user's next action – in eXpresser – was to create a number (note, variables in queries are distinguished by an initial question mark):

```
(?X,?Y) <- (?X,occurrenceOf,interventionShown),
           (?X,next,?Y),
           (?Y,occurrenceOf,numberCreated)
```

The result would contain pairs such as (23923,24115) from Figure 2, demonstrating that there are indeed situations where an intervention message displayed by the MiGen system leads directly to the creation of a number by the student.

The following query returns pairs of events x, y such that that x is an intervention message shown to the user by the system and y is the user's next action; the type of y is also returned, through the variable ?Z:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionShown),
              (?X,next,?Y),
              (?Y,occurrenceOf,?Z)
```

The result would contain triples such as (23923,24115,numberCreated) from Figure 2 and (344760,344761,clickButton) from Figure 3, allowing researchers to see what types of events directly follow the display of an intervention message. This would allow the confirmation or contradiction of researchers' expectations regarding the immediate effect of intervention messages on students' behaviours.

Focussing for the rest of this section on the data in Figure 2, the following query returns pairs of events x, y such that x is any type of event and y indicates that the user's next action was to unlock a number; the type of x is also returned, through the variable ?Z:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,?Z),
              (?X,next,?Y),
              (?Y,occurrenceOf,numberUnlocked)
```

The result would allow researchers to see what types of events immediately precede the unlocking of a number (i.e. the creation of a variable). This would allow confirmation of researchers' expectations about the design of the MiGen system's intelligent support in guiding students towards generalising their models by changing a fixed number to an 'unlocked' one.

The following query returns pairs of events x, y such that that x is an intervention generated by the system and y is any subsequent event linked to x through a path comprising one or more 'next' edges; the type of y is also returned, through the variable ?Z:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionGenerated),
              (?X,next+,?Y),
              (?Y,occurrenceOf,?Z)
```

The result would contain triples such as (23921, 23923, interventionShown), (23921, 24115, numberCreated), ... (23921, 33154, endTask), allowing researchers to see what types of events directly or indirectly follow the display of an intervention message by the system. This would allow the confirmation or contradiction of researchers' expectations regarding the longer-term effect of intervention messages on students' behaviours.

We can modify the query to retain only pairs x, y that relate to the same construction:

```
(?X,?Y,?Z) <- (?X,occurrenceOf,interventionGenerated),
              (?X,constrID,?C), (?X,next+,?Y),
              (?Y,constrlID,?C), (?Y,occurrenceOf,?Z)
```

The result would contain triples such as
(23921, 23923, interventionShown),
(23921, 24115, numberCreated),
(23921, 24136, numberUnlocked),
(23921, 24189, unlockedNumberChanged),
relating to construction 1 made by user 5 during session 9 for task 2 (two more events — 24136 and 24189 — relating to construction 1 have been assumed here, in addition to 23923 amd 24115 shown in Figure 2, for illustrative purposes). The results would not contain (23921,33154,endTask), since event 33154 relates to construction 3.

To show more clearly the answers to the previous query in the form of possible event *paths*, we can use *extended regular path* (ERP) queries [11], in which a regular expression can be associated with a *path variable* and path variables can appear in the left-hand-side of queries. Thus, for example, the following query returns the possible paths from x to y:

```
(?X,?P,?Y,?Z) <-
      (?X,occurrenceOf,interventionGenerated),
      (?X,constrID,?C), (?X,next+:?P,?Y),
      (?Y,constrID,?C), (?Y,occurrenceOf,?Z)
```

The result would contain answers such as
(23921, [next], 23923, interventionShown),
(23921, [next, 23923, next], 24115, numberCreated),
(23921, [next, 23923, next, 24115, next], 24136, numberUnlocked),
(23921, [next, 23923, next, 24115, next, 24136, next], 24189, unlockedNumberChanged).

The use of the regular expressions `next` and `next+` in the previous queries matches precisely one edge labelled 'next', or any number of such edges (greater than or equal to 1), respectively. However, for finer control and ranking of query answers, it is possible to use *approximate* answering of CRP and ERP queries (see [11, 17]), in which edit operations such as insertion, deletion or substitution of an edge label can be applied to regular expressions.

For example, using the techniques described in [11, 17], the user can chose to allow the insertion of the label 'next' into a regular expression, at an edit cost of 1. Submitting then this query:

```
(?X,?P,?Y,?Z) <-
      (?X,occurrenceOf,interventionGenerated),
      (?X,constrID,?C), APPROX(?X,next:?P,?Y),
      (?Y,constrID,?C), (?Y,occurrenceOf,?Z)
```

would return first exact answers, such as
(23921, [next], 23923, interventionShown). The regular expression `next` in the conjunct `APPROX(?X,next:?P,?Y)` would then be automatically approximated to `next.next`, leading to answers such as

(23921, [next, 23923, next], 24115, numberCreated) at an edit distance of 1 from the original query. Following this, the regular expression `next.next` would be automatically approximated to `next.next.next`, leading to answers such as
(23921, [next, 23923, next, 24115, next], 24136, numberUnlocked)
at distance 2. This incremental return of paths of increasing length can continue for as long as the user wishes, and allows researchers to examine increasingly longer-term effects of intervention messages on students' behaviours. It would also be possible for users to specify from the outset a minimum and maximum edit distance to be used in approximating and evaluating the query, for example to request paths encompassing between 2 and 4 edges labelled 'next'.

Queries based on evaluating regular expressions over a graph-based representation of interaction data, such as those above, can aid in the exploration of students' behaviours as they are undertaking tasks using ELEs and the effectiveness of the intelligent support being provided by the ELE. The query processing techniques employed are based on incremental query evaluation algorithms which run in polynomial time with respect to the size of the database graph and the size of the query and which return answers in order of increasing edit distance [11]. A recent paper [19] gives details of an implementation, which is based on the construction of an automaton (NFA) for each query conjunct, the incremental construction of a weighted product automaton from each conjunct's automaton and the data graph, and the use of a ranked join to combine answers being incrementally produced from the evaluation of each conjunct. The paper also presents a performance study undertaken on two data sets — lifelong learning data and metadata [17] and YAGO [22]. The first of these has rather 'linear' data, similar to the interaction data discussed here, while the second has 'bushier' connectivity. Query performance is generally better for the former than the latter, and the paper discusses several possible approaches towards query optimisation.

In addition to evaluating queries over the interaction data, by representing the data in the form of a graph it is possible to apply graph structure analyses such as the following:

- *path finding and clustering*: this would be useful for determining patterns of interest across a whole dataset, or focussing on particular students, tasks or sessions c.f. [4];
- *average path length*: this would be useful for determining the amount of student activity (i.e. the number of indicator occurrences being generated per task) across a whole dataset, or focussing on particular students, tasks or sessions;
- *graph diameter*: to determine the greatest distance between any two nodes (which, due to the nature of the data, would be event type nodes); this would be an indication the most long-running and/or most intensive task(s);
- *degree centrality*: determining the in-degree centrality of event type nodes would identify key event types occurring in students' interactions; this analysis could be applied across a whole dataset, or focussing on particular students, tasks or sessions;
- nodes that have a high probability of being visited on a randomly chosen shortest path between two randomly chosen nodes have high *betweenness centrality*; determining this measure for pairs of event type nodes (ignoring the directionality of the 'occurrenceOf' edges) would identify event types that play key mediating roles between other event types.

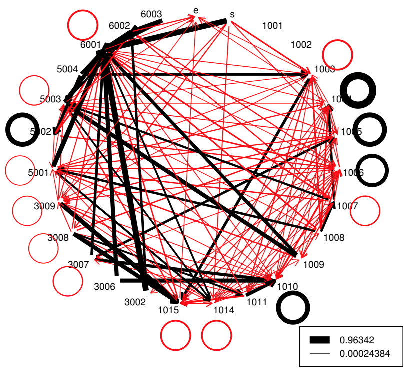


Figure 4: Transitions between Event Types

We have already undertaken some *ad hoc* analyses of interaction data arising from classroom sessions using ELEs. For example, Figure 4 shows the normalised incoming transitions for a 1-hour classroom session involving 22 students using MiGen (in the diagram, s denotes the 'startTask' and e the 'endTask' event types). Event types with an adjacent circle show transitions where this type of event occurs repeatedly in succession. The thickness of each arrow or circle indicates the value of the transition probability: the thicker the line, the higher the probability. Red (light grey) is used for probabilities < 0.2 and black for probabilities ≥ 0.2. We can observe a black arrow $3007 \rightarrow 1005$, indicating transitions from events of type 3007 (detection by the system that the student has made an implausible building block for this task) to events of type 1005 (modification of a rule by the student). Such an observation raises a hypothesis for more detailed analysis or further student observation, namely: "does the construction of an incorrect building block lead students to self-correct their rules?". Developing a better understanding of such complex interaction can lead to improvement of the system. For this particular example, we designed a new prompt that suggests to students to first consider the building block against the given task before proceeding unnecessarily in correcting their rules. More examples of such *ad hoc* analyses are given in [14]. Representing the interaction data in graph form will allow more systematic, flexible and scalable application of graph-structure algorithms such as those identified above.

4. CONCLUSIONS AND FUTURE WORK

We have presented a graph model for representing event-based interaction data arising from Exploratory Learning Environments, drawing on the data generated when students undertake exploratory learning tasks with the eXpresser and

FractionsLab microworlds. Although developed in the context of these systems, the model is a very general one and can easily be used or extended to model similar data from other ELEs.

We have explored the possibilities that evaluating regular path queries over this graph-based representation might provide for exploring the behaviours of students as they are working in the ELE and the effectiveness of the intelligent support that it provides to them. We have also identified additional graph algorithms that may yield further insights about learners, tasks and significant indicators.

Planned worked includes transformation and uploading of the interaction data sets gathered during trials and full classroom sessions of the two systems into an industrial-strength graph database such as Neo4J, following the graph model presented in Section 2; followed by the design, implementation and evaluation of meaningful queries, analyses and visualisations over the graph data, building on the work presented in Section 3. Equipped with an appropriate user interface, educational researchers, designers or even teachers with less technical expertise could in this way explore the data from their perspective. This has the potential to lead to an improved understanding of interaction in this context and to feed back to the design of the ELEs. We see this approach very much in the spirit of "polyglot persistence" (i.e. using different data storage methods to address different data manipulation problems), and hence being used in conjunction with other EDM resources such as DataShop [12]. Another direction of research is investigation of how the flexible querying processing techniques for graph data (including both query approximation and query relaxation) that have been developed in the context of querying lifelong learners' data and metadata [11, 17] might be applied or adapted to the much finer-granularity interaction data described here and the more challenging pedagogical setting of providing effective intelligent support to learners undertaking exploratory tasks in ELEs.

Acknowledgments

This work has been funded by the ESRC/EPSRC MiGen project, the EU FP7 projects iTalk2Learn (#318051) and M C Squared (#610467). We thank all the members of these projects for their help and insights.

5. REFERENCES

[1] N. Belacel, G. Durand, and F. LaPlante. A binary integer programming model for global optimization of learning path discovery. *G-EDM*, 2014.

[2] D. Calvanese and et al. Containment of conjunctive regular path queries with inverse. *KRR*, pages 176–185, 2015.

[3] R. Dekel and K. Gal. On-line plan recognition in exploratory learning environments. *G-EDM*, 2014.

[4] M. Eagle and T. Barnes. Exploring differences in problem solving with data-driven approach maps. *EDM*, 2014.

[5] M. Eagle, D. Hicks, B. Peddycord III, and T. Barnes. Exploring networks of problem-solving interactions. *LAK*, pages 21–30, 2015.

[6] M. Eagle, M. Johnson, and T. Barnes. Interaction networks: Generating high level hints based on network community clustering. *EDM*, 2012.

[7] B. Grawemeyer and et al. Light-bulb moment?: Towards adaptive presentation of feedback based on students' affective state. *IUI*, pages 400–404, 2015.

[8] S. Gutierrez-Santos, E. Geraniou, D. Pearce-Lazard, and A. Poulovassilis. Design of Teacher Assistance Tools in an Exploratory Learning Environment for Algebraic Generalization. *IEEE Trans. Learn. Tech.*, 5(4):366–376, 2012.

[9] S. Gutierrez-Santos, M. Mavrikis, and G. D. Magoulas. A Separation of Concerns for Engineering Intelligent Support for Exploratory Learning Environments. *J. Research and Practice in Inf. Tech.*, 44:347–360, 2013.

[10] A. Harrer, R. Hever, and S. Ziebarth. Empowering researchers to detect interaction patterns in e-collaboration. *Frontiers in Artificial Intelligence and Applications*, 158:503, 2007.

[11] C. Hurtado, A. Poulovassilis, and P. Wood. Finding top-k approximate answers to path queries. *FQAS*, pages 465–476, 2009.

[12] K. Koedinger and et al. A data repository for the EDM community: The PSLC datashop. *Handbook of Educational Data Mining*, 43, 2010.

[13] M. Mavrikis and S. Gutierrez-Santos. Not all Wizards are from Oz: Iterative design of intelligent learning environments by communication capacity tapering. *Computers and Education*, 54(3):641–651, 2010.

[14] M. Mavrikis, Z. Zheng, S. Gutierrez-Santos, and A. Poulovassilis. Visualisation and analysis of students' interaction data in exploratory learning environments. *Workshop on Web-Based Technology for Training and Education (at WWW)*, 2015.

[15] R. Noss and et al. The design of a system to support exploratory learning of algebraic generalisation. *Computers and Education*, 59(1):63–82, 2012.

[16] N. Pinkwart and et al. Graph grammars: An ITS technology for diagram representations. *FLAIRS*, pages 433–438, 2008.

[17] A. Poulovassilis, P. Selmer, and P. Wood. Flexible querying of lifelong learner metadata. *IEEE Trans. Learn. Tech.*, 5(2):117–129, 2012.

[18] O. Scheuer and B. McLaren. CASE: A configurable argumentation support engine. *IEEE Trans. Learn. Tech.*, 6(2):144–157, 2013.

[19] P. Selmer, A. Poulovassilis, and W. P.T. Implementing flexible operators for regular path queries. *GraphQ (EDBT/ICDT Workshops)*, pages 149–156, 2015.

[20] V. Sheshadri, C. Lynch, and T. Barnes. InVis: An EDM tool for graphical rendering and analysis of student interaction data. *G-EDM*, 2014.

[21] J. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *Artificial Intelligence in Education*, pages 345–352, 2011.

[22] F. Suchanek, G. Kasneci, and G. Weikum. YAGO: a core of semantic knowledge. *WWW*, 2007.

[23] D. Suthers. From contingencies to network-level phenomena: Multilevel analysis of activity and actors in heterogeneous networked learning environments. *LAK*, 2015.

Exploring the function of discussion forums in MOOCs: comparing data mining and graph-based approaches

Lorenzo Vigentini
Learning & Teaching Unit
UNSW Australia,
Lev 4 Mathews, Kensington 2065
+61 (2) 9385 6226
l.vigentini@unsw.edu.au

Andrew Clayphan
Learning & Teaching Unit
UNSW Australia,
Lev 4 Mathews, Kensington 2065
+61 (2) 9385 6226
a.clayphan@unsw.edu.au

ABSTRACT

In this paper we present an analysis (in progress) of a dataset containing forum exchanges from three different MOOCs. The forum data is enhanced because together with the exchanges and the full text, we have a description of the design and pedagogical function of forums in these courses and a certain level of detail about the users, which includes achievement, completion, and in some instances more details such as: education; employment; age; and prior MOOC exposure.

Although a direct comparison between the datasets is not possible because the nature of the participants and the courses are different, what we hope to identify using graph-based techniques is a characterization of the patterns in the nature and development of communication between students and the impact of the 'teacher presence' in the forums. With the awareness of the differences, we hope to demonstrate that student engagement can be directed 'by-design' in MOOCs: teacher presence should therefore be planned carefully in the design of large-scale courses.

Keywords

MOOCs, Discussion forums, graph-based EDM, pedagogy.

1. INTRODUCTION

In the past couple of years MOOCs (Massive Open Online Courses) have become the center of much media hype as disruptive and transformational [1, 2]. Although the focus has been on a few characteristics of the MOOCS – i.e. free courses, massive numbers, massive dropouts and implicit quality warranted by the status of the institutions delivering these courses – a rapidly growing research interest has started to question the effectiveness of MOOCS for learning and their pedagogies. If one ignores entirely the philosophies of teaching driving the design and delivery of MOOCs going from the the socio-constructivist (cMOOC, [4, 5]) to instructivist (xMOOC, [3]), at the practical level, instructors have to make specific choices about how to use the tools available to them. One of these tools is the discussion forum. Forums are one of the most popular asynchronous tools to support students' communication and collaboration in web-based learning environments [6]. These can be deployed in a variety of ways, ranging from a tangential support resource which students can refer to when they need help, to a space for learning with others, driven by the activities students have to carry out (usually sharing work and eliciting feedback). The latter, in a sense, emulates class-time in traditional courses providing a space for structured discussions about the topics of the course. One could argue that like in face-to-face classes, the value of the interaction depends on the importance attributed to the forums by the instructors. This is an interesting point to explore teachers' presence and the value of their input in directing such conversations. Mazzolini & Maddison characterize the role of the teacher and teacher presence in online discussion forums as varying from being the 'sage on the stage', to the 'guide on the side' or even 'the ghost in the wings' [7]. Furthermore they argue that the 'ideal' degree of visibility of the instructor in discussion forums depends on the purpose of forums and their relationship to assessment. There are also a number of accounts indicating that students' learning in forums is not very effective [8, 9]. However if one looks at the data there are numerous examples indicating that behaviours in forums are good predictors of performance in the courses using them, particularly if forum activities are assessed [10,11,12,13]. Yet, forums in MOOCs tend to attract only a small portion of the student activity [14]. This is setting forums in MOOCs apart from 'tutorial-type' forums used to support students' learning in online or blended courses in higher education. Furthermore, some argue that active engagement is not the only way of benefiting from discussion forums [15] and students' characteristics and preferences could be more important than the course design in determining the way in which they take full advantage of online resources [16].

2. THE THREE MOOCS IN DETAIL

In order to investigate the way in which students use the discussion forums, we have extracted data from three MOOCS delivered by a large, research intensive Australian university. The three courses are: P2P (From Particles to Planets - Physics); LTTO (Learning to Teach Online); and INTSE (Introduction to Systems Engineering), which are broadly characterised in the top of Table 1. The courses were specifically designed in quite different ways to test hypotheses about their design, delivery and effectiveness.

In particular, P2P was designed emulating a traditional university course in a sequential manner. All content was released on a week-by-week basis dictating the pace of instruction. LTTO and INTSE, instead were designed to provide a certain level of flexibility for the students to elect their learning paths. All content was readily available at the start, however for LTTO, the delivery followed a week-on-week delivery focusing on the interaction with students and a selective attention to particular weekly topics (i.e. weekly feedback videos driven by the discussion forums as well as weekly announcements). Although announcements were used also in INTSE, the lack of weekly activities in the forums did not impose a strong pacing. In INTSE, the forums had only a tangential support value and were used mainly to respond to students' queries and to clarify specific topics emerging from the quizzes. Table 1 provides an overview of the different courses. This also shows that the forum activity in the various courses is a very small portion of all actions emerging from the logs of activity which has been reported in the literature [9].

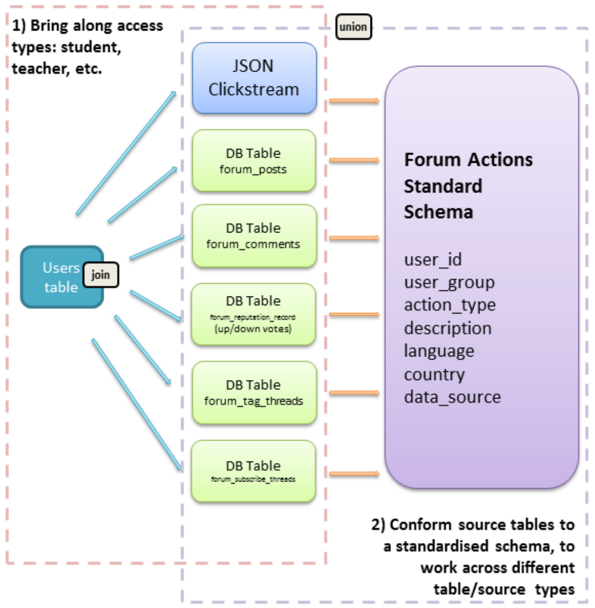
3. DETAILS OF THE DATASET

3.1 The dataset

The data under consideration is an export form the Coursera platform. Raw forum database tables (posts, comments, tags, votes) as well as a JSON based web clickstream were used. The clickstream events consist of a key which specifies action – either a 'pageview' or 'video' item. Forum clickstream events were identified by a common '/forum' prefix.

The clickstream was further classified into: browsing; profile lookups; social interaction (looking at contributions); search; tagging; and threads. From the classification it became evident the clickstream did not record all events, such as when a post or comment was made, or when votes were applied. For these, specific database tables were used. In order to manage different data sets and sources, a standardized schema was built, allowing disparate sources to feed into, but exposing a common interface to conduct analysis over forum activities. This is shown in Figure 1.

Figure 1. Forum data transformation process


1) Bring along access types: student, teacher, etc.
union
JSON Clickstream
DB Table forum_posts
DB Table forum_comments
Users table
join
DB Table forum_reputation_record (up/down votes)
DB Table forum_tag_threads
DB Table forum_subscribe_threads
Forum Actions Standard Schema
user_id
user_group
action_type
description
language
country
data_source
2) Conform source tables to a standardised schema, to work across different table/source types

3.2 An overview of forums activity

There are very interesting trends which require more detailed examination (bottom of table 1). As expected, in LTTO the forum activity is larger than in the other courses and this is probably due to the fact that students were asked to submit post in forums following the learning activities. The proportion of active students in forum is 4x in magnitude compared to the other courses. Yet, if we look at the average amount of posts or comments, the patterns are not straightforward to interpret, as the level of engagement is similar across the courses with 3 to 5 posts per student and 1 to 3 comments (i.e. replies to existing posts), but with P2P showing a higher level of engagement than the other courses. One possible explanation is the different target group of the different courses with INTSE including a majority of professional engineers with postgraduate qualifications, P2P focusing on high school student and teachers, and LTTO targeting a broad base of teachers across different educational levels.

	INTSE	LTTO	P2P
Target group	Engineers	Teachers at all levels	High school and teachers
Course length	9 weeks	8 weeks	8 weeks
Forums	54 (14 top level)	105 (17 top-level)	63 (15 top-level)
Design mode	All-at-once	All-at-once	Sequential
Delivery mode	All-at-once	Staggered	Staggered
Use of forums	**Tangential**	**Core activity**	**Support**
N in forum	422 (2.1%)	1685 (9.3%)	293 (2.8%)
Tot posts	1361 (avg=3.3)	6361 (avg=3.8)	1399 (avg=4.8)
Tot comments	285 (avg=0.7)	2728 (avg=1.7)	901 (avg=3.1)
Registrants	32705	28558	22466
Active students[1]	60%	63%	47%
Completing[2]	4.2% (0.3% D)	4.4% (2.4 D)	0.7% (0.2%)

Table 1. Summary of the courses under investigation. NOTE: 1. Active students are those appearing in the clickstream; 2. Completing students achieve the pass grade or Distinction (D)

The type of activity is summarised in Figure 2. In the chart, the five categories refer to the following: View corresponds to listing forums, threads and viewing posts; Post is the writing of a post or start of a new thread; Comment is a reply to an existing post; Social refers to all actions engaging directly with other's status (up-vote, down-vote and looking at profiles/reputation); Engage refers to the additional interaction with forums content (searching, tagging, 'watching' or subscribing to posts or threads).

The viewing behaviour is the most prominent for both the student and instructor groups and the figures are pretty much similar across the board. A two-way ANOVA (2x5, role by activity) on the percentage of distributions, shows that there is no significant difference between students and instructors, but there is an obvious difference between views and the other types of behaviour ($F(4,29) = 1656.3$, $p < .01$).

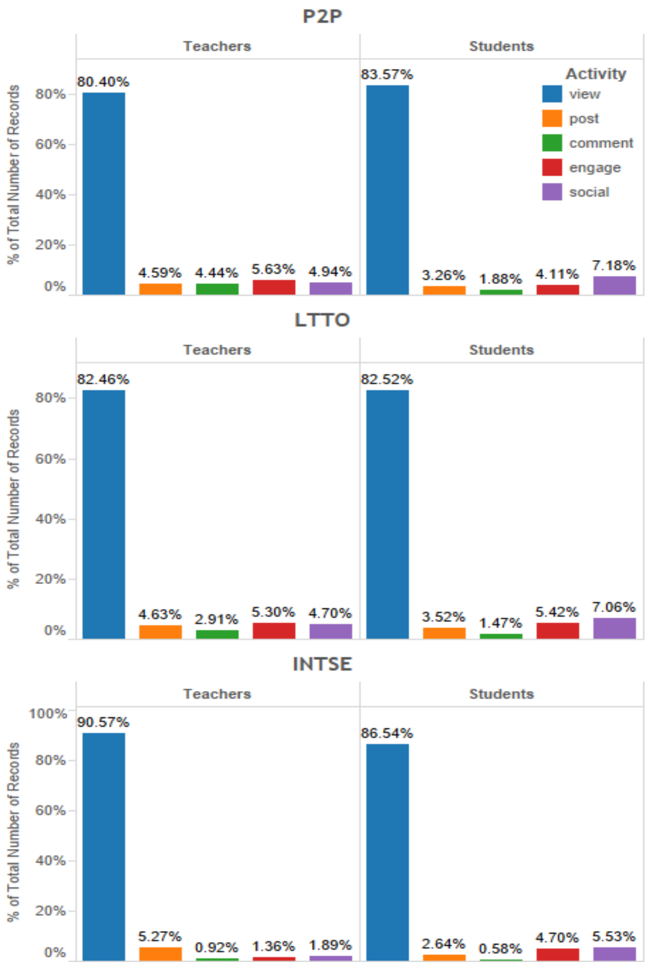
If we consider the engagement over the timeline and compare the type of activities carried out by students and instructors, Figure 3 (end of the paper) shows the patterns for the three courses. The most striking pattern is that there doesn't seem to be an obvious one. For what concerns posts and views in all the three courses there is a sense of synchronicity between the two groups, however from this chart it is not possible to understand in more detail what are the connections between what students and teachers do.

Instructors' comments are slightly offset, possibly as a reaction to students' posts. An interesting aspect is the amount of 'social' engagement in the P2P course that merits further analysis.

4. DIRECTIONS AND OPEN QUESTIONS

From this coarse analysis it is apparent that there seem to be minimal behavioural differences in the way students and instructors interact in the different courses, however more analysis is required to tackle questions about the individual differences in students' and instructors' patterns of interaction and their interrelations. Furthermore little can be said about how the nature of interactions drives the development of communication and engagement. However a number of questions like the following remain open and unanswered: how do discussions develop over time? How teacher presence affects the development of discussions? Is the number of forums affecting how students engage with them (i.e. causing disorientation)?

Figure 2. Distribution of forum activities by role



4.1 The DM and graph-based approaches

A possible way to answer the questions about the types/patterns of behaviours, the structure and development of networks and the growth of groups/communities over time might be using data mining and graph-based approaches. For example, [6] used a combination of quantitative, qualitative and social network information about forum usage to predict students' success or failure in a course by applying classification algorithms and classification via clustering algorithms. In their approach the activity of students in the forums is organized according to a set of commonly used quantitative metrics and a couple of measures borrowed from Social Network Analysis (table 2). Although this seems to be a promising approach, there are two issues with this methodology in the MOOCs: 1) only a tiny proportion of students can be considered active and 2) it is hard to scale the instructor's evaluation. The first problem is not easily resolved and it is an issue in the literature reviewed [17, 18]; non-posting behavior is considered as an index of disengagement, partly because this is easy to measure. In principle the latter could be substituted by peer evaluation (up-vote, down-vote), but there is no easy way to ensure consistency.

Indicator	Type	Description
Messages	Quantitative	Number of messages written by the student.
Threads	Quantitative	Number of new threads created by the student.
Words	Quantitative	Number of words written by the student.
Sentences	Quantitative	Number of sentences written by the student.
Reads	Quantitative	Number of messages read on the forum by the student.
Time	Quantitative	Total time, in minutes, spent on forum by the student.
AvgScoreMsg	Qualitative	Average score on the instructor's evaluation of the student's messages.
Centrality	Social	Degree centrality of the student.
Prestige	Social	Degree prestige of the student.

Table 2. Possible indicators characterising forum engagement

An alternative method that can be explored is graph-based approaches. For example, Bhattacharya et al. [19] used graph-based techniques to explore the evolution of software and source branching providing an insight in the process. Kruck et al. [20] developed GSLAP, an interactive, graph-based tool for analyzing web site traffic based on user-defined criteria.

Kobayashi et al. [18] used a method to quickly identify and track the evolution of topics in large datasets using a mix of assignment of documents to time slices and clustering to identify discussion topics. Yang et al [21] integrated graph-based clustering to characterize the emergence of communities and text-based analysis to portray the nature of exchanges. In fact, students move in the various sub-forums taking different roles or stances as they engage with different subsets of students. As the reasons to engage in these discussions are partly determined by different interests, goals, and issues, it is possible to construct a social network graph based on the post-reply-comment structure within threads. The network generated provides a possible view of a student's social participation within a MOOC, which may indicate some detail about their values, beliefs and intentions.

Furthermore, Brown et al [22] have already shown the value of exploring the communities in discussion forums in MOOCs particularly for what concerns the homogeneity of performance but dissimilarity of motivations characterizing student hubs.

4.2 Discussion points

The examples above provide evidence of the potential for using graph-based methods to obtain better insights into the process and content analysis for our dataset and to extend its applicability to MOOCs, however there are a number of contentious points to raise which will provide opportunities for discussion.

Firstly the number of students who are actively involved in discussion is a very small proportion of the *active* participants. This means that the subset may not be representative at all. One could argue that these students are already engaged or desperately need help. Previous literature [21, 22, 23] focused on the ability to predict performance and on the peer effect which can emerge from the analysis of the graphs/social networks.

Secondly, one could question the value of the communities in xMOOCs: especially when courses are designed with an instructivits approach leading to mastery, by definition this is an individualistic perspective focused on the testing of one's own skills/learning. Of course in cMOOCs -connectivists by design- the importance of the development of social support is essential. This seems to be supported by Brown et al [22]: they were not able to uncover a direct relation between stated goals and motivations with the participation in forums, and attributed this to pragmatic needs. However, as the authors suggested earlier, the instructors might play a fundamental role in shaping the communities based on the value attributed to forums in their plans/design and the level of engagement/interaction. Considering the split between cMOOCs and xMOOCs again, interesting work might come out of the experiment conducted by Rose' and colleagues in the DALMOOC in which automated agents were deployed to support students' conversations. In Coursera the deployment of 'community mentors' will be an interesting space to explore, given that the importance of design seems to be removed from instructors in the 'on-demand' model.

Lastly, more research is needed in the time-based dimension of development of forums in MOOCs. Questions like how students bond and create stable relations, how they become authoritative and what motivates them to contribute over time are all open questions which the analysis of graphs over time might be able to address.

5. REFERENCES

[1] Dirk Jan van den Berg and Edward Crawley. Why MOOCS Are Transforming the Face of Higher Education. Retrieved April 12, 2015 from http://www.huffingtonpost.co.uk/dirk-jan-van-den-berg/why-moocs-are-transforming_b_4116819.html

[2] Chris Parr. The evolution of Moocs. Retrieved April 12, 2015 from http://www.timeshighereducation.co.uk/comment/opinion/the-evolution-of-moocs/2015614.article

[3] C. Osvaldo Rodriguez. 2012. MOOCs and the AI-Stanford Like Courses: Two Successful and Distinct Course Formats for Massive Open Online Courses. *European Journal of Open, Distance and E-Learning* (January 2012).

[4] George Siemens. 2005. Connectivism: A learning theory for the digital age. *International journal of instructional technology and distance learning* 2, 1 (2005), 3–10.

[5] Stephen Downes. 2008. Places to go: Connectivism & connective knowledge, Innovate.

[6] Cristóbal Romero, Manuel-Ignacio López, Jose-María Luna, and Sebastián Ventura. 2013. Predicting students' final performance from participation in on-line discussion forums. *Computers & Education* 68 (October 2013), 458–472. DOI: http://dx.doi.org/10.1016/j.compedu.2013.06.009

[7] Margaret Mazzolini and Sarah Maddison. 2007. When to jump in: The role of the instructor in online discussion forums. *Computers & Education* 49, 2 (September 2007), 193–213. DOI:http://dx.doi.org/10.1016/j.compedu.2005.06.011

[8] M.j.w. Thomas. 2002. Learning within incoherent structures: the space of online discussion forums. *Journal of Computer Assisted Learning* 18, 3 (September 2002), 351–366. DOI:http://dx.doi.org/10.1046/j.0266-4909.2002.03800.x

[9] Daniel F.O. Onah, Jane Sinclair, and Russell Boyatt. 2014. Exploring the use of MOOC discussion forums. In *Proceedings of London International Conference on Education*. London: LICE, 1–4.

[10] Alstete, J.W. and Beutell, N.J. Performance indicators in online distance learning courses: a study of management education. *Quality Assurance in Education 12*, 1 (2004), 6–14.

[11] Cheng, C.K., Paré, D.E., Collimore, L.-M., and Joordens, S. Assessing the effectiveness of a voluntary online discussion forum on improving students' course performance. *Computers & Education 56*, 1 (2011), 253–261.

[12] Palmer, S., Holt, D., and Bray, S. Does the discussion help? The impact of a formally assessed online discussion on final student results. *British Journal of Educational Technology 39*, 5 (2008), 847–858.

[13] Patel, J. and Aghayere, A. Students' Perspective on the Impact of a Web-based Discussion Forum on Student Learning. *Frontiers in Education Conference, 36th Annual*, (2006), 26–31.

[14] Jacqueline Aundree Baxter and Jo Haycock. 2014. Roles and student identities in online large course forums: Implications for practice. *The International Review of Research in Open and Distributed Learning* 15, 1 (January 2014).

[15] Vanessa Paz Dennen. 2008. Pedagogical lurking: Student engagement in non-posting discussion behavior. *Computers in Human Behavior* 24, 4 (July 2008), 1624–1633. DOI:http://dx.doi.org/10.1016/j.chb.2007.06.003

[16] René F. Kizilcec, Chris Piech, and Emily Schneider. 2013. Deconstructing Disengagement: Analyzing Learner Subpopulations in Massive Open Online Courses. In *Proceedings of the Third International Conference on Learning Analytics and Knowledge*. LAK '13. New York, NY, USA: ACM, 170–179. DOI:http://dx.doi.org/10.1145/2460296.2460330

[17] Dennen, V.P. Pedagogical lurking: Student engagement in non-posting discussion behavior. *Computers in Human Behavior 24*, 4 (2008), 1624–1633.

[18] Kobayashi, M. and Yung, R. Tracking Topic Evolution in On-Line Postings: 2006 IBM Innovation Jam Data. In T. Washio, E. Suzuki, K.M. Ting and A. Inokuchi, eds., *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2008, 616–625.

[19] Bhattacharya, P., Iliofotou, M., Neamtiu, I., and Faloutsos, M. Graph-based Analysis and Prediction for Software Evolution. *Proceedings of the 34th International Conference on Software Engineering*, IEEE Press (2012), 419–429.

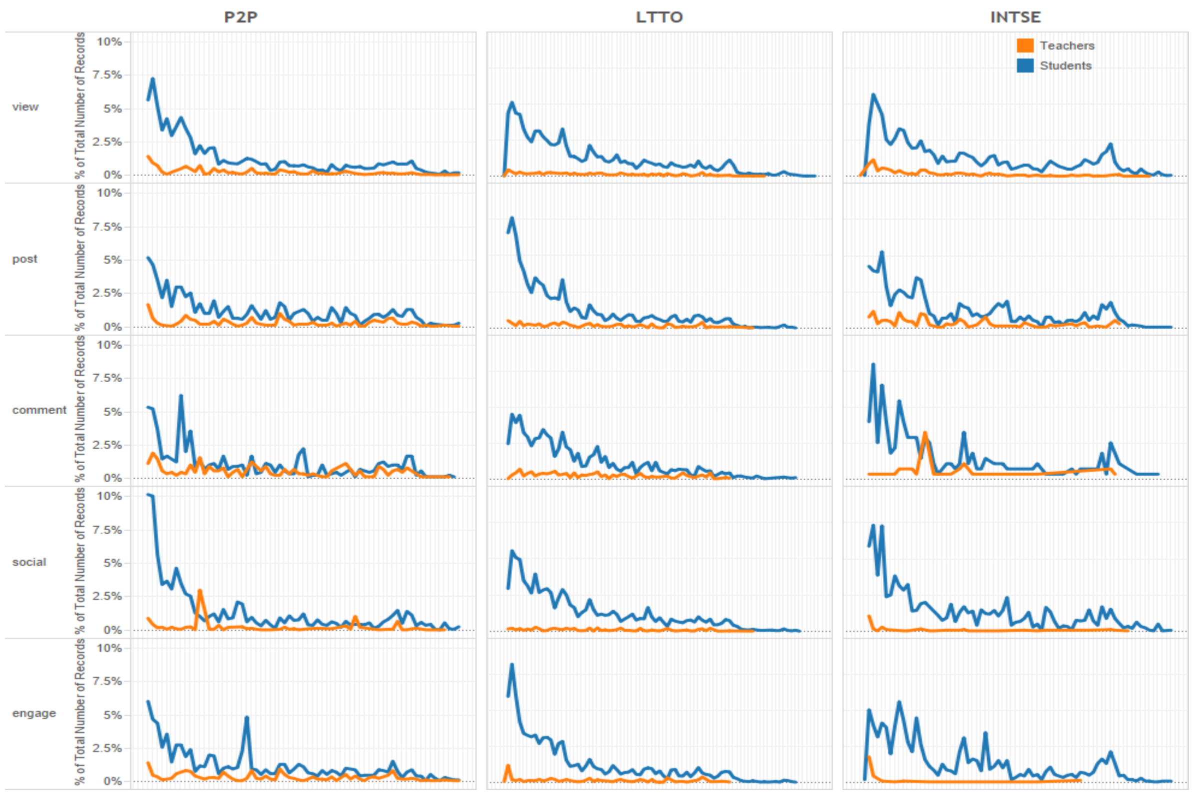
[20] Kruck, S.E., Teer, F., and Jr, W.A.C. GSLAP: a graph‐based web analysis tool. *Industrial Management & Data Systems 108*, 2 (2008), 162–172.

[21] D. Yang, M. Wen, A. Kumar, E. P. Xing, and C. P. Rose, “Towards an integration of text and graph clustering methods as a lens for studying social interaction in MOOCs,” *The International Review of Research in Open and Distributed Learning*, vol. 15, no. 5, Oct. 2014.

[22] R. Brown, C Lynch, Y. Wang, M. Eagle, J. Albert, T. Barnes, R. Baker, Y. Bergner, D. McNamara. Communities of performance and communities of preference. GEDM 2015, in press.

[23] M. Fire, G. Katz, Y. Elovici, B. Shapira, and L. Rokach, “Predicting Student Exam’s Scores by Analyzing Social Network Data,” in *Active Media Technology*, R. Huang, A. A. Ghorbani, G. Pasi, T. Yamaguchi, N. Y. Yen, and B. Jin, Eds. Springer Berlin Heidelberg, 2012, pp. 584–595.

Figure 3. Time sequence of activity in the forums in the three courses by students and instructors grouped by activity type



BOTS: Selecting Next-Steps from Player Traces in a Puzzle Game

Drew Hicks
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
aghicks3@ncsu.edu

Yihuan Dong
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
ydong2@ncsu.edu

Rui Zhi
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
rzhi@ncsu.edu

Veronica Cateté
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
vmcatete@ncsu.edu

Tiffany Barnes
North Carolina State University
911 Oval Drive
Raleigh, NC 27606
tmbarnes@ncsu.edu

ABSTRACT

In the field of Intelligent Tutoring Systems, data-driven methods for providing hints and feedback are becoming increasingly popular. One such method, Hint Factory, builds an interaction network out of observed player traces. This data structure is used to select the most appropriate next step from any previously observed state, which can then be used to provide guidance to future players. However, this method has previously been employed in systems in which each action a player may take requires roughly similar effort; that is, the "step cost" is constant no matter what action is taken. We hope to apply similar methods to an interaction network built from player traces in our game, BOTS; However, each edge can represent a varied amount of effort on the part of the student. Therefore, a different hint selection policy may be needed. In this paper, we discuss the problems with our current hint policy, assuming all edges are the same cost. Then, we discuss potential alternative hint selection policies we have considered.

Keywords

Hint Generation, Serious Games, Data Mining

1. INTRODUCTION

Data-driven methods for providing hints and feedback are becoming increasingly popular, and are especially useful for environments with user- or procedurally-generated content. One such method, Hint Factory, builds an interaction network out of observed player traces. An Interaction Network is a complex network of student-tutor interactions, used to model student behavior in tutors, and provide insight into problem-solving strategies and misconceptions. This data structure can be used to provide hints, by treating the Interaction Network similarly to a Markov Decision Process and selecting the most appropriate next step from the requesting user's current state. This method has successfully been employed in systems in which each action a player may take is of similar cost; for example in the Deep Thought logic tutor each action is an application of a particular axiom. Applying this method to an environment where actions are of different costs, or outcomes are of varying value will require some adaptations to be made. In this work, we discuss how we will apply Hint Factory methods to an interaction network built from player traces in a puzzle game, BOTS. In BOTS, each "Action" is the set of changes made to the program between each run. Therefore, using the current hint selection policy would result in very high-level hints comprising a great number of changes to the student's program. Since this is undesirable, a different hint selection policy may be needed.

2. DATA-DRIVEN HINTS AND FEEDBACK

In the ITS community, several methods have been proposed for generating hints/feedback from previous observations of users' solutions or behavior. Rivers et al propose a data-driven method to generate hints automatically for novice programmers based on Hint Factory[8]. They present a domain-independent algorithm, which automates hint generation. Their method relies on solution space, which utilizes graph to represent the solution states. In solution space, each node represents a candidate solution and each edge represents the action used to transfer from one state to another. Due to the existence of multiple ways to solve a programming problem, the size of the solution space is huge and thus it is impractical to use. A Canonicalizing model is used to reduce the size of the solution space. All states are transformed to canonicalized abstract syntax trees (ASTs). If the canonical form of two different states are identical, they can be combined together. After simplifying the solution space, hint generation is implemented. If the current state is incorrect and not in the solution space, the path construction algorithm will find an optimal goal state in the solution space which is closest to current state. This algorithm uses change vectors to denote the change between current state and goal state. Once a better goal state is found during enumerating all possible changes, it returns the current combination of change vectors. Each change vector can be applied to current state and then form an intermediate state. The intermediate states are measured by desirability score, which represents the value of the state. And then the path construction algorithm generates optimal next states based on the rank of the desirability scores of all the intermediate states. Thus a new path can be formed and added to the solution space, and appropriate hints can be generated.

Jin et al propose linkage graph to generate hints for programming courses[4]. Linkage graph uses nodes to represent program statements and direct edges to indicate ordered dependencies of those statements. Jin's approach applies matrix to store linkage graph for computation. To generate linkage matrix, first, they normalize variables in programs by using instructor-provided variable specification file. After variable normalization, they sort the statement with 3 steps: (i) preprocessing, which breaks a single declaration for multiple variables (e.g. int a, b, c) into multiple declaration statements (e.g. int a; int b; int c;); (ii) creating statement sets according to variable dependencies, which put independent statements into first set, put statements depend only on statements in the first set into second set, put statements depends only on statements in the first and second set into third set, and so on; (iii) in-set statement sorting, during which the statements are sorted in decreasing order within set using their variable signatures. In hint generation, they first generate linkage graphs with a set of correct solutions, as the sources for hint generation. They also compose the intermediate steps during program development into a large linkage graph, and assign a reward value to each state and the correct solution. Then, they apply value iteration to create a Markov Decision Process (MDP). When a student requires hint, tutor will generate a linkage graph for the partial program and try to find the closest match in MDP. If a match is found in MDP, the tutor would generate hint with the next best state based on highest assigned value. If a match is not found in current MDP, which means the student is taking a different approach from existing correct solutions, the tutor will try to modify those correct solutions to fit student's program and then provide hints.

Hint Factory[9] is an automatic hint generation technique which uses Markov decision processes (MDPs) to generate contextualized hints from past student data. It mainly consists of two parts - Markov Decision Process (MDP) generator and hint provider. The MDP generator runs a process to generate MDP values for all states seen in previous students' solutions. In this process, all the students' solutions are combined together to form a single graph. Each node of the graph represents a state, and each edge represents an action one student takes to transform from current state to another state. Once the graph is built, the MDP generator uses Bellman backup to assign values for all nodes. After updating all values, a hint file is generated. The hint provider uses hint file to provide hint. When a student asks for a hint at a existing state, hint provider will retrieve current state

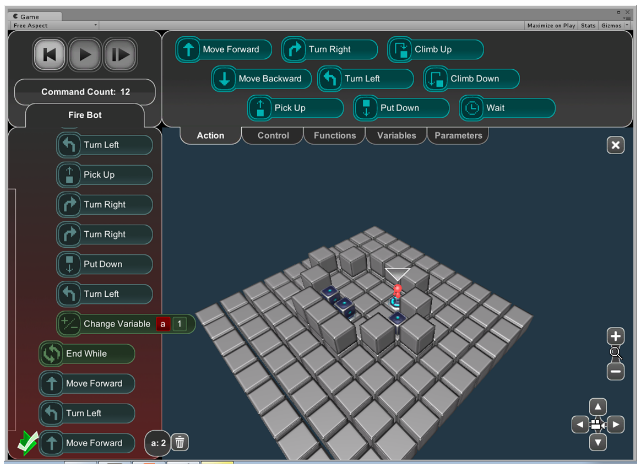


Figure 1: The BOTS interface. The robot's program is along the left side of the screen. The "toolbox" of available commands is along the top of the screen.

information and check if hints are available for the state. The action that leads to subsequent state with the highest value is used to generate a hint sequence. A hint sequence consists of four types of hints and are ordered from general hint to detailed hint. Hint provider will then show hint from top of the sequence to the student. Hint Factory has been applied in logic tutors which helps students learn logic proof. The result shows that the hint-generating function could provide hints over 80% of the time.

3. BOTS

BOTS is a programming puzzle game designed to teach fundamental ideas of programming and problem-solving to novice computer users. The goal of the BOTS project is to investigate how to best use community-authored content within serious games and educational games. BOTS was inspired by games like LightBot [10] and RoboRally [2], as well as the success of Scratch and it's online community [1] [5]. In BOTS, players take on the role of programmers writing code to navigate a simple robot around a grid-based 3D environment, as seen in Figure 1. The goal of each puzzle is to press several switches within the environment, which can be done by placing an object or the robot on them. To program the robots, players will use simple graphical pseudo-code, allowing them to move the robot, repeat sets of commands using "for" or "while" loops, and re-use chunks of code using functions. Within each puzzle, players' scores depend on the number of commands used, with lower scores being preferable. In addition, each puzzle limits the maximum number of commands, as well as the number of times each command can be used. For example, in the tutorial levels, a user may only use the "Move Forward" instruction 10 times. Therefore, if a player wants to make the robot walk down a long hallway, it will be more efficient to use a loop to repeat a single "Move Forward" instruction, rather than to simply use several "Move Forward" instructions one after the other. These constraints are meant to encourage players to re-use code and optimize their solutions.

In addition to the guided tutorial mode, BOTS also contains an extensive "Free Play" mode, with a wide selection of puzzles created by other players. The game, in line with the "Flow of Inspiration" principles outlined by Alexander Repenning [7], provides multiple ways for players to share knowledge through authoring and modifying content. Players are able to create their own puzzles to share with their peers, and can play and evaluate friends' puzzles, improving on past solutions. Features such as peer-authored hints for difficult puzzles, and a collaborative filtering approach to rating are planned next steps for the game's online element. We hope to create an environment where players can continually challenge their peers to find consistently better solutions for increasingly difficult problems.

User-generated content supports replayability and a sense of a community for a serious game. We believe that user-created puzzles could improve interest, encouraging students to return to the game to solidify their mastery of old skills and potentially helping them pick up new ones.

4. ANALYSIS

4.1 Dataset

Data for the BOTS studies has come from a middle school computer science enrichment program called SPARCS. In this program, the students attend class on Saturday for 4 hours where computer science undergraduates teach them about computational thinking and programming. Students attend a total of 7 sessions, each on a different topic, ranging from security and encryption to game design. The students all attend the same magnet middle school. The demographics for this club are 74.2% male, 25.8% female, 36.7% African American, and 23.3% Hispanic. The student's grade distribution is 58% 6th grade, 36% 7th grade and 6% 8th grade.

From these sessions, we collected gameplay data for 20 tutorial puzzles as well as 13 user-created puzzles, With this data, we created an Interaction Network in order to be able to provide hints and feedback for future students [3]. However, using program edits as states, the interaction networks produced were very sparse. In order to be better able to relate similar actions, we produced another interaction network using program output as our state definition [6].

4.2 States and Transitions

Based on the data collected, we can divide the set of observed states into classes. First among these is the *start state* in which the problem begins. By definition, every player's path must begin at this state. Next is the set of *goal states* in which all buttons on the stage are pressed. These are reported by the game as correct solutions. Any complete solution, by definition, ends at one such state. Among states which are neither start nor goal states, there are three important classifications: *Intermediate states* (states a robot moves through during a correct solution), *mistake states* (states a robot does not move through during a correct solution), and *error states* (states which result from illegal output, like attempting to move the robot out-of-bounds). Based on these types of states, we classified our hints based on the transitions they represented.

4.2.1 Subgoal Transition

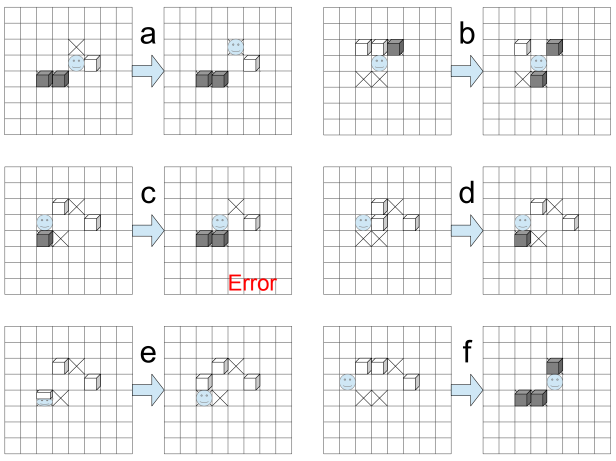


Figure 2: Several generated hints for a simple puzzle. The blue icon represents the robot. The 'X' icon represents a goal. Shaded boxes are boxes placed on goals, while unshaded boxes are not on goals. Hint F in this figure depicts the start and most common goal state of the puzzle.

(start/intermediate) $\rightarrow$ *(intermediate/goal)* These transitions occur when a student moves the robot to an intermediate state rather than directly to the goal. Since players run their programs to produce output, we speculate that these may represent subgoals such asmoving a box onto a specific switch. After accomplishing that task, the user then appends to their program, moving towards a new objective, until they reach a goal state. Hint B in Figure 2 shows a hint generated from such a transition.

4.2.2 Correction Transition

(error/mistake) $\rightarrow$ *(intermediate/goal)* This transition occurs when a student makes and then corrects a mistake. These are especially useful because we can offer hints based on the type of mistake. Hints D and E in Figure 2 show hints built from this type of transition; however, hint E shows a case where a student resolved the mistake in a suboptimal way.

4.2.3 Simple Solution Transition

(start) $\rightarrow$ *(goal)* This occurs when a student enters an entire, correct program, and solves the puzzle in one attempt. This makes such transitions not particularly useful for generating hints, other than showing a potential solution state of the puzzle. Hint F in Figure 2 shows this type of transition.

4.2.4 Rethinking Transition

(intermediate) $\rightarrow$ *(intermediate/goal)* This transition occurs when rather than appending to the program as in a subgoal transition, the user deletes part or all of their program, then moving towards a new goal. As a result, the first state is unrelated to the next state the player reaches. Offering this state as a hint would likely not help guide a different user. Hint A in Figure 2 shows an example of this. Finding and recognizing these is an important direction for future work.

4.2.5 Error Transition

(start/intermediate) → (mistake/error) This corresponds to a program which walks the robot out of bounds, into an object, or other similar errors. While we disregarded these as hints, this type of transition may still be useful. In such a case, the last *legal* output before the error could be a valuable state. Hint C in Figure 2 is one such case.

4.3 Next Steps

While this approach was able to help us identify interesting transitions, as well as significantly reduce the sparseness of the Interaction Network by merging states with similar output, we violate several assumptions of the Hint Factory technique by using user compilation as an action. Essentially, the cost of an action can vary widely. In the most extreme examples, the best next state selected by Hint Factory will simply be the goal state.

4.4 Current Hint Policy

Our current hint selection policy is the same as the one used in the logic tutor Deep Thought with a few exceptions [9]. We combine all student solution paths into a single graph, mapping identical states to one another (comparing either the programs or the output). Then, we calculate a fitness value for each node. We assign a large positive value (100) to each goal state, a low value for dead-end states (0) and a step cost for each step taken (1). Setting a non-zero cost on actions biases us towards shorter solutions. We then calculate fitness values $V(s)$ for each state s, where $R(s)$ is the initial fitness value for the state, γ is a discount factor, and $P(s, s')$ is the observed frequency with which users in state s go to state s' next, via taking the action a. The equation for determining the fitness value of a state is as follows:

$$V(s) := R(s) + \gamma \max_a \sum_{s'} P_a(s, s')V(s') \tag{1}$$

However, in our current representation there is only one available action from any state: "run." Different players using this action will change their programs in different ways between runs, so it is not useful to aggregate across all the possible resulting states. Instead, we want to consider each resulting state on its own. As a result, we use a simplified version of the above, essentially considering each possible resulting state s' as the definite result of its own action:

$$V(s) := R(s) + \gamma \max_{s'} P(s, s')V(s') \tag{2}$$

Since the action "run" can encompass many changes, selecting the s' which maximizes the value may not always be the best choice for a hint. The difference between s and s' can be quite large, and this is usually the case when an expert user solves the problem in one try, forming an edge directly between the "start" state and "goal" state. These and other "short-circuits" make it difficult to assess which of the child nodes would be best to offer as a hint by simply using the calculated fitness value.

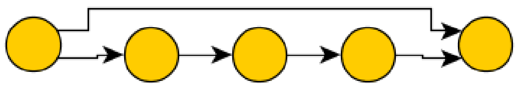


Figure 3: This subgraph shows a short-circuit where a player bypasses a chain of several steps.

Another problem which arises from this state representation is seen in Hints C and E above. These hints show states where a student traveled from a state to a worse state before ultimately solving the problem. Since we limit our search for hintable states to the immediate child states of s in s', we are unable to escape from such a situation if the path containing the error is the best or only observable path to the goal.

4.5 Proposed Hint Policies

One potential modification of the hint policy involves analyzing the programs/output on the nodes, using some distance metric $\delta(s, s')$. This measurement would be used in addition to the state's independent fitness value $R(s)$ which takes into account distance from a goal, but is irrespective of the distance from any previous state. For example in the short-circuit example above, using "difference in number of lines of code" as a distance metric we could take into account how far the "Goal" state is from the "Start" state, and potentially choose a nearer state as a hint. This also helps correct for small error-correction steps in player solutions; if the change between the current state and the target hint state is very small, we may want to consider hinting toward the next step instead, or a different solution path altogether.

$$V(s) := R(s) + \gamma \max_a \sum_{s'} \delta(s, s')P(s, s')V(s') \tag{3}$$

One potential downside to this approach is that it requires somewhat more knowledge of the domain to be built into the model. If the distance metric used is inaccurate or flawed, there may be cases where we choose a very suboptimal hint. using difference in lines of code as our distance metric, the change between a state where a player is using no functions and a state where the user writes existing code into a function may be very small. Hints selected in these cases might guide students away from desired outcomes in our game.

Another problem we need to resolve with our current hint policy, as discussed above, is the case where the best or only path to a goal from a given state s has an error as a direct child s'. One method of resolving this could be, instead of offering s' as a hint, continuing to ask for next-step hints from s' until some s' is a hintable, non-error state. This solution requires no additional knowledge of the game domain, however it's possible that the hint produced will be very far from s, or that we may skip over important information about how to resolve the error or misconception that led the student into state s in the first place.

Other modifications to the hint selection policy may produce better results than these. We hope to look into as many possible modifications as we can, seeing which modifications produce the most suitable hints on our current dataset before settling on an implementation for the live version of the game.

5. ACKNOWLEDGMENTS

Thanks to the additional developers who have worked on this project or helped with our outreach activities so far, including Aaron Quidley, Trevor Brennan, Irena Rindos, Vincent Bugica, Victoria Cooper, Dustin Culler, Shaun Pickford, Antoine Campbell, and Javier Olaya. This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. 0900860 and Grant No. 1252376.

6. REFERENCES

[1] I. F. de Kereki. Scratch: Applications in computer science 1. In *Frontiers in Education Conference, 2008. FIE 2008. 38th Annual*, pages T3B–7. IEEE, 2008.

[2] R. Garfield. Roborally. [Board Game], 1994.

[3] A. Hicks, B. Peddycord III, and T. Barnes. Building games to learn from their players: Generating hints in a serious game. In *Intelligent Tutoring Systems*, pages 312–317. Springer, 2014.

[4] W. Jin, T. Barnes, J. Stamper, M. J. Eagle, M. W. Johnson, and L. Lehmann. Program representation for automatic hint generation for a data-driven novice programming tutor. In *Intelligent Tutoring Systems*, pages 304–309. Springer, 2012.

[5] D. J. Malan and H. H. Leitner. Scratch for budding computer scientists. *ACM SIGCSE Bulletin*, 39(1):223–227, 2007.

[6] B. Peddycord III, A. Hicks, and T. Barnes. Generating hints for programming problems using intermediate output.

[7] A. Repenning, A. Basawapatna, and K. H. Koh. Making university education more like middle school computer club: facilitating the flow of inspiration. In *Proceedings of the 14th Western Canadian Conference on Computing Education*, pages 9–16. ACM, 2009.

[8] K. Rivers and K. R. Koedinger. Automating hint generation with solution space path construction. In *Intelligent Tutoring Systems*, pages 329–339. Springer, 2014.

[9] J. Stamper, T. Barnes, L. Lehmann, and M. Croy. The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track*, pages 71–78, 2008.

[10] D. Yaroslavski. LightBot. [Video Game], 2008.

Semantic Graphs for Mathematics Word Problems based on Mathematics Terminology

Rogers Jeffrey Leo John
Center for Computational Learning Systems
Columbia University
New York, NY, USA
rl2689@columbia.edu

Thomas S. McTavish
Center for Digital Data, Analytics & Adaptive Learning
Pearson
Austin, TX, USA
tom.mctavish@pearson.com

Rebecca J. Passonneau
Center for Computational Learning Systems
Columbia University
New York, NY, USA
becky@ccls.columbia.edu

ABSTRACT

We present a graph-based approach to discover and extend semantic relationships found in a mathematics curriculum to more general network structures that can illuminate relationships within the instructional material. Using words representative of a secondary level mathematics curriculum we identified in separate work, we constructed two similarity networks of word problems in a mathematics textbook, and used analogous random walks over the two networks to discover patterns. The two graph walks provide similar global views of problem similarity within and across chapters, but are affected differently by number of math words in a problem and math word frequency.

1. INTRODUCTION

Curricula are compiled learning objects, typically presented in sequential order and arranged hierarchically, as in a book's Table of Contents. Ideally, a *domain model* captures relationships between the learning objects and the knowledge components or skills they exercise. Unfortunately, domain models are not often granular enough for optimal learning experiences. For example, prerequisite relationships may be lacking, or the knowledge components associated with an exercise may be unknown. In such cases, assessments on those learning objects will be insufficient to enable appropriate redirection unless expert (i.e. teacher) intervention is explicitly given. Domain models remain coarse because using experts to enumerate and relate the knowledge components is costly.

As a means to automatically discover relationships among learning objects and to reveal their knowledge components, we demonstrate the use of direct similarity metrics and random graph walks to relate exercises in a mathematics curriculum. We first apply a standard cosine similarity measure between pairs of exercises, based on bag-of-word vectors consisting of math terms that we identified in separate work [7]. Then, to extract less explicit relationships between exercises, we randomly walk a graph using the cosine distance as edge weights. We also recast the problem as a bipartite graph with exercises on one side and words on the other, providing an edge when an exercise contains the math word. We contrast these two different types of random walks and find somewhat similar results, which lends confidence to the analysis. The bipartite graph walks, however, are more sensitive to differences in word frequency. Casting measures of similarity as graphs and performing random walks on them affords more nuanced ways of relating objects, which can be used to build more granular domain models for analysis of prerequisites, instructional design, and adaptive learning.

2. RELATED WORK

Random walks over graphs have been used extensively to measure text similarity. Applications include similarity of web pages [15] and other documents [5], citations [1], passages [14], person names in email [12] and so on. More recently, general methods that link graph walks with external resources like WordNet have been developed to produce a single system that handles semantic similarity for words, sentences or text [16]. Very little work compares walks over graphs of the same content, where the graphs have different structure. We create two different kinds of graphs for mathematics word problem and compare the results. We find that the global results are very similar, which is good evidence for the general approach, and we find differences in detail that suggest further investigation could lead to customizable methods, depending on needs.

An initiative where elementary science and math tests are a driver for artificial intelligence has led to work on knowledge extraction from textbooks. Berant et al. [2] create a system to perform domain-specific deep semantic analysis of a 48 paragraphs from a biology textbook for question answering. Extracted relations serve as a knowledge base against which to answer questions, and answering a question is treated as finding a proof. A shallow approach to knowledge extraction from a fourth grade science curriculum is taken in [6], and the knowledge base is extended through dialog with users until a path in the knowledge network can be found that supports a known answer. In the math domain, Kushman et al. [10] generate a global representation of algebra problems in order to solve them by extracting relations from sentences and aligning them. Seo et al. [18] study text and diagrams together in order to understand the diagrams better through textual cues. We are concerned with alignment of content

Two machines produce the same type of widget. Machine A produces W widgets, X of which are damaged. Machine B produces Y widgets, Z of which are damaged. The **fraction** of damaged widgets for Machine A is $\frac{X}{W}$ or (*simplified fraction*). The **fraction** of damaged widgets for Machine B is $\frac{Z}{Y}$ or (*simplified fraction*). Write each **fraction** as a **decimal** and a **percent**. Use pencil and paper. Select a small **percent** that would allow for a small **number** of damaged widgets. Find the **number** of widgets by which each machine exceeded the acceptable **number** of widgets.

Figure 1: Sample problem; math terms are in boldface.

across rather than within problems, and our objective is finer-grained analysis of curricula.

Other work that addresses knowledge representation from text includes ontology learning [3], which often focuses on the acquisition of sets of facts from text [4]. There has been some work on linking lexical resources like WordNet or FrameNet to formal ontologies [17, 13], which could provide a foundation for reasoning over facts extracted from text. We find one work that applies relation mining to e-learning: Šimko and Bieliková [19] apply automated relation mining to extract relations to support e-course authoring in the domain of teaching functional programming. Li et al. [11] apply k-means clustering to a combination of problem features and student performance features, and propose the clusters correspond to Knowledge Components [8].

3. METHODS

3.1 Data

We used 1800 exercises from 17 chapters of a Grade 7 mathematics curriculum. Most are word problems, as illustrated in Figure 1. They can incorporate images, tables, and graphs, but for our analysis, we use only the text. The vocabulary of the resulting text consists of 3,500 distinct words. We construct graphs where math exercises are the nodes, or in a bipartite graph, math exercises are the left side nodes and words are the right side nodes. Our initial focus is on exercise similarity due to similarity of the math skills that exercises tap into, and we use mathematics terminology as an indirect proxy of skills a problem draws upon.

3.2 Math Terminology

The text of the word problems includes ordinary language expressions unrelated to the mathematics curriculum, such as the nouns *machines, widgets* shown in problem in Figure 1, or the verbs *produces, damaged*. For our purposes, mathematics terminology consists of words that expresses concepts that are needed for the mathematical competence the curriculum addresses. To identify these terms, we developed annotation guidelines for human annotators who label words in their contexts of use, and assessed the reliability of annotation by these guidelines. Words can be used in the math texts sometimes in a math sense and sometimes in a non-math sense. Annotators were instructed to label terms based on the most frequent usage.

Using a chance-adjusted agreement coefficient in [-1,1] [9], reliability among three annotators was 0.81, representing high agreement. All the non-stop words were then labeled by a trained annotator. We developed a supervised machine learning approach to classify vocabulary into math and non-math words [7] that can be applied to new mathematics curricula. For the text used here, there were 577 math terms.

3.3 Random Walks in Graphs

A random walk on a graph starts at a given node and steps with random probability to a neighboring node. The same random decision process is employed at this and every subsequent node until a termination criterion is met. Each time a node is visited, it is counted. Open random walks require that the start node and end nodes differ. Traversal methods may employ a bias to navigate toward or away from certain neighbors through edge weights or other graph attributes.

In a graph, $G = (V, E)$ with nodes V and edges E, a random walk that begins at v_x and ends at v_y can be denoted as $(v_x, ..., v_y)$. By performing several random walks, the fraction of times the node v_y is visited converges to the probability of target v_y being visited given the start node v_x, which can be expressed as $P(v_y|v_x)$ under the conditions of the walk. In the case of a random walk length of 1, $P(v_y|v_x)$ will simply measure the probability of v_y being selected as an adjacent node to v_x.

3.4 Cosine Similarity Graph

Math exercises are represented as bag-of-words vectors with boolean values to indicate whether a given math term is present. Cosine similarity quantifies the angle between the two vectors, and is given by the dot product of two vectors.

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{te}}{\|\mathbf{t}\|\|\mathbf{e}\|} = \frac{\sum_{i=1}^{n} \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^{n} (\mathbf{t}_i)^2}\sqrt{\sum_{i=1}^{n} (\mathbf{e}_i)^2}} \tag{1}$$

Similarity values of 1 indicate that both the vectors are the same whereas a value of zero indicates orthogonality between the two vectors. Pairwise cosine similarities for all 1800 exercises were computed, yielding a cosine similarity matrix M_{cos}. The matrix corresponds to a graph where non-zero cosine similarities are edge weights between exercises.

In a graph walk, the probability that a node v_y will be reached in one step from a node v_x is given by the product of the degree centrality of v_x and the normalized edge weight (v_x, v_y). With each exercise as a starting node, we performed 100,000 random walks on the cosine-similarity graph, stepping with proportional probability to all outgoing cosine similarity weights. To measure 2nd degrees of separation, with each walk we made two steps.

For two math vectors considered as the sets A and B, cosine similarity can be conceptualized in terms of the intersection set $C = A \cup B$ and set differences $A \setminus B$ and $B \setminus A$. Cosine similarity is high when $|C| \gg A \setminus B$ and $|C| \gg B \setminus A$.

The degree of a node affects the probability of traversing any edge from that node. The two factors that affect degree centrality of a start node are the document frequencies of its math words, and the total number of math words. Here, document frequency (df) is the normalized number of exercises a word occurs in. A high df math word in a problem increase its degree centrality because there will be more

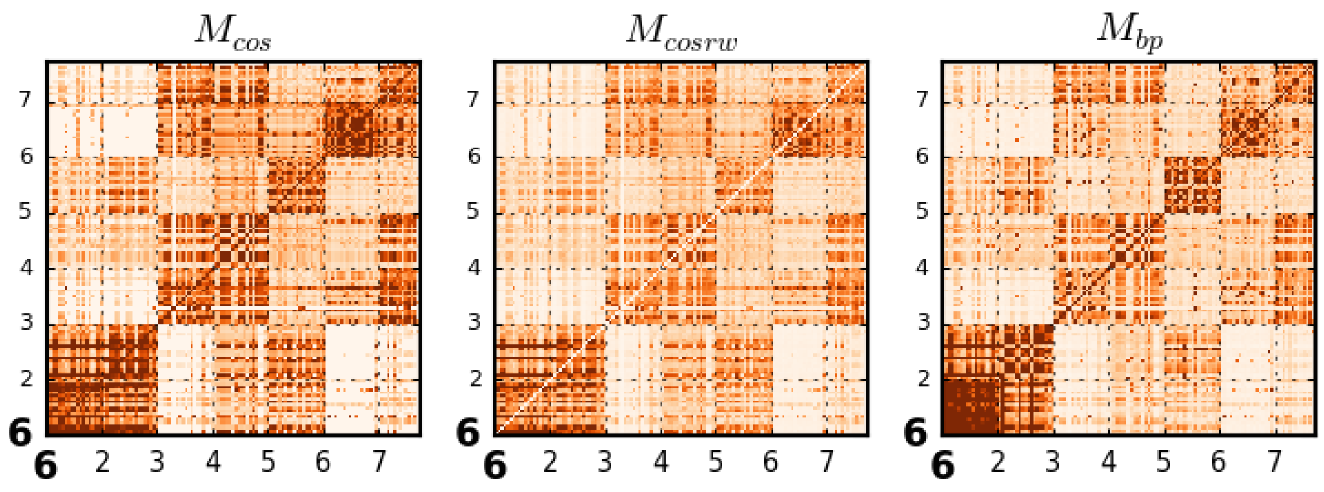


Figure 2: Exercise-to-Exercise similarity in Chapter 6. The exercises of Chapter 6 are displayed in row-columns in a square matrix. The rows represent source nodes and columns represent targets. Each row has been normalized across the book, even though only Chapter 6 is shown. The axes demarcate the sections of the chapter. M_{cos} is the cosine similarity. M_{cosrw} is the output from the random walk using cosine similarity edge weights, M_{bp} is the output from the random bipartite walk. Raw values displayed between 0 and 0.005 corresponding to light and dark pixels, respectively.

problems it can share words with, resulting in non-zero cosine values and therefore edges. The number of math words in a problem also increases its degree centrality.

3.5 Bipartite exercise and word graph

The set of exercises V_e are the left-side nodes and the math words V_w are the right-side nodes in the undirected bipartite graph $G = (V_e, V_w, E)$, where an edge exists between v_{ex} and v_{wi} if exercise x contains the math word i.

We performed open random walks on this graph to measure similarity between nodes. To measure the similarity of exercises, we walk in even steps – a step to a connected word followed by a step back to one of the exercises that shares that word. The degrees of separation between vertices on the same side of the graph (e.g. exercise-to-exercise) will be $l/2$ where l is the length of the walk. In this paper, we explored first and second degrees of separation so our bipartite graphs had a walk length of 4.

Table 1: Summary statistics of the similarity distributions

	cosine	rw$_{cos}$	rw$_{bp}$
minimum	0	0	0
maximum	6.3×10^{-2}	0.50	0.11
mean	5.55×10^{-4}	5.55×10^{-4}	5.55×10^{-4}
median	0	2.41×10^{-4}	2.06×10^{-4}
std. dev.	1.19×10^{-3}	8.57×10^{-4}	1.24×10^{-3}

Because exercise nodes are connected via word nodes, we interpret the fraction of node visits as a similarity measure between the source node and any node visited. We performed 100,000 random walks from each node. Exercise-to-exercise similarity can be visualzed as square matrices with source nodes in the rows and target nodes in the columns. To factor out the times a source may have been selected as one of the targets, we set the diagonal of the matrix to zero. We then normalized across the rows so that we could interpret the distribution across the row as a probability distribution to all other nodes for that source node.

4. RESULTS

We compare the three measures of similarity between exercises: 1) cosine similarity, 2) random walks using cosine similarity as edge weights, and 3) random walks along a bipartite graph of exercises and words.

4.1 Exercise-to-Exercise Similarity

We describe exercise-to-exercise similarity with square matrices where each exercise is represented as a row-column. A number of features of the measures are embedded in Figure 2, which shows heatmaps of color values for pairs of exercises in chapter 6 for each matrix. We find that within chapters and especially within sections of those chapters, there is a high degree of similarity between exercises regardless of the measure. This demonstrates that words within sections and chapters share a common vocabulary. We can see that M_{cos} has more extreme values than M_{cosrw}; as explained below, it has both more zero cosine values, and more very high values. This is most likely because M_{cosrw}, from doing the walk, picks up exercises that are another degree of separation away. When the row of the matrix is normalized to capture the distribution of the source node, the otherwise high values from M_{cos} are tempered in the M_{cosrw} matrix. This shift to a large number of lower scores is shown in the bottom panel of Figure 3. M$_{bp}$ and M$_{cosrw}$ are very similar, but M$_{bp}$ generally has a wider dynamic range.

4.2 Comparison of the Graph Walks

Table 1 provides summary statistics for cosine similarity and the two random walks for all pairs of problems (N=3,250,809). The cosine matrix is very sparse, as shown by the median value of 0. Of the two random walk similarities, rw$_{cos}$ has a lower standard deviation around the mean, but otherwise the two random walks produce similar distributions.

The similarity values given by cosine and the cosine random walk will increasingly differ the more that the start problem has relatively higher degree centrality due either to more words or higher frequency of words in exercises (df). For reference, the word that occurs most frequently, *number*, has a df of 0.42, and the second most frequent occurs in only 15% of the exercises. Fifty eight nodes have no edges (0 degree), the most frequent number of edges is 170, and the maximum is 1,706. Table 2 gives the summary statistics for df, number of math words, and degree centrality.

Inspection of the data shows that for pairs of problems in the two chapters for our case study, if the cosine similarity between a pair is high (≥ 0.75), the similarity values for rw_{cos} tend to go down as the number of shared word increases from 3 to between 5 and 7. For the rw_{bp}, the opposite trend occurs, where the similarity goes up as the number of words increases. This difference helps account for an observed divergence in the two graph walks for sections 5 and 6 of Chapter 6.

Table 3 illustrates two pairs of problems from section 5 that have high cosine similarities, and relatively higher rw_{bp} similarities (greater than the rw means of 0.0055) and relatively lower rw_{cos} (lower than the rw means). The reverse pattern is seen for two pairs of problems from section 6 that have high cosine similarities. These problems have higher than average rw_{cos} and lower than average rw_{bp}. What differentiates the two pairs of problems is that the section 5 problems have a relatively large number of words in common: 14 for the first pair, 12 for the second pair. In both pairs, some of the words have relatively high document frequency. As discussed above, these two properties increase the degree centrality of the start node of a step in the rw_{cos} graph, and thus lower the probability of hitting each of the start node's one-degree neighbors. This effect propagates along the two steps of the walk. For the rw_{bp} graph, however, as the number of shared math words for a pair of problems increases, the number of paths from one to the other also increases, thus raising the probability of the traversal. This effect also propagates through a two-step walk. In contrast to the section 5 problems, the two section 6 problems have relatively fewer words in common: 3 for both pairs.

For problem pairs where the cosine similarity is between 0.40 and 0.60, the mean similarity from rw_{bp} is 30% higher than for rw_{cos} for when the number of math words in common is 3 (0.0033 vs. 0.0043), 80% higher when the number of math words in common is 6 (0.0024 versus 0.0045), and three as high when the number of math words in common is 9 (0.0023 versus 0.0068). For problems pairs where the cosine similarity is less than 0.20, the two walks produce very similar results. The average similarity values for the bipartite walk are about 20% higher, and the maximum values are higher, but the two walks produce similar means, independent of the lengths of the common word vectors, or the total number of math words.

Table 2: Summary statistics of document frequency (df) of math words, number of math words in problems, and degree centrality of rw_{cos}

	df	math words	degree ctr. rw_{cos}
minimum	5.54×10^{-4}	1	0
maximum	0.424	24	1,706
mean	0.183	8.35	418.4
median	6.66×10^{-3}	8.00	340
std. dev.	0.314	3.64	317.1

Table 3: Two pairs of problems with high cosine similarity and reverse patterns of graph walk similarity. The first pair, from section 5, have lower than average rw_{cos} and higher than average rw_{bp} due to relatively many words in common (12 and 14). The second pair, from section 6, have higher than average rw_{cos} and lower than average rw_{bp} due to relatively few words in common.

Prob 1	Prob 2	cosine	rw_{cos}	rw_{bp}	N	max df
6.5.99	6.5.85	1.0000	0.0032	0.0102	12	0.42
6.5.94	6.5.83	0.8819	0.0026	0.0064	14	0.13
6.6.109	6.6.102	0.8660	0.0068	0.0037	3	0.11
6.6.104	6.6.102	0.7746	0.0068	0.0029	3	0.11

Since we normalized the matrices across rows, which are the source nodes, differences between the bipartite matrix, M_{bp}, and the cosine matrices implied that the degree of the target node had a greater impact on the variability in the bipartite matrix. To measure the impact of the edge degree on the target nodes, we considered the column sum for those targets that had 1 edge, those that had 2, etc. up to 20 edges. The results are summarized in Figure 4. As can be seen, the column sum varies linearly by the number of target edges in the bipartite matrix, whereas the cosine matrices do not. We found the cubed root of the column sum in M_{bp} approaches the distribution of column sums of the cosine matrices, which is provided in Figure 4.

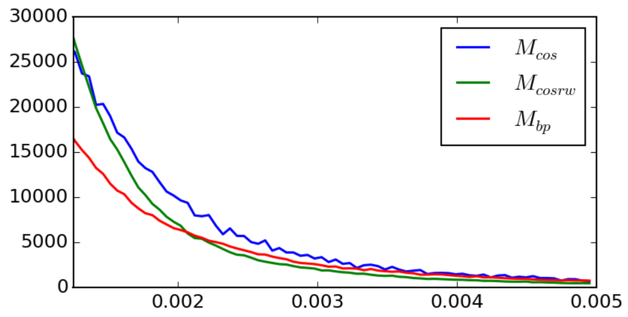


Figure 3: Tail distribution of similarity values in M_{cos}, M_{cosrw}, and M_{bp}. Because 62% of the values in M_{cos} are 0, the plot shows only non-zero values.

5. CONCLUSION

Visualization of the three similarity matrices shows they reveal the same overall patterns, thus each is confirmed by the others. However, the bipartite walk was the most sensitive to word frequency across exercises, and the number of words in problems. With our goal of automatically discovering knowledge components and identifying their relationships, the random walk that stepped in proportion to its cosine similarity performed best. It was able to discover second-degree relationships that seem reasonable as we explore by eye those matches. Future work will test these re-

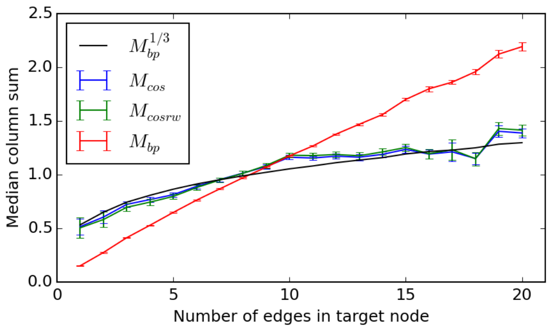


Figure 4: Distribution of column sums by number of edges in the target node represented by the column. Error plots show the mean and standard error for each type. Black line is the cubed root of the mean of the column sums of M_{bp}.

lationships with student performance data. We should find, for example, that if two exercises are conceptually similar, then student outcomes should also be similar and learning curves should reveal shared knowledge components. In this respect, such automatically constructed knowledge graphs can create more refined domain models that intelligent tutoring systems and robust assessments can be built upon.

6. REFERENCES

[1] Y. An, J. Janssen, and E. E. Milios. Characterizing and mining the citation graph of the computer science literature. *Knowledge and Information Systems*, 6(6):664–678, 2004.

[2] J. Berant, V. Srikumar, P.-C. Chen, A. Vander Linden, B. Harding, B. Huang, P. Clark, and C. D. Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1499–1510, Doha, Qatar, October 2014. Association for Computational Linguistics.

[3] P. Buitelaar, A. Frank, M. Hartung, and S. Racioppa. Ontology-based information extraction and integration from heterogeneous data sources, 2008.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. H. Jr., and T. M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*, volume 2, pages 1306–1313, 2010.

[5] G. Erkan and D. R. Radev. Lexrank: Graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, Dec. 2004.

[6] B. Hixon, P. Clark, and H. Hajishirzi. Learning knowledge graphs for question answering through conversational dialog. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, CO, May-June 2015.

[7] R. J. L. John, R. J. Passonneau, and T. S. McTavish. Semantic similarity graphs of mathematics word problems: Can terminology detection help? In *Proceedings of the Eighth International Conference on Educational Data Mining*, 2015.

[8] K. R. Koedinger, A. T. Corbett, and C. Perfetti. The knowledge-learning-instruction (KLI) framework: Toward bridging the science-practice chasm to enhance robust student learning. *Cognitive Science*, 36(5):757–798, 2012.

[9] K. Krippendorff. *Content analysis: An introduction to its methodology*. Sage Publications, Beverly Hills, CA, 1980.

[10] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[11] N. Li, W. W. Cohen, and K. R. Koedinger. Discovering student models with a clustering algorithm using problem content. In *Proceedings of the 6th International Conference on Educational Data Mining*, 2013.

[12] E. Minkov, W. W. Cohen, and A. Y. Ng. Contextual search and name disambiguation in email using graphs. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 27–34, New York, NY, USA, 2006. ACM.

[13] I. Niles and A. Pease. Mapping WordNet to the SUMO ontology. In *Proceedings of the IEEE International Knowledge Engineering Conference*, pages 23–26, 2003.

[14] J. Otterbacher, G. Erkan, and D. R. Radev. Biased lexrank: Passage retrieval using random walks with question-based priors. *Information Processing and Management*, 45(1):42–54, 2009.

[15] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

[16] T. M. Pilehvar, D. Jurgens, and R. Navigli. Align, disambiguate and walk: A unified approach for measuring semantic similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1341–1351. Association for Computational Linguistics, 2013.

[17] J. Scheffczyk, A. Pease, and M. Ellsworth. Linking FrameNet to the suggested upper merged ontology. In B. Hennett and C. Fellbaum, editors, *Formal Ontology in Information Systems*, pages 289–. IOS Press, 2006.

[18] M. J. Seo, H. Hajishirzi, A. Farhadi, and O. Etzioni. Diagram understanding in geometry questions. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, 2014.

[19] M. Šimko and M. Bieliková. Automatic concept relationships discovery for an adaptive e-course. In *Proceedings of the Second International Conference on Educational Data Mining (EDM)*, pages 171–179, 2009.

Workshop

on

Tools and Technologies in Statistics, Machine Learning and Information Retrieval for Educational Data Mining

Workshop on Tools and Technologies in Statistics, Machine Learning and Information Retrieval for Educational Data Mining (SMLIR@EDM2015)

Within Intelligent Data Analysis (IDA) research area there may be found several subareas such as Machine Learning/Statistics, Information Retrieval, Data Mining and lately Big Data and Cloud Computing. The exact boundaries between these research areas are not very clear and may mislead research efforts, each one having its own particularities in terms of input type and size, data processing methodologies and obtained output. As Data Mining makes intensive use of all these research subareas, it is mandatory to be aware of their subtle differences and, therefore, design and implement IDA systems that make research efforts sound and effective. The goal of this workshop is to gather research efforts that fall into any of the categories of subareas and have results into the application area of Education. The workshop is looking for contributions that provide experimental results in the area of EDM/Information Retrieval and are focused on data processing fundamentals from Machine Learning/Statistics perspective. From the practical point of view, the focus should be on presenting the details regarding what and how tools and technologies are used in order to obtain relevant data analysis engines.

The integration of tools and technologies for building IDA processes is a key issue for developing applications that improve the effectiveness of the e-Learning platforms. The EDM community will benefit from the discussions related to the advantages and drawbacks of various options in a practical context with experimental results, by improving the efficiency of building high quality software systems supporting the research efforts.

The first step of developing an IDA application should focus on choosing the right tool or technology that fits the task requirements (e.g., input size, algorithm type, running time efficiency, scalability, etc.). The diversity of the available options is an indication of the necessity for a detailed analysis. From this point of view, the EDM community needs to be aware of success and failure attempts of many practical research efforts in order to provide the possibility of a proper future design choice.

Existing tools and technologies implement in different ways recent advances on techniques from statistical/machine learning, information retrieval and data mining domains in terms of programming language (e.g., Java, C/C++, C#, R, etc.), toolkits (e.g., Weka, Apache Mahout, MLTK, Maple, Matlab, etc.) and implementation details that may have a great impact on running times, scalability, effectiveness or efficiency.

This workshop brings together researchers from academia and industry practitioners with special interest in statistics/machine learning, information retrieval, data mining to (1) discuss current state of the art tools and technologies, (2) identify patterns for proper usage of various options for different tasks, and (3) lay out a vision regarding the modality in which tools and technologies will influence future applications. The organizers hope to obtain common background knowledge for integrating various tools and technologies in future EDM applications.

I gratefully acknowledge the following members of the workshop program committee:
Ivan Lukovic, University of Novi Sad, Serbia
Lucian Vintan, "Lucian Blaga" University of Sibiu, Romania
Stefan Trăuşan-Matu, University “Politehnica” of Bucharest, Romania
Andrea Zielinski, Fraunhofer IOSB, Karlsruhe, Germany
Mihai Lupu, Vienna University of Technology, Austria

Vladimir Fomichov, National Research University Higher School of Economics (HSE), Moscow, Russian Federation
Costin Bădică, University of Craiova, Romania
Innar Liiv, Tallinn University of Technology, Estonia
Adrian Graur, University of Suceava, Romania
Vladimir Ivančević, University of Novi Sad, Serbia
Vladimir Cretu, University “Politehnica” of Timisoara, Romania
Mihai Garboveanu, University of Craiova, Romania
Ivan Chorbev, Ss. Cyril and Methodius University Skopje, R. of Macedonia
Ion Iancu, University of Craiova, Romania
Liana Stănescu, University of Craiova, Romania
Marius Brezovan, University of Craiova, Romania
Urszula Markowska-Kaczmar, Wroclaw University of Technology, Poland

SMLIR Workshop Chair
Marian Cristian Mihăescu

Steering and Organizing Committee
Mihai Mocanu
Costel Ionascu
Mirel Cosulschi

Integrating an Advanced Classifier in WEKA

Paul Ştefan Popescu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
sppopescu@gmail.com

Mihai Mocanu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
mocanu@software.ucv.ro

Marian Cristian Mihăescu
Deparment of Computers and Information Technology
Bvd. Decebal no. 107
Craiova, Romania
mihaescu@software.ucv.ro

ABSTRACT

In these days WEKA has become one of the most important data mining and machine learning tools. Despite the fact that it incorporates many algorithms, on the classification area there are still some unimplemented features. In this paper we cover some of the missing features that may be useful to researchers and developers when working with decision tree classifiers. The rest of the paper presents the design of a package compatible with the WEKA Package Manager, which is now under development. The functionalities provided by the tool include instance loading, successor/predecessor computation and an alternative visualization feature of an enhanced decision tree, using the J48 algorithm. The paper presents how a new data mining/machine learning classification algorithm can be adapted to be used integrated in the workbench of WEKA.

Keywords

Classifier, J48, WEKA, Machine learning, Data Mining

1. INTRODUCTION

Nowadays huge amounts of data can be gathered from many research areas or industry applications. There is a certain need for data mining or knowledge extraction [6] from data. From this large amount of data, the data analysts gather many variables/features and many machine learning techniques are needed to face this situation. There are many application domains such as medical, economics (i.e., marketing, sales, etc.), engineering or in our case educational research area [16] in which machine learning techniques can be applied. Educational data mining is a growing domain [4] in which a lot of work has been done.

Because the application domains are growing continuously, the tools that support the machine learning processes must live up to market standards providing good performances and intuitive visualization techniques. In these days there are many tools that deal with a wide variety of problems. In order to be more explicit, we have tools like RapidMiner [12], KEEL [2], WEKA, Knime [3] or Mahout [13]. RapidMiner is a graphical drag and drop analytics platform, formerly known as YALE, which provides an integrated environment for data mining, machine learning, business and predictive analytics. Keel is an application package of machine learning software tools, specialized on the evaluation of evolutionary algorithms. KNIME, the Konstanz Information Miner, is a modular data exploration platform, provided as an Eclipse plug-in, which offers a graphical workbench and various components for data mining and machine learning. Mahout is a highly scalable machine learning library based on the Hadoop framework [18], an implementation of the MapReduce programming model, which supports distributed processing of large data sets across clusters of computers.

For our approach we choose WEKA because it has become one of the most popular machine learning and data mining workbenches and its success is due to its constant improvement and development. Moreover, WEKA is a very popular tool used in many research domains, widely adopted by the educational data mining communities.

WEKA is developed in Java and encapsulates a collection of algorithms that tackle many data mining or machine learning tasks like preprocessing, regression, clustering, association rules, classification and also visualization techniques. In some cases, these algorithms are referring only the basic implementation.

One aspect that needs to be taken into consideration is that WEKA has a package manager which simplifies the developers contribution process. There are two kind of packages that can be installed in WEKA and used via the application interface: official and unofficial packages. This is a very important feature because if there is an algorithm that fits your problem description and there is a package for it you can just add it to the application and use it further. Moreover, you don't need to be a programmer to do that, you don't need to write code, just install the package and then use the algorithm like it had been there forever.

According to the real life experiences, many of the included algorithms can hardly be used because of their lack of flexibility. For example, in standard decision trees from WEKA we can perform a classification process but we cannot access a particular instance from the tree. Suppose that we have a training data file and we create the tree model. When we try to see where is the place of the instance “X” in the tree we can’t do that in the application interface, neither when you add the WEKA library in your code. This is a big drawback because retrieving the leaf to which the instance belongs to provides more information than retrieving its class. Usually, when performing a classification task, the data analyst divides test instances into classes that have little meaning from application domain of perspective.

In a real life scenario in a training dataset we may have a large number of features describing the instances. A data analyst should be able to parse a decision tree, see the rule that derived to a specific decision and then draw very accurate conclusions In this paper we will address classification and visualization issues by adding new functionalities and improving the decision tree visualization.

Several classification algorithms have been previously contributed to WEKA but non of them is able to output a data model that is loaded with instances. Based on the previous statement it is clear that there aren’t WEKA visualization techniques that are able to present the data in the model in a efficient way and also, there are no available parsing methods ready to implement such functionalities. Traversal of leaves is another task that is missing and it is important because instances from neighbour leaves have a high degree of similarity and share many attributes with similar values.

One aspect that differs at WEKA from other similar software regards its architecture that allows developers to contribute in a productive way. All the work that needs to be done refers to creating a specific folders layout, completing a “description.props” file, adding the “.jar” file to the archive and the build script.

2. RELATED WORK

WEKA is a open source machine learning library that allows developers and researchers to contribute very easily. There are more than twenty years since WEKA had it’s first release [9] and there were constant contributions added on it. Not only machine learning algorithms were implemented, for example, in 2005 a text data mining module was developed [20]. An overview of the actual software was made in [8].

Several classifiers were developed and contributed as packages to WEKA. In 2007 a classifier that was build based on a set of sub-samples was developed [14] and compared to C4.5 [15] which have it’s implementation called J48 [11] in WEKA. Other classifiers refers the “Alternating Decision Trees Learning Algorithms” [7] which is a generalization of the decision trees, voted decision trees and voted decision stumps. This kind of classifiers are relatively easy to interpret and the rules are usually smaller in size. Classical decision trees, such as c4.5 were expanding nodes in a depth-first order; an improvement came from “Best-first decision trees” [17]which expands nodes in a best-first order. A package with these trees was contributed to WEKA.

Some other contributions refers libraries of algorithms that can be accessed via WEKA. One of them is JCLEC [5] an evolutionary computation framework which has been successfully employed for developing several evolutionary algorithms. Other environment for machine learning and data mining knowledge discovery that was contributed to WEKA is R [10]. This contribution was developed in order to include different sets of tools from both environments available in a single unified system.

Also as related work we must take into consideration some of the last algorithms development. In the last year it is presented a new fast decision tree algorithm [19]. Based on their experiments, the classifier outperforms C5.0 which is the commercial implementation of C4.5.

3. SYSTEM DESIGN

The package is designed to be used both by developers, in their Java applications, and researchers, using the WEKA Explorer. At the moment of writing this paper the package with the Advanced Classifier is still under development, offering more functionalities as a tool for developers than in the explorer view of WEKA.

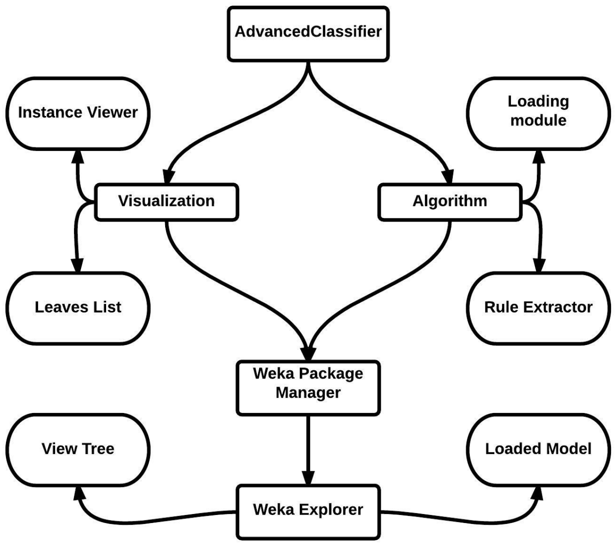


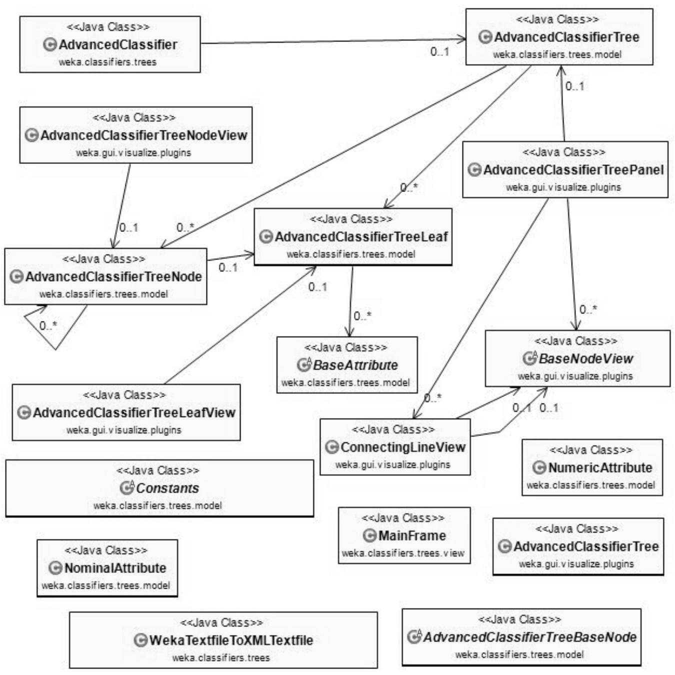
Figure 1: Package Integration in WEKA

In Fig. 1 we present the main design of the algorithm and how it can be used in WEKA. On the top of the figure we have the classifier which can be divided in two main modules: the algorithm and the visualization. As we can see on the next level, both of the modules can be divided further. All the functionalities are then installed in WEKA via the package manager and then, in the explorer, we can perform data analysis tasks using a model loaded with data and it’s associated visualization techniques.

3.1 General Architecture

The packages is a zip archive, structured with respect to the WEKA guidelines. That is, it unpacks to the current directory and it contains: the source files, a folder with the required libraries, a build script, a properties file required by WEKA for installing and managing the package, and the actual “.jar” file. A detailed structure of the package is presented below.

```
<current directory>
  +-AdvancedClassifier.jar
  +-Description.props
  +-build_package.xml
  +-src
  |  +-main
  |    +-java
  |      +-resources
  |      |  +-background_node.png
  |      |  +-background_leaf.png
  |      |  +-background_leaf_pressed.png
  |      |  +-font_node.ttf
  |      |  +-font_decision.ttf
  |      +-weka
  |          +-classifiers
  |          |  +-trees
  |          |    +-model
  |          |    |  +-AdvancedClassifierTree.java
  |          |    |  +-AdvancedClassifierTreeBaseNode.java
  |          |    |  +-AdvancedClassifierTreeNode.java
  |          |    |  +-AdvancedClassifierTreeLeaf.java
  |          |    |  +-BaseAttributeValidator.java
  |          |    |  +-NominalAttributeValidator.java
  |          |    |  +-NumericAttributeValidator.java
  |          |    |  +-Constants.java
  |          |    +-AdvancedClassifier.java
  |          |    +-WekaTextfileToXMLTextfile.java
  |          +-gui
  |             +-visualize
  |                +-plugins
  |                   +-AdvancedClassifierTree.java
  |                   +-AdvancedClassifierTreePanel.java
  |                   +-BaseNodeView.java
  |                   +-AdvancedClassifierTreeNodeView.java
  |                   +-AdvancedClassifierTreeLeafView.java
  |                   +-ConnectingLineView.java
  +-lib
    +-weka.jar
    +-simple-xml.jar
    +-rt.jar
```



<<Java Class>>
AdvancedClassifier
weka.classifiers.trees
<<Java Class>>
AdvancedClassifierTree
weka.classifiers.trees.model
0..1
<<Java Class>>
AdvancedClassifierTreeNodeView
weka.gui.visualize.plugins
0..1
<<Java Class>>
AdvancedClassifierTreePanel
weka.gui.visualize.plugins
0..*
<<Java Class>>
AdvancedClassifierTreeLeaf
weka.classifiers.trees.model
0..1
0..*
<<Java Class>>
AdvancedClassifierTreeNode
weka.classifiers.trees.model
0..1
0..1
0..*
0..*
<<Java Class>>
BaseAttribute
weka.classifiers.trees.model
0..*
<<Java Class>>
BaseNodeView
weka.gui.visualize.plugins
<<Java Class>>
AdvancedClassifierTreeLeafView
weka.gui.visualize.plugins
0..*
0..1
0..1
<<Java Class>>
ConnectingLineView
weka.gui.visualize.plugins
<<Java Class>>
NumericAttribute
weka.classifiers.trees.model
<<Java Class>>
Constants
weka.classifiers.trees.model
<<Java Class>>
MainFrame
weka.classifiers.trees.view
<<Java Class>>
AdvancedClassifierTree
weka.gui.visualize.plugins
<<Java Class>>
NominalAttribute
weka.classifiers.trees.model
<<Java Class>>
WekaTextfileToXMLTextfile
weka.classifiers.trees
<<Java Class>>
AdvancedClassifierTreeBaseNode
weka.classifiers.trees.model

Figure 2: Class Diagram

In Figure 2 is presented the system's class diagram. This diagram includes all the java packages from the project and their relations. As we can see in the above mentioned figure, we have two type of classes: independent classes and composed. Independent classes are gathered from the model part of the Model-View-Controller architecture or just classes that perform one time tasks like "WekaTextFileToXMLTextFile" which is able to generate an XML based on the text file outputted by WEKA. On the other side, the composed classes are dependent on each other and these relations are shared across packages. One important class that is worth to be mentioned is "AdvancedClassifierTreeLeaf.java" in which we store the leaves of our tree along with rules that define the leaf. Discussions about implementation of the packages are more related to the software engineering research area and beyond of the scope of this paper.

3.1.1 Design and Implementation of the Algorithm

The algorithm needs to generate custom rules (dependent on the training dataset) for every leaf of the decision tree. These rules are computed by tracing the path from the root of the tree to the specified leaf. Each decision that leads to a leaf is therefore translated into a rule that encapsulates the name of the attribute and the value on which the decision was made. For each type of attribute defined by WEKA, we need to have a corresponding rule that matches that type. For this purpose an abstract class has been created to act as a base class for any of the custom rules. The name of this class is "BaseAttributeValidator" and exposes the required methods that a superclass needs to implement: a "clone" method required by the workflow of the system and methods that validate if an instance or set of instances have the required values of the attribute targeted by the rule. At the moment, the only implemented rules are the ones that handle "NOMINAL" and "NUMERIC" attribute types.

The rule that validates each nominal attribute is called "NominalAttributeValidator" and receives as parameters the name of the targeted attribute and a string variable representing the accepted value of the attribute. The rule that handles the numeric attributes is called "NumericAttributeValidator" and also receives the name of the attribute and either a particular value or the boundaries of an interval.

In the following paragraphs, we present a brief overview of the algorithm for which we adopt a straightforward approach.

Firstly, the algorithm retrieves instances from the ".arff" file using the methods provided by WEKA. The next step is applying the desired classification process. Currently the only supported classifier is J48, but employing other decision tree classifiers is foreseen as future work. Using the text representation of the outputted model and a predefined set of rules and tags, an XML is then generated. This is an important step during the workflow because the structured XML format allows us to obtain the base model for our decision tree. The deserialization is done using a third-party Java library("Simple XML" [1]).

The model obtained this way contains a list of nodes and leaves with the following significance: each node corresponds to a decision in the tree; the data stored in each object (node) refers the information about the name of the actual attribute, operator and value on which the decision was made; and the results to which making the decision leads (a list of other nodes or an output leaf). Using this model and the set of attributes provided by WEKA, the set of rules is computed. This step is performed by parsing the model from the first node (i.e., the root) to the last available leaf and gradually composing the set of rules that defines each leaf. The setup of the algorithm is finally completed with the loading of the training dataset into the model.

The classifier and processed data can now be easily handled and different operations can be applied. The method currently implemented include basic per leaf manipulation of instances, i.e. loading new instances into the model and retrieving the part of the dataset contained in each leaf, as well as predecessor and successor computation.

3.1.2 *Visualization Plugin*

For the visualization feature, a custom panel has been designed to hold the components that build up the decision tree and expose the data available in the leaves. The constructor of the panel requires the decision tree model as a parameter, and takes care of adding the corresponding views to the interface. In order to include this functionality in WEKA, a specialized class that implements WEKA's TreeVisualizePlugin interface has been created. After adding the package through the Package Manager and selecting this visualization option, a new JFrame that holds the custom panel is displayed.

```
@RELATION StudentClass

@ATTRIBUTE userid numeric
@ATTRIBUTE nrHours numeric
@ATTRIBUTE avgMark numeric
@ATTRIBUTE present {NO,YES}
@ATTRIBUTE class {low,high}

@DATA
4,17,6.025,NO,low
5,11,7.0,YES,low
7,30,7.375,NO,low
8,10,7.714286,NO,low
9,43,5.1666665,YES,high
10,31,4.5,YES,high
```

Figure 3: Sample from the Dataset

In Figure 3 we present a dataset sample. In order to validate the classifier and it's extra functionalities several tests have been made but for this case study we used three attributes and 788 instances. The feature called "userid" doesn't provide any information gain but can be easily used for instances localization in leaves. The attributes significance is beyond the scope of this paper.

In Figure 4 is presented a screen-shot of the tree generated based on the dataset from figure 3. Each node contains

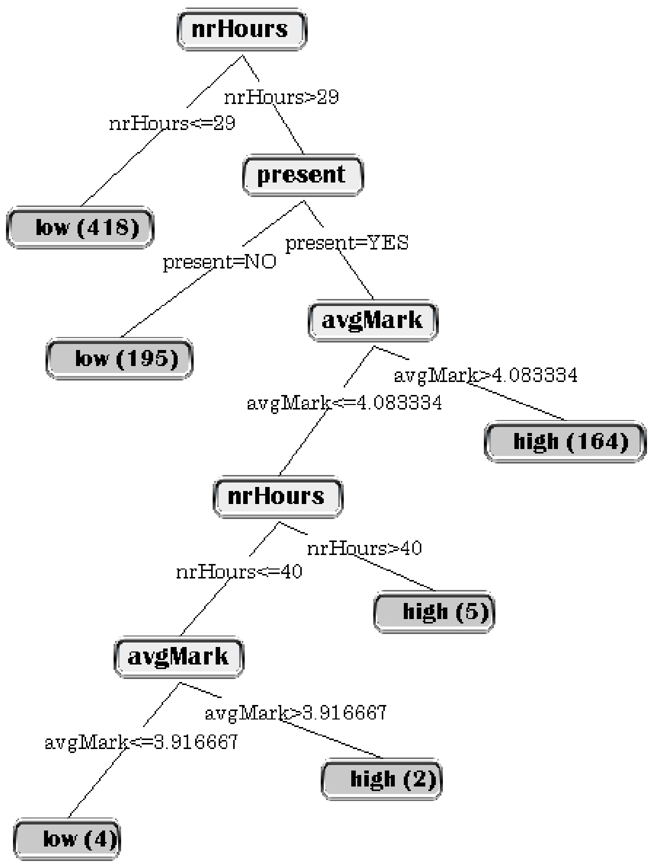


Figure 4: Tree Sample

the name of the attribute, and each decision is printed on top of the connecting line. Surely, each leaf can be clicked, and the set of enclosed instances is displayed. As previously noted, there is still some work to be made to finalize the development of the package, and the visualization tool needs to be included as well. Efforts will have to be made toward providing the means to visualize and handle the successors/predecessors, outliers and other relevant information.

4. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the integration of a data analysis tool in WEKA. This tool is important because brings a new classifier to WEKA that aims to improve the classification procedures. Here, are also presented some implementing procedures and details.

A workflow is also described and all the mechanism that is used to bring new features for the users. One important thing that needs to be mentioned is that the data loading module opens new data analysis opportunities for researchers.

As future work we plan to implement Other types of attributes supported by WEKA like "DATE", "String" and "Relational".

5. REFERENCES

[1] Simple xml. http://simple.sourceforge.net.

[2] J. Alcalá-Fdez, L. Sánchez, S. García, M. del Jesus, S. Ventura, J. Garrell, J. Otero, C. Romero, J. Bacardit, V. Rivas, J. FernÃąndez, and F. Herrera. Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Computing*, 13(3):307–318, 2009.

[3] M. R. Berthold, N. Cebron, F. Dill, T. R. Gabriel, T. Kötter, T. Meinl, P. Ohl, K. Thiel, and B. Wiswedel. Knime - the konstanz information miner: Version 2.0 and beyond. *SIGKDD Explor. Newsl.*, 11(1):26–31, Nov. 2009.

[4] R. Campagni, D. Merlini, R. Sprugnoli, and M. C. Verri. Data mining models for student careers. *Expert Systems with Applications*, (0):–, 2015.

[5] A. Cano, J. M. Luna, J. L. Olmo, and S. Ventura. Jclec meets weka! In E. Corchado, M. Kurzynski, and M. Wozniak, editors, *HAIS (1)*, volume 6678 of *Lecture Notes in Computer Science*, pages 388–395. Springer, 2011.

[6] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. The kdd process for extracting useful knowledge from volumes of data. *Commun. ACM*, 39(11):27–34, Nov. 1996.

[7] Y. Freund and L. Mason. The alternating decision tree learning algorithm. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 124–133, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[8] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[9] G. Holmes, A. Donkin, and I. H. Witten. Weka: a machine learning workbench. pages 357–361, August 1994.

[10] K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: R meets weka. *Computational Statistics*, 24(2):225–232, 2009.

[11] W.-Y. Loh. *Classification and Regression Tree Methods*. John Wiley & Sons, Ltd, 2008.

[12] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. Yale: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '06, pages 935–940, New York, NY, USA, 2006. ACM.

[13] S. Owen, R. Anil, T. Dunning, and E. Friedman. *Mahout in Action*. Manning Publications Co., Greenwich, CT, USA, 2011.

[14] J. M. Pérez, J. Muguerza, O. Arbelaitz, I. Gurrutxaga, and J. I. Martín. Combining multiple class distribution modified subsamples in a single tree. *Pattern Recognition Letters*, 28(4):414–422, 2007.

[15] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[16] C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, 33(1):135 – 146, 2007.

[17] H. Shi. Best-first decision tree learning. Technical report, University of Waikato, 2007.

[18] K. Shvachko, H. Kuang, S. Radia, and R. Chansler. The hadoop distributed file system. In *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, pages 1–10. IEEE, 2010.

[19] S.-G. P. V. Purdila. Fast decision tree algorithm. *Advances in Electrical and Computer Engineering*, 14(1):65–68, 2014.

[20] I. H. Witten, G. W. Paynter, E. Frank, C. Gutwin, and C. G. Nevill-Manning. Kea: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM Conference on Digital Libraries*, DL '99, pages 254–255, New York, NY, USA, 1999. ACM.

A Systematic Mapping Study on the Usage of Software Tools for Graphs within the EDM Community

Vladimir Ivančević*
University of Novi Sad, Faculty of Technical Sciences
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
dragoman@uns.ac.rs

Ivan Luković
University of Novi Sad, Faculty of Technical Sciences
Trg Dositeja Obradovića 6
21000 Novi Sad, Serbia
ivan@uns.ac.rs

ABSTRACT

The field of educational data mining (EDM) has been slowly expanding to embrace various graph-based approaches to interpretation and analysis of educational data. However, there is a great wealth of software tools for graph creation, visualization, and analysis, both general-purpose and domain-specific, which may discourage EDM practitioners from finding a tool suitable for their graph-related problem. For this reason, we conducted a systematic mapping study on the usage of software tools for graphs in the EDM domain. By analysing papers from the proceedings of previous EDM conferences we tried to understand how and to what end graph tools were used, as well as whether researchers faced any particular challenges in those cases. In this paper, we compile studies that relied on graph tools and provide answers to the posed questions.

Keywords

Systematic Mapping Study, Graphs, Software Tools, Educational Data Mining.

1. INTRODUCTION

The field of educational data mining (EDM) has significantly expanded over the past two decades. It has attracted numerous researchers with various backgrounds around the common goal of understanding educational data through intelligent analysis and using the extracted knowledge to improve and facilitate learning, as well as educational process. In 2010, Romero and Ventura published a comprehensive overview of the field with 306 references [26]. In this review, the authors identified 11 categories of educational tasks, two of which dealt with graph structures (for brevity these will be referred to as graphs): social network analysis (SNA) and developing concept maps. However, the authors noted that these two categories featured a lower number of papers (15 or less references collected). Somewhat different categories of work were presented in another review of EDM [2] but they did not include any explicit references to graphs.

However, since that time, the interest in approaches and technologies utilizing graphs has increased within EDM. In addition to the results of a literature search on the topic, this could be also evidenced by the appearance of the Workshop on Graph-Based Educational Data Mining (G-EDM)[1] in 2014. As a result, software tools that help researchers or any other user group to utilize graphs or graph-based structures (for brevity these will be referred to as graph tools) are becoming a valuable resource for both the G-EDM and the broader EDM community. As graphs are only slowly gaining wider recognition in EDM, there could still be a lot of questions about which graph tools exist or what educational tasks might be supported by these tools.

In an attempt to help EDM researchers discover more useful information about potentially suitable graph tools, we reviewed the papers presented at the past EDM conferences, selected those that mentioned any usage of graph tools, and extracted from them information about which graph tools the authors employed, what features of these tools were used, to what end the research in question was conducted, and if there were any particular challenges while using these tools.

The present study may be classified as a secondary study since we base our approach on collecting other research works and assembling relevant information from them. Secondary studies might be more typical of medical and social sciences but there are proposed methodologies concerning secondary studies in software engineering as well [13]. Two kinds of secondary studies might be particularly important in this context: systematic review studies and systematic mapping studies [20]. In both cases, there is a clear methodology that is set to reduce bias when selecting other research works, which gives these secondary studies the quality of being systematic. Some of the differences pointed out by Petersen et al. [20] are that systematic reviews tend to focus on the quality of reviewed studies with the aim of identifying best practices, while systematic maps focus more on classification and thematic analysis but with less detailed evaluation of collected studies. Moreover, the same authors consider that the two study types form a continuum, which might complicate some attempts at categorization.

We categorize the present study as a systematic mapping study. This classification is justified by the fact that:

1. we employed a concrete methodology,
2. we did not evaluate the quality of collected papers or the presented results, but
3. we focused on identifying the employed graph tools and the manner in which these tools were used, with the aim

*Corresponding Author

[1] http://ceur-ws.org/Vol -1183/

of providing an overview of the current practice of using graph tools within the EDM community

However, we did not restrict our investigation to analysing exclusively titles, abstracts, or keywords, but went through the complete texts to find the necessary information. This aspect might better suit systematic reviews, but it does not change the principal goal or character of our study.

The exact details of the employed methodology, including the research questions, sources of studies, and study selection criteria, are given in Section 2. Section 3 contains the answers to the research question, most importantly the list of identified graph tools and the trends in their usage in EDM. Section 4 covers the potential limitations of the present study.

2. METHODOLOGY

We mainly followed the guidelines given in [20] but also relied on the example of a mapping study presented in [21]. Given the specificity of our study and the posed research questions, there were some necessary deviations from the standard suggested procedure. The overall process of selecting papers and extracting information, together with the resolution methods for non-standard cases, is presented and discussed in the following subsections.

2.1 Overview

The first step was defining research questions to be answered by the present study. The choice of research questions influenced the subsequent steps: conducting the search for papers, screening the papers, devising the classification scheme, extracting data, and creating a map.

2.2 Research Questions

We defined four principal research questions (RQ1-RQ4) concerning the use of graphs and graph tools in studies by EDM researchers:

- RQ1: Which graph tools were directly employed by researchers in their studies?
- RQ2: Which features of the employed graph tools were used by researchers?
- RQ3: What was the overall purpose of the research that involved or relied on graph tools?
- RQ4: What features did researchers consider to be missing or inadequate in the employed graph tools?

2.3 Search for Papers

We searched through all the papers that were published in the proceedings of the EDM conference series till this date, i.e., papers from the first EDM conference in 2008 to the latest, seventh, EDM conference in 2014. The latest EDM conference was special because it also included four workshops (G-EDM being one of them) for the first time. The papers from these workshops were also considered in our search. This amounted to eight relevant conference proceedings that represented the complete source of research works for our study:

1. Proceedings of the 1st International Conference on Educational Data Mining 2008 (Montreal, Canada)
2. Proceedings of the 2nd International Conference on Educational Data Mining 2009 (Cordoba, Spain)
3. Proceedings of the 3rd International Conference on Educational Data Mining 2010 (Pittsburgh, Pennsylvania, USA)
4. Proceedings of the 4th International Conference on Educational Data Mining 2011 (Eindhoven, Netherlands)
5. Proceedings of the 5th International Conference on Educational Data Mining 2012 (Chania, Greece)
6. Proceedings of the 6th International Conference on Educational Data Mining 2013 (Memphis, Tennessee, USA)
7. Proceedings of the 7th International Conference on Educational Data Mining 2014 (London, UK)
8. Extended Proceedings of the 7th International Conference on Educational Data Mining 2014 (London, UK), which included only the workshop papers

All the proceedings are freely offered as PDF files by the International Society of Educational Data Mining[2] and may be accessed through a dedicated web page.[3]

The papers from these proceeding represented our Level 0 (L0) papers, i.e., the starting set of 494 papers. This set included different categories of papers: full (regular) papers, short papers, different subcategories of posters, as well as works from the young researcher track (YRT) or demos/interactive events. The starting set did not include abstracts of invited talks (keynotes), prefaces of proceedings, or workshop summaries.

These papers were then searched and evaluated against our keyword criterion (KC), which led to a set of Level 1 (L1) papers. Our keyword string is of the form KC1 AND KC2 where KC1 and KC2 are defined in the following manner:

- KC1: graph OR subgraph OR clique
- KC2: tool OR application OR software OR framework OR suite OR package OR toolkit OR environment OR editor

The first part of the criterion (KC1) was defined to restrict the choice to papers that dealt with graphs, while the second part (KC2) served to narrow down the initial set of papers to those mentioning some kind of a tool or program in general.

When evaluating KC on each L0 paper, we did a case-insensitive search for whole words only, whether in their singular form (as written in KC1 and KC2) or their plural form (except for the case of "software"). This search also included hyphenated forms that featured one of the keywords from KC, e.g., "sub-graph" was considered to match the "graph" keyword.

As each proceedings file is a PDF document, we implemented a search in the Java programming language using the Apache PDFBox[4] library for PDF manipulation in Java. However, when extracting content from some papers, i.e., page ranges of a proceedings file, we could not retrieve text in English that could be easily searched. This was most probably caused by the fact that

[2] http ://www.educationaldatamining.org/

[3] http://www.educationaldatamining.org/proceedings

[4] https://pdfbox.apache.org/

authors used different tools to produce camera ready versions in PDF, which were later integrated into a single PDF file.

In these instances, usually one of the two main problems occurred: no valid text could be extracted or valid text was extracted but without spacing. In the case of invalid text, we had to perform optical character recognition (OCR) on the problematic page ranges. We used the OCR feature of PDF-XChange Viewer,[5] which was sufficient as confirmed by our manual inspection of the problematic page ranges (six problematic papers in total). In the case of missing spacing, we had to fine-tune the extraction process using the capabilities of the PDFBox library.

This PDF library proved adequate for our task because we had to search only through PDF files and could customize the text extraction process to solve the spacing problem. However, in the case of a more varied data source, a more advanced toolkit for content indexing and analysis would be needed.

2.4 Screening of Papers

EDM researchers used many of our keywords with several different meanings, e.g., a graph could denote a structure consisting of nodes and edges, which was the meaning that we looked for, or some form of a plot. In order to determine the final set of papers we performed a two-phase selection on L1 papers:

1. We examined the portions of L1 papers that contained some KC1 keyword and eliminated papers that did not significantly deal with graphs (as structures) – this led to a set of Level 2 (L2) papers.
2. We read each L2 paper and eliminated those that did not mention some use of graphs tools – this led to the final set of Level 3 (L3) papers.

In the first phase of selection, we examined the sentences that contain KC1 keywords. If this proved insufficient to determine the nature or scope of use of the mentioned graphs, we read the whole paragraph, and sometimes even the paragraph before and the paragraph after. In these cases, we also checked the referenced figures, tables, or titles of the cited papers. If there were still any doubts, we consulted the paper's title and abstract, as well as glanced over the figures looking for graph examples. If the authors did not use graphs in their presented study or just made a short comment about graphs giving an analogy or mentioning graphs in the context of related or future work, we did not select the paper for the next phase.

In the second phase of selection, we kept only those papers that mention explicit use of a graph tool by the authors. In the cases when the actual use of a mentioned graph tool was not clear, the paper was selected if some of its figures contain a screenshot featuring the tool or a graph visualized using that tool.

The term tool was considered rather broadly in the present study. We did not restrict the search only to well-rounded software applications, but also included libraries for various computer languages, and even computer languages or file formats that were used by researchers to manipulate graphs. By making this decision, we aimed to provide a greater breadth of information to researchers interested in applying graphs within their studies.

[5] http://www.tracker-software.com/product/pdf-xchange-viewer

2.5 Classification Scheme

The mode of tool usage was categorized in the following manner:

1. CREATION (C) – the tool was developed by the paper authors and introduced in the paper;
2. MODIFICATION (M) – the tool being modified, either through source code or by adding extensions/plugins; and.
3. UTILIZATION (U) – the tool being utilized without modification.

We also checked the distribution of the collected studies by the continent and the country corresponding to the authors' affiliation. In cases when there were authors from different countries, we indicated the country of the majority of authors, or, if there was no majority then the country corresponding to the affiliation of the first author.

2.6 Data Extraction and Map Creation

Relevant data from L3 papers was extracted into a table that for each paper included the following information: author list, title, proceedings where it was published, page range within the proceedings, answers to the research question and classifications according to the scheme presented in the previous subsection.

3. RESULTS AND DISCUSSION

An overview of the paper selection process is given in Table 1. In each step, the number of relevant papers is significantly reduced. As expected, the required effort in paper analysis was inversely proportional to the number of selected papers. In the L1 step, the usage of the keyword criterion relatively quickly eliminated many papers. However, in subsequent steps, the selected papers had to be read, either partially (in the L2 step) or fully (in the L3 step). The set of L3 papers represents a selection of EDM studies that were used to identify the usage patterns concerning graph tools. The list of the selected papers is publicly available.[6]

Table 1. The number of selected papers at each step

Step	Number of papers
L0 – papers from EDM proceedings	494
L1 – papers containing keywords	146
L2 – papers mentioning graphs	82
L3 – papers mentioning graph tools	27

Most studies (15) are from North America: USA (14) and Canada (1). Europe is represented by 8 studies from 6 countries: Czech Republic (2), Spain (2), Germany (1), Ireland (1), Russia (1), and UK (1). The remaining two continents represented are Asia (Japan only) and Australia, each providing 2 studies. This somewhat resembles the EDM community present at the EDM conferences and differs little from the structure of the EDM community as reported in 2009 [2].

3.1 Overview of Graph Tools

In Table 2, we list 28 graph tools mentioned in the 27 selected papers.

[6] http://www.acs.uns.ac.rs/en/user/31

Table 2. Overview of graph tools from the selected papers

No	Tool	Usage	Features	Purpose	Issues
1	<Untitled framework>	C[1]	argument database – retrieval and mining	retrieve, analyse, and reuse arguments	WIP
2	<Untitled tool>	C[25]	vis. and mine visit trails from WBESs	discover student trails in WBESs	/
3	AGG Engine	C[14], U[15]	augmented graph grammar engine with recursive graph matching	analyse student-produced argument diagrams	inefficiency in some cases
4	CASSI	C[19]	collect bullying data via web-form and use them to form a social graph	support classroom management	/
5	CLOVER framework	U[25]	generate graph vis.	(used in vis. in No. 2)	/
6	Cmate	U[16]	provide a list of concepts and linking words to build a concept map	tabletop concept mapping	/
7	D3.js	U[17]	program interactive graph vis.	facilitate graph interpretation in EDA	/
8	DOT	U[28]	describe graphs	(used in export in No. 14)	/
9	EDM Vis	C[9], M[10]	interactively vis. ITS log data	understand student problem solving in ITSs	WIP
10	eJUNG lib.	U[11]	layout graphs	(used in vis. in No. 14)	/
11	FuzzyMiner (ProM)	U[16]	generate fuzzy models (of student collaboration processes)	discover and analyse student strategies in tabletop collaboration	/
12	Gephi	U[7]	vis. graphs	identify similarities between LE course content (used together with No. 22)	/
13	graphML	U[30]	describe graphs (of student resolution proofs)	analyse student solutions of resolution proofs	/
14	InVis	C[11], M[12, 28]	interactively vis. and edit ITS log data	understand student interaction in ITSs	WIP
15	LeMo	C[18]	interactively vis. learning object networks	understand how students perform and succeed with resources in LMSs and LPs	/
16	Meerkat-ED toolbox	C[22]	vis., monitor, and evaluate participation of students in discussion forums	analyse student interaction and messages in discussion forums	/
17	meud	U[24]	create diagrams (concept lattices)	analyse choices of study programmes	/
18	Ora	U[6]	calculate SNA metrics	study SNA metrics to improve student performance classifiers	/
19	pajek	U[3],[32]	vis. networks and calculate network measures	use student social data to predict drop-out and failure; understand growth of communities on SNSs	/
20	R	U[8]	use scripts to vis. ELE interaction data	explore ELE interaction data and improve ELEs	WIP
21	R – igraph package	U[5],[32]	create, refine, vis., and analyse networks	compare student problem solving-approaches in ITSs; understand growth of communities on SNSs	/
22	RapidMiner	M[7]	create an operator for graph generation	identify similarities between LE course content (used together with No. 12)	/
23	RSP	C[4]	discover issues in the ITS process	support teachers through AT adaptation	/
24	SEMILAR toolkit	C[27]	semantic similarity methods for text	assess student natural language input in ITSs	/
25	SketchMiner	C[29]	generate graphs for student symbolic drawings; compare and cluster drawings	assess student symbolic drawings in ITSs	/
26	STG	C[4]	interactively vis. student interaction in ITSs	understand student problem solving in ITSs	/
27	TRADEM	C[23]	perform analysis on content corpus and generate a concept map in ITSs	support development of instructional content in ITSs	/
28	Visone	U[31]	vis. and analyse SNs, clique analysis	analyse user relationships in WBATs	/

The rows (graph tools) are ordered alphabetically by the tool name (the "Tool" column), which represents the answer to RQ1. In general, we discovered a diverse list of infrequently used graph tools. The usage of the graph tools, which represents the answer to RQ2, is covered by the columns "Usage" and "Features". In "Usage", we listed the mode of usage (see Section 2.5) and the references to the papers mentioning the graph tool. In "Features", we listed tool functionalities and capabilities that were created or employed by the researchers. The most often used feature was to visualize (vis.) graphs. The purpose of the selected studies, which represents the answer to RQ3, is given in the "Purpose" column. Researchers often analysed data from various interrelated systems: intelligent tutoring systems (ITSs) and adaptive tutorials (ATs), learning environments (LEs) including exploratory learning environments (ELEs), learning management systems (LMSs), learning portals (LPs), social network services (SNSs), web-based authoring tools (WBATs), and web-based educational systems (WBESs). Some frequent tasks were analysis of social networks (SNs) and exploratory data analysis (EDA).

The issues that the researchers faced when using the tools, which represents the answer to RQ4, are listed in the "Issues" column. In the majority of the selected papers, the researchers did not discuss problems related to tool usage. The main exceptions are studies in which researcher presented their own tools and discussed missing or incomplete features that should be fully implemented in future – this was labelled as work in progress (WIP).

4. POTENTIAL LIMITATIONS

The findings might not be representative of the whole EDM community but only of the practitioners who presented their work at one of the EDM conferences. An important issue in the analysis was the lack of information about the used tools. There were various instances when researchers obviously used a graph tool, or at least it could be expected that they relied on such tools, but failed to report the information.

Moreover, we used a somewhat "relaxed" definition of a graph tool. This allowed for the inclusion of both general-purpose tools for graph manipulation and domain-specific tools that were developed for educational domain but also utilize a graph-based structure. The primary motive behind this choice was to provide a list of graph tools potentially usable in a wider range of studies, as well as a list of tools that illustrates how graphs were implemented or used in a more specific problem. The former tool category generally includes tools associated with the "U" usage (tools utilized without modification), while the latter tool category mostly covers tools associated with the "C" usage (new tools introduced by their authors).

On the other hand, we excluded graph-based tools that could be labelled as data mining tools or causal modelling tools. For instance, some popular predictive and/or explanatory models (decision trees, random forests, and Bayesian networks) are graph-based, while causal modelling usually assumes creation or discovery of causal graphs. As these tools are more often featured in EDM studies, we assumed that EDM researchers are more familiar with their usage, so the focus of the present study is on other less frequently used graph tools.

5. CONCLUSION

We hope that the collected information about the usage of graph tools within the EDM community may prove valuable for researchers considering the use of graphs to solve educational problems. For future work, we plan to include other publication series, even those that are not solely devoted to the EDM research. The results of such an attempt could demonstrate whether EDM practitioners from other regions of the world are more represented in the graph-based research than indicated by the results of the present study.

6. ACKNOWLEDGMENTS

The research presented in this paper was supported by the Ministry of Education, Science, and Technological Development of the Republic of Serbia under Grant III-44010: "Intelligent Systems for Software Product Development and Business Support based on Models".

7. REFERENCES

[1] Abbas, S. and Sawamura, H. 2008. Argument mining using highly structured argument repertoire. In *Proceedings of the First International Conference on EDM* (Montreal, Canada, June 20 - 21, 2008). EDM'08. 202-209.

[2] Baker, R.S.J.d. and Yacef, K. 2009. The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining,* 1 (1), 3-17.

[3] Bayer, J., Bydzovska, H., Geryk, J., Obsivac, T. and Popelinsky, L. 2012. Predicting drop-out from social behaviour of students. In *Proceedings of the Fifth International Conference on EDM* (Chania, Greece, June 19 - 21, 2012). EDM'12. 103-109.

[4] Ben-Naim, D., Bain, M. and Marcus, N. 2009. A user-driven and data-driven approach for supporting teachers in reflection and adaptation of adaptive tutorials. In *Proceedings of the Second International Conference on EDM* (Cordoba, Spain, July 1 - 3, 2009). EDM'09. 21-30.

[5] Eagle, M. and Barnes, T. 2014. Exploring differences in problem solving with data-driven approach maps. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 76-83.

[6] Garcia-Saiz, D., Palazuelos, C. and Zorrilla, M. 2014. The predictive power of SNA metrics in education. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 419-420.

[7] Goslin, K. and Hofmann, M. 2013. Identifying and visualizing the similarities between course content at a learning object, module and program level. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 320-321.

[8] Gutierrez-Santos, S., Mavrikis, M., Poulovassilis, A. and Zhu, Z. 2014. Indicator visualisation for adaptive exploratory learning environments. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 377-378.

[9] Johnson, M.W. and Barnes, T. 2010. EDM visualization tool: Watching students learn. In *Proceedings of the Third International Conference on EDM* (Pittsburgh, PA, USA, June 11 - 13, 2010). EDM'10. 297-298.

[10] Johnson, M.W., Eagle, M.J., Joseph, L. and Barnes, T. 2011. The EDM Vis tool. In *Proceedings of the Fourth*

International Conference on EDM (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 349-350.

[11] Johnson, M.W, Eagle, M. and Barnes, T. 2013. InVis: An interactive visualization tool for exploring interaction networks. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 82-89.

[12] Johnson, M.W., Eagle, M., Stamper, J. and Barnes, T. 2013. An algorithm for reducing the complexity of interaction networks. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 248-251.

[13] Kitchenham, B. and Charters, S. 2007. *Guidelines for Performing Systematic Literature Reviews in Software Engineering.* Technical Report EBSE-2007-01, School of Computer Science and Mathematics, Keele University.

[14] Lynch, C.F. 2014. AGG: Augmented graph grammars for complex heterogeneous data. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 37-42.

[15] Lynch, C.F. and Ashley, K.D. 2014. Empirically valid rules for ill-defined domains. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 237-240.

[16] Martinez-Maldonado, R., Yacef, K. and Kay, J. 2013. Data mining in the classroom: discovering groups' strategies at a multi-tabletop environment. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 121-128.

[17] McTavish, T.S. 2014. Facilitating graph interpretation via interactive hierarchical edges. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 59-61.

[18] Merceron, A., Schwarzrock, S., Elkina, M., Pursian, A, Beuster, L., Fortenbacher, A., Kappe, L. and Wenzlaff B. 2013. Visual exploration of interactions and performance with LeMo. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 396-397.

[19] Olson, R., Daily, Z., Malayny, J. and Szkutak, R. 2013. Project CASSI: A social-graph based tool for classroom behavior analysis and optimization. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 398-399.

[20] Petersen, K., Feldt, R., Mujtaba, S. and Mattsson, M. 2008. Systematic mapping studies in software engineering. In *Proceedings of the Twelfth International Conference on Evaluation and Assessment in Software Engineering* (Bari, Italy, June 26 - 27, 2008).

[21] Portillo-Rodriguez, J., Vizcaino, A., Piattini, M. and Beecham, S. 2012. Tools used in Global Software Engineering: A systematic mapping review. *Information and Software Technology.* 54 (Mar. 2012), 663-685.

[22] Rabbany Khorasgani R., Takaffoli, M. and Zaiane, O.R. 2011. Analyzing participation of students in online courses using social network analysis techniques. In *Proceedings of the Fourth International Conference on EDM* (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 21-30.

[23] Ray, F, Brawner, K. and Robson, R. 2014. Using data mining to automate ADDIE. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 429-430.

[24] Romashkin, N., Ignatov, D. and Kolotova, E. 2011. How university entrants are choosing their department? Mining of university admission process with FCA taxonomies. In *Proceedings of the Fourth International Conference on EDM* (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 230-233.

[25] Romero, C., Gutierrez, S., Freire, M. and Ventura, S. 2008. Mining and visualizing visited trails in web-based educational systems. In *Proceedings of the First International Conference on EDM* (Montreal, Canada, June 20 - 21, 2008). EDM'08. 182-186.

[26] Romero, C. and Ventura, S. 2010. Educational data mining: A review of the state of the art. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews,* 40 (6), 601-618.

[27] Rus, V., Banjade, R., Lintean, M., Niraula, N. and Stefanescu, D. 2013. SEMILAR: A semantic similarity toolkit for assessing students' natural language inputs. In *Proceedings of the Sixth International Conference on EDM* (Memphis, TN, USA, July 6 - 9, 2013). EDM'13. 402-403.

[28] Sheshadri, V., Lynch, C. and Barnes, T. 2014. InVis: An EDM tool for graphical rendering and analysis of student interaction data. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 65-69.

[29] Smith, A., Wiebe, E., Mott, B. and Lester, J. 2014. SKETCHMINER: Mining learner-generated science drawings with topological abstraction. In *Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 288-291.

[30] Vaculik, K., Nezvalova, L. and Popelinsky, L. 2014. Graph mining and outlier detection meet logic proof tutoring. In *Extended Proceedings of the Seventh International Conference on EDM* (London, UK, July 4 - 7, 2014). EDM'14. 43-50.

[31] Xu, B. and Recker, M.M. 2010. Peer production of online learning resources: A social network analysis. In *Proceedings of the Third International Conference on EDM* (Pittsburgh, PA, USA, June 11 - 13, 2010). EDM'10. 315-316.

[32] Yamakawa, O., Tagawa, T., Inoue, H., Yastake, K. and Sumiya T. 2011. Combining study of complex network and text mining analysis to understand growth mechanism of communities on SNS. In *Proceedings of the Fourth International Conference on EDM* (Eindhoven, The Netherlands, July 6 - 8, 2011). EDM'11. 335-336.

A predictive model for identifying students with dropout profiles in online courses

Marcelo A. Santana
Institute of Computing
Federal University of Alagoas
marcelo.almeida@nti.ufal.br

Evandro B. Costa
Institute of Computing
Federal University of Alagoas
evandro@ic.ufal.br

Baldoino F. S. Neto
Institute of Computing
Federal University of Alagoas
baldoino@ic.ufal.br

Italo C. L. Silva
Institute of Computing
Federal University of Alagoas
italocarlo@nti.ufal.br

Joilson B. A. Rego
Institute of Computing
Federal University of Alagoas
jotarego@gmail.com

ABSTRACT
Online education often deals with the problem related to the high students' dropout rate during a course in many areas. There is huge amount of historical data about students in online courses. Hence, a relevant problem on this context is to examine those data, aiming at finding effective mechanisms to understand student profiles, identifying those students with characteristics to drop out at early stage in the course. In this paper, we address this problem by proposing predictive models to provide educational managers with the duty to identify students whom are in the dropout bound. Four classification algorithms with different classification methods were used during the evaluation, in order to find the model with the highest accuracy in prediction the profile of dropouts students. Data for model generation were obtained from two data sources available from University. The results showed the model generated by using SVM algorithm as the most accurate among those selected, with 92.03% of accuracy.

Keywords
Dropout, Distance Learning, Educational Data Mining, Learning Management Systems

1. INTRODUCTION
Every year, the registration marks in E-learning modality has increased considerably, in 2013, 15.733 courses were offered, in E-learning or semi-presence modality. Furthermore, the institutions are very optimistic, 82% of researched places, believe that the amount of registration marks will have a considerable expansion in 2015 [1], showing the E-learning evolution and its importance as a tool for citizen's formation. The Learning Management Systems (LMS) [15] can be considered one of factors that has had an important role for popularization of this learning modality [1].

Despite the rapid growth of online courses, there has also been rising concern over a number of problems. One issue in particular that is difficult to ignore is that these online courses also have high dropout rates. Specifically, in Brazil, in 2013, according with the latest Censo, published by the E-learning Brazilian Association (ABED), the dropout average was about 19,06% [1].

Beyond the hard task on identifying the students who can have possible risk of dropping out, the same dropout also brings a huge damage to current financial and social resources. Thus, the society also loses when they are poorly managed, once the student fills the vacancy but he gives up the course before the end.

Online education often deals with the problem related to the high students' dropout rate during a course in many areas. There is huge amount of historical data about students in online courses. Hence, a relevant problem on this context is to examine those data, aiming at finding effective mechanisms to understand student profiles, identifying those students with characteristics to drop out at early stage in the course.

In this paper, we address this problem by proposing predictive models to provide educational managers with the duty of identifying students who are in the dropout bound. This predictive model took in consideration academic elements related with their performance at the initial disciplines of the course. Data from System Information course at Federal University of Alagoas (UFAL) were used to build this model, which uses a very known LMS, called Moodle.

A tool to support the pre-processing phase was used in order to prepare data for application of Data Mining algorithms. The Pentaho Data Integration [2] tool covers the extraction areas, transformation and data load (ETL), making easier the archive generation in the compatible format with the data mining software adopted, called WEKA[5].

Therefore, for what was exposed above, it justifies the needing of an investment to develop efficient prediction methods, assessment and follow up of the students with dropout risk, allowing a future scheduling and adoption of proactive measures aiming the decrease of the stated condition.

The rest of the paper is organized as follows. Section 2 presents some related work. Section 3 Environment for Construction of predictive model. Afterwards, we present the experiment settings in Section 4, and in Section 5 we discuss the results of the experiment. Section 6 presents some concluding remarks and directions of future work.

2. RELATED WORK

Several studies have been conducted in order to find out the reasons of high dropout indices in online courses. Among them, Xenos [18] makes a review of the Open University students enrolled in a computing course. In this studies, five acceptable reasons, that might have caused the dropout, were identified: Professional (62,1%), Academic (46%), Family (17,8%), Health Issues (9,5%), Personal Issues (8,9%). According to Barroso and Falcão (2004) [6] the motivational conditions to the dropout are classified in three groups: i) Economic - Impossibility of remaining in the course because of socio-economics issues; ii) Vocational - The student is not identified with the chosen course. iii) Institutional - Failure on initial disciplines, previous shortcomings of earlier contents, inadequacy with the learning methods.

Manhães et al.[14] present a novel architecture that uses EDM techniques to predict and identify those who are at dropout risk. The paper shows initial experimental results using real world data about of three undergraduate engineering courses of one the largest Brazilian public university. According to the experiments, the classifier Naive Bayes presented the highest true positive rate for all datasets used in the experiments.

A model for predicting students' performance levels is proposed by Erkan Er [9]. Three machine learning algorithms were employed: instance-based learning Classifier, Decision Tree and Naive Bayes. The overall goal of the study is to propose a method for accurate prediction of at-risk students in an online course. Specifically, data logs of LMS, called METU-Online, were used to identify at-risk students and successful students at various stages during the course. The experiment were realized in two phases: testing and training. These phases were conducted at three steps which correspond to different stages in a semester. At each step, the number of attributes in the dataset had been increased and all attributes were included at final stage. The important characteristic of the dataset was that it only contained time-varying attributes rather than time-invariant attributes such as gender or age. According to the author, these data did not have significant impact on overall results.

Dekker [8] in your paper presents a data mining case study demonstrating the effectiveness of several classification techniques and the cost-sensitive learning approach on the dataset from the Electrical Engineering department of Eindhoven University of Technology. Was compared two decision tree algorithms, a Bayesian classifier, a logistic model, a rule-based learner and the Random Forest. Was also considered the OneR classifier as a baseline and as an indicator of the predictive power of particular attributes. The experimental results show that rather simple classifiers give a useful result with accuracies between 75 and 80% that is hard to beat with other more sophisticated models. We demonstrated that cost-sensitive learning does help to bias classification errors towards preferring false positives to false negatives. We believe that the authors could get better results by making some adjustments to the parameters of the algorithms.

Jaroslav [7], aims to research to develop a method to classify students at risk of dropout throughout the course. Using personal data of students enriched with data related to social behaviours, Jaroklav uses dimensionality reduction techniques and various algorithms in order to find which of the best results managing to get the accuracy rates of up to 93.51%, however the best rates are presented at the end of the course. Whereas the goal is to identify early on dropout, the study would be more relevant if the best results were obtained results at the beginning of the course.

In summary, several studies investigating the application of EDM techniques to predict and identify students who are at risk dropout. However, those works share similarities: (i) identify and compare algorithm performance in order to find the most relevant EDM techniques to solve the problem or (ii) identify the relevant attributes associated with the problem. Some works use past time-invariant student records (demographic and pre-university student data). In this study, contribution to those presented in this section, makes the junction between two different systems, gathering a larger number of attributes, variables and time invariant. Besides being concerned with the identification and comparison of algorithms, identify the attributes of great relevance and solve the problem the predict in more antecedence the likely to dropout students.

3. ENVIRONMENT FOR CONSTRUCTION OF PREDICTIVE MODEL

This subsection presents an environment for construction for a predictive model for supporting educators in the task of identifying prospective students with dropout profiles in online courses. The environment is depicted in Figure 1.

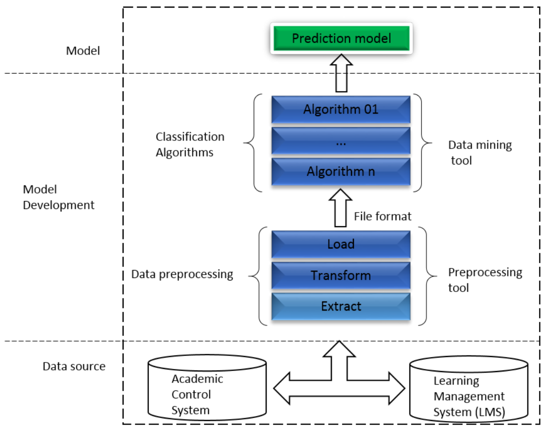


Figure 1: Environment for Construction of predictive model

The proposed environment in this work is composed by three layers: Data source, Model development and Model. The data sources are located in the first layer. Data about all students enrolled at the University are stored in two data sources: The first one contains students' personal data, for example: age, gender, income, marital status and grades from the academic control system used by the University. Information related with frequency of access, participation, use of the tools available, and grades of students related the activities proposed within the environment are kept in second data source.

In the second layer, the pre-processing [11] activity over the data is initiated. Sequential steps are executed in this layer in order to prepare them to data mining process. In the original data some information can not be properly represented in a expected format by data mining algorithm, data redundancy or even data with some kind of noise. These problems can produce misleading results or make the algorithm execution becomes computationally more expensive.

This layer is divided into the following stages: data extraction, data cleaning, data transformation, data selection and the choice of algorithm that best fits the model. Just below, will be displayed briefly each step of this layer.

Data extraction: The extraction phase establishes the connection with the data source and performs the extraction of the data.

Data cleaning: This routine tries to fill missing values, smooth out noise while identifying outliers, and correct data inconsistencies.

Data transformation: In this step, data are transformed and consolidated into appropriate forms for mining by performing summary or aggregation operations. Sometimes, data transformation and consolidation are performed before the data selection process, particularly in the case of data warehousing. Data reduction may also be performed to obtain a smaller representation of the original data without sacrificing its integrity.

Data selection: In this step, relevant data to the analysis task are retrieved from the database.

Choice of algorithm: An algorithm to respond with quality in terms of accuracy, which has students elusive profile, was considered the algorithm that best applies to the model.

Finally, the last layer is the presentation of the model. This layer is able to post-processing the result obtained in the lower layer and presenting it to the end-user of a most understandable way.

4. EXPERIMENT SETTINGS

The main objective of this present research is to build a predictive model for supporting educators in the hard task of identifying prospective students with dropout profiles in online courses, using Educational Data Mining (EDM) techniques [16]. This section is organized as follows: Section 4.1 describes the issue which drives our assessment. Section 4.2 shows which data were selected for to the data group utilized in the experiment and which algorithms were chosen for data mining execution. Section 4.3 indicates the employed tools during the execution of experiment. Finally, Section 4.4 shows every step in experiment execution, including data consolidation, data preprocessing and algorithms execution.

4.1 Planning

The research question that we would like to answer is:

RQ. Is our predictive model able to early identify the students with dropout risk?

In order to answer this question, EDM techniques with four different classification methods were used, aiming to get a predictive model which answers us with quality in precise ways which students have a dropout profile, taking in consideration only data about the initial disciplines of a specified course.

4.2 Subject Selection

4.2.1 Data Selection

The Federal University of Alagoas offers graduation courses, postgraduate courses and E-learning courses. In the on line courses, there are more than 1800 registered students[4].

An E-learning course is usually partitioned in semesters, where different disciplines are taught along these semesters. Each semester usually has five disciplines per semester, and each discipline has a duration between five to seven weeks. Anonymous data, from the Information Systems E-learning course, were selected from this environment, relative to first semester in 2013. Data of one discipline (Algorithm and Data Structure I), chosen based on its relevance, were analysed. Such discipline has about 162 students enrolled.

4.2.2 Machine Learning Algorithms Selection

In this work to predict student dropouts, four machine learning algorithms were used, using different classification methods. The methods used were: simple probabilistic classifier based on the application of Bayes' theorem, decision tree, support vector's machine and multilayer neural network.

These techniques have been successfully applied to solve various classification problems and function in two phases: (i) training and (ii) testing phase. During the training phase each technique is presented with a set of example data pairs (X, Y), where X represents the input and Y the respective output of each pair [13]. In this study, Y can receive one of the following values, “approved” or “reproved”, that corresponds the student situation in discipline.

4.3 Instrumentation

The Pentaho Data Integration [2] tool was chosen to realize all preprocessing steps on selected data. Pentaho is a open-source software, developed in Java, which covers extraction areas, transform and load of the data [2], making easier the creation of an model able to : (i) extract information from data sources, (ii) attributes selection, (iii) data discretization and (iv) file generation in a compatible format with the data mining software.

For execution of selected classification algorithms (see Section 4.2.2), the data mining tool Weka was selected. Such algorithms are implemented on Weka software as NaiveBayes (NB), J48 (AD), SMO (SVM), MultilayerPerceptron (RN)

[17] respectively. Weka is a software of open code which contains a machine learning algorithms group to data's mining task [5].

Some features were taken in consideration for Weka [10] adoption, such as: ease of acquisition, facility and availability to directly download from the developer page with no operation cost; Attendance of several algorithms versions set in data mining and availability of statistical resources to compare results among algorithms.

4.4 Operation

The evaluation of experiment was executed on HP Probook 2.6 GHz Core-I5 with 8Gb of memory, running Windows 8.1.

4.4.1 Data's Preprocessing

Real-world data tend to be dirty, incomplete, and inconsistent. Data preprocessing techniques can improve data quality, thereby helping to improve the accuracy and efficiency of the subsequent mining process [11].

Currently, the data is spread in two main data sources: LMS Moodle, utilized by the University as assistance on E-learning teaching, including data which show the access frequency, student's participation using the available tools, as well as the student's success level related to proposed activities. Meanwhile, student's personal files as age, sex, marital status, salary and disciplines grades are kept in the Academic Control System (ACS), which is a Software designed to keep the academic control of the whole University [4].

Aiming to reunite a major data group and work only with relevant data to the research question that we want to answer, we decided to perform consolidation of these two data source in a unique major data source, keeping their integrity and ensuring that only relevant information will be used during data mining algorithms execution.

Careful integration can help reduce and avoid redundancies and inconsistencies in the resulting data set. This can help improve the accuracy and speed of the data mining process [11].

To maintain the integrity and reliability between data, a mandatory attribute, with unique value and present between in both data sources, was chosen. Thus, the CPF attribute was chosen to make data unification between the two selected data sources, once it permits the unique identification among selected students.

In order to facilitate algorithms execution and comprehension of results,predicting the dropout in an early stage of the study. In order to achieve a high rate of accuracy and minimum of false negatives, i.e. students that have not been recognized to be in danger of dropout. Some attributes were transformed, as we can seen below:

- The corresponding attributes related with discipline grades were discretized in a five-group-value (A,B,C,D e E), depending on the discipline's achieved grades. The student with a grade higher or equal 9, was allocated for "A" group. Those ones who had their grades between 8,99 and 7 were allocated for "B" group. the "C" students are those that had a grade between 6,99 and 5, and those who had grades under 5,99 stayed at "D" group and finally those that doesn't have a grade associated were allocated in "E" group.

- Every student was labelled as approved or reproved based on the situation informed by the academics registers. The final score of each discipline is composed by two tests, if the student did not succeed in obtaining the minimum average, he will be leaded to the final reassessment and final test.

- In the "City" attribute, some inconsistencies were found, where different data about the same city were registered in database. For instance, the instances of **Ouro Branco** and **Ouro Branco/AL** are related to same city. This problem was totally solved, with application of techniques for grouping attributes.

- The attribute "age" had to be calculated. For this, the student's birth date, registered in database, was taken in consideration.

When all the attributes were used the accuracy was low. That is why we utilized feature selection methods to reduce the dimensionality of the student data extracted from dataset. We improved the pre-processing method the data.

In order to preserve reliability of attributes for classification after the reduction. We use InfoGainAttributeEval algorithm that builds a rank of the best attributes considering the extent of information gain based on the concept of entropy.

After this procedure, we reduced the set of attributes from 17 to 13 most relevant. The list of the refined set of attributes in relevance ordercan be found in Table 1.

Table 1: Selected Attributes

Attributes	Description
AB1	First Evaluation Grade
Blog	Post count and blog view
Forum	Post count and forum views
Access	Access Count in LMS
Assign	Sent files count e viewed
City	City
Message	Count of sent messages
Wiki	Post count and wiki view
Glossary	Post count and glossary view
Civil status	Civil status
Gender	Gender
Salary	Salary
Status	Status on discipline

Taking in consideration that the main objective is to predict student's final situation with the earlier advance as possible inside the given discipline, to this study we will only use data until the moment of the first test.

The Figure 2 presents all the executed stages, during the preprocessing phase, in order to generate a compatible file with the mining software.

4.4.2 Algorithms Execution

The k-fold method was applied to make a assessment the model generalization capacity, with k=10 (10-fold cross validation). The cross validation method, consists in splitting of the model in k subgroups mutually exclusive and with the same size, from these subgroups, one subgroup is selected for test and the remaining k-1's are utilized for training. The average error rate of each training subgroup can be used as an estimate of the classifier's error rate. When Weka implements the cross validation, it trains the classifier k times to calculate the average error rate and finally, leads the build classifier back utilizing the model as a training group. Thus, the average error rate provides a better solution in terms of classifier's error accuracy reliability [12].

In order to get the best results of the algorithms without losing generalization, some parameters of SVM algorithms were adjusted.

The first parameter was set the parameter "C". This parameter is for the soft margin cost function, which controls the influence of each individual support vector; this process involves trading error penalty for stability [3].

The default kernel used by Weka tool is the polynomial we changed to the Gaussian setting the parameters Gamma. Gamma is the free parameter of the Gaussian radial basis function [3].

After several adjustments to the values of the two parameters mentioned above, which showed the best results in term of accuracy and lower false positive rate, was C = 9.0 and Gamma = 0.06 parameter.

For comparison of results related to selected algorithms, we used Weka Experiment Environment (WEE). The WEE allows the selection of one or more algorithms available in the tool as well as analyse the results, in order to identify, if a classifier is, statistically, better than the other. In this experiment, the cross validation method, with the parameter "k=10" [5], is used in order to calculate the difference on the results in each one of the algorithms related to a chosen standard algorithm (baseline).

5. RESULTS AND DISCUSSIONS

In this section, the results of the experiment, described in Section 4, are analyzed.

The WEE tool calculated the average accuracy of each classifier. Table 2 shows the result of each algorithms execution. The accuracy represents the percentage of the test group instance which are correctly classified by the model built during training phases. If the built model has a high accuracy, the classifier is treated as efficient and can be put into production [11].

Comparing the results among the four algorithms, we can verify that the accuracy oscillates around 85.5 to 92.03%. Furthermore, a classifier which has a high error rate to false positives is not suitable to our solution. In this case, we have considered the algorithm which has the lower false positive rates.

Table 2: Accuracy and rates

Classifiers	NB	AD	SVM	RN
Accuracy	85.50	86.46	92.03	90.86
True Positives	0.76	0.77	0.88	0.85
False Negatives	0.24	0.23	0.12	0.15
True Negatives	0.89	0.91	0.94	0.93
False Positives	0.11	0.09	0.06	0.07

As we can see on table 2 the algorithm SVM presented a low false positive rate and better accuracy. Therefore, only the best algorithm was considered to our solution. The Naive Bayes classifier had the worst result in terms of accuracy and a high false positive rate. The other ones had an error average of 8%, and then, we end up with 8% of the students with dropout risk not so correctly classified.

5.1 Research Question

As can be seen in table 2, in our experiment, the SVM algorithm obtained 92% of accuracy. According to Han J. *et al.* [11] if the accuracy of the classifier is considered acceptable, the classifier can be used to classify future data tuples for which the class label is not known. Thus, the results are pointing to the viability of model able to early identify a possible student's dropout, based on their failures in the initial disciplines.

5.2 Statistical Significance Comparision

We often need compare different learning schemes on the same problem to see which is the better one to use. This is a job for a statistical device known as the t-test, or Student's t-test. A more sensitive version of the t-test known as a paired t-test it was used. [17]. Using this value and desired significance level (5%), consequently one can say that these classifiers with a certain degree of confidence (100 - significance level) are significantly different or not. By using the t-test paired in the four algorithms, performed via Weka analysis tool, observed that the SVM algorithm is significantly respectful of others.

5.3 Threats to validity

The experiment has taken in consideration data from the Information System course and the Data Structure Algorithm discipline. However, the aforementioned discipline was chosen, based on its importance in the context of Information System course.

6. CONCLUSION AND FUTURE WORK

Understand the reasons behind the dropout in E-learning education and identify in which aspects can be improved is a challenge to the E-learning. One factor, which has been pointed as influencer of students' dropout, is the academic element related with their performance at the initial disciplines of the course.

This research has addressed dropout problem by proposing predictive models to provide educational managers with the duty to identify students whom are in the dropout bound.

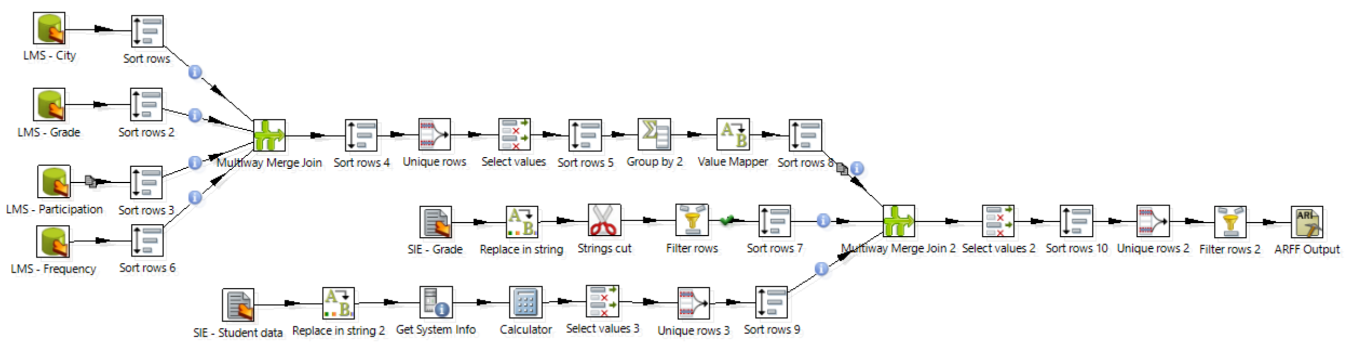


Figure 2: Steps Data Preprocessing

The adopted approach allowed us to perform predictions at an initial discipline phase. The preliminaries results has shown that prediction model to identify students with dropout profiles is feasible. These predictions can be very useful to educators, supporting them in developing special activities for these potential students, during the teaching-learning process.

As an immediate future work, some outstanding points still should be regarded to the study's improvement, as apply the same model in different institution databases with different teaching methods and courses, including new factors related to dropout as: professional, vocational and family data, execute some settings in algorithms' parameters in order to have the best achievements. Furthermore, a integrated software to LMS, to provide this feedback to educators, will be developed using this built model.

7. REFERENCES

[1] Abed - E-learning Brazilian Association. http://www.abed.org.br/. Accessed December 2014.

[2] Pentaho - Pentaho Data Integration. http://www.pentaho.com/. Accessed January 2015.

[3] SVM - support vector machines (svms). http://www.svms.org/parameters/. Accessed December 2014.

[4] UFAL - Federal University of Alagoas. http://www.ufal.edu.br/. Accessed January 2015.

[5] Weka - the University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/. Accessed January 2015.

[6] M. F. Barroso and E. B. Falcao. University dropout: the case of ufrj physics institute. *IX National Meeting of Research in Physics Teaching*, 2004.

[7] J. Bayer, H. Bydzovská, J. Géryk, T. Obsívac, and L. Popelínsky. Predicting drop-out from social behaviour of students. In A. H. M. Y. Kalina Yacef, Osmar Zaiane and J. Stamper, editors, *Proceedings of the 5th International Conference on Educational Data Mining - EDM 2012*, pages 103–109, Greece, 2012.

[8] G. Dekker, M. Pechenizkiy, and J. Vleeshouwers. Predicting students drop out: A case study. In T. Barnes, M. C. Desmarais, C. Romero, and S. Ventura, editors, *EDM*, pages 41–50, 2009.

[9] E. Er. Identifying at-risk students using machine learning techniques: A case study with is 100. In *International Journal of Machine Learning and Computing*, pages 476–481, Singapore, 2012. IACSIT Press.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[11] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[12] S. B. Kotsiantis, C. Pierrakeas, and P. E. Pintelas. Preventing student dropout in distance learning using machine learning techniques. In V. Palade, R. J. Howlett, and L. C. Jain, editors, *KES*, volume 2774 of *Lecture Notes in Computer Science*, pages 267–274. Springer, 2003.

[13] I. Lykourentzou, I. Giannoukos, V. Nikolopoulos, G. Mpardis, and V. Loumos. Dropout prediction in e-learning courses through the combination of machine learning techniques. *Comput. Educ.*, 53(3):950–965, Nov. 2009.

[14] L. M. B. Manhães, S. M. S. da Cruz, and G. Zimbrão. Wave: An architecture for predicting dropout in undergraduate courses using edm. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, SAC '14, pages 243–247, New York, NY, USA, 2014. ACM.

[15] M. Pretorius and J. van Biljon. Learning management systems: Ict skills, usability and learnability. *Interactive Technology and Smart Education*, 7(1):30–43, 2010.

[16] C. Romero and S. Ventura. Educational data mining: A review of the state of the art. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(6):601–618, Nov 2010.

[17] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2011.

[18] M. Xenos, C. Pierrakeas, and P. Pintelas. A survey on student dropout rates and dropout causes concerning the students in the course of informatics of the Hellenic Open University. *Computers Education*, 39(4):361 – 377, 2002.

Automatic Age Detection Using Text Readability Features

Avar Pentel
Tallinn University,Tallinn, Estonia
+372 51 907 739
pentel@tlu.ee

ABSTRACT

In this paper, we present the results of automatic age detection based on very short texts as about 100 words per author. Instead of widely used n-grams, only text readability features are used in current study. Training datasets presented two age groups - children and teens up to age 16 and adults 20 years and older. Logistic Regression, Support Vector Machines, C4.5, k-Nearest Neighbor, Naïve Bayes, and Adaboost algorithms were used to build models. All together ten different models were evaluated and compared. Model generated by Support Vector Machine with Adaboost yield to f-score 0.94, Logistic regression to 0.93. A prototype age detection application was built using the best model.

Keywords

Automatic age detection, readability features, logistic regression, support vector machines, Weka.

1. INTRODUCTION

One important class of information in user modeling is related to user age. Any adaptive technology can use age prediction data. In educational context automatic tutoring systems and recommendation systems, can benefit on age detection.

Automatic age detection has also utilities in crime prevention. With widespread of social media, people can register accounts with false age information about themselves. Younger people might pretend to be older in order to get access to sites that are otherwise restricted to them. In the same time older people might pretend to be younger in order to communicate with youngster. As we can imagine, this kind of false information might lead to serious threats, as for instance pedophilia or other criminal activities.

But besides serious crime prevention, automatic age detection can by used by educators as indirect plagiarism detector. While there are effective plagiarism detection systems, they do not work when parents are doing pupils homework or students are using somebody else's original work, which is not published anywhere. There are closed communities where students can buy homework's for any topic.

Full scale authorship profiling is not an option here, because large amount of author texts is needed. Some authors [1] argue, that at least 10000 words per author is needed, other that 5000 [2]. But if we think about business purpose of this kind of age detector, especially when the purpose is to avoid some criminal acts, then there is no time to collect large amount of text written by particular user.

When automatic age detection studies fallow authorship profiling conventions then it is related to second problem – the features, widely used in authorship profiling, are semantic features. Probability that some sequence of words, even a single word, occur in short text is too low and particular word characterizes better the context [3] than author. Some authors use character n-grams frequencies to profile users, but again, if we speak about texts that are only about 100 words long, these features can also be very context dependent.

Semantic features are related to third problem - they are costly. Using part of speech tagging systems to categorize words and/or large feature sets for pattern matching, takes time and space. If our goal is to perform age detection fast and online then it is better to have few features that can be extracted instantly on client side.

In order to avoid all three previously mentioned shortcomings, we propose other set of features. We call them readability features, because they are previously used to evaluate texts readability. Texts readability indexes are developed already before computerized text processing, so for example Gunning Fog index [4] takes into account complex (or difficult) words, those containing 3 or more syllables and average number of words per sentence. If sentence is too long and there are many difficult words, the text is considered not easy to read and more education is needed to understand this kind of text. Gunning Fog index is calculated with a formula (1) below:

$$GunningFogIndex = 0.4 \times \left[\left(\frac{words}{sentences} \right) + 100 \times \left(\frac{complexwords}{words} \right) \right] \quad (1)$$

We suppose that authors reading skills and writing skills are correlated and by analyzing author's text readability, we can infer his/her education level, which at least to the particular age is correlated with actual age of an author. As readability indexes work reliably on texts with about 100 words, these are good candidates for our task with short texts.

As a baseline we used n-gram features in pre testing. Comparing readability features with n-gram features, we found that with wider age gap between young and adult groups, readability features making better classifiers if using short texts [5]. Now we continue this work with larger dataset and with readability features only.

Using best fitting model, we created an online prototype age detector.

Section 2 of this paper surveys the literature on age prediction. In Section 3 we present our data, features, used machine learning algorithms, and validation. In Section 4 we present our classification results and prototype application. We conclude this paper in Section 5 by summarizing and discussing our study.

2. RELATED WORKS

In this section we review related works on age- and other author-specific profiling. There are no studies that dealing particularly with effect of text sizes in context of age detection. In previous section we mentioned that by literature for authorship profiling 5000 to 10000 words per author is needed [1,2]. Luyckx and

Daelemans [6] reported a dramatic decrease of the performance of the text categorization, when reducing the number of words per text fragment to 100. As authorship profiling and authors age prediction is not the same task, we focus on works that dealing particularly with user age.

The best-known age based classification results are reported by Jenny Tam and Craig H. Martell [7]. They used age groups 13-19, 20-29, 30-39, 40-49 and 50-59. All age groups were in different size. As features word and character n-grams were used. Additionally they used emoticons, number of capital letters and number of tokens per post as features. SVM model trained on youngest age group against all others yield to f-score 0,996. Moreover this result seems remarkable, while no age gap between two classes was used.

However we have to address to some limitations of their work that might explain high f-scores. Namely they used unbalanced data set (465 versus 1263 in training data set and 116 versus 316 in test set). Unfortunately their report gave only one f-score value, but no confusion matrices, ROC or Kappa statistics. We argue, that with unbalanced data sets, single f-score value is not sufficient to characterize the models accuracy. In such test set – 116 teenagers versus 316 adults - the f-score 0.85 (or 0.42 depending of what is considered positive result) will simply be achieved by model that always classifies all cases as adults. Also, it is not clear if reported f-score is weighted average of two classes' f-scores or presenting only one class f-score. Secondly it is not clear if given f-score was result of averaging cross validation results.

It is worth of mentioning, that Jane Lin [8], used the same dataset two years earlier in her postgraduate thesis supervised by the Craig Martell, and she achieved more modest results. Her best average f-score in teens versus adult's classification with SVM model was 0.786 as compared to Tam's and Martell reported 0.996. But besides averaged f-scores, Jane Lin also reported lowest and highest f-scores, and some of her highest f-scores were indeed 0.996 as reported in Tam and Martell paper.

Peersman et al [9] used large sample 10,000 per class and extracted up to 50,000 features based on word and character n-grams. Report states, that they used posts average of 12,2 tokens. Unfortunately it is not clear if they combined several short posts from the same author, or used single short message as a unique instance in feature extraction. They tested three datasets with different age groups –11-15 versus 16+, 11-15 versus 18+ and 11-15 versus 25+. Also experimentations carried out with number of features, and training set sizes. Best SVM model and with largest age gap, largest dataset and largest number of features yield to f-score 0.88.

Santosh, et al [10,11] used word n-grams as content-based features and POS n-grams as style based features. They tested three age groups 13-17, 23-27, and 33-47. Using SVM and kNN models, best classifiers achieved 66% accuracy.

Marquart [12] tested five age groups 18-24, 25-34, 35-49, 50-64, and 65-xx. Used dataset was unbalanced and not stratified. He also used some of the text readability features as we did in current study. Besides of readability features, he used word n-grams, HTML tags, and emoticons. Additionally he used different tools for feature extraction like psycholinguistic database, sentiment strength tool, linguistic inquiry word count tool, and spelling and grammatical error checker. Combining all these features, his model yield to modest accuracy of 48,3%.

Dong Nguyen and Carolyn P. Rose [13] used linear regression to predict author age. They used large dataset with 17947 authors with average text length of 11101 words. They used as features word unigrams and POS unigrams and bigrams. Text was tagged using the Stanford POS tagger. Additionally they used linguistic inquiry word count tool to extract features. Their best regression model had r^2 value 0.551 with mean absolute error 6.7.

As we can see, most of previous studies are using similar features, word and character n-grams. Additionally special techniques were used like POS tagging, Spell Checker, and Linguistic inquiry word count tool to categorize words. While text features extracted by this equipment are important, they are costly to implement in real life online systems. Similarly large feature sets up to 50,000 features, most of which are word n-grams, means megabytes of data. Ideally this kind of detector could work using client browser resources (JavaScript), and all feature extraction routines and models have to be as small as possible.

Summarizing previous work in the following table (1), we don't list all possible features. So for example features that are generated using POS tagging or features generated some word databases are all listed here as word n-grams. Last column gives f-score or the accuracy (with %) according to what characteristic was given in paper. Most of papers reported many different results, and we list in this summary table only the best result.

Table 1. Summary of previous work

Authors	Used feature types				training dataset size	avg. words per author	separation gap (year)	result f-score or accuracy (%)
	readability	word n-grams	char n-grams	emoticons				
Nguyen (2011)		x			17947*	11101	0	55.1%
Marquardt (2014)	x	x		x	7746	N/a	0	47.3%
Peersman (2011)		x	x		20000	12.2**	9	0.917
Lin (2007)		x		x	1728*	343	0	0.786
Tam & Martell (2009)		x	x	x	1728*	343	0	0.996***
Santosh (2014)		x			236600*	335	5	66%
This Study	**x**				**500**	**93**	**4**	**0.94**

*unbalanced datasets

**12.2 words was reported average message length, but it is not clear if only one message per user was used or user text was composed form many messages.

***not enough data about this result

3. METHODOLOGY

3.1 Sample & Data

We collected short written texts in average 93 words long from different social media sources like Facebook, Blog comments, and Internet forums. Additionally we used short essay answers from school online feedback systems and e-learning systems, and e-mails. No topic specific categorization was made. All authors were identified and their age fall between 9 and 46 years. Most authors in our dataset were unique, but we used multiple texts from the same author only in case, when the texts were written in different age. All texts in the collections were written in the same language (Estonian). We chose balanced and stratified datasets with 500 records and with different 4-year age gaps.

3.2 Features

In current study we used in our training dataset different readability features of a text. Readability features are quantitative data about texts, as for instance an average number of characters in the word, syllables in the word, words in the sentences, commas in the sentence and the relative frequency of the words with 1, 2,.., n syllable. All together 14 different features were extracted from each text plus classification variable (to which age class text author belongs).

In all features we used only numeric data and normalized the values using other quantitative characteristics of the text.

Used Feature set with explanations is presented in Table 2:

Table 2. Used features with calculation formulas and explanations

Feature	Explanation
Average number of Characters in Word	$= \frac{NumberOfCharactersInText}{NumberOfWordsInText}$ We excluded all white space characters when counting number of all characters in text
Average number of Words in Sentence	$= \frac{NumberOfWordsInText}{NumberOfSentencesInText}$
Complex Words to all Words ratio	$= \frac{NumberOfComplexWordsInText}{NumberOfWordsInText}$ Complex word is loan from Cunning Fog Index, where it means words with 3 or more syllables. As Cunning Fog index was designed for English, and Estonian language has as average more syllables per word, we raised the number of syllables according to this difference to five. Additionally we count the word complex if it has 13 or more characters.
Average number of Complex Words in Sentence	$= \frac{NumberOfComplexWordsInText}{NumberOfSentencesInText}$
Average number of Syllables per Word	$= \frac{NumberOfSyllablesInText}{NumberOfWordsInText}$
Average number of Commas per Sentence	$= \frac{NumberOfCommasInText}{NumberOfSentencesInText}$
One Syllable Words to all Words ratio	$= \frac{NumberOfWordsWith1syllableInText}{NumberOfWordsInText}$
Similarly as previous feature, we extracted 7 features for words containing 2, 3, 4 to 8 and more syllables.	$= \frac{NumberOfWordsWith\_N-SyllableInText}{NumberOfWordsInText}$ Novel syllable counting algorithm was designed for Estonian language, which is only few lines length and does not include any word matching techniques

3.3 Data Preprocessing

We stored all the digitalized texts in the local machine as separate files for each example. A local program was created to extract all previously listed 14 features from each text file. It opened all files one by one; extracted features form each file, and stored these values in a row of a comma-separated file. In the end of every row it stored data about the age group. A new and simpler algorithm was created for syllable counting. Other analogues algorithms for Estonian language are intended to exact division of the word to syllables, but in our case we are only interested on exact number of syllables. As it turns out, syllable counting is possible without knowing exactly where one syllable begins or ends.

In order to illustrate our new syllable counting algorithm, we give some examples about syllables and related rules in Estonian language. For instance the word *rebane* (fox) has 3 syllables: *re – ba – ne*. In cases like this we can apply one general rule – when single consonant is between vowels, then new syllable begins with that consonant.

When in the middle of word two or more consecutive consonants occur, then usually the next syllable begins with last of those consonants. For instance the word *kärbes* (fly) – is split as *kär-bes*, and *kärbsed* (flies) is split as *kärb-sed*. The problem is that this and previous rule does not apply to compound words. So for example, the word *demokraatia* (democracy) is split before two consecutive consonants as *de-mo-kraa-tia*.

Our syllable counting algorithm deals with this problem by ignoring all consecutive consonants. We set syllable counter on zero and start comparing two consecutive characters in the word, first and second character, then second and third and so on. General rule is, that we count a new syllable, when the tested pair of characters is vowel fallowed by consonant. The exception to this rule is the last character. When the last character is vowel, then one more syllable is counted.

Implemented syllable counting algorithm as well as other automatic feature extraction procedures can be seen in section 4.3 and in the source code of the prototype application.

3.4 Machine Learning Algorithms and Tools

For classification we tested six popular machine-learning algorithms:

- Logistic regression
- Support Vector Machine
- C4.5
- k-nearest neighbor classifier
- Naive Bayes
- AdaBoost.

Motivation of choosing those algorithms is based on literature [14,15]. The suitability of listed algorithms for given data types and for given binary classification task was also taken in to account. Last algorithm in the list – Adaboost – is actually not classification algorithm itself, but an ensemble algorithm, which is intended for use with other classifying algorithms, in order to make a weak classifier stronger. In our task we used Java implementations of listed algorithms that are available in freeware data analysis package Weka [16].

3.5 Validation

For evaluation we used 10 fold cross validation on all models. It means that we partitioned our data to 10 even sized and random parts, and then using one part for validation and other 9 as training dataset. We did so 10 times and then averaged validation results.

3.6 Calculation of final f-scores

Our classification results are given as weighted average f-scores. F-score is a harmonic mean between precision and recall. Here is given an example how it is calculated. Let suppose we have a dataset presenting 100 teenagers and 100 adults. And our model classifies the results as in fallowing Table 3:

Table 3. Example illustrating calculation of f-scores

Classified as =>	teenagers	adults
teenagers	88	12
adults	30	70

When classifying teenagers, we have 88 true positives (teenagers classified as teenagers) and 30 false positives (adults classified as teenagers). We also have 12 false negatives (teenagers classified as not teenagers) and 70 true negatives (adults classified as not teenagers). In following calculations we use abbreviations: TP = true positive; FP = false positive; TN = true negative; FN = false negative.

Positive predictive value or precision for teenagers' class is calculated by formula 2.

$$precision = \frac{TP}{TP+FP} = \frac{88}{88+30} = 0.746 \qquad (2)$$

Recall or sensitivity is the rate of correctly classified instances (true positives) to all actual instances in predicted class. Calculation of recall is given by formula 3.

$$recall = \frac{TP}{TP+FN} = \frac{88}{88+12} = 0.88 \qquad (3)$$

F-score is harmonic mean between precision and recall and it is calculated by formula 4.

$$f-score = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2TP}{2TP+FP+FN} \qquad (4)$$

Using data in our example the f-score for teenager class will be 0.807, but if we do the same calculations for adult class then the f-score will be 0.769.

Presenting our results, we use a single f-score value, which is an average of both classes' f-score values.

4. RESULTS

4.1 Classification

Classification effect was related to placement of age separation gaps in our training datasets. We generated 8 different datasets by placing 4-year separation gap in eight different places. We generated models for all datasets, and present the best models' f-scores on figure 1. As we can see, our classification was most effective, when the age separation gap was placed to 16-19 years.

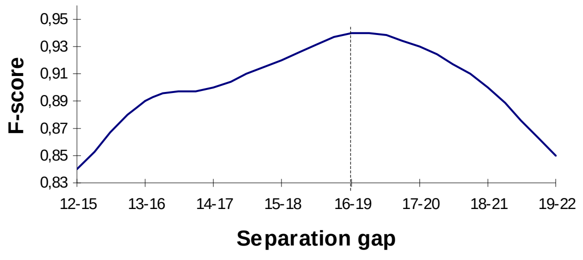


Figure 1. Effect of the position of separtion gap

With a best separation gap (16-19) between classes, Logistic regression model classified 93,12% of cases right, and Support Vector Machines generated model classified 91,74% of cases. Using Adaboost algorithm combined with classifier generated by Support Vector Machine yield to 94.03% correct classification and f-score 0.94. Classification models built by other algorithms performed less effectively as we can see in Table 4.

Results in fallowing table are divided in to two blocks. In the left side there are the results of the models generated by listed algorithms. In the right side there are the results of the models generated by Adaboost algorithm and the same algorithm listed in the row.

Table 4. Averaged F-scores of different models

	F-score	
		Using Adaboost
Logistic Regression	0.93	0.93
SVM (standardized)	0.92	0.94
KNN (k = 4)	0.86	0.86
Naïve Bayes	0.79	0.84
C4.5	0.75	0.84

As we can see in the table above, the best performers were classifiers generated by Logistic Regression algorithm and Support Vector Machine (with standardized data). In the right section of the table, where the effect of Adaboost algorithm is presented, we can see that Adaboost here cannot improve results with Logistic regression classifier, and kNN, but it improves results of SVM, Naïve Bayes and most significantly on C4.5. As Adaboost is intended to build strong classifiers out of weak classifiers, than the biggest effect on C4.5 is expectable. Two best performing classifiers remained still the same after using Adaboost, but now Support Vector Machine outperformed Logistic Regression by 0.91 percent points.

4.2 Features with highest impact

As there is relatively small set of readability features, we did not used any special feature selection techniques before generating models, and evaluating features on the basis of SVM model with standardized data. The strongest indicator of an age is the average number of words in sentence. Older people tend to write longer sentences. They also are using longer words. Average number of characters per word is in the second place in feature ranking. Best predictors of younger age group are frequent use of short words with one or two syllables.

In following Table (5), coefficients of standardized SVM model are presented.

Table 5. Features with highest impact in standardized SVM model

Coefficient	Feature
1.3639	Words in sentence
0.8399	Characters in word
0.258	Complex words in sentence
-0.2713	Ratio of words with 4 syllables
-0.3894	Commas per sentence
-0.7451	Ratio of words with 1 syllable
-0.762	Ratio of words with 2 syllables

4.3 Prototype Application

As the difference between performance of models generated by Adaboost with SVM and Logistic Regression is not significant, but as from the point of view of implementation, models without Adaboost are simpler, we decided to implement in our prototype application Logistic Regression model, which performed best without using Adaboost.[1] We implemented feature extraction routines and classification function in client-side JavaScript. Our prototype application uses written natural language text as an input, extracts features in exactly the same way we extracted features for our training dataset and predicts author's age class (Fig. 2.).

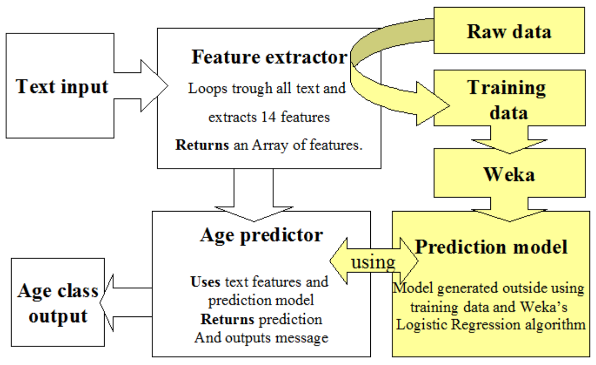

Text input
Feature extractor
Loops trough all text and extracts 14 features
Returns an Array of features.
Raw data
Training data
Weka
Age predictor
Uses text features and prediction model
Returns prediction
And outputs message
using
Prediction model
Model generated outside using training data and Weka's Logistic Regression algorithm
Age class output

Figure 2. Application design

Our feature extraction procedure (Figure 3.) consists 3 stages:

1. Text input is split to sentences, and to words, and all excess white space chars are removed. Some simple features, number of characters, number of words, number of sentences, are also calculated in this stage.
2. In second stage syllables in words are counted.
3. All calculated characteristics are normalized using other characteristics of the same text. For example number of characters in text divided to number of words in text.

[1] http://www.tlu.ee/~pentel/age_detector/

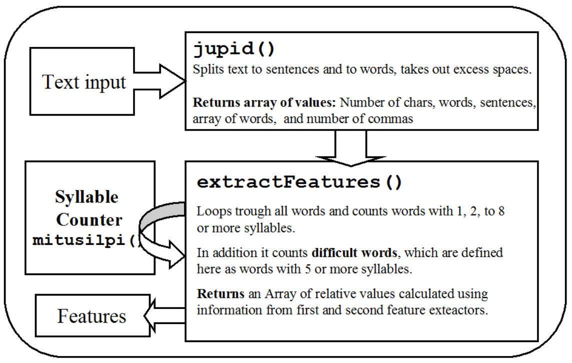

Text input
jupid()
Splits text to sentences and to words, takes out excess spaces.
Returns array of values: Number of chars, words, sentences, array of words, and number of commas
Syllable Counter mitusilpi()
extractFeatures()
Loops trough all words and counts words with 1, 2, to 8 or more syllables.
In addition it counts difficult words, which are defined here as words with 5 or more syllables.
Returns an Array of relative values calculated using information from first and second feature exteactors.
Features

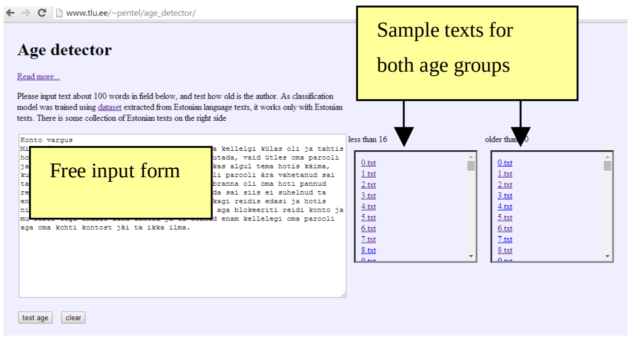
Figure 3. Feature Extractor

A new and simpler algorithm (5) was created for syllable counting. Other analogues algorithms for Estonian language are intended to exact division of the word to syllables, but in our case we are only interested on exact number of syllables. As it turns out, syllable counting is possible without knowing exactly where one syllable begins or ends. Unfortunately this is true only for Estonian (and maybe some other similar) language.

```
function number_of_syllables(w){              (5)

v="aeiouõäöü"; /* all vowels in Estonian lang. */

counter=0;

w=w.split('');/* creates char array of word */

wl=w.length; /* number of char’s in word */

  for(i=0; i < wl - 1; i++){

   if(v.indexOf(w[i])!=-1 && v.indexOf(w[i+1])==-1)

      counter++;

 /*

if char is vowel and next char is not, then count a
syllable (there are some exceptions to this rule, which
are easy to program).

*/

   }

  if( v.indexOf(w[wl-1]) != -1) counter++;

// if last char in the word is vowel, count new syllable

  return counter;

}
```

Implemented syllable counting algorithm as well as other automatic feature extraction procedures can be seen in the source code of the prototype application.[2]

Finally we created simple web interface, where everybody can test prediction by his/her free input or by copy-paste. As our classifier was trained on Estonian language, sample Estonian texts are provided on website for both age groups (Fig. 4.).


www.tlu.ee/~pentel/age_detector/
Sample texts for both age groups
Free input form
less than 16
older than 30
test age
clear

Age detector

Read more...

Please input text about 100 words in field below, and test how old is the author. As classification model was trained using dataset extracted from Estonian language texts, it works only with Estonian texts. There is some collection of Estonian texts on the right side

```
Konto vargus
Mi                          a kellelgi külas oli ja tahtis
ho                          utada, vaid ütles oma parooli
ja                          kas algul tema hotis käima,
ku                          li parooli ära vahetanud sai
ta                          branna oli oma hoti pannud
re                          da sai siis ei suhelnud ta
en                          kagi reidis edasi ja hotis
ni                          aga blokeeriti reidi konto ja
mu                          d enam kellelegi oma parooli
aga oma kohti kontost jäi ta ikka ilma.
```

0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt 7.txt 8.txt

0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt 7.txt 8.txt

Figure 4. Prototype application at http://www.tlu.ee/~pentel/age_detector/

5. DISCUSSION & CONCLUSIONS

Automatic user age detection is a task of growing importance in cyber-safety and criminal investigations. One of the user profiling problems here is related to amount of text needed to perform reliable prediction. Usually large training data sets are used to make such classification models, and also longer texts are needed to make assumptions about author's age. In this paper we tested novel set of features for authors age based classification of very short texts. Used features, formerly known as text readability features, that are used by different readability formulas, as Gunning Fog, and others, proved to be suitable for automatic age detection procedure. Comparing different classification algorithms we found that Logistic Regression and Support Vector Machines created best models with our data and features, giving both over 90% classification accuracy.

While this study has generated encouraging results, it has some limitations. As different readability indexes measure how many years of education is needed to understand the text, we can not assume that peoples reading, or in our case writing, skills will continuously improve during the whole life. For most people, the writing skill level developed in high school will not improve further and therefore it is impossible to discriminate between 25 and 30 years old using only those features as we did in current study. But these readability features might be still very useful in discriminating between younger age groups, as for instance 7-9, 10-11, 12-13. The other possible utility of similar approach is to use it for predicting education level of an adult author.

In order to increase the reliability of results, future studies should also include a larger sample. The value of our work is to present suitability of a simple feature set for age based classification of short texts. And we anticipate a more systematic and in-depth study in the near future.

[2] http://www.tlu.ee/~pentel/age_detector/source_code.txt

6. REFERENCES

[1] Burrows, J. 2007. All the way through: testing for authorship in different frequency strata. Literary and Linguistic Computing. 22, 1, pp. 27–47. Oxford University Press.

[2] Sanderson, C., and Guenter, S. 2007. Short text authorship attribution via sequence kernels, Markov chains and author unmasking: an investigation. EMNLP'06. Association for Computational Linguistics. pp. 482–491. Stroudsburg, PA, USA.

[3] Rao, D. et al. 2010. Classifying latent user attributes in twitter, SMUC '10 Proceedings of the 2nd international workshop on Search and mining user-generated contents. pp. 37-44.

[4] Gunning, R. 1952. The Technique of Clear Writing. New York: McGraw–Hill

[5] Pentel, A. 2014. A Comparison of Different Feature Sets for Age-Based Classification of Short Texts. Technical report. Tallinn University, Estonia. www.tlu.ee/~pentel/age_detector/Pentel_AgeDetection2b.pdf

[6] Luyckx, K. and Daelemans, W. 2011. The Effect of Author Set Size and Data Size in Authorship Attribution. Literary and Linguistic Computing, Vol-26, 1.

[7] Tam, J., Martell, C. H. 2009. Age Detection in Chat. International Conference on Semantic Computing.

[8] Lin, J. 2007. Automatic Author profiling of online chat logs. Postgraduate Thesis.

[9] Peersman, C. et al. 2011. Predicting Age and Gender in Online Social Networks. SMUC '11 Proceedings of the 3rd international workshop on Search and mining user-generated contents, pp 37-44, ACM New York, USA.

[10] Santohs, K. et al. 2013. Author Profiling: Predicting Age and Gender from Blogs. CEUR Workshop Proceedings, Vol-1179.

[11] Santosh, K. et al. 2014. Exploiting Wikipedia Categorization for Predicting Age and Gender of Blog Authors. UMAP Workshops 2014.

[12] Marquart, J. et al. 2014. Age and Gender Identification in Social Media. CEUR Workshop Proceedings, Vol-1180.

[13] Nguyen, D. et al. 2011. Age Prediction from Text using Linear Regression. LaTeCH '11 Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities. pp 115-123, Association for Computational Linguistics Stroudsburg, PA, USA.

[14] Wu, X. et al. 2008. Top 10 algorithms in data mining. Knowledge and Information Systems. vol 14, 1–37. Springer.

[15] Mihaescu, M. C. 2013. Applied Intelligent Data Analysis: Algorithms for Information Retrieval and Educational Data Mining, pp. 64-111. Zip publishing, Columbus, Ohio.

[16] Weka. Weka 3: Data Mining Software in Java. Machine Learning Group at the University of Waikato. http://www.cs.waikato.ac.nz/ml/weka/

Mining an Online Judge System to Support Introductory Computer Programming Teaching

Rodrigo Elias Francisco
Instituto Federal Goiano
Campus Morrinhos
Morrinhos – GO – Brazil
+55 (64) 34137900
rodrigo.francisco@ifgoiano.edu.br

Ana Paula Ambrosio
Instituto de Informatica
Universidade Federal de Goiás
Goiânia – GO – Brazil
+55 (62) 35211181
apaula@inf.ufg.br

ABSTRACT

Computer programming is an activity which requires a set of cognitive processes that naturally develop through practice, writing algorithmic solutions. Students learn a lot from their mistakes, but for this they need feedback on their workouts. Marking students' work outs is very time consuming, which often limits a teacher's capacity to offer close guidance individually. The PROBOCA project aims to build a tool, based on the BOCA online judge, suited for the purpose of learning computer programming by practice. In addition to a problem database organized by theme and difficulty, the system provides functionalities to support the teacher in the classroom. One of the main endeavors is to develop a procedure for estimating the degree of difficulty of a certain problem. This "nominal" parameter may then be compared to the difficulty level as perceived by each student. The result is a valuable indicator of those students that are experiencing challenges. This paper presents the preliminary specification of PROBOCA´s architecture and functional requirements along with its current state of development.

Keywords

Online Judge; Computer Programming Education.

1. INTRODUCTION

The learning process of Computer Programming (CP) usually involves practicing the resolution to many problems. Students write code to implement algorithms that meet exercise requirements and teachers should review those codes and present their feedback to the students. As this is a time consuming task, some teachers are installing and using online judge systems as automatic reviewing and scoring tools.

Online judge refers to software tools originally designed for programming competitions. They usually run on a server, and contestants access it online. Basically, its role is to present the contestant teams with a list of problems, to which they should respond by uploading program codes that satisfy the criteria of each problem. The tool then evaluates the answer code by using a set of predefined inputs and comparing the program results to predefined outputs. If the output of the answer code exactly matches the expected output for the corresponding input, the answer code is considered correct. Otherwise it is considered incorrect. No indication of where the program went wrong is given. Although helpful as a teaching assistant, these tools were not designed for use in a classroom and therefore lack some features that are important for academic management.

The BOCA software [1] is an online judge system used in programming marathons in Brazil. It is freely available for institutions to download and install. This particular system allows teachers to register problems and to track their students' work. However this system is neither easy to handle nor has an exercise database (DB), needed to facilitate the generation of problem lists.

This project proposes to extend the BOCA online judge to make it more suitable for use in introductory programming teaching. The resulting system, called PROBOCA, complements the online judge functionality with features that improve its ease-of-use, enhancing teacher productivity and course effectiveness.

One of the introduced features provides automatic classification of problem difficulty, based on submitted solutions and students' evaluation of degree of difficulty encountered in solving a given problem. This will aid teachers while composing the exercise lists, allowing them to better gauge the list complexity. It also lets students organize the order of problems to solve, tackling the easier problems first before turning to more complex ones.

Another additional feature is the production of student reports based on the submitted solutions and the students' behavior while accessing the tool. Student evaluation of program difficulty, number of submissions for the same exercise, time between submissions, order of submissions, among other information gathered by the tool, reveal information about student behavior and his ease in solving the proposed problems.

2. EXERCISES ON PROGRAMMING EDUCATION

Aiming at automatic correction of program code and student monitoring and evaluation during the programming process, several initiatives have been developed.

Within this line of research, the study of Chaves et al [2] aims to explore resources of online judges looking to integrate them into the Moodle system. The goal is to provide a Virtual Learning Environment (VLE) with automatic evaluation feature, allowing the teacher to monitor students´ problem solving. The authors defined an architecture containing a module for integration with online judges (MOJO). The Integration Module integrated the VLE structure to URI Online Judge [3] and SPOJ Brazil [4].

In [5], the authors have developed a prototype intended to be an educational online judge. They argue that online judge tools are suited to competition and have few educational features. They also criticize the way online judges provide feedback to students only indicating whether the answer is right/wrong or if there were errors at compilation/runtime. Their project, called JOnline, aims to add the following didactic features: Presenting tips in

Portuguese to fix compilation errors found in the submitted source code; presenting the test cases that generate erroneous results; organization of the problems by topic; difficulty level deduced from a poll conducted by the system and a resource for collaborative programming that allows two students to co-write one common code.

Automatic generation of exercise lists based on user defined criteria requires the classification of problems according to these criteria. In [6], the authors conclude that there is a strong relationship between the difficulty of a problem and the total number of lines of code and amount of flow control in the program (IF, WHILE, FOR ...). New students have difficulty in reading and interpreting problem statements. This problem has been related to the difficulty in dealing with abstraction [9].

3. BOCA

Designed for use in programming marathons, the BOCA online judge system has vocabulary and requirements contextualized by Competition and Teams. The main interest in this system is that it was designed to enable its users, the competitors, to interact with a set of problems. Figure 1 shows a use case diagram with the different functions provided by BOCA. Its software allows registration of problems for a given competition. That is done by submitting a PDF file stating the problem to be tackled and the files containing the input and output data sets for the corresponding test cases. The registered problems are associated with a single competition. If needed, to reuse the problems for another competition, the files must be inserted again. BOCA does not include a database to store the problems.

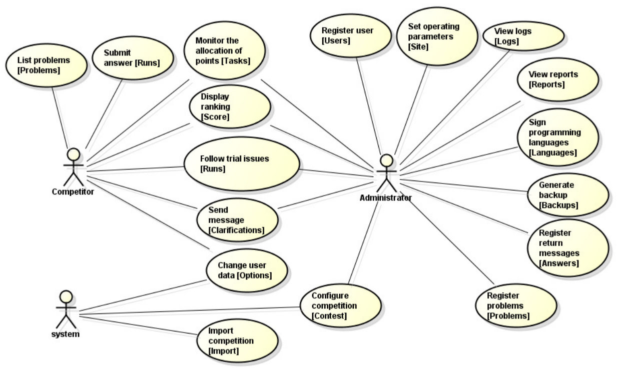

List problems [Problems]
Submit answer [Runs]
Monitor the allocation of points [Tasks]
Register user [Users]
Set operating parameters [Site]
View logs [Logs]
View reports [Reports]
Display ranking [Score]
Sign programming languages [Languages]
Follow trial issues [Runs]
Competitor
Administrator
Generate backup [Backups]
Send message [Clarifications]
Register return messages [Answers]
Change user data [Options]
system
Configure competition [Contest]
Register problems [Problems]
Import competition [Import]

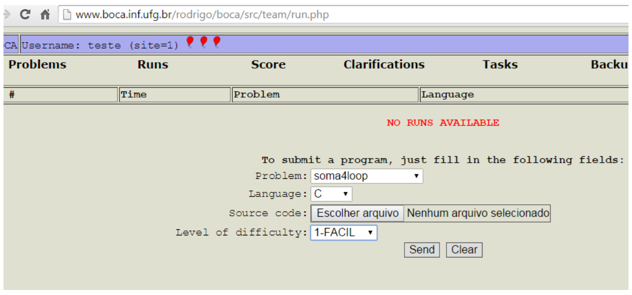
Figure 1. BOCA´s use case diagram

In the system, colored balloons represent the points gained by the competitors. This feature is part of the programming marathons context where correctly answering a problem yields a balloon to its team. Besides allowing teams to monitor their submissions and the related results, BOCA also provides a screen displaying the teams ranking including the team's overall score, and a list of the solved exercises using balloons to identify those successfully done. It also produces information on students' interaction with the problems containing the time spent and the number of resolution attempts.

BOCA has been used in CP introductory classes at our institute for some years, with good results. Using this system enhances the practice of problem solving by providing automatic feedback to students. We have observed that the number of problems solved by students using BOCA is significantly higher when compared to traditional exercise lists. The burden on the teacher is significantly reduced and the students get feedback. Teachers are then able to focus on helping the students that fail to solve the problems even after several attempts. This strategy allows students to tackle a larger number of exercises, thus increasing their potential to master cognitive skills required for programming. However, in spite of these advantages, some drawbacks were revealed.

Looking at the process of reviewing students' answers two issues were identified. First, the system assesses a submitted program by comparing its output, as generated by the student's code in response to a given input data set, to the registered expected output. For the two outcomes to be identical, exact formatting of the program's output is required. At the beginning students incurred in many errors just for using a slightly different format. This situation marks the problem's result as wrong whereas the program's logic maybe right, causing frustration among students as the system does not point out where the error is. Students, however, managed to adapt and began paying more attention to output formatting. Second, there is no cross-answers plagiarism identification, in other words no control over cheats. It was observed that some students simply copied their colleagues' program and submitted them as their own, thus considered to have solved the problem without any real comprehension of the solution.

Figure 2 shows the BOCA code submission screen. It was modified to include the "level of difficulty" selection where the students evaluate how hard it was to solve that problem they are submitting.



#	Time	Problem	Language

NO RUNS AVAILABLE

To submit a program, just fill in the following fields:

Problem: soma4loop
Language: C
Source code: Escolher arquivo Nenhum arquivo selecionado
Level of difficulty: 1-FACIL

Figure 2. BOCA's submission functionality answers

Regarding the class management functionality, it was noted that each installed instance of BOCA supports one active competition at a time, meaning a single class per instance. Thus, if a teacher wants to have several competitions running at the same time i.e. different groups doing the same or different sets of exercises, he must install multiple instances of BOCA. Each instance is independent. So even if two competitions are composed of the same problems (e.g. for different classes), these problems must be separately registered for each instance. As each competition is independent, the system does not group the results of the different competitions. This feature would be interesting in case of multiple exercise lists delivered to the same class. It should also be noted that the system is not trivial to install and manage, which ends up discouraging the teachers from adopting it. On top of that, the teacher needs to add each student to the system, which proves to be quite tedious, especially for large classes. To overcome this drawback, teachers have implemented programs that automatically generate registration ids and corresponding passwords based on student university registration number. As BOCA does not allow password modification, students often know other students password as they are generated following certain patterns. This facilitates copying other students' solutions.

This shows that, although BOCA has several advantages, there are problems that hinder the use of the system in the classroom. To solve this, beyond what has already been presented, we aim to tackle the following requirements: automatic generation of lists of problems based on subjects and level of difficulty, measurement of the student experience with problem solving by subject, and rank problems by levels of difficulty. We consider the last item is important for didactic purposes, not finding a technique or algorithm to use, this is the focus of this work.

4. PROBOCA

From an architectural viewpoint, the proposed system builds upon BOCA that is considered an internal component and offers its structure, PHP source code and PostgreSQL DB to be reused.

In order to avoid further complication of the development process, the "Competitions and Teams" context is kept, with some adaptations. The terms "*Competition*" and "*Team*" are used to loosely indicate *exercise list* and *student* respectively. BOCA´s user interface allows for such extrapolation which is already in practice by the teachers who use this system in their classroom.

Adapting BOCA to support CP teaching brought the need to introduce new functional requirements, as presented in the use case diagram in Figure 3.

PROBOCA required some changes to the BOCA DB structure. It was necessary to modify the internal tables in order to link the stored problems to given course's syllabus and certain difficulty levels as well as to include more detailed data about the students. Furthermore, the original competition-based structure will be altered to adapt it to the concept of multiple exercise lists.

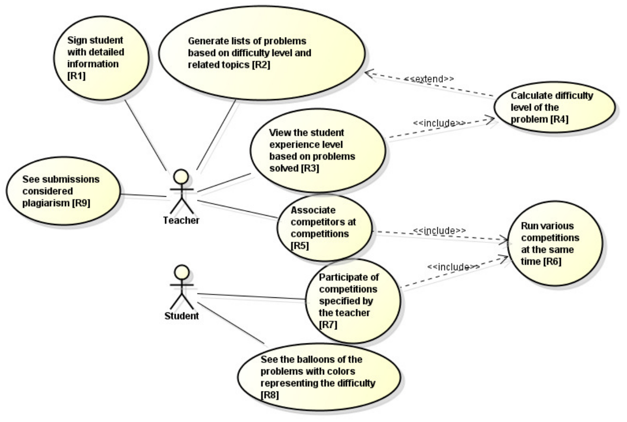
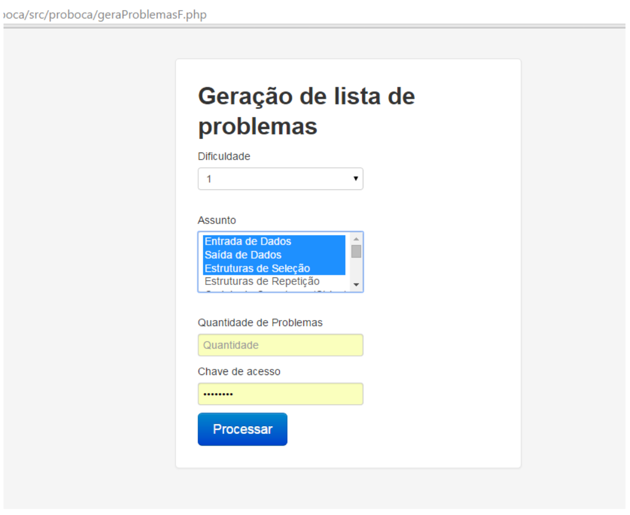

Sign student with detailed information [R1]
Generate lists of problems based on difficulty level and related topics [R2]
<<extend>>
Calculate difficulty level of the problem [R4]
<<include>>
View the student experience level based on problems solved [R3]
See submissions considered plagiarism [R9]
Teacher
Associate competitors at competitions [R5]
<<include>>
Run various competitions at the same time [R6]
<<include>>
Participate of competitions specified by the teacher [R7]
Student
See the balloons of the problems with colors representing the difficulty [R8]

Figure 3. Use Case Diagram with New Requirements

Following requirement R1, student registration is now done by the student himself using a token which is handed out by the teacher. Besides saving a password, other information is collected during registration that permits class wide statistical analysis.

R2 was intended to simplify the task of competition generation, which is performed by the teacher. To achieve this goal, a database of problems was implemented. Currently, to generate a competition, as shown in figure 4, the user indicates three parameters, namely a difficulty level (1-easy; 2-medium; 3-difficult); the desired component syllabus´ elements for the given problem set and finally the quantity of problems to compose the list. The system then analyzes which problems best fit the user-defined parameters and generates a competition list, from among the problems available in its DB. The difficulty level for each problem is estimated automatically by the system based on data obtained from solutions submitted by students and other parameters as described in section 6.


›oca/src/proboca/geraProblemasF.php

Geração de lista de problemas

Dificuldade: 1

Assunto: Entrada de Dados, Saída de Dados, Estruturas de Seleção, Estruturas de Repetição

Quantidade de Problemas: Quantidade

Chave de acesso: ••••••••

Processar

Figure 4. Generate problem lists\competitions

R3's purpose is to provide teachers with a report of every student´s experience level based on his interaction with the system. Using the collected data, it should be possible to present information about a student's experience regarding the syllabus elements addressed in each problem, thus allowing the teacher to detect where students show more difficulty. This requirement depends on R4, which measures the difficulty level of a given problem. A simple mechanism to measure the degree of difficulty of a given problem was developed and tested.

For R3 to be reliable, it is necessary that a plagiarism identification strategy (R9) be also implemented in the system.

R5 is aimed at associating a student to multiple competitions without the need to register each time. Thus, the data from different competitions can be associated to the student, allowing the integration of results obtained for the different lists presented to a given class. This requirement has a dependency upon R6, which aims to adapt BOCA to judge problems from several competitions. R7 is also related to R5, with the goal of allowing a student to participate in several competitions - analog to answering several problem lists.

R8 aims to display the problems' balloons, color coded to indicate the problem's difficulty level. Currently, balloon colors are registered by the system administrator along with the problems and have no relation to difficulty or content, unless the classification and corresponding color attribution is done by the teacher when uploading the problems.

5. IMPLEMENTATION

The functionalities of detailing the student information and providing automatic generation of exercise lists are already implemented and tested. Requirement R4, estimating the difficulty level of a given problem, is currently in progress.

Within this requirement, the first executed task was to create 74 exercises using BOCA´s problem-registration functionality. These exercises cover the syllabus of the "Introduction to programming" course (CS1) and range from debutant programming instructions (Hello World style) up to coding with matrixes. The inserted problems populated an initial database to be used for testing, but additional problems are expected to be easily included. An estimated level of difficulty for each problem was supplied by the teacher inserting the problem in the database.

To submit an answer, the student uploads one code file per problem solved. Along with the file upload, the student is asked to evaluate the problem´s difficulty, by indicating one of three choices: Easy, Medium or Difficult, based on his experience when solving the problem (Figure 4).

Success was obtained in computing several parameters that are needed to calculate problem difficulty. They include different measures, such as counting the number of repetition and selection structures used in the solution and the number of topics involved in the problem. Problem topics were defined using the introductory programming course syllabus. They include, input/output; attribution; selection; repetition; vectors; strings; matrices and functions. In this list, topics appear in the order they are taught and therefore in increasing level of complexity for novel students. Since programming is incremental, more complex topics usually base upon lesser complex topics. Several tests have been conducted to define and validate the mathematical function that will calculate the difficulty of the problem, but this is still an open issue.

Based on the calculated difficulty and the topics involved in the problem, colored balloons are associated to the problem. Different colors represent different topics, and within each topic, the level of difficulty is represented by the intensity of the color. For example, blue represents problems whose most complex topic is selection. Light blue balloons are associated to easy problems using selection. Medium blue balloons are associated to medium difficulty problems and dark blue balloons are associated to problems that demand more from students.

The task of providing teachers with a report on student achievements (R3) has not been tackled yet. Ideas in this sense include: (1) comparing student's perceived problem difficulty and mean problem difficulty. If students perceive problems as harder or easier than the mean this could indicate that they are at a lesser or higher level of programming competency; (2) comparing successive submission of the same problem. This may show if students adopt a trial-and-error approach; (3) mean time taken to solve problems; (4) mean number of submission per problem; (5) score when compared to the other students; among others.

6. Estimating Problem Difficulty

In a sense, "difficulty" expresses the lack of ability, or amount of skill/effort needed to accomplish something. Obviously, the perception of difficulty is an individual matter that is related to many aspects, including the time at which the question was posed. Nonetheless, this work investigates the possibility of characterizing a programming problem in a way such that calculated parameters may correlate to a determined "Difficulty-level" as expressed by students.

In their work [6], Alvarez and Scott present the program control flow and number of lines of code as variables that correlate to the difficulty of a problem. The experiments undertaken in the current study corroborate this information.

Also, other works [7, 8] show the need to deal with abstraction to solve problems. A highly abstract problem is one that implies a greater level of generalization. Thus, a problem that involves many different topics can become more abstract and hence more difficult. It is fairly safe to assume that in order to measure the difficulty of a problem, it is necessary to make a detailed analysis of the topics that are addressed within the problem. That proposition helps in formulating the hypothesis below.

6.1 Problem Difficulty Mechanism

On one hand, the survey of the student´s opinion of the submitted problem's difficulty provided the first estimate. For each of the 74 registered exercises, this information was stored, along with the respective answer in the DB. Although information was collected for all students, statistics were calculated only considering those that completed at least 95% of the exercises (70 out of 74). This excluded students that dropped out the course in order to prevent their partial answers from skewing the results. After filtering, the mean difficulty "Mean_Dif" was then calculated for each exercise based on the answers of the resulting 62 students. "Median_Dif" and "Mode_Dif" were also calculated.

In addition, C source code was written to correctly workout each of the registered problems. A PHP program was also developed in order to analyze the programs´ internal structures and to extract some measures from each program. The measures include counting:

- Lines of code, represented by variable N_LC;
- Repetition structures used in the solution, N_RP;
- Selection structures, N_SL;
- Edges in a graph that represents the algorithm, N_EDG;
- Edges in a graph that represent repetition structures in the algorithm, N_EDG_RP;
- Height of a tree, with each sub-block of internally nested code representing a node, MX_HT and
- Number of topics involved in the problem, N_TPC. This last count was obtained manually.

To verify if a combination of variables obtained better results, the following formulae were also tested against the Mean_Dif variable to verify their correlation. Table 2 shows the results. Mean_Dif correlated positively with all measured variables, being the best correlation associated to f4, $r = .76$, $p = .000$, that improves on the individual correlations with N_TPC (number of topics) and MX_HT (maximum tree height).

Table 1. Correlation between measured variables and student evaluation

		Median_ Dif	Mean _Dif	Mode_ Dif
N_RP (Repetitions)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.49 .000 74	.52 .000 74	.44 .000 74
N_SL (Selections)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.16 .171 74	.34 .003 74	.19 .108 74
N_LC (Lines of Code)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.44 .000 74	.62 .000 74	.45 .000 74
N_TPC (Topics)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.57 .000 74	**.69** .000 74	.59 .000 74
N_EDG (Edges)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.42 .000 74	.58 .000 74	.41 .000 74
MX_HT (Tree Height)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.52 .000 74	**.67** .000 74	.56 .000 74
N_EDG_RP (Rep. Edges)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.53 .000 74	.60 .000 74	.51 .000 74

$$f1 = \frac{N\_RP+N\_SL+N\_LC+N\_TPC+N\_EDG+MX\_HT+N\_EDG\_RP}{7}$$

$$f2 = N\_EDG \,.\, 0.3 + MX\_HT \,.\, 1.4 + N\_EDG\_RP \,.\, 1.4 + N\_LC \,.\, 0.2$$

$$f3 = \frac{N\_TPC+MX\_HT+N\_LC}{3} + \frac{N\_EDG\_RP+N\_EDG+N\_RP + N\_SL}{8}$$

$$f4 = \frac{N\_TPC + MX\_HT}{2}$$

Table 2 shows the results. Mean_Dif correlated positively with all measured variables, being the best correlation associated to f4, r = .76, p= .000, that improves on the individual correlations with N_TPC and MX_HT.

Table 2. Correlation of student perceived difficulty and developed formulae

		Mean_Dif
f1	*Pearson Correlation* *Sig. (2-tailed)* *N*	.66 .000 74
f2	*Pearson Correlation* *Sig. (2-tailed)* *N*	.68 .000 74
f3	*Pearson Correlation* *Sig. (2-tailed)* *N*	.67 .000 74
f 4	*Pearson Correlation* *Sig. (2-tailed)* *N*	**.76** .000 74

We also verified correlations between teacher evaluations of the problems' difficulty. For this we asked five teachers to read each problem and attribute a level of difficulty in the range 1-5. Since we had only five evaluations we chose to work with the median difficulty obtained (Median_Prof_Dif).

Correlating this variable to the measured variables we obtained the results presented in table 3. Best correlations were obtained with N_RP and N_EDG_RP, *r* = .62, *p* = .000, both related to the number of repetition structures found in the solution codes.

Table 3. Correlation between measured variables and teacher evaluation

		Median_Prof_Dif
N_RP (Repetitions)	*Pearson Correlation* *Sig. (2-tailed)* *N*	**.62** .000 74
N_SL (Selections)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.20 .093 74
N_LC (Lines of Code)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.56 .000 74
N_TPC (Topics)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.50 .000 74
N_EDG (Edges)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.53 .000 74
MX_HT (Tree Height)	*Pearson Correlation* *Sig. (2-tailed)* *N*	.50 .000 74
N_EDG_RP (Rep. Edges)	*Pearson Correlation* *Sig. (2-tailed)* *N*	**.62** .000 74

Table 4 correlates the teacher defined difficulty with the proposed formulae defined above. Positive correlation was found with all formulae, being the best correlation associated to f2, *r* = .63, *p*=.000. Furthermore, a positive correlation was found between the teacher defined difficulty and mean student perceived difficulty, *r* = .69, *p* = .000.

Table 4. Correlation of teacher defined difficulty and developed formulae

		Mean_Dif
f1	*Pearson Correlation* *Sig. (2-tailed)* *N*	.59 .000 74
f2	*Pearson Correlation* *Sig. (2-tailed)* *N*	**.63** .000 74
f3	*r** *Pearson Correlation* *Sig. (2-tailed)* *N*	.59 .000 74
f4	*Pearson Correlation* *Sig. (2-tailed)* *N*	.55 .000 74

Although student perceived difficulty and teacher defined difficulty are correlated, there are differences that can be verified by the correlations found with the measured variables and proposed formulae. This could be explained by the fact that students evaluate difficulty based on the knowledge they have when developing the solution. As they do not know what comes ahead, they cannot base their evaluation on the overall knowledge of programming. Teachers on the other hand, have this overall view of the domain and evaluate difficulty accordingly. It must be observed that the teachers that did the evaluation are new to teaching and have not yet acquired a more critical understanding of the difficulties students encounter when learning to program. After developing algorithmic reasoning, people tend to forget how they thought before, and find that many concepts are obvious when in fact, for beginners, they are not.

7. CONCLUSION

PROBOCA is a system under development whose goal is to adapt the BOCA software for use in teaching programming. While BOCA itself was developed for programming marathons, it is already in use, as a support tool, in programming introductory courses. As a learning aid, BOCA has several limitations, especially relative to class administration and student monitoring. In addition, as a system specifically developed for competitions, it lacks mechanisms that facilitate the creation of exercise lists, such as a question bank, and analysis of student performance.

PROBOCA supports the persistence of problems registered by teachers in the database and provides greater access to students' related information. Unlike other "Online Judge" systems that are available exclusively online and are managed by their creators, PROBOCA can be downloaded and installed by the teacher, giving the teacher control over the problems stored in the database. Since teachers are responsible for introducing the problems, this solution has the additional advantage that it is language free, i.e., it is not limited to teachers and students that speak the language in which the problem specifications were written as is the case of online systems administered by third parties.

In addition to implementing an environment that facilitates the use of BOCA in teaching programming, PROBOCA also aims to provide teachers and students with information that will help students in their learning process. One of the important features of PROBOCA is an automatic evaluation of problem difficulty. This gives students direction in the path to follow when choosing which exercises to solve first, allowing them to solve easier exercises before more complex ones, diminishing student frustration at not solving problems. It also allows teachers to better gauge the level of difficulty of exercise lists. As shown by the collected data, teacher and student evaluation regarding problem difficulty do not match, and this may lead to distortions when teaching programming. Future work could include developing a system that will automatically suggest problems to students based on their performance and calculated problem difficulty.

This work shows the approach being taken to calculate the difficulty of the problems. This approach differs from others by treating the algorithms submitted by students in the form of graphs and trees to identify properties that could be correlated with the difficulty of problems. For this, data mining using correlations was undertaken.

Part of the system has already been implemented, and has been successfully used in the classroom. The success of these tasks shows the feasibility of the project and encourages further work.

REFERENCES

[1] BOCA. *BOCA Online Contest Administrator*. Available in <http://www.ime. usp.br/~cassio/boca/> Access on March 20th, 2015

[2] Chaves, J. O. et al. *Uma Ferramenta de Auxílio ao Processo de Ensino Aprendizagem para Disciplinas de Programação de Computadores (2013).* TISE 2013

[3] URI. *URI Online Judge*. Available in <http:// www.urionlinejudge.com.br/> Access on May 20th, 2014.

[4] SPOJ. *SPOJ Brasil*. Available in <http://br.spoj.com/embed/info/> Access on May 20th, 2014.

[5] Santos, J. C. S., Ribeiro, A. R. L. *JOnline - proposta preliminar de um juiz online didático para o ensino de programação (2007).* SBIE 2011.

[6] Alvarez, A. and Scott, T. A. (2010). *Using student surveys in determining the difficulty of programming assignments.* Journal of Computing Sciences in Colleges, 26(2):157–163.

[7] Gomes, A. et al. (2008). *Aprendizagem de programação de computadores: dificuldades e ferramentas de suporte. Revista Portuguesa de Pedagogia*. 42(2).

[8] Mendonça, A. et al. *Difficulties in solving ill-defined problems: A case study with introductory computer programming students*. In Frontiers in Education Conference, 2009. FIE'09. 39th IEEE, pages 1–6. IEEE.

[9] Mendonça, A., de Oliveira, C., Guerrero, D., and Costa, E. (2009). *Difficulties in solving ill-defined problems: A case study with introductory computer programming students*. In Frontiers in Education Conference, 2009. FIE'09. 39th IEEE, pages 1–6. IEEE.

Workshop

on

International Workshop on Affect, Meta-Affect, Data and Learning

Preface

Emotions and affect play an important role in learning. There are indications that meta-affect (i.e., knowledge about self-affect) also plays a role. There have been various attempts to take them into account both during the design and during the deployment of AIED systems. The evidence for the consequential impact on learning is beginning to strengthen, but the field has been mostly focused on addressing the complexities of affective and emotional recognition and very little on how to intervene. This has largely slowed down progress in this area.

Research is needed to better understand how to respond to what we detect and how to relate that to the learner's cognitive and meta-cognitive skills. One goal might be to design systems capable of recognizing, acknowledging, and responding to learners' states with the aim of promoting those that are conducive to learning by means of tutorial tactics, feedback interventions, and interface adaptations that take advantage of ambient intelligence, among others. Therefore, we need to deepen our knowledge of how changes in learners' affective states and associated emotions relate to issues such as cognition and the learning context.

The papers submitted to the workshop address issues that bridge the existing gap between previous research with the ever-increasing understanding and data availability. In particular, these papers report progress on issues relevant to the broad and interdisciplinary AIED and EDM communities. AMADL 2015 workshop raises the opportunity to bring these two communities together in a lively discussion about the overlap in the two fields. To achieve this, we explicitly address and target both com- munities, as indicated by the workshop's organizers background and the programme committee set up. This workshop builds on the work done in affect related workshops in past AIED conferences, such as Modelling and Scaffolding Affective Experiences to Impact Learning in AIED 2007. The format of the workshop is based on presentations, demonstrations and discussions according to themes addressed by the papers accepted for the workshop.

Genaro Rebolledo-Mendez, Manolis Mavrikis, Olga C. Santos, Benedict du Boulay, Beate Grawemeyer and Rafael Rojano-Cáceres
Workshop Co-Chairs

The potential of Ambient Intelligence to deliver Interactive Context-Aware Affective Educational support through Recommendations

Olga C. Santos[1], Mar Saneiro[1], M. Cristina Rodriguez-Sanchez[2], Jesus G. Boticario[1], Raul Uria-Rivas[1], Sergio Salmeron-Majadas[1]

[1] aDeNu Research Group. Artificial Intelligence Dept. Computer Science School, UNED.
Calle Juan del Rosal, 16. Madrid 28040. Spain
http://adenu.ia.uned.es
{ocsantos,marsaneiro,jgb,raul.uria,sergio.salmeron}@dia.uned.es
[2] Electronics Department, Universidad Rey Juan Carlos.
Calle Tulipán s/n. Móstoles 28933 (Madrid), Spain.
cristina.rodriguez.sanchez@urjc.es

Abstract. There is a challenge and opportunity to research if the ambient intelligent support that can be deployed with a recommender system extended with an open hardware infrastructure that can sense and react within the learners' context is of value to supports learners' affectively. In this paper, we summarize the status of our research on eliciting an interactive recommendation for a stressful scenario (i.e., oral examination of a foreign language) that can be delivered through the Ambient Intelligence Context-aware Affective Recommender Platform (AICARP), which is the infrastructure we have designed and implemented with Arduino, an open-source electronic prototyping platform.

1 Eliciting Interactive Recommendations with TORMES

We have reported elsewhere [1] our progress on analyzing the potential of Ambient Intelligence to deliver more interactive educationally oriented recommendations that can deal with the affective state of the learner. In particular, following the TORMES methodology [2], we elicited an educational **scenario** focused on helping the learner when preparing for the oral examination in a second language learning course, which is widely considered as a stressful situation.

The **recommendation** identified in this scenario consists in suggesting the learner to breathe slowly (at a rate of 4 breaths/minute) and is aimed to calm her down when she is nervous. The *applicability conditions* that trigger the recommendation take into account physiological (i.e., heart rate, pulse, skin temperature, skin conductance) and behavioral (facial/body movements and speech speed) information that show evidence of restlessness. The recommendation *output* has been coded in a multisensory way by simultaneously modulating light, sound and vibration behavior at aforementioned breath rate, so the learner can perceive the recommended action through alternative sensory channels (i.e., sight, hearing and touch) without interrupting her activity.

2 Delivering Interactive Recommendations with AICARP

To deliver the aforementioned recommendation elicited with TORMES, the Ambient Intelligence Context-aware Affective Recommender Platform (AICARP) is being implemented with open source software and open hardware following a modular design controlled by an Arduino board (see [1] for details). In the current version, AICARP receives information from physiological **sensors** regarding changes in the learner affective state through corresponding physiological signals. The sensors are integrated into the e-Health platform [3] and a wireless electrocardiogram system [4]. Taking into account this information, AICARP is able to provide the elicited interactive recommendation to the learner by modulating the output of alternative sensorial **actuators** with the recommended breath rhythm. In particular, the following actuators have already been integrated into AICARP: i) white and red flashlights, ii) an array of blue LEDs, iii) a buzzer that vibrates and sounds, and iv) a speaker reproducing a pure tone at 440 Hz (i.e., “La” musical note).

To get some insight on the users’ perception on the recommendation delivery, we have deployed the educational scenario outlined in Section 1 in order to deliver the corresponding recommendation elicited with TORMES. So far, in this context we have carried out **2 pilot studies**, one with 6 university students with various interaction needs -including a blind participant-, and another with 4 participants within the 2014 Madrid Science Week. Since we wanted to test the potential of this approach in detecting not only the physiological information but also the behavioral information, we used the Wizard of Oz method [5]. In this way, the recommendation was triggered by the wizard (in our case, a psycho-educational expert) considering participants’ information on both physiological evidences detected with AICARP, as well as body/facial movements and speech speed that the wizard observed while the participants carried out the two tasks defined in the pilots (i.e., talking aloud in English about two specific given topics selected from those usually considered in oral exams).

3 Evaluation Outcomes and Open Issues identified

We evaluated AICARP in the 2 pilot studies with the analysis of the participants’ responses to the System Usability Scale [6] and to a post-study consisting in a semi structured interview led by the psycho-educational expert. This **evaluation** showed that the implemented infrastructure can actually sense the physiological state of the learner (which seems to be related to some affective state) and deliver ambient intelligent interactive feedback aimed to transform a negative affective (i.e., nervousness) state into a positive one (i.e., relaxation) (see [1] for details on the evaluation results). To the latter, actuators considered aim to provide a natural interaction support not interfering with the participant’s task, and consisted of visual, audio and/or tactile feedback.

As discussed in [1], the analysis of the evaluation outcomes has identified several **open issues** to be addressed in future research, as follows:

1. **How to deliver interactive recommendations:** this issue deals with selecting the preferred sensory channels from those available, the format to display the recommendation, the support to understand the purpose of the recommendation and the intrusion level.
2. **When recommendations are to be provided:** in terms of physiological and behavioral changes, while interfering as less as possible with the task. Here, and following TORMES methodology, data mining techniques can be explored to automatically identify the criteria that characterize the appropriate moment to deliver the recommendation [7].
3. **Learners' features of potential relevance in order to design other recommendations:** such as domain dependent attributes (i.e., the English level) and personality traits.
4. **Social aspects involved when collaboration takes place:** in the current scenario, collaboration can occur when learners are asked to perform the oral examination in pairs by dialoging a given situation. The training can be done using a videoconferencing system. In this context, other issues should be considered, such as the intensity of collaboration, the type of collaborative task, the individual acceptance of the technology used to support the collaboration, as well as specific personality traits.

Acknowledgements

This work was supported by the Spanish Ministry of Economy and Competitiveness (MINECO) under Grant TIN2011-29221-C03-01 (MAMIPEC project).

References

1. Santos, O.C., Saneiro, M., Rodriguez-Sanchez, M.C. and Boticario, J.G. (2015) Towards Interactive Context-Aware Affective Educational Recommendations in Computer Assisted Language Learning. New Review of Hypermedia and Multimedia, in press.
2. Santos, O.C., Boticario, J.G. (2015) Practical guidelines for designing and evaluating educationally oriented recommendations. In Computers and Education, vol. 81, 354–374.
3. Cooking Hacks. E-Health Platform. Available from: http://www.cooking-hacks.com.
4. Torrado-Carvajal, A., Rodriguez-Sanchez, M.C., Rodriguez-Moreno, A., Borromeo, S., Garro-Gomez, C., Hernandez-Tamames, J. A., and Luaces, M. (2012) Changing communications within hospital and home health care. In 2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), 6074-6077.
5. Dahlbäck, N., Jönsson, A., and Ahrenberg, L. (1993) Wizard of Oz studies: why and how. In Proceedings of Intelligent User Interfaces, 193–200.
6. Brooke, J. (1996) SUS: a 'quick and dirty' usability scale. In Jordan, P.W., Thomas, B., Weerdmeester, B.A. and McClelland, A.L. Usability Evaluation in Industry. London: Taylor and Francis.
7. Salmeron-Majadas, S., Arevalillo-Herráez, M., Santos, O.C., Saneiro, M., Cabestrero, R., Quirós, P., Arnau, D. and Boticario, J.G. (2015) Filtering of Spontaneous and Low Intensity Emotions in Educational Contexts. 17th Int. Conf. on Artificial Intelligence in Education (AIED 2015). Lecture Notes in Artificial Intelligence, vol. 9112, 429-438.

The impact of feedback on students' affective states

Beate Grawemeyer[1], Manolis Mavrikis[2], Wayne Holmes[2], Alice Hansen[2], Katharina Loibl[3], and Sergio Gutiérrez-Santos[1]

[1] London Knowledge Lab, Dep of Computer Science and Information Systems, Birkbeck, London, UK
beate@dcs.bbk.ac.uk, sergut@dcs.bbk.ac.uk
[2] London Knowledge Lab, UCL Institute of Education, University College London, London, UK
m.mavrikis@ioe.ac.uk, w.holmes@ioe.ac.uk, a.hansen@ioe.ac.uk
[3] Institute of Educational Research, Ruhr-Universität Bochum, Germany
katharina.loibl@rub.de

Abstract. Affective states play a significant role in students' learning behaviour. Positive affective states can enhance learning, while negative affective states can inhibit it. This paper describes a Wizard-of-Oz study that investigates the impact of different types of feedback on students' affective states. Our results indicate the importance of providing feedback matched carefully to the affective state of the students in order to help them transition into more positive states. For example when students were confused affect boosts and specific instructive feedback seem to be effective in helping students to be in flow again. We discuss this and other ways to adapt the feedback, together with implications for the development of our system and the field in general.

1 Introduction

This paper reports the results of a set of two Wizard-of-Oz studies which explore the effect of different feedback types on students' affective states.

It is well understood by now that affect interacts with and influences the learning process [9, 6, 2]. While positive affective states such as surprise, satisfaction or curiosity contribute towards constructive learning, negative ones including frustration or disillusionment at realising misconceptions can lead to challenges in learning. The learning process is indeed full of transitions between positive and negative affective states and regulating those is important. For example, a student may seem interested in exploring a particular learning goal, however s/he might have some misconceptions and need to reconsider her/his knowledge. This can evoke frustration and/or disappointment. However, this negative affective state may turn into deep engagement with the task again. D'Mello et al., for example, elaborate on how confusion is likely to promote learning under appropriate conditions [6].

It is important therefore, to deepen our understanding of the role of affective states for learning, and to be able to move students out of states that inhibit learning. Pekrun [13] discusses achievement emotions or affective states, which arise in a learning situation. Achievement emotions are states that are linked to learning, instruction, and achievement. We focus on a subset of affective states identified by Pekrun: flow/enjoyment, surprise, frustration, and boredom. We also add confusion, which has been identified elsewhere as an important affective state during learning [15] for tutor support and for learning in general [6].

As described in Woolf et al. [20] students can become overwhelmed (very confused or frustrated) during learning, which may increase cognitive load [19] for low-ability or novice students. However, appropriate feedback might help to overcome such problems. Carenini et al. [3] describe how effective support or feedback needs to answer three main questions: (i) when the support should be provided during learning; (ii) what the support should contain; and (iii) how it should be presented.

In this paper we focus on the question of *what* the support should contain with respect to affect i.e. the types of feedback that are able to induce a positive affective state.

In related work students' affective states have been used to tailor motivational feedback and learning material in order to enhance the learning experience. For example, Santos et al. [17] show that affect as well as motivation and self-efficacy impact the effectiveness of motivational feedback and recommendations. Additionally, Woolf et al. [20] developed an affective pedagogical agent which is able to mirror a student's affective state, or acknowledge a student's affective state if it is negative. Another example is Conati & MacLaren [5], who developed a pedagogical agent to provide support according to the affective state of the students and the user's personal goal. Also, Shen et al. [18] recommend learning material to the student based on their affective state. D'Mello et al. [7] developed a system that is able to respond to students via a conversation that takes into account the affective state of the student.

In contrast, in this paper, we investigate the impact of different types of feedback on students' affective state and how and whether they can help students regulate their affect and thus improve learning. In what follows we present two sets of Wizard-of-Oz studies where feedback was provided to students interacting with an exploratory learning environment designed to learn fractions. From these studies, the affective states of the students were carefully annotated in order to address our research questions.

2 The Wizard-of-Oz studies

2.1 Aims

One of our research aims is to develop intelligent support that enhances the learning experience by taking into account the student's affective state. We were specifically interested in identifying how different feedback types modify affective states.

In order to address this question we conducted two sets of ecologically valid Wizard-of-Oz studies (e.g. [11, 8]) which investigated the effect of affective states on different feedback types at different stages of the task.

2.2 Participants and Procedure

In total, 26 Year-5 (9 to 10-year old) students took part in the Wizard-of-Oz studies. Each session lasted on average 20 minutes. Each student participated in one Wizard-of-Oz session.

The sessions were run in an ordinary classroom with multiple computers, where additional children were working with the learning platform (not wizarded) in order to support ecological validity. This was important particularly as in early settings we identified that children would not speak that much to the platform if they felt that they were monitored [10]. Figure 1 shows the setup of the studies. Wizards followed a script with pre-canned messages to send messages to the students through the learning platform and deliberately limited their communication capacity in order to simulate the actual system. To achieve that wizards were only able to see students' screen. An assistant was able to hear students' reactions to reflections or talk-aloud prompts (as prompted by the 'system') and provide recommendations to the wizard with respect to the detected affective state. Any feedback provided was both shown on screen and read aloud by the system to students.

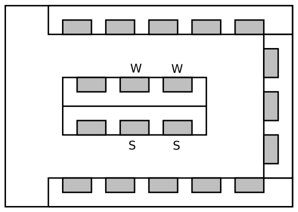


Fig. 1. The layout. The Wizard-of-Oz studies took place on the central isle while the rest of the students worked on a version of the system which only sequences tasks and provides minimal support (W=wizard, S=student).

2.3 Feedback types

Different types of feedback were presented to students at different stages of their learning task. The feedback provided was based on interaction via keyboard and mouse, as well as speech.

We explore different types of feedback that are known from the literature to support students in their learning and fit our context. The following different feedback types were provided:

- **AFFECT BOOSTS - affect boosts**. As described in [20] affect boosts can help to enhance student's motivation in solving a particular learning task. These included prompts that acknowledged for example that a task is difficult or that the student may be confused but they should keep trying.

- **INSTRUCTIVE FEEDBACK - instructive task-dependent feedback**. This feedback provided detailed instructions, what subtask or action to perform in order to solve the task.
- **OTHER PROBLEM SOLVING FEEDBACK - task-dependent feedback**. This support was centred on helping students to solve a particular problem that they are facing during their interaction by providing either questions to challenge their thinking or specific hints designed to help them identify the next step themselves.
- **TALK ALOUD PROMPTS - talking aloud**. With respect to learning in particular, the hypothesis that automatic speech recognition (ASR) can facilitate learning is based mostly on educational research that has shown benefits of verbalization for learning (e.g., [1]).
- **REFLECTIVE PROMPTS - reflecting on task performance and learning**. Self-explanation can be viewed as a tool to address students' own misunderstandings [4] and as a 'window' into students' thinking.
- **TALK MATHEMATICS PROMPTS - using particular domain specific mathematics vocabulary**. The aim of this prompt was to encourage students to use mathematical vocabulary in order continually revise their interpretations. In early studies [10] we found that students' reflections were often procedural and pragmatic (e.g. talking about the user interface) rather than mathematical.
- **TASK SEQUENCE PROMPTS - moving to the next task**. This feedback is centred on providing support regarding what action to perform next in order to change the task, such as clicking the 'Next' button.

Table 1 shows examples of the different feedback types.

3 Annotation of affective states and feedback reactions

From the Wizard-of-Oz studies we recorded the students' screen display and their voices. From this data, we annotated affective states (e.g. screen interaction and what the students said) before and after feedback was provided.

As described earlier, for the affective state detection we discriminated between five different affective types: enjoyment, surprise, confusion, frustration, and boredom. For the annotation of those affective states we used a similar strategy to that described in [15], where a dialogue between a teacher and a student was annotated retrospectively by categorising utterances in terms of different feedback types. Also, [2] describe how they coded different affective states based on observations of students interacting with a learning environment. Similarly, we annotated student's affective states for each type of feedback provided. In addition to the student's voice we also used the video of the screen capture to support the annotation process. Students' affective states were annotated as follows:

- **FLOW:** Engagement with the learning task. Statements like 'I am enjoying this task' or 'This is fun'. Sustained interaction with the system.

Feedback type	Example
AFFECT BOOSTS	You're working really hard! Keep going!
INSTRUCTIVE FEEDBACK	Use the comparison box to compare your fractions.
OTHER PROBLEM SOLVING FEEDBACK	If you add fractions, they need to have the same denominators first.
REFLECTIVE PROMPTS	What do you notice about the two fractions?
TALK ALOUD PROMPTS	Remember to talk aloud, what are you thinking?
TALK MATHEMATICS PROMPTS	Can you explain that again using the terms denominator, numerator?
TASK SEQUENCE PROMPTS	Well done. When you are ready click 'next' for the next task.

Table 1. Examples of feedback types

- **SURPRISE:** Gasping. Statements like 'Huh?' or 'Oh, no!'.
- **CONFUSION:** Failing to perform a particular task. Statements such as 'I'm confused!' or 'Why didn't it work?'. Uncertain interaction with the system.
- **FRUSTRATION:** Tendency to give up, repeatedly clicking or deleting of objects in the system or repeatedly failing to perform a particular task, sighing, statements such as, 'What's going on?!'.
- **BOREDOM:** Inactivity or statements such as 'Can we do something else?' or 'This is boring'.

4 Results

In total 396 messages were sent to 26 students. The video data in combination with the sound files were analysed independently by three researchers (one was independent of the project) who categorised the affective states of students before and after the feedback messages were provided.

The data is combined from two sets of Wizard-of-Oz studies. We use kappa statistics to measure the degree of the agreements of the annotations for reliability. Kappa was .46, p<.001. This is generally expected from retrospective annotation of naturalistic affect experiences [14]. We consolidated the annotations based on discussion between the annotators and the rest of the authors of the paper in order to agree upon the annotations that did not match originally. In the second set we had resources to introduce the Baker-Rodrigo Observation Method Protocol (BROMP) and the HART mobile app that facilitates the coding of students affective states in the classroom [12]. Kappa based on the retrospective annotation was still .56, p<.001. We first consolidated the data with the same approach as before and then compared against the field annotations. Kappa between the consolidated annotation and the HART data was .71, p<.05 (note that is may appear low but we did not expect the retrospective annotation to get surprise and frustration accurately). We used the HART data to improve the annotation by mapping feedback actions against the observation for 20 seconds prior to the delivery of the feedback to 20 seconds after the student had closed the corresponding feedback window. We marked the changes for an independent annotator to revisit the first set of annotations.

The student's affective states, that occurred before and after the different types of feedback was provided, can be seen in figure 2. Each block shows an affective state *before* feedback was provided. The colour within the bars indicates the type of affective states that occurred *after* the feedback was provided. The number within the bars indicate the number of times the affective state occurred.

In order to investigate whether there was an effect of the feedback on the learning experience, we looked at whether a student's affective state was enhanced, stayed the same or worsened. An affective state was enhanced for example, when it was changed from confusion to flow, or (given the findings about confusion [6]) from frustration to confusion, frustration to flow, boredom to flow etc. An affective state was worsened if it moved for example, from flow to frustration or confusion, or from confusion to frustration.

As the data is categorical [16], we apply chi-square tests to investigate statistical significant differences between the groups. We present them below and discuss in more detail in the next section.

Flow When students were in flow, there was no significant difference between the feedback types on whether the affective state stayed in the same flow state (X^2(6, N=169) = 4.31, p>.05) or worsened (X^2(6, N=169) = 4.89, p>.05). As flow is the most positive affective state, the affective state in this sub-sample cannot be enhanced.

Confusion When students were confused, there was a significant effect of the feedback type on whether students' affective state was enhanced into a flow state (X^2(6, N=181) = 13.65, p<.05). The most effective feedback types were affect boosts with 68% of the cases, followed by guidance feedback with 67%, and task sequence prompts with 63%. Reflective prompts resulted in a flow state in 48% of the cases, talk aloud prompts 38%, and problem solving support with 34%. Talk maths prompts were the least effective with only 25% of the cases.

There was also a significant effect of the feedback type and whether the affective state stayed the same (X^2(6, N=181) = 14.34, p<.05). Talk maths prompts were highest associated with a continuing confused state with 75% of the cases. This was followed by problem solving support with 66%, talk aloud prompts with 59%, reflective prompts with 52%, task sequence prompts with 37%, affect boosts with 32%, and the least feedback type that was associated with a continuing confused state were guidance feedback with 29% of the cases.

There was no significant association between the feedback type and whether the affective state worsened (X^2(6, N=181) = 4.65, p>.05).

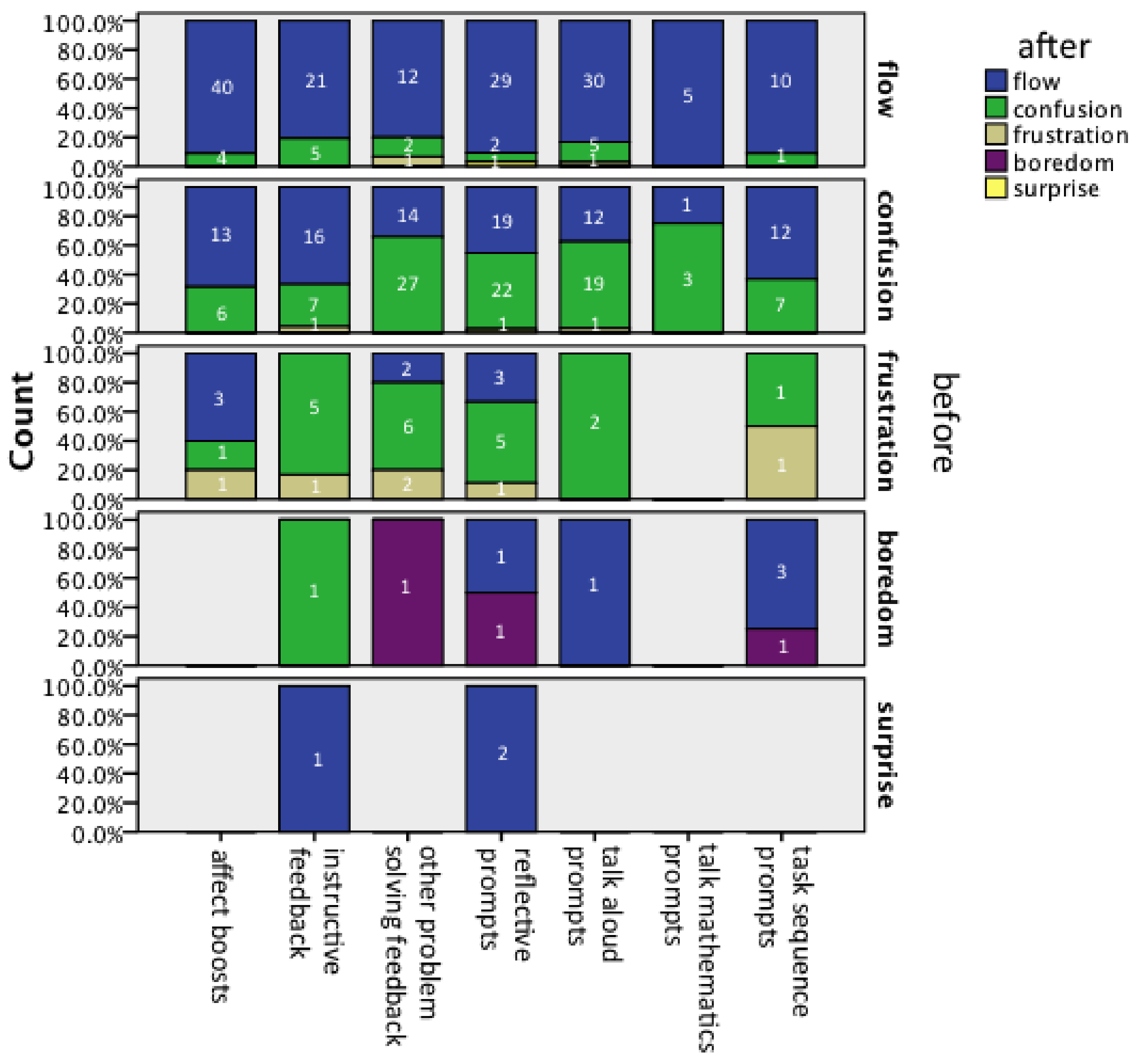


Fig. 2. Students' affective states before and after feedback was provided. Each block shows an affective state *before* feedback was provided. The colour within the bars indicates the type of affective states that occurred *after* the feedback was provided. The number within the bars indicate the number of times the affective state occurred.

Frustration, boredom and surprise There was not sufficient data available when students were frustrated (36 cases), nor when they were bored (9 cases), or surprised (3 cases) to run a statistical test across the different affective states and feedback types.

However, the data indicates that some of the provided feedback types were better able to change the affective state of the student when they were frustrated, bored or surprised, as can be seen in figure 2. For example, 60% of the affect boosts were able to change frustration into flow, followed by reflective prompts 33% and problem solving support 20%.

5 Discussion

The results presented in the previous section show that feedback can enhance students' affective states, and that the impact of the various feedback types mostly depends on the students' affective state before the feedback was provided.

When students were in flow there was no significant difference between the feedback types on whether or not the affective state stayed the same or worsened. This suggests that, when students are in flow, challenging feedback can be provided without negative implications.

However, when students were confused there was a difference between the feedback types on whether the affective state was enhanced, stayed the same or worsened. The feedback types that most effectively moved the student out of a confusion state were affect boosts, instructive, and task sequence prompts. When they were struggling to overcome problems, affect boosts appeared to encourage some students to redouble their efforts without the need for task specific support. We can hypothesise that this enabled students to self-regulate their affect and move forward. As expected, instructive feedback appears to have given the students the next steps that they needed, whereas other problem solving was less successful. Other problem solving feedback seems to have led students to be more confused because of the increased cognitive load caused by them having to understand the hint or the question provided.

While talk aloud prompts and talk maths, encouraged them to vocalize what they are trying to achieve, they appear not to have helped the students address their confusions. Instead, when they were confused, students appeared to have welcomed a new task (the opportunity to abandon the cause of their confusion). While as a strategy this can be pedagogically debatable, there is scope to provide tasks aimed to help them at the same concepts in a different, simpler way or to allow them to practice first some skills in a practice-based rather than exploratory task.

Although there was insufficient data to analyse the impact of the different feedback types on students' affective state when they were frustrated, some tentative observations can be made. For example, it was evident that the affect of students who were frustrated was enhanced whatever the feedback they were provided with. However, it is notable that the frustrated students who were provided affect boosts were most likely to move to a flow. We have other anecdotal evidence in the same scenario with different students that suggest that explicitly addressing affect and helping students to think of their emotions during learning can help them move to confused or to flow state without need for immediate problem solving support.

It is worth noting that compared to other research we may have been unable to detect more negative states, especially boredom, because of the nature of the environment that the students were using – an exploratory learning environment that encouraged them to speak. The combination of unstructured learning and speech might prevent students from becoming bored.

6 Conclusion and future work

The affective state of students can be modified with feedback. There is a difference in the impact of different feedback types according to the affective state the student is in before the feedback was provided. Although there seems not to be too much of a difference when students are in flow, when students were confused different feedback types seem to matter more. While, for example, affect boosts and instructive feedback were able to change confusion into flow, prompting students to use mathematical vocabulary or providing other problem solving support, were associated with the same confused state or even lead to frustration.

In the light of findings like D'Mello et al. [6] for example of the importance of confusion under appropriate conditions in learning, our findings have important implications for learning and teaching in general, and AIED in particular. Problem solving support specifically in exploratory learning environments is difficult to achieve successfully, particularly when students are in a situation that was not previously encountered during a system's design. However, detecting affect may be relatively easier in certain contexts particularly in speech-enabled software like in our case and therefore affective support matters as much, if not more than, problem solving support. In addition, the exact type of support provided when students are frustrated is important. To understand this better we need to investigate more the different types of problem solving support and their combination with affective feedback that can act both as a way to self-regulate affect and take student into a more positive state like confusion or flow.

In our current study we are implying that learning performance is enhanced when students are in a positive affective state. In the future we are planning to evaluate if learning performance will be enhanced when students are moved out of a negative into a positive affective state. Our next step is to train an intelligent system that is able to tailor the type of feedback according to the affective state of the student in order to enhance the learning experience.

References

1. Askeland, M.: Sound-based strategy training in multiplication. European Journal of Special Needs Education 27(2), 201–217 (2012)
2. Baker, R.S.J.d., DMello, S.K., Rodrigo, M.T., Graesser, A.C.: Better to be frustrated than bored: The incidence, persistence, and impact of learners cognitive-affective states during interactions with three different computer-based learning environments. Int. J. Hum.-Comput. Stud. 68(4), 223–241 (2010)

3. Carenini, G., Conati, C., Hoque, E., Steichen, B., Toker, D., Enns, J.: Highlighting interventions and user differences: Informing adaptive information visualization support. In: Proceedings of CHI 14. pp. 1835–1844 (2014)
4. Chi, M.: Self-explaining expository texts: The dual processes of generating inferences and repairing mental models. In: Glaser, R. (ed.) Advances in instructional psychology, pp. 161–238. Mahwah, NJ: Lawrence Erbaum Associates (2000)
5. Conati, C., MacLaren, H.: Empirically building and evaluating a probabilistic model of user affect. User Modeling and User-Adapted Interaction (2009)
6. DMello, S.K., Lehman, B., Pekrun, R., Graesser, A.C.: Confusion can be beneficial for learning. Learning & Instruction 29(1), 153–170 (2014)
7. DMello, S., Craig, S., Gholson, B., Franklin, S., Picard, R., Graesser, A.: Integrating affect sensors in an intelligent tutoring system. In: Affective Interactions: The Computer in the Affective Loop Workshop at IUI 2005. pp. 7–13 (2005)
8. Eynon, R., Davies, C., Holmes, W.: Supporting older adults in using technology for lifelong learning: the methodological and conceptual value of wizard of oz simulations. In: Proceedings of NLC 2012. pp. 66–73 (2012)
9. Kort, B., Reilly, R., Picard, R.: An affective model of the interplay between emotions and learning. In: Proceedings of ICALT 2001. No. 43–46 (2001)
10. Mavrikis, M., Grawemeyer, B., Hansen, A., Gutiérrez-Santos, S.: Exploring the potential of speech recognition to support problem solving and reflection. In: EC-TEL 2014. pp. 263–276 (2014)
11. Mavrikis, M., Gutiérrez-Santos, S.: Not all wizards are from Oz: Iterative design of intelligent learning environments by communication capacity tapering. Computers & Education 54(3), 641–651 (2010)
12. Ocumpaugh, J., Baker, R.S.J.d., Rodrigo, M.M.T.: Baker-Rodrigo Observation Method Protocol (BROMP) 1.0. Training Manual version 1.0. Tech. rep., New York, NY: EdLab. Manila, Philippines: Ateneo Laboratory for the Learning Sciences. (2012)
13. Pekrun, R.: The control-value theory of achievement emotions: Assumptions, corollaries, and implications for educational research and practice. J. Edu. Psych. Rev. pp. 315–341 (2006)
14. Porayska-Pomsta, K., Mavrikis, M., DMello, S., Conati, C., de Baker, R.S.J.: Knowledge elicitation methods for affect modelling in education. I. J. Artificial Intelligence in Education 22(3), 107–140 (2013)
15. Porayska-Pomsta, K., Mavrikis, M., Pain, H.: Diagnosing and acting on student affect: the tutors perspective. UMUAI 18(1), 125-173 (2008)
16. Rosenthal, R., Rosnow, R.: Essentials of Behavioral Research: Methods and data analysis. McGraw Hill, 3rd edn. (2008)
17. Santos, O., Saneiro, M., Salmeron-Majadas, S., J.G., B.: A methodological approach to elicit affective educational recommendataions. In: Proceedings of ICALT 2014 (2014)
18. Shen, L., Wang, M., Shen, R.: Affective e-learning: Using emotional data to improve learning in pervasive learning environment. Educational Technology & Society 12(2), 176–189 (2009)
19. Sweller, J., van Merrienboer, J.G., Paas, G.W.: Cognitive Architecture and Instructional Design. Educational Psychology Review 10, 251–296+ (1998)
20. Woolf, B., Burleson, W., Arroyo, I., Dragon, T., Cooper, D., Picard, R.: Affect-aware tutors: recognising and responding to student affect. Int. J. Learning Technology 4(3-4), 129–164 (2009)

Recognising Perceived Task Difficulty from Speech and Pause Histograms

Ruth Janning, Carlotta Schatten, and Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL), University of Hildesheim
{janning,schatten,schmidt-thieme}@ismll.uni-hildesheim.de

Abstract. Currently, a lot of research in the field of intelligent tutoring systems is concerned with recognising student's emotions and affects. The recognition is done by extracting features from information sources like speech, typing and mouse clicking behaviour or physiological sensors. In former work we proposed some low-level speech features for perceived task difficulty recognition in intelligent tutoring systems. However, by extracting these features some information hidden in the speech input is loosed. Hence, in this paper we propose and investigate speech and pause histograms as features, which preserve some of the loosed information. The approach of using speech and pause histograms for perceived task difficulty recognition is evaluated by experiments on data collected in a study with German students solving mathematical tasks.

Keywords: Intelligent tutoring systems, perceived task difficulty recognition, low-level speech features, speech and pause histograms

1 Introduction

Automatic cognition, affect and emotion recognition is a relatively young and very important research field in the area of adaptive intelligent tutoring systems. Some research has been done to identify useful information sources and appropriate features able to describe student's cognition, emotions and affects. Those information sources can be speech input, written input, typing and mouse clicking behaviour or input from physiological sensors. In former work ([5], [6], [7]) we proposed low-level speech features for perceived task difficulty recognition in intelligent tutoring systems. These features are extracted from the amplitudes of speech input of students interacting with the system and contain for instance the maximal and average length of speech phases and pauses. However, by extracting those features some more fine granulated information contained within the sequence of speech and pause segments is loosed and the question arises if there is a way to create features which preserve the loosed information. Histograms contain much more information than only the maximal, minimal and average value. Hence, in this work we propose and investigate speech and pause histograms as features for perceived task difficulty recognition, i.e. for recognising if a student feels *over-challenged* or *appropriately challenged* by a task. Speech and pause histograms share the advantages of low-level speech features (they do not inherit the error from speech recognition and there is no need that students use words related to emotions or affects, see also sec. 2) and avoid to lose information hidden in the sequences of speech and pause segments.

2 Related Work

For the purpose to recognise emotion or affect in speech one can distinct linguistics features, like n-grams and bag-of-words, and low-level features like prosodic features, disfluencies, e.g. speech pauses ([5], [6]), (see e.g. [17]) or articulation features ([7]). If linguistics features are not extracted from written but from spoken input, a transcription or speech recognition process has to be applied to the speech input before emotion or affect recognition can be conducted. Linguistic features for affect and emotion recognition from conversational cues were presented and investigated e.g. in [10] and [11]. Low-level features are used in the literature for instance for expert identification, as in [18], [13] and [8], for emotion and affect recognition as in [12] and [5], [6], [7] or for humour recognition as in [15]. The advantage of using low-level features like disfluencies is that instead of a full transcription or speech recognition approach only for instance a pause identification has to be applied before computing the features. That means that one does not inherit the error of the full speech recognition approach. Furthermore, these features are independent from the need that students use words related to emotions or affects. Another kind of features which is independent from the need that students use words related to emotions or affects are features gained from information about the actions of the students interacting with the system (see e.g. [9]) like features extracted from a log-file (see e.g. [2], [16], [14]). In [9] such kind of features is used to predict whether a student can answer correctly questions in an intelligent learning environment without requesting help and whether a student's interaction is beneficial in terms of learning. Also the keystroke dynamics features used in [4] belong to this kind of features. In [4] emotional states were identified by analysing the rhythm of the typing patterns of persons on a keyboard. A further possibility of gaining features is using the information from physiological sensors as for instance in [1]. However, bringing sensors into classrooms is time consuming and expensive and one has to cope with students' acceptance of the sensors.

3 Speech and Pause Histograms

As mentioned above, in this paper we investigate the ability of speech and pause histograms for perceived task difficulty recognition. How these speech and pause histograms are created from students' speech input is described in sec. 3.2 and the data which we used for our experiments is described in the next section.

3.1 Data

We conducted a study in which the speech and actions of ten 10 to 12 years old German students were recorded and their perceived task-difficulties were

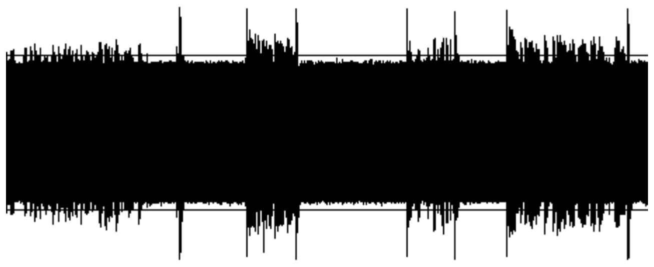


Fig. 1. Graphic of the decibel scale of an example sound file of a student. The two straight horizontal lines indicate the threshold.

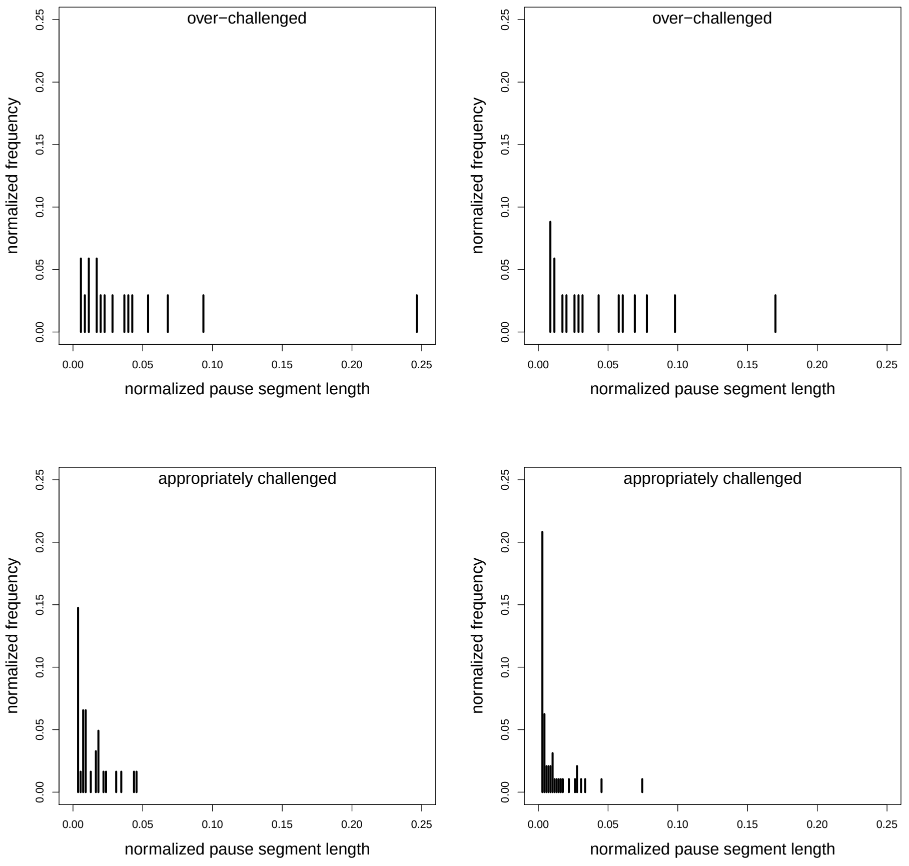


Fig. 2. Normalised pause histograms for a task of four different students, where two are labelled as *over-challenged* and the other two as *appropriately challenged*.

reported per task. The labelling of these data was done on the one hand concurrently by a human tutor and on the other hand retrospectively by a second reviewer (with a Cohen's kappa for inter-rater reliability of 0.747, $p < 0.001$). Divergences in the both labellings were clarified later on by discussions between the reviewers. During the study a paper sheet with fraction tasks was shown to the students and they were asked to paint – by means of a software for painting with a computer – their solution and they were prompt to explain aloud their observations and answers. The fraction tasks were subdivided into similar subtasks and covered exercises like assigning fractions to coloured parts of a circle or rectangle, reducing, adding or subtracting fractions and fraction equivalence. Originally, there were 10 tasks with 1 up to 10 subtasks but not each task was seen by each student. We made a screen recording to record the painting of the students and an acoustic recording to record the speech of the students. The screen recordings were used for the retrospective annotation. The acoustic speech recordings, consisting of 10 wav files with a length from 15 up to 20 minutes, were used to gain the speech and pause histograms. The data collection resulted in 36 examples (tasks) labelled with *over-challenged* (12 examples) or *appropriately challenged* (24 examples), respectively 48 examples (24 of class *appropriately challenged*, 24 of class *over-challenged*) after applying oversampling to the smaller set of examples of class *over-challenged* to eliminate the unbalance in the data.

3.2 Histograms for Classification

In the above mentioned study we observed that the children often exhibited longer pauses of silence while thinking about the problem when they were *over-challenged* or produced fewer and shorter pauses while communicating when they were *appropriately challenged*. Hence, in this paper we investigate information about pauses and speech segments within the speech input of students in connection with the perceived task difficulty. The first step to gain this information is to segment the acoustic speech recordings for identifying segments containing speech and segments corresponding to pauses. The most easy way to do this is to define a threshold on the decibel scale as done e.g. in [8]. For our study of the data we also used a threshold, which was estimated manually. The manual threshold estimation was done by extracting the amplitudes of the sound files, computing the decibel values and generating a graphic of it like the one in fig. 1. Subsequently, it was investigated which decibel values belong to speech and which ones to pauses to create from this information an appropriate threshold. By means of this threshold the pause and speech segments can be extracted. From the pause segments the pause histogram is generated by counting how often each possible pause length occur. This pause histogram is then normalised, to make the pause histograms of different speech inputs (of different students, different tasks and different lengths) comparable. The normalisation is done by dividing each occurring pause length by the length of the whole speech input as well as dividing the frequency of each occurring pause length by the number of all speech and pause segments, so that the resulting values stem from the interval between 0 and 1. The same is done with the speech segments for generating the speech histogram. Examples of normalised pause histograms and speech histograms are shown in fig. 2 and fig. 3. The examples stem from the speech input for a task of four different students, where two were labelled as *over-challenged* and the other two as *appropriately challenged*. One can see some

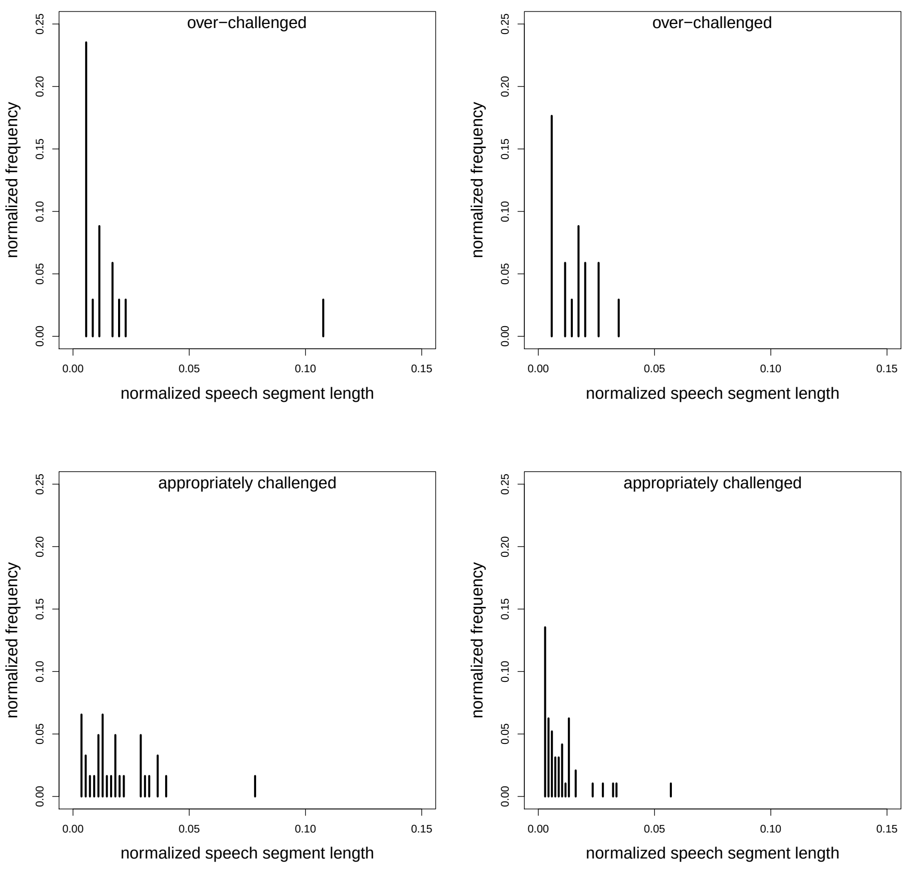


Fig. 3. Normalised speech histograms for a task of four different students, where two are labelled as *over-challenged* and the other two as *appropriately challenged*.

differences between the histograms of the *over-challenged* students and the *appropriately challenged* students as well as some similarities of the examples with the same label. The pause histograms of the *appropriately challenged* students show that there are a lot of very small pauses within their speech, but no very large pauses. The pause histograms of the *over-challenged* students in contrast report long pauses and less smaller pauses than for the *appropriately challenged* students. In the speech histograms one can see that the *over-challenged* students used a lot of very small speech segments of the same length whereas for *appropriately challenged* students there is a large variance in the speech segment length. In the following section we investigate how these histograms can be used for classifying the speech input of a student for a task as either *over-challenged* or *appropriately challenged*.

4 Experiments

To investigate if the above described speech and pause histograms are applicable for distinguishing *over-challenged* and *appropriately challenged* students we conducted experiments with the perprocessing and settings described in the following section. The experimental results are reported in sec. 4.2.

4.1 Preprocessing and Experimental Settings

To be computationally comparable the normalised histograms still need to be preprocessed, or more explicitly generalised, as the set of possibly occurring segment lengths is infinite (it is a real value between 0 and 1). Hence, we divide the x-axis (the different normalised lengths of pause or speech segments) into a number of equal sized intervals, the *buckets*. Each occurring normalised segment length is then put into the bucket to whose interval it belongs. The number of buckets, or the bucket size respectively, is a hyper parameter and in the experiments we investigated different values for that parameter, i.e. we conducted experiments with 2 up to $1,000,000$ buckets (bucket size 0.5 up to 1.0E-6) where the numbers of buckets are multiples of the numbers by which 100 is divisible without remainder. A comparison of two different histograms can now be done by comparing the content of each bucket in both histograms, that means that for each bucket the normalised frequencies of segments belonging to that bucket are compared. In our experiments we compute the difference between two histograms by computing the differences between the frequencies in all buckets by means of the root mean square error (RMSE):

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{b}(b_i(H_x) - b_i(H_y))^2}{b}}\,, \tag{1}$$

where H_x and H_y are the two histograms to compare, $b_i(H_x)$ and $b_i(H_y)$ are the normalised frequency values belonging to bucket b_i of H_x and H_y and b is the number of buckets. For deciding to which class (*over-challenged* or *appropriately challenged*) a histogram belongs we applied the K-Nearest-Neighbour (KNN) approach. KNN (see e.g. [3]) classifies an example by a majority vote of its neighbours, that is the example is assigned to the class most common among its K nearest neighbours. These K nearest neighbours are the K closest training examples in the feature space. The *closeness* in our case is measured by means of the RMSE. That is a histogram is assigned to that class to which the majority of the K closest (in terms of RMSE) histograms belongs. K is a further hyper parameter and also for that parameter we tried out different values, i.e. we conducted experiments with a number of 1 up to 35 neighbours where that value is an odd number less than the number of unique examples. For the evaluation we used a Leave-one-out cross-validation in the experiments. The results of our experiments with pause and speech histograms are discussed in the next section.

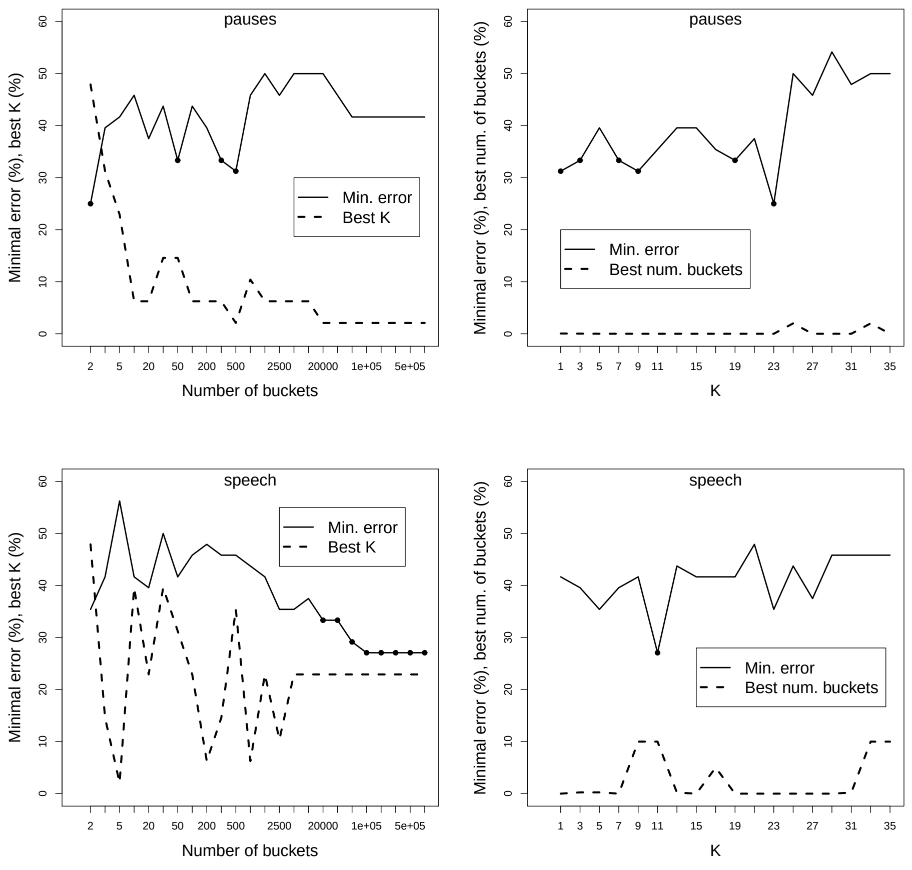


Fig. 4. Different numbers of buckets and different numbers K of neighbours mapped to the minimal classification error (%) and the belonging best value for K (% of the number of examples) and for the number of buckets (% of the max. number of buckets) for pause and speech histograms.

4.2 Experiments with Speech and Pause Histograms

As mentioned above, we conducted experiments with different numbers of buckets and different values for the K nearest neighbours. In fig. 4 we report the minimal classification error and the belonging best value of K for each bucket number as well as the the minimal classification error and the belonging best number of buckets for each value of K for the pause and the speech histograms. The *classification error* is the number of incorrectly classified histograms divided by the number of all histograms. The black dots in fig. 4 indicate the best results which are also reported in tab. 1 and 2. As one can see in fig. 4 for the

Table 1. Number of buckets, bucket size, K, classification error and F-measures of class *over-challenged* & *appropriately challenged* of the experiments with pause histograms with best result (classification error < 34%, black dots in fig. 4).

Number of buckets	2	2	2	50	250	500
Bucket size	0.5	0.5	0.5	0.02	0.004	0.002
K	9	19	23	7	3	1
Error (%)	31.25	33.33	25.00	33.33	33.33	31.25
F-measure	0.57, 0.82	0.55, 0.80	0.67, 0.83	0.59, 0.63	0.59, 0.57	0.60, 0.71

Table 2. Number of buckets, bucket size, K, classification error and F-measures of class *over-challenged* & *appropriately challenged* of the experiments with speech histograms with best result (classification error < 34%, black dots in fig. 4).

Number of buckets	20000	25000	50000	100000	200000	250000	500000	1000000
Bucket size	5.0E-5	4.0E-5	2.0E-5	1.0E-5	5.0E-6	4.0E-6	2.0E-6	1.0E-6
K	11	11	11	11	11	11	11	11
Error (%)	33.33	33.33	29.17	27.08	27.08	27.08	27.08	27.08
F-measure	0.57, 0.73	0.57, 0.73	0.62, 0.78	0.64, 0.77	0.64, 0.77	0.64, 0.77	0.64, 0.77	0.64, 0.77

pause histograms a smaller number of buckets delivers the best results whereas for the speech histograms the number of buckets has to be large, i.e. a more fine granulated division of the x-axis is needed for good results. The reason might be that the pause histograms of *over-challenged* and *appropriately challenged* students are easier distinguishable as in the pause histogram of an *over-challenged* student there are typically long pause segments which usually do not occur in the speech of *appropriately challenged* students (see also fig. 2). As fig. 3 shows, speech histograms of *over-challenged* and *appropriately challenged* students are not so easy to distinct. Tab. 1 and 2 show the results of the best choices for hyper parameter K and number of buckets and reports the classification error as well as the F-measures of both classes (*over-challenged* and *appropriately challenged*).

The F-measure is a value between 0 and 1 and the closer it is to 1 the better. It is the harmonic mean between the ratio of examples of a class c which are correctly recognised as members of that class (*recall*) and the ratio of examples classified as belonging to class c which actually belong to class c (*precision*). In our experiments the F-measures of class *appropriately challenged* are better than those of class *over-challenged*. The reason could be that originally there were more examples of class *appropriately challenged* and we just oversampled class *over-challenged* to receive a balanced example set. Nevertheless, the best classification errors of 25% and 27.08% and F-measures $0.67, 0.83$ and $0.64, 0.77$ in tab. 1 and 2 indicate that speech and pause histograms are applicable for perceived task difficulty recognition.

5 Conclusions and Future Work

We proposed and investigated speech and pause histograms, build from the sequences of speech and pause segments within the speech input of students, as features for perceived task difficulty recognition. To evaluate the approach of using the histograms for distinguishing *over-challenged* and *appropriately challenged* students we applied a K-Nearest-Neighbour classification delivering a classification error of 25% for pause histograms and 27.08% for speech histograms. Next steps will be to try out other classification approaches, for instance from time series classification. Furthermore, the information from the speech histograms and pause histograms could be combined to reach a better classification performance, e.g. by ensemble methods.

Acknowledgements. This work is co-funded by the EU project iTalk2Learn (www.italk2learn.eu) under grant agreement no. 318051.

References

1. Arroyo, I., Woolf, B.P., Burelson, W., Muldner, K. , Rai, D. and Tai, M.: A Multimedia Adaptive Tutoring System for Mathematics that Addresses Cognition, Metacognition and Affect. In International Journal of Artificial Intelligence in Education, Springer, Vol. 24, pp. 387–426 (2014)
2. Baker, R.S.J.D., Gowda, S., Wixon, M., Kalka, J., Wagner, A., Salvi, A., Aleven, V., Kusbit, G., Ocumpaugh, J. and Rossi,L.: Towards Sensor-Free Affect Detection in Cognitive Tutor Algebra. In Proceedings of the 5th International Conference on Educational Data Mining (EDM 2012), pp. 126–133 (2012)
3. Cover, T. and Hart, P.: Nearest neighbor pattern classification. EEE Transactions on Information Theory, Vol. 13(1), pp. 21–27, doi:10.1109/TIT.1967.1053964 (1967)
4. Epp, C., Lippold, M. and Mandryk, R.L.: Identifying Emotional States Using Keystroke Dynamics. In Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems (CHI 2011), pp. 715–724 (2011)
5. Janning, R., Schatten, C., Schmidt-Thieme, L.: Multimodal Affect Recognition for Adaptive Intelligent Tutoring Systems. In Extended Proceedings of the 7th International Conference on Educational Data Mining (EDM 2014), pp. 171–178 (2014)

6. Janning, R., Schatten, C., Schmidt-Thieme, L.: Feature Analysis for Affect Recognition Supporting Task Sequencing in Adaptive Intelligent Tutoring Systems. In Proceedings of the European Conference on Technology Enhanced Learning (EC-TEL 2014), pp. 179–192 (2014)
7. Janning, R., Schatten, C., Schmidt-Thieme, L. and Backfried, G.: An SVM Plait for Improving Affect Recognition in Intelligent Tutoring Systems. In Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI) (2014)
8. Luz, S.: Automatic Identification of Experts and Performance Prediction in the Multimodal Math Data Corpus through Analysis of Speech Interaction. Second International Workshop on Multimodal Learning Analytics, Sydney Australia (2013)
9. Mavrikis, M.: Data-driven modelling of students interactions in an ILE. In Proceedings of the International Conference on Educational Data Mining (EDM 2008), pp. 87–96 (2008)
10. DMello, S.K., Craig, S.D., Witherspoon, A., McDaniel, B. and Graesser, A.: Automatic detection of learners affect from conversational cues. User Model User-Adap Inter, DOI 10.1007/s11257-007-9037-6 (2008)
11. D'Mello, S.K. and Graesser, A.: Language and Discourse Are Powerful Signals of Student Emotions during Tutoring. IEEE Transactions on Learning Technologies, Vol. 5(4), pp. 304–317, IEEE Computer Society (2012)
12. Moore, J.D., Tian, L. and Lai, C.: Word-Level Emotion Recognition Using High-Level Features. Computational Linguistics and Intelligent Text Processing (CICLing 2014), pp. 17–31 (2014)
13. Morency, L.P., Oviatt, S., Scherer, S., Weibel, N. and Worsley, M.: ICMI 2013 grand challenge workshop on multimodal learning analytics. In Proceedings of the 15th ACM on International conference on multimodal interaction (ICMI 2013), pp. 373–378 (2013)
14. Pardos, Z.A., Baker, R.S.J.D, San Pedro, M., Gowda, S.M. and Gowda, S.M.: Affective States and State Tests: Investigating How Affect and Engagement during the School Year Predict End-of-Year Learning Outcomes. Journal of Learning Analytics, Vol. 1(1), Inaugural issue, pp. 107–128 (2014)
15. Purandare, A. and Litman, D.: Humor: Prosody Analysis and Automatic Recognition for F * R * I * E * N * D * S *. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006), pp. 208–215 (2006)
16. San Pedro, M.O.C., Baker, R.S.J.D., Bowers, A. and Heffernan, N.: Predicting College Enrollment from Student Interaction with an Intelligent Tutoring System in Middle School. In Proceedings of the 6th International Conference on Educational Data Mining (EDM 2013), pp. 177–184 (2013)
17. Schuller, B., Batliner, A., Steidl, S. and Seppi, D.: Recognising realistic emotions and affect in speech: State of the art and lessons learnt from the first challenge. Speech Communication, Elsevier (2011)
18. Worsley, M. and Blikstein, P.: What's an Expert? Using Learning Analytics to Identify Emergent Markers of Expertise through Automated Speech, Sentiment and Sketch Analysis. In Proceedings of the 4th International Conference on Educational Data Mining (EDM '11), pp. 235–240 (2011)

Analyzing Student Action Sequences and Affect While Playing Physics Playground

Juan Miguel L. Andres[1], Ma. Mercedes T. Rodrigo[1],
Ryan S. Baker[2], Luc Paquette[2], Valerie J. Shute[3], Matthew Ventura[3]

[1] Ateneo de Manila University, Quezon City, Philippines
[2] Teachers College, Columbia University, New York, NY, USA
[3] Florida State University, Tallahassee, FL, USA
{mandres, mrodrigo}@ateneo.edu,
baker2@exchange.tc.columbia.edu, luc.paquette@gmail.com,
{vshute, mventura}@fsu.edu

Abstract. Physics Playground is an educational game that supports physics learning. It accepts multiple solutions to most problems and does not impose a stepwise progression through the content. Assessing student performance in an open-ended environment such as this is therefore challenging. This study investigates the relationships between student action sequences and affect among students using Physics Playground. The study identified most frequently traversed student action sequences and investigated whether these sequences were indicative of either boredom or confusion. The study found that boredom relates to poor performance outcomes, and confusion relates to sub-optimal performance, as evidenced by the significant correlations between the respective affective states, and the student action sequences.

Keywords: Affect modeling, action sequences, boredom, confusion, Physics Playground

1 Introduction

Physics Playground (PP) is an educational game that immerses learners in a choice-rich environment for developing intuitive knowledge about simple machines. As the environment does not impose a stepwise sequence on the learner, and because some problems can have multiple solutions, learners have the freedom to explore, attempt to solve, or abort problems as they wish. The challenge these types of environments impose on educators is that of assessment. Within such an open-ended system, how do educators and researchers assess learning as well as the quality of the learning process?

This study focuses its attention on two main phenomena: student learning and student affect. Student learning within PP refers to how well a player can understand the concepts surrounding four simple machines through their efficient execution in attempting to solve levels, as evidenced by the badges they earn.

Student affect refers to experiences of feelings or emotions. In this study, the affective states of interest are confusion and boredom, as prior studies have shown them to relate significantly with learning [4, 10]. Confusion is uncertainty about what to do next [5]. Confusion is scientifically interesting because it has a positive and negative dimension, wherein it either spurs learners to exert effort deliberately and purposefully to resolve cognitive conflict, or leads learners to become frustrated or bored, and may lead to disengagement from the learning task altogether [7].

Boredom, on the other hand, is an unpleasant, transient affective state in which the individual feels a pervasive lack of interest in and difficulty concentrating on the current activity [8]. Boredom has been a topic of interest because of the negative effects usually associated with it, such as poor long-term learning outcomes when students are not provided any scaffolding [10] and its being characteristic of less successful students [11].

A study conducted by Biswas, Kinnebrew, and Segedy [2] investigated frequently traversed sequences of student actions using bottom-up, data-driven sequence mining, the results of which contributed to the development of performance- and behavior-based learner models. The analyses in this paper seek to perform similar sequence-mining methods in order to find student sequences that inform either of the affective states of interest.

This study conducted data-driven sequence-mining analyses to answer the following research questions:

1. What were the frequently traversed student action sequences among students playing Physics Playground?
2. Are these action sequences indicative of either boredom or confusion?

The analyses in this study are limited to the data collected during gameplay of Physics Playground from six data gathering sessions conducted at a public school in Quezon City in 2013. Data is limited to the interaction logs generated by the game as well as human observation of affect as logged by two coders trained in the Baker-Rodrigo-Ocumpaugh Monitoring Protocol [9].

2 Methodology

2.1 Participant Profile

Data were gathered from 60 eighth grade public school students in Quezon City, Philippines. Students ranged in age from 13 to 16. Of the participants, 31% were male and 69% were female. As of 2011, the school had 1,976 students, predominantly Filipino, and 66 teachers. Participants had an average grade on assignments of B (on a scale from A to F).

2.2 Physics Playground

Physics Playground (PP) is an open-ended learning environment for physics that was designed to help secondary school students understand qualitative physics. Qualitative physics is a nonverbal, conceptual understanding of how the physical world operates [12].

PP has 74 levels that require the player to guide a green ball to a red balloon. An example level is shown in Fig. 1. The player achieves this goal by drawing agents (ramps, pendulums, springboards, or levers) or by nudging the ball to the left or right by clicking on it. The moment the objects are drawn, they behave according to the law of gravity and Newton's 3 laws of motion [12].

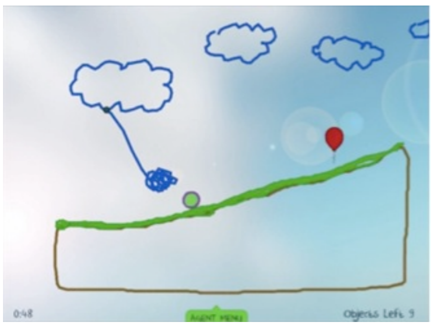


Fig. 1. Example PP level.

Performance Metrics. Gold and silver badges are awarded to students who manage to solve a level. A gold badge is given to a student who is able to solve the level by drawing a number of objects equal to the particular level's par value (i.e., the minimum number of objects needed to be drawn to solve the level). A student who solves a level using more objects will earn a silver badge. A student earns no badge if he was not able to solve the level. Many levels in PP have multiple solutions, meaning a player can solve the level using different agents.

2.3 Interaction Logs

During gameplay, PP automatically generates interaction log files. Each level a student plays creates a corresponding log file, which tracks every event that occurs as the student interacts with the game. Per level attempt, PP tracks begin and end times, the agents used, and the badges awarded upon level completion. PP also logs the *Freeform Objects* that player draw, or objects that cannot be classified as any of the four agents. The physics agents within PP are as follows:

- Ramp, any line drawn that helps to guide a ball in motion,
- Lever, an agent that rotates around a fixed point, usually called a fulcrum,
- Pendulum, an agent that directs an impulse tangent to its direction of motion,
- Springboard, an agent that stores elastic potential energy provided by a falling weight.

2.4 The Observation Protocol

The Baker-Rodrigo-Ocumpaugh Monitoring Protocol (BROMP) is a protocol for quantitative field observations of student affect and engagement-related behavior, described in detail in [9]. The affective states observed within Physics Playground in this study were engaged concentration, confusion, frustration, boredom, happiness, delight, and curiosity. The affective categories were drawn from [6].

BROMP guides observers in coding affect through different utterances, body language, and interaction with the software specific to each affective state. A total of seven affective states were coded, however, this study focuses on three: concentration, confusion, and boredom. These were identified as follows:

1. Concentration — immersion and focus on the task at hand, leaning toward the computer and attempting to solve the level, a subset of the flow experience described in [5].
2. Confusion — scratching his head, repeatedly attempting to solve the same level, statements such as "I don't understand?" and "Why didn't it work?"
3. Boredom — slouching, sitting back and looking around the classroom for prolonged periods of time, statements such as "Can we do something else?" and "This is boring!"

Following BROMP, two trained observers observed ten students per session, coding students in a round-robin manner, in 20-second intervals throughout the entire observation period of 2 hours. During each 20-second window, both BROMP observers code the current student's affect independently. If the student exhibited two or more distinct states during a 20-second observation window, the observers only coded the first state. The inter-coder reliability for affect for the two observers in the study was acceptably high with a Cohen's Kappa [3] of 0.67. The typical threshold for certifying a coder in the use of BROMP is 0.6, a standard previously used in certifying 71 coders in the use of BROMP (e.g., [9]).

The observers recorded their observations using HART, or the Human Affect Recording Tool. HART is an Android application developed to guide researchers in conducting quantitative field observations according to BROMP, and facilitate synchronization of BROMP data with educational software log data.

2.6 Data Collection Process

Before playing PP, students answered a 16-item multiple-choice pretest for 20 minutes. Students then played the game for 2 hours, during which time two trained observers used BROMP to code student affect and behavior on the HART application. A total of 4,320 observations were collected (i.e., 36 observations per participant per each of the two observers). After completing gameplay, participants answered a 16-item multiple-choice posttest for 20 minutes. The pretest and posttest were designed to assess knowledge of physics concepts, and have been used in previous studies involving PP [12].

To investigate how students interacted with PP, the study made use of the interaction logs recorded during gameplay to analyze student performance. Of the 60 participants, data from 11 students were lost because of faulty data capture and corrupted log files. Only 49 students had complete observations and logs. As a result, the analysis in this paper is limited to these students, and the 3,528 remaining affect observations. Engaged concentration was observed 72% of the time, confusion was observed 8% of the time, and boredom and frustration were observed 7% of the time. Happiness, delight, and curiosity comprise the remaining 6% of the observation time.

3 Analyses and Results

3.1 Agent Sequences

All PP-generated logs were parsed and filtered to produce a list containing only the events relevant to the study. Sequences were then separated into one of two categories: 1) silver sequences, or the sequences that ultimately led to a silver badge, which comprised 44% of all level attempts, and 2) unsolved sequences, or the sequences that led to the student quitting the level without finding a solution, which comprised 39% of all level attempts. Sequences that ended in gold badges were dropped from the analysis because they only comprised 17% of all level attempts.

Every time a student earns a badge after solving a level, the badge is awarded for one of the four agents (e.g., a player is awarded a silver ramp badge for solving the level using a ramp, and another player is awarded a gold pendulum badge for solving another level using a pendulum). We tracked the agents the badges were awarded for per level, and used this list of badges to relabel the sequences based on correctness. If the level awarded a badge for an agent, that agent was labeled as `correct` for that level; if not, the agent was labeled as `wrong` for the level. For example, on a level that awarded badges for springboards and levers, a sequence of `Lever > Ramp > Springboard > Level End (silver-springboard)` would be relabeled as `correct > wrong > correct > Level End (silver)`.

The relabeling was done because most of the sequences were level-dependent, that is, a majority of some sequences appeared on only one or two levels. By relabeling based on correctness, we were able to ensure level-independence among sequences. Sequences were tabulated and their frequencies calculated (i.e., how many times each of the 49 students traversed each of the sequences). We calculated for distribution of sequence frequencies, and the sequences we found to occur rarely (i.e., less than 30% of the population traversed them) were dropped from the analysis. We found that the gold sequences occurred rarely, which was another reason they were dropped from the analysis. The resulting silver and unsolved sequences can be found in Tables 1 and 2, respectively, along with the frequency means and standard deviations.

Table 1 lists the top 7 silver sequences within PP, which were traversed by more than 30% of the study's population. The Sequences column shows what the respective sequences look like, and the Frequency column shows the average number of times the 49 students traversed them and the standard deviations.

Highlighted sequences showed significant correlations with either boredom or confusion, as discussed further in Section 3.2. Table 2 is presented in the same manner.

Table 1. Top 7 silver sequences, their traversal frequency means, and standard deviations.

	Sequences	Frequency	
		Mean	SD
1	`correct>Level End (silver)`	3.53	2.34
2	`Level End (silver)`	2.61	2.33
3	`wrong>Level End (silver)`	1.90	1.37
4	`correct>correct>Level End (silver)`	1.61	1.15
5	`wrong>correct>Level End (silver)`	0.90	1.01
6	`correct>correct>correct>Level End (silver)`	0.80	1.00
7	`wrong>correct>correct>Level End (silver)`	0.69	0.77

The silver sequences in Table 1 show signs of experimentation, with students playing around with the correct and incorrect agents to solve the levels, as seen in sequences 5 and 7. Sequences 1, 4, and 6 show students using the correct agents, but are unable to earn gold badges. This suggests that students, while knowing which agents to use, do not have a full grasp of the physics concepts surrounding the agents' execution. Sequence 3 shows students using wrong objects to solve the levels. While this may suggest that students are still struggling to understand how the agents work and which agent would best solve a level given the ball and the balloon's positions, this may have also been caused by the PP logger labeling the objects they drew as freeform objects, and not one of the correct agents.

Sequence 1 shows the students drawing only the correct agent, but are still unable to earn a gold badge. The sequence-mining algorithm only pulled events related to drawing any of the four main agents, which are enumerated in Section 2.3. Drawing a lever or a springboard, for example, would require drawing more than one component. A lever requires the fulcrum, the board, and the object dropped on the board to project the ball upwards. In order for the agent to work, it has to be executed correctly (i.e., the board must be long enough, with the fulcrum in the right position, and the object dropped on the board must be heavy enough to propel the ball into the air). Sequence 1 may have been caused by students drawing the correct agent, but improperly executing it. For example, the student may not have drawn the right-sized weight to drop on the lever, and thus had to draw another. While drawing another weight to drop on the lever counts towards the level's object count, it was not logged as a separate event by the sequence mining analysis because the player did not draw another agent, only a component of it. Sequence 2, on the other hand, is suspect because despite the student drawing no objects to solve a level, he ends up with only a silver badge. This was most likely caused by the improper logging of the game. The top 7 most frequently traversed silver sequences account for 58% of the total number of silver sequences.

Table 2. Top 6 unsolved sequences, their traversal frequency means, and standard deviations.

	Sequences	Frequency	
		Mean	SD
1	`Level End (none)`	10.69	8.17
2	`wrong>Level End (none)`	1.55	1.65
3	`correct>Level End (none)`	1.29	1.50
4	`wrong>wrong>Level End (none)`	0.45	0.65
5	`correct>correct>Level End (none)`	0.41	0.73
6	`wrong>correct>Level End (none)`	0.39	0.57

Table 2, which shows the top 6 unsolved sequences, shows signs of students giving up. Sequence 1 shows students giving up without even drawing a single object, which could have been caused by one of two things: 1) the student saw the level and decided to quit without attempting to solve it, or 2) again, the logger did not log the objects correctly. This sequence is similar to one of the silver sequences in that no objects were drawn. What makes them different, however, is what the sequences ultimately led to. The silver sequences ended in a silver badge, and the unsolved sequences ended in the student earning no badge. The majority of the sequences listed in Table 2 show students experimenting mainly with wrong objects, whether agents or freeform objects. This implies that the students are lacking in the understanding of how to solve the levels. Sequences 3 and 5 are interesting because it is unclear whether or not the students understood the concepts of the agents. That is, students were drawing the correct agents, but could not get the ball to reach the balloon. Despite drawing one or two correct agents, the students decided to give up and quit. The top 6 unsolved sequences account for 81% of the total number of unsolved sequences.

3.2 Relationship with Affect

We computed frequencies for each of the 13 sequences that the 49 students traversed. Correlations were then run between each of the 13 arrays and the incidences of confusion and boredom. Because the number of tests introduces the possibility of false discoveries, Storey's adjustment [13] was used as a post-hoc control, which provides a *q*-value, representing the probability that the finding was a false discovery. Tables 3 and 4 show the results. Highlights and asterisks (*) were used on significant findings ($q \leq 0.05$).

Table 3 lists the top 7 most frequently traversed silver sequences, from left to right. The sequences these header numbers represent can be found in Table 1. The table shows the correlation between each of the top 7 silver sequences using a metric that represents the percentage of all attempts that match each of the sequences, the percentage of time the students were observed to be confused (r, con), and the percentage of time the students were observed to be bored (r, bor).

Table 4 is presented in the same manner, with sequence information in Table 2 for the top 6 unsolved sequences.

Table 3. Correlations between top 7 silver sequences, confusion, and boredom.

	Top 7 silver sequences						
	1	2	3	4	5	6	7
r, con	-0.33	0.23	0.41*	0.03	0.17	0.54*	0.28
r, bor	-0.20	-0.17	-0.19	-0.05	0.14	-0.19	-0.20

Table 3 shows two significant positive correlations between confusion and the silver sequences. The two sequences showed signs of lesser understanding of the agents. Sequence 3 shows students using only a wrong object to solve a level, which may have been caused either by incorrect object labeling (e.g., PP logged a ramp as a Freeform Object), or the student found a different way of solving the level. Like in most learning environments, players are able to game the system – or systematically misuse the game's features to solve a level [1] – within PP through stacking. Stacking is done when players draw freeform objects to either prop the ball forward or upward, which may have been the case in sequence 3. Sequence 6 shows students drawing only correct agents. These sequences having significant correlations with confusion may imply lesser understanding among confused students as the they are not only dealing with proper agent execution, but also with deciding which agent would best solve the level. Despite the challenges faced by these students, however, they still managed to find a solution to the level. Our findings suggest that the inability to grasp the physics concepts surrounding the agents is a sign of confusion.

Table 4. Correlations between top 6 unsolved sequences, confusion, and boredom.

	Top 6 unsolved learning sequences					
	1	2	3	4	5	6
r, con	-0.17	0.00	-0.12	-0.01	-0.06	0.04
r, bor	-0.12	0.13	0.12	-0.03	0.48*	0.06

Table 4 shows that one of the most frequently traversed unsolved sequences has a significant positive correlation with boredom. This sequence shows students using only correct agents, but ultimately deciding to give up. This may have been caused by the inability to execute the agents correctly, which may imply that, unlike confused students, bored students were not likely to exert additional effort to try to solve the level or understand proper agent execution. As mentioned previously, boredom has been found to have significant relationships with negative performance outcomes. In this case, sequences all ultimately led to disengagement: students quitting the level before finding a solution, showing signs of giving up and lack of understanding of any of the four agents.

4 Conclusions and Future Work

This study sought to identify the most frequently traversed student action sequences among eighth grade students while interacting with an education game for physics called Physics Playground. Further, the study sought to investigate how these sequences may be indicative of affective states, particularly boredom and confusion, which have been found to significantly affect student learning.

Data-driven sequence mining techniques were conducted to identify most frequently traversed actions sequences in two categories: the sequences that would eventually lead the student to a silver badge, and the paths that would eventually lead the student to not earning a badge.

In the silver sequences, students played around with freeform objects and some of the four agents in attempting to solve the level. The study found confusion to correlate significantly with two of the silver sequences, which supports previous findings regarding the relationship between confusion and in-game achievement, which suggest that because students are unable to grasp the concepts surrounding the agents and their executions, students resort to finding other solutions.

In the unsolved sequences, students would give up and quit without finding a solution, despite already using the correct agents to solve the level. The study found boredom to correlate significantly with one of the unsolved sequences. This finding supports the literature that has shown that boredom relates to poor learning outcomes. This work provides further evidence that boredom and disengagement from learning go hand-in-hand.

This study provides specific sequences of student actions that are indicative of the boredom and confusion, which has implications on the design and further development of Physics Playground. This study also contributes to the literature by providing empirical support that boredom and confusion are affective states that influence performance outcomes within open-ended learning environments, and are thus affective states that learning environments must focus on detecting and providing remediation to. We found that both bored and confused students will tend to continuously use correct agents in attempting to solve levels, but execute them incorrectly. The difference between the two, however, is that confused students tend to end up solving the level, while bored students give up.

The analyses run in this paper were part of a bigger investigation, and as such, there are several interesting ways forward in light of our findings. The paper aims for its findings to contribute to the creation of a tool that can automatically detect affect given a sequence of student interactions, and provide necessary remediation in order to curb student experiences of boredom.

Relationship analyses run between student action sequences and incidences of affect in this paper were done through correlations. However, findings were not able to determine whether boredom or confusion occurred more frequently during specific action sequences. We want to find out whether boredom or confusion occurred before, during, or after the students' execution of the action sequences, and in doing so, see whether or not the affective states were causes or effects of the action sequence executions. We are currently investigating this relationship in a separate study.

Acknowledgements. We would like to thank the Ateneo Center for Educational Development, Carmela C. Oracion, Christopher Ryan Adalem, and the officials at Krus Na Ligas High School, Jessica O. Sugay, Dr. Matthew Small, and the Gates Foundation Grant #OP106038 for collaborating with us.

References

1. Baker, R. S., Corbett, A. T., Koedinger, K. R., & Wagner, A. Z. (2004). Off-task behavior in the cognitive tutor classroom: when students game the system. In Proceedings of the SIGCHI conference on Human factors in computing systems (pp. 383-390). ACM.
2. Biswas, G., Kinnebrew, J. S., & Segedy, J. R. (2011). Using a Cognitive/Metacognitive Task Model to analyze Students Learning Behaviors.
3. Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. Educational and Psychological Measurement, 20 (1960), 37-46.
4. Craig, S., Graesser, A., Sullins, J., & Gholson, B. (2004). Affect and learning: an exploratory look into the role of affect in learning with AutoTutor. Journal of Educational Media, 29(3), 241-250.
5. Csikszentmihalyi, M. (1990). Flow: The psychology of optimal experience. New Y ork: Harper Perennial.
6. D'Mello, S. K., Craig, S. D., Witherspoon, A., McDaniel, B., & Graesser, A. (2005). Integrating affect sensors in an intelligent tutoring system. In Proceedings of the Workshop on Affective Interactions: The computer in the affective loop workshop, International conference on intelligent user interfaces (pp. 7- 13). New York: Association for Computing Machinery.
7. D'Mello, S., Graesser, A. (2012). Dynamics of affective states during complex learning. Learning and Instruction, 22(2): 145-157.
8. Fisherl, C. D. (1993). Boredom at work: A neglected concept. Human Relations, 46(3), 395-417.
9. Ocumpaugh, J., Baker, R.S., Rodrigo, M.M.T. (2015) Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP) 2.0 Technical and Training Manual. Technical Report. New York, NY: Teachers College, Columbia University. Manila, Philippines: Ateneo Laboratory for the Learning Sciences.
10. Pardos, Z. A., Baker, R. S., San Pedro, M., Gowda, S. M., & Gowda, S. M. (2014). Affective States and State Tests: Investigating How Affect and Engagement during the School Year Predict End-of-Year Learning Outcomes. Journal of Learning Analytics, 1(1), 107-128.
11. San Pedro, M. O. Z., d Baker, R. S., Gowda, S. M., & Heffernan, N. T. (2013, January). Towards an understanding of affect and knowledge from student interaction with an Intelligent Tutoring System. In Artificial Intelligence in Education (pp. 41-50). Springer Berlin Heidelberg.
12. Shute, V. J., Ventura, M., & Kim, Y. J. (2013). Assessment and Learning of Qualitative Physics in Newton's Playground. The Journal of Educational Research, 106(6), 423-430.
13. Storey, J.D. (2002). A direct approach to false discovery rates. Journal of the Royal Statistical Society, Series B, 64: 479-498.

La Mort du Chercheur: How well do students' subjective understandings of affective representations used in self-report align with one another's, and researchers'?

Wixon[1], Danielle Allessio[2], Jaclyn Ocumpaugh[3], Beverly Woolf[2], Winslow Burleson[4] and Ivon Arroyo[1]

[1]Worcester Polytechnic Institute, Worcester Massachusetts
{mwixon, iarroyo}@wpi.edu
[2]University of Massachusetts, Amherst Massachusetts
{allessio@educ, bev@cs}.umass.edu
[3]Teachers College, Columbia University, New York, New York
jocumpaugh@wpi.edu
[4]New York University, New York, New York
Wb50@nyu.edu

Abstract. We address empirical methods to assess the reliability and design of affective self-reports. Previous research has shown that students may have subjectively different understandings of the affective state they are reporting [18], particularly among younger students[10]. For example, what one student describes as "extremely frustrating" another might see as only "mildly frustrating." Further, what students describe as "frustration" may differ between individuals in terms of valence, and activation. In an effort to address these issues, we use an established visual representation of educationally relevant emotional differences [3, 8, 25]. Students were asked to rate various affective terms and facial expressions on a coordinate axis in terms of valence and activation. In so doing, we hope to begin to measure the variability of affective representations as a measurement tool. Quantifying the extent to which representations of affect may vary provides a measure of measurement error to improve reliability.

Keywords: Affective States; Intelligent Tutoring Systems; Reasons for Affect

1 Introduction

The evaluation of students' affective states remains an incredibly difficult challenge. While recognized as a key indicator of student engagement [14, 17, 26], there remains no clear gold-standard for identifying an affective state, leading to researchers such as Graesser & D'Mello [13] to call for greater attention to the theoretical stances that certain research methods entail. A full theoretical review is beyond the scope of this paper; instead, the current work presents a pilot study designed to empirically evaluate the reliability of two different types of affective self-reports in an educational context. Reliability is measured both in terms of inter-rater reliability (the degree of agreement between students), and “inter-method” reliability (i.e. given words or facial expressions as representations of affective states, which representation produces more consistent results).

A considerable body of research has been devoted to affect computing, and in particular to affect detection in educational software [9]. Progress has been made with methods that include self-report [8, 10], physiological sensors [1, 24], video-based retrospective reports [5, 15], text-based [11, 19], and field observation [16, 23] data. However, much of this research evaluates success based on the ability of a model to predict when a training label is present or absent, without giving deeper consideration to questions about the appropriateness of the training label itself.

Even within limited to the body of research that relies on self-report research, there are serious concerns about how methodological decisions might impact student responses. In addition to issues about the frequency and timing of surveys, one primary area of concern is that students may have subjectively different understandings of the state they are reporting [19], an effect that is likely to be even greater among younger students [10]. For example, Graesser and D’Mello [13] have suggested that a students’ tolerance of cognitive disequilibrium (e.g., confusion or frustration) is probably conditioned by their knowledge and prior success with the topic they are interacting with. Further, what students describe as “frustration” in itself may differ between individuals in terms of dimensional component measures of affect: valence, activation, and dominance. The former two dimensions are typically used to differentiate affective states [4], and the latter used in some cases [7].

In this study, we explore these interpretative issues using three different types of representations that have been employed in previous self-report studies: words, facial expressions, and dimensional measures. In particular, we are interested in verifying that students’ understanding of the meaning of these representations aligns with interpretations of these labels that are present in the literature (as constructed by experts). To this end, we use dimensional measures (valence & activation) to compare how students respond to both linguistic representations and pictorial representations, further testing hypotheses that the latter might be more appropriate for surveying students [19, 21, 22]. Our goal is to determine the extent to which this student population shows variance in the interpretation of these two different types of representations, since substantial variation in student perception should be accounted for in subsequent research. Last, while we might achieve researcher agreement in terms of methods and terminology for self-reported affects, that will do little good if there is a large degree of variance in terms of our subject pool’s agreement on the meaning of these constructs.

1.1 Methods

Students surveyed included eighty one 7th graders from two Californian middle schools in a major city (among the 30 most populous cities in California), where a majority of census respondents identified as Hispanic or Latino and median household income was within one standard deviation of California’s overall median household income. They were surveyed at the end of the academic year.

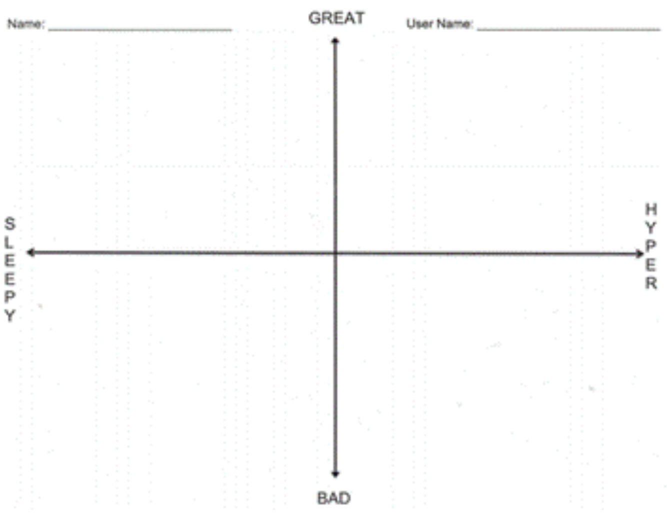


Fig. 1. Blank Valence & Activation Sheet given to Students

Students were asked to place both textual and facial representations of affect on an XY axis of Activation=X Valence=Y. Textual representations of affect were selected based on the affective states that have been used in the past [2, 12], that corresponded to quite different levels of activation x valence according to us researchers, so that words would theoretically cover all quadrants. These terms and their researcher-hypothesized valence x arousal placements included: Angry (low valence x high activation), Anxious (low valence x high activation), Bored (low valence x low activation), Confident (high valence x low activation), Confused (med-low valence x med-high activation), Enjoying (high valence x medium activation), Excited (high valence x high activation), Frustrated (low valence x high activation), Interested (high valence x medium activation) and Relieved (high valence x med-low activation). In general, it was clear to the researchers which word corresponded to which face, with a few exceptions, such as the level of activation that should be associated to enjoying and interest. An established set of emoticons were chosen from previous affective research [8] that corresponded to extreme emoticon states of activation x valence x dominance. While the emoticons possessed these three attributes, our participants were asked only to orient them based on activation and valence.

Each student was presented with a sheet of paper depicting a coordinate axis with activation from “sleepy” to “hyper” on the x-axis and “bad” to “great” on the y-axis. These terms were used to express what valence and activation mean experientially, using language that children are familiar with and could relate to. Activation is then expressed more as a physical experience of arousal, while Valence is expressed not as much as a physical experience but as a judgment of the positive or negative nature of the experience. Later, during coding, these axes were mapped discretized into a seven point scale of -3 to 0 to +3 at either extreme of each axis, defining a grid of 7 x 7.

Students were also given stickers for the 10 separate affective terms: Angry, Anxious, Bored, Confident, Confused, Enjoying, Excited, Frustrated, Interested, & Relieved, see Figure 2; as well as 8 stickers to depict each extreme emoticon expression from the ends of each of the 3 axis coordinate systems including: pleasure, activation, and dominance [8]. Students placed each of these stickers on their coordinate axes according to where they felt each term or emoticon should be placed with respect to valence and activation.

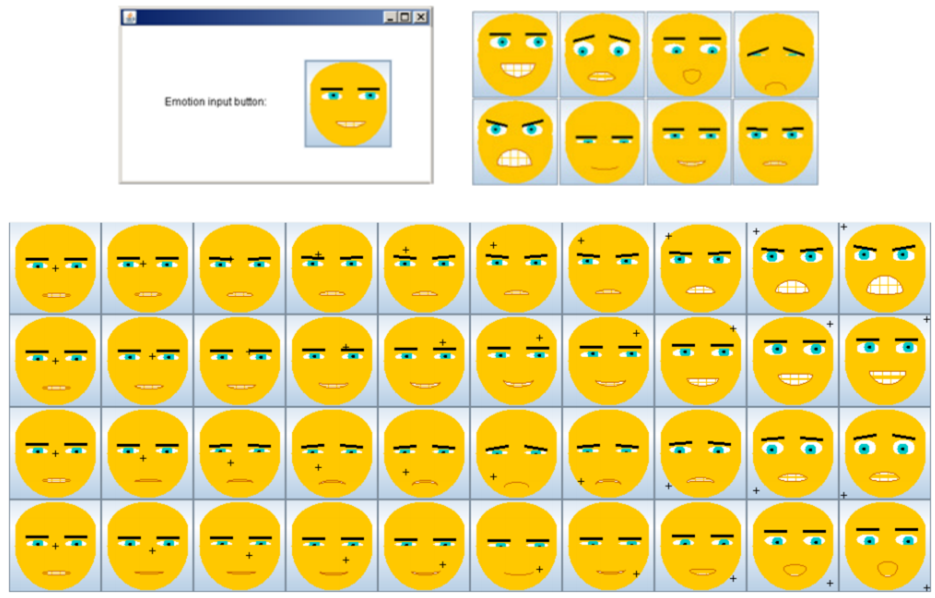


Fig. 2. Directly from Broekens, & Brinkman, 2013 [8]. Top left displays the affect button inter-face. Students use the cursor to change the expression in the inter-face. Depending on their actions, one of 40 affective expressions may be displayed; these expressions, shown across the bottom of this figure, are designed to vary based on pleasure (valence), activation, and dominance (PAD for brevity). From left to right first row: elated (PAD=1,1,1), afraid (-1,1,-1), surprised (1,1,-1), sad (-1,-1,-1). From left to right second row: angry (-1,1,1), relaxed (1,-1,-1), content(1,-1,1), frustrated (-1,-1,1). Top right displays PAD extremes, which serve as the basis for this research.

2 Results

Mean positioning results are displayed visually in figure 3, corresponding to the position that each word or emoticon sticker was placed averaged across all respondents. Missing data occurred in which students may not have placed every sticker. On average any given term or emoticon was missing 16.6 reports, with a maximum of 23 students of 81 missing reports for boredom, frustration, and relief. The average student was only missing 3.7 out of 18 terms and emoticons from their sheet, and there were 5 students who turned in completely blank sheets.

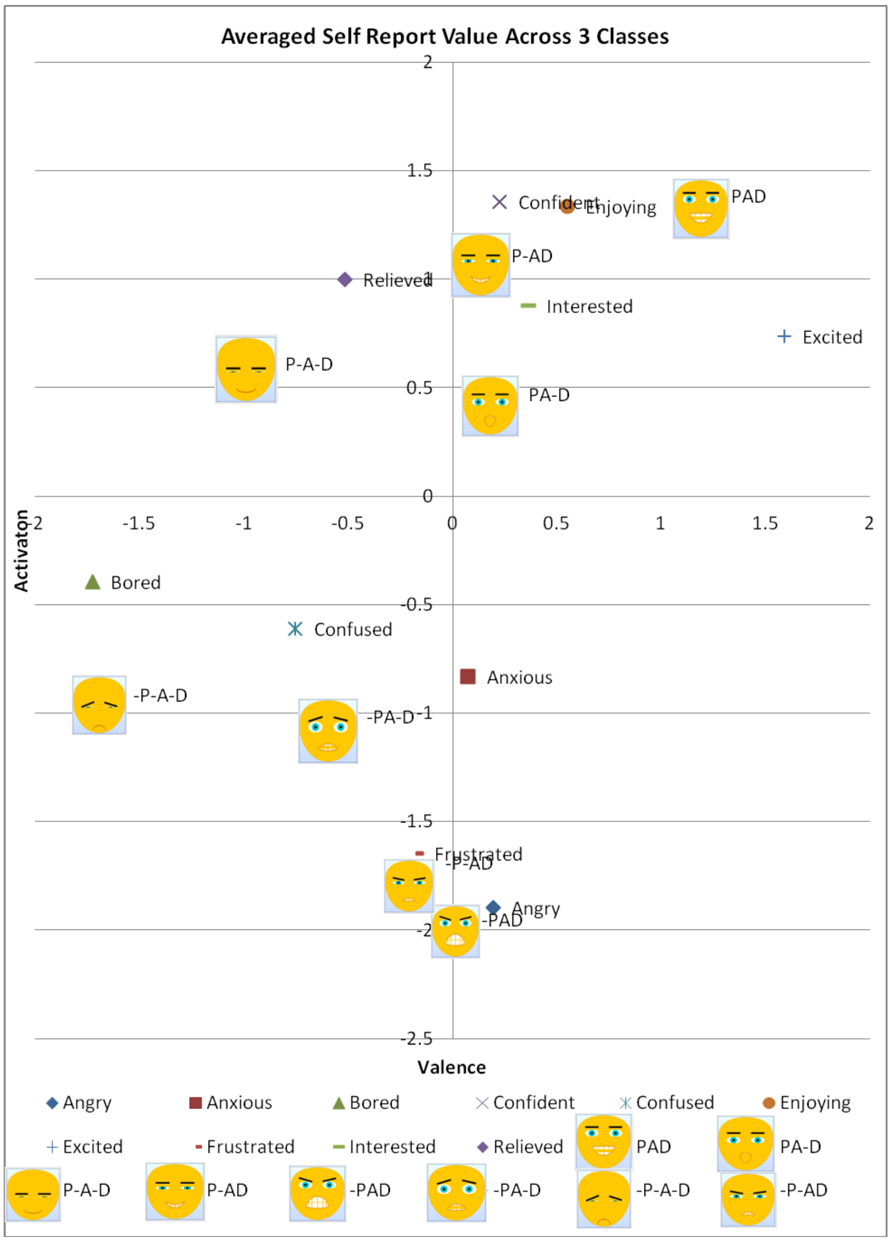


Fig. 3. Averaged Placement of Text and Emoticon Stickers

Interestingly, the placement of -PAD and -P-AD (negative sign indicating most extreme negative activation, pleasure, dominance, lack of a negative indicating most extreme positive, see figure 2 caption) match up with their respective terms "Angry" and "Frustrated" very closely. However, while both seem to be at the extreme end of negative valence, on average both seem to be viewed as fairly neutral in terms of activation by students. Although all emoticons and terms fall under the expected half of the coordinate axes in terms of valence (i.e. those we would expect to be pleasurable are categorized as above the origin, those displeasurable below it), activation does not follow this trend. For example anxiety is rated as neutral activation. One possible explanation, consistent with the results, is that students may be grouping activation and dominance together as a single measure. Emoticons with both negative activation and dominance were rated most negatively in terms of activation, those with either negative activation or dominance tended to fall in the middle, and the rating with all positive PAD was the emoticon with the highest rated activation.

Text or Emoticon	Activation Mean (StdDev)	Valence Mean (StdDev)
Angry	0.19 (1.09)	-1.9 (0.99)
Anxious	0.07 (1.78)	-0.87 (1.19)
Bored	-1.72 (1.28)	-0.4 (1.02)
Confident	0.23 (1.22)	1.35 (0.99)
Confused	-0.75 (1.36)	-0.61 (1.12)
Enjoying	0.55 (1.18)	1.34 (1.14)
Excited	1.59 (1.04)	0.74 (1.26)
Frustrated	-0.17 (1.33)	-1.65 (1.05)
Interested	0.36 (1.34)	0.88 (0.98)
Relieved	-0.52 (1.43)	1 (1.12)
Face_PAD	1.25 (1.3)	1.38 (1.13)
Face_PA-D	0.28 (1.86)	0.47 (0.93)
Face_P-A-D	-0.89 (1.57)	0.61 (0.91)
Face_P-AD	0.2 (1.26)	1.11 (1.08)
Face_-PAD	0.05 (0.95)	-1.95 (0.93)
Face_-PA-D	-0.5 (1.39)	-1.01 (1.01)
Face_-P-A-D	-1.61 (1.41)	-0.91 (1.11)
Face_-P-AD	-0.12 (1.15)	-1.69 (0.89)
Average	-0.08 (1.33)	-0.12 (1.05)

Table 1. Means and Standard Deviations of Students' placement of stickers.

One key goal of this work was to determine the degree of variance between students in terms of where they placed each term or emoticon. Given any affective term, there was little difference between the standard deviation for terms (mean S.D for terms = 1.20) and faces (mean S.D. for faces = 1.18). However, there was a larger difference between the standard deviation in activation (mean S.D for activation of terms or faces = 1.33) and valence (mean S.D for valence of terms or faces = 1.05), suggesting that students may have a greater degree of agreement in regarding rating the valence of affective representations than the activation it produces in them, which is consistent with the finding that affective representations fall on the division between positive and negative valence as we would categorize them, but not necessarily in terms of activation.

3 Discussion

The results presented in this article highlight a few different conclusions: a) students did not necessarily match emoticons or affective terms to the quadrants where researchers would have placed them, mostly in relation to activation; b) there is a large variation across these middle-school students in terms of where they placed a specific emotion within the axes of valence x arousal.

Characterizing researcher common expectations for arousal or activation is difficult, as many researchers only tentatively suggest how emotional states may be characterized in terms of activation. Pekrun found data to support boredom being somewhat deactivating, [18]. Russell [25] explores the components of affect and offers a few hypotheses which are summarized in figure 1 of Baker et al 2010 [3] wherein boredom is characterized as deactivating, while frustration, surprise, and delight are characterized as activating. Broekens' [8] emoticons follow the scheme outlined in the figure 2 caption: elation, fear, surprise, and anger are seen as activating, while sadness, relaxation, contentment, and frustration are seen as deactivating.

Students seem to agree that delight or elation is highly activating along with excitement, and boredom is deactivating along with sadness and relaxation. However, we found that students viewed an emoticon of fear as deactivating, and other affective states placed relatively close to neutral in terms of activation.

There are a few points of methodological concern. Firstly, the order that the students' place their stickers may be important: beyond a simple priming effect of considering one term/emoticon before another, by placing one item first students are changing the affordance of the coordinate axis itself by adding a milestone in the form of a term or emoticon. In future research, we could consider including fewer stimuli for placement or giving students a clean chart for each stimuli.

A second point of concern is one of validity. The terms, emoticons, and even the coordinate axis itself are abstract descriptors of affective states, which in this experiment are divorced from the actual experiences students may be having.

By placing our study outside the experimental environment we are likely reducing the validity of this work in exchange for simplicity of study design (i.e. not requiring students to respond with faces and words on the axis at various points in their experience).

The work of Bieg et al. [6] tells a much larger story than recommending against self-reports out of context. Out of context self-reports were found to bias in a consistent direction as compared to in context self-reports. However we maintain this method is "less valid" rather than "invalid". Further, if we take into consideration the savings in class time an out of context self-report may actually be a better study design choice in some cases. It is our position that establishing more quantitative comparisons of reliability will yield better relative comparisons of validity and allow for improved study design.

This argument can be extended to affective research in general in the distinction between emotional experience and appraisal. We conceptualize the experience itself as the construct, and the cognitive appraisal process as a means of communicating that experience. The appraisal may be performed to send communication (e.g. having an experience and generating a representation of that experience for others), or receive communication (e.g. identify a representation as signifying an emotional state).

From this standpoint we suggest that the fewer steps of appraisal exist, the greater the face validity of an appraisal is in terms of reflecting an emotional experience. This is consistent with the findings of [6] wherein aggregate appraisal may differ from immediate contextual appraisal and we tend to view immediate appraisal as having greater face validity. This hypothesis also lends credence to the belief that external appraisal of an unconsciously generated representation (which may still be unconsciously meant to communicate an experience), in the form of facial expressions may be more valid than self-report measures wherein experiences are appraised by both subject and researcher. However, while passing through multiple appraisals may risk loss of information, the quality and richness of the appraisal may also play a role.

While validity remains very difficult to establish with regard to affect by testing "inter-method" or "representational" reliability perhaps we can building convergent and discriminant validity: multiple representations indicating the same construct across multiple participants. We maintain that reliability and validity are continuous rather than discrete traits of models. Therefore, we wish to reach consensus on methods of determining reliability and validity and then begin applying them to methods of inferring the experience of emotion. This work is a means of determining reliability between appraisals of representations of emotion rather than reliability of appraisals of emotions themselves. This is to say that matching particular facial expression to their personal lexicon of categorical affective terms, a high degree of agreement may validate the relationship between depictions of affect textually and facially, but not between either of those representations and the experience of an emotion.

A potential way towards greater validity and reliability could be to cognitively induce an emotional experience by asking students to respond to how they would feel given a particular situation (e.g. "Report on how you'd feel if you failed a math test."). Of course there may be a distinction between induced affect and "organic" affect, further there will be a broad degree of subjectivity based on how individual students might feel about any given situation. Therefore the variance in responses could be attributed at least to two types of factors: those pertaining to both how students' believe they would feel in a given context, and those pertaining to students' ability to report that subjective experience through self-report measures. While there isn't a clear way to disambiguate between which type of factor is responsible for the variance here, such an approach might be able to establish a conservative maximum of error in self-report measurements, because two students might have very different feelings about failing a math exam. In essence, we have measured variance in reliability here, not validity.

Finally, while reliability of self-report measures should inform their design, there may be cases of diminishing returns where a slight improvement in reliability has heavy costs for implementation workload, response time, or other practical concerns. We need not pick the measure with the highest available reliability; however it would be good to have some empirical handle on the relative reliabilities of different types of self-report measures. Perhaps the greatest thing to come out of this work would be future collaborations which might better address these concerns.

4 References

1. AlZoubi, O., D'Mello, S. K., & Calvo, R. A. (2012). Detecting naturalistic expressions of nonbasic affect using physiological signals. Affective Computing, IEEE Transactions on, 3(3), 298-310.
2. Arroyo, I., Woolf, B.P., Royer, J.M. and Tai, M. (2009b) 'Affective gendered learning companion', Proceedings of the International Conference on Artificial Intelligence and Education, IOS Press, pp.41–48.
3. Baker, R.S.J.d., D'Mello, S.K., Rodrigo, M.M.T., Graesser, A.C. (2010) Better to Be Frustrated than Bored: The Incidence, Persistence, and Impact of Learners' Cognitive-Affective States during Interactions with Three Different Computer-Based Learning Environments. *International Journal of Human-Computer Studies*, 68 (4), 223-241.
4. Barrett, L. F. (2004). Feelings or Words? Understanding the Content in Self-Report Ratings of Experienced Emotion. Journal of Personality and Social Psychology, 87(2), 266–281.
5. Bosch, N., D'Mello, S., Baker, R., Ocumpaugh, J., Shute, V., Ventura, M., & Zhao, W. (2015). Automatic Detection of Learning-Centered Affective States in the Wild. In Proceedings of the 2015 International Conference on Intelligent User Interfaces (IUI 2015). ACM, New York, NY, USA.
6. Bieg, M., Goetz, T., & Lipnevich, A.A. (2014). What Students Think They Feel Differs from What They Really Feel – Academic Self-Concept Moderates the Discrepancy between Students' Trait and State Emotional Self-Reports. PLoS ONE 9(3): e92563.
7. Bradley, M. M., & Lang, P. J. (1994). Measuring emotion: the Self-Assessment Manikin and the Semantic Differential. Journal of Behav Ther Exp Psychiatry, 25, 49-59.
8. Broekens, J., & Brinkman, W.-P. (2013). AffectButton: a method for reliable and valid affective support. International Journal of Human-Computer Studies, 71(6), 641-667.
9. Calvo, R. A., D'Mello, S., Gratch, J., & Kappas, A. (Eds.) (2015). The Oxford Handbook of Affective Computing. Oxford University Press: New York, NY.
10. Conati, C., & Maclaren, H. (2009). Empirically building and evaluating a probabilistic model of user affect. User Modeling and User-Adapted Interaction, 19(3), 267-303.
11. D'Mello, S. , Craig, S. D., Sullins, J., & Graesser, A. C. (2006). Predicting Affective States expressed through an Emote-Aloud Procedure from AutoTutor's Mixed-Initiative Dialogue. International Journal of Artificial Intelligence in Education, 16(1), 3-28.
12. D'Mello, S., & Graesser, A. C. (2012). Language and Discourse Are Powerful Signals of Student Emotions during Tutoring. IEEE Transactions on Learning Technologies, 5(4): 304–317.

13. Graesser, A., & D'Mello, S. (2011). Theoretical perspectives on affect and deep learning. In New perspectives on affect and learning technologies (pp. 11-21). Springer New York.
14. Linnenbrink-Garcia, L., & Pekrun, R. (2011). Students' emotions and academic engagement. Introduction to the special issue. Contemporary Educational Psychology, 36, 1–3.
15. McDaniel, B. T., D'Mello, S., King, B. G., Chipman, P., Tapp, K., & Graesser, A. C. (2007). Facial features for affective state detection in learning environments. In Proceedings of the 29th Annual Cognitive Science Society (pp. 467-472).
16. Ocumpaugh, J., Baker, R.S., Rodrigo, M.M.T. (2015) Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP) 2.0 Technical and Training Manual.. Technical Report. New York, NY: Teachers College, Columbia University. Manila, Philippines: Ateneo Laboratory for the Learning Sciences.
17. Pardos, Z. A., Baker, R. S., San Pedro, M. O., Gowda, S. M., & Gowda, S. M. (2013). Affective states and state tests: Investigating how affect throughout the school year predicts end of year learning outcomes. Proc. 3rd Int.Conf. Learning Analytics & Knowledge, 117-124.
18. Pekrun, R., Goetz, T., Daniels, L. M., Stupnisky, R. H., & Perry, R. P. (2010). Boredom in achievement settings: Exploring control–value antecedents and performance outcomes of a neglected emotion. *Journal of Educational Psychology, 102*(3), 531-549.
19. Porayska-Pomsta, K., Mavrikis, M., D'Mello, S., Conati, C., Baker, R.S.J.d. (2013) Knowledge Elicitation Methods for Affect Modeling in Education. International Journal of Artificial Intelligence in Education, 22 (3), 107-140.
20. Porayska-Pomsta, K., Mavrikis, M., & Pain, H. (2008). Diagnosing and acting on student affect: the tutor's perspective. User Modeling and User-Adapted Interaction, 18(1-2), 125-173.
21. Read, J., McFarlane, S., and Cassey, C. (2002). Endurability, engagement and expectations: Measuring children's fun. In Proceedings of International Conference for Interaction Design and Children.
22. Read J. C. and MacFarlane, S.(2006). Using the fun toolkit and other survey methods to gather opinions in child computer interaction. In Proceedings of the 2006 conference on Interaction design and children (IDC '06). ACM, New York, NY, USA, 81-88.
23. Rodrigo, M. M. T., Baker, R. S. J. d., Lagud, M. C. V., Lim, S. A. L., Macapanpan, A. F., Pascua, S. A. M. S., et al. (2007). Affect and Usage Choices in Simulation ProblemSolving Environments. In R. Luckin, K. R. Koedinger & J. Greer (Eds.), Proceeding of the 2007 conference on Artificial Intelligence in Education: Building Technology Rich Learning Contexts that Work (Vol. Frontiers in Artificial Intelligence and Applications 158). Amsterdam: IOS Press.
24. Rowe, J. P., Mott, B. W., & Lester, J. C. (2014) It's All About the Process: Building Sensor-Driven Emotion Detectors with GIFT. In Generalized Intelligent Framework for Tutoring (GIFT) Users Symposium (GIFTSym2) (p. 135).
25. Russell J.A, Barrett L,F. (1999) Core affect, prototypical emotional episodes, and other things called emotion: dissecting the elephant. J. Pers. Soc. Psychol. 76(5):805–19.
26. San Pedro, M.O.Z., Baker, R.S.J.d., Bowers, A.J., Heffernan, N.T. (2013) Predicting College Enrollment from Student Interaction with an Intelligent Tutoring System in Middle School. Proceedings of the 6th International Conference on Educational Data Mining, 177-184.

Cultural aspects related to motivation to learn in a Mexican context

Erika-Annabel Martínez-Mirón[1,*], Genaro Rebolledo-Méndez[2]
[1]Universidad Politécnica de Puebla, Puebla, México
*Corresponding Author: erika.martinez@uppuebla.edu.mx
[2]Universidad Veracruzana, Xalapa, México
g.rebolledo@gmail.com

Abstract. The development of motivationally intelligent tutoring systems has been based on a variety of motivational models from the psychology field. These models mainly consider characteristics from de areas of values, expectancies and feelings [1]. However, this paper proposes to take into account some cultural aspects when operationalizing such models. The basis of this proposal is presented from the perspective of some cultural aspects that effect career choice, in particular for a Mexican context.

Keywords: Motivation, career choice, Mexican cultural context

1 Introduction

Research in motivation to learn when using educational technology has operationalized different motivational models found in the psychological literature in order to develop motivationally intelligent tutoring systems. According to these models, motivationally aware tutoring systems should combine expertise and knowledge about user's cognitive, affective, meta-cognitive and meta-affective levels in order to appropriately react and be able to favor user's learning [2, 3]. That is, these models should mainly consider characteristics from the areas of values, expectancies and feelings [1].

However, this paper argues also for the inclusion of other aspects that have been seldom taken into account so far. We refer to cultural aspects inherent to each group of individuals from a certain background. Since there is evidence that students from different cultural origin react to the same motivational strategy in a different way [4, 5, 6] or have different attitudes for online assessment [7], the cultural aspect of learning with technology becomes an important issue. For instance, if a female student from a highly gender-stereotyped cultural background is asked to attend a course considered to be strongly oriented to men, then she might perceived to be in the wrong course and probably will not exert her maximum effort. Or even she might believe that her role in society is to be protected by someone, and she attends courses just to be in the possibility to meet that expectation. It will not matter what motivational strategy the teacher uses, since the female student's cultural belief is in an apparently superior level and she will only be concerned to learn at the minimum, just to continue studying until meeting her protector [8].

In order to develop the arguments to support the inclusion of cultural aspects in the design of motivationally-aware tutoring systems, the following sections describe some of these elements within a Mexican context from the perspective of career choice, based on the findings that instrumental motivation is an important predictor for course selection, career choice, and performance [9, 10]. That is, students may pursue to perform well in some tasks because they are important for future goals, even if the student is not interested on the task.

2 Motivation, career guidance and cultural context

Motivation is related to the student's desire to participate in the learning process. Current research findings suggest that motivational constructs do change over time [11, 12, 13] and/or contexts [14, 15, 16]. In particular, it is well documented that cultural differences affect achievement motivation [4, 5, 6].

We believe that if teachers truly want to promote the success of all students, they must recognize how achievement motivation varies culturally within the population it serves.
Similarly, career counseling must incorporate different variables and different processes to be effective for students from different cultural contexts. Career counseling is defined as "the process of assisting individuals in the development of a life-career with focus on the definition of the worker role and how that role interacts with other life roles" [17].
According to Rivera [18], there are characteristics that prevail among Hispanic/Latino American children and adolescents, such as: A) Restraint of feelings, particularly anger and frustration; B) Limited verbal expressions toward authority figures; C) Preference for closer personal space; avoidance of eye contact when listening or speaking to authority figures; D) Relaxation about time and punctuality; and immediate short-term goals; E) Collective, group identity; interdependence; cooperative rather than competitive; emphasis on interpersonal relations. To certain extent, these characteristics can be considered part of one of the four sources of information, social persuasion, included in the model of the Socio Cognitive Career Theory [19], (see Table 1). This framework conceptualizes career choice as a process with multiple stages and different sources of information. We propose that cultural aspects of the Mexican context might have an impact not just the process of choosing a career, but on the way students undertake their learning activities as described in the following paragraphs.

Table 1. Sources of information proposed in the model of social cognitive influences on career choice behavior [19]

Source of information	**Description**
Performance accomplishment	Success in performing the target task or behavior
Vicarious learning or modeling	To watch others who could perform the target behavior successfully.
Emotional arousal	Anxiety when performing the target behavior
Social persuasion	Support and encouragement from others in the process of performing the target behavior.

2.1 Machismo

There is growing research supporting that achievement differences between genders are smaller during early years of school or being reduced [20]. The succession of career behaviors for women is far more complex than for men. In particular, in Mexican students, the complexities might lay in the cultural aspect of machismo. In Mendoza's review [21], machismo is defined as a strong sense of masculine pride, and it is suggested that machismo should be considered in any Latino study, but it is often forgotten. The social behavior pattern associated to machismo includes the expectation of men being caring, responsible, decisive, strong of character, and the protector of probably extended family. At the same time, negative aspects of machismo include aggressiveness, physical strength, emotional insensitivity, and a womanizing attitude towards the opposite sex.
Galanti [22], cited in [21], surveyed a group of Latino students who reported that the relationship between male and female would be of protector and protected. More specifically, according to them, the role of the traditional Hispanic woman is to look after the family; her job is to cook, clean, and care for the children. Other characteristics of a good wife include submission and obedience to her husband's orders without questioning him but rather standing behind whatever he decides, even if she disagrees. She must also be tolerant of his behavior. Taking into account these views it is understandable that women´s career choice might be influenced by the fulfillment of this profile rather than freely choosing a career that may imply a great amount of dedication. In some Mexican contexts, women may prefer to undertake studies that are less demanding. Women also must strive to overcome obstacles such as gender discrimination and sex stereotyping. For instance, Gallardo-Hernández *et. al.* reported the results of a questionnaire applied to 637 first-year medical nutrition, dentistry and nursing students

[23]. The findings suggest that among women of low socioeconomic strata, more traditional gender stereotypes prevail which lead them to seek career choices considered feminine. Among men, there is a clear relationship between career choice, socioeconomic level and internalization of gender stereotypes.

2.2 Social orientation

Cooperative learning is very important for Mexicans [24]. They do not seem to openly want to show what they know for fear of embarrassing those who do not know [25]. It is not common in a Hispanic family to encourage children to excel over siblings or peers but rather, it is considered bad manners. It is worth noting that most of the studies reported have taken into account the Mexican context around Mexican American students but no studies so far focus on comparison between this population and a Mexican population living in Mexico. Nevertheless, their findings can, to some extent, be considered valid for Mexican population. For instance, Ojeda and Flores [26] considered the educational aspirations of 186 Mexican American high school students to test a portion of social-cognitive career theory [19]. Their results indicated that perceived educational barriers significantly predicted students' educational aspirations above and beyond the influence of gender, generation level, and parents' education level. Similarly, Flores, Romero and Arbona [27] found that Mexican American men and women with high measures of ethnic loyalty might be at risk for perceiving social costs of pursuing a higher education.

2.3 Perception of time and career guidance

Mexicans are oriented toward present time; they are focused on “right now” rather than on the past or on future events or outcomes. They often live the phrase “Dios dirá” or "God will tell," that is, time is relative. To arrive late for an engagement is called in the southwest "Mexican time." This perception permeates career-counseling programs in the Mexican context, since its interventions start in the educational level just behind the university program [28]. Therefore, students have to decide in a relatively short period of time which career suits them best. Sometimes the students might have a great amount of career information, making it difficult to make a good analysis of each of the options. But it also might occur that there is little availability of information and students might end up making an inadequate career choice.

3 Discussion

Increasingly, researchers are calling for studies of change in motivation, rather than treating motivation as a static trait-like factor [1], [4]. However, those studies mainly consider motivation to be influenced by characteristics from the areas of values, expectancies and feelings [1], without taking into account that some cultural aspects like machismo, social orientation or perception of time might also be influencing how students approach to a learning activity. For instance, women could be avoiding pursuing a career that would not allow them to easily integrate their expected roles as mother and spouse with their future professional activity. Also, the perception of educational barriers, such as gender and ethnicity, nurtured by the social context could reinforce the idea of choosing a career according to the students’ sex, which in turn might influence students’ motivation to learn a particular area of study. Although there is little research evidence that establishes a direct connection between career choice and motivation to learn a particular topic, this paper reviewed some cultural aspects in the Mexican context that have an impact on students’ learning behavior. Based on this, we consider plausible to do some research that consider these aspects when designing a motivationally tutoring system. For example, in a Mexican context, a tutoring system for Mathematics could emphasize women’s capacity to solve problems regardless of their gender, like providing feedback including mentions to important contributions from female scientists, or listing the advantages of achieving personal professional success as a woman, or maybe using a very strong female character showing high IQ as the main avatar.

4 REFERENCES

1. du Boulay, B. Towards a Motivationally-Intelligent Pedagogy: How should an intelligent tutor respond to the unmotivated or the demotivated? In R. A. Calvo & S. D'Mello (Eds.), New Perspectives on Affect and Learning Technologies (pp. 41-54). New York: Springer (2011)
2. Avramides, K. and du Boulay, B. Motivational Diagnosis in ITSs: Collaborative, Reflective Self-Report. In V. Dimitrova, R. Mizoguchi, B. du Boulay & A. Graesser (Eds.), Artificial Intelligence in Education. Building Learning Systems that Care: from Knowledge Representation to Affective Modelling AIED2009 14th International Conference on Artificial Intelligence in Education (Frontiers in Artificial Intelligence and Applications No. 200 pp. 587-589). Amsterdam: IOS Press (2009)
3. du Boulay, B., Rebolledo Mendez, G., Luckin, R. & Martinez Miron, E. (2007). Motivationally Intelligent Systems: Diagnosis and Feedback. In R. Luckin, K. Koedinger & J. Greer (Eds.), Artificial Intelligence in Education: Building Technology Rich Learning Contexts. Proceedings of AIED2007, Los Angeles (Frontiers in Artificial Intelligence and Applications No. 158 pp. 563-565). Amsterdam: IOS (2007)
4. Henderlong, J., and Lepper, M. R. The effects of praise on children's intrinsic motivation: A review and synthesis. *Psychological Bulletin, 128,* 774–795 (2002)
5. Kaplan, A., Karabenick, S., & De Groot, E. Introduction: Culture, self, and motiva- tion: The contribution of Martin L. Maehr to the fields of achievement motivation and educational psychology. In A. Kaplan, S. Karabenick, & E. De Groot (Eds.), *Culture, self, and motivation: Essays in honor of Martin L. Maehr* (pp. vii-xxi). Charlotte, NC: Informa- tion Age Publishing (2009)
6. Trumbull, Elise; Rothstein-Fisch, The Intersection of Culture and Achievement Motivation. Carrie School Community Journal, v21 n2 p25-53 (2011)
7. Terzis V., Moridis C., Economides A.A., Rebolledo-Mendez G. Computer Based Assessment Acceptance: A Cross-Cultural Study in Greece and Mexico. *Journal of Educational Technology and Society. 16(3),* 411-424 (2013)
8. Schmitz, K. and Diefentahler, S. An examination of traditional gender roles among men and women in Mexico and the United States. Retrieved, Vol. 12, p. 2008. (1998)
9. Wigfield, A., and Eccles, J.S. (1992) The development of achievement task values: A theoretical analysis. Developmental Review 12: 265 – 310.
10. Wigfield, A., Eccles, J.S., and Rodriguez, D. (1998) The development of children's motivation in school context. Review of Research in Education 23: 73-118.
11. Bong, M., and Skaalvik, E. M. Academic Self-Concept and Self-Efficacy: How Different Are They Really?. Educational Psychology Review, Vol. 15, No. 1, 1-40 (2003)
12. Chouinard, R. and Roy, N. Changes in high-school students' competence beliefs, utility value and achievement goals in mathematics. British Journal of Educational Psychology, Vol. 78, No. 1, 31-50 (2008)
13. Corpus, J., McClintic-Gilbert, M. S., and Hayenga, A. O. Within-year changes in children's intrinsic and extrinsic motivational orientations: Contextual predictors and academic outcomes. Contemporary Educational Psychology, Vol. 34, No. 2, 154-166 (2009)
14. Otis, N., Grouzet, F. E. and Pelletier, L. G. Latent Motivational Change in an Academic Setting: A 3-Year Longitudinal Study. Journal Of Educational Psychology, Vol. 97, No. 2, 170-183 (2005)
15. Caprara, G. V., Fida, R., Vecchione, M., Del Bove, G., Vecchio, G. M., Babaranelli, C. and Bandura, A. Longitudinal analysis of the role of perceived self-efficacy for self-regulated learning in academic continuance and achievement. Journal of Educational Psychology, Vol. 100, 525-534 (2008)
16. Wang, Q. and Pomerantz, E. The motivational landscape of early adolescence in the United States and China: a longitudinal investigation. Child Development, Vol. 80, No. 4, 1272-1287 (2009)
17. National Career Development Association. http://www.ncda.org/aws/NCDA/pt/sd/news_article/37798/_self/layout_ccmsearch/true
18. Rivera, B. D., and Rogers-Adkinson. D. Culturally sensitive interventions: Social skills training with children and parents from culturally and linguistically diverse backgrounds. Intervention in School and Clinic. 33(2), 75-80 (1997)
19. Lent, R. W., Brown, S. D., and Hackett, G. Toward a Unifying Social Cognitive Theory of Career and Academic Interest, Choice, and Performance, Journal of Vocational Behavior, 45, p. 93 (1994)
20. Hyde, J., Lindberg, S., Linn, M., Ellis, A., & Williams, C. Gender similarities characterize math performance. Science, 321(5888), 494 – 495 (2008)
21. Mendoza, E. Machismo Literature Review. Center for Public Safety Initiatives. Rochester Institute of Technology (2009)

22. Galanti, G. The Hispanic Family and Male-Female Relationships: An overview. Journal of Transcultural Nursing, 14(3), 180-185 (2003)
23. Gallardo-Hernández, G., Ortiz-Hernández, L., Compeán-Dardón, S., Verde-Flota, E., Delgado-Sánchez, G., Tamez-González, S. Intersection between gender and socioeconomic status in medical sciences career choice. Gaceta Médica Mex. 2006 Nov-Dec; 142(6):467-76 (2006)
24. Gorodnichenko, Y. and Roland, G. Culture, Institutions and the Wealth of Nations, CEPR Discussion Paper No 8013 (2010). http://eml.berkeley.edu/~ygorodni/gorrol_culture.pdf
25. Losey, K. M. Mexican American students and classroom interaction: An overview and critique. Review of Educational Research, 65, 283-318 (1995)
26. Ojeda, L. and Flores, L. The Influence of Gender, Generation Level, Parents' Education Level, and Perceived Barriers on the Educational Aspirations of Mexican American High School Students. The Career Development Quarterly 57, 1, 84–95. (2008)
27. Flores, Y.N., Romero A., and Arbona, C. Effects of Cultural Orientation on the Perception of Conflict Between Relationship and Educational Goals for Mexican American College Students. Hispanic Journal of Behavioral Sciences 22, 1: 46-63 (2000)
28. POV (2011). Programa de Orientación Vocacional para el bachillerato general, tecnológico y profesional técnico. www.dgb.sep.gob.mx/04-m2/.../Programa_Orientacion_Vocacional.pdf

Tutorial

on

Using Natural Language Processing Tools in Educational Data Mining

Using Natural Language Processing Tools in Educational Data Mining

Scott Crossley
Georgia State U.
Atlanta, GA 30303
scrossley@gsu.edu

Laura K Allen
Arizona State Univ.
Tempe, AZ, 85287
LauraKAllen@asu.edu

Danielle S. McNamara
Arizona State Univ.
Tempe, AZ, 85287
dsmcnama@asu.edu

ABSTRACT

The workshop will cover the development, use, and educational data mining applications of a number of freely available natural language processing (NLP) tools such as Coh-Metrix, the Writing Assessment Tool (WAT), the Simple NLP (SiNLP) tool, the Tool for the Automatic Analysis of Lexical Sophistication (TAALES), and the Tool for the Automatic Analysis of Cohesion (TAACO). The workshop will provide the participants with an overview of the types of linguistic features that can be measured with these NLP tools. Additionally, it will describe how these features have been (and can be) used in text analyses that are of importance to the educational data mining community. Participants will receive hands-on training with the tools using data from computerized learning environments. Participants will also be shown how the output from these tools can be used to develop machine learning algorithms that can aid in predicting educational outcomes.

Keywords

Natural language processing, Data mining, machine learning

1. Description of the Tutorial Content and Themes

In this workshop, participants will be introduced to a number of freely available natural language processing tools. The primary focus of the tutorial will be to familiarize participants with the linguistic constructs measured by the tools, including lexical sophistication, text cohesion, rhetorical style, syntactic complexity, and text organization. The constructs that will be discussed have all been shown to correlate with student success in various educational settings. The remainder of the tutorial will focus on how these constructs can be applied in educational data mining research to predict educational outcomes. The basic outline for the tutorial is an introduction to linguistic constructs, an overview of available NLP tools, a description of how the linguistic constructs are calculated in the tools, and a discussion of the links between these linguistic constructs and educational outcomes (e.g., performance, attitudes, etc.).

1.1 Justification for Importance of Topic

Natural language processing tools have been widely used to better inform research in a number of fields including cognitive science, medical discourse, literary studies, language learning, and the social sciences. However, large-scale use of NLP tools in educational data mining is much less common. Recently, the strength of NLP has begun to be recognized, as evidenced by a number of funding opportunities, and research findings in fields related to EDM (e.g., MOOCS, ITSs, etc.). This suggests that NLP is poised to become an important element of educational research, particularly when used in combination with more traditional measurements of student success (i.e., psychometric data, system interaction data, and sensor data). The convergence of readily available NLP tools, large-scale educational data sets, and data mining techniques can provide EDM researchers with new approaches to better understand and predict variables related to educational success.

Tutorial

on

Student Modeling Applications, Recent Developments & Toolkits

Student Modeling Applications, Recent Developments & Toolkits [SMART tutorial]

José P. González-Brenes
Pearson
jose.gonzalez-brenes@pearson.com

Michael Yudelson
Carnegie Learning, Inc.
myudelson@carnegielearning.com

Kai-min Chang
Carnegie Mellon University
kkchang@cs.cmu.edu

Yoav Bergner
Educational Testing Services
ybergner@ets.org

Yun Huang
University of Pittsburgh
yuh43@pitt.edu

ABSTRACT

This tutorial is intended for researchers, practitioners and students interested in student modelling. We will cover both the theoretical foundations of student models, and a hands-on approach of how to use existing state-of-the-art toolkits.

LENGTH

Full day

INTENDED OUTCOMES

Participants will be able to understand and apply the selected theory and toolkits for student modeling.

INTENDED AUDIENCE

30 participants with a mix of expertise, and backgrounds (academic, industry).

POTENTIAL FUNDING

Presenters will be either self-funded or funded by their host institution.

DESCRIPTION AND JUSTIFICATION

The educational data mining community is starting to fulfill the promise of using data to improve education. The advancement of the field requires the community to be aware of existing tools and results from student modeling. But with a myriad of student modeling techniques and toolkits available, it is easy to be overwhelmed.

In this tutorial we will cover popular and promising toolkits – and the theory behind them. We will demystify the acronym soup in the educational data mining field (BKT, IRT, 1PL, etc). We will help practitioners and researchers alike to get up to speed on student modeling using the latest technology. We are fortunate enough that the toolkits will be presented by the authors who developed them.

MAIN TOPICS

BNT-SM toolkit. We will cover the foundation of Dynamic Bayes Nets (DBNs), which provide a powerful way to represent and reason about uncertainty in time series data, and are therefore well-suited to model a student's changing knowledge state during skill acquisition. Many general-purpose Bayes net packages have been implemented and distributed; however, constructing DBNs often involves complicated coding effort. To address this issue, we teach how to use a popular extension of the Bayes Net Toolbox (Murphy, 1998) called BNT-SM. BNT-SM is a toolkit designed for the student modeling community. BNT-SM inputs a data set and a compact XML specification of a Bayes net model hypothesized by a researcher to describe causal relationships among student knowledge and observed behavior. BNT-SM generates and executes the code to train and test the model using the Bayes Net Toolbox. BNT-SM allows researchers to easily explore different hypothesis with respect to the knowledge representation in a student model.

FAST (Knowledge Tracing with Features) toolkit. Bayesian Knowledge Tracing (BKT) is an extremely popular method for student modeling. Unfortunately, it
does not allow modeling the feature-rich data that is now possible to collect in modern digital learning environments. Because of this, many ad hoc Knowledge Tracing variants have been proposed to model a specific feature of interest. For example, variants have studied the effect of students' individual characteristics, the effect of help in a tutor, and subskills. These ad hoc models are successful for their own specific purpose, but are specified to only model a single specific feature.
We present the FAST (Feature Aware Student knowledge Tracing) toolkit, an efficient, novel method that allows integrating general features into Knowledge Tracing. FAST is available online at http://ml-smores.github.io/fast/

Bayesian Knowledge Tracing at Scale toolkit. Bayesian Knowledge Tracing (BKT) is an extremely popular method for student modeling. We will present the Toolkit for Bayesian Knowledge Tracing at scale (codenamed ***hmmsclbl***). Originally targeted at being used with KDD Cup 2010 data donated by Carnegie Learning, Inc. (millions of records), it has been successfully tested with datasets in the hundred of million records range. ***Hmmsclbl*** accepts a simple data format file with four columns – success, student, step, skill(s) and can fit and cross-validate models using a number of algorithms. In addition to standard BKT model, ***hmmsclbl*** is capable of fitting individualized models with factors accounting for student variance. Individualization is done without changing the structure of the model by modifying how the objective function is computed. Open version of the toolkit that fits standard BKT models is available for downloading at https://github.com/IEDMS/standard-bkt. The tutorial will cover the public version of the ***hmmsclbl*** (standard BKT), currently private individualized BKT, as well as several experimental settings, including: estimating using differentiated evolution and context-based parameter multiplexing.

CFIRT toolkit: Collaborative Filtering in the Style of Item Response Theory. We will introduce classic psychometric theory approaches called Item Response Theory. We will present a machine learning approach to multidimensional item response theory (MIRT), showing how collaborative filtering leads to a general class of models that includes many MIRT models. We motivate a class of models which contains many well known psychometric IRT models (e.g. the 1PL, 2PL, M2PL) as well as a large number of models not previously named. The focus will be on compensatory models, although this is not an intrinsic limitation of the method. We will explain how the model space can be systematically searched by ordering the candidate models in terms of their parametric complexity.

PANEL

José P. González-Brenes is a Research Scientist at Pearson. José is the first prize winner of an international data mining competition against 350+ teams for predicting time travel in highways. José likes to study applications of machine learning to education, for example, he has published in both machine learning venues (AISTATS, NIPS) and educational data mining venues (EDM). José received a PhD and a masters degree in Language Technologies from Carnegie Mellon University (USA), an IMBA from National Tsing Hua University (Taiwan) and a BSc in Computer Science from Instituto Tecnológico de Costa Rica (Costa Rica).

Michael Yudelson is a Research Scientist at Carnegie Learning, Inc. His research focuses on educational data mining, big data analytics, knowledge transfer, hierarchical linear models, and hidden markov models. Michael got his Ph.D. from the University of Pittsburgh in 2010 under the supervision of Peter L. Brusilovsky. Michael's doctoral research focused on investigating scalability of user-adaptive hypermedia systems. Early results of this work were awarded the best student paper at the User Modeling 2007 conference. Michael was a postdoctoral fellow at the Human-Computer Interaction Institute of Carnegie Mellon University in 2010-2013. There he worked with Kenneth R. Koedinger and Geoffrey J. Gordon on knowledge transfer and big data analytics using Hidden Markov Models.

Kai-min Chang is a Research Associate at the School of Computer Science in Carnegie Mellon University. His main topic of research is to understand the neural underpinning of semantic knowledge, and knowledge representation in the context of an Intelligent Tutoring System. He has disseminated various machine learning and student modeling toolkits (Yuan et al., 2014; Chang et al, 2006) and coorganized various international workshops on machine learning and brain imaging at leading conferences includes NIPS, ITS, and NAACL.

Yoav Bergner is a Research Scientist in the Center for Advanced Psychometrics at Educational Testing Service. Yoav earned a bachelor's degree in physics from Harvard University and a PhD in theoretical physics from MIT before turning to the applied side and becoming a sculptor and furniture maker for five years. He then turned from shaping the material world to shaping young minds as a NYC public school science and math teacher for the next three years. In the classroom, Yoav developed a pragmatic interest in the potential of digital environments for personalized learning and assessment, so he returned to the research fold, first at the Research on Learning, Assessing and Tutoring Effectively (RELATE) Lab at MIT (just in time for the MOOC explosion!) and now at ETS. Yoav's work bridges educational data mining and psychometric approaches to modeling process data, including online courses and simulation-based tasks, with particular interests in multidimensionality issues, model fit, and collaborative learning and assessment.

Yun Huang is a PhD student in the Intelligent Systems Program, University of Pittsburgh. Her research focus is on applying machine learning techniques to understand student learning. Her advisor is Dr. Peter Brusilovsky, and many of her projects also received advice from Dr. Jose P. González-Brenes. She has been working on online Java Programming Tutoring systems, tackling modeling challenges for complex programming skills, parameterized programming exercises, and cross-content learning environment. One of the model that addresses these challenges is Feature-Aware Student Knowledge Tracing, a general model that allows arbitrary features in Knowledge Tracing, which she developed with Dr. González-Brenes. Their work was nominated for Best Paper Award in 2014 Educational Data Mining Conference. Yun is working towards applying probabilistic methods to construct skill and student models that can take insights from both cognitive theories and student data.

PROPOSED TIMELINE

09:00 - 09:45	Introduction to student modeling, bayesian networks, IRT, Knowledge Tracing, evaluation (TBA)
09:45 - 10:00	*coffee break*
10:00 - 11:00	BNT-SM toolkit for student modeling & brain interfaces (Chang & Xu)
11:00 - 12:00	FAST framework & toolkit (González-Brenes & Huang)
12:00 - 02:00	*lunch*
02:00 - 03:00	Knowledge tracing at scale (Yudelson)
03:00 - 03:15	*coffee break*
03:15 - 04:15	Collaborative filtering & Item Response Theory (Bergner)
05:15 - 05:30	Concluding remarks