Notes on analysis of student submissions in programming

1 Related work: By purpose / high level goals

1.1 General

- [IVA+15] – general overview of EDM in programming; stress on replication, verification of previous studies

1.2 Graphical environments, Blockly, novices

- [KP05] – A taxonomy of programming environments and languages for novice programmers
- [BCH+97] Mini-languages (incl. turtle)
- block/text interface:
  - [PB15] – comparison;
  - [WW15] – comparing assesment using block/text; nice overview of block/text issues, references
  - [BBDP15] Pencilcode (going from block to comparison)
- Blocks and Beyond Workshop
- Turtle graphics: [CC00] (turtle in CS1)

1.3 Modeling Students

Modeling knowledge:

- [PSK+12], journal version [BWP+14] – robot Karel problems, similarity of programs, clustering, HMMs
- [WSLP17] – deep knowledge tracing for programming; Hour of code data
- [KN09] – estimating using BKT model
- [HGHB16] – modeling skill combinations; Java data
- [YHVB14] – student modeling based on logistic model (AFM), Java mooc

Classification of students:

- tinkerers and planners? – mentioned in [BWP+14]

1.4 Generating Hint, Feedback

- Barnes/Eagle – "interaction networks"
  - [EHPIB15] – overview of interaction networks, properties
  - [EB14] – logic problems; clustering of states in network
  - [IHB14] BOTS game, codestate/wordstate
- [RK17] (and more similar by Rivers), Python, AST manipulation
- [GRZ14] – using specification language to test execution
- [GRZ16] – clustering solutions, proposing repairs, formal algorithm... probably many other "repairs papers"
- [PHN+15] – feedback via program embeddings; Hour of code problems

1.5 Visualizing solutions, feedback for teachers

- MOOC data analysis/visualization:
  - [HPNG13] – machine learning, final submissions
  - [NPHG14] – extension of the previous one
  - [GSS+15] – programming codes, final submissions, clustering for teachers (with user evaluation)
- [HVB14] – Java, sequences of submission, changes to number of concepts
- [Bli11] – traces in open-ended programming, NetLogo

1.6 Domain modeling, problem similarity, recommmendations

These do not make much use of student submissions (rather instructor provided code).

Brusilovsky group, Java, Java ontology and parser:

- [HB17] – similarity, recommending examples
- [HSB10,HHGB15]– adaptive navigation / sequencing for content (examples, ...)
- [HB13] – JavaParser, automatic extraction of concepts

2 Related work: By techniques

2.1 Granularity of analysis

Analysis of different levels of granularity [VLI14]

- every edit / submission
  - Python: [RK17]
  - hour of code: [WSLP17]
  - logic tutor: [EHPIB15]
  - Java: [HVB14]
  - ? Netlogo: [Bli11]
- final submissions – only correct submissions
  - Python: [GSS+15]?
- final submissions – all
  - machine learning (matlab): [HPNG13]

2.2 Syntactic manipulation of code

The goal is typically to achieve some kind of "canonization" of "same programs".

- [RK17] – Python (rather simple programs), canonization, AST manipulation, heuristical transformations of code
- [GSS+15] – renaming of variables

2.3 Use of semantics of the code

- [RK17] – number of test cases passed (input-output pairs)
- [HVB14] – Java, unit tests
- [GSS+15] – variable sequences (used for canonization)
- [IHB14] – using "wordstates" in robot programming

2.4 Solution state space, trajectories

Different terminology: Solutions space, problem state space, interaction network. Relevant when we have data on steps.

- [EHPIB15,IHB14] (and others by Eagle) – interaction networks, logic tutor, simple programming
- [PSK+12,BWP+14] – programming, clusters of states, finite state machines obtained using HMM
- [WSLP17] – hour of code, trajectories, deep learning
- [HVB14] – trajectories just wrt number of concepts / concept change

2.5 Similarity measures, clustering

How to measure similarity of two programs.

- [PSK+12,BWP+14] – bag of words; API calls; AST change
- [HB17] – concepts (obtained from JavaParser), similarity using bag of words (concepts) with cosine similarity and tree edit distance (over AST)
- [EB14] – clustering based on the solution space (??)
- [PHN+15] – embeddings using deep learning (autoencoder)
- TODO plagiarism research

3 Experiment: Classification of states

3.1 Goal

Classify states (edits / submissions while solving a problem) into few general classes:

- correct solution
- state on a typical path towards correct solution
- small digression – close to a typical path towards correct solution
- blind path, typical mistake, sink state... (differentiate among them??)
- uncommon state

3.2 Application

- student modeling, estimating knowledge
- providing hints, recommendations
- feedback for teachers, problem creators (statistics of state distributions)

3.3 Algorithm outline

1. preprocessing: raw logged data into AST, canonization, ...
2. clustering of very similar states (based on similarity measure such as edit distance)
3. construction of the state space (graph), detection of paths toward correct solution
4. classification of states

3.4 Evaluation

How do we tell whether the classification is a good one?

- robustness – given independent parts of data as a training set, do classifications agree?
- coverage – how many states are classified (something else then "uncommon state")
- usefulness for predictions, e.g., in predicting next problem success
- usefulness for people (subjective evaluation)

References

[BBDP15] David Bau, D Anthony Bau, Mathew Dawson, and C Pickens. Pencil code: block code for a text world. In *Proceedings of the 14th International Conference on Interaction Design and Children*, pages 445–448. ACM, 2015.

[BCH+97] Peter Brusilovsky, Eduardo Calabrese, Jozef Hvorecky, Anatoly Kouchnirenko, and Philip Miller. Mini-languages: a way to learn programming principles. *Education and Information Technologies*, 2(1):65–83, 1997.

[Bli11] Paulo Blikstein. Using learning analytics to assess students' behavior in open-ended programming tasks. In *Proceedings of the 1st international conference on learning analytics and knowledge*, pages 110–116. ACM, 2011.

[BWP+14] Paulo Blikstein, Marcelo Worsley, Chris Piech, Mehran Sahami, Steven Cooper, and Daphne Koller. Programming pluralism: Using learning analytics to detect patterns in the learning of computer programming. *Journal of the Learning Sciences*, 23(4):561–599, 2014.

[CC00] Michael E Caspersen and Henrik Bærbak Christensen. Here, there and everywhere- on the recurring use of turtle graphics in cs 1. In *ACM International Conference Proceeding Series*, volume 8, pages 34–40, 2000.

[EB14] Michael Eagle and Tiffany Barnes. Exploring differences in problem solving with data-driven approach maps. In *Educational Data Mining 2014*, 2014.

[EHPIB15] Michael Eagle, Drew Hicks, Barry Peddycord III, and Tiffany Barnes. Exploring networks of problem-solving interactions. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, pages 21–30. ACM, 2015.

[GRZ14] Sumit Gulwani, Ivan Radiček, and Florian Zuleger. Feedback generation for performance problems in introductory programming assignments. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 41–51. ACM, 2014.

[GRZ16] Sumit Gulwani, Ivan Radiček, and Florian Zuleger. Automated clustering and program repair for introductory programming assignments. *arXiv preprint arXiv:1603.03165*, 2016.

[GSS+15] Elena L Glassman, Jeremy Scott, Rishabh Singh, Philip J Guo, and Robert C Miller. Overcode: Visualizing variation in student solutions to programming problems at scale. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 22(2):7, 2015.

[HB13] Roya Hosseini and Peter Brusilovsky. Javaparser: A fine-grain concept indexing tool for java problems. In *CEUR Workshop Proceedings*, volume 1009, pages 60–63. University of Pittsburgh, 2013.

[HB17] Roya Hosseini and Peter Brusilovsky. A study of concept-based similarity approaches for recommending program examples. *New Review of Hypermedia and Multimedia*, 23(3):161–188, 2017.

[HGHB16] Yun Huang, Julio D Guerra-Hollstein, and Peter Brusilovsky. Modeling skill combination patterns for deeper knowledge tracing. In *UMAP (Extended Proceedings)*, 2016.

[HHGB15] Roya Hosseini, I-Han Hsiao, Julio Guerra, and Peter Brusilovsky. What should i do next? adaptive sequencing in the context of open social student modeling. In *Design for Teaching and Learning in a Networked World*, pages 155–168. Springer, 2015.

[HPNG13] Jonathan Huang, Chris Piech, Andy Nguyen, and Leonidas Guibas. Syntactic and functional variability of a million code submissions in a machine learning mooc. In *AIED 2013 Workshops Proceedings Volume*, page 25, 2013.

[HSB10] I-H Hsiao, Sergey Sosnovsky, and Peter 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.

[HVB14] Roya Hosseini, Arto Vihavainen, and Peter Brusilovsky. Exploring problem solving paths in a java programming course. 2014.

[IHB14] Barry Peddycord Iii, Andrew Hicks, and Tiffany Barnes. Generating hints for programming problems using intermediate output. In *Educational Data Mining 2014*. Citeseer, 2014.

[IVA+15] Petri Ihantola, Arto Vihavainen, Alireza Ahadi, Matthew Butler, Jürgen Börstler, Stephen H Edwards, Essi Isohanni, Ari Korhonen, Andrew Petersen, Kelly Rivers, et al. Educational data mining and learning analytics in programming: Literature review and case studies. In *Proceedings of the 2015 ITiCSE on Working Group Reports*, pages 41–63. ACM, 2015.

[KN09] Jussi Kasurinen and Uolevi Nikula. Estimating programming knowledge with bayesian knowledge tracing. *ACM SIGCSE Bulletin*, 41(3):313–317, 2009.

[KP05] Caitlin Kelleher and Randy Pausch. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)*, 37(2):83–137, 2005.

[NPHG14] Andy Nguyen, Christopher Piech, Jonathan Huang, and Leonidas Guibas. Codewebs: scalable homework search for massive open online programming courses. In *Proceedings of the 23rd international conference on World wide web*, pages 491–502. ACM, 2014.

[PB15] Thomas W Price and Tiffany Barnes. Comparing textual and block interfaces in a novice programming environment. In *Proceedings of the eleventh*

annual International Conference on International Computing Education Research, pages 91–99. ACM, 2015.

[PHN+15] Chris Piech, Jonathan Huang, Andy Nguyen, Mike Phulsuksombati, Mehran Sahami, and Leonidas Guibas. Learning program embeddings to propagate feedback on student code. *arXiv preprint arXiv:1505.05969*, 2015.

[PSK+12] Chris Piech, Mehran Sahami, Daphne Koller, Steve Cooper, and Paulo Blikstein. Modeling how students learn to program. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, pages 153–160. ACM, 2012.

[RK17] Kelly Rivers and Kenneth R Koedinger. Data-driven hint generation in vast solution spaces: a self-improving python programming tutor. *International Journal of Artificial Intelligence in Education*, 27(1):37–64, 2017.

[VLI14] Arto Vihavainen, Matti Luukkainen, and Petri Ihantola. Analysis of source code snapshot granularity levels. In *Proceedings of the 15th Annual Conference on Information technology education*, pages 21–26. ACM, 2014.

[WSLP17] Lisa Wang, Angela Sy, Larry Liu, and Chris Piech. Learning to represent student knowledge on programming exercises using deep learning. In *Proceedings of the 10th International Conference on Educational Data Mining; Wuhan, China*, pages 324–329, 2017.

[WW15] David Weintrop and Uri Wilensky. Using commutative assessments to compare conceptual understanding in blocks-based and text-based programs. In *11th Annual ACM Conference on International Computing Education Research, ICER 2015*. Association for Computing Machinery, Inc, 2015.

[YHVB14] Michael Yudelson, Roya Hosseini, Arto Vihavainen, and Peter Brusilovsky. Investigating automated student modeling in a java mooc. *Educational Data Mining 2014*, pages 261–264, 2014.