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

Thomas Price  Tiffany Barnes

North Carolina State University

Workshop on Graph-based EDM 2015
Introduction

Data-driven hint generation
The process of extracting contextualized hints from previous students’ solutions to a problem

- Avoids the need for an expensive, hand-authored expert model
- Primary example: the Hint Factory, has been applied in a variety of domains
  - Logical proofs (Stamper et al. 2013)
  - Linked list problems (Fossati, Eugenio, and Ohlsson 2009)
  - A programming game (Hicks, Peddycord III, and Barnes 2014)
- How can we generate hints in complex, open-ended programming problems?
Hint Generation

Our input is log data from previous students solving a problem, which can be represented as an *interaction network* (Eagle and Barnes 2015)

\[
\begin{align*}
3x + 6 &= 9 \\
3x &= 9 - 6 \\
x + 6/3 &= 9/3 \\
x &= 3/3 \\
3x + 6 - 9 &= 0 \\
x + 2 &= 3 \\
x &= 3 - 2 \\
x &= 1
\end{align*}
\]
Some approaches have successfully generated hints for programming problems (e.g. Jin, Barnes, and Stamper 2012; Rivers and Koedinger 2014)

These are most successful on smaller, well-constrained programming problems, with a clear solution

We are interested in the opposite type of problem:

- Large state space
- Multiple, loosely ordered subgoals
- Unstructured output
- A creative, design task

Many novice programming activities have these traits, such as making games and apps
Challenges

How do we represent a student’s state in a programming problem, and when should two states be connected?

- Naive state representation: states correspond to snapshots of students’ code
  - Problem: It is unlikely any two students will have the exact same state
- Simple connection rule: connect states that past students have traversed
  - Problem: May lead to a very sparse network
Current Approaches

This challenge has been addressed previously in three ways:

- **Canonicalization**: Remove semantically unimportant information from student code to increase state overlap (e.g. Lazar and Bratko 2014; Rivers and Koedinger 2012)

- **Connecting States**: Connect similar, existing states in the network with synthetic actions (Rivers and Koedinger 2013)
  - Or add whole paths between states, including synthetic states (Rivers and Koedinger 2014)

- **Alternate State Definitions**: Choose a non-code state representation, such as the output of the student’s code (Hicks, Peddycord III, and Barnes 2014)
An Open-Ended Problem

We wanted to investigate the applicability of current approaches to an open-ended programming problem.
We wanted to investigate the applicability of current approaches to an open-ended programming problem.
Data Collection

- Collected data at a middle school STEM outreach program called SPARCS (Cateté, Wassell, and Barnes 2014)
- 17 6th grade students (12 male; 5 female)
- Students had 45 minutes to work on the activity
- Instructor help was provided only upon request
Canonicalization

Students’ programs were represented as Abstract Syntax Trees (ASTs). We analyzed three levels of state canonicalization:

- **Raw**: No canonicalization; states represent exact code
- **Basic**: Removed variable names and literal values
  - Rivers and Koedinger suggest a number of other measures, but these were much less applicable to our data
- **Ordered**: Also recursively sort all child nodes in the AST
  - Effective serves as an upper bound for removing unimportant ordering information in code
# Canonicalization - Results

<table>
<thead>
<tr>
<th></th>
<th>Raw</th>
<th>Basic</th>
<th>Ordered</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total States</td>
<td>2380</td>
<td>1781</td>
<td>1656</td>
</tr>
<tr>
<td>Percent Unique</td>
<td>97.5%</td>
<td>94.8%</td>
<td>92.8%</td>
</tr>
<tr>
<td>Mean Non-Unique Freq.</td>
<td>3.44</td>
<td>3.95</td>
<td>2.82</td>
</tr>
<tr>
<td>Median Non-Unique Freq.</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Mean % Path Unique</td>
<td>89.9%</td>
<td>83.0%</td>
<td>78.9%</td>
</tr>
<tr>
<td>Standard Deviation</td>
<td>(6.67)</td>
<td>(10.5)</td>
<td>(13.3)</td>
</tr>
</tbody>
</table>

For comparison: In (Rivers and Koedinger 2012), 300 out of 500 final solutions to a basic problem were identical after canonicalization.
Visualizing Distances

- Constructed and plotted distance matrices
- Used Tree Edit Distance as a distance metric
- Lighter shades represent smaller distances
- Min-distance “path” through the matrix shown in green/yellow
- Red crosses indicate where subgoals were completed
Quantifying Distances

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Max</th>
<th>Farthest</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.25 (0.27)</td>
<td>0.76 (0.56)</td>
<td>2.23 (0.75)</td>
</tr>
<tr>
<td>2</td>
<td>4.88 (3.93)</td>
<td>9.18 (5.74)</td>
<td>12.73 (6.10)</td>
</tr>
<tr>
<td>4</td>
<td>4.92 (2.77)</td>
<td>10.11 (3.69)</td>
<td>14.67 (4.77)</td>
</tr>
<tr>
<td>5</td>
<td>7.79 (1.32)</td>
<td>13.17 (1.72)</td>
<td>18.17 (1.72)</td>
</tr>
<tr>
<td>6</td>
<td>7.49 (1.11)</td>
<td>13.17 (0.98)</td>
<td>18.67 (1.75)</td>
</tr>
</tbody>
</table>

How close are the closest *students*?

- Defined the distance between two students as the mean or max distance they get from one another when solving an objective.
- For each student, for each objective solved, find the paired student with minimum distance.
Conclusions

Canonicalization

- The strongest canonicalization reduced the number of states by 30.4%
- Over 90% of states in the network were only reached by one student
- Important but insufficient by itself

Connecting States

- Students maintain proximity when pursuing the same objective, on average within 8 tree edits, but slowly diverge
- This may not be close enough to connect states
- How do we take advantage of similar but distinct states?
Limitations

- This was intentionally exploratory, using only 17 students
- Many data-driven techniques use hundreds, though the Hint Factory has been historically successful with much less data
- The open-ended programming assignment was very complex compared to those used in previous work
- It is difficult to say where these results should be generalized
Future Work

- Can we break problems down into sub-problems, where more overlap is likely?
- Are there more appropriate distance metrics we should be using?
- How can we use output-based state representations to apps or games with non-deterministic results?
Thank You!

Questions? Comments?


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


