Giving Hints is Complicated: Understanding the challenges of an automated hint system based on frequent wrong answers

By Kristin Stephens-Martinez, Krishna Parashar, Regina Ongwarsito, Nikunj Jain, Kavi Gupta, and Armando Fox

Abstract
Formative feedback is important for learning. Code-tracing is a vital skill in computer science learning. We set out to deliver formative feedback to students on code-tracing, constructed-response assessments by building a student error model using insights gained from inspecting the assessment's frequent wrong answers. Moreover, we compared two different kinds of hints: reteaching and knowledge integration. We found wrong answer co-occurrence provides useful information for our model. However, we were unable to find evidence in our intervention experiment that our hints improved student outcomes on post-test questions.

Therefore, we performed a retrospective, exploratory analysis to understand potential reasons why our results are null. We found our current experimental setup (1) allows students to brute-force guess the answer, (2) has Human-Computer Interaction (HCI) problems, and (3) lets the model deliver too many hints. Moreover, we considered the percent of times where there were zero, one, or more wrong answers after a hint or wrong answer from a set. With these statistics, we found areas for further investigation, such as students to brute-force guess the answer, (2) has Human-Computer Interaction (HCI) problems, and (3) lets the model deliver too many hints. Moreover, we considered the percent of times where there were zero, one, or more wrong answers after a hint or wrong answer from a set. With these statistics, we found areas for further investigation, such as

Context and Data
Terms
- Machine-marked-wrong answer (MMWA) - A (question, string) pair the automated system marks as incorrect
- Response - A student, MMWA pair
- Wrong answer - A MMWA that is confirmed incorrect, as opposed to a false positive marked incorrect by the system
- Tag - A human expert's interpretation of a specific student difficulty that could lead to an observed student error
- (Un)inspected MMWA - Whether a MMWA has gone through the tagging process such that we have decided zero or more tags should be assigned the the MMWA

TAGS Data
This is prior work data that includes the entire corpus of inspected and uninspected MMWAs from 9 question sets. The inspected MMWAs are divided into two sets.
- FrequentSet - Each question sets' most frequent MMWAs that cover a total of 90% of the responses and all MMWAs that individually cover 0.4% of the responses.
- StudentSet - All MMWAs from a random 50 student subsample per question set.

All data is collected from the terminal tool OK [1]. OK administers answer-until-correct, code-tracing questions (such as, What would Python display?). Along with every response we record a student identifier and timestamp.

Performance Metric
Q - Set of questions
C = (0, 1) - Whether the student correctly answered question i on the first try
D = 1 minus the fraction of students that correctly answered question i on the first try

Modified Area Under the Curve (AUC)
We have high precision and recall due to our model definition. Therefore, we defined our AUC by scaling the precision vs. recall plot into a 1x1 square and taking that area under the curve.

Student Error Model

Rules
- Primary rule
  - Two wrong answers that share a tag
  - Wrong Answer A: Tag
  - Wrong Answer B: Tag

Propagation rule
- There is only one wrong answer for the tag and an uninspected MMWA whose co-occurrence metric is lower than a threshold

Co-occurrence
To understand the model's performance we compared it to an unaltered baseline that propagated based on a probability rather than co-occurrence. For each model, we used the FrequentSet for training and the StudentSet for testing.

Two Kinds of Hints:
- Reteaching
  - The hint is "Remember the difference between..."
  - "%0" column
- Knowledge Integration (KI)
  - "%1" column
  - Reteaching range function
  - KI range function
  - Reteaching Assignment
  - KI stacked calls cause errors

Future work/Open Questions
1. Given our current data:
   a. What else can we learn from it?
   b. How can we investigate which hints were more effective?
2. What other performance metrics should we consider?
3. Rerun the experiment, but first:
   a. How do we addressing the undesirable behavior?
   b. How do we validating the questions?
   c. How do we improve the hints?
   d. How do we improve the model?
   e. Are there better ways to run the experiment?