Mid-term Report, Intelligent Hinting 1.1

Yi Ding, Weimeng Luo, Junyu Zhang

March 23, 2018

1 Architecture

The architecture is shown in the diagram: github link
Green part in the diagram needs to be modified in the next step.

2 Implementation

Mock Data

We choose one project: completeSE/Face Morphing.se to generate the mock data.

1. We use read_se.py to generate some fake projects, these projects are classified into two types: success, failure. They are located in mockSE.

- Success: We shuffle the order of some random blocks to generate fake successful .se data from completeSE.
- Failure: We delete some blocks to generate fake failure se data from completeSE.

2. From mockSE files, we use mockdataProducer.py to imitate the operations of four kinds of students: extremeMover(1), mover(2), stopper(3), tinker(4). Each type students have 25 mock data.
3. Each csv file represent the timestamps of a student completing one project.
We also process the raw data into two types of .csv files:

- original: Represent the original snapshots of students.
- differential: The differential between two neighbored snapshots.

HMM to Model Students' Learning Paths

We modeled student's learning path (how students developed the program over time) using a Hidden Markov Model(HMM). We observe the source code of each snapshot as a noisy output of a latent variable–milestone. The HMM model they used that at each move assumes that a student is in the state of some high-level "milestone". We can observe the source code of each snapshot as a noisy output of a latent variable - milestone. For example, milestones could be "student has just started" or "student got frustrated". HMM is parameterized by the probabilities of a student

going from one state to another and the probability that a given snapshot came from a particular milestone. We may cluster the student's path according to these probabilities.

We first compute the set of high-level states using the origin csv files with K-Medioids method. This method will distill K *.se* files as the center for each cluster. These centers are the milestones of the project, and we utilized the clustered result as the prior means and prior probabilities of the Gaussian Mixture Model. And then use the EM algorithm to compute the transition and emission probabilities. Currently, we successfully get a HMM for each student. We could get the clusters using the K-Means algorithm over the space of HMM. This will help us to generate accurate hints according to the type of the student's learning path.

Hint Generation and HMM Evaluation

We change the rule for hint generation in order to quantify the performance of the cluster method. We choose poisson path–the path from a node s to a terminal which has the least expected time required, to generate the hints. The equation to assign weights for each each is shown as below:

$$\gamma(s) = \arg \min_{p \in Z(s)} \sum \frac{1}{\lambda_x}$$

We implement Dijkstra's algorithm to get the shortest path from student's current snapshot to the destination, assuming that the amount of time required for an average student to generate a correct solution is a Poisson process according to how many students have even come up with this solution.

3 Limitations and Assumptions

For HMM evaluation, we use the distance between the hints and the destination built by the graph as the metric. For example, after cluster the student into a specific HMM group, if the hint shows to be closer to the final state compared with some other cluster method, then HMM is helpful for the student. This method is limited since we don't have the data of student's interaction, for example, whether s/he accept this hint. We leave this problem into the evaluation of the hinting system part.

4 Future Work

Next step is to implement the evaluation of the hinting system, which is meaningful for teachers to know whether this intelligent system is helpful. We need to connect with the database in the system, and save our hints for further analysis.