Intelligent Hinting 1.1 in SAGE
COMS 6901, Section 28

Yi Ding, Weimeng Luo, Junyu Zhang

February 3, 2018

Contents

1 Abstract

The TFS Epic(s) and Feature(s) to be discussed:
1. HMM evaluation: Evaluate whether a Hidden Markov Model can improve the intelligent hinting system.
2. Alternative hint System opportunities: Alternative hint system implementation options.
3. Hint validation evaluation: Identified a quantitative measure that expresses the accuracy of hints in a way teachers can understand simply. (determined using mock data.)

2 Introduction

This project aims to improve the intelligent hints generation approach by implement different kind of machine learning algorithms, and then build an evaluation system for teachers to see the validation of the hints. Intelligent hinting is an important part for the Social Addictive Gameful Engineering (SAGE) project.[1]

The previous hinting generation method in SAGE gives users hints according to their own characters, the current state and the sequence frequency in hinting pool to generate hints.[2] The limitation is that time-series information is lost during feature extraction. Considering such limitations, the motivation for the project is first implement Hidden-Markov Model for time series data to generate hints.

Another important thing is how to define the "validity" of the hints generated from different approaches.[6] This is useful for teacher to know whether your intelligent system works on s/he students, and which hinting method is more helpful to students. After implement HMM, the evaluation system would test the performance of the hints generated in different ways, and then decide which kind of hinting methods should be selected.

In the second chapter, we introduce the related work of this project, the first part is the summary of the previous hinting generation approach. This could help us compare different methods and design a reasonable hinting evaluation system; the second part focuses on Hidden-Markov Model(HMM) in order to model students' learning behavior.[4]; The third part is about alternative hinting methods; And the last part shows the idea of the validation for different hinting methods. In the third and forth part, we make the plan and time line for this project.

3 Related Work

3.1 Existing Hinting System in SAGE

Current Hinting System in SAGE already embeds several machine learning methods to extract project features and also models student behaviors according to their history data. [1] It first uses K-medoids algorithm to cluster the user snapshots for a specific project. The cluster's center is called project "milestones". Then users behavior is classified into different types according to their paths compared with the milestones. For each type user, the hinting system implements sequential pattern mining to select the next step with the highest possibility.

The processing method is shown as below:
1. Feature extraction: The system first parses the .se file to transform the snapshots to a set of trigrams (consists of 3 blocks), then extracts and clusters the features according to these trigrams, finally matches k_1 centroids to k_1 .se snapshots.

2. Behavior classification: After feature extraction, the milestones of each project is defined. For each student that participates in this project, n snapshots with equal time interval are used as the input. For each snapshot, it implements Needleman-Wunsch algorithm to calculate the similarity between it and the k_1 milestones. Several machine learning methods have been utilized in the system to classify user behaviors.

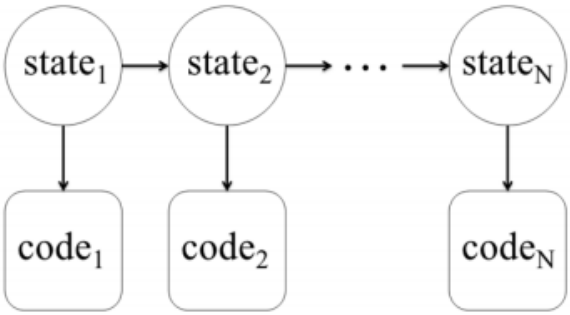
3. Hinting generation: In order to use sequential pattern mining algorithm, it first transforms the snapshots into binomial feature vectors. The training data is collected from students who successfully finish the project (do this processing for each type of students after 2.). Hints are generated according to Generalized Sequential Pattern (GSP). The hinting generation frequency is based on what kind of the user behavior is.

3.2 HMM for hints generation

[Modeling How Students Learn to Program] This paper[3] mainly proposed that (1) how to build models of student's progress pathways using machine learning methods (2) how to autonomously extract features of a student's progress (3) how to predict student's future performance by their grades.

Student's coding progress(path) is modeled using a Hidden Markov Model(HMM). 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 relevant to our problem because student's coding progress is generally incremental and the hidden state(milestone) is more likely to be independent to one's previous time step.

HMM makes Markov assumption that the future state is independent to the past state given the current state of a student. Markov assumption is not entirely correct, but this assumption helps to simplify student's progress, find patterns and make predictions of future state.



To model with HMM, we usually identify three variables: hidden states(milestone), transition probability $P(State_{t+1}|State_t)$ and emission probability $P(Code_t|State_t)$. For emission probability distribution, we first need to calculate dissimilarity of two pieces of code. We have several approaches to measure dissimilarity: BoW(Bag-of-Words) difference, API Call(treated like DNA sequence) difference and AST(Abstract Syntax Tree) change severity. Given dissimilarity of a student's snapshot and its milestone representative snapshot, we can calculate the probabilty that this student snapshot generated from milestone snapshot using a Gaussian distribution based on their dissimilarity.

To learn transition probability and emission probabilty, a popular inference algorithm used in this paper for building HMM is Expectation Maximization(EM) algorithm. The learning process is divided into two phases, E-step and M-step.

To compute the set of high-level states(the number of milestones), this paper uses K-Medioids, a variation of K-Means clustering where centroids are represented by median instead of nu-

merical means, which is less meaningful for a real-valued code section. K-Medioids calculates representitive snapshots for HMM model.

3.3 Alternative hinting method

This paper focus on how to autonomously generating hints. This process can be decoupled into two tasks: 1. Deciding for any learner what partial solution an educational expert would suggest they transition to next. 2. Choosing how to communicate that information to the student such that they improve on a learning objective.

This paper uses Code.org, which is the largest MOOC to date, as a case study. They define the problem as follows: At each point in time, the student's current work can be represented by a partial solution $\phi \in S$, where S is the set of all possible responses. Students will traverse through a series $T = \phi_0, \phi_1, \ldots, \phi_n$ of partial solutions from step 0. They try to solve the first problem by what they call a Problem Solving Policy (PSP), which is a decision for any partial solution as to what next partial solution a student should take (from expert view). They develop a family of algorithms to predict the way how an expert would encourage a student to make forward progress.

They tested three classes of solutions:
Desirable Path algorithms, previously proposed algorithms and vanilla baseline algorithms.
Desirable Path Algorithms: Poisson Path, Independent Probable Path
Baseline Algorithms: Markov Decision Problem, Rivers Policy, Static Analysis, Most Likely Next, Most Common Path, Expected Success, Min Time, Ability Model

The results shows that Poisson Path policy had the highest accuracy for both two problems.

	P_A Accuracy	P_B Accuracy
Algorithm		
Random	8.2%	4.3%
Shortest Path	49.5%	33.6%
Min Time	67.2%	42.2%
Rivers Policy†	72.9%	78.2%
Expected Success	77.9%	56.2%
MDP†	80.5%	47.6%
Most Common Next	81.1%	49.0%
Static Analysis†	86.3%	-
Most Popular Path	88.3%	52.8%
Ability Model	88.4%	63.3%
Independent Probable Path ‡	**95.5%**	**83.3%**
Poisson Path ‡	**95.9%**	**84.6%**
Variation		
Unconditioned on Success	72.2%	68.3%
No Interpolation	86.8%	70.5%
Allow Cycles	94.3%	82.0%
Poisson Interpolation	94.6 %	83.2%

3.4 Validation

To evaluate a PSP π we calculate the percent of overlap between the policy and the expert map, weighted by number of students who submitted partial solutions. Let λk be the number of students that submitted partial solution k. Accuracy can be computed as follows:

$$acc_\pi = \frac{\sum_{k \in T} \lambda_k \delta(k)}{\sum_{k \in T} \lambda_k}$$

$$\delta(k) = \begin{cases} 1, & \text{if } \pi(k) \in t(k) \\ 0, & \text{otherwise} \end{cases}$$

4 Proposal

4.1 Motivation

We plan to improve the intelligent hinting in three parts:
1. User behavior Classification
We plan to use Hidden-Markov Model in order to take the time series information when doing the milestone extraction, which is different from previous work that collected all raw data in a set.
2. Hints Generation Approach
Previous work uses sequence pattern mining algorithm to select the pattern with highest frequency. Here we still use Hidden-Markov Model to predict the next hidden state, and then transform it to the hint.
3. Evaluation of the Hinting System
Since we have different methods to generate the hint, one important thing is to evaluate the hinting system. This is meaningful for teachers to know whether this intelligent system is helpful.

4.2 HMM Evaluation

Hidden Markov Model could be used to:
(1) find hidden pattern of student's development progress so we could cluster students into different groups to decide whether hints need to be provided accurately
(2) populate predicted code blocks from its hidden state's Gaussian distribution to generate hints.

Previous work on SAGE uses clustering and bag-of-words edit distance to group students. However, it failed to model the incremental progress of individuals made during learning process. In this project, we plan to construct a HMM for each student with prior probabilities based on HMM constructed from data per project. Thus, dissimilarity between two student's development snapshots could be measure using cross-model symetric probabilities calculated from two HMM models. We could cluster those students and find hidden patterns from student's development progress.

In this project, we will evaluate whether adding HMM is beneficial to SAGE automatic hinting system by two aspects. First, we want to know that whether student groupings using HMM is better than previous approach, which could be done by comparing grouping qualities, e.g., through statistical testing methods so we are able to decide whether or not to

add HMM with strong evidence. Second, we want to quantify the prediction quality of hints HMM given using methods in our hint validity evaluation section.

4.3 Alternative Hint System Opportunities

Previous works on SAGE utilize Generalized Sequential Pattern (GSP) algorithm in hint generation, which simply extract the most frequently occurring two-block sequences that could be used as rules for hint generation.

As stated in Section 2.3, there are many alternative algorithm hinting methods, among which the best one is Poisson Path.
Poisson Path: The Poisson Path is the path from a node s to a terminal which has the least expected time required.

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

Besides Poisson Path, there are also many methods available. We can conduct comparative experiments to identify which method is preferable for different experiments and different students.

In order to achieve a higher accuracy, we can combine the Poisson Path method with the previous HMM method mentioned, and conduct comparative experiments with students.

4.4 Hint Validity Evaluation

We can use the validation method mentioned in Section 2.4, and we can also implement a new quantitative measure depending on our experiment.

5 Timeline

Milestone	Estimated completion	Report
Environment setup and preparation	February 9	team
Extract feature to fit our new model	February 16	Weimeng, Luo
Implement HMM to the hinting system	February 23	Junyu Zhang
Implement evaluation method	March 9	Yi Ding
Midterm report	March 23	team
Embed the validation algorithm in the system	March 30	Weimeng Luo
Implement alternative methods	March 30	team
Generate the hinting validation result	April 13	Yi Ding
HMM performance evaluation and improvement	April 13	Junyu Zhang
Final report	April 27	team

6 Future Work

1. Integrate the elements of the intelligent hinting system via the affinity space to make it convenient to modify RapidMiner and various scripts consistently when conducting field studies.

2. Implement the controls for the hinting system then researcher could change the version of sage-scratch in the Student view of the affinity space.

7 References

[1] Bender, J. (2015). *"Tooling Scratch: Designing a Collaborative Game-Based Learning System to Infuse Computational Thinking within Grade 6-8 Curricula."*

[2] R.C. Murray, K. Vanlehn, and J. Mostow, *"Looking Ahead to Select Tutorial Actions: A Decision-Theoretic Approach,"* in International Journal of Artificial Intelligence in Education, 14. 2004, pp. 235-278.

[3] Piech, Chris and Sahami, Mehran and Koller, Daphne and Cooper, Steve and Blikstein, Paulo, *"Modeling How Students Learn to Program"*, SIGCSE '12.

[4] Benjamin Paaßen and Barbara Hammer and Thomas William Price and Tiffany Barnes and Sebastian Gross and Niels Pinkwart, *" The Continuous Hint Factory - Providing Hints in Vast and Sparsely Populated Edit Distance Spaces"*, http://arxiv.org/abs/1708.06564

[5] Thomas W. Price, Rui Zhi, and Tiffany Barnes, *" Hint Generation Under Uncertainty: The Effect of Hint Quality on Help-Seeking Behavior"*, AIED '17.

[6] Piech, Chris and Sahami, Mehran and Huang, Jonathan and Guibas, Leonidas,'' *"Autonomously Generating Hints by Inferring Problem Solving Policies"*, Proceedings of the Second (2015) ACM Conference on Learning @ Scale, pp.195-204.