Estimating the Local Size and Coverage of Interaction Network Regions

Michael Eagle
North Carolina State University
Department of Computer Science
Raleigh, NC 27695-8206
mjeagle@ncsu.edu

Tiffany Barnes
North Carolina State University
Department of Computer Science
Raleigh, NC 27695-8206
tmbarnes@ncsu.edu

ABSTRACT

Interactive problem solving environments, such as intelligent tutoring systems and educational video games, produce large amounts of transactional data which make it a challenge for both researchers and educators to understand how students work within the environment. Researchers have modeled the student-tutor interactions using complex network representations to automatically derive next-step hints, derive high-level approaches, and create visualizations of student behavior. However, students do not explore the complete problem space. The nonuniform exploration of the problem results in smaller networks and less next-step hints. In this work we explore the possibility of using frequency estimation to uncover locations in the network with differing amounts of student-saturation. Identification of these regions can be used to locate specific problem approaches and strategies that would be most improved by additional student-data.

Keywords

Interaction Networks, Data-Driven, Problem Solving

1. INTRODUCTION

Data-driven methods to provide automatic hints have the potential to vastly reduce the cost associated with developing tutors with personalized feedback. Modeling the student-tutor interactions as a complex network provides a platform for researchers to generate hint-templates and automatically generate next-step hints; the interaction networks also work as useful visualization of student problem-solving, as well as a structure from which to mine high level approaches of student problem-solving approaches. Data-driven approaches require an uncertain amount of data collection before they can produce feedback, and it is not always clear how much is needed for different environments. Eagle et al. explored the structure of these student interaction networks and argued that networks could be interpreted as an empirical sample of student problem solving [4]. This would mean that students who are similar in problem-solving approaches would also be represented in the same parts of the interaction network. This would suggest that students who are more similar would have smaller networks as they explore the same parts of the problem space. We argue that as the expectation is for different populations of students to have different interaction networks and that different domains will require different amounts of student-data, there need to be good metrics for describing the quality of the networks.

In this work, we will make use of Good-Turing frequency estimation on interaction level data to predict the local size and hint-producing capability of interaction network regions. Our estimator makes use of Good-Turing frequency estimation [5]. Good-Turing frequency estimation estimates the probability of encountering an object of a hitherto unseen type, given the current number and frequency of observed objects. It was originally developed by Alan Turing and his assistant I. J. Good for use in cryptography efforts during World War II. In our context, the object types will refer to network-states (vertices,) and observations will refer to the student interactions (edges.)

Creation of adaptive educational programs is expensive, intelligent tutors require content experts and pedagogical experts to work with tutor developers to identify the skills students are applying and the associated feedback to deliver [7]. In order to address the difficulty in authoring intelligent tutoring content, Barnes and Stamper built an approach called the Hint Factory to use student data to build a Markov Decision Process (MDP) of student problem-solving approaches to serve as a domain model for automatic hint generation [12]. Other approaches to automated generation of feedback have attempted to condense similar solutions in order to address sparse data sets. One such approach converts solutions into a canonical form by strictly ordering the dependencies of statements in a program [9]. Another approach compares *linkage graphs* modelling how a program creates and modifies variables, with nested states created when a loop or branch appears in the code [6]. In the Andes physics tutor, students may ask for hints about how to proceed. Similarly to Hint Factory-based approaches, a solution graph representing possible correct solutions to the problem was used, however it was automatically generated rather than being derived from data, and uses plan recognition to decide which of the problem derivations the student is working towards [13].

Interaction networks are scale-free, in that there is a small

subset of the overall network-states which contain the largest number of neighboring states [4]. Eagle et al. argued that this was in part due to the nature of the problem-solving environment, where by students with similar problem solving ability and preferences would travel into similar parts of the network and problem-features would result in some states being more important to the problem than others [4]. With this interpretation as a basis sub-regions of the network corresponding to high-level approaches to the problem were shown to capture problem-solving differences between two experimental groups [3]. A region of the network representing a minority approach, would result in locations of the network that would not produce adequate hints for students taking that approach.

2. INTERACTION NETWORKS

An *Interaction Network* is a complex network representation of all observed student and tutor interactions for a given problem in a game or tutoring system [4]. To construct an Interaction Network for a problem, we collect the set of all solution attempts for that problem. Each solution attempt is defined by a unique user identifier, as well as an ordered sequence of interactions, where an interaction is defined as {initial state, action, resulting state}, from the start of the problem until the user solves the problem or exits the system. The information contained in a *state* is sufficient to precisely recreate the tutor's interface at each step. Similarly, an *action* is any user interaction which changes the state, and is defined as {action name, pre-conditions, result}. Regions of the network can be discovered by applying network clustering methods, such as those used by Eagle et al. for deriving maps high-level student approaches to problems [3].

Stamper and Barnes' Hint Factory approach generates a next-step Hint Policy by modeling student-tutor interactions as a Markov Decision Process [12]. This has been adapted to work with interaction networks by using a value-iteration algorithm [2] on the states [4]. We define a state, S to be *Hintable* if there exists a path on the network to a goal-state starting from S. We define the *Hintable* network to be the induced subset of the interaction network containing only *Hintable* states.

The "cold start problem" is an issue that arises in all data-driven systems where for early users of the system, predictions made are inaccurate or incomplete [11, 10]. Barnes and Stamper [1] approached the question of how much data is needed to get a certain amount of overlap in student solution attempts by incrementally adding student attempts and measuring the step overlap over a large series of trials. This was done with the goal of producing automatically generated hints, and thus solution-attempts that did not reach the goal were excluded. Peddycord et al. [8] performed a similar technique to evaluate differences in overlap between two different interaction network state representations.

2.1 Good-Turing Network Estimation

In this work, we are presenting a new method for estimating the size of the unobserved portion of a partially constructed Interaction Network. Our estimator makes use of Good-Turing frequency estimation [5]. Good-Turing frequency estimation estimates the probability of encountering an object of a hitherto unseen type, given the current number and frequency of observed objects. It was originally developed by Alan Turing and his assistant I. J. Good for use in cryptography efforts during World War II. Gale and Sampson revisited and simplified the implementation [5]. In its original context, given a sample text from a vocabulary, the Good-Turing Estimator will predict the probability that a new word selected from that vocabulary will be one not previously observed.

The Good-Turing method of estimation uses the frequency of frequencies for the sample text in order to estimate the probability that a new word will be of a given frequency. Based on this distribution, we calculate the probability of observing a new word in the vocabulary based on the observed probability of observing a word with frequency 1. Therefore, the expected probability of the next observation being an unseen word P_0 is estimated by:

$$P_0 = \frac{N_1}{N} \tag{1}$$

Where N_1 is the total number of words occurring with frequency 1, and N is the total number of observations. Since N_1 is the largest and best explored group of words, the so-far observed value of N_1 is a reasonable estimate of P_1. To apply this method to an interaction network, we will estimate the probability of encountering a new state, based on the previously seen state frequencies. P_0 can then be used to smooth the estimation proportions of the other states.

Our version of P_0 is the probability of encountering a new state (a state that currently has a frequency of zero,) on a new interaction. We also interpret this as the proportion of the network missing from the sample. We will refer to an interaction with a unobserved state as having *fallen off* of the interaction network. We will use the complement of P_0 as the estimate of *network coverage*, I_C, the probability that a new interaction will remain on the network: $I_C = 1 - P_0$.

The *state space* of the environment is the set of all possible state configurations. For both the BOTS game and the Deep Thought tutor the potential state space is infinite. For example, in the Deep Thought tutor a student can always use the addition rule to add new propositions to the state. However, as argued in Eagle et. al. [4], the actions that reasonable humans perform is only a small subset of the theoretical state space; the actions can also be different for different populations of humans. We will refer to this subset as the *Reasonable State Space*, with *unreasonable* being loosely defined as actions that we would not expect a human to take. An interaction network is an empirical sample of the problem solving behavior from a particular population, and is a subset of the state space of all possible *reasonable* behaviors. Therefore, our metrics P_0 and I_C are estimates of how well the observed interaction network represents the reasonable state space.

3. DISCUSSION

Figure 1 shows the results of a preliminary analysis on an interaction network based on student-log data from a tutoring environment. For each region we calculated values of network coverage, I_C, and have highlighted regions of the network which have values below 90% coverage. Good-

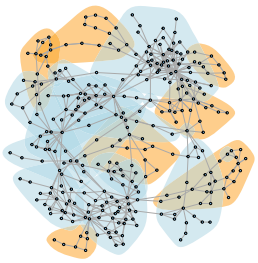

Figure 1: An interaction network with regions of high coverage highlighted in light blue and regions of low coverage highlighted in orange. The low coverage regions of the network require more data before coverage could reach a I_C level above 90%.

Turing Estimation works well in the contexts of interaction networks. Our network coverage metric I_C allows a quick and easy to calculate method of comparing different state representations, as well as quantifying the difference. New methods for improving automatic hint generation can target these areas of the network which have the lowest coverage, such as asking for instructor input on specific regions or by starting advanced students in these regions in order to observe their paths out.

We were also able to interpret this metric as measure of the proportion of the network not yet observed P_0. On a high-level this value alone is a useful metric for the percentage of times a student-interaction is to a not yet observed state. The P_0 score for the hint-able network is likewise a measure for the probability that a student will "fall off" of the network from which we can provide feedback. Therefore, we can use the P_0 metric to predict next-step "fall off" we could estimate the "risk" of different network regions. If we are reasonably sure that the majority of successful paths to the goal have been previously observed then falling off of the network likely means that the student is unlikely to reach the goal.

Region-level coverage also has implications given our previous theories on the network being a sample created from bias (non-random) walks on the problem-space, as the more homogeneous the bias-walkers are, the faster the network will represent the population and smaller total states explored will be. We revisited the results of [3], and have added more description to the effect of hint; students with access to hints explored less overall unique states which implies that the students were more similar to each other in terms of the types of actions and states they visited within the problem.

Future directions for this research include general improvements to the network clustering algorithms which generate the regions. Regions which have low coverage might not be worth separating from their parent region for visualization or high-level hint generation processes. The local and global measures of network coverage can help identify problematic regions in interaction networks which could harm hint production; they also provide a metric to evaluate new, "cold start" problems and make sure that enough data has been collected in order produce hints to multiple problem solving approaches. Finally, exploration of coverage between groups has the potential to uncover differences in problem solving behavior, and improve automatic hinting and understanding of student approaches to problems.

4. REFERENCES

[1] T. Barnes and J. Stamper. Toward automatic hint generation for logic proof tutoring using historical student data. *Proceedings of the 9th International Conference on Intelligent Tutoring Systems (ITS 2008)*, pages 373–382, 2008.

[2] R. Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[3] M. Eagle and T. Barnes. Exploring differences in problem solving with data-driven approach maps. *Proceedings of the Seventh International Conference on Educational Data Mining*, 2014.

[4] M. Eagle, D. Hicks, P. III, and T. Barnes. Exploring networks of problem-solving interactions. *Proceedings of the Fifth International Conference on Learning Analytics and Knowledge (LAK 15)*, 2015.

[5] W. A. Gale and G. Sampson. Good-turing frequency estimation without tears*. *Journal of Quantitative Linguistics*, 2(3):217–237, 1995.

[6] W. Jin, T. Barnes, J. Stamper, M. J. Eagle, M. W. Johnson, and L. Lehmann. Program representation for automatic hint generation for a data-driven novice programming tutor. In *Intelligent Tutoring Systems*, pages 304–309. Springer, 2012.

[7] T. Murray. Authoring intelligent tutoring systems: An analysis of the state of the art. *International Journal of Artificial Intelligence in Education (IJAIED)*, 10:98–129, 1999.

[8] B. Peddycord III, A. Hicks, and T. Barnes. Generating hints for programming problems using intermediate output.

[9] K. Rivers and K. R. Koedinger. Automating hint generation with solution space path construction. In *Intelligent Tutoring Systems*, pages 329–339. Springer, 2014.

[10] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer, 2007.

[11] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260. ACM, 2002.

[12] J. Stamper, M. Eagle, T. Barnes, and M. Croy. Experimental evaluation of automatic hint generation for a logic tutor. *International Journal of Artificial Intelligence in Education (IJAIED)*, 22(1):3–18, 2013.

[13] K. Vanlehn, C. Lynch, K. Schulze, J. A. Shapiro, R. Shelby, L. Taylor, D. Treacy, A. Weinstein, and M. Wintersgill. The andes physics tutoring system: Lessons learned. *International Journal of Artificial Intelligence in Education*, 15(3):147–204, 2005.