Utility in hint generation: Selection of hints from a corpus of student work

John Stamper[1] and Tiffany Barnes
Department of Computer Science, University of North Carolina at Charlotte

Abstract. We have developed a tool for generating hints within computer-aided instructional tools based on a corpus of student work. This tool allows us to select source problem solutions that match the current user solution and generate hints based on next problem steps that are most likely to lead to a successful solution. However, within such a tool it is possible to generate hints that did not turn out to be useful in the source problem solution. Therefore, we have proposed a metric to measure and integrate a "utility" function to choose source material for hint generation. In this paper we present our metric and an experiment to investigate its use on real data from a logic proof tutorial.

Keywords. data mining, machine learning, logic tutor

Introduction

In our previous work, we have created an automated hint generator that uses past student data stored in a Markov Decision Process (MDP) to provide hints in a tutor to teach logic [2]. The most important feature of our MDP method is the ability to assign a value to the states, since this allows the tutor to identify the action that will lead to the next state with the highest value. The value of the state could also be called the expected utility of the state. Our current method of assigning utility is a straightforward implementation of value iteration on a MDP [4]. While this method has been successful in generating valid hints there have been instances where the hint was not directly helpful in solving the problem. In this current research we show our work towards a better utility metric that could improve or replace the MDP value.

The utility metrics that we present here show capable methods for determining the "goodness" of a state by taking into account the features contained in a particular state. In our original MDP method, all paths which solved the problem were directed to a single goal state which was given a high reward value. The use of a single goal state works well in most cases, but we have found two issues where the method could be improved. Occasionally, a student will come up with a unique way to solve a problem, and do so with few errors. Using a straightforward value iteration scheme with high negative rewards (e.g penalties) assigned to errors, steps in these rare but completely correct solutions can have higher values than those that are much more typical or frequent. The reason this occurs is the occurrence of more errors along a more common solution path. In these cases, our automated hint generator will suggest hints derived from the rare but higher-valued solution, which might be harder for some students to

[1] Corresponding Author.

understand. To address this we can place a higher weight on overall frequency relative to penalties for errors. The second issue is that our MDP method currently expects labels for correct problem solutions. Unlike our original method where the goal state was known, the utility metrics we propose here can be applied with our MDP method for hint generation in ill-defined domains where correctness may be difficult to assess.

2. Frequency Weight Metric

The frequency weight metric applies a weighted factor to the value of each state. This weight can be added during or after the value iteration processing of the MDP. The weight is based on the frequency of the individual features of each state. In order to determine the frequency weight we extract a term-document matrix to obtain the frequency of state features. A term-document matrix is used in Latent Semantic Indexing (LSI) research for search large amounts of text [3]. In our work in the domain of solving logic proofs, the term-document matrix is simply large grid containing all the statements (state features) in the proof on one axis and all of the student attempts on the other axis. We then mark which of the statements occur in each individual student attempt and then compute the frequency by summing the columns. The frequency can be used to identify which features are most frequent. For a given state "usefulness" was computed by combining the frequencies of each feature in the state.

When this metric was used for initial state values we found it was too strong and caused the metric to reinforce itself during value iteration making it the sole factor in hints being the frequency and not nearness to the problem solution. To mitigate this issue we next applied the usefulness weights after value iteration. However, this new weight tended to reward high frequency errors. These issues led us to focus on just the potential goal states which became the basis of the terminal state utility metric.

3. Terminal State Utility Metric

For this metric we start with the state feature frequency determined using the term-document matrix as described in the same way as the frequency weight metric. We then set a percentage frequency threshold such that all state features above the threshold had a good potential of being a part of the solution. Setting this threshold can be done automatically or with the help of a domain expert. In the problem we studied, a graph of the feature frequencies showed the possible threshold points and a domain expert picked the one that best represented the break between high and low value features. Once a list of frequent statements is determined, we calculate initial utility values for all terminal states (leaves) in the MDP. This replaces our original approach of creating a goal state with a single positive value. Valid terminal states are therefore candidate goal states. The utility value of a terminal state is the sum of the value for each step (or feature) in the student attempt. The value of each step is positive if it was frequent and negative otherwise. Error states receive a high negative start value, and all other states start at zero. After the initial values are set, value iteration is applied until the state values stabilize resulting in a value for every state.

The most important use of the MDP method is to give students hints. Hints are given by providing the student details of the best state reachable from their current state [cite]. To compare the utility metric value to those generated by our original MDP we calculated both values on the same problem 1 dataset that was used in our previous validation study [1]. Both methods create the same 821 states, of which 384 were non-error states. From the non-error states, 180 states had more than one action resulting in new state. These 180 states are the ones that we focused on since these are the only states that could potentially lead to different hints. Comparing the two methods, they agree on the next best state in 163 states out of 180 (90.56%). For the remaining 17 states where the two methods disagreed, experts identified 4 states where the MDP method identified the better choice, 9 states where the utility method identified the better choice, and 4 states that were essentially equivalent.

These results show that the utility method does at least as good a job as the traditional MDP method in determining state values even when it is not known if the student attempt was successful. In all cases, the hints that would be delivered with either method would be helpful and appropriate, although the fact that the utility method focuses more on the frequency may make this a stronger method since it more closely follows the majority of students. In the past we have suggested that a different value function that relies more on frequency could help students solve problems in ways they are more ready.

4. Conclusion and Future Work

Using MDP values generated from past student data to provide context specific hints to students is a useful way to add ITS capabilities automatically. There is however, a possibility of giving less than perfect hints with this method. In this work we are attempting to improve the hints by enhancing the state values using an additional metric. We have shown how metrics based on frequency could help improve our traditional MDP especially in instances where the completeness of the student attempts is not known. In our original work, we emphasized expert-like solutions. This work relies on the frequency of state features exclusively to determine the utility of terminal states. However, other strategies could be used to automatically assign values to states or steps in a problem, particularly general features that are known for solutions in the given domain, such as containment of particular words, phrases, or structures. Since our simulation showed that the utility metric would perform similarly to our original metric, we next plan to verify that the utility metric can deliver valuable hints in a real tutor environment.

References

[1] Barnes, T., Stamper, J. (2008). Toward Automatic Hint Generation for Logic Proof Tutoring Using Historical Student Data. In E. Aimeur, & B. Woolf (Eds.) *Intelligent Tutoring Systems (ITS 2008)*, pp. 373-382. Berlin, Germany: Springer Verlag.

[2] Barnes, T., Stamper, J., Croy, M., Lehman, L. (2008). A Pilot Study on Logic Proof Tutoring Using Hints Generated from Historical Student Data. In R. Baker, T. Barnes, J. Beck (Eds.) *Educational Data Mining (EDM 2008)*, pp. 197-201. Montreal, Canada.

[3] Landauer, T. K., Foltz, P. W., and Laham, D. (1998). Introduction to Latent Semantic Analysis. *Discourse Processes*, 25, 259-284.

[4] Sutton, R. & A. Barto. Reinforcement Learning: An Introduction, 1998, The MIT Press, Cambridge, MA.