Uncovering Latent Knowledge: A Comparison of Two Algorithms

Danny J. Lynch and Colm P. Howlin

Research and Development Laboratory,
CCKF Limited, Greenhills Road, Dublin, Ireland
{danny.lynch,colm.howlin}@cckf-it.com

Abstract. At the beginning of every course, it can be expected that several students have some syllabus knowledge. For efficiency in learning systems, and to combat student frustration and boredom, it is important to quickly uncover this latent knowledge. This enables students to begin new learning immediately. In this paper we compare two algorithms used to achieve this goal, both based on the theory of Knowledge Spaces. Simulated students were created with appropriate answering patterns based on predefined latent knowledge from a subsection of a real course. For each student, both algorithms were applied to compare their efficiency and their accuracy. We examine the trade-off between both sets of outcomes, and conclude with the merits and constraints of each algorithm.

Keywords: Latent Knowledge, Knowledge Spaces, Technology Enhanced Learning, Intelligent Tutoring Systems, Learner Models.

1 Introduction

There are now many intelligent learning environments that guide students through the learning material within a curriculum [1]. When students are first introduced to such environments, it is reasonable to expect they might have some prior knowledge of those materials. We say that each student has *latent knowledge*. A goal of these learning environments is to uncover this knowledge, so that each student may begin learning new material and not be forced to cover knowledge items already known. One naïve approach is to test the student on each item in the curriculum. This is obviously impractical due to time constraints and the cognitive burden on the student. More intelligent methods of selecting knowledge items and interpreting the results are needed. This is akin to Computerized Adaptive Testing where a pool of items or questions is optimally searched. However, in this case, additional structure can be added to the item pool by taking advantage of the prerequisite nature of knowledge. Mastery of certain items is necessary in order to begin others. This forms the set-theoretical basis of Knowledge Spaces introduced by Doigon and Falmagne in 1985 [2,3]. In this framework, a curriculum is treated as a graph structure where each node represents an item or a question, and a directed edge between two nodes represents a prerequisite connection. A *knowledge state* is defined as a

V. Dimitrova et al. (Eds.): UMAP 2014, LNCS 8538, pp. 363–368, 2014.

feasible subset of nodes that a person could be capable of answering. The *knowledge space* is then the collection of all these knowledge states. Once this graph structure is defined for a curriculum, the list of possible knowledge states can be combinatorially calculated. While trivial for small networks, this is a computationally difficult problem for larger networks, as we detail in Section 2.2. The goal of intelligent learning environments is to uncover the *latent knowledge state* of every student. In this paper, we investigate two algorithms that achieve this. One of the fundamental differences between the algorithms is the requirement of pre-computing all knowledge states prior to run-time. In any case, intelligent learning environments should strive to efficiently achieve this goal by using as few knowledge items as possible and in the shortest possible time.

2 Uncovering the Latent State

For the purposes of this paper, we compare two algorithms that can be used to uncover the latent state of a student. To emulate real world behavior we test these algorithms on the knowledge space of a real network used for Grade 6 Probability in the Realize$^{\text{it}}$ learning environment. This particular network is a subset of a much larger network and, as shown in Fig. 1(a), has 23 nodes and 27 prerequisites links. This produces a total of 6932 feasible knowledge states that a student may have in this curriculum network. The distribution of these states by size, which represents the number of items a student could know, is shown in Fig. 1(b). For example, we find there are only three knowledge states with one item while there are 850 states with thirteen items.

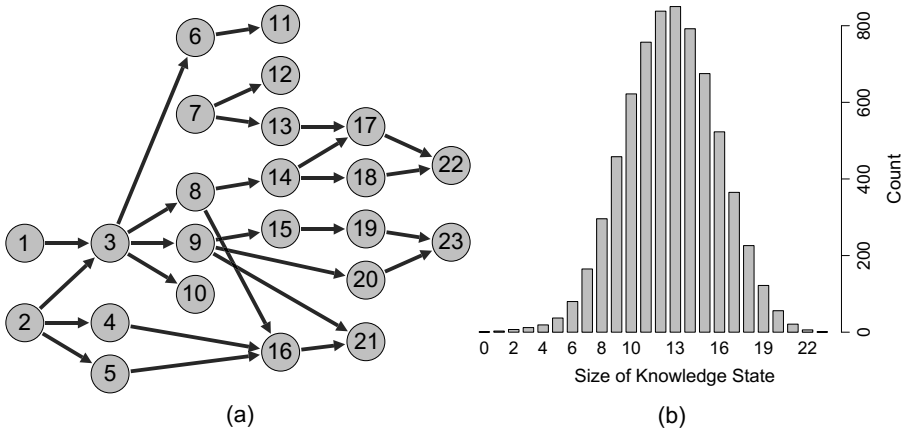

Fig. 1. (a) The 23 node network knowledge domain with a total of 6932 knowledge states and (b) the distribution of the knowledge states according to their size

2.1 Simulation

Simulated students are used to evaluate the algorithms in this paper, which is a common technique found in education research [4]. For each possible knowledge state, a simulated student was created with that state defining their latent knowledge. When probed on items from the domain, these students would respond appropriately based on their latent state. However, to emulate more realistic behavior, a probabilistic response rate was introduced to create cases whereby the simulated student does not respond appropriately.[1] For instance, a response rate of 0.9 signifies the student responds accurately every 9 out of 10 questions. Moreover rather than generating these students' responses on the fly, they were well-defined prior to the algorithm simulation. In this manner, both algorithms obtained the same response when probing the simulated students on the same item, allowing for a fairer comparison of results. The two algorithms were then applied to each simulated student with the objective of uncovering the latent knowledge state. Both of these algorithms select the item which they deem *best* and test the student with it. Based on the student response, the algorithm can then narrow down the feasible knowledge states and select the next *best* item. This process is repeated until a unique state remains or a stopping condition is reached. A number of data points were recorded at the end of each simulation. These include the total number of questions asked (to measure algorithm efficiency) and the count of differences between the defined and the uncovered latent knowledge state (to measure algorithm error rate). Further, due to the non-deterministic nature of these algorithms, each simulation was run five times to produce a mean value for all the collected data points.

2.2 The Algorithms

The first algorithm was developed by Doigon and Falmagne through their work on Knowledge Spaces. Although not formally titled, we refer to this algorithm as *Knowledge Space Theory* (KST) in this paper. The process assigns a probability of being latent to every knowledge state [2,3]. These probabilities are then updated as students are tested on items. This is repeated until a stopping condition is reached. The second algorithm is a component of the Realize$^{\text{it}}$ learning environment developed by CCKF Limited. Referred to as *Determine Knowledge* (DK), this process begins with all items having the potential to be part of the latent state. When students are tested on these items, their response dictates which should be included or excluded. This is repeated until no items are left.

One requirement of the KST algorithm is that all knowledge states are known by the system. This is indeed a trivial process for small networks. For instance, it took under ten seconds to calculate the knowledge states for the 23 node network in Fig. 1(a). To prevent loading times and delays, this computation should not be done on the fly and must instead be completed prior to any student

[1] In the literature this is commonly referred to as the probability of a lucky guess or careless error. For simplicity, these are treated as identical in this paper.

interaction. Further, it would need to be recomputed with every change to the network as this can result in dramatically altered knowledge states. Although this in itself is quite manageable, the required computation time explodes as these networks get larger. In fact, we can show that this computation is equivalent to finding all maximal cliques of an associated network of the same size. This is NP-complete and one of Karp's NP-complete problems [5]. The enumerative algorithms for computing these cliques run in exponential time. A modified version of the Bron-Kerbosch algorithm is documented to be of time complexity of order $O(3^{n/3})$ where n is the size of the network. Indeed, we have found that this computation quickly becomes impractical for larger networks, requiring long computation times and large memory footprints. It was avoidance of these issues that prompted the creation of the DK algorithm.

In comparing the two algorithms, we find that KST lends itself to being flexible whereas DK is more rigid. This is clear from the scenario where a student makes a careless error on an item. This error made during the DK process will eliminate the true latent state, whereas the error in KST just diminishes the probability. On the flip side, this means that the KST process can repeatedly ask the same questions whereas DK only asks each at most once. The impact of this rigidity will be investigated though the simulations and discussed in Section 3.

Algorithm 1: Knowledge Space Theory

There are many variations of the KST algorithm. In particular, there are multiple methods for selecting items (known as Questioning Rules), and multiple methods for updating the probabilities after response (known as Updating Rules). For the purposes of this paper, the following algorithm was used to represent KST.[2]

1. Start with all states having equal probability.
2. Choose the item to be tested using the *Half Split* questioning rule.
3. Update the probabilities based on student response using the *Multiplicative with Parameters* updating rule.
4. Repeat steps 2 and 3 until a unique state has the highest probability, or earlier if other stopping criteria have been met (see [3] 17.2).

Algorithm 2: Determine Knowledge

1. Start with an empty knowledge state and an item pool containing all items.
2. Calculate the information content for each item in the pool.[3]
3. Choose the item to be tested with the maximum information content.
4. Based on the student response, remove appropriate items from the pool and update the knowledge state accordingly.
5. Repeat steps 2 through 4 until the item pool is empty.

[2] Detailed documentation can be found in [3], sections 13.4.4 and 13.4.7, with an overview of the process in section 17.2.

[3] This is based on the structure of the network and defined using Shannon Entropy.

3 Results

For each simulated student and algorithm, the number of questions asked and the calculated accuracy were recorded. These values were classified according to two simulation parameters: the size of the latent knowledge state and the student response rate. For all parameter pairings, the data for each algorithm was averaged and compared to determine if the results were significantly different (at a 95% confidence level). The performance output can be visualized by the two heat maps shown in Fig. 2. In general we find that the DK algorithm asks students fewer questions. The main exception occurs in a region shown at the top of Fig. 2(a). In Fig. 2(b) we find that the KST algorithm is more accurate, but that this accuracy only starts to dominate for knowledge states above size 7. However the most striking feature of this heat map is the yellow bar across the top. This shows that in the absence of student errors, both algorithms achieve the same accuracy for all knowledge states, and in fact determine the correct state. Nevertheless they do use differing numbers of questions to achieve this.

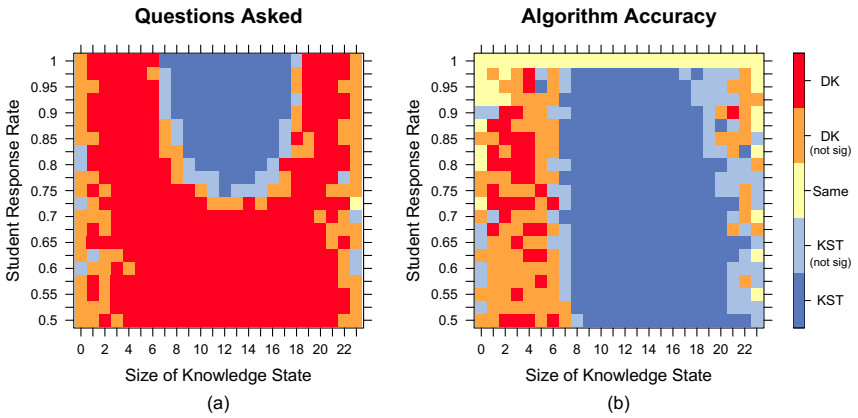

Fig. 2. Heat maps showing which algorithms performed better for (a) questions asked and (b) algorithm accuracy, across all simulation parameters

For a more detailed analysis, the recorded values for each simulation were also subtracted from each other (KST − DK). These were then averaged according to each simulation parameter to produce the four graphs of Fig. 3. In terms of knowledge state size, graph (a) shows that DK tends to ask fewer questions. However graph (b) indicates that KST is often more accurate in these simulations, being on average one item closer to the true latent state for sizes over 7 items. Secondly, in terms of student response rate, graph (c) exhibits a linear relationship where initially DK outperforms with fewer questions but inverts at a response rate of 0.75. Finally graph (d) shows us that KST has greater accuracy across all student response rates, but that the difference diminishes for higher rates. The lack of DK in this graph can be accounted for by the dominance of KST in Figure 2(b) between states of size 8 and 19 in each horizontal slice.

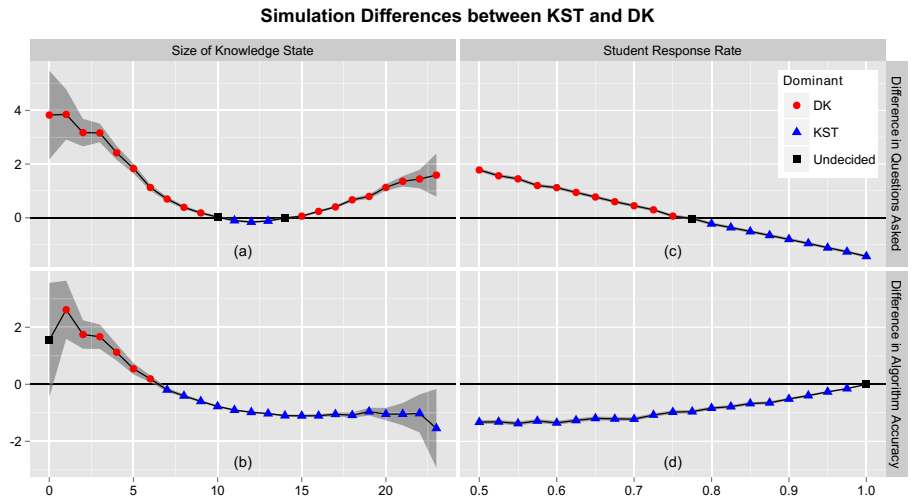

Fig. 3. Summary of simulation differences. The mean value and 95% confidence interval are shown for the *Differences in Questions Asked* and *Difference in Algorithm Accuracy*, outlined according to the size of latent state and student response rate of the simulation.

4 Conclusions

There is a clear trade-off between the number of questions asked and the algorithm accuracy. While DK tends to ask fewer questions, KST makes up for it in accuracy. However, as these differences are often not large, the question remains as to which side of the trade-off is more beneficial. Similar results have been found for simulations run on other networks of various sizes and densities. For those where the knowledge states are computable in a reasonable amount of time, KST makes the overall preferable choice while DK remains a practical alternative. However as the networks evolve and grow, the required computation times become excessive and unmanageable, and DK becomes the better algorithmic choice. Part of our future work aims to introduce more flexibility into the DK algorithm in order to capture student errors.

References

1. Desmarais, M.C., de Baker, R.S.J.: A Review of Recent Advances in Learner and Skill Modeling in Intelligent Learning Environments. UMUAI 22, 9–38 (2012)
2. Falmagne, J.C., Cosyn, E., Doignon, J.P., Thiéry, N.: The Assessment of Knowledge, in Theory and in Practice. In: Missaoui, R., Schmidt, J. (eds.) Formal Concept Analysis. LNCS (LNAI), vol. 3874, pp. 61–79. Springer, Heidelberg (2006)
3. Falmagne, J.C., Doignon, J.P.: Learning Spaces. Springer, Berlin (2011)
4. Abdullah, S.C., Cooley, R.E.: Using Simulated Students to Evaluate an Adaptive Testing System. In: ICCE, pp. 614–618. IEEE Computer Society (2002)
5. Bomze, I.M., Budinich, M., Pardalos, P.M., Pelillo, M.: The Maximum Clique Problem. In: Handbook of Combinatorial Optimization, pp. 1–74. Springer (1999)