Identifying High-Level Student Behavior Using Sequence-based Motif Discovery

David H. Shanabrook[1], David G. Cooper[2], Beverly Park Woolf[2], and Ivon Arroyo[2]
dhshanab@acad.umass.edu
[1]Department of Education, University of Massachusetts, Amherst
[2]Computer Science Department, University of Massachusetts, Amherst

Abstract. We describe a data mining technique for the discovery of student behavior patterns while using a tutoring system. Student actions are logged during tutor sessions. The actions are categorized, binned and symbolized. The resulting symbols are arranged sequentially, and examined by a motif discovery algorithm to detect repetitive patterns, or motifs, that describe frequent tutor events. These motifs are examined and categorized as student behaviors. The categorized motifs can be used in real-time detection of student behaviors in the tutor system.

1 Introduction

Tutoring systems have demonstrated effective learning in the classroom [11]. However, even the most effective tutoring system will fail if the student's behavior is not receptive to the material being presented. For example, lack of motivation has been shown empirically to correlate with a decrease in learning rate [2]. In addition, students often use tutors ineffectively and adopt behavioral strategies that allow them to avoid learning, e.g., deliberately entering incorrect answers to elicit hints and, eventually, the correct answer from the tutor [2]. Although tutor instruction is beneficial, its effectiveness might be increased if maladaptive student behaviors could be identified [4].

Intelligent tutoring systems can identify student behaviors that are likely to be unproductive in the long term. The potential to help students is much greater if their keystroke information is logged to provide teachers with real-time data on students' performance, formative assessments, instead of summative assessments [8]. Recent research has utilized dynamic assessment of students' performance to enhance the effectiveness of their tutor sessions [4].

Previous methods for examining behavioral trends have focused on behaviors that are correlated to specific outcomes. One example is correlating engagement (e.g. fluctuations in attention, willingness to engage in effortful cognition, etc.) with successful and unsuccessful problem solving [3]. A potential drawback of this traditional approach is that behavior patterns during tutor usage can be masked in the overall data.

The work presented here approaches the goal of understanding student use/misuse of tutoring systems in a different way. Instead of correlating to specific outcomes, we data-mine patterns of student behavior that are frequent. Interesting frequent patterns over series of student data emerge. This process is a variation of time-based motif discovery [7]. Continuous variables of student tutor progress are binned into discrete categories represented by "a, b, c …" These letters chronologically form a single string that represents one students actions over hours of tutor use. Concatenating these strings we can create a single string representing of the work of hundreds of students in several schools over several work sessions. Motif discovery is applied and the discovered patterns are examined and categorized qualitatively. The discovered patterns are a potential input for tutor interventions.

2 Relevant Literature

Several models have been proposed to infer high-level student behavior, such as student motivation from behavioral measures. A latent response model [2] was learned to classify student actions as either gaming or not gaming the system. Two cases of gaming were identified: gaming with no impact on pretest-posttest gain and gaming with a negative impact on pretest-posttest gain. The latent response model features consisted of a student's actions in the tutor, such as response time, and probabilistic information regarding a student's latent skills. Beck [4] proposed a function relating response time to the probability of a correct response to model student disengagement in a reading tutor. He adapted the item characteristic curve from Item Response Theory (IRT) to include a student's speed, proficiency, response time, and other problem-specific parameters. The learned model showed that disengagement negatively correlated with performance gain. These models embody different assumptions about the variables required to estimate student motivation (e.g. static versus dynamic models, complex versus simple features, user specified versus learned model parameters, generic versus domain specific models).

A dynamic mixture model was proposed that used a student's behavior to disambiguate between proficiency, modeled as a static, continuous variable and motivation, modeled as a dynamic, discrete variable [6]. These assumptions were based on a student's tendency to exhibit different behavioral patterns over the course of a tutoring session. The model suggested four novel principles: the model should estimate both student motivation and proficiency, run in real time, be able to easily include other forms of unmotivated behavior, and motivation should be treated as a dynamic variable. Empirical evidence suggests that a student's motivation level tends to ebb and flow in spurts [6].

Following the above literature, our method examines student interaction with the tutor during problem solving. However, rather than correlating behaviors with outcomes, we examine frequent behaviors to find meaning in the behaviors on their own. This involves four steps: 1) The raw tutor data is binned and categorized both by hand and statistically. This results in a string of symbols representing the tutor interactions of all students concatenated. 2) The discovery algorithm searches for reoccurring motifs representing student actions over 10 problem intervals. 3) The discovered motifs are consolidated (combined) and categorized by hand, so that they describe high-level student behaviors. 4) These motifs are applied to the original student log and predict student behaviors during tutor usage. In the future, these will be utilized within the tutor to predict real-time student behaviors and correlated to performance outcomes.

3 Tutor description and method

Wayang Outpost is an adaptive tutoring system that helps students learn to solve standardized-test type of questions, in particular state-based exams taken at the end of high school in the USA. This multimedia tutoring system teaches students how to solve geometry, statistics and algebra problems of the type that commonly appear on standardized tests. To answer problems in the Wayang interface, students choose a solution from a list of multiple choice options. Students are provided immediate feedback when they click on an answer (a check for correct or a cross for incorrect). Students may click on a help button for hints, and teachers/researchers encourage them to do so as many times as necessary, as hints are displayed in a progression from general suggestions to bottom-out solution.

Despite efforts to emphasize the importance of going slowly through problems, students tend to “game” the system by using a variety of speeding strategies. Students either click through answers fast in order to get the immediate reward of a correct answers (a green check); or they skip problems without making any attempts; or they click through hints to get to the “bottom-out” hint that reveals the correct choice. Decisions about content sequencing are based on a model of student effort, used to assess the degree of cognitive effort a student invests to develop a problem solution [1].

The data involved in this paper comes from 250 high school students from a variety of math classes in public high schools during Spring 2009. Students came to the computer lab to use Wayang Outpost during that spring semester for about a week, one-hour periods approximately, instead of their regular math class. Students went through various topics such as perimeter problems, area problems, angles, triangles, Pythagorean theorem, etc. The first day and the last day students took a mathematics pretest and posttest. For some students, the topics in Wayang were a review to their math class, to others it was a way to encounter new concepts (there is a short tutorial at the beginning of each topic) and for others it was a way to practice strategies for their upcoming standardized state-wide exams.

4 Data binning and categorization

During interaction with our tutoring system, each student’s actions are logged in a central database. We focus on the data collected during the course of a problem. We utilize four metrics: hints seen (hints), seconds to first attempt (secFirst), seconds between subsequent attempts (secOther), and incorrect attempts (numIncorrect). These metrics are manually binned based on the meaning of the value. For example, there are two indicators for hints seen in our database. There is the count of hints, and there is an indicator that the last hint was seen. From this we have three bins. No hints seen, some hints seen, and last hint solved (because the last hint reveals the answer).

Once the metrics are binned, each problem is represented as a four character *problem string*. For any given student, the tutor interaction can be summarized by the sequence ordered concatenation of problem strings that we will call a *student string*. The problem string construction and meaning is discussed in the following paragraphs.

In motif discovery, data is sometimes binned into discrete categories to significantly reduce the data footprint in memory.   Our reasons for binning are to increase the clarity and meaning of the data; with our understanding of the tutor we categorize each metric so that the bins more clearly describe student behavior. Each metric is categorized in three or five categories represented by a single character from 'a' through 'q', as follows:

> hints (a, b, c) – Hints is a measure of the number of hints viewed for this problem. Although each problem has a maximum number of hints, the hint count does not have an upper bound because students can repeat hints and the count will increase at each repeated view.   The three categories for hints are: (a) no hints, meaning that the

student did not use the hint facility for that problem, (b) meaning the student used the hint facility, but was not given the solution, and (c) last hint solved, meaning that the student was given the solution to the problem by the last hint. As described above, this metric combines two values logged by the tutor: the count of hints seen, and an indicator that the final hint giving the answer was seen. The data could have been simply binned low, medium, high hints; however, this would have missed the significance of zero hints and using hints to reveal the problem solution.

secFirst (d, e, f) – The seconds to first attempt is an important measure as it is during this time that the student is reading the problem and formulating their response. In previous research [6], five seconds was determined to be a threshold for this metric representing gaming: students who make a first attempt in less than five seconds are considered not working on-task. We divide secFirst into three bins: (d) less than 5 sec, (e) 5 to 30 sec, (f) greater than 30 sec. (d) represents students who are gaming the system, (e) represents a moderate time to the first attempt, (f) represents a long time to the first attempt. The cut at 30 seconds was chosen because it equalizes the distribution of bins (e and f), representing a division between a moderate and a long time to the first attempt.

secOther (g, h, i, j, k) – This variable represents actions related to answering the problem after the first attempt was made. While the first attempt includes the problem reading and solution time, subsequent solution attempts could be much quicker and the student could still be making good effort. secOther is categorized in five bins: (g) skip, (h) solved on first, (i) 0 to 1.2 sec, (j) 1.2 to 2.9 sec, (k) greater than 2.9 sec. First, there are two categorical bins, skip and solve on first attempt. These are each determined from an indicator in the log data for that problem. Skipping a problem implies only that students never clicked on a correct answer; they could have worked on the problem and then given up, or immediately skipped to the next problem with only a quick look. Solved on first attempt indicates correctly solving the problem. If neither of the first two bins are indicated in the logs, then the secOther metric measures the mean time for all attempts after the first. The divisions of 1.2 sec and 2.9 sec for the latter three bins were obtained using the mean and one standard deviation above the mean for all tutor usage; (i) less than 1.2 seconds would indicate guessing, (j) would indicate normal attempts, and (k) would indicate a long time between attempts.

numIncorrect – (o, p, q) - Each problem has four or five possible answer choices, that we divide into three groups: (o) zero incorrect attempts, indicates either solved on first attempt, skipped problem, or last hint solves problem (defined by the other metrics); (p) indicates choosing the correct answer in the second or third attempt, and (q) obtaining the answer by default in a four answer problem or possibly guessing when there is five answer problem.

Some of the bins have dependencies that effect motif discovery and analysis. For example, last hint solves (c) precludes solved on first (h) and skipped (g); solved on first (h) requires zero incorrect attempts (o). In addition, by binning skip (g) in the secOther group, the timing of incorrect attempts is lost when the problem is skipped.

479 student strings representing 3762 total problems were constructed and concatenated into a 15048 character input string for motif discovery. The first 160 characters of the string (40 words/problems, separated at problems for clarity) are:

afkq bfho cekp bfho aeho cekq bfiq bfkq bfip aeho aeho aeip
aeho aeip afip cekp aeho afip cfho aeho cfho bfkp bekp bekp
aeho aekp beho bekp aeho cfho bfkq aekp ceho cfkp bfjp aeip
bfkp aeho afip afho

In the first problem, afkq indicates no hints used, greater than 30 seconds to first attempt, over 2.9 mean seconds in other attempts, and most or all choices were made to find the solution. In the next problem, coded bfho, the student asks for one or two hints, greater than 30 seconds to first attempt, then solves the problem on first attempt. The sequence based motif discovery algorithm searches the input string in step two of the process. A detailed description follows.

5 Word-skipping sequence-based motif discovery algorithm

Our sequence-based motif discovery algorithm is a modification of the PROJECTION algorithm [10]. It is similar to the Chiu et al. [5] projection algorithm, except that our sliding window moves per word rather than per character. The PROJECTION algorithm is an efficient way to find planted strings in a long sequence of characters, and our modification to the algorithm allows us to apply it in a multivariate fashion where each character of the word represents a variable. In order to illustrate the algorithm, we first present an example, and then present a formal description of the algorithm.

5.1 Example word skipping projection

Take as input a string words, four characters each, with no separation. Construct a matrix *S* as a 40 character sliding window that slides 4 characters (1 word) per row. The below 6 x 40 character matrix (Figure 1a) represents the first 64 characters or 16 words. Randomly select 10 columns in the *S* matrix as the projection highlighted below. Project the selected columns to a new matrix (Figure 1b). If there is a string match between any pair of rows, then add a collision to the collision matrix (Figure 1c). The highlighted rows (3 and 4) match and thus the 3rd row and the 4th column has a collision.

5.2 Random projection multivariate motif discovery algorithm

Function Name: wordMotif

Inputs: word sequence (*t*), motif size in characters (*n*), word size (*w*), projection length (*p*), number of iterations (*m*), max character distance (*d*), number of motifs to find (*c*).

Outputs: c motif lists with start index and strings of length n for each motif example.

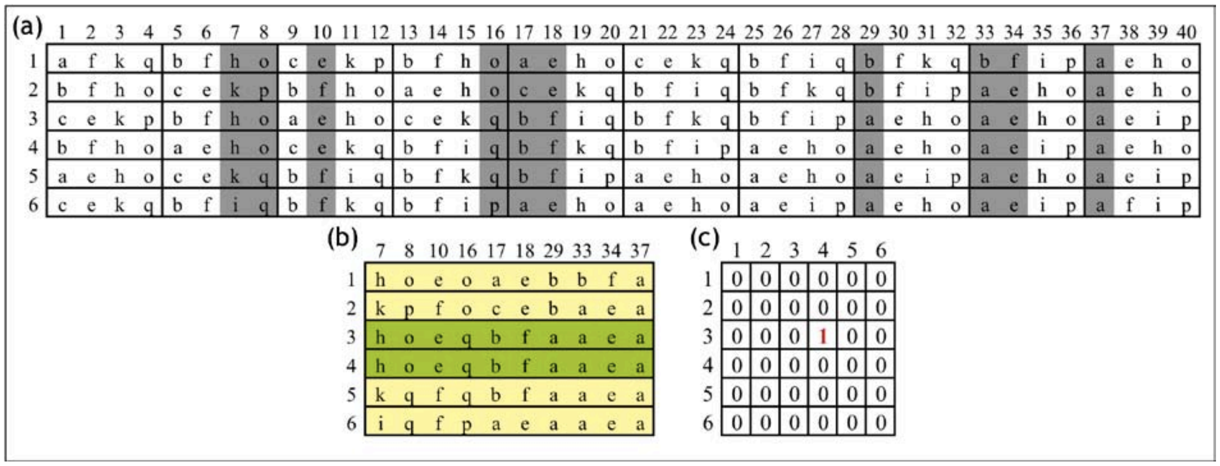
Figure 1. Steps of word skipping projection. (a) the S matrix of 40 character substrings highlighted with a random projection (b) the projected matrix with 2 matching substrings (c) the resulting collision added to the collision matrix.

wordMotif(*t*,*n*,*w*,*p*,*m*,*d*,*c*)

1. Construct S matrix of size (*t*/*w* * *n*) by sliding an *n* sized window *w* characters at a time.
2. Create collision matrix using **project**(*S*,*p*,*m*)
3. Compare pair of examples (*A*,*B*) with highest collision value
4. If (*A*,*B*) do not overlap and are within a hamming distance of *d* characters, then test them against other members of the *S* matrix, adding all members that are within hamming distance *d* to the motif set, and removing the collision values from the collision matrix.
5. Repeat 3 and 4 until *c* motifs are found or the collision matrix is exhausted.
6. Return the lists of up to *c* motifs

project(*S*,*p*,*m*)

1. let *k* be the number of rows in S
2. construct an empty *k* x *k* collision matrix
3. repeat *m* times
    a. make a *k* x *p* matrix based on a random mask of *p* columns of the *S* matrix
    b. add a collision for each pair whose string is equivalent.
4. Return collision matrix

The algorithm outputs a matrix of indices into the string, with a column for each discovered motif. For this study we found thirty motifs. The first two indices of the first 10 motifs are shown in Table 1.

Table 1. The first 2 indices of the first 10 motifs.

M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
2305	6789	8989	8993	9485	2301	18181	29469	29301	48953
6533	7717	9101	9105	11525	18061	19557	49577	58825	67561

Each value in the matrix is an index to the first character of the motif. We can index into the original data to determine the symbols in each of these motifs. The first instance of the motif (in row 1) is representative of the pattern, and the other instances (in row 2, etc.) will be identical or within 10 characters of both the first and the second instance. Table 2 shows the first instance of the first ten motifs.

Table 2. Ten of the motifs The index of the first character is followed by the 40 character motif with a separation each word to see problem characteristics.

M1	2305	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo
M2	6789	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo
M3	8989	adiq	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo
M4	8993	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo
M5	9485	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo
M6	2301	cdgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo	adgo
M7	18181	adgo	adgo	adgo	adgo	adgo	bdgo	adgo	adgo	adgo	adgo
M8	29469	afho	aeho	aeho	aeho	aeho	aeho	aeho	afho	aeho	aeho
M9	29301	aeho	aeho	aeho	aeho	aeho	aeho	aeho	afho	afho	aeho
M10	48953	aeho	afho	afho	afho	aeho	aeho	aeho	aeho	afho	aeho

6 Discovered motifs

In the third step, we examine and analyze the discovered motifs to determine the high level behavior that each motif discovered. We group motifs with similar behaviors. For this study we used the algorithm to detect thirty motifs. Chiu et al. [5] suggest a need to eliminate degenerate motifs, which are motifs that have no informational content, such as a sequence of repeated characters. In the case of the tutor data, repeated words are not degenerate because they inform us that the student is repeating a particular behavior. Repetition of undesired behavior is an important feature to capture, so we do not remove such motifs as degenerate.

In the 30 discovered motifs, there were a number of repeated motifs. This is because a number of motifs were essentially straight, i.e. repetitions of a motif word, so there are cases of motifs that have essentially the same structure; these would overlap with each other, so they are considered distinct patterns. By grouping these exact match motifs, and also by comparing the different motifs by eye, we grouped the 30 motifs into 7 distinct *meaning groups* coded g, f, F, k, r, z, n:

Game-like *(g).* adgo (10), adip (10), or adiq (10) – Student is not reading the problems and either skipping or making quick guesses.

Frustration (guess) *(f).* adiq (1) adgo (9) – Guessing to find solution, then skipping the next 9 problems. This could be an indication of frustration.

Frustration (hints) *(F).* cdgo (1) adgo (9) – Using the hints to find solution, then skipping the next 9 problems. This also could indicate frustration, and was grouped together with the previous motif.

Not challenged *(k).* a[ef]ho (10) – Solving the problem on first attempt, not using hints, not guessing. This student is using the tutor appropriately, but not being challenged.

Too difficult *(r).* ceho (10) - student is taking time to read the problem, then **using hints** to find the answer. These students could be working but the material is too difficult for them to solve the problem themselves.

Skipping *(z).* adgo (5) aeho (5) – Student skips 5 problems, then solves 5 normally on first attempt. Taking time to read all problems, then answering or skipping depending on whether the answer is known.

On-task(n) aeiq aeho aeho aekp aeho aeiq aeho aeip aeho aeip - This is the most complex motif found and seems to indicate "on task learning" tutor usage; the student is always reading the problem and making a good first attempt, with a mixture of solving on first attempt (aeho), solving after some attempts (aekp, aeip) and guessing (aeiq). The student is not using the hints nor skipping.

With these groups coded as a single character, we can look a the progress of one student by converting the string of words to a string of motif groups for quicker interpretation. For example, in the ***too difficult*** *(r)* grouping, these problems show the student is taking 5 to 30 seconds before taking any action. However, if the student initially chose the wrong answer, subsequent attempts were quick guesses or gaming hints. In contrast, in the ***game-like*** *(g)* grouping the student was not reading the problem, simply skipping, or making quick guesses. In the ***frustration (guess)*** *(f)* grouping the student was skipping after an attempt. This could just be an indication of skipping problems, or possibly the error in the first attempt triggering a frustration response to skip the next 9 problems. And the ***on-task(n)*** grouping appears to be a mixture of responses for on task tutor use.

7 Evaluating student interaction

The final step in the process is to apply the 7 meaning groups to individual students. To do this we can convert each student string into a *student motif string.* This is done by scanning the student string and replacing each problem string with a dash (-) until a motif is detected; at this point, the problem string is replaced with the meaning group code associated with the motif. The meaning group code is only placed at the final problem of each motif (the last four characters). The conversion is shown below for two students.

Using the meaning group code, we generated the student number followed by patterns for each student representing their tutor interaction. We examine two students below:

4362,"---------k-kkkkkk---k-------k-kkkkk----------------------kkkk-kkkkkkkk------------kkkkkk-kkkk-k----k---kk---------"

4363,"--------------g------------k---------g----------------------g-------------rrrrrr----rr------g-----"

For student 4362 the ***not challenged*** *(k)* motif indicates solving of problems on the first attempt without using hints. (The (-) patterns are those which did not match any motif.) The behavior is a lagging indicator, representing the current and previous 9 problem.

Student 4363 started by gaming the tutor, ***game-like*** *(g)* either skipping or guessing quickly to find answers. This is followed by ***not challenged*** (k), a string problems solved on first attempt followed by more tutor gaming. The ***too difficult*** *(r)* string of r's are where the student began using the hint facility but in a manner to find answers. After about 20 of these too difficult problems he/she returned to skipping or guessing.

This conversion process illustrates how our discovered motifs can be used in a real-time application. A tutor can detect these patterns and respond based on the meaning group in order to have a more personalized interaction with the student. With student 4362 a tutor could increase problem difficulty. For student 4363, a tutor could intervene at problem 15 where gaming was detected, perhaps by introducing the hint system, or directing the student to a teaching video.

8 Discussion and future work

This paper describes a novel method for determining student behavior without linking the behavior to performance outcomes. We have shown a case study where a number of meaningful behaviors were discovered using a combination of hand chosen features and automatic pattern discovery. The features that have been found can be used to detect student behaviors so that the tutor can react in real time. However, these outcome of our study needs to be verified in future work. A number of future directions are discussed below.

We will validate that the discovered motifs accurately represent student behavior by implementing them in the tutor. Upon motif detection, the corresponding meaning group will be verified by a person in real time. The student or a teacher observer will verify the meaning group by responding to a query during the tutoring session, e.g.. "Are you skipping problems because..." These responses will be compared with the predicted behaviors for validation.

We will study automatic data categorization as modifications to our process. The data binning is the most user intensive part of this process. Finding methods to automate it would allow for broader use of these methods.

We will compare a number of different motif window sizes in order to understand the time scale of problem behavior patterns. The value used in this paper, ten problems, is sufficient to describe behavior, and it yielded a manageable number of informative motifs. However, other window levels may yield motifs of different quality and quantity.

References

[1] Arroyo, I., Mehranian, H., Woolf, B.P.: Effort-based Tutoring: An empirical approach to Intelligent Tutoring. *EDM 2010, this volume.*

[2] Baker, R.S., Corbett, A.T., Koedinger, K.R., Roll, I. Detecting when students game the system, across tutor subjects and classroom cohorts. In Ardissono, L., Brna, P., Mitrovic, A. (Eds.) *UM 2005. LNCS (LNAI)*, 2005. 3538, p. 220–224. Heidelberg: Springer.

[3] Beal, C. R., Cohen, P.R. Temporal Data Mining for Educational Applications. In Ho, T.-B. & Zhou, Z.-H. (Eds.) *Trends in Artificial Intelligence: 10th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2008, LNAI*, 2008. p. 66–77. Springer

[4] Beck, J. Engagement tracing: Using response times to model student disengagement. In Looi, C., McCalla, G., Bredeweg, B., Breuker, J. (Eds.) *Artificial Intelligence in Education: Supporting Learning through Intelligent and Socially Informed Technology*, 2005. p. 88–95. Amsterdam: IOS Press.

[5] Chui B., Keogh E., Lonardi S. Probabilistic discovery of time series motifs. *Proceedings of Knowledge Discovery in Data*, 2003. p. 493–498.

[6] Johns, J., Woolf, B.P. A Dynamic Mixture Model to Detect Student Motivation and Proficiency, *Proceedings of the National Conference on Artificial Intelligence*,2006. p. 163.

[7] Lin, J., Keogh, E., Lonardi, S., Patel, P. Finding motifs in time series. *Proceedings of the 2nd Workshop on Temporal Data Mining*, 2002. p. 53-68.

[8] Stevens, R., Johnson, D., Soller, A. Probabilities and prediction: Modeling the development of scientific problem solving skills. *Life Sciences Education*, 2005, 4(1), p. 42–57.

[9] Stevens, R., Soller, A. ,Cooper, M.,Sprang, M. Modeling the Development of Problem-Solving Skills in Chemistry with a Web-Based Turtor, *7th International Conference on Intelligent Tutoring Systems* , 2004. p. 580-591.

[10] Tompa, M., Buhler, J.. Finding motifs using random projections. *Proceedings of the 5th Int’l Conference on Computational Molecular Biology*. 2001. p 67-74.

[11] VanLehn, K., Lynch, C., Schulze, K., Shapiro, J., Shelby, R., Taylor, L., Treacy, D., Weinstein, A., Wintersgill, M. The Andes Physics Tutoring System: Lessons Learned. *International Journal of Artificial Intelligence in Education* ,2005,15(3), p. 147–204.