A Review of Educational Data Mining Features

An Interactive Qualifying Project submitted to the Faculty of WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the degree of Bachelor of Science

by
Brad Cosma

Date:
January 10, 2021

Report Submitted to:

Neil Heffernan
Worcester Polytechnic Institute

Acknowledgments

I would like to thank NSF (e.g., 2118725, 2118904, 1950683, 1917808, 1931523, 1940236, 1917713, 1903304, 1822830, 1759229, 1724889, 1636782, & 1535428), IES (e.g., R305N210049, R305D210031, R305A170-137, R305A170243, R305A180401, & R305A120125), GAANN (e.g., P200A180088 & P200A150306), EIR (U411B190024), ONR (N00014-18-1-2768) and Schmidt Futures. Additionally I would like to thank Professor Neil Heffernan and Ethan Prihar for advising this IQP.

Abstract

This article reviews publications related to the use of student data as features in educational systems. As education becomes more and more digital, systems are able to collect and analyze different types of student data. These features can be used to improve the teaching materials or student's learning experiences. This review investigates methods of collecting, processing, and using student data. This review covers the knowledge tracing problem and method and its extensions and alternatives, as well as systems measuring students' affect and behavior such as wheel spinning, gaming the system, and stopout. Topics relating to problem features and content are reviewed, as well as students' written responses. This review also covers research about the quality and learning outcome of feedback types as well as demographics and socioeconomic status and their impact on student's learning. In each section, this review describes methods from publications using each type of data or features.

Table of Contents

List of Tables

Introduction

Many learning platforms, both online and offline, collect data on their students. The amount and nature of this data can be varied, but often they consist of numeric or categorical features relating to a student's learning experience. These features can be used to analyze and represent student preferences, traits, learning speed or competency, and features from learning materials among other things. These features can be extracted from a student's interactions with a system, from the correctness and patterns in their responses, or the written content of their feedback or answers to questions. Developing automated methods of gathering and analyzing these features can be useful to improve the quality of education if the systems can be shown to be accurate. Much research has been done on the subject of student data and modeling, and a vast amount of literature has been published.

In this review, we cover topics relating to the extraction and use of student data as features. We review student modeling techniques for predicting short-term performance and the use of student performance in the selection of learning material. Similarly, we investigated the use of the performance and quality of feedback and hint material, including crowdsourced hints. We also explored the use of student affect and behaviors such as stopout (quitting) and gaming (exploiting systems). We discuss how student demographics and socioeconomic status have also been used as features of student modeling. We explore the use of natural language processing techniques to analyze both problem content and student responses. The review is sectioned by the publications' subject, with subsections for commonly studied subsets of each subject.

Methodology

To ensure sufficient breadth, we selected 100 publications covering topics related to student modeling and behaviors. Publications were sourced through Google Scholar searches and from Worcester Polytechnic Institute Professor Neil Heffernan's publications site. Professor Heffernan's publications website was used due to its large number of publications focusing on subjects of interest to our goal and usually featuring implementations or data from the ASSISTments online learning platform. We broadened the scope of our review using Google Scholar. We conducted searches for publications matching a variety of combinations of domain and method keywords relating to various forms of student modeling or behaviors and methods of using data. A sample of keywords used is shown below.

Domain Keywords	Method Keywords
Knowledge Tracing	Deep Learning
Affect Detection	Dimensionality Reduction
Sentiment Analysis	Autoencoder
Difficulty Prediction	Anomaly Detection
Success Prediction	Principal Component Analysis
Mastery-Based Learning	Independent Component Analysis
Personalized Learning	Linear Discriminant Analysis
Gaming Behavior	Clustering
Wheel Spinning	Manifold Learning
Student Modeling	Feature Engineering
Performance Factors Analysis	Text Classification
Knowledge Components	Natural language processing

Table 1: Domain and Method keywords used to find relevant publications. Combinations of keywords were used as query terms in Google Scholar.

Literature Review

Short Term Performance

Knowledge Tracing

One of the most commonly used and studied methods of evaluating student performance is knowledge tracing (KT). First described by Corbett and Anderson in 1994 [4], KT uses a Bayesian network to track the probabilities that a student has mastered a particular skill based on the binary correctness of their answers to a skill's questions. As originally described, KT only allows skills to move from the unlearned state to the learned state. It maintains parameters for the probability that a student may slip and incorrectly apply a learned skill, or guess and correctly apply an unlearned skill. Corbett and Anderson applied the KT model to the ACT Programming Tutor and used it to predict students' performance on test exercises. The model achieved mean absolute errors of 0.09, 0.11, and 0.18 across three experiments [4]. Knowledge tracing is a well-studied and used baseline model for student performance prediction.

While knowledge tracing is already a powerful tool, efforts have been made to improve it. A significant portion of these efforts have focused on adding more context to the model. Pardos and Heffernan [10] in 2010 individualized the prior knowledge parameter of KT, showing that using a student's prior correctness across all skills is a better starting point for this parameter

than using all students' correctness on the skill. The improved model achieved a better mean absolute error than KT on 33 of 42 problem sets using the overall percent correctness heuristic [10]. Pardos and Heffernan [46] integrated item difficulty in 2011, effectively individualizing the guess and slip parameters for each problem. They reported that while accuracy was improved for problems with large numbers of data points, it may be beneficial to not individualize for problems without a large amount of data [46]. Wang and Heffernan [48] utilized the amount of assistance a student used to enhance KT accuracy. They created a table defining parameters based on 3 discrete values of how many attempts the student used, and how many hints the student used. This table dictates that in general, the more hints and attempts used, the lower the probability the student will get subsequent questions correct. This prediction was combined with KT's prediction using linear regression. The combination of the assistance model and KT achieved a lower MAE and RMSE, and higher AUC than just KT [48]. Hawkins et al. [31] calculated empirical probabilities of when a student learned a skill and used the probability of when the skill was used instead of correctness. They used a simple heuristic that calculated the differences between the real data and all possibilities. This achieved an MAE of 0.3742, slightly lower than but very similar to the baseline of 0.3830. Ultimately, the two models performed similarly [31]. In 2012, Wang and Heffernan [41] personalized all the parameters of knowledge tracing, using student and skill level data to influence the values of the KT parameters. This method showed increased accuracy over the base KT model when the number of students and skills was large, achieving an RMSE of 0.4199 compared to KT's 0.4331 [41]. Wang and Heffernan [35] modified KT in 2013, applying a similar idea by integrating continuous partial credit instead of binary correctness. This increased the accuracy of the model, achieving a root-mean-square error (RMSE) of 0.28 when using partial credit compared to KT's 0.41 [35]. Hawkins et al. [33] extended the assistance model, introducing a general framework for the use of any raw data in a KT model. They describe tabling models which can be used for any type of data and then integrated into KT using means or regression as previously described or using random forest or decision tree models. All models achieved better MAE and very similar RMSE and AUC scores compared to KT [33]. In 2014, Khajah et al. [55] integrated KT with a latent-factors model, producing a single LFKT network. This new model achieved the lowest negative log-likelihood when compared to KT with and without latent student ability or problem difficulty [55]. In 2016, Huang et al. [82] explored the use of KT on problems that cover a combination of skills. They propose a model where individual skills are connected to the combination skills they are a part of and have their masteries influenced by all the combinations they appear in. The new model showed comparable accuracy but higher predictive performance when compared to KT [82]. The introduction of other features in knowledge tracing, especially personalization, is of interest in improving student modeling.

It is also possible to incorporate temporal information into knowledge tracing. Wang and Heffernan [43] integrated student first response time into the KT, creating a simple model to predict student performance based on first response time and combining it with KT predictions using a linear regression model. They found that first response time had a small influence on the final result, but that the combined model had an RMSE of 0.4213, slightly lower than KT's 0.4251 [43]. Qiu et al. [44] added the concept of forgetting and rust to KT. They proposed that students may either forget knowledge from the previous day or need some time to remove rust

or warm-up and may slip on their first problem of the day. They used a split model to allow different parameters for forgetting or slipping on questions answered on a new day. Testing showed that the forget model performed better than KT, which in turn performed better than the slip model. The forget model achieved an AUC of 0.7003 as opposed to the 0.6416 and 0.6110 achieved by KT and the slip model, respectively. These results show that modeling students' tendency to forget over time can improve KT substantially [44]. Ghosh et al. [16] created attentive knowledge tracing (AKT), an approach to the knowledge-tracing problem using attention-based neural networks with human-interpretable components. They use autoencoders to create a context-aware encoding of each question and question-answer pair based on a monotonic attention mechanism that takes into account previous answers and the length of time between the present and previous questions. The AKT model outperformed all baseline models [16]. These results show that temporal representations of data in a knowledge tracing context are useful in improving student modeling, and show that time has a large impact on students' applications of skills, perhaps more than some systems account for.

Another useful extension of knowledge tracing is the reduction of complexity, through the implementation of clustering or otherwise. Clustering seeks to maintain or improve the model's predictive accuracy while decreasing the parameter space. Ritter et al. [54] utilized a strict k-means algorithm to cluster skills that have similar representations in the KT model. They assumed that there would be a reasonable number of skill groups that could be given meaningful names, like "hard to learn but easy to guess", and could use the reduced parameter space to more accurately estimate starting parameters for new learning materials. The clustered parameters produced a slightly higher mean squared error at 0.1245 as opposed to the best fit's 0.1204, and the systems behaved similarly. For skills with a small amount of data, it was shown that the clustering model provided a substantially better fit [54]. Pardos et al. [40] performed clustering based on students' interactions with the tutoring system, finding student groups instead of skill groups. They compared the use of a k-means clustering algorithm and a spectral clustering algorithm, training a KT model for each cluster. For each number of clusters, they blended the predictions of each cluster's model. They then averaged predictions over the models produced for K-1 to 1 clusters. The ensemble models achieved lower RMSE overall, with RMSE generally falling as the cluster count increased. However, some datasets showed better results when only using user features rather than including all features [40]. Nooraei et al. [45] explored the idea of reducing KT's complexity by limiting the number of past opportunities used. They found that not limiting the amount of data achieved an RMSE of 0.289123. Limiting the number of problems used as data points to 75 achieved the best RMSE (0.288648), while taking one-fifth the time to train. Limiting the number of data points to 10 resulted in an accuracy loss of 0.6% compared to a model trained on all the data. The paper concluded that the accuracy gained from training on a large number of data points is likely not worth the large increases in training time [45]. Clustering is often a powerful tool, and the benefits of complexity reduction are not wasted on the knowledge tracing problem.

An alternative solution to the knowledge tracing problem is Deep Knowledge Tracing (DKT). Created by Piech et al. [12], DKT uses recurrent neural networks to model student learning. The network takes an input consisting of the correctness of a student's answers to all

problems and outputs the student's probability of correctly answering each question. They tested DKT against KT on data from Khan Academy and ASSISTments and found that while KT achieved an AUC of 0.68 and 0.67, respectively, DKT achieved 0.85 and 0.86. This is a substantial gain [12]. Zhang et al. [53] extended DKT, incorporating problem-level features. They encoded features like correctness, the z-score of the first response time, attempt count, and first action type as sparse one-hot vectors, concatenated them together, and used these as the input for each question. They discuss methods of encoding this information in a lower-dimensional space to reduce computation time, then test their methods on real problem data. On the ASSISTments data, the new model with the autoencoder achieved an AUC of 86.7 compared to base DKT's AUC of 85.8. On the OLI Statics data, the new model achieved 73.5 compared to DKT's 73.1 [53]. Scruggs et al. [67] modified DKT to directly produce estimates of student knowledge instead of problem correctness. They took the predicted values of correctness for all problems in a skill and averaged them. This method was applied to DKT, PFA, KT, and Dynamic Key-Value Memory Networks for Knowledge Tracing, and it produced accurate results [67]. These publications suggest that deep learning is a viable method for solving the knowledge tracing problem. While it differs significantly in nature, the above publications show it is equally useful and extensible.

The knowledge tracing method has also been used for goals other than predicting student performance. For example, Qiu et al. [39] used linear regression to determine which sets of features produced the best models. The features considered were percent correct of a student and skill, time intervals between responses, the number of days spent trying to master a skill, and the number of practice opportunities (problems) completed for the skill. At the opportunity level, the time interval feature achieved the lowest RMSE at 0.3891 compared to KT's 0.3934. At the skill level, it again had the lowest RMSE at 0.3898 while KT achieved 0.3934. Finally, at the student level, the time interval feature again achieved the lowest RMSE with 0.3892 compared to KT's 0.3934 [39]. Pardos and Heffernan [38] upended the KT model, creating a model which determined the probability of learning between two specific questions. They allowed every question to have an individual guess and slip rate to account for problem difficulties. The same method could be applied to control for other problem-level features. They did not conduct experiments to test the method [38]. Thus, the knowledge tracing model has proven it applies to problems outside its intended one.

Alternative Performance Models

Knowledge tracing is not the only method of modeling a student's knowledge. Pavlik Jr et al. [5] introduced Performance Factor Analysis (PFA). PFA is built on the Rasch item response model, modified to take correctness and incorrectness as parameters. It makes use of a standard logistic regression. The use of both correctness and incorrectness allows the model to be sensitive not only to the quantities of each but the ratios of correct to incorrect responses. They compared PFA to KT and showed that PFA was more accurate and overcompensated less in the case of a single slip [5]. Minn et al. [81] proposed a system for the Dynamic Student Classification on Memory Networks (DSCMN). They encoded a vector of a student's prior performance which is updated at every time interval. These vectors were used to cluster students based on learning ability in both training and testing at each time interval using a

k-means clustering algorithm. They include a metric of problem difficulty. In three of four datasets, the DSCMN model outperformed all other tested models when comparing AUC [81]. Abdi et al. [85] built on the knowledge tracing machine (KTM) framework to include data from learner-sourced contributions as features. They used an encoder to allow multiple types of tasks to be included in the input. It encodes the type of task, the resources and concepts used, the task itself, and the outcome. They evaluated their method on data from the RiPPLE educational system and found that on the two datasets, the new model achieved an AUC of 0.723 and 0.778, while PFA achieved only 0.551 and 0.625 [85]. Botelho et al. [8] proposed a method of modeling students' knowledge of skills based on their mastery of prerequisite skills. They calculated each student's speed of mastery in a skill using the number of problems required to learn a skill. To calculate a student's first problem correctness in a skill, they binned the students by mastery speeds of the prerequisite skills and calculated the average first problem correctness for each bin. They called the process Prerequisite Binning (PB). They compared the PB model to KT using data from ASSISTments and found that PB achieved a higher AUC than KT, scoring 0.651, compared to KT's 0.626 [8]. Baker et al. [50] modeled the moment at which concepts are learned using correctness and other student data, exploring which skills are gradually learned or having "eureka" moments. The model uses linear regression to predict the probability that a student learned the skill at a specific step based on metrics of correctness within the skill, time taken, and hint usage. On Cognitive Tutor data, the model achieved a correlation coefficient of 0.446 between the training labels and the model's predictions, while on the ASSISTments data it achieved a correlation coefficient of 0.446. Predictions made by the model can be plotted with problem count as the dependent variable to visualize how a concept is learned. A plot showing a large spike indicated a skill learned by a "eureka" moment, while a steady curve indicated gradual learning [50]. These models achieve the same task as knowledge tracing with significant accuracy but with alternate methods.

An often explored concept in student modeling is partial credit. Van Inwegen et al. [20] utilized partial credit to create a new student modeling method. Using attempt count, hint count, usage of the final "bottom-out" hint, and first action type, students were grouped into 5 distinct "SuperBins". Two-tailed t-tests revealed that there were only five significantly different bins. To predict student success, they used the progression between bins, comparing the impact of looking at the current bin (method 1), previous and current bins (method 2), previous and current with opportunity count (method 3), and the last two and the current bin (method 4). When testing on ASSISTments data, the average of methods 3 and 4 scored an AUC of 0.728 and RMSE of 0.406, compared to KT's 0.710 and 0.413. The model proved to be an improvement over KT [20]. Ostrow et al. [28] proposed a method of tabling a student's correctness and a problem's difficulty in an attempt to predict next-problem correctness on not only a binary basis but also a partial credit one. Partial credit was determined using an algorithm that places the answer in one of five partial credit bins from 0 (completely incorrect) to 1 (completely correct) by considering first action type, attempt use, and hint use. Problem difficulty was calculated by the average correctness rate for that problem. These tables were then combined and probabilities of correctness were trained for each possible outcome. When compared to KT and PFA, the new model performed similarly to KT but was outperformed by PFA. The authors argue that the computational simplicity of the new method and its similar

performance to KT are worth noting [28]. It seems that partial credit does provide valuable insight into student learning and its use as a feature can enhance a model.

Similarity Metrics

Deciding what content best serves a learner or group of learners is an important part of teaching, and the use of a student's performance to recommend content automatically is a useful idea. To this end, Brigui-Chtioui et al. [57] created an agent-based recommendation system to find the best next piece of learning content for a student. The recommender agent calculates the similarities between users based on their previous interactions with the systems and selects a learning resource based on similar users. The filtering agent takes outputs from the recommendation agent and determines which is best for the learner, and the management agent decides if the recommender agent needs to update its rating of a resource based on the results [57]. Zhao et al. [59] utilized KT to provide personalized recommendations for the next learning activity. They used an encoding of learner attributes alongside problem correctness with an attention mechanism to predict the performance of certain learning materials. The model matches the student's attributes and performance on specific materials with the most similar activity sequences from other learners and provides a recommendation for the next activity, taking the most popular next activity from the similar students [59]. Such recommender systems show a promising start to improving the quality of chosen learning materials.

Clustering has been explored as an effective way to identify opportunities for personalization. Kausar et al. [63] presented a recommender system that could be based on clustering using the Clustering by Fast Search and Finding of Density Peaks via Heat Diffusion (CFSFDP-HD) algorithm. They clustered based on attendance and grades in quizzes, assignments, and exams. They describe how the clusters could be used by an instructor to provide certain interventions to groups of the class [63]. Zhou et al. [64] presented a system for generating personalized learning paths using clustering and long short-term memory (LSTM) neural networks. They used features about resource type and student preferences. Data relating to a learner's interactions with the systems were also recorded, namely facial expressions, time spent on resources, the evaluation rank of the resources, what learning impact the resource had, and what impact the overall path had. They use a clustering algorithm to determine what group a student belongs to, predict the learning outcomes of various paths using an LSTM, and select content based on feature similarity between students. On a dataset from Junyi Academy, the clustering + LSTM method achieved better precision than other models [64]. Also seeking to generate learning paths, Kardan et al. [65] proposed a two-stage algorithm called ACO-Map. ACO-Map uses a two-stage approach. In the first stage, students are given a pretest and clustered using a k-means algorithm based on concept knowledge. In the second phase, an ant colony optimization (ACO) method is used to identify optimal learning paths [65]. Another way to cluster is by formalized learning styles. Jyothi et al. [66] clustered students based on the result of the Felder-Silverman learning style model (FSLSM). In their model, an instructor specifies which resources are good for which learning styles, and the instructor creates clusters of students with similar styles by defining characteristics that should be grouped. The system uses proficiency in individual course modules to allow the instructor to develop learning paths by specifying what types of services/activities should be provided. They

implemented the recommender in an e-learning system and found that it “complements the in-house training activities in providing learning content and in offering learning services” [66].

Actions, Behavior, Affect

Affect

Affect is a term describing a person's emotional state and mood. In the educational context, it can describe a student as bored, confused, engaged and concentrating, and other states. [22] As a student's affect state can greatly influence their learning, teachers must take note of when their teaching produces certain states. This is a task that takes significant effort and would benefit greatly from automation. Botelho et al. [6] used deep learning to create an affect detector using only data from an online educational system. They broke up a student’s use of the system into small time windows, for which they predicted affect state. Features used included the sum, minimum, maximum, and mean values across the time window for a student’s response behavior, response time, and hint and scaffold usage, with 204 total features. They trained a Gated Recurrent Unit (GRU) neural network, an RNN, and an LSTM network each with and without resampling on these features. They measured performance using AUC, Cohen’s Kappa, and Fleiss’ Kappa. The best performing network under AUC was the RNN without resampling, with an AUC of 0.78. The best performing under both Kappas was the LSTM without resampling, with a Cohen’s Kappa of 0.21 and a Fleiss’ Kappa of 0.27. They argue that the deep learning models performed well and can be effectively used [6]. Yang et al. [23] employed active machine learning to improve the detection of student affect on small amounts of data. They used data consisting of 88 unique features such as time spent on problems, hint use, and response correctness in a 20-second time interval. They achieved an AUC of 0.685 for predicting the presence of engaged concentration with 45 observations and concluded that active learning methods can increase the accuracy of affect detectors on low numbers of observations [23]. Studying a student’s time management skills can also be useful. Bosch [9] used AutoML methods to automatically create feature sets to predict efficiency and therefore time management skills for the National Assessment of Educational Progress competition. They used Featuretools and TSFRESH (Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests), two methods for automating the feature engineering process on data from logs of student interactions with the educational system. To compare accuracy, they trained Extra-Trees models, an extension of the random forest model, to predict student efficiency. They compared Featuretools, TSFRESH, and expert-engineered features by comparing per-feature AUC. TSFRESH was the most accurate, with an AUC of 0.530, followed by the expert selected features at 0.512, and Featuretools with 0.502. They investigated the interpretability of automatically generated features and concluded that they were difficult to understand and gave instructors less insight into a student’s learning [9]. These publications show it is possible to automate affect detection with significant accuracy.

Once known, affect state can be used to study other aspects of a student’s educational experience. Hawkins et al. [32] analyzed the causes of boredom in students to determine if problem content or student predisposition is more responsible for boredom. They used students’

problem data labeled by an existing boredom detector to fit two linear regression models, one predicting boredom by student ID and one predicting boredom by problem ID. They found that problem ID had a significantly higher correlation than student ID, with an r-squared of 0.0516 compared to 0.0061, indicating that problem content is significantly more indicative of boredom [32]. Gurung et al. [14] examined student effort and used it to predict performance. They analyzed the time spent between actions for two action-pair types, based on the next action after requesting a hint: hint request into hint request and hint request into answer attempt. They found that the hint request, answer attempt pair had a unimodal distribution, but the hint request, hint request pair was bimodal, with a large number of students taking a small amount of time before requesting another hint and another large amount of students taking a markedly longer time before requesting another hint. They describe the former as “low effort” behavior, and the latter as “high effort” behavior. Since students in the high-effort section spend a similar amount of time as students who requested a hint then submitted an answer, they chose to look at all action pairs and use Gaussian Mixture Models (GMM) to estimate the mean time of high and low effort behavior, using the means and standard deviation to classify a student’s effort based on the time between actions. They then trained a logistic regression model to predict next-problem correctness based on effort and achieved an r-squared of 0.048. They state that low effort behavior was a significant predictor of correctness. They also trained a logistic regression on wheel spinning behavior and achieved similar results [14]. Botelho et al. [24] analyzed the changes in affect state over time to determine relationships between states. They computed D’Mello’s L for each student and affect state change, and compared the set of changes using a one-sample two-tailed t-test. They find that the most common transitions are between confusion to engaged concentration and frustration to engaged concentration and that transitions between boredom and engaged concentration are more likely than chance. They observe that there are no states that transition to confusion more likely than chance [24]. San Pedro et al. [34] applied affect detectors to test Csikszentmihalyi’s Flow theory, which states that adequately challenging material leads to engaged concentration while material that is too easy leads to boredom and material that is too hard leads to frustration and anxiety. They analyzed the correlations of student knowledge and affect on a problem as determined by ASSISTments’ existing detectors. They found that boredom got less common as student knowledge increased, contrary to Flow theory. Frustration appeared most often when student knowledge was at the high or low extremes, partially agreeing with flow theory, but disagreeing for students with high skill. Engaged concentration was directly proportional to student skill, being highest for high skill students. The authors note that engaged concentration is the most common affect state in general [34]. These papers prove that affective state does influence student learning to a significant extent.

Using affect to predict a student’s performance beyond a single course or skill is useful to allow advisors and teachers to provide higher-level guidance. San Pedro et al. [22] predicted scores on the Massachusetts Comprehensive Assessment System (MCAS), a standardized test administered throughout a student’s K-12 education, using measures of affect and behavior. For features, they used existing detectors of student affect and KT from the ASSISTments platform. They obtained entropy scores for fluctuations in student affect, behavior, and knowledge series, describing if the student’s behavior was predictable or not. Hurst exponents were calculated

using Detrended Fluctuation Analysis for fine-grained encodings of how each moment in the time series influences the next. They trained a linear regression to predict a student's MCAS score, and this model achieved an RMSE of 7.862 [22]. San Pedro et al. [29] used student engagement to predict enrolment in a post-secondary STEM program. Training labels came from a survey that asked past students to detail their education/career path after high school. Features used were the results of the ASSISTments system's pre-existing detectors of knowledge (using KT), carelessness, gaming, or other off-task behavior, and student affect, as well as students' usage of the system. The presence of behaviors and affect were averaged across each student. They fitted a logistic regression on the features to predict STEM major enrolment and achieved a mean accuracy of 0.663 [29]. Ramesh et al. [58] used student engagement to predict course completion with a Probabilistic Soft Logic (PSL) model. The behavioral features used were a student's posting, viewing, and voting activity on the class forum, and their reputation on the forum. They used an automated tool called OpinionFinder to qualify forum posts as subjective/objective and define their polarity. A post's thread and forum location were also used. For temporal features, they used a discrete classification of the last time a student posted, voted, or viewed the forum, and the last time they took a quiz or attended a lecture. There were three values for these, "start", "mid", and "end". They created two PSL models, a "flat" one that makes a direct prediction, and a "latent" one that calculates predicted student engagement and uses that to predict course completion. The latent model scored best with AUCs of 0.969, 0.983, and 0.944 across three classes [58]. Affect again proves itself a powerful indicator of student performance.

Gaming The System.

Gaming the system (here "gaming") in the context of educational systems is the practice of abusing hints and scaffolding to arrive at the correct answer without thinking through and learning/applying required skills [7]. This behavior is not conducive to learning. If an instructor or system can detect gaming behavior, they can provide interventions or targeted instruction to help. Walonoski and Heffernan [7] developed machine learning models for the detection of gaming behavior in students. To create training data, they manually observed students' classroom behavior. They broke interaction data from ASSISTments into time windows, with each window recording the number of student's actions on problems that occurred during the window, the total number of attempts, correct and incorrect (first) attempts, time taken for each type of attempt, number of hints and answers given, how much time it took to request hints, the number of questions and scaffold questions answered, multiple choice and short response questions answered, problems replayed, and prior knowledge as described by overall percent correct. Problem difficulty statistics were included, with measures for min/max, average, and standard deviation. Ratios of attempts per problem, correct and incorrect attempts per problem, hints and bottom-out hints per problem, and the number of replays per problem were included. The models were also provided with all 1378 pairwise interactions between features. 12 different models were trained, none of which performed significantly differently from each other, but the authors chose to focus on the J48 decision tree algorithm. The average accuracy was 96%, suggesting a strong ability to detect gaming behavior [7]. Adjei et al. [90] used clustering with school-level data to explore student behavior patterns and predict end-of-year exam scores. Features used for each student consisted of averages of problem features such as percent

correctness, hint use, assignment completion, and problem counts and time use. Attached to the student features were school-level features, including continuous data and categorical features like location. K-means clustering was used to create four clusters of students based on these features, and separate regressions were fitted on each cluster to predict end-of-year exam performance. The paper concludes that the models did not achieve significant accuracy, but clustering in this way is still useful for the goal of personalization. They stated that the four clusters represented distinct learning behaviors, “struggling”, “proficient”, “learning”, and “gaming”. Gaming was an identifiable behavioral cluster, meaning it can be identified and intervention can be provided based on these features [90]. Prihar et al. [2] sought to identify behaviors that negatively impact learning. They used pairs of an action and its previous action from the ASSISTments system as features to train logistic regression, neural network, decision trees, and Bernoulli naive Bayes models. To detect anomalous behavior, they created an “anomaly score” which was the average error of the model’s predictions when compared to the student’s real actions. The students with the top 95% of anomaly scores were labeled as exhibiting abnormal behavior, and they compared the correlation between anomaly score and performance as well as the behavior of students with abnormal behavior. They found that anomaly scores correlated negatively with correctness. They find that students with higher anomaly scores spend slightly less time on problems on average, but get significantly more problems wrong. They also find that students with high anomaly scores spend less time before requesting help, spend less time reading help, and spend less time between getting help and submitting an answer. They claim that this is highly indicative of gaming behavior. Gaming detection in educational systems can be automated and used as an indicator for students who require assistance [2].

Wheel Spinning

In learning, wheel spinning is a type of behavior where a student spends their time practicing a skill but makes no progress towards learning. The student may spend an excessive amount of time on problems but still not learn. This behavior is undesirable, and being able to intervene can help improve a student’s learning experience. Gong and Beck [93] sought to predict wheel spinning using linear regression models with features that are easily accessible in most educational systems. For the skill in question, they used the total number of questions, the skill id, number of correct responses, length of the current streak of first-attempt correctness, z-score of average response time, the number of correct responses that required a hint first, and the number of correct responses that required at least five hints first, and the current streak of questions that required one or more than 5 hints. They also recorded the number of problems over one standard deviation below, within one standard deviation of, and over one standard deviation above the average response time for both correct and incorrect responses. They trained logistic regressions for each of the educational systems they evaluated the models on. On test data, the model received accuracies of 85% and 86% and an AUC of 88% on both [93]. Similarly, Kai et al. [96] used decision trees to distinguish between productive persistence and wheel spinning. They extended the definition of wheel spinning to include students who had completed more than 10 problems but either not successfully answered three problems correctly in a row afterward or failed a post-test. The features they used were the number of unique problems tried, how many of the past five attempts were wrong, the attempt count, the

amount of time since a problem in this set was last seen, the total number of hints used, and the number of bottom-out hints used in the last 8 problems. They then trained a decision tree model, which achieved an AUC of 0.684 with student-skill level cross-validation. Under only student-level cross-validation, it achieved an AUC of 0.636. They note a distinct tradeoff between precision and recall [96]. Botelho et al. [95] used deep learning to identify wheel spinning behavior and low persistence behavior, known as stopout. The features consisted of action type, attempt and hint count, problems seen so far, the probability of the current action, probability of a response, z-score of time taken (compared to other students on the same action type and problem), usage of the penultimate and bottom-out hints, correctness, the last three actions excluding the current one, and the last three actions including the current one. The labels used were same-assignment wheelspin/stopout and cross-assignment wheelspin/stopout. They trained an LSTM network, logistic regression, and decision tree model on each of the four target labels. The LSTM performed best in all labels. In current-assignment wheel spinning, it achieved an AUC of 0.887 and RMSE of 0.313. In cross-assignment wheel spinning, it achieved an AUC of 0.6 and RMSE of 0.251 [95]. These publications prove that automatic gaming detectors are possible and could be put into use.

Once automated detectors are built, it is worthwhile investigating the phenomenon further. Wang et al. [97] explored the number of practice opportunities required to detect wheel spinning. They used decision tree models trained which used data from the first three to the first ten questions in the problem set. The features used were from ASSISTments data relating to hint usage, number of practice opportunities in a skill, number of practice opportunities in a problem set, and the time between actions. For each feature, the sum, minimum, maximum, average, and standard deviation were used as input to the model, resulting in 25 total core features. They applied a forward feature selection algorithm and found that the most used features related to hint use and time. They state that their model was able to differentiate wheel-spinning from productive persistence with only three problems and differentiate wheel-spinning from non-persistence after the first five [97]. Beck and Rodrigo [94] investigated the relationship between wheel spinning behavior and affect state. They computed partial correlations between wheel spinning and affect state, partialling by student's knowledge as determined by a pretest score. They found that wheel spinning was negatively correlated with flow and positively correlated with confusion and gaming the system (correlations of -0.523, 0.476, and 0.437 respectively), and not significantly correlated with boredom or delight (correlations of 0.145 and 0.053 respectively). They conclude that students are more likely to wheelspin if they are not understanding the work and are stuck rather than based on their mood or motivation [94]. As previously mentioned, Gurung et al. [14] predicted wheel spinning behavior based on student effort with a logistic regression model. They achieved a correlation of 0.091, and state that low-effort behavior is a strong predictor of wheel-spinning behavior [14].

Stopout

Studying students' ability to complete assignments is crucial in ensuring they are getting as much practice as is necessary to learn their material. Dropping assignments before completion is referred to as stopout. Detecting a student who is struggling to complete an assignment and might exhibit stopout can help instructors provide aid and encouragement to

help a student's understanding or motivate them to complete the assignment. Botelho et al. [18] sought to analyze student stopout behavior. They grouped students into six clusters based on the percent correct excluding the current assignment, and the last action the student performed before stopping out and analyzed which problem opportunity students stopped out on. They found that the majority of students stopped out before taking an action and that a disproportionate number of students stopped out on the first learning opportunity. This occurred at all knowledge levels. They define this behavior as "refusal". They analyzed student confidence by a survey and found that students who exhibited refusal also were significantly less confident of their knowledge [18]. As previously mentioned, Botelho et al. [95] created machine learning models to predict students' wheel spinning behavior. They also predicted stopout. The LSTM model performed best at predicting stopout, achieving an AUC of 0.7589 and RMSE of 0.223 on current assignment stopout, and an AUC of 0.557 and RMSE of 0.221 on cross-assignment stopout [95].

Demographic and Socioeconomic Status

Student demographics have been investigated as predictors of performance and can be used to provide targeted instruction to certain groups. For example, Zhong et al. [27] tracked student performance across three months in a mathematics course on the ASSISTments platform in an attempt to understand if students learned at a similar rate or if there was a difference between male and female students. They tracked student gender, id, and performance on exams in three consecutive months. They performed a one-way multivariate ANOVA test to determine if there was a difference between the two groups and concluded that there was none ($p=0.3974529$). They performed an analysis on the growth between months to determine if there were differences and found none ($p=0.3974529$). Both tests were conducted at a significance level of 0.05 [27]. Kupczynski et al. [102] examined the relationship between gender and final letter grade in an online course when holding GPA constant. They used simple main effect tests to analyze the potential differences between genders in three different GPA groups, split by percentile. The difference was significant for the low-GPA group, but not significant for the mid and high-GPA groups. The tests showed that female students in the low-GPA group scored higher than their male counterparts on average, while there was no significant difference for the others [102]. These studies show that while gender has no significant role overall, struggling students may behave differently based on their gender.

It is important to measure the impact of demographics other than gender. Students have many characteristics which might determine their learning outcomes such as socioeconomic status and race. Bahar [91] examined the relation of perceived social support, social status, and gender on academic performance. Social status was measured by surveying students for a list of the three other students they liked spending time with the most, and the three they liked spending time with the least, in order. The listed students were assigned points from 3 for the most liked to -3 for the most disliked. A student's popularity was determined by summing the points they were assigned by all other students. Perceived social support from family, friends, and a special person was measured with the 12 Item Multi-Dimensional Perceived Social Support Scale. Gender was also used as a feature. A multiple linear regression was fitted to the

features to predict an unspecified academic success score. It was found that the features predicted 15% of academic success. All features contributed, but for perceived social support, only support from family impacted success. Perceived support from friends and someone special had no impact [91]. Battle and Lewis [100] measured the impact of race and socioeconomic status on academic achievement. Academic achievement as measured was composed of test scores from 12th grade and the level of post-secondary education achieved two years after high school graduation, from none to studying for a degree to having achieved a degree. Features used were 8th-grade exam scores, race, gender, school setting (urban or not), if the school is public or private, family size, amount of parental control, Cultural Capital (access to classes or events/museums in music, arts, history), Social Capital (parental support of academics), the percentage of students at the school eligible for free or reduced-price lunch, and Socioeconomic Status. The effects of these features were analyzed with regression models. They found that race, public/private schools, cultural capital, and socioeconomic status played a significant impact at all times. Gender was found to have significant differences in 8th grade and two years after graduation, but not for 12th grade. Family size and percentage eligible for free or reduced-price lunch played a role only in 12th grade. Social Capital was found to play a significant role only two years after graduation. Urbanicity had no significant impact. Overall, they conclude that socioeconomic status is three times more significant a predictor than race and that students of African American descent “don’t perform as well as their white counterparts”, but performed better when controlling for socioeconomic status, and that white students benefited more from increased socioeconomic status [100]. As previously mentioned, Adjei et al. utilized demographics alongside student behavior to measure end-of-year exam scores [90]. Based on these studies, demographic characteristics play a much lower role in predicting a student’s success than available resources and socioeconomic status.

Problem Content

Often, methods of modeling educational progress focus primarily on student data. However, there is substantial value in incorporating problem features into student modeling. The content of educational resources, from content tags to linguistic data, can provide important insight useful to understanding and improving student education. Student engagement can provide insight into the educational process. If the content is not engaging enough, students may not learn as well or efficiently as they could. In 2016, Slater et al. [19] explored the relationship between problem content and student learning and engagement. Problem content features were word count, semantic features in the form of base and appended tags created by the Wmatrix text analysis and comparison tool, the count of mathematical elements, and certain HTML tags used in the questions. To create training labels, existing detectors of learning and engagement were used with the Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP) to identify affect states and measure learning. Then, the correlations between each problem feature and the training labels were measured, and the Benjamini and Hochberg post-hoc procedure was applied to statistical tests to prevent false findings of significance. They found that 41.3% of all possible correlations between features and the dependent variables were significant. Confusion had the most features correlated with it, while gaming the system, frustration, and concentration all also had large numbers of features correlated with them. They

propose that the features used could be used to help improve the learning quality of problems, reworking them to remove features correlated with unwanted behavior and low learning growth [19]. Similarly, Slater et al. [25] used natural language processing (NLP) to measure student learning and engagement while exploring the effects of different linguistic features. Training labels were again created using BROMP. Linguistic features were generated using Wmatrix for content tagging, and the Tool for the Automatic Analysis of Lexical Sophistication (TAALES) for sophistication. They also included information about the presence of mathematical symbols in the HTML and the amount of assistance available to the student in the problem as well as problem type and average correctness. They fitted a regression model for each affect state and investigated the quality of each model's fit. The model for confusion performed the best, achieving an RMSE of 0.042, followed by frustration at 0.078, boredom at 0.138, and concentration scored the worst at 0.441. They investigated the influence of individual linguistic features on each affect state and found relations between features that predict concentration and confusion, with similar relations found between frustration and boredom. Again, they conclude that the features analyzed could be used to enhance problems for learning [25]. These studies suggest that the textual content of a problem is related to student engagement on that problem, implying that problems with low engagement can be improved to enhance learning.

Text classification in the context of education is not necessarily limited to just problem contents. It can be beneficial to explore methods of classifying other types of texts for other goals, as these could be applied to problems like the methods discussed in the above paragraph. Kurdi [68] analyzed the linguistic contents of a text to classify its complexity for use in English Second Language teaching. They used phenological, morphological, lexical, syntactic, inter-sentential, and psycholinguistic features, as well as using formulas designed to measure text readability. They then trained five machine learning models to predict text complexity as one of three classes based on these features. Models used were logistic regression, multi-layer perceptron networks, AdaBoost, bagging, and a random forest algorithm. Logistic regression performed the best, with all other models performing similarly except for AdaBoost. AdaBoost performed significantly worse than all other models tested [68]. Vajjala and Meurers [75] created models for differentiating two texts by relative reading level. They used metrics of lexical richness based on Second Language Acquisition research and complexity, utilizing variation in and densities of parts of speech. For complexity, the mean lengths of units of text, coordination and subordination, use of certain structures, the occurrence of and length of types of phrases, the height of the parse tree. They also used the morphological and syntactical components of lemmas from the Celex Lexical Database, a database describing various properties of words and their frequency in a large corpus, as features. From the MRC Psycholinguistic Database, a database cataloging psycholinguistic features of words, they used the average across the entire text of familiarity, concreteness, imageability, meaningfulness, and age of word acquisition, as well as the average number of senses per word. They then applied a document-level model, a sentence-level binary classification model to classify sentences between easy and hard, and a relative model using the document-level one applied to the sentence level. The first model achieved an RMSE of 0.53, the best out of all tested models. The sentence-level binary classification model only achieved an accuracy of 66%, and the authors applied the document-level model to sentences because of this low accuracy. The

document-level model applied to sentences had difficulty identifying the simplified version of an already simple sentence, achieving an accuracy of between 65 and 75%, but for sentences where the difference was large, it achieved accuracy between 80 and 95% [75]. These papers show that classifying other aspects of a text can lead to metrics about the text that can be useful in an educational context.

Assigning Knowledge Components

Knowledge components (KCs) are the individual skills or concepts that are present or tested in a resource, and it can be useful to systematically assign them to content, so the content can be adequately organized and easily accessible. This task normally takes instructors an excessive amount of time which could otherwise be used to enhance students' learning. On top of this, it can be error-prone. To attempt to solve these issues, Kardian and Heffernan in 2006 [61] created a semi-automatic system for tagging KCs in questions based on a NaiveBayes model. They focused only on questions that only had one KC component. They used the words in the problem as features in a NaiveBayes model and achieved an accuracy of 40% when selecting from 78 skills, and 51% when selecting from only 39. They conclude that the model can provide reasonably accurate suggestions to human coders, who would then have to select the best one [61]. Karlovčec et al. [70] created a system based on support vector machines (SVMs) and K-nearest neighbors (KNN) algorithms to suggest KCs to human coders. The SVM model was trained on a set of labeled questions for a text mining approach. The KNN model found the most similar texts from the training set and suggested their KCs. Models of both types were trained on sets of problems with 109 KCs, 39 KCs, and 5 KCs. The SVM model performed better than the KNN model, especially when suggesting more KCs [70]. Moore et al. [83] experimented with crowdsourcing the identification of KC tags and attempted to evaluate the effect of priming on KC choice. Crowdworkers were first either primed with two problems similar in concept to the target question or directly shown the target question. Workers were asked to identify three skills required for the target question. Generated tags were then compared to expert-created tags. It was found that roughly a third of generated tags directly matched the expert’s choices. Priming workers did not have a significant impact on accuracy. They conclude that with the current method, crowd workers can generate a selection of KCs for experts to choose from [83].

Slater et al. [26] automated the labeling of KCs for crowdsourced content using Correlational Topic Modeling (CTM). They first preprocessed content text by removing HTML tags and replaced mathematical content with distinct strings, such that expressions would be detected as one string preserving meaning rather than many which are just numbers and operators. They trained three CTM models, one which predicted 5 KCs per question, one which predicted 15, and another which predicted 25. They measured accuracy through a perplexity score, and the model predicting 25 KCs scored the best at 189.28, followed by the 15 KC model at 227.40 and the 5 KC model at 319.91. They state that the model was not only able to identify major skill categories, but also identify and separate common hints or phrases and non-mathematical themes such as the contextual setting of a problem [26]. Yacobson [89] used machine learning models to attach KCs to problems. They selected 50 mathematical keywords which were commonly used in the problems and encoded the presence of each in the problem

as a one-hot vector. Then, a Naive Bayes, Random Forest, and Logistic Regression model were trained on the vectors. The Naive Bayes and Random Forest models achieved an accuracy of 95% [39]. Shen et al. [80] proposed a new method of classifying KC components using Task-adaptive Pre-trained BERT as well as a method for correcting incorrect predictions. Bidirectional Encoder Representations From Transformer (BERT) is a language processing model that comes pre-trained on a large corpus of books. It is possible to do a custom pre-training process to similar problems. Such a model is called Task-adaptive Pre-trained BERT, or TAPT BERT. The features are the raw text, and the output consists of KCs which are present in the text. They trained two models, a baseline BERT with only fine-tuning, and a TAPT BERT model which was pre-trained on problem data and fine-tuned with labeled texts. Model performance was measured as prediction accuracy on a test set. They then propose a metric called TEXSTR for evaluating the fitness of KCs, which compares the similarity of KCs descriptions and titles. It uses semantic similarity to capture similarities in description and structural similarity to capture how similar KCs are to each other based on prerequisites. TAPT BERT performed the best, while the baseline BERT performed 2nd best in three of four datasets. Then, they used TEXSTR to evaluate misclassifications, judging how correct or incorrect a choice might be, and choosing a different one if needed. They were able to reconsider up to 73% of misclassifications [80]. These papers suggest that machine learning models can be used to classify the knowledge components contained in a resource.

Video recordings are becoming increasingly important in learning, and labeling KCs accurately, especially with timestamps, can take even longer than for text content. To attempt to solve this issue, Liu et al. [89] proposed ConceptScape, a collaborative system where learners work together to crowdsource a concept map for video lectures. Each concept in the map links to a specific timestamp in the lecture video where it is discussed. The ConceptScape system consists of three steps, one where learners add the locations of concepts while watching, a pruning step to remove duplicates, and an adjust step to modify existing entries. A similar workflow was created for adding and labeling links between concepts. ConceptScape performed significantly better than a novice at creating a map, and similar to an expert [89]. Chang et al. [98] created the Keyword-based Video Summarization (KVSUM) model to generate summaries for video lectures. The summaries consisted of a keyword cloud with video fragments and transcripts for parts of the lecture attached to each keyword. The input for the model was raw video, which was then split into individual frames, subtitles, and timestamps. The subtitles are used to extract a weighted map of keywords and times of their use. They performed a study where students were given a pretest, presented with either a video annotated with the KVSUM model or a fast-forwarded and human-annotated video, then given a posttest. Fast-forwarded video was used as a control, as the authors describe it is a method used by students who want to obtain the information quicker. The KVSUM model performed significantly better with a mean posttest score of 4.02 compared to fast-forwarding's mean score of 1.47, even at a p-value of 0.001. This suggests that the KVSUM model is very useful to students [98]. These papers show that the automatic classification of knowledge components in video content is not only possible but creates useful content for students.

Student Responses

Automated Grading of Essay Responses

Just as the content of problems can be used to extract meaningful information about learning, so can students' responses to problems or other educational content. Grading student essays or short response answers is a time-consuming but necessary task. It is also one that is not trivial to automate. If it can be automated, the time normally spent grading can be used to improve learning materials or provide more personalized learning to students. Erickson et al. [17] attempted to automate the grading of math open-response questions using deep and machine learning NLP techniques. They tokenized student responses, splitting them up into individual words by spaces, punctuation, and contractions, and recording the frequency of each token in the response. They used the tokenized representations to train a random forest, XGBoost, and LSTM model, using them in an ensemble with a baseline Rasch model. They found that the Rasch model using student response length and the Random Forest model achieved the highest AUC of 0.850, while all models incorporating machine or deep learning models achieved AUCs of at least 0.827. They conclude that the models are reliably able to predict a response's grade [17]. Rosé et al. [71] created CarmelTC, a system that evaluates the syntactic content of students' responses to Physics open-response questions to predict correctness. They aimed to classify each sentence in a student's response as expressing one or none of a set of components required for a correct answer. To accomplish this, they broke up the answer into individual sentence strings, splitting run-on sentences. They then extract the structure of the sentence using a parser, as well as run the sentence through a Naive Bayes classification using a bag of words approach. They encode the results of these processes as a vector and use a decision tree algorithm to determine the sentences' classification. The authors describe the approach as using the features provided by the syntax analysis to determine if the bag-of-words classification is accurate, or what to look at instead. They state that CarmelTC achieved a precision of 90% with a recall of 80%, the best of all tested models. They propose that the model can be used to determine what feedback a student needs to see based on the correctness of components in a potential answer [71]. Chuan et al. [72] created an automatic grading system for determining the writing quality of a student's argument. They used a "Coh-metrix" to determine the coherence and cohesion of the writing and the occurrence of n-gram patterns between a list of the connector and content words related to the problem prompt. They then trained a decision tree, SVM, and k-nearest neighbors model on the "Coh-metrix" data, and a k-nearest neighbors model on the n-gram data. They found that the best model for multi-class classification was the n-gram model with an n of 3 achieved an accuracy of 81.74% [72]. Nielsen et al. [76] developed models for recognizing when a student is learning from tutor intervention based on their response content. They pre-processed data to fix spelling mistakes, but not grammatical ones, and removed non-answers from consideration. They utilized syntactic dependency parsing and syntactic parsing to generate features such as the amount of content matching reference answers, the differences in the structure of the student answer and the reference answer, and the number of negations and content words present. They evaluated several machine learning models and concluded that the C4.5 model produced the best results, with an accuracy of 75.5% on their Unseen Answers test, which they

state is most similar to real ITS conditions [76]. These papers suggest that the automatic classification of student essays based on their content and machine learning algorithms is a viable method of grading.

User's Ratings and Opinions of Content

Student feedback is crucial in understanding the efficacy of educational content. Students are the experts on their learning experience, Barrón-Estrada et al. [69] analyzed students' written feedback to track their opinions on educational content. They utilized Sentiment Analysis, an NLP technique, to track the positive or negative sentiment in feedback. To accomplish this, they first preprocessed text by translating slang terms into consistent text equivalents, removing modifications on words to get to their root, and filtering out common words such as articles and prepositions. Then, they used the number of times each word was used in a response and the number of documents in the training data that contain that word as features. They then trained a Naive Bayes classifier to classify feedback as positive or negative sentiment. The model achieved a recall value of 0.81, which the authors consider substantially good accuracy [69]. Oramas-Bustillos et al. [92] created a corpus of computer-science-related texts for use with sentiment analysis and tested a sentiment analysis method on it. They allowed students to access a series of learning materials associated with their course and gathered their written (Spanish) feedback and opinion category for their selected resource. Then, they trained a Bernoulli Naive Bayes classifier after pre-processing the text as in [69]. They achieved an accuracy of 40.70%, and they note that while this is low they believe it is due to the small sample size and the unbalanced nature of the data, where a vast minority of the opinions were negative [92]. Wang et al. [84] created a crowdsourcing technique for evaluating students' average opinions about the content. In their framework, a teacher sends surveys to students about specific portions of lecture materials after they are presented. The questions ask students to rate how well they think they learned the presented KC. Students' response contents are then converted into a set of linguistic 2-tuples, using a symbolic translation process, which is then aggregated to determine the quality of teaching as indicated by the student response. Each student's score then updates the overall score for the question. They experimented with using their system in a real class setting, and state that it proved resilient to abnormal behavior and was able to provide an estimation of teaching quality [84]. These papers all present viable methods of incorporating student sentiment into teaching using machine learning methods.

Personalizing and Generating Feedback

An interesting application of understanding students' responses is the automatic generation of feedback or hints. In the course of normal learning, an instructor must either provide generalized hints for a problem and hope they apply to most situations a student will encounter or actively look at a student's responses to provide personalized hints. The former method is not always useful as it is often impossible or impractical to provide hints for every situation a student may find themselves in with a vast amount of problems. The latter is exceptionally time-consuming, especially in the ever-growing space of online courses with large enrollment. To alleviate this issue, Kochmar et al. [56] created a system to automatically generate personalized hint text based on which aspects of an answer a student got wrong. They

generated a collection of potential hints based on the contents of reference answers. They used the spaCy tool to analyze the reference answers. They identified keywords and key phrases and eliminated potential explanations that contain them. Hints were generated using discourse-based modifications of phrases that did not contain overly specific keywords. They created an algorithm to determine which generated hint is the best quality and most appropriate for the current student. They trained a random forest classifier on features from the hint, student performance, and the student's previous interactions with the tutor. Hint-level features included hint length, the proportion of the sentences with a complete subject-verb structure, a "perplexity score" that encompassed the quality and grammatical correctness of the writing, word overlap with the question, uniqueness of keywords compared to reference solutions, the ambiguity of keywords, and proportions of various parts of speech. Performance features were the number of questions and attempts the student has used, the proportion of correct and incorrect questions both in total and at the current time, and the total length of the student's use of the system. Features from the student's interactions with the tutor were the proportion of keywords present, the topic overlap between questions and student statements, and the "perplexity score" for each of the student's last four interaction turns. The model incorporating all features achieved an accuracy of approximately 86.71%. After performing a pilot study in the KORBIT ITS, they claim that the results show that personalized hints generated with this method are helpful to students and significantly increase performance [56].

Williams et al. [11] sought to alleviate the issues of hint creation by asking learners to create their own explanations for problems. The system both asks users to generate their own explanations while working on a problem and rate existing explanations by other students. The system treats the selection of explanations as an instance of the multi-armed-bandits problem, presenting options and measuring their effectiveness, and using a Thompson sampling algorithm. They claim that explanations generated and curated by their system were significantly better and showed a higher impact on learning than non-curated explanations [11]. Rosé et al. [62] created a system to select hints to encourage students to include components of answers to physics problems that they may have missed. The system combines a bag-of-words approach and features created by a parser by the use of a decision tree algorithm to predict which components of the answer are present in the text. They found that their model achieved a precision of 88%, and recall of 75%, while only having an 8% false-positive rate [62]. Rosé and VanLehn [73] explored the use of CarmelTC [71] for the selection of hints. CarmelTC pre-processes student text into sentences, breaking run-on sentences, and extracting the semantic structure using parsers. They combine these features with a Naive Bayes classifier to determine which parts of the answer the sentence contains. They conclude that CarmelTC is an appropriate method of evaluating the content of student answers and providing hints to improve the answer [73]. Ye and Manoharan [86] grouped students' answers by their correctness as determined by the meaning of their writing, allowing teachers to provide a single piece of feedback to cover an entire group's mistake. They produced vector sentence embeddings that would take into account the meaning and positioning of words in the answer using a 4-layer bidirectional LSTM. They then used k-means clustering to group together students with similar answers. They concluded that their model showed high accuracy in its grouping, though it made mistakes on sentences that were very similar in structure and only differed in a few keywords

[86]. These publications describe how generating and selecting hints automatically is viable based either on natural language processing techniques or crowdsourcing.

Processing students' written text can also be useful to provide effective feedback in peer tutoring situations. Peer tutoring is a learning activity in which a learner guides another learner through problems, providing their own interventions and help as needed. The quality of the tutor's help is important, as good help can not only increase the learners' knowledge but also the tutors. Walker et al. [74] proposed a system that automatically determines a tutor's teaching quality and provides teaching hints to the tutor. They classified the type and quality of the tutor's help using features from Taghelper Tools, a tool for automatically coding dialogue. They augmented these features with the problem context, the presence of problem-specific dialogue, such as the name of a mathematical operation, the use of problem-specific dialogue in previous messages, and students' self-classification of the dialogue. They used these features to train Taghelper's classifier and achieved a kappa of 0.81 on the classification of help type. They conclude that their system provides a solid argument for the use of such systems in improving students' peer tutoring [74]. Walker et al. [77] used the Taghelper Tools method to create an adaptive peer tutoring system and investigated its effectiveness. The classifier achieved an accuracy of 94% but only achieved a kappa of 0.53. They state that this is because it only correctly identified the presence of conceptual help 50% of the time. They claim, however, that the classifier is overall accurate in determining the quality of students' conceptual help. They found that the adaptive system produced increased posttest scores compared to the control (0.39 vs 0.28), as well as increased help quality [77]. These papers show that using natural language processing to enhance users' peer-tutoring can provide valuable insight to tutors and help students learn better.

Usage and Effectiveness of Feedback and Hints

Different types of feedback or interventions can have differing effects on students. Studying their effects can allow instructors to better utilize and administer the different types of feedback. Lang et al. [21] investigated the effect of questioning students about their confidence as an intervention. Students were placed into a control and experimental group. Students in the experimental group were asked to rate their confidence in solving problems in the topic before starting an assignment, then after answering three questions. Students in the control group were asked filler questions. They found that in most situations, there was no difference, but for students who got the first question wrong, the probability of getting the second question wrong was higher. They conclude that measuring student confidence may be useful in other modeling tasks [21]. Kehrer et al. [36] investigated the effectiveness of immediate feedback during assignments. Students were placed in either a control group that received correctness feedback for homework on the next day or the experimental group that received immediate feedback on completing a question. They found that students in the immediate feedback group learned 12% more and that the difference was significant [36]. Singh et al. [47] confirmed prior findings that immediate feedback was helpful, and extended them by questioning whether the quality of the feedback was more important than its timeliness. They conducted a study in which students were either given only feedback on their problem correctness or feedback containing

correctness and tutoring. They found that “quality” feedback containing tutoring had a much greater effect on learning than just correctness alone [47]. Razzaq and Heffernan [49] also studied the timing of hints, comparing proactively giving students hints on incorrect answers and waiting for students to ask for hints. Students in the study participated in a repeated-measure experiment using the ASSISTments system, where they completed assignments with proactive hints or on-demand hints. Gain between pre and post-tests was used to measure learning. They report that students learned significantly more with on-demand hints [49]. Gong et al. [42] experimented with providing pre-made hint feedback from external websites. They hand-selected web resources that would be presented to students on demand. In cases where there were multiple skills associated with a problem, the system selected the resource associated with the most difficult skill. They split students into groups, one for students who got a problem wrong and requested web help, and one for students who got a problem wrong but did not request web help. They normalized next-problem correctness based on difficulty so that an easier correct problem shows less learning than a difficult correct problem. Their results showed that students who received web feedback learned more, averaging a learning value of 0.50 compared to 0.40 for regular feedback [42]. These studies show that feedback that does more than just reveal the answer, even feedback from an external source, and which is provided promptly as soon as a student requests it can have a significant positive impact on learning.

Videos can provide an alternate and often more engaging form of feedback than text. To study this, Kelly et al. [37] investigated the addition of teacher-created motivational videos as feedback. They performed two studies, one to measure the effect of videos on perceived learning, and one to measure the effects on homework completion. The first study placed students in one of three groups, one with no video, one with a control video encouraging the student to continue, and an experimental group explaining the value of learning to the students’ future and learning. They found that the experimental value video had a significant positive impact, while no impact was found for the control video. In the second study, they measured the effect of the three video types on homework completion. They found the same results, with the value video having a significant effect but the control having no effect [37]. Ostrow and Heffernan [30] compared the effects of video and text feedback. Students participating in the study were split into four groups, each one experiencing video and text feedback in different orders. They analyzed performance on the second question for students who got the first question wrong, comparing between feedback types. They found that students who received video feedback performed significantly better than those who received text feedback. Interestingly, they also found a much higher percentage of gamers in students who received text feedback. Only 7.1% of students in the video feedback condition were removed from the data for gaming, while 43.5% of students in the text feedback condition were removed for gaming [30]. Zhao and Heffernan [52] created a model to predict whether a student would benefit more from video or text feedback. They trained a Residual Counterfactual Network (RCN) using ASSISTments data containing the students’ information, the intervention types they received previously, and 15 other features such as student and class past performance. The RCN performed best when compared to other models, achieving an RMSE of 1.1 [52]. These publications reveal that video feedback can often provide a student with better learning than text feedback and that it is possible to predict when a student would benefit more from each type.

Crowdsourced Hints

The creation of hints can consume a lot of time that could be better spent helping students in other ways. If crowdsourced hints can be created with equal quality to teacher-created hints, and have a similar learning effect, it would certainly be beneficial to remove the workload from individual teachers. Whitehill and Seltzer [87] crowdsourced math tutorial videos from untrained web users to determine if crowdsourcing quality tutoring is possible. They asked Amazon Mechanical Turk users to create short tutorial videos about logarithms, and randomly sampled 40 of them to test. They had more Mechanical Turk users take a pre-test, watch a randomly assigned video from the sample, and take a post-test to measure learning gain. Two percent of the time, the users were randomly assigned to a control video. They found that the average learning gain of 0.105 was significantly higher than the control video's 0.045, and not significantly different from an expert-created Khan Academy video. They conclude that crowdsourced tutorial videos are feasible [87]. Patikorn and Heffernan [15] tested the effectiveness of crowdsourced feedback using the ASSISTments platform. They created a system for crowdsourcing content they dubbed TeacherASSIST. The system allowed teachers to create assistance for problems they assigned to their students, and have their hints distributed to other teachers using the same problems. They then conducted a randomized controlled trial between students given the crowdsourced hints and students who were simply given the answer. They measured the student's behavior on the next problem, categorizing the behaviors as getting the problem correct on the first try, asking for help, stopping out, or the number of attempts needed. They grouped observations once by the student and once by problem, and conducted paired t-tests. They conclude that students who received the crowdsourced hints learned better, as they were less likely to require assistance on the next problem [15]. Prihar et al. [3] tested the effectiveness of teacher-created crowdsourced materials and presented a system for comparing effectiveness between teachers. Following up on a previous study [15], they found that crowdsourced teacher-created content had a positive effect on students' learning. To determine the relative quality of tutoring, they trained a regression to predict next-problem correctness on student features, which teacher's tutoring they received, features of the problem they got wrong, and features of the next problem. They were able to determine relative quality between teachers using a variance-covariance matrix using each teacher's mean and standard deviation for teaching quality. They found that only a subset of teachers had a statistically significant impact on student learning. They also found that while it was not possible to create a definitive ranking of teachers, it was possible to determine whether or not a certain teacher's content was better than a certain other teacher's content. They investigated the possibility of personalizing which teacher's content was presented to a student based on knowledge level, but did not find any significant effect. However, they conclude that personalization based on other factors could be possible [3]. We conclude that crowdsourcing is a valuable tool and can be leveraged to produce quality feedback and hints.

Limitations

This review is inherently limited in its scope. While we covered a broad swath of subjects in the selected papers, there are many more which could not be included. Additionally, a number

of the initially chosen papers turned out to be unsuited to the structure and purpose of the review and had to be dropped. While it may be impractical to include every publication in a single review, an increased number of publications can broaden the scope if the papers are well selected. All publications in this review were written in English or received a quality translation into English. There are many publications from universities and institutions which are not written in English, or only have poor quality translations. These undoubtedly contain useful information and interesting discoveries. However, it was impractical for us to translate foreign papers or make sense of existing but poor-quality translations. Future reviews will undoubtedly help cover more topics of interest and allow more depth to topics we have covered here.

Future Works

While we have reviewed a substantial amount of papers covering a broad sweep of subjects, there is always room for more improvements. While knowledge tracing is a very well-studied problem, we feel there is room for more research. The knowledge tracing problem is often treated as a binary correctness problem based on student correctness. As such, any effort to personalize its parameters could benefit from partial pooling. We believe [10], [41], and [46] are good examples. We reviewed several publications proposing and testing systems for the personalization of learning paths or selection of learning materials [57] [59] [63] [64] [65]. A natural next step for this research is to apply similar systems to the selection of feedback at each problem.

We have reviewed papers [36, 47, 52] that suggest that the right type of timely feedback can enhance learning. We believe recommendation systems such as those described in [57] [64] could be extended to select feedback for students. Similarly, clustering approaches like [86] could be applied to automated hint selectors like [73] to potentially reduce complexity. [56] is a paper of great interest, and we believe the methods discussed should be applied and tested in other domains. Other publications related to the selection and generation of personalized hints should have their performance in various domains explored. In particular, we would suggest modifying the approach of [71] and [73] for a mathematical context. Automated grading systems such as those described in [17] and [71] work well for objective facts like mathematical questions or descriptions of real systems, but we would like to see these concepts applied to more subjective problems such as essay questions relating to the understanding and comprehension of literature.

Affect detection is another area of significant interest. We would also like to see more investigations into the causes of student behavior and affect. For example, [18] and [95] investigated the occurrence of Stopout but did not delve into its causes. We believe affect is related to Stopout and would encourage future investigation. We believe investigations into the presence of affect states before stopout or gaming behavior similar to the study of affect state transitions in [24] would be useful. We would like to see investigations of how different interventions impact affect state. [21], [36], [37], [42], and [47] all study the educational impact of different feedback types. We would like to see investigations of the impact of types of feedback

and learning materials on motivation, affect, and potentially behaviors like gaming, wheel-spinning, or stopout.

Conclusion

Online educational systems naturally lend themselves to the collection and use of student data. They can record and analyze data from a student's correctness and learning, their affect state, their use of feedback, and even the written content of their responses. We have observed numerous extensions to the knowledge tracing method of student modeling, including clustering, individualizing parameters, the method's use in other problems, and a deep-learning implementation of it. These methods provide accurate models of student performance, but we have also reviewed alternative solutions to the knowledge tracing problem such as PFA. We reviewed methods of using partial credit or prerequisites to inform more accurate knowledge predictions. We reviewed the use of similarity metrics to create learning path recommendation systems based on the performance of similar students. We have reviewed automated detectors of student affect as well as gaming, wheel spinning, and stopout behaviors. These detectors can provide insight into students' behaviors. We reviewed the impact demographics and socioeconomic status may have on student learning to provide everyone with the support they may need and discovered that a student's socioeconomic status plays a major role in the quality of their learning. We investigated the use of problem content and features as data, including methods for automatically tagging knowledge components and connections between problem features and student learning or affect. We reviewed methods of using student responses as data, from using natural language processing to grade answers to automatically generating quality personalized hints. Students' opinions of and learning outcomes from educational content have provided valuable insight into the creation of quality educational material. Overall, these publications indicate that student data can be used to improve a student's educational experience to great effect. The methods we discussed should be useful to anyone looking to utilize student data for personalization.

Bibliography

2. Prihar, Ethan, Alexander Moore, and Neil Heffernan. "Identifying Struggling Students by Comparing Online Tutor Clickstreams." *International Conference on Artificial Intelligence in Education*. Springer, Cham, 2021.
3. Prihar, Ethan, et al. "Toward Personalizing Students' Education with Crowdsourced Tutoring." *Proceedings of the Eighth ACM Conference on Learning@ Scale*. 2021.
4. Corbett, Albert T., and John R. Anderson. "Knowledge tracing: Modeling the acquisition of procedural knowledge." *User modeling and user-adapted interaction* 4.4 (1994): 253-278.
5. Pavlik Jr, Philip I., Hao Cen, and Kenneth R. Koedinger. "Performance Factors Analysis--A New Alternative to Knowledge Tracing." *Online Submission* (2009).
6. Botelho, Anthony F., Ryan S. Baker, and Neil T. Heffernan. "Improving sensor-free affect detection using deep learning." *International conference on artificial intelligence in education*. Springer, Cham, 2017.
7. Walonoski, Jason A., and Neil T. Heffernan. "Detection and analysis of off-task gaming behavior in intelligent tutoring systems." *International Conference on Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 2006.
8. Botelho, Anthony, Hao Wan, and Neil Heffernan. "The prediction of student first response using prerequisite skills." *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. 2015.
9. Bosch, Nigel. "AutoML Feature Engineering for Student Modeling Yields High Accuracy, but Limited Interpretability." *Journal of Educational Data Mining* 13.2 (2021): 55-79.
10. Pardos, Zachary A., and Neil T. Heffernan. "Modeling individualization in a bayesian networks implementation of knowledge tracing." *International conference on user modeling, adaptation, and personalization*. Springer, Berlin, Heidelberg, 2010.
11. Williams, Joseph Jay, et al. "Axis: Generating explanations at scale with learnersourcing and machine learning." *Proceedings of the Third (2016) ACM Conference on Learning@ Scale*. 2016.
12. Piech, Chris, et al. "Deep knowledge tracing." *arXiv preprint arXiv:1506.05908* (2015).
14. Gurung, Ashish, Anthony F. Botelho, and Neil T. Heffernan. "Examining Student Effort on Help through Response Time Decomposition." *LAK21: 11th International Learning Analytics and Knowledge Conference*. 2021.
15. Patikorn, Thanaporn, and Neil T. Heffernan. "Effectiveness of crowd-sourcing on-demand assistance from teachers in online learning platforms." *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 2020.
16. Ghosh, Aritra, Neil Heffernan, and Andrew S. Lan. "Context-aware attentive knowledge tracing." *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020.
17. Erickson, John A., et al. "The automated grading of student open responses in mathematics." *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*. 2020.
18. Botelho, Anthony F., et al. "Refusing to try: Characterizing early stopout on student assignments." *Proceedings of the 9th International Conference on Learning Analytics & Knowledge*. 2019.
19. Slater, Stefan, et al. "Semantic Features of Math Problems: Relationships to Student Learning and Engagement." *International Educational Data Mining Society* (2016).
20. Van Inwegen, Eric G., et al. "Using Partial Credit and Response History to Model User Knowledge." *International Educational Data Mining Society* (2015).
21. Lang, Charles, et al. "The Impact of Incorporating Student Confidence Items into an Intelligent Tutor: A Randomized Controlled Trial." *International Educational Data Mining Society* (2015).

22. San Pedro, Maria Ofelia Z., et al. "Exploring Dynamical Assessments of Affect, Behavior, and Cognition and Math State Test Achievement." *International Educational Data Mining Society* (2015).
23. Yang, Tsung-Yen, et al. "Active learning for student affect detection." *International Conference on Educational Data Mining (EDM)*. 2019.
24. Botelho, Anthony F., et al. "Studying Affect Dynamics and Chronometry Using Sensor-Free Detectors." *International Educational Data Mining Society* (2018).
25. Slater, Stefan, et al. "Using natural language processing tools to develop complex models of student engagement." *2017 Seventh International Conference on Affective Computing and Intelligent Interaction (ACII)*. IEEE, 2017.
26. Slater, Stefan, et al. "Using correlational topic modeling for automated topic identification in intelligent tutoring systems." *Proceedings of the Seventh International Learning Analytics & Knowledge Conference*. 2017.
27. Zhong, Xiao, et al. "Learning Curve Analysis Using Intensive Longitudinal and Cluster-Correlated Data." *Procedia computer science* 114 (2017): 250-257.
28. Ostrow, Korinn, et al. "Improving student modeling through partial credit and problem difficulty." *Proceedings of the Second (2015) ACM Conference on Learning@ Scale*. 2015.
29. San Pedro, Maria Ofelia, et al. "Predicting STEM and Non-STEM College Major Enrollment from Middle School Interaction with Mathematics Educational Software." *EDM*. 2014.
30. Ostrow, Korinn, and Neil Heffernan. "Testing the multimedia principle in the real world: a comparison of video vs. Text feedback in authentic middle school math assignments." *Educational Data Mining 2014*. 2014.
31. Hawkins, William J., Neil T. Heffernan, and Ryan SJD Baker. "Learning Bayesian knowledge tracing parameters with a knowledge heuristic and empirical probabilities." *International Conference on Intelligent Tutoring Systems*. Springer, Cham, 2014.
32. Hawkins, William, Neil Heffernan, and Ryan SJd Baker. "Which is more responsible for boredom in intelligent tutoring systems: students (trait) or problems (state)?." *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*. IEEE, 2013.
33. Hawkins, William, et al. "Extending the assistance model: Analyzing the use of assistance over time." *Educational Data Mining 2013*. 2013.
34. San Pedro, Maria Ofelia Z., et al. "Towards an understanding of affect and knowledge from student interaction with an intelligent tutoring system." *International Conference on Artificial Intelligence in Education*. Springer, Berlin, Heidelberg, 2013.
35. Wang, Yutao, and Neil Heffernan. "Extending knowledge tracing to allow partial credit: Using continuous versus binary nodes." *International conference on artificial intelligence in education*. Springer, Berlin, Heidelberg, 2013.
36. Kehrer, Paul, Kim Kelly, and Neil Heffernan. "Does immediate feedback while doing homework improve learning?." *The Twenty-Sixth International FLAIRS Conference*. 2013.
37. Kelly, Kim, et al. "Adding teacher-created motivational video to an ITS." *Proceedings of 26th Florida Artificial Intelligence Research Society Conference*. 2013.
38. Pardos, Zachary A., and Neil T. Heffernan. "Tutor modeling vs. student modeling." *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*. 2012.
39. Qiu, Yumeng, Zachary A. Pardos, and Neil T. Heffernan. "Towards Data Driven Model Improvement." *Twenty-Fifth International FLAIRS Conference*. 2012.
40. Pardos, Zachary A., et al. "Clustered knowledge tracing." *International Conference on Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 2012.
41. Wang, Yutao, and Neil T. Heffernan. "The student skill model." *International Conference on Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 2012.

42. Gong, Yue, Joseph E. Beck, and Neil T. Heffernan. "WEBsistments: enabling an intelligent tutoring system to excel at explaining rather than coaching." *International Conference on Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 2012.
43. Wang, Yutao, and Neil T. Heffernan. "Leveraging First Response Time into the Knowledge Tracing Model." *International Educational Data Mining Society* (2012).
44. Qiu, Yumeng, et al. "Does Time Matter? Modeling the Effect of Time with Bayesian Knowledge Tracing." *EDM*. 2011.
45. Nooraei, Bahador B., et al. "Less is More: Improving the Speed and Prediction Power of Knowledge Tracing by Using Less Data." *EDM*. 2011.
46. Pardos, Zachary A., and Neil T. Heffernan. "KT-IDEM: Introducing item difficulty to the knowledge tracing model." *International conference on user modeling, adaptation, and personalization*. Springer, Berlin, Heidelberg, 2011.
47. Singh, Ravi, et al. "Feedback during web-based homework: the role of hints." *International Conference on Artificial Intelligence in Education*. Springer, Berlin, Heidelberg, 2011.
48. Wang, Yutao, and Neil T. Heffernan. "The “Assistance” model: Leveraging how many hints and attempts a student needs." *Twenty-fourth international FLAIRS conference*. 2011.
49. Razzaq, Leena, and Neil T. Heffernan. "Hints: is it better to give or wait to be asked?." *International conference on intelligent tutoring systems*. Springer, Berlin, Heidelberg, 2010.
50. d Baker, Ryan SJ, Adam B. Goldstein, and Neil T. Heffernan. "Detecting the moment of learning." *International Conference on Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 2010.
52. Zhao, Siyuan, and Neil Heffernan. "Estimating Individual Treatment Effect from Educational Studies with Residual Counterfactual Networks." *International Educational Data Mining Society* (2017).
53. Zhang, Liang, et al. "Incorporating rich features into deep knowledge tracing." *Proceedings of the fourth (2017) ACM conference on learning@ scale*. 2017.
54. Ritter, Steven, et al. "Reducing the Knowledge Tracing Space." *International Working Group on Educational Data Mining* (2009).
55. Khajah, Mohammad, et al. "Integrating latent-factor and knowledge-tracing models to predict individual differences in learning." *Educational Data Mining 2014*. 2014.
56. Kochmar, Ekaterina, et al. "Automated Data-Driven Generation of Personalized Pedagogical Interventions in Intelligent Tutoring Systems." *International Journal of Artificial Intelligence in Education* (2021): 1-27.
57. Brigui-Chtioui, Imène, Philippe Caillou, and Elsa Negre. "Intelligent digital learning: Agent-based recommender system." *Proceedings of the 9th International Conference on Machine Learning and Computing*. 2017.
58. Ramesh, Arti, et al. "Learning latent engagement patterns of students in online courses." *Twenty-Eighth AAAI Conference on Artificial Intelligence*. 2014.
59. Zhao, Jinjin, et al. "Interpretable personalized knowledge tracing and next learning activity recommendation." *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 2020.
61. Kardian, Kevin, and Neil T. Heffernan III. "Knowledge Engineering for Intelligent Tutoring Systems: Using machine learning assistance to help humans tag questions to skills based upon the words in the questions."
62. Rosé, Carolyn P., et al. "A hybrid language understanding approach for robust selection of tutoring goals." *International Conference on Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 2002.
63. Kausar, Samina, et al. "Integration of data mining clustering approach in the personalized E-learning system." *IEEE Access* 6 (2018): 72724-72734.
64. Zhou, Yuwen, et al. "Personalized learning full-path recommendation model based on LSTM neural networks." *Information Sciences* 444 (2018): 135-152.

65. Kardan, Ahmad A., Molood Ale Ebrahim, and Maryam Bahojb Imani. "A new personalized learning path generation method: Aco-map." *Indian Journal of Scientific Research* 5.1 (2014): 17-24.
66. Jyothi, Nava, et al. "A recommender system assisting instructor in building learning path for personalized learning system." *2012 IEEE Fourth International Conference on Technology for Education*. IEEE, 2012.
67. Scruggs, Richard, Ryan S. Baker, and Bruce M. McLaren. "Extending Deep Knowledge Tracing: Inferring Interpretable Knowledge and Predicting Post-System Performance." *arXiv preprint arXiv:1910.12597* (2019).
68. Kurdi, M. Zakaria. "Text Complexity Classification Based on Linguistic Information: Application to Intelligent Tutoring of ESL." *arXiv preprint arXiv:2001.01863* (2020).
69. Barrón-Estrada, María Lucía, et al. "Sentiment analysis in an affective intelligent tutoring system." *2017 IEEE 17th international conference on advanced learning technologies (ICALT)*. IEEE, 2017.
70. Karlovčec, Mario, Mariheida Córdova-Sánchez, and Zachary A. Pardos. "Knowledge component suggestion for untagged content in an intelligent tutoring system." *International Conference on Intelligent Tutoring Systems*. Springer, Berlin, Heidelberg, 2012.
71. Rosé, Carolyn, et al. "A hybrid text classification approach for analysis of student essays." *Proceedings of the HLT-NAACL 03 workshop on building educational applications using natural language processing*. 2003.
72. Chuan, Ching-Hua, et al. "An Intelligent Tutoring System for Argument-Making in Higher Education: A Pilot Study." *2014 13th International Conference on Machine Learning and Applications*. IEEE, 2014.
73. Rosé, Carolyn, and Kurt VanLehn. "An evaluation of a hybrid language understanding approach for robust selection of tutoring goals." *International Journal of Artificial Intelligence in Education* 15.4 (2005): 325-355.
74. Walker, Erin, et al. "Using problem-solving context to assess help quality in computer-mediated peer tutoring." *International conference on intelligent tutoring systems*. Springer, Berlin, Heidelberg, 2010.
75. Vajjala, Sowmya, and Detmar Meurers. "Assessing the relative reading level of sentence pairs for text simplification." *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 2014.
76. Nielsen, Rodney D., Wayne Ward, and James H. Martin. "Recognizing entailment in intelligent tutoring systems." *Natural Language Engineering* 15.4 (2009): 479-501.
77. Walker, Erin, Nikol Rummel, and Kenneth R. Koedinger. "Using automated dialog analysis to assess peer tutoring and trigger effective support." *International conference on artificial intelligence in education*. Springer, Berlin, Heidelberg, 2011.
80. Shen, Jia Tracy, et al. "Classifying Math Knowledge Components via Task-Adaptive Pre-Trained BERT." *International Conference on Artificial Intelligence in Education*. Springer, Cham, 2021.
81. Minn, Sein, et al. "Dynamic student classification on memory networks for knowledge tracing." (2019): 163.
82. Huang, Yun, Julio Daniel Guerra Hollstein, and Peter Brusilovsky. "Modeling Skill Combination Patterns for Deeper Knowledge Tracing." *UMAP (Extended Proceedings)*. 2016.
83. Moore, Steven, Huy A. Nguyen, and John Stamper. "Towards Crowdsourcing the Identification of Knowledge Components." *Proceedings of the Seventh ACM Conference on Learning@ Scale*. 2020.
84. Wang, Tiejun, et al. "Crowdsourcing-based framework for teaching quality evaluation and feedback using linguistic 2-tuple." *Cmc-computers Mater. Continua* 57.1 (2018): 81-96.

85. Abdi, Solmaz, Hassan Khosravi, and Shazia Sadiq. "Modelling learners in crowdsourcing educational systems." *International Conference on Artificial Intelligence in Education*. Springer, Cham, 2020.
86. Ye, Xinfeng, and Sathiamoorthy Manoharan. "Providing automated grading and personalized feedback." *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*. 2019.
87. Whitehill, Jacob, and Margo Seltzer. "A Crowdsourcing Approach to Collecting Tutorial Videos--Toward Personalized Learning-at-Scale." *Proceedings of the Fourth (2017) ACM Conference on Learning@ Scale*. 2017.
88. Liu, Ching, Juho Kim, and Hao-Chuan Wang. "ConceptScape: Collaborative concept mapping for video learning." *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018.
89. Liu, Ching, Juho Kim, and Hao-Chuan Wang. "ConceptScape: Collaborative concept mapping for video learning." *Proceedings of the 2018 CHI conference on human factors in computing systems*. 2018.
90. Adjei, Seth, et al. "Clustering students in assistments: exploring system-and school-level traits to advance personalization." *The 10th International Conference on Educational Data Mining*. 2017.
91. Bahar, Hüseyin Hüsnü. "The effects of gender, perceived social support and sociometric status on academic success." *Procedia-Social and Behavioral Sciences* 2.2 (2010): 3801-3805.
92. Oramas-Bustillos, Raúl, et al. "A corpus for sentiment analysis and emotion recognition for a learning environment." *2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)*. IEEE, 2018.
93. Gong, Yue, and Joseph E. Beck. "Towards detecting wheel-spinning: Future failure in mastery learning." *Proceedings of the second (2015) ACM conference on learning@ scale*. 2015.
94. Beck, Joseph, and Ma Mercedes T. Rodrigo. "Understanding wheel spinning in the context of affective factors." *International conference on intelligent tutoring systems*. Springer, Cham, 2014.
95. Botelho, Anthony F., et al. "Developing early detectors of student attrition and wheel spinning using deep learning." *IEEE Transactions on Learning Technologies* 12.2 (2019): 158-170.
96. Kai, Shimin, et al. "Decision tree modeling of wheel-spinning and productive persistence in skill builders." *Journal of Educational Data Mining* 10.1 (2018): 36-71.
97. Wang, Yeyu, Shimin Kai, and Ryan Shaun Baker. "Early Detection of Wheel-Spinning in ASSISTments." *International Conference on Artificial Intelligence in Education*. Springer, Cham, 2020.
98. Chang, Wen-Hsuan, Jie-Chi Yang, and Yu-Chieh Wu. "A keyword-based video summarization learning platform with multimodal surrogates." *2011 IEEE 11th International Conference on Advanced Learning Technologies*. IEEE, 2011.
100. Battle, Juan, and Michael Lewis. "The increasing significance of class: The relative effects of race and socioeconomic status on academic achievement." *Journal of poverty* 6.2 (2002): 21-35.
102. Kupczynski, Lori, et al. "The Relationship between Gender and Academic Success Online." *Journal of Educators Online* 11.1 (2014): n1.