Creating Features for Personalized Tutoring in ASSISTments

An Interactive Qualifying Project (IQP) Report
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements
for the Degree of Bachelor of Science in

Computer Science

By:

Matthew Hendrickson
Caleb Scopetski

Project Advisor:

Professor Neil Heffernan

Sponsored By:

ASSISTments

Date: July 2022

Abstract

This IQP was an eclectic development of various features for the ASSISTments online learning platform. We designed features to identify student gaming behaviors, used trigram matching to determine similarity between hints and explanations of problems, concatenated data to find streaks of correctly answered problems, and clustered Common Core descriptions based on embeddings from MathBERT. We also simulated using deep Bayesian bandits to recommend content in the form of supports to struggling students. Our models were able to predict whether or not a student would get the next problem correct more frequently than random using an epsilon-greedy (RMS) model. All features were completed successfully and integrated into the ASSISTments Automatic Personalized Learning Service (APLS). These results all had significant findings to be expanded upon in further research.

Acknowledgements

A very special thanks to...

1. Professor Neil Heffernan for the opportunity to work for ASSISTments and gain invaluable experience.
2. Ethan Prihar for his guidance and help in working with significant programming challenges.

Contents

List of Tables

List of Figures

1 Introduction

ASSISTments is a popular math tutoring software developed to help K-12 students and is utilized in many schools across the world. Our goal with this project was to develop features and models to personalize student learning experiences and improve learning rates.

1.1 TeacherASSIST

Previous work on ASSISTments has already explored personalizing student learning experiences. One such example was the exploration of TeacherASSIST[1], a feature of ASSISTments that redistributes personalized crowd sourced tutoring to other students on the ASSISTments platform. This study aimed to discover the effectiveness of this crowd sourced tutoring, and to see if any further personalization of tutoring could be done. The study ultimately concluded that tutoring provided through TeacherASSIST had a positive impact on students' learning[1]. The features used in the study's models did not capture 100% of the variance in problems and students[1], so in this project we aimed to create features that could increase the reliability of predicting next problem correctness (NPC).

1.2 Automatic Personalized Learning Service (APLS)

Creating more personalized learning was also previously explored through research of APLS[2]. The study tested the effectiveness of multiple multi-armed bandit algorithms at recommending the best tutoring support for a student. The study concluded that Beta-Bernoulli Thompson Sampling performed only slightly better than random, and that Decision Tree Thompson Sampling performed the best out of the bandit-algorithms explored[2]. In this project we aimed to expand on the research of effective bandit algorithms for recommending personalized tutoring to students.

1.3 Features

Our goal with developing features was to improve the reliability of predicting NPC and to better understand what effects student learning. By raising the accuracy of predicting NPC, we hoped that future work could use it to evaluate the effectiveness of new personalized learning methods.

1.3.1 Detecting Gaming Behaviors

Gaming the system is a term used when a student attempts to trick a system into progressing without actually putting in the cognitive effort to solve the assigned problems[3]. We set out to create a detector for some of these behaviors based on prior research.

1.3.2 Wheel-spinning/Answer Streaks

Wheel-spinning is a student behavior that occurs when a student has spent sufficient time practicing a skill, but has made minimal progress in mastering that skill[4]. The correct/incorrect answer streak feature was developed as an offshoot of a wheel-spinning detector. It was created as a simple statistic for estimating student success.

1.3.3 Similarity Detection

Using strict trigram matching from PostgreSQL, we wished to find the similarity between hints, explanations, and questions in the ASSISTments database. Similarity between these has numerous applications for further research. There may exist correlations between similar questions/explanations and NPC.

1.3.4 Common Core MathBERT Clustering

The Common Core State Standards Initiative was created in 2010 to ensure that students had a fundamental understanding of mathematics and English language arts at the end of each grade[5]. The Common Core has a comprehensive website detailing the skills of many of these concepts in mathematics. Combining MathBERT, a large language model, we set out to embed and cluster around 450 mathematical Common Core skill descriptions using several different clustering methods.

1.4 Simulating the use of Contextual Bandits in ASSISTments

Our goal with evaluating deep learning models was to simulate using contextual bandit algorithms for recommending tutoring support. By identifying and applying this support, student learning rates would be improved as a result of the personalized learning environment.

Determining the NPC of a student is vital in discovering what tutor supports help or harm them and has been an area of interest for ASSISTments. Our goal was to turn ASSISTments data into a contextual bandit problem and then apply several popular bandit algorithms from past research to maximize NPC among students using tutor supports.

2 Background

2.1 Detecting Gaming Behaviors

2.1.1 Defining Gaming the System

Gaming the system is a form of student disengagement that is seen often in online tutoring systems. It is defined as “attempting to succeed in the environment by exploiting properties of the system rather than by learning the material and trying to use that knowledge to answer correctly”[3]. Gaming the system has been shown to have an immediately harmful impact on student learning outcomes[6]. Therefore, being able to detect when students game the system and prevent them from continuing to do so quickly and efficiently is imperative for online tutoring systems.

2.1.2 Methods For Detection

There are many well-researched methods for detection of gaming the system, with the most widely used being forms of data mining/machine learning and knowledge engineering[7]. Data mining/machine learning methods require large datasets pre-labeled by experts as gaming or non-gaming to train. Once trained, they algorithmically identify relationships between student actions/behaviours and gaming the system. This avoids the need to have explicit knowledge on the types of student behaviour that indicate gaming. Knowledge engineering models are created by designing a set of rules that replicate known patterns of gaming. These models do not require large pre-labeled datasets, since the rules for the model are elicited directly from experts. Since the knowledge engineering models are developed from known behaviours/patterns that indicate gaming, they are more applicable to other tutoring systems than machine learning models[8].

2.2 Wheel-spinning/Answer Streaks

2.2.1 Defining Wheel-spinning

Wheel-spinning is a student behavior that occurs when a student has spent significant time practicing a skill, but has made minimal progress in mastering that skill[4]. Research has shown that wheel-spinning is related to a decrease in flow and an increase in confusion in students[9], forming an unproductive work environment. Since wheel-spinning is cognitive in nature, students will often need outside intervention in order to break out of the cycle. This is why it is important for tutoring systems like ASSISTments to identify wheel-spinning quickly, so that intervention methods may be administered before students become too frustrated or lose hope.

2.2.2 Methods for Detection

In general, wheel-spinning revolves around skill mastery. One simplistic approach to mastery is to assume a student has mastered a skill once a student has correctly answered 3 questions in a row[4]. In a study previously done on ASSISTments data, a student was considered to be wheel-spinning upon failing to master a skill within 10 problems[4]. These rules define the bounds for determining whether or not a student is wheel-spinning.

2.3 Similarity Detector

Trigram matching is a method of splitting two strings into every possible in-place group of three characters, and then determining how many matches between them exist. In the ASSISTments database, there are records of every problem and explanation. We can use trigram matching to determine the similarity between the problems and the hints/explanations written for them. We can then use this similarity metric to determine trends in student learning.

2.4 Common Core MathBERT Clustering

BERT (Bidirectional Encoder Representations from Transformers) models have been widely praised for their ability to understand and process NLP tasks and input, and have been used in numerous studies[10]. MathBERT is a BERT model trained again on mathematical texts, and has many potential uses for ASSISTments. In ASSISTments, each problem is tagged with a common core skill like the following:

Compare two fractions with different numerators and different denominators, e.g., by creating common denominators or numerators, or by comparing to a benchmark fraction such as 1/2.[11]

MathBERT is able to generate embeddings for these skills, which could then be clustered to determine the main focuses of the Common Core for math education. Problems arise when attempting to determine which clustering method categorizes these embeddings correctly, since MathBERT generates a significant amount of dimensions to cluster on.

2.5 Simulating the use of Contextual Bandits in ASSISTments

A contextual bandit is a derivative of a popular Computer Science problem known as the multi-armed bandit. This problem details several ”one-armed bandits” (slang for slot machines) where the gambler has a finite amount of money to spend, and needs to figure out which machine has the highest payout[12]. There is a trade off between exploration and exploitation that must be balanced: If the gambler decides to spend all his money on one machine, there might be others that payout better. If he spends it all equally on the machines, he does not maximize the profit of the one machine that pays the most. A contextual bandit problem is similar, except that there is some outside context affecting the payout of the machines. Perhaps the machines pay out differently on different days, or the owner decides to cut down on the payout of the machines if there are more people in the casino. The goal of each of these problems is to discover the trends of the contexts and then use them to exploit the bandits for as much money as possible.

Deep Bayesian Bandits Showdown[13] details an analysis of several different bandit algorithms in an attempt to find which performed best on several distinct and popular datasets. Previous research on contextual bandits in personalized learning with ASSISTments data using Beta-Bernoulli Thompson Sampling or Decision Tree Thompson Sampling has only performed slightly better than random[2].

3 Methodology

3.1 Detecting Gaming Behaviors

3.1.1 Restrictions

When developing our gaming detector, there were some restrictions we had to be wary of. The current APLS uses bandit algorithims that store thousands of actions in real time. As such, our gaming detector needed to be computationally low cost in order to finish computing in a reasonable time frame. Another restriction was the lack of time and resources to produce a dataset coded as gaming or non-gaming by an expert. Without a large pre-labeled dataset, we would be unable to train any machine learning models to detect gaming. Considering the restrictions, we decided that a form of knowledge engineering model would be ideal, as they do not rely on pre-labeled datasets and are more efficient than machine learning models.

3.1.2 Eliciting the Knowledge Engineered Rule Set

The biggest challenge with knowledge engineering models is defining the set of rules elicited from an expert that the model will follow. Through research we were able to obtain a knowledge engineered set of rules to detect gaming[14]. This model first interprets the type of a student's action, then identifies patterns in sequences of actions that display gaming. Although the model was originally developed for Cognitive Tutor Algebra, a math tutoring service, it was proven to be able to transfer and perform reasonably well on the ASSISTments platform[15].

3.1.3 Creating a Simplified Model

Upon obtaining a reasonable set of knowledge engineered rules for detecting gaming on ASSISTments data, we had to create a simplified model that would be able to deploy and run on the 150,000 student action logs within a short time frame. This meant that we would have to sacrifice accuracy for speed by reducing the amount of rules the model would follow. Upon further analysis, we found that a common attribute of the rules was time taken between actions. We decided that time between actions would be the major rule our model would follow in order to detect gaming. Based on the rule set, we concluded that an average time of less than or equal to 3 seconds between wrong answer and tutoring request actions would be flagged as gaming. This was derived from the rules on guessing and requesting help from the rule set. The second major rule for our model was whether or not the student got the problem correct. In order to simplify our model, we decided that the model would only flag students as gaming if they incorrectly answered the question. In summary, the model considers a student to have gamed the system on a particular problem if the average time between wrong answer and tutoring request actions is less than or equal to 3 seconds *and* their response was incorrect.

3.2 Wheel-spinning/Answer Streaks

3.2.1 Transitioning to Answer Streaks

Our research led to the conclusion that wheel-spinning revolves around correct and incorrect answer streaks. We decided to transition from making a detector for wheel-spinning, to making a student answer streak tracker that could be used more generally for future features. An answer streak is defined as getting the same result (correct or incorrect) for multiple problems consecutively. For example, a student that got two problems correct in a row would have a correct answer streak of two.

3.2.2 Calculating Answer Streaks

In order to calculate answer streaks, we retrieved all student responses and ordered them by time submitted and counted consecutive correct and incorrect answers. These counts were then combined with the students' current correct and incorrect median answer streaks to create a running median.

3.3 Similarity Detector

Creating this feature had several issues to overcome, specifically in parsing the data for NLP applications. All of the data we were trying to use contained HTML body tags, extra spaces, newline characters, and other noise. Filtering was accomplished with several distinct RegEx equations. These equations could be concatenated together, but for simplicity's sake they were separated to make understanding and iterating upon them simpler. The code used to filter the HTML is shown below:

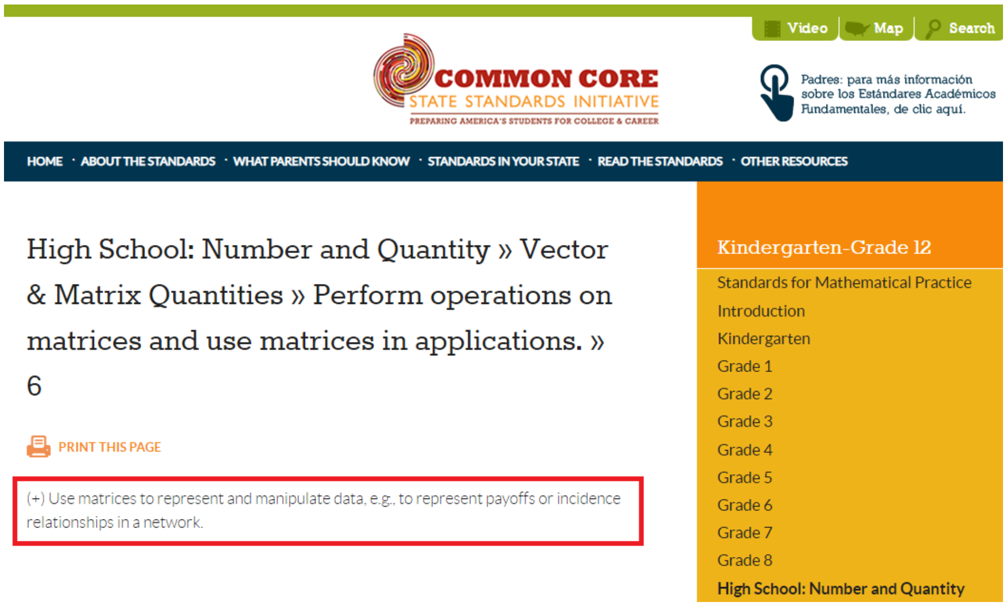
```
create or replace function filter_string(unfiltered_string varchar)
returns varchar
language plpgsql
as
$$
declare
   filtered_string varchar;
begin
     filtered_string := trim(both from
     regexp_replace(
     regexp_replace(
     regexp_replace(
     regexp_replace(
     unfiltered_string, E'<.*?>', '', 'g'),
    E'&nbsp;', '␣', 'g'),
    E'[\\n\\r]+', '␣', 'g'),
    E'[␣]+', '␣'));
   return filtered_string;
end;
$$;
```

Figure 1: PostgreSQL Similarity Detector Code

Each RegEx had a specified goal to filter a specific part of HTML. First, we remove all data within the less than/greater than signs (⪌), then all tab/newline characters (&nbsp, \n), and finally remove extra spacing left after the previous operations. We then enter the filtered text into strict word similarity trigram matching from PostgreSQL[16]. The result was a fast, effective, and reliable method to determine the text similarity between questions and their tutoring strategies.

3.4 Common Core MathBERT Clustering

Using similar RegEx code described in the similarity detector, as well as some Python code for determining the location of the skill in the HTML webpage, we were able to extract Common Core skillcode descriptions from the Common Core webpage shown in Figure 2.


Video
Map
Search
COMMON CORE
STATE STANDARDS INITIATIVE
PREPARING AMERICA'S STUDENTS FOR COLLEGE & CAREER
Padres: para más información sobre los Estándares Académicos Fundamentales, de clic aquí.
HOME · ABOUT THE STANDARDS · WHAT PARENTS SHOULD KNOW · STANDARDS IN YOUR STATE · READ THE STANDARDS · OTHER RESOURCES
PRINT THIS PAGE
Kindergarten-Grade 12
Standards for Mathematical Practice
Introduction
Kindergarten
Grade 1
Grade 2
Grade 3
Grade 4
Grade 5
Grade 6
Grade 7
Grade 8
High School: Number and Quantity

High School: Number and Quantity » Vector & Matrix Quantities » Perform operations on matrices and use matrices in applications. » 6

(+) Use matrices to represent and manipulate data, e.g., to represent payoffs or incidence relationships in a network.

Figure 2: Common Core Skill (with the desired text outlined in red)

We could then use the body tags in HTML to grab the desired text and then discard the rest. Once filtered, we then ran the skills through MathBERT, which embedded the semantic information of each skill in an array with 3072 dimensions. Clustering was done in several different ways, each with varying levels of success. PCA (Principal Component Analysis) was done due to the high number of dimensions, and led to significantly better results both in clustering and further analysis. The links, filtered skills, and PCA data can be found in the appendix.

3.5 Simulating the use of Contextual Bandits in ASSISTments

We decided to run each algorithm from the Deep Bayesian Bandits Showdown paper through 5 trials, each with 10,000 randomly selected and normalized data points. The goal of the algorithms was to correctly identify when a tutor strategy would be helpful. When the algorithm correctly identified a tutor strategy as helpful it would be given a reward of 1. When the algorithm incorrectly identified a tutor strategy as helpful it would be given a reward of -1. When the algorithm identified a tutor strategy as unhelpful it would be given a reward of 0. We also studied variations with these rewards in powers of 10, but found that significant gaming occurred in these models as they would simply maximize their scores by abusing higher rewards where the punishments were negligible, leading to unusable results for our purposes.

Tutor strategies helped a student answer the next problem correctly roughly 33% of the time. Therefore, we can calculate the optimal choice by assuming our algorithm guess correctly in every instance (around 3333 times) and chooses to abstain in every other instance where supports will be unhelpful. This can be represented with equation 1:

$$10000 * ((\frac{2}{3} * 0) + (\frac{1}{3} * 1)) = 3333 \tag{1}$$

We can calculate random choice by assuming half the time the model chooses no supports and half of the time it will roll the dice: 33% of the time the reward will be positive and 67% of the time the reward will be negative. This can be calculated using equation 2:

$$10000((\frac{1}{2} * 0) + \frac{1}{2}(1 * \frac{1}{3} + -1 * \frac{2}{3})) = -1667 \tag{2}$$

Our models need to compete against these two scores to determine if contextual bandits are a viable option for predicting NPC.

4 Findings

4.1 Detecting Gaming Behaviors

After running the gaming detector on around 1.03 million different users and assigning users an average gaming frequency score ranging from 0 (never games) and 1, (games on every problem) we obtained the following results:

Summary of Average Gaming Frequency	
Variable	**Value**
count:	1,035,000
mean:	0.00307
std:	0.01667
min:	0
25%	0
50%	0
75%	0
max:	0.8

Figure 3: Summary of the Distribution of Mean Gaming Frequency

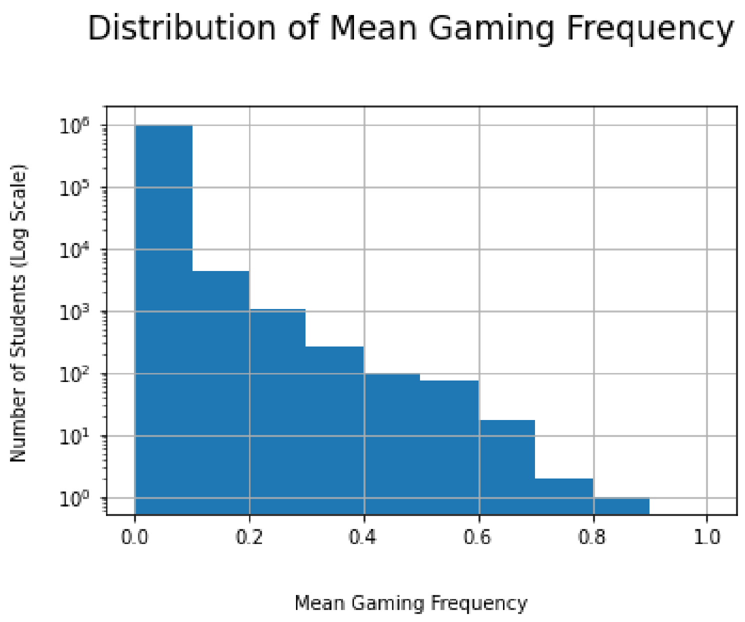


Figure 4: Mean Gaming Frequency Distribution

The data 3,4 suggests that users up to the 75th percentile have an average gaming frequency of 0; 100% of all gaming is done by less than 25% of students on the ASSISTments platform. This conclusion is supported by the mean and standard deviation. The mean is extremely low at 0.003 and the low standard deviation suggests that the majority of users gaming scores are clustered around 0.003.

4.2 Wheel-spinning/Answer Streaks

After running the answer streak tracker on around 1.03 million different users and calculating the median right and wrong answer streaks for each user we obtained the following results:

Summary of Median Answer Streaks		
Var	**Correct**	**Incorrect**
count:	1,035,000	1,035,000
mean:	12.0237	4.1763
std:	19.5762	4.7423
min:	0.0	0.0
25%	2.0	1.50
50%	5.5	2.6363
75%	13.750	5.0
max:	635.50	155.0

Figure 5: Summary of the Distribution of Median Correct and Incorrect Answer Streaks

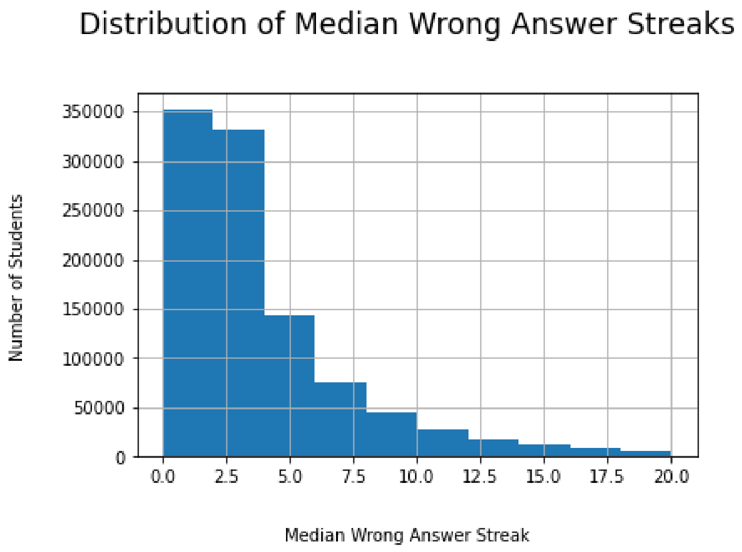


Figure 6: Median Incorrect Answer Streak Distribution

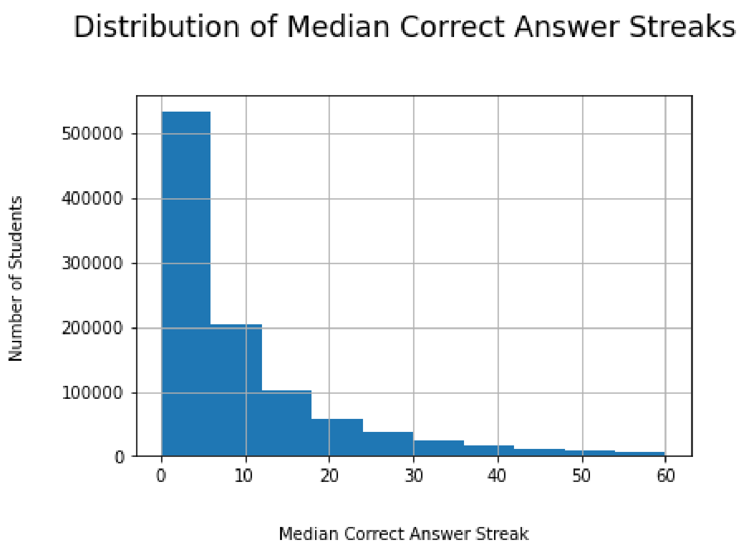


Figure 7: Median Correct Answer Streak Distribution

As shown in Figures 5,6,7 the mean for the correct answer streaks is much higher than the mean for the incorrect answer streaks. This could relate to the idea of skill mastery, showing that once 3 problems are answered correctly in a row, a student has mastered a skill[4]. The standard deviation for correct answer streaks is also much higher than incorrect answer streaks suggesting that there is a greater variety in student correct answer streak length. Both incorrect and correct answer streaks are right skewed, however the correct answer streaks contain many more outliers.

4.3 Similarity Detector

There was an average similarity of 0.22 across tutor strategies to the problems they were created for. The majority of similarity scores were less than 0.5, with only a small percentage reaching 1.0 similarity. These outliers can likely be attributed to teachers copying and pasting content from a problem into the tutoring for the problem when creating problem sets in ASSISTments.

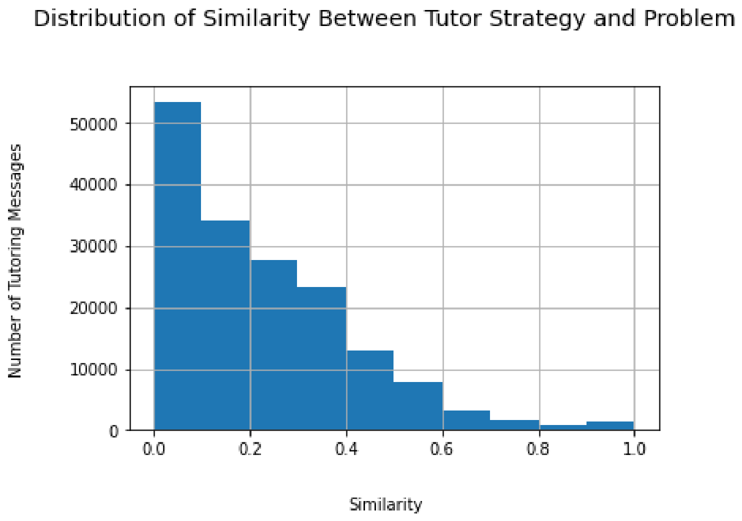


Figure 8: Similarity Between Tutor Strategies and Their Problem

There are also significant correlations between other tutor strategy features and similarity. Running tutor data through RapidMiner Studio attempting to predict text similarity showed that Deep Learning models and Gradient Boosted Trees were the most successful in determining trends with a correlation of 0.718 and 0.732 respectively. However, there was significant relative error in both models of 41%. Number of messages, containing a question, and color use was positively correlated with text similarity. Hints containing a video, images, and different font usages were negatively correlated with text similarity. Curiously, text length had no effect on similarity whatsoever.

4.4 Common Core MathBERT Clustering

4.4.1 K-means

K-means clustering was the least successful clustering algorithm applied to this dataset, with high levels of variance in both the Calinski-Harabasz score, Davies-Bouldin score, and the silhouette score, even after filtering 95% of the non-variant data points.

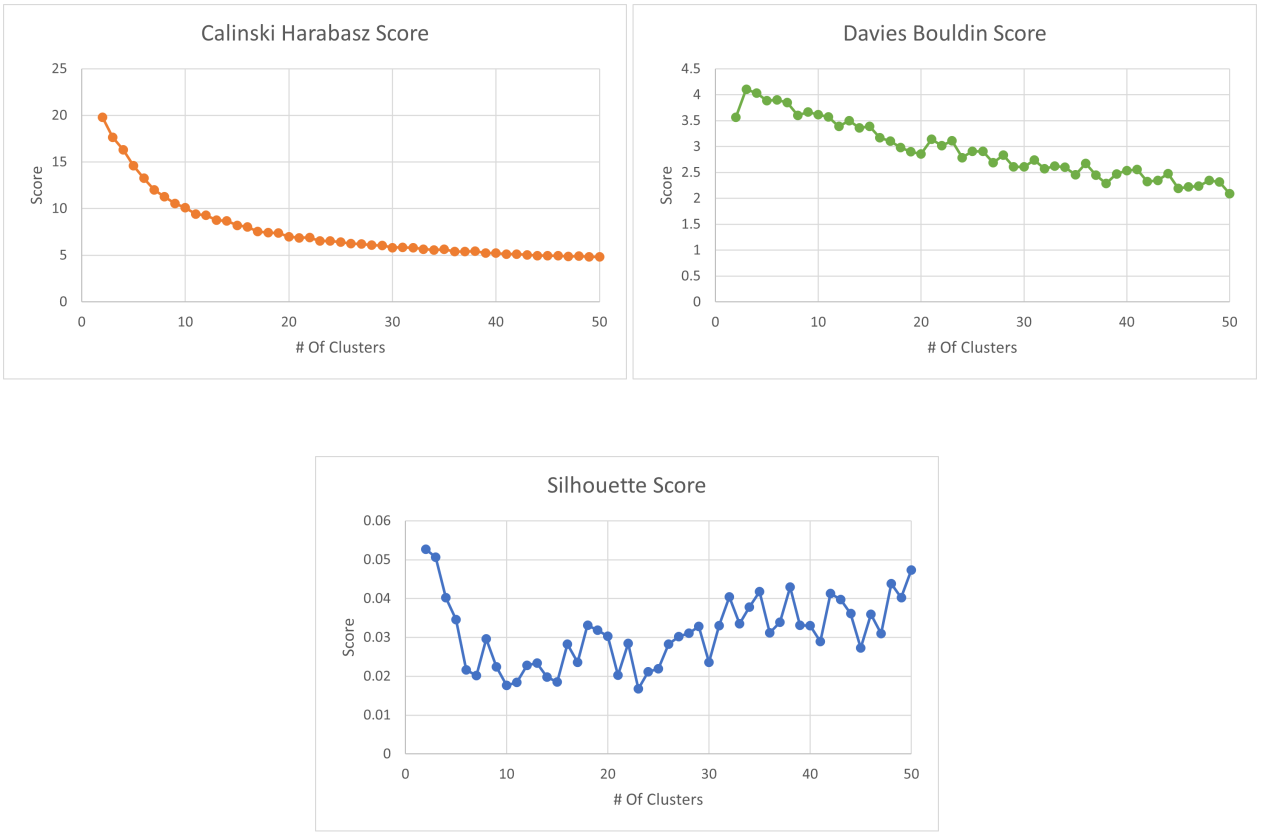


Figure 9: Scores of Common Clustering Metrics

These results led us to believe that K-means was not a viable method of clustering MathBERT embeddings given the erratic silhouette score. The failure of this method of clustering led to the idea to try hierarchical clustering instead, operating under the hypothesis that a single skill could belong to multiple clusters, and that in essence all of these skills fall under one large cluster of mathematical terms/phrases.

4.4.2 Hierarchical-Linkage Clustering

Linkage clustering showed significantly more appealing results, with clusters being much more qualitatively reasonable.

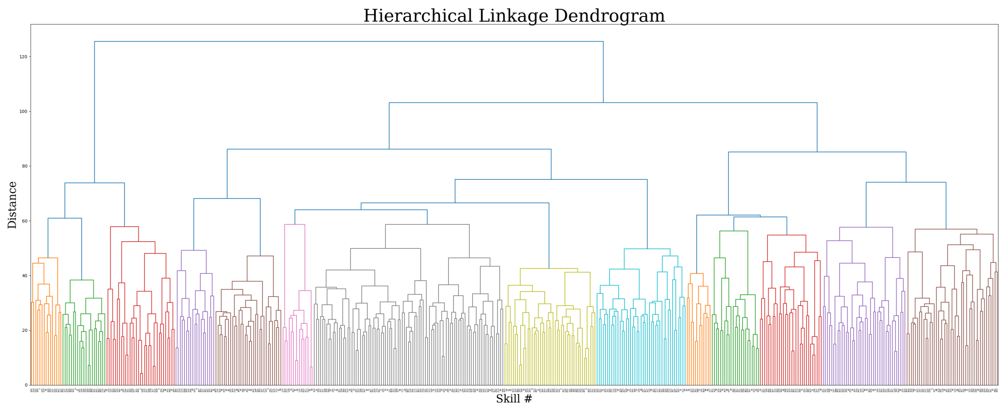


Figure 10: Hierarchical-Linkage Clustering of MathBERT Embeddings

We can clearly see that there are similarities between the two below statements, which both belong to the same node of the dendrogram:

"Understand that a two-dimensional figure is similar to another if the second can be obtained from the first by a sequence of rotations, reflections, translations, and dilations; given two similar two-dimensional figures, describe a sequence that exhibits the similarity between them."[17] (119)

"Understand that a two-dimensional figure is congruent to another if the second can be obtained from the first by a sequence of rotations, reflections, and translations; given two congruent figures, describe a sequence that exhibits the congruence between them."[18] (190)

We can also observe similarities between the above statements and other statements from leaf nodes belonging to the same parent node:

"Given a geometric figure and a rotation, reflection, or translation, draw the transformed figure using, e.g., graph paper, tracing paper, or geometry software. Specify a sequence of transformations that will carry a given figure onto another."[19] (397)

However, we can observe significantly less similarity between the above statements and other statements from leaf nodes of different roots:

Use ratio reasoning to convert measurement units; manipulate and transform units appropriately when multiplying or dividing quantities.[20] (66)

Linkage clustering provides results for categorizing MathBERT embeddings for Common Core descriptions, and these results seem to suggest that NLP categorization is best done hierarchically due to the heavy amount of semantic overlap in sentences.

4.4.3 Agglomerative-Ward

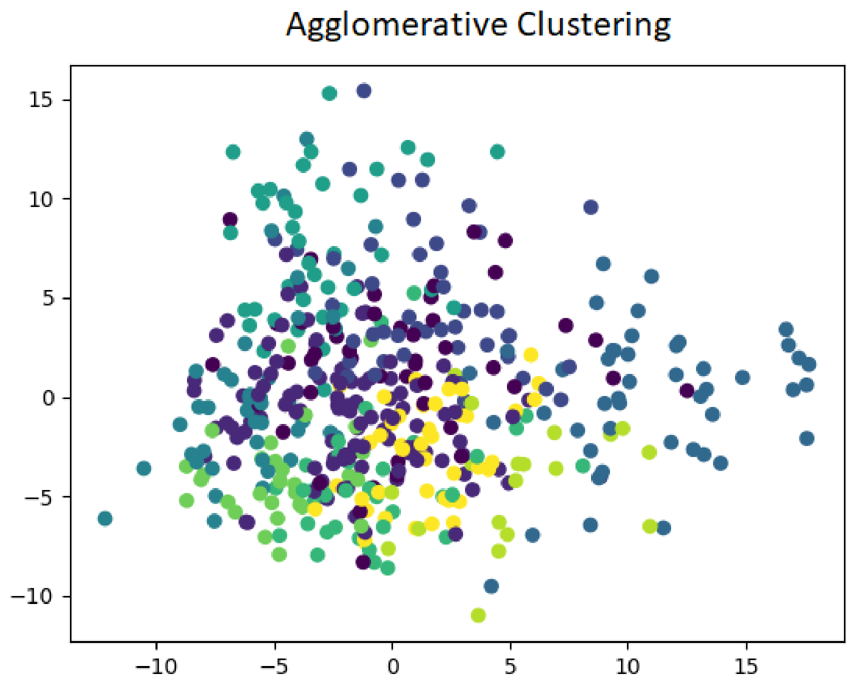


Figure 11: Agglomerative Clustering of MathBERT Embeddings

Agglomerative clustering also showed more visually appealing results than KMeans. We can clearly observe and quantify specific clusters even though several thousand contextual dimensions are missing from our field of vision.

4.5 Simulating the use of Contextual Bandits in ASSISTments

The majority of the Bayesian bandits seem to perform better than random chance, with RMS (epsilon greedy) performing the best, predicting 92% more effectively than random chance. This is significantly better than current research done with contextual bandits on ASSISTments data, which only showed marginal improvements over random chance. [2]

Bandit Performance						
Bandit	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5	Sum
Optimal	3295	3376	3417	3325	3379	16792
RMS	-117	-166	-115	-164	**-109**	-671
Dropout	**-99**	-144	-106	-222	-176	-747
LinFullPost	**-182**	-212	-215	-213	-197	-1019
BootRMS	**-152**	-284	-182	-236	-243	-1097
NeuralLinear2	**-207**	-267	-214	-257	-232	-1177
BBB	-269	**-86**	-120	-168	-547	-1190
NeuralLinear	-246	-248	**-204**	-239	-268	-1205
fixed1	-837	**-761**	-798	-882	-820	-4098
BBAlphaDiv	-1367	-1265	**-1105**	-1412	-1359	-6508
Uniform Sampling	-1730	-1616	**-1510**	-1648	-1621	-8125
ParamNoise	-1683	-1718	-1619	**-1606**	-1643	-8269
Random Choice	-1667	-1667	-1667	-1667	-1667	-8333
Uniform Sampling 2	-1788	-1718	**-1556**	-1715	-1675	-8452
MultitaskGP	**-1654**	-1668	-1694	-1744	-1708	-8468
fixed2	-2549	-2478	**-2298**	-2551	-2402	-12278

Figure 12: Deep Bayesian Bandits on ASSISTments Data

It should also be noted that most of these results are far from the optimal scores that could be achieved with perfect models. This is likely due to the limitations of contextual bandit models as well as the complicated dataset fed to the algorithms. There was only 881 rows out of 160 thousand that contained no empty data in each row. Normalizing the dataset and then filling these empty cells with zeros may have led to a significant source of error for our models. It should also be noted that there are many variables that we do not have access to that are crucial in determining NPC. Despite these setbacks, our models are still able to perform better on ASSISTments data than current research, leading us to postulate that epsilon-greedy models may perform better when significant randomness is present into a dataset.

5 Conclusion

5.1 Features

The gaming, similarity and streak detectors are currently added into the APLS and can now be used in further research to determine trends in student learning as well as NPC. There is also significant potential for improvement in these models.

5.2 Clustering MathBERT Embeddings

Our findings show that embeddings from BERT based models are clustered more effectively using hierarchical methods. K-means and other non-hierarchical clustering methods are not nearly as effective. Further research in this area will involve using these results to explore the discovered Common Core trends in ASSISTments. Other online math tutor services like Khan Academy make use of these Common Core skills[21], as such it may be beneficial to implement a similar feature for ASSISTments.

5.3 ASSISTments as a Contextual Bandit Problem

Simulating ASSISTments data using deep Bayesian bandits is best done using an epsilon-greedy (RMS) model and is significantly better than random chance. Since this model cannot detect close to an optimal level, it still needs improvement. Perhaps adding additional context (such as gaming, wheel-spinning, affect, etc...) may lead to more reliable results. Contextual bandits are potentially not be the optimal way to predict NPC, and other methods should be tested as well.

References

[1] E. Prihar, T. Patikorn, A. Botelho, A. Sales, and N. Heffernan, “Toward personalizing students’ education with crowdsourced tutoring,” in Proceedings of the Eighth ACM Conference on Learning @ Scale, L@S ’21, (New York, NY, USA), p. 37–45, Association for Computing Machinery, 2021.

[2] E. Prihar, A. Haim, A. Sales, and N. Heffernan, “Automatic interpretable personalized learning,” in Proceedings of the Ninth ACM Conference on Learning @ Scale, L@S ’22, (New York, NY, USA), p. 1–11, Association for Computing Machinery, 2022.

[3] R. S. Baker, A. T. Corbett, I. Roll, and K. R. Koedinger, “Developing a generalizable detector of when students game the system,” User Modeling and User-Adapted Interaction, vol. 18, no. 3, p. 287–314, 2008.

[4] Y. Gong and J. E. Beck, “Towards detecting wheel-spinning: Future failure in mastery learning,” in Proceedings of the Second (2015) ACM Conference on Learning @ Scale, L@S ’15, (New York, NY, USA), p. 67–74, Association for Computing Machinery, 2015.

[5] Common Core State Standards Initiative, “Frequently asked questions.” ”http://www.corestandards.org/about-the-standards/frequently-asked-questions/”. Online; accessed 30 July 2022.

[6] M. Cocea, A. Hershkovitz, and R. Baker, “The impact of off-task and gaming behaviors on learning: Immediate or aggregate,” in Frontiers in Artificial Intelligence and Applications, no. 1 in Frontiers in Artificial Intelligence and Applications, pp. 507–514, IOS Press, 2009.

[7] R. S. J. d. Baker, A. Mitrović, and M. Mathews, “Detecting gaming the system in constraint-based tutors,” in User Modeling, Adaptation, and Personalization (P. De Bra, A. Kobsa, and D. Chin, eds.), (Berlin, Heidelberg), pp. 267–278, Springer Berlin Heidelberg, 2010.

[8] L. Paquette and R. S. Baker, “Comparing machine learning to knowledge engineering for student behavior modeling: a case study in gaming the system,” Interactive Learning Environments, vol. 27, no. 5-6, pp. 585–597, 2019.

[9] J. Beck and M. M. T. Rodrigo, “Understanding wheel spinning in the context of affective factors,” in Intelligent Tutoring Systems (S. Trausan-Matu, K. E. Boyer, M. Crosby, and K. Panourgia, eds.), (Cham), pp. 162–167, Springer International Publishing, 2014.

[10] A. Rogers, O. Kovaleva, and A. Rumshisky, “A primer in BERTology: What we know about how BERT works,” Transactions of the Association for Computational Linguistics, vol. 8, pp. 842–866, 2020.

[11] “Grade 4 number; operations-fractionsâ^1 ” extend understanding of fraction equivalence and ordering. ” 2.” ”http://www.corestandards.org/Math/Content/4/NF/A/2/.

[12] R. Weber, “On the Gittins Index for Multiarmed Bandits,” The Annals of Applied Probability, vol. 2, no. 4, pp. 1024 – 1033, 1992.

[13] C. Riquelme, G. Tucker, and J. Snoek, “Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling,” in International Conference on Learning Representations, 2018.

[14] L. Paquette, A. M. J. B. de Carvalho, and R. Baker, “Towards understanding expert coding of student disengagement in online learning,” Cognitive Science, 2014.

[15] L. Paquette, R. S. Baker, A. de Carvalho, and J. Ocumpaugh, “Cross-system transfer of machine learned and knowledge engineered models of gaming the system,” in User Modeling, Adaptation and Personalization (F. Ricci, K. Bontcheva, O. Conlan, and S. Lawless, eds.), (Cham), pp. 183–194, Springer International Publishing, 2015.

[16] O. Bartunov, “F.33. pg_trgm.” ”https://www.postgresql.org/docs/current/pgtrgm.html”, 2022.

[17] “grade 8 - geometry - understand congruence and similarity using physical models, transparencies, or geometry software. - 4.” ”http://www.corestandards.org/Math/Content/8/G/A/4/”, 2022.

[18] “grade 8 - geometry - understand congruence and similarity using physical models, transparencies, or geometry software. - 2.” ”http://www.corestandards.org/Math/Content/8/G/A/2/”, 2022.

[19] “high school: geometry - congruence - experiment with transformations in the plane - 5 - 2.” ”http://www.corestandards.org/Math/Content/HSG/CO/A/5/”, 2022.

[20] “grade 6 - ratios & proportional relationships - understand ratio concepts and use ratio reasoning to solve problems. - 3 - d.” ”http://www.corestandards.org/Math/Content/6/RP/A/3/d/”, 2022.

[21] “Common core map.” ”https://www.khanacademy.org/commoncore/map”, 2022.

Appendices

A GitHub Repositories

- Deep Bayesian Bandit

B Datasets

- Common Core Skills
- Links to Common Core Skills
- MathBERT Embeddings of Common Core Skills
- PCA of Common Core Score Metrics
- Hierarchical-Linkage Clustering Output
- ASSISTments Contextual Bandit Data