Learn With Martian: A Tool For Creating Assignments That Can Write And Re-Write Themselves

Shriyash Upadhyay
Martian
yash@withmartian.com

Etan Ginsberg
Martian
etan@withmartian.com

Chris Callison-Burch
University of Pennsylvania
ccb@upenn.edu

Abstract

In this paper, we propose Learn, a unified, easy-to-use tool to apply question generation and selection in classrooms. The tool lets instructors and TAs create assignments that can write and re-write themselves. Given existing course materials, for example a reference textbook, Learn can generate questions, select the highest quality questions, show the questions to students, adapt question difficulty to student knowledge, and generate new questions based on how effectively old questions help students learn. The modular, composable nature of the tools for handling each sub-task allow instructors to use only the parts of the tool necessary to the course, allowing for integration in a large number of courses with varied teaching styles. We also report on the adoption of the tool in classes at the University of Pennsylvania with over 1000 students. Learn is publicly released at https://learn.withmartian.com.

1 Introduction

Advances in natural language processing, particularly through large langauge models, will enable the creation of powerful applications in many fields (Bommasani et al., 2021). One of the most positive and most promising may be education, where rapid improvement has been achieved in fields like question generation (Dugan et al., 2022; Drori et al., 2022; Zhang et al., 2022).

In this paper, we propose Learn, a unified, easy-to-use tool to apply question generation in classrooms. The tool allows instructors to create assignments that can write and re-write themselves. Instructors provide existing course materials, and the platform is able to generate new questions, select the best questions, show those questions to students, provide analytics, and improve as it collects more data on student performance. We also report on a case study of the successful adoption of Learn in classes at the University of Pennsylvania.

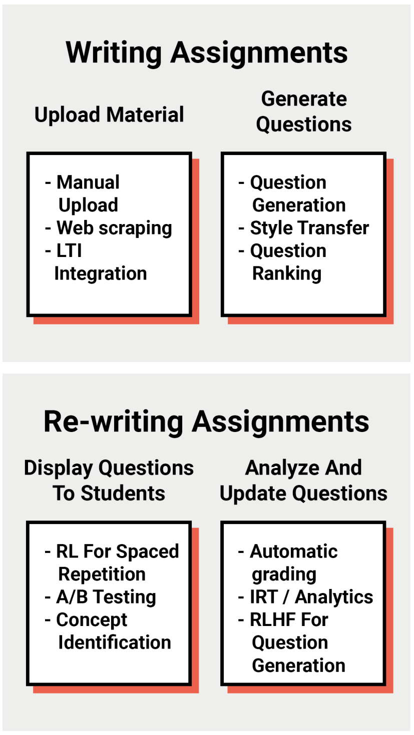
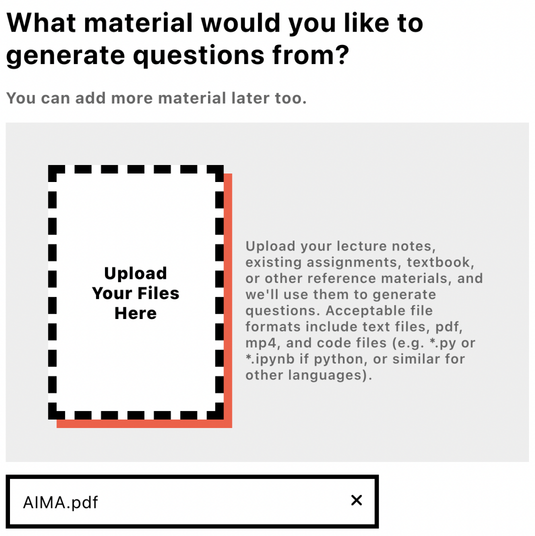


Figure 1: A diagram of steps, sub-steps, and features in Learn. Learn is a tool for creating assignments that write and re-write themselves. To write new assignments, users upload material and the platform generates questions. To re-write assignments and improve them, the questions are first shown to students, then the data from student interactions is analyzed to update the questions.

We hope this tool can be used by course staff to save time and improve education, and that Learn can serve as an example of a high-quality social-good tool which inspires the development of further applications.

2 Tool Tour and Design

We begin with a brief tour. Generating questions is done simply by uploading the materials from which the questions should be generated.


What material would you like to generate questions from?

You can add more material later too.

Upload Your Files Here

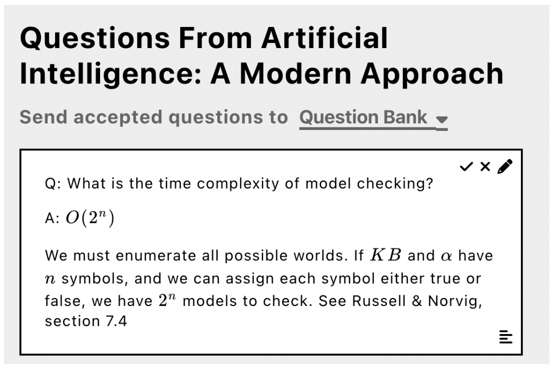
Upload your lecture notes, existing assignments, textbook, or other reference materials, and we'll use them to generate questions. Acceptable file formats include text files, pdf, mp4, and code files (e.g. *.py or *.ipynb if python, or similar for other languages).

AIMA.pdf ×

Figure 2: The interface for uploading material to Learn.

Uploaded materials can be text, code, pdf, or audio/video. In the case of PDF, they are converted to text through OCR. In the case of audio/video, they are converted to text through ASR.

Once the materials are uploaded and text is extracted, they are used to generate questions. The generated questions are then shown to the course staff.


Questions From Artificial Intelligence: A Modern Approach

Send accepted questions to Question Bank

Q: What is the time complexity of model checking?

A: $O(2^n)$

We must enumerate all possible worlds. If KB and α have n symbols, and we can assign each symbol either true or false, we have 2^n models to check. See Russell & Norvig, section 7.4

Figure 3: The interface which allows course staff to review questions. Users can accept, reject, or edit questions by clicking on the check, cross, and pencil buttons respectively.

Members of the course staff can accept, reject, or edit any question. Accepting a question adds it either to the general question bank for the course or to a particular assignment.

After questions are added to an assignment, the assignment can be released to students, who can then complete the assignment.

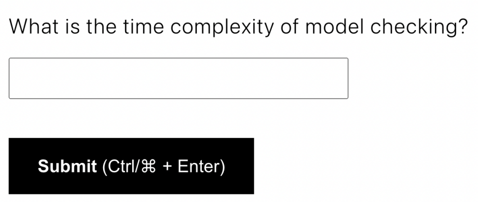
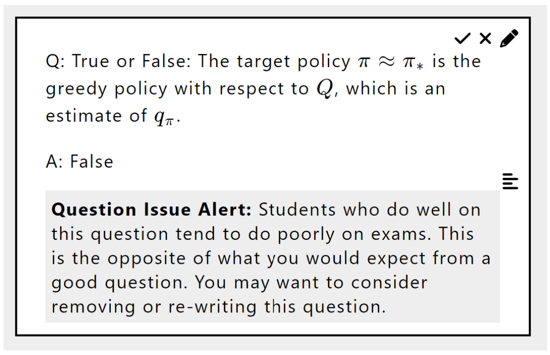

What is the time complexity of model checking?

Figure 4: The interface through which a student completes assignments by answering questions.

When the student is completing the assignment, their answers are stored, alongside relevant metadata such as the amount of time to complete the question and whether the student got the question correct. Students can also mark questions as being “really good” for studying, or as “not helpful”.

The data collected in this way is then analyzed, both to inform instructors and to improve future question generation. That data is used to provide suggestions to course staff.


Q: True or False: The target policy $\pi \approx \pi_*$ is the greedy policy with respect to Q, which is an estimate of q_π.

A: False

Question Issue Alert: Students who do well on this question tend to do poorly on exams. This is the opposite of what you would expect from a good question. You may want to consider removing or re-writing this question.

Figure 5: The interface alerting instructors about issues and potential improvement to existing questions.

Note 1: Uploading Material. Users have multiple means of uploading material to Learn. Although they can upload all their content manually, that is typically not necessary. Instead, if they have an existing course website, they can point a scraper built into the Learn tool at that website, and it will find the relevant material (e.g. the course textbook, lecture notes, youtube links, etc.) and only upload the material which cannot be found on their course website. We also have an LTI integration, allowing us to pull data from existing learning management systems (like Canvas and Moodle), as well as pushing data (like generated questions or student grades) to those systems. These alternative methods of uploading material decrease effort required to adopt the platform, making it something which can be used in minutes.

Note 2: Rich Question Formats. In most tools for creating assignments, questions are purely text. This is not the case for Learn. Instead, all questions are javascript functions of the type

```
() => ({question: HTML, answer: HTML})
```

If a user wants to create a question which acts like text, we provide an interface which lets them create such questions without writing any code, just as they would in a traditional assignment-creation software. However, we also give users access to a coding environment which they can use to create questions. This lets them – or our question generation models – generate questions in rich formats. That includes questions with images, Latex, or interactive components like coding challenges, animations, or full-fledged games (see figure 8).

Note 3: Composability. Courses can use the various parts of the tool independently. For example, if a course largely uses written assignments and does not want to switch to online assignments, the tool can be used purely for question generation by sending all questions to the question bank and then copying them into the physical exams. A course which has existing questions can bring them onto the platform, display them to students, then get analytics and have them re-written to be more effective. Courses with an LTI or which use other tools like gradescope can display all questions through those platforms, link them to Learn, and get analytics using their existing data. This allows many different kinds of courses to use the tool, and also for gradual adoption of those parts of the tool which are most useful to any given course.

Complete Flow An instructor uploads the material for their course, either manually, by pointing our scraper to their course website, or through an LTI integration. We then generate richly-formatted questions which can include interactive components through code. Those questions are placed into assignments which can be completed by students. Once students complete the assignments, the results are used to re-write the assignments and improve future question generation.

3 How Natural Language Processing Is Being Used To Augment Education

Although the most visible application of NLP in Learn is that questions are generated automatically, many aspects of writing and re-writing assignments are enhanced by NLP research. In this section, we detail the features of the platform enabled by NLP.

3.1 Question Generation

LLMs For Question Generation

Our platform uses multiple large langauge models, including GPT-3 (Brown et al., 2020), Codex (Chen et al., 2021), and a fine-tuned T5 (Raffel et al., 2019) variant to generate questions from material uploaded by course staff. Recent work has shown the ability of these models to generate questions for STEM subjects such as mathematics (Drori et al., 2022) and computer science (Zhang et al., 2022) at a college level. Our prior work (Dugan et al., 2022) showed that question quality can be greatly enhanced by summarizing input before using it to generate questions. Using that insight, we first extractively summarize the content using BERT (Devlin et al., 2018), before passing the summarized inputs to the larger models.

Prompting For Question Style Transfer

When generating questions for a course, there is a kind of cold-start problem. Courses use a wide variety of different questions, from multiple-choice to short-answer to long-form and proof-based. Courses may even prefer more fine-grained distinctions than those. How do we know what kind of question to generate, and how do we get our model to generate questions of that type?

Few-shot prompting, as in (Brown et al., 2020), allows us to solve this problem. In the materials which professors upload, there will often be examples of questions from assignments used in their class. We can then either generate question and then re-write them using an arbitrary style transfer method like that in (Reif et al.), or prepend a course's existing questions to our question generation prompt. This allows us to generate questions in a wide variety of styles, matching a wide variety of courses. As a demonstration, we can generate questions at all levels of Bloom's taxonomy (Bloom, 1956). Examples of questions at the different levels can be seen in appendix A (see table 1).

Over-generation & Ranking

Despite the successes of large language models, many of the questions which these models generate are still not acceptable according to human annotators. Our prior work established that even high-performing completely-automated methods only generate acceptable questions 50% of the time (Dugan et al., 2022). To mitigate this problem, we over-generate and rank questions using BERT. So, if a course needs 20 questions for an assignment, we might generate 200 questions and surface the top 20.

Course staff make the final decision about what questions should be shown to students. In particular, course staff are shown the generated questions, and either accept, reject, or edit questions. This has two advantages. First, it ensure that all questions which students see are of high quality. Second, it provides data to train a ranking model. Accepted questions are better than rejected questions, and edited questions are better after editing than before. As a result, our ranking model (and the quality of questions shown to users) improves over time.

3.2 Showing Questions To Students

Spaced Repetition & Reinforcement Learning

Not all orders of presenting material to students are equally effective. It is clear, for example, that showing a student material on calculus before they understand arithmetic is much less effective than the other way around.

A particularly effective method of ordering material to present it to students is spaced repetition (Tabibian et al., 2019). The core idea is that, instead of showing a piece of material to a student only once or cramming many reviews all at once, they should review that material multiple times at intervals to avoid forgetting it. This technique has been shown to improve student outcomes in subjects as varied as mathematics (Smith and Rothkopf, 1984; Mayfield and Chase, 2002; Patac and Patac JR, 2013), language learning (Cepeda et al., 2009), and medicine (Kerfoot and Brotschi, 2009; Kerfoot, 2009). (Donovan and Radosevich, 1999) find an overall mean weighted effect size of d=0.46 when evaluating the effectiveness of spaced repetition across 63 studies.

Learn implements spaced repetition to determine what questions should be shown to students. This improves student understanding without any additional effort for course instructors.

As noted in (Rafferty et al., 2016), the question of what material should be shown to students can be formulated as a Partially-Observable Markov Decision Process (POMDP). That means we can construct a schedule via reinforcement learning, as in (Reddy et al., 2017), to determine what material should be shown to a student in order to maximize their understanding of the material. This has the potential to significantly out-perform more traditional spaced-repetition algorithms.

A/B Testing Assignments

In addition to allowing the platform to select the questions shown to the students, we can test collections of questions by A/B testing assignments. This ability is also provided to instructors, who can assign different versions of an assignment to different students. This allows instructors to collect information about their assignments and get additional analytics (see section 3.3).

Automatically Identifying Course Concepts

A common criticism of spaced repetition is that it primarily aids in "rote memorization": if a student reviews the same question repeatedly, they may only remember the content of the question, as opposed to remembering the underlying concept.

A solution to this problem is to repeat a concept with multiple different questions, rather than a single question. For example, if asking questions about breadth-first search, the first repetition might ask about the running time, the second repetition about the implementation of the algorithm, the third repetition about a specific application, etc.

Learn implements both supervised and unsupervised methods to identify concepts. First, we use BERT to identify questions which are explicitly asking the same thing in different ways. Then, we cluster embedded vectors of the questions to identify groups of questions with high semantic similarity. By showing students questions selected from these groups, we can avoid the problem of memorization while still benefiting from spaced repetition.

3.3 Analytics & Improving Questions

Item Response Theory

Item response theory (IRT) is a statistical framework for analyzing the effectiveness of tests and of individual questions within those tests. An introduction to the theory can be found in (Partchev,

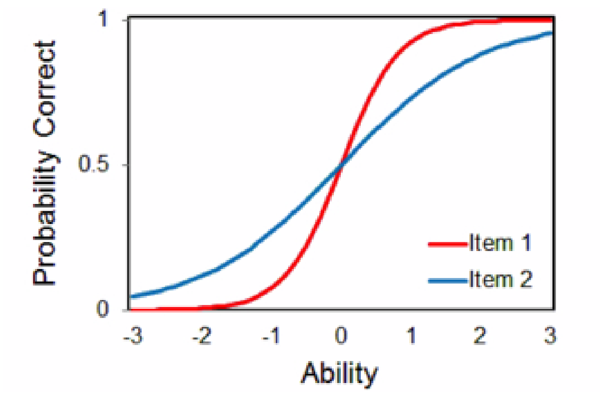


Figure 6: Examples of two IRT curves from Columbia Public Health. The x-axis represents a student's ability, as measured by standard deviations from average performance in the overall course. They y-axis represents the probability that a student gets the question correct. Both curves here are 3 parameter logistic regression models. Item 1 is a better question than item 2 because its IRT curve is steeper, are therefore distinguishes ability better. Both questions are of equal difficulty, as they are centered around the same point (ability=0).

2004).

At a basic level, we want questions to tell us what students know. A question is a good indicator of a student's knowledge if students who get that question correct tend to do well in the course overall. So, we can measure the quality of a question by seeing how strongly it correlates with overall performance on assignments (see figure 6).

By computing this correlation, we can inform course staff of particularly good or bad questions (see figure 5), and refine assignments over time.

Automating The Creation of Autograders

Some questions are easy to grade, and the act of creating them results in an auto-grader (e.g. multiple choice questions, or questions that are solved by invoking an algorithm such as linear algebra problems). Many questions are harder to grade, and in many classes a majority of TA time is spent grading such questions.

To reduce the burden on TAs, Learn can automatically construct auto-graders for questions using data collected when students complete the questions. There are multiple methods which Learn uses to do this. The first is that students or TAs can mark the questions as correct or incorrect, then the platform looks for exact strings matches in the future and is able to tell that those are certainly correct. We can also use a BERT model to classify the correctness of a new answer given previous answers. These can both be done in an automated manner, automatically reducing the grading burden for course staff. We also cluster the errors which students make, allowing TAs to grade the work of many students at once.

RLHF For Question Generation

The annotations we collect from TAs (accepting, rejecting, or editing questions) provides us with data about human feedback for question generation (see section 3.1). Following the methodology from (Ziegler et al., 2019), we can then use this data to improve our question generation models through the following process:

1. Starting with a supervised policy, in this case, our existing question generation model.

2. Training a reward model from the human feedback on question generation.

3. Optimizing a policy against the reward model using Proximal Policy Optimization (PPO). (Schulman et al., 2017)

This allows us to improve our question generation models over time as we collect additional usage data.

In the future, we will explore the use of other data we collect (for example, IRT metrics) as additional supervision signals; if analytics can improve questions that TAs have written, then they also express preferences which should be encoded in the reward model, potentially allowing for more rapid improvement.

4 Case Study

Learn has been integrated in multiple courses at the University of Pennsylvania, and over 1000 students have used the platform. The different classes used the tool in different ways, verifying its composability. In many of those classes, for example, Learn was a supplementary tool, whereas the Introduction to Artificial Intelligence class used it as a quizzing platform for reading quizzes. In the latter class, we were able to collect significant data on the efficacy of the platform, which we report here.

4.1 Methodology

We conducted a control trial in the Introduction to Artificial Intelligence course at the University of Pennsylvania. The course is divided into two sections of approximately 300 students each. We randomly assigned one section to use traditional reading quizzes (which had been used in the class during prior years), while the other section used the Learn platform. On both platforms, the quizzes were administered on a weekly basis. For the first three weeks of the class, all students used the traditional quizzes, then the section using Martian switched to using the new platform. The quizzes covered the required readings for that week (from the textbooks *Artificial Intelligence: A Modern Approach* and *Speech and Language Processing*). The students using Learn were also shown material from prior weeks due to the integration of spaced repetition (see section 3.2). Material on Learn was created through a combination of automatic generation and TA-written questions.

There is one notable confounder in the experiment, which is that one section of the class was entirely online, while the other had the option of both online and in-person classes. After random assignment, the online-only section used traditional reading quizzes, while the mixed section used Learn. However, from exams scores in previous semesters, we do not expect a difference in exam scores between the online and mixed sections.

We collected two evaluation metrics to determine the effectiveness of the platform. First, we collected the students' exam scores, which we could compare across sections. Second, as a more subjective metric, we asked the students who used both traditional reading quizzes and Learn which platform they preferred.

4.2 Results

The exam scores for students using Learn were higher than those using traditional reading quizzes. The students in the section that used Learn had exam scores which were 0.29σ higher on average. Among students who used Learn, every 15 minutes of additional studying led to a 0.08σ improvement in exam scores.

Similarly, Learn outperformed traditional reading quizzes according to the subjective evaluations of the students. 83% of students preferred Learn to traditional reading quizzes (see figure 7). 11% of students preferred traditional reading quizzes. 6% had no preference. We also received comments from students regarding the platform, overwhelmingly positive, some of which can be found in Appendix A.

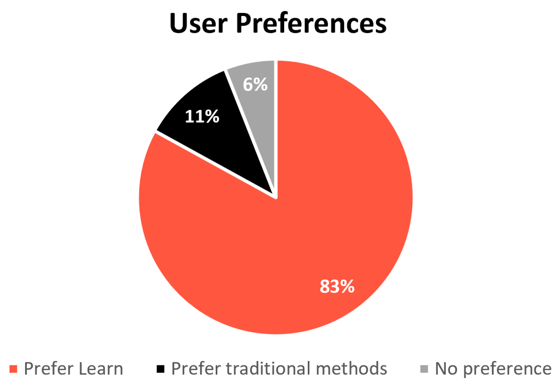


Figure 7: At the end of the semester, students were asked which section they preferred. A large majority of students preferred Learn to traditional methods.

5 Conclusion

Learn is a unified, easy-to-use tool to apply question generation in classrooms. The tool is able to create assignments which write and re-write themselves. It achieves this through a number of features enabled by recent advances in NLP. Having been tested in multiple classes with over 1000 students, it has been demonstrated to improve test scores and is prefered by students to traditional alternatives.

Limitations

While the platform is well-featured, we hope to be able to evaluate those features more thoroughly in the future. The limitations to evaluating the system so far stem from limits in how much the system has been used.

This is a direct limit in the case of testing certain features, like RLHF for question generation (see section 3.3), which require more data than we have currently collected. It is an indirect limit in the kinds of classes which the tool has been tested in. At the time of authorship, all the classes in which the tool has been tested are computer science classes. Although we see no reason why the tool should not generalize to other courses (we have tested the quesion generation outside of computer science), that should be verified through further expermentation.

Ethics Statement

If our platform works as intended and sees broader use in higher-ed, course staff and students can benefit.

For course staff, it can reduce the time taken in writing and grading assignments. It should also help to alleviate concerns around cheating, as new questions can be generated each semester, potentially even creating new questions for each individual student. By reducing this logistical burden, Learn will allow course staff to focus on the parts of teaching which are higher-leverage, and to improve their courses in ways that weren't before possible.

For students, the introduction of Learn into a classroom is an opportunity to use more effective methods for studying (such as spaced repetition), to see higher-quality study material, and to have many more questions with which to study. The ability to automatically generate questions provides an opportunity for students to get much more practice – an arbitrarily large quantities of practice question could be created, tailored to the topics in which any particular student is struggling. As the system improves over time we hope that students will have study materials of higher quality than they would without automated tools.

There are also large disparities in computer science, especially for certain underrepresented groups. These groups face structural issues which may lead to differences in achievement. For example, students in underrepresented groups may feel uncomfortable asking questions of or reaching out to course staff, and therefore may receive less attention and aid from instructors. We can help tackle those disparities by improving accessibility through more effective learning systems.

It is also important to note, however, the potential risks to privacy when building and using education applications. Given the sensitive nature of the data, such as grades, it is important to take proper privacy and security safeguards when deploying such systems. In order to minimize such risks, we went through thorough evaluation by the privacy office for the Computer and Information Science Department at the University of Pennsylvania. We also filled out the HECVAT lite security questionnaire, receiving an A rating.

As education is downstream of many important societal factors, such as economic growth and public welfare, we are optimistic about the positive impacts of new AI applications for education, like Learn.

Acknowledgements

This work would not be possible without Dr. Eleni Miltsakaki, Liam Dugan, Hannah Gonzalez, and Zhiqi "Tracy" Cui who helped do the fundamental question generation research underlying this work. We'd also like to express our thanks to Dylan Fields and the privacy team at the University of Pennsylvania for reviewing our application, and to Dr. Zachary Ives and Dr. Boon Thau Loo for their support in adopting Learn.

This research is based upon work supported in part by the the NSF (Award 1928631). Approved for Public Release, Distribution Unlimited. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the NSF or the U.S. Government.

References

Benjamin Samuel Bloom. 1956. Taxonomy of educational objectives: The classification of educational goals. *Cognitive domain*.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Nicholas J Cepeda, Noriko Coburn, Doug Rohrer, John T Wixted, Michael C Mozer, and Harold Pashler. 2009. Optimizing distributed practice: theoretical analysis and practical implications. *Experimental psychology*, 56(4):236.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail

Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding.

John J Donovan and David J Radosevich. 1999. A meta-analytic review of the distribution of practice effect: Now you see it, now you don't. *Journal of Applied Psychology*, 84(5):795.

Iddo Drori, Sarah Zhang, Reece Shuttleworth, Leonard Tang, Albert Lu, Elizabeth Ke, Kevin Liu, Linda Chen, Sunny Tran, Newman Cheng, Roman Wang, Nikhil Singh, Taylor L. Patti, Jayson Lynch, Avi Shporer, Nakul Verma, Eugene Wu, and Gilbert Strang. 2022. A neural network solves, explains, and generates university math problems by program synthesis and few-shot learning at human level. *Proceedings of the National Academy of Sciences*, 119(32):e2123433119.

Liam Dugan, Eleni Miltsakaki, Shriyash Upadhyay, Etan Ginsberg, Hannah Gonzalez, DaHyeon Choi, Chuning Yuan, and Chris Callison-Burch. 2022. A feasibility study of answer-agnostic question generation for education. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1919–1926, Dublin, Ireland. Association for Computational Linguistics.

B Price Kerfoot. 2009. Learning benefits of on-line spaced education persist for 2 years. *The Journal of urology*, 181(6):2671–2673.

B Price Kerfoot and Erica Brotschi. 2009. Online spaced education to teach urology to medical students: a multi-institutional randomized trial. *The American Journal of Surgery*, 197(1):89–95.

Kristin H Mayfield and Philip N Chase. 2002. The effects of cumulative practice on mathematics problem solving. *Journal of applied behavior analysis*, 35(2):105–123.

Ivailo Partchev. 2004. A visual guide to item response theory. *Retrieved November*, 9:2004.

Louida P Patac and Adriano V Patac JR. 2013. The analysis of two teaching programs: Massed and distributed. *IAMURE International Journal of Mathematics, Engineering & Technology*, 6:59.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer.

Anna N Rafferty, Emma Brunskill, Thomas L Griffiths, and Patrick Shafto. 2016. Faster teaching via pomdp planning. *Cognitive science*, 40(6):1290–1332.

Siddharth Reddy, Sergey Levine, and Anca Dragan. 2017. Accelerating human learning with deep reinforcement learning. In *NIPS'17 Workshop: Teaching Machines, Robots, and Humans*, pages 5–9.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. A recipe for arbitrary text style transfer with large language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL 2022)*, Dublin, Ireland.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Steven M Smith and Ernst Z Rothkopf. 1984. Contextual enrichment and distribution of practice in the classroom. *Cognition and Instruction*, 1(3):341–358.

Behzad Tabibian, Utkarsh Upadhyay, Abir De, Ali Zarezade, Bernhard Schölkopf, and Manuel Gomez-Rodriguez. 2019. Enhancing human learning via spaced repetition optimization. *Proceedings of the National Academy of Sciences*, 116(10):3988–3993.

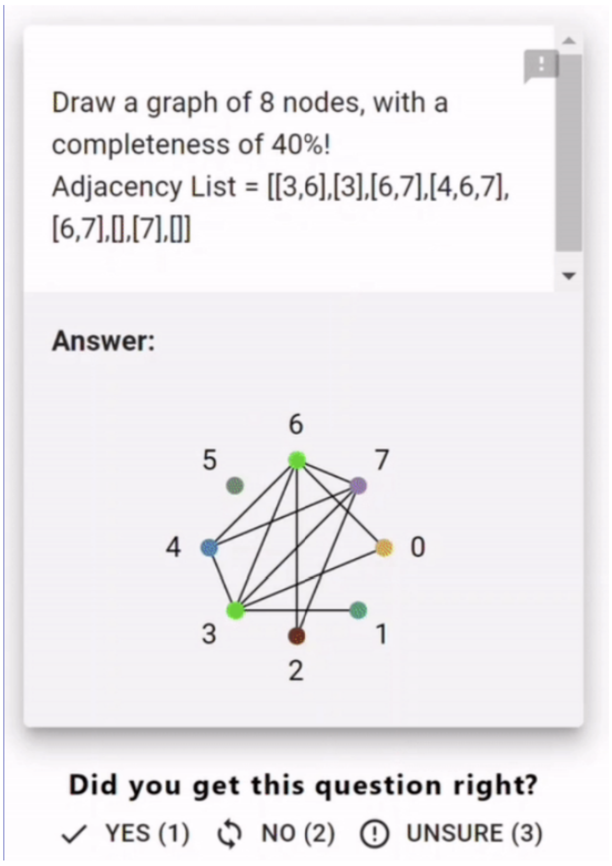
Sarah Zhang, Reece Shuttleworth, Derek Austin, Yann Hicke, Leonard Tang, Sathwik Karnik, Darnell Granberry, and Iddo Drori. 2022. A dataset and benchmark for automatically answering and generating machine learning final exams.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Select Comments From Students

1. "Thank you for making this! I really wasn't sure how much it would help me since I have established study techniques I've used throughout hs/college and didn't want to change. But this has actually been so immensely helpful and I now really believe in the proven science behind how tools like this can really help retain information."

2. "This is a great tool - amazing job overall!"


Draw a graph of 8 nodes, with a completeness of 40%!
Adjacency List = [[3,6],[3],[6,7],[4,6,7],[6,7],[],[7],[]]

Answer:

Figure 8: A question created using code which can randomly generate an graph from an adjacency list, and display an image of the generated graph.

3. "It was very helpful (the repetitive review of materials). The variety of some cards asking the same material helped a lot too."
4. "I thought [Learn] was [a] very helpful way to learn the material!"
5. "Thank you guys! This was really helpful!"

B Example Questions

Level	Question	Answer
Remember	What is the policy improvement theorem?	The policy improvement theorem is a theorem that says if we have two deterministic policies π and π' such that, for all $s \in \mathcal{S}$, $q_\pi\left(s, \pi'(s)\right) \geq v_\pi(s)$, then the policy π' must be as good as, or better than, π. That is, it must obtain greater or equal expected return from all states $s \in \mathcal{S}$: $v_{\pi'}(s) \geq v_\pi(s)$.
Understand	Why is policy evaluation guaranteed to converge?	Because the value function is a fixed point of the Bellman equation.
Apply	What is $v_\pi(15)$ for the equiprobable random policy in this case?	-1
Analyze	What is the difference between Synchronous DP and Asynchronous DP?	Synchronous DP updates state values in a deterministic order, e.g. from small to large. Asynchronous DP updates state values in a stochastic order.
Evaluate	What are the limitations of Dynamic Programming?	The limitations of Dynamic Programming are that it requires a perfect model of the environment and that it is computationally expensive.
Create	Draw a graph of 8 nodes, with a completeness of 40%! Adjacency List = [[3, 6], [3], [6, 7], [4, 6, 7], [6, 7], [], [7]]	Varies by response

Table 1: A collection of questions generated using Learn at each level of Bloom's taxonomy (Bloom, 1956). Bloom's taxonomy establishes a series of levels of that represent different cognitive goals. The cognitive goals become more difficult, going from remembering to creating.