A Dataset for Learning University STEM Courses at Scale and Generating Questions at a Human Level

Iddo Drori[1, 4, 5], Sarah Zhang[1], Zad Chin[2], Reece Shuttleworth[1], Albert Lu[1], Linda Chen[1], Bereket Birbo[1], Michele He[1], Pedro Lantigua[1], Sunny Tran[1], Gregory Hunter[4], Bo Feng[4], Newman Cheng[4], Roman Wang[4], Yann Hicke[3], Saisamrit Surbehera[4], Arvind Raghavan[4], Alexander Siemenn[1], Nikhil Singh[1], Jayson Lynch[6], Avi Shporer[1], Nakul Verma[4], Tonio Buonassisi[1], Armando Solar-Lezama[1]

[1]Massachusetts Institute of Technology,
[2]Harvard University,
[3]Cornell University,
[4]Columbia University,
[5]Boston University,
[6]University of Waterloo,
{idrori@mit.edu, idrori@cs.columbia.edu, idrori@bu.edu, sazhang@mit.edu, zadchin@college.harvard.edu, rshuttle@mit.edu, albert03@mit.edu, linda55@mit.edu, bereketb@mit.edu, mjhe@mit.edu, lantigua@mit.edu, sunnyt@mit.edu, geh2129@columbia.edu, bf2477@columbia.edu, nc2893@columbia.edu, rzw2002@columbia.edu, ylh8@cornell.edu, ss6365@columbia.edu, ar4284@columbia.edu, asiemenn@mit.edu, nsingh1@mit.edu, jayson.lynch@waterloo.ca, jayson.lynch@waterloo.ca, shporer@mit.edu, verma@cs.columbia.edu, buonassi@mit.edu, asolar@csail.mit.edu}

Abstract

We present a new dataset for learning to solve, explain, and generate university-level STEM questions from 27 courses across a dozen departments in seven universities. We scale up previous approaches to questions from courses in the departments of Mechanical Engineering, Materials Science and Engineering, Chemistry, Electrical Engineering, Computer Science, Physics, Earth Atmospheric and Planetary Sciences, Economics, Mathematics, Biological Engineering, Data Systems, and Society, and Statistics. We visualize similarities and differences between questions across courses. We demonstrate that a large foundation model is able to generate questions that are as appropriate and at the same difficulty level as human-written questions.

Introduction

We scale-up the automatic generation of new university-level problem set and exam questions to 27 STEM courses across a dozen departments, each from unique disciplines, from seven universities (MIT, Harvard, Princeton, Cornell, Brown, UPenn, and Yale). We curate a new dataset of course questions and answers across a dozen departments: Mechanical Engineering, Materials Science and Engineering, Chemistry, Electrical Engineering, and Computer Science, Physics, Earth Atmospheric and Planetary Sciences, Economics, Mathematics, Biological Engineering, Data Systems, and Society, and Statistics. The courses, departments, and universities are summarized in Table 1. The machine generated questions are found to be nearly indiscernible from the human-written questions. Furthermore, we automatically solve and explain calculations to the questions from one of the 27 courses with higher accuracy. Through this contribution, artificial intelligence supports the improvement and increased access to education by automatically generating and solving university STEM class questions.

In recent years, transformers and other large language models have significantly progressed in question-answering. A recent breakthrough solved university-level mathematics course problems using program synthesis and few-shot learning at a human level (Drori et al. 2022), and tackled Introduction to Machine Learning final exams (Zhang et al. 2022). In this work, we scale up to a broad range of STEM courses across a dozen departments in the schools of engineering and science. Example questions, answers, and explanations for a sample of different STEM courses are shown in Figure 2. We make the dataset available in the Supplementary Material.

Previous specialized work answers numerical machine learning questions (Tran et al. 2021); however, it overfits a specific course. While this specialized method may be applied to other individual courses, such as solving Introduction to Astronomy by incorporating additional operations into expression trees of the specialized approach, such as integrals and derivatives, and incorporating a table of astrophysical constants and units, it is a course-specific method and does not solve any other course and does not scale up to many courses. In contrast, program synthesis by a Transfomer model pre-trained on text and fine-tuned on code using few-shot learning scales up to many STEM courses.

We compare different approaches for answering

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

ID	University	Department	Course	Number
1	MIT	Mechanical Engineering	Hydrodynamics	2.016
2	MIT	Mechanical Engineering	Nonlinear Dynamics I: Chaos	2.050J
3	MIT	Mechanical Engineering	Information & Entropy	2.110J
4	MIT	Mechanical Engineering	Marine Power and Propulsion	2.611
5	MIT	Materials Science and Engineering	Fundamentals of Materials Science	3.012
6	MIT	Materials Science and Engineering	Math for Materials Scientists & Engineers	3.016
7	MIT	Materials Science and Engineering	Introduction to Solid-State Chemistry	3.091
8	MIT	Chemistry	Principles of Chemical Science	5.111
9	MIT	Electrical Engineering & Computer Science	Signal Processing	6.003
10	MIT	Electrical Engineering & Computer Science	Introduction to Machine Learning	6.036
11	MIT	Electrical Engineering & Computer Science	Introduction to Probability	6.041
12	MIT	Physics	Quantum Physics	8.04
13	MIT	Physics	Introduction to Astronomy	8.282
14	MIT	Earth, Atmospheric & Planetary Sciences	Geobiology	12.007
15	MIT	Economics	Principles of Microeconomics	14.01
16	MIT	Aeronautics and Astronautics	Unified Engineering 1–2	16.01–02
17	MIT	Aeronautics and Astronautics	Unified Engineering 3–4	16.03–04
18	MIT	Mathematics	Probability and Random Variables	18.600
19	MIT	Mathematics	Theory of Numbers	18.781
20	MIT	Biological Engineering	Systems Microbiology	20.106J
21	MIT	Institute for Data, Systems & Society	Statistical Thinking & Data Analysis	IDS.013J
22	Brown	Mathematics	Intermediate Calculus	MATH0180
23	Cornell	Computer Science	Computer Architecture	CS4420
24	Harvard	Statistics	Probability	STATS110
25	Princeton	Mathematics	Calculus II	MATH104
26	UPenn	Mathematics	Calculus	MATH110
27	Yale	Mathematics	Fundamentals of Physics	PHYS200

Table 1: A new dataset of questions and solutions from STEM courses by university and department: 27 courses across a dozen departments in seven universities. We curate a dataset and generate new questions for each course.

university-level STEM course questions: (1) code-based approach of program synthesis using OpenAI Codex, (2) text-based approach using GPT-3 with and without chain-of-thought prompting, (3) a specialized transformer-based model pre-trained on text and a graph neural network.

In this work, we introduce an open-source dataset of 667 STEM questions across 27 different courses for a dozen different disciplines. We demonstrate a generalized method to scale-up the generation of new university-level STEM questions for these 27 different courses. A general model enables scale-up of STEM question generation across a broad range of course material because no additional model parameter tuning is required, unlike specialized models that are often highly tuned for a narrow range of applications. Lastly, we survey a dozen STEM students who have taken at least one of the 27 courses to anonymously evaluate the generated question quality, appropriateness relative to the course, and the question difficulty. This work is a significant step forward in applying artificial intelligence to education, automating a considerable part of the work involved.

Related Work

Foundation Models. Transformers, or foundation models such as BERT and GPT, have become a standard tool in modern machine learning and AI (Bommasani et al. 2021). These large models have been shown to act as few-shot learners for various tasks (Brown et al. 2020) and may be used as a base model and tuned on specific tasks. The success of these models has been enabled not only by massive datasets and computational resources but also by the development of the transformer architecture (Vaswani et al. 2017) which has been essential in both NLP (Devlin et al. 2019; Shoeybi et al. 2019; Raffel et al. 2020; Brown et al. 2020) and computer vision (Dosovitskiy et al. 2020). Despite incredible success at many NLP-related tasks, these models have difficulty with quantitative questions and mathematical reasoning (Hendrycks et al. 2020, 2021). However, there have been various attempts to use these models as part of a system to answer mathematical or quantitative questions and augment these systems to handle formulas and other formal systems better. One example is MathBERT (Peng et al. 2021), a Transformer based, pre-trained language model that uses symbol layout trees and operator trees as intermediary representations of formulas to improve performance on information retrieval questions related to mathematical formulas. The model is trained on masked prediction modeling and context correspondence prediction tasks using the textual formula, context, and formula trees as input.

Graph Neural Networks. In addition to Transformers, our specialized system uses graph neural networks (GNNs) (Kipf and Welling 2017; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2018; Xu et al. 2019) which maintain associations between data. We use GNNs to maintain relationships between numerical quantities, descriptions, and representation in formulas and our predicted expression tree.

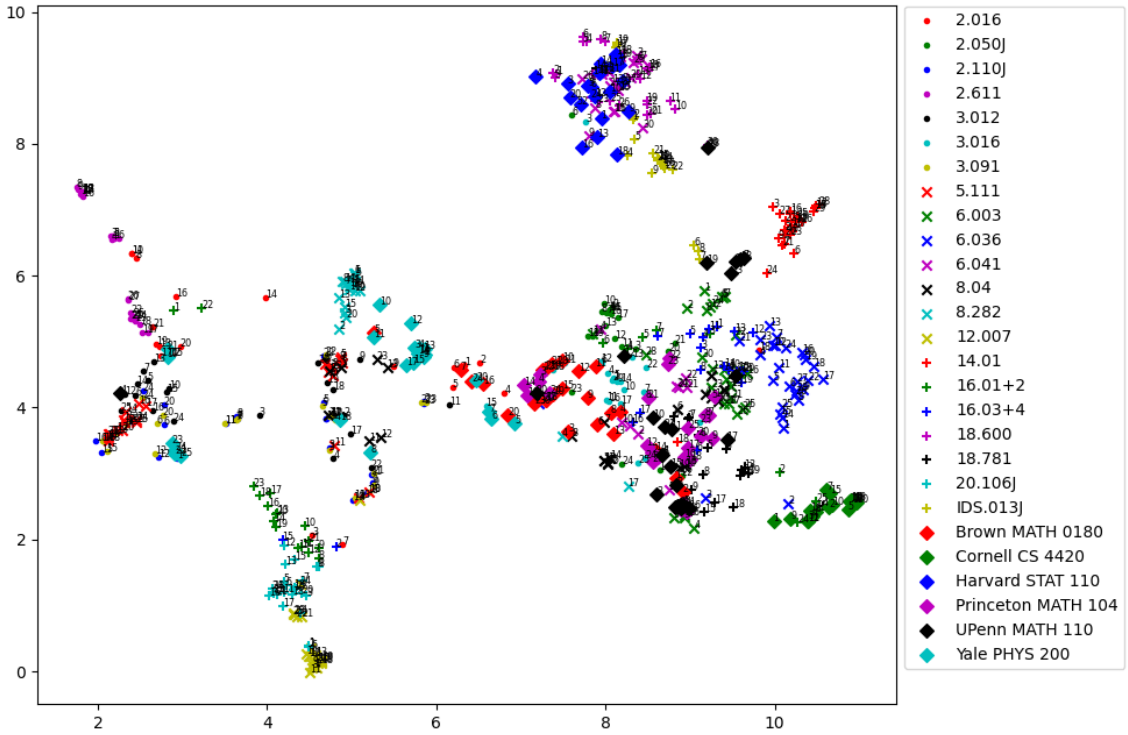

Figure 1: Visualization of embeddings of course questions: We embed the course questions into a 2,048-dimensional space using OpenAI's *text-similarity-babbage-001* embedding, which captures semantic similarity between texts. We then use uniform manifold approximation and projection to reduce the dimensionality to two, observing distinct clusters based on course topics. On the top right, we see a cluster of questions from probability and statistics courses: MIT's 6.041 Introduction to Probability, 18.600 Probability and Random Variables, MIT's IDS013J Statistical Thinking and Data Analysis, and Harvard's STAT 110. On the left side, we see a cluster of the questions from Mechanical Engineering courses: MIT's 2.611, 2.016, and 2.110J. Next, we see a cluster of questions from Chemistry and Materials Science and Engineering on the left. On the bottom left, we see a cluster of the questions from Aeronautics and Astronautics: 16.01, 16.02, 16.03, and 16.04. Below is a cluster of questions from MIT's 20.106J System Microbiology and 12.007 Geobiology. In the center, we see a cluster of questions from Yale PHYS 200, MIT's Quantum Physics, and Introduction to Astronomy, all related to Physics. On the right, we see a cluster of questions from MIT EECS courses. A cluster of questions from math courses appears in the center between EECS and Physics.

Math Word Problems. Math word problems are natural language questions with numerical answers that are solved with mathematical deduction. Recent work has focused on a large database of questions collected from Chinese elementary school math classes. Techniques have included sequence-to-sequence and graph-to-tree Transformers which achieve around 80% accuracy on Math23k and MAWPS (Koncel-Kedziorski et al. 2016; Li et al. 2019; Wang et al. 2019; Zhang et al. 2020a; Li et al. 2020a; Wu et al. 2020; Qin et al. 2020; Lan et al. 2021). There is a growing line of work carried out in order to reach this point, starting with direct learning with various models, to hand-designed features with predicted relations (Kushman et al. 2014), to graphical models of equations (Zhang et al. 2020b) and prediction of expression trees (Xie and Sun 2019; Wu et al. 2020; Qin et al. 2020), and the incorporation of GNNs (Li et al. 2020b), auto-encoders, and pointer transformers (Kim et al. 2020). MWP-BERT, which adds additional masked fine-tuning to the BERT model using a large corpus of over 100,000 math word problems, achieves an impressive 96.2% accuracy on the Math23k dataset (Liang et al. 2021). Knowledge graphs of geometry formulas were incorporated into sequence-to-tree transformers to improve performance on the geometry section of Math23k (Tsai et al. 2021). Using sequence-to-sequence neural nets to generate symbolic expression trees has also been applied to solving integration and ordinary differential equation (Lample and Charton 2019) problems which outperform Mathematica and Maple on test problems.

Machine Learning Physics Models The AI Feynman system (Udrescu and Tegmark 2020; Udrescu et al. 2020) and improvements (Xing, Salleb-Aouissi, and Verma 2021) uses machine learning to perform symbolic regression on data from physical systems. AI Feynman uses neural networks to guide automated search over symbolic expression trees. The system finds the analytic expressions for most of the 120 physics equations from standard textbooks. We use similar ideas regarding expression trees as intermediary rep-

resentations; however, numerical question answering is fundamentally different from symbolic regression.

Automatic question generation Manual question generation is a difficult endeavor that demands time, experience, and ample educational content. To cut down on these costs a lot of research has been done to develop techniques that generate questions automatically. (Kurdi et al. 2020) reviewed the literature extensively in 2020. Before then (Ch and Saha 2018) and (Papasalouros and Chatzigiannakou 2018) did a related review of research done up until 2018 and before them (Alsubait, Parsia, and Sattler 2016) and (Rakangor and Ghodasara 2015) reviewed the literature up until 2014.

Methods

Dataset. We curate a dataset of 667 STEM questions from 27 courses. We embed these questions into a 2,048-dimensional space using OpenAI's *text-similarity-babbage-001* embedding engine, that captures semantic similarity between texts. We use uniform manifold approximation and projection (UMAP) (McInnes, Healy, and Melville 2018) to visualize the questions in two dimensions, as shown in Figure 1. We see distinct clusters based on course topics. On the top-right is a cluster of questions from probability and statistics courses: MIT's 6.041 Introduction to Probability, 18.600 Probability and Random Variables, MIT's IDS013J Statistical Thinking and Data Analysis, and Harvard's STAT 110. On the left is a cluster of the questions from Mechanical Engineering courses: MIT's 2.611, 2.016, and 2.110J. Next, on the left is a cluster of Chemistry, Materials Science, and Engineering questions. On the bottom left is a cluster of the questions from Aeronautics and Astronautics: 16.01, 16.02, 16.03, and 16.04. Below is a cluster of questions from MIT's 20.106J System Microbiology and 12.007 Geobiology. In the center is a cluster of questions from Yale PHYS 200, MIT's Quantum Physics, and Introduction to Astronomy, all related to Physics. On the right is a cluster of questions from MIT EECS courses. A cluster of questions from math courses appears in the center between EECS and Physics.

Program synthesis. Program synthesis aims to automatically generate executable code that achieves a goal specified using natural language. We turn each given question into the task of writing a program that can use relevant programming libraries such as SymPy (Meurer et al. 2017) to solve the question. We find program synthesis tools like OpenAI's Codex (Chen et al. 2021) to be powerful problem-solvers for three core reasons: (1) Codex is a generic Transformer pretrained on text and fine-tuned on code, hence, allowing us to synthesize programs, (2) Generic models support scalability to generate and solve problems from courses within different disciplines without major model adjustments, unlike that of specialized models that have a scope limited to a single discipline, (3) Program synthesis techniques support the use of programming libraries, for example, Astropy (Robitaille et al. 2013) [1], which consist of physical constants, units conversions, and many functions used in Astronomy.

Figures 2 and 3 demonstrate our pipeline using Codex to solve and explain questions using program synthesis and explanation synthesis. We use Codex to generate explanations for the solutions by appending the following text:"" ""Here's what the above code is doing: 1.", or "# Explanation of what the code does", below the synthesized programs, and then running Codex.

Generalized approach. The generalized approach uses few-shot learning of questions by their nearest embeddings, and OpenAI's GPT-3 and Codex to generate questions and explanations.

Specialized approach. A specialized architecture is used to predict answers to the questions from weekly assignments and exams in the MIT Astronomy course (8.282). Implementation details and predicted answers are shown in the Supplementary Material.

Model Evaluation

We use both the numerical solution and expression as two types of performance metrics. We use the generated expression tree as the training objective of our system. Similar to human grading, the evaluation metric is the number of correct answers divided by the number of questions, which is the measure of accuracy in this work. This is the same for measuring the performance of both numerical answer correctness and expression tree correctness. We use a random disjoint training and test set with a $0.9/0.1$ split from our augmented dataset.

Implementation Details

Generalized approach. In all of our experiments, we set Codex hyperparameters to be the same fixed default values for all solution and explanation experiments to yield deterministic and reproducible results. We use the latest version of OpenAI's GPT-3 *text-davinci-003* and Codex *code-davinci-002* engines with temperature 0 to avoid randomness and set the response length 500 tokens.

Few-shot learning prompts are generated by question–code examples of the nearest embedded questions followed by the question. Chain of thought (CoT) prompts for GPT-3 are implemented by adding the text "Let's think step by step." (Kojima et al. 2022b) after the question.

Specialized approach. Our code is in Python, using the libraries Pytorch[2], Deep Graph Library (DGL)[3], and HuggingFace's Transformers[4]. We use DGL's graph convolutional network to generate our graph model and perform tree decoding. The Transformers library provides the necessary tokenizers and pre-trained models. All experiments are run on a cloud instance optimized for computing, with 30 vCPUs and 120 GB of memory. Total training time takes around a day, augmentation time takes less than a minute, and test time is milliseconds per question. Each experiment takes roughly 6 hours to train, including data augmentation,

[1] https://github.com/astropy/astropy

[2] https://github.com/pytorch/pytorch

[3] https://github.com/dmlc/dgl

[4] https://github.com/huggingface/transformers

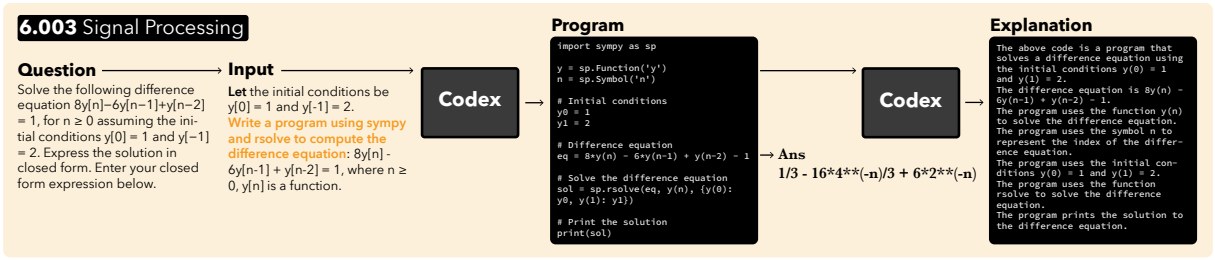

Figure 2: MIT 6.003 Signal Processing workflow: The question is solved as is and the prompt adds programming context to use symbolic math sympy package to produce code snippets that generate answers in form of a symbolic mathematical equation.

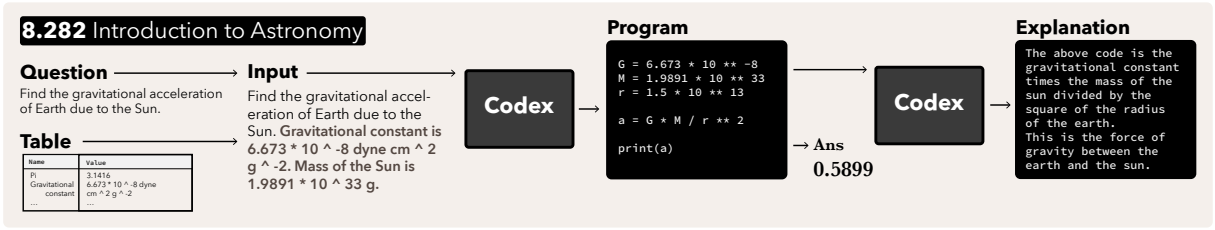

Figure 3: MIT 8.282 Introduction to Astronomy workflow: In this course, Codex often requires context about physical constants. This question involves the gravitational constant (G), mass of the Sun (M), and the distance between the Earth and the Sun (r; this is the definition of one Astronomical Unit).

model training, and model evaluation. Model training takes about 15 minutes per epoch, and evaluating the model on the 500 test examples takes a few seconds.

Results

We make our entire dataset and benchmark available online [5] and information in the Supplementary Material. We test various approaches as a benchmark and perform multiple ablation studies. Table 3 compares nine different methods on a sample course (Introduction to Astronomy):

- Program synthesis: generalized use of Codex (Chen et al. 2021).
- Language models: GPT-3 (text-davinci-002 and text-davinci-003) (Brown et al. 2020) without and with chain-of-thought (CoT) prompting (Kojima et al. 2022a) and Jurassic-1 (Lieber et al. 2021).
- Wolfram Alpha: based on Mathematica (Wolfram 2021).
- Specialized models trained on astronomy and machine learning problems (Tran et al. 2021).

Language Models. Large language models have been noted for their ability to solve a surprising range of problems. A prime example of this is GPT-3 (Brown et al. 2020). We employ this model to provide an informative language-based baseline against our method. Here, we describe the results of this approach in four regimes. First, we use GPT-3 for completion as open-domain question answering. Second, we use it in a few-shot question answering setting: we select one question as an example with the solution and provide the answer explanation as a search context. Third and fourth, we replicate the first two approaches with a table of constants added as either (1) a question-dependent block of text (e.g. "π is 3.1416", etc.) or (2) a context document. Jurassic-1 (Lieber et al. 2021) is a recent large pre-trained language model that achieves high performance on many tasks, parallel to GPT-3. Both Jurassic-1 and GPT-3 fail without the table of conversions, and with the table provided, both answer 1/10 canonical questions correctly (approximately rounded).

Wolfram Alpha. We considered Wolfram Alpha (Wolfram 2021) as an additional baseline, given its ability to answer some math word problems. We test Wolfram Alpha on the Astronomy problems without any correct answers. Multiple wordings were attempted, as Wolfram Alpha can be sensitive to phrasing and formatting; even so, Wolfram Alpha fails to answer Astronomy course problems.

Learning to Learn Machine Learning. We tested the Astronomy problems on a model trained to solve machine learning problems (Tran et al. 2021). We received no correct answers, suggesting the expression trees required for solving machine learning course problems differ from those for solving the Astronomy course problems, as expected. Table 3 shows that language models, Wolfram Alpha, and a system trained on Machine Learning all fail to answer Introduction to Astronomy course questions and do not provide useful baselines; whereas our specialized model achieves 92% accuracy our generalized approach based on Codex achieves 67% accuracy on the same questions. The answers to the Introduction to Astronomy questions generated by our specialized model are in the Supplementary Material along with an ablation test to evaluate all models with and without the tabular data of mathematical constants and units while keeping all other factors equal.

[5] Link temporarily redacted. Please contact the authors.

ID	Course	Method	Question
4	Nonlinear Dynamics I: Chaos	Human	Find all the fixed points of the flow $\dot{x} = \sin(x)$
		Machine	Approximate the value of r at which the logistic map has a superstable 4-cycle. Please give a numerical approximation that is accurate to at least four places after the decimal point.
5	Fundamentals of Materials Science	Human	In discussing molecular rotation, the quantum number J is used rather than l. Using the Boltzmann distribution, calculate $\frac{n_J}{n_0}$ for $H_{35}Cl$ for $J = 0$, 5, 10, and 20 at $T = 1025$ K.
		Machine	A sample of an ideal gas is heated reversibly and adiabatically from an initial temperature of 300 K to a final temperature of 600 K. If the initial volume of the sample is 1.00 L, what is the final volume of the sample?
8	Principles of Chemical Science	Human	Using the table of mean bond enthalpies provided, predict the bond enthalpy (in kJ/mol) for the CO bond marked with an arrow in the molecule below. (Bond enthalpeis: C-H - 412 kJ/mol, C-C - 348 kJ/mol, C=C - 612 kJ/mol, C-O - 320 kJ/mol, C=O - 743 kJ/mol).
		Machine	10.0 mL sample of 0.20 M HNO_2 (aq) solution is titrated with 0.10 M NaOH (aq). (Ka of HNO_2 is 4.3×10^{-4}). Calculate the pH of the solution when 10.0 mL of 0.10 M NaOH has been added.
9	Signal Processing	Human	Determine the Laplace transforms (including the regions of convergence) of the following signal: $x(t) = (1 - (1 - t) * \exp(-3 * t))u(t)$.
		Machine	Find the inverse Fourier transforms of the following signal: $X(f) = (\frac{\pi}{2}) * \text{sinc}(\pi * f)$
13	Introduction to Astronomy	Human	Measurements of the radial recession velocity of five galaxies in a cluster give velocities of 9700, 8600, 8200, 8500, and 10000 km s^{-1}. What is the distance to the cluster if the Hubble parameter is $H_0 = 72$ km s^{-1} Mpc^{-1} ? Hint: Use the Hubble law for the average velocity of the members in the cluster.
		Machine	A star has a luminosity of $10^4 L_\odot$ and a temperature of 10^4 K. Find the star's radius in units of the Sun's radius.
14	Geobiology	Human	Determine the kind of nonmarine sedimentary deposits that reflects arid environmental conditions.
		Machine	Determine the term 'Precambrian shield' and the place where it is present in North America
15	Principles of Microeconomics	Human	Chloe consumes only books (x) and video games (y). Her preferences can be represented by the following utility function: $U(x, y) = x * (y^2)$. Calculate the Marginal Rate of Substitution (at an arbitrary bundle (x, y)).
		Machine	Suppose the demand for apples is $Q_D = 550 - 50 * P$ and the industry supply curve is $Q_S = -12.5 + 62.5 * P$. Calculate the equilibrium price and quantity.
16	Unified Engineering	Human	Define a thermoplastic and a thermoset.
		Machine	What is the difference between an isothermal and an isentropic process?
19	Theory of Numbers	Human	Tabulate the number of primes less than x, for $x = 10000$, 20000, ..., 100000. Also tabulate the number of primes less than x and of the form $4k + 1$, and the number of the form $4k + 3$.
		Machine	Find the smallest integer $n > 1$ such that n^2 divides the factorial of n.
20	Systems Microbiology	Human	What does proton motive force mean, and why is it important in biology?
		Machine	Describe the difference between a batch culture and a continuous culture.
22	Intermediate Calculus	Human	Find the value of $\frac{dz}{dx}$ at the point (1, 1, 1) if the equation $xy + z^3x - 2yz = 0$ defines z as a function of the two independent variables x and y and the partial derivative exists.
		Machine	Find the surface area of the portion of the paraboloid $z = 4 - x^2 - y^2$ that lies above the xy-plane.
23	Computer Architecture	Human	What is 01100110 XOR 00111011 in binary?
		Machine	The CPU runs an operating system kernel. A user process occupies the bottom half of the 32-bit address space (i.e., the lower addresses), while the kernel occupies the top half of the same address space (i.e., the higher address) What is the address of the first byte of the kernel?
24	Probability	Human	You have a group of couples that decide to have children until they have their first girl, after which they stop having children. What is the expected gender ratio of the children that are born? What is the expected number of children each couple will have?
		Machine	A fair coin is tossed repeatedly until a head is followed by a tail. What is the expected number of coin tosses?

Table 2: A subset of the human- and machine-generated questions for various courses. The table of questions for all of the 27 STEM courses is found in the Supplementary Material.

Id	Approach	Accuracy
1	Specialized model trained on astronomy	92%
2	Generalized use of Codex (Chen et al. 2021)	67%
3	GPT-3 (003) with CoT (Kojima et al. 2022a)	37.5%
4	GPT-3 (003) (Brown et al. 2020)	30%
5	GPT-3 (002) with CoT (Kojima et al. 2022a)	27%
6	GPT-3 (002) (Brown et al. 2020)	15%
7	Jurassic-1 (Lieber et al. 2021)	10%
8	Wolfram Alpha (Wolfram 2021)	0%
9	Specialized model trained on ML (Tran et al. 2021)	0%

Table 3: Comparison of accuracy of solving Introduction to Astronomy course questions using different approaches. The specialized approach trained on Astronomy achieves 92% accuracy. The generalized approach of turning the questions into the task of writing programs to solve the questions, and synthesizing the programs using OpenAI Codex, achieves 67%. GPT-3 (text-davinci-003) with chain-of-thought (CoT) prompting achieves 37.5%, and GPT-3 (text-davinci-003) without chain-of-thought prompting 30%. GPT-3 (text-davinci-002) with chain-of-thought (CoT) prompting achieves 27% and GPT-3 (text-davinci-002) without chain-of-thought prompting 15%. Jurassic-1 achieves 10%. Wolfram Alpha and a specialized model trained on a different course completely fail.

Automatically Explaining Solution Programs

We generate explanations by appending one of the following texts: (1) "Starting at the beginning of the code/function, please provide a line by line explanation of the above", (2) "Here's what the above code is doing: 1.", or (3) "# Explanation of what the code does", below a synthesized program, and running Codex.

Automatically Generating New Questions

We use the curated questions to generate new questions, as shown in Table 2. This is done by prompting Codex with the numbered human-written question and generating the next question. The generation process uses Codex to complete the next question. We set the Codex temperature parameter to 0.1 to avoid a deterministic or repetitive generation.

Student Survey

To evaluate the machine-generated questions, we conducted an anonymous online student survey comparing them with the human-written questions in terms of quality, appropriateness relative to the course, and question difficulty. We surveyed 12 STEM students who took one of the courses listed. The survey was optional and included informed consent, with the following description: "We are conducting a survey to assess the quality and difficulty of automatically generated questions. You will be presented with a series of questions, either human-written (taken from an actual course final exam) or machine generated, but you will not be told the source of a given question. For each question, you will be asked (a) whether you think the question is human-written or machine-generated, (b) whether the question is appropriate for the given course final, and finally (c) how you would rate the difficulty of the question. Please carefully read each question and answer to the best of your ability".

We randomly sampled one generated and human-written question for each of the 27 STEM courses. Students were asked to read these 54 questions in the survey, mixed and presented randomly, and then answer three questions for each: (1) "Is the question human-written or machine-generated?", (2) "Is the question appropriate or not appropriate for the specific course final?", and (3) "What is the question's difficulty level on a scale between 1 (easiest) and 5 (hardest)?". We ask the students to provide ratings and not to solve the questions. The results of our survey are as follows: Of human-written questions, students identified 62.7% of them correctly as human-written and 37.4% incorrectly as machine-generated. Of machine-generated questions, students identified 37.0% correctly as machine-generated and 63.0% incorrectly as human-written. The difficulty ratings are between 1 (the easiest) and 5 (the hardest). Students rated machine-generated questions with a difficulty level of 2.75 with a 0.89 standard deviation and human-written questions with a difficulty level of 2.94 with a 0.90 standard deviation. Students rated machine-generated questions as appropriate 89.2% of the time and human-written questions as appropriate 86.1% of the time. Survey participants considered machine-generated questions as likely to be human-written as human-written questions themselves. Participants considered machine-generated questions slightly easier than human-written questions. Finally, survey participants considered machine-generated questions slightly more appropriate for a course than human-written ones. We conclude that the machine-generated questions are indistinguishable across multiple aspects from human questions and, in cases, even better suited for the content of a university-level STEM course than human-written questions.

Limitations

These approaches cannot solve problems that rely on information in images, solutions that require proof, and computationally intractable problems, such as factoring large primes. This last category is not expected in STEM course assignment as students themselves would also be unable to answer them; however, many questions that students answer have computationally intractable generalizations. The specialized method trained on a specific course achieves good performance; however, it fails on other courses and does not scale.

Conclusions

This work takes the next step in understanding and filling in components required for expanding learning to learn courses to many STEM topics. Although we achieved high accuracy using our specialized approach, it does not scale to other courses. The generalized approach easily scales up to many STEM courses without retraining. We demonstrate that a generalized approach using few-shot prompts from the nearest embeddings generates new questions across diverse STEM courses indistinguishable from human-written questions. There is a clear path to learning all university STEM courses using large transformers and datasets with few-shot learning. We extended this work to the entire curriculum required for graduating from MIT EECS.

References

Alsubait, T.; Parsia, B.; and Sattler, U. 2016. Ontology-based multiple choice question generation. *KI-Künstliche Intelligenz*, 30(2): 183–188.

Bommasani, R.; Hudson, D. A.; Adeli, E.; Altman, R.; Arora, S.; von Arx, S.; Bernstein, M. S.; Bohg, J.; Bosselut, A.; Brunskill, E.; et al. 2021. On the Opportunities and Risks of Foundation Models. *arXiv preprint arXiv:2108.07258*.

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Ch, D. R.; and Saha, S. K. 2018. Automatic multiple choice question generation from text: A survey. *IEEE Transactions on Learning Technologies*, 13(1): 14–25.

Chen, M.; Tworek, J.; Jun, H.; Yuan, Q.; Pinto, H. P. d. O.; Kaplan, J.; Edwards, H.; Burda, Y.; Joseph, N.; Brockman, G.; et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 4171–4186.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv preprint arXiv:2010.11929*.

Drori, I.; Zhang, S.; Shuttleworth, R.; Tang, L.; Lu, A.; Ke, E.; Liu, K.; Chen, L.; Tran, S.; Cheng, N.; Wang, R.; Singh, N.; Patti, T. L.; Lynch, J.; Shporer, A.; Verma, N.; Wu, E.; and Strang, G. 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 (PNAS)*.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 1024–1034.

Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hendrycks, D.; Burns, C.; Kadavath, S.; Arora, A.; Basart, S.; Tang, E.; Song, D.; and Steinhardt, J. 2021. Measuring Mathematical Problem Solving With the MATH Dataset. *arXiv preprint arXiv:2103.03874*.

Kim, B.; Ki, K. S.; Lee, D.; and Gweon, G. 2020. Point to the Expression: Solving Algebraic Word Problems Using the Expression-Pointer Transformer Model. In *Conference on Empirical Methods in Natural Language Processing*, 3768–3779.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *International Conference on Learning Representations*.

Kojima, T.; Gu, S. S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022a. Large Language Models are Zero-Shot Reasoners. *arXiv preprint arXiv:2205.11916*.

Kojima, T.; Shane Gu, S.; Reid, M.; Matsuo, Y.; and Iwasawa, Y. 2022b. Large Language Models are Zero-Shot Reasoners. *arXiv preprint arXiv:2205.11916*.

Koncel-Kedziorski, R.; Roy, S.; Amini, A.; Kushman, N.; and Hajishirzi, H. 2016. MAWPS: A math word problem repository. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1152–1157.

Kurdi, G.; Leo, J.; Parsia, B.; Sattler, U.; and Al-Emari, S. 2020. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30(1): 121–204.

Kushman, N.; Artzi, Y.; Zettlemoyer, L.; and Barzilay, R. 2014. Learning to automatically solve algebra word problems. In *Annual Meeting of the Association for Computational Linguistics*, 271–281.

Lample, G.; and Charton, F. 2019. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*.

Lan, Y.; Wang, L.; Zhang, Q.; Lan, Y.; Dai, B. T.; Wang, Y.; Zhang, D.; and Lim, E.-P. 2021. MWPToolkit: An Open-Source Framework for Deep Learning-Based Math Word Problem Solvers. *arXiv preprint arXiv:2109.00799*.

Li, J.; Wang, L.; Zhang, J.; Wang, Y.; Dai, B. T.; and Zhang, D. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, 6162–6167.

Li, S.; Wu, L.; Feng, S.; Xu, F.; Xu, F.; and Zhong, S. 2020a. Graph-to-Tree Neural Networks for Learning Structured Input-Output Translation with Applications to Semantic Parsing and Math Word Problem. In *Conference on Empirical Methods in Natural Language Processing*, 2841–2852.

Li, S.; Wu, L.; Feng, S.; Xu, F.; Xu, F.; and Zhong, S. 2020b. Graph-to-Tree Neural Networks for Learning Structured Input-Output Translation with Applications to Semantic Parsing and Math Word Problem. In *Conference on Empirical Methods in Natural Language Processing*, 2841–2852.

Liang, Z.; Zhang, J.; Shao, J.; and Zhang, X. 2021. MWP-BERT: A Strong Baseline for Math Word Problems. *arXiv preprint arXiv:2107.13435*.

Lieber, O.; Sharir, O.; Lenz, B.; and Shoham, Y. 2021. Jurassic-1: Technical Details And Evaluation. Technical report, AI21 Labs.

McInnes, L.; Healy, J.; and Melville, J. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *Journal of Open Source Software*, 3(29): 861.

Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; et al. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3: e103.

Papasalouros, A.; and Chatzigiannakou, M. 2018. Semantic Web and Question Generation: An Overview of the State of the Art. *International Association for Development of the Information Society*.

Peng, S.; Yuan, K.; Gao, L.; and Tang, Z. 2021. MathBERT: A Pre-Trained Model for Mathematical Formula Understanding. *arXiv preprint arXiv:2105.00377*.

Qin, J.; Lin, L.; Liang, X.; Zhang, R.; and Lin, L. 2020. Semantically-Aligned Universal Tree-Structured Solver for Math Word Problems. *Conference on Empirical Methods in Natural Language Processing*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21(140): 1–67.

Rakangor, S.; and Ghodasara, Y. 2015. Literature review of automatic question generation systems. *International journal of scientific and research publications*, 5(1): 1–5.

Robitaille, T. P.; Tollerud, E. J.; Greenfield, P.; Droettboom, M.; Bray, E.; Aldcroft, T.; Davis, M.; Ginsburg, A.; Price-Whelan, A. M.; Kerzendorf, W. E.; et al. 2013. Astropy: A community Python package for astronomy. *Astronomy & Astrophysics*, 558: A33.

Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; and Catanzaro, B. 2019. Megatron-LM: Training multi-billion parameter language models using GPU model parallelism. *arXiv preprint arXiv:1909.08053*.

Tran, S.; Krishna, P.; Pakuwal, I.; Kafle, P.; Singh, N.; Lynch, J.; and Drori, I. 2021. Solving Machine Learning Problems. *Asian Conference on Machine Learning*.

Tsai, S.-h.; Liang, C.-C.; Wang, H.-M.; and Su, K.-Y. 2021. Sequence to General Tree: Knowledge-Guided Geometry Word Problem Solving. *arXiv preprint arXiv:2106.00990*.

Udrescu, S.-M.; Tan, A.; Feng, J.; Neto, O.; Wu, T.; and Tegmark, M. 2020. AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *arXiv preprint arXiv:2006.10782*.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16): eaay2631.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, 5998–6008.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2018. Graph attention networks. *International Conference on Learning Representations*.

Wang, L.; Zhang, D.; Zhang, J.; Xu, X.; Gao, L.; Dai, B. T.; and Shen, H. T. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7144–7151.

Wolfram. 2021. Mathematica.

Wu, Q.; Zhang, Q.; Fu, J.; and Huang, X.-J. 2020. A Knowledge-Aware Sequence-to-Tree Network for Math Word Problem Solving. In *Conference on Empirical Methods in Natural Language Processing*, 7137–7146.

Xie, Z.; and Sun, S. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In *International Joint Conference on Artificial Intelligence*, 5299–5305.

Xing, H.; Salleb-Aouissi, A.; and Verma, N. 2021. Automated Symbolic Law Discovery: A Computer Vision Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 660–668.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How powerful are graph neural networks? In *International Conference on Learning Representations*.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Wang, Y.; Shao, J.; and Lim, E.-P. 2020a. Graph-to-Tree Learning for Solving Math Word Problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 3928–3937.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Wang, Y.; Shao, J.; and Lim, E.-P. 2020b. Graph-to-Tree Learning for Solving Math Word Problems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 3928–3937.

Zhang, S.; Shuttleworth, R.; Austin, D.; Hicke, Y.; Tang, L.; Karnik, S.; Granberry, D.; and Drori, I. 2022. A dataset and benchmark for answering and generating machine learning final exams. In *Submitted*.