Research Article

# Could an LLM Qualify for the MIT Integration Bee? An AI Learning Experiment

**Max Gregg**[1]

1. Michigan State University, United States

**The MIT Integration Bee is a well-known competition showcasing students' proficiency in advanced calculus integration. This experimental study investigates whether large language models (LLMs)—including eleven versions of ChatGPT and Claude—can solve and evaluate integration questions sourced from fourteen years of MIT Integration Bee qualifier tests (2010–2024). Each model attempted to solve integrals, grade its own solutions, and then grade solutions generated by other models. A final "true" assessment was derived by comparing each response to official solutions. Results reveal that while certain specialized or "trained" models performed best at correctly solving integrals (up to 53% accuracy), smaller, lower-cost models could equal or surpass more expensive ones in cost-effectiveness. Additionally, LLMs showed limited ability to recognize their own earlier outputs during assessment, as reflected by stable or reduced self-confidence scores. These findings demonstrate the trade-offs between computational cost, accuracy, and reliability. There are implications for educators and researchers looking to leverage LLMs as automated tutors or graders in mathematics and beyond. Further work might explore advanced prompt-engineering methods such as Tree of Thoughts to improve solution quality and alignment.**

**Corresponding author:** Max Gregg, greggmax@msu.edu

## Introduction

In January at the Massachusetts Institute of Technology, the university's Independent Activities Period takes place. This short pause in normalcy allows students to focus on unique opportunities, projects, events, and competitions. Every year within the mathematics department, a competition called the MIT Integration Bee occurs, drawing in math-brained students to compete for the glory of being an integration champion[1]. The Bee has attracted internet fame by posting rounds of the competition as full videos to YouTube, drawing in millions of curious viewers[2]. The head-to-head style of the Bee puts the brilliance of these students on display as they attempt integration questions, many of which could have nuanced, tough, or highly specific answers. This Bee allows students to model their learning through the competition; in turn, it is an exhibition event for the attending crowd and the Internet for complex, knowledgeable integration methods.

This research experiment posits popular AI model products, with various parameters and constraints, against the Bee itself. These products attempt to solve the questions, grade themselves, and grade the outputs of other AI models. From this, we investigate how models 'align' to completing a task, understand grading and assessment, and what is the most 'optimal' AI across this experiment. There are conjectures and smaller findings that allude to the role AI may have in our existing and soon-to-be learning environments.

The emergence of the LLM world has changed the ability for students to learn and understand tough math concepts, like the core calculus concept of integration. Before the age of reliable, predictable LLM models, a student stuck on a tough math question may have turned to a search engine like Google or Bing for help, or an educational website like Khan Academy, Quizlet, or Chegg to find a similar solution, but now, a student can go to their favorite chat-based AI and ask it how to do the question and get help for their specific question on the spot.

This process is a winning strategy for students who want help with their specific question, desire a good tutor, or are simply on a time crunch and want the correct answer. And, indeed, there exists a lucrative time savings for math educators who can effectively use an AI to grade questions precisely, reliably, and accurately, and provide feedback to those students on why they got the grade. This experiment is a part of the journey that helps answer that: the cost, the sycophancy, the predictability, and the understanding of these models become ever more important with their adoption.

There exists a growing community of research and adoption of chat-style LLMs into classrooms as a part of their learning environment[3]. However, there does not exist contemporary research about the ability of ChatGPT and similar LLM products to be highly proficient in solving and casing tough math exercises, and poignantly, many tests of ChatGPT proficiency at such tasks rarely consider the different model specifications available to the task (such as a no pre-training scenario, heavily trained, or generically trained options). Additionally, an optimized, tailored learning experience can be made with ChatGPT using a GPT-generated textbook in its training to produce a student model[3], but there is curiosity about how GPT might accomplish similar tasks with a textbook not generated from its training dataset, and to work on tasks that occur outside of its textbook that uses the core task trained in the textbook, such as integration methods.

## Problem Statement

This research paper aims to communicate the findings of having various Chat LLM models perform the task of finding solutions to all the integration qualifying tests from the MIT Integration Bee. Additionally, these findings have impacts for specific academic groups to which we might allude across education, computational mathematics, machine learning, and beyond. Principally, this work is of interest to those who wish to apply LLMs to specific tasks, objectives, or functions, especially in the context of task gamification, education, and enterprise use of LLMs, where the cost of their deployment is critical. Across 300 questions and 14 years of the Bee's qualifier tests (2010-2024), different models are tested in directly solving and grading the Bee. Lastly, there is a lens to be had for learning environment design and the use of a chat-style LLM in the supplemental instruction of students in a mathematics course; this research identifies limits a GPT model may encounter in the task of assisting or showing a solution to a complex task or identifying a correct answer to a complex problem. A set of results is given here as well as in a GitHub Repository where the used data will be uploaded and a Jupyter Notebook file is hosted with .csv data of each model's result and the combined results.

## Literature Review

This experiment is novel but does build into the emerging discourse on the use of AI—specifically LLMs and Chatbots—in educational contexts. Researchers at Berkeley implementing an LLM interface called *EvalGen* found that, while useful, the LLM grading capability holds a strong level of subjectivity: "This raises a deeper epistemic question for evaluation assistants—is "alignment" an actualizable goal? To what extent does our common terminology and assumptions—e.g., that there is a "ground truth" set of labels we merely need to elicit—fail us?"[4]. Here, these researchers refer to the alignment to the grading standards the LLM used in tandem with a set of expert human graders. They compare the alignment of their grader to the alignment of a judge to the law and open the discussion to what a 'perfect' alignment means: "...subjectivity is not necessarily a sign of irrationality (contrasting with some imagined future AI that is entirely objective and rational, entirely “aligned” or “better” than humans). On the contrary: ‘There are good reasons to accept the imperfect in a judge’”[4].

This leads down another path: the usage of direct LLM products as graders, maybe with little to no task alignment, as a general grader. A programming course in Sweden was studied by researchers at the KTH Royal Institute of Technology, who were excited to find that the GPT grader proved to be of good use, if not notably inconsistent with human graders: “The results exceeded expectations, with GPT-4 achieving an overall accuracy of 75% when comparing its grades to those done by TA:s. However, these results should be considered with two caveats: first, that GPT-4 was significantly worse at predicting ’komplettering’ correctly, and secondly, that a significant part of incorrect grades was due to GPT’s and TA:s’ different interpretations of edge cases in a course with quite lenient grade requirements”[5]. In that, we see the same issue arise: alignment to the learning environment, to real human graders, and across queries tends to be inconsistent. Yet, the power of the model is evident, and the researchers at KTH are happy to share this: “If students were to receive high-quality feedback by GPT-4 on every task, it would not only take pressure off of TA:s but perhaps also impact the progression of their programming skills positively. Furthermore, another study conducted at KTH that gave automated feedback on students’ commit behaviour showed that students generally desired lengthier and more detailed feedback, something that presumably also applies to the implementations of the tasks themselves”[5]. This is, in essence, the draw of good LLM products to serious grading tasks; they can improve many aspects of the grading process, providing longer and detailed feedback that students desire, reducing the time burden of grading, and providing new opportunities to improve grading as a feedback process. This type of complex task is certainly suited to LLMs and their ability to tokenize to a task, but not to the seriousness and context of the binary outcomes found in education: pass and fail. In this, we have to investigate how LLMs arbitrate, deal with new decisions, and further prod at their alignment to tasks.

This speaks to the ethicality and usage of AI products in learning environments. Many of these themes are well summarized by researchers at Michigan State, who investigate a graduate course in engineering using ChatGPT-4, producing responses from the model based on a question bank derived from materials pertaining to the engineering course. After investigating its abilities and comparing how different levels of training affected its task efficacy, they concluded this idea: “While the rapid advancements in LLMs hold promise for revolutionizing educational practices, caution is advised.

Custom instructions, although powerful, may sometimes result in misleading or irrelevant outputs. Instructors and students are urged to critically assess the generated content rather than accepting it unquestioningly”[6].

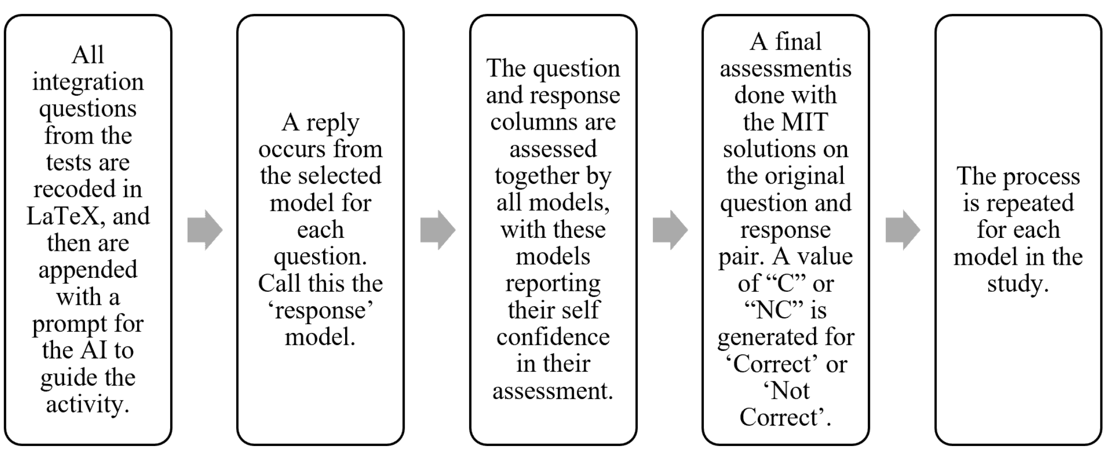
A major consideration in cross-assessment by LLMs is data leakage, and there are certainly questions on how using LLMs with shared or similar training datasets will perform in assessing tasks performed. Certainly, there is an expectation that GPT products, especially similar models, will have identical training datasets. However, it is curious to think about how data leakage in another LLM product, like Claude Sonnet 3.5, could be trained on the same information as ChatGPT products, and what the outcomes of this could be. It is difficult to measure this leakage, given the enormity and complexity of each model; however, the effects of the leakage may certainly be present.

# Methodology

## *Methodology 1.1: Research Design*

The research design structures the information posed to GPT in specific ways. Following a discussion, a demo is provided as a form of IDA (Introductory Data Analysis) and as a demonstration of the paths prompts follow in the development of a prompt and the cross-examination procedure that is described above. The diagram below shows the path a single integration question follows.

The flow of experimentation and capturing this data is represented below:




 doi.org/10.32388/GGTCEY 

*1. All integration questions from the tests are recoded in LaTeX, and then are appended with a prompt for the AI to guide the activity*

Each specific integration question is produced as a LaTeX string and stored as a list object inside of the Jupyter notebook. This is an example of how the prompt was designed. To handle processing, each set of prompts was produced in batches by the qualifier test year:

```
questions_2019 = [
"Provide the solution only for the following integral written in LaTeX format:$ \int_{0}^{2 \pi} \tan (\cos (x)) d x$",
"Provide the solution only for the following integral written in LaTeX format:$ \int \frac{x+1}{x(x+\log x)} d x$",
"Provide the solution only for the following integral written in LaTeX format:$ \int e^{x+e^{x}}+e^{x-e^{x}} d x$",
```

The prompt is designed to ask the LLM to be concise, mentioning that the response must identify if it thinks it is correct, and it's confidence in its assessment, and that's it. The reasoning is twofold, with a need to keep token cost low, and to also ensure the assessments have a level of possible comparison. When I asked ChatGPT about this, it agreed with me that it is more likely to focus on the correct task by limiting its output:

**Impact of Reduced Prompt Size**

- **Conciseness and Clarity:** Keeping the prompt concise can actually benefit the interaction by focusing the model's "attention" on the specific task at hand, potentially reducing processing time and making the responses more to the point.

This snip was grabbed on March 24, 2024, in a conversation with model ChatGPT-4. Reducing the token cost was a good choice for a novice researcher who does not have research funding, and fits the ultimate task of keeping the AI concise. This became a doctrine in all prompt engineering for this experiment: task limitation through prompt engineering constrains AI outputs in a positive way which allows for reproducibility, comparison, and efficiency in performing a task.

However—importantly—this doctrine does not reflect much of the literature surrounding having an AI prompted towards certain tasks. Researchers at Google DeepMind and Princeton University have a strong belief in a 'balanced' response: "...a thought should be "small" enough so that LMs can generate promising and diverse samples (e.g. generating a whole book is usually too "big" to be coherent), yet "big" enough so that LMs can evaluate its prospect toward problem solving (e.g. generating one token is usually too "small" to evaluate)"[7]. They show that a Tree of Thoughts (ToT) method greatly improves models with complex queries or returns: "Our experiments show that ToT significantly enhances language models' problem-solving abilities on three novel tasks requiring non-trivial planning or search"[7]. To thoroughly explore this would require many tokens to be generated and read: this is a worthy follow-up opportunity, but was not tested due to the burdensome computational and financial cost to test ToT prompts that generate many tokens. In this study, AIs are prompted to only respond with specific formats, or the direct solution.

### 2. *A reply occurs from the selected model for each question. Call this the 'response' model*

Each question is looped through the model's API, and a response comes back and is stored in a DataFrame. By now, we have the 'question' column, its associated 'response', the 'model' which made the response, with 'question_number', and 'qualifier_year' being added, as well. This is what the DataFrame object looks like:

| question | response | qualifier_year | model | question_number |
|---|---|---|---|---|
| Provide only the solution for the following in... | Here's the solution to the integral in LaTeX f... | 2010 | claude-3-5-sonnet-20240620 | 1 |
| Provide only the solution for the following in... | Here's the solution to the integral in LaTeX f... | 2010 | claude-3-5-sonnet-20240620 | 2 |
| Provide only the solution for the following in... | Here's the solution to the integral in LaTeX f... | 2010 | claude-3-5-sonnet-20240620 | 3 |

The AI-generated part of the DataFrame here is the 'response' column, which only knows the context from the individual question and prompt in the 'question' column.

*3. The question and response columns are assessed together by all models, with these models reporting their self-confidence in their assessment*

The model's response with its question is sent off to the API as a pair to be looped through all models we are examining. Here is the exact prompt used in the call:

```
def query_gpt_for_assessment(question, response):
    prompt = f"""
   Assess whether you believe the solution to the provided integral is true or false,
   and offer your confidence level to your own assessment, where a confidence level of 10 is very confident
   and a 1 is not very confident at all. Limit your output to only this format:
   "Prediction: True/False, Confidence: X."

    Question: {question}
    Solution: {response}
    """
```

Here is the assessment product associated with this step:

> " I am going to give you a set of integrals with solutions. Assess whether you believe the solution to the provided integral is true or false, and offer your confidence level to your own assessment, where a confidence level of 10 is very confident and a 1 is not very confident at all. Limit your output to only this format: "Prediction: True/False, Confidence: X."

Question 1:
\int_{2023}^{2025} 2024 , dx = 2024x \Big|_{2023}^{2025} = 2024(2025) - 2024(2023) = 4048000 - 4047472 = 528
Prediction: True, Confidence: 10

Question 2:
\int \frac{(x-1)^{\log(x+1)}}{(x+1)^{\log(x-1)}} , dx = \frac{(x-1)^{\log(x+1)+1}}{\log(x+1)+1} + C

Prediction: True, Confidence: 9

Question 3:
\int (x \log{x}+2x) , dx = \frac{x^2}{2} \log{x} + \frac{x^2}{4} + C
Prediction: True, Confidence: 10

Question 4:
\int \frac{dx}{x \log{x}+2x} , dx
Prediction: False, Confidence: 10 (This is just the original integral, not the solution)

Question 5:
\int_{0}^{2\pi} \arccos(\sin{x}) , dx = 2\pi
Prediction: True, Confidence: 9

*This example, captured from Claude Opus, March 2024 in the Claude product interface.*

*4. A final assessment, by an arbitrarily chosen model, is done with the MIT solutions on the original question and response pair. A value of "C" or "NC" is generated for 'Correct' or 'Not Correct'*

In the final process, all of the solutions from the MIT qualifier tests were parsed to LaTeX and then were stored as a list of strings, much like how the questions were stored:

```
solns_2014 = ["$ \\int_{1}^{e} \\log \\left(x^{2}\\right) d x=2$",
"$ \\int_{-9}^{9} \\sin (\\sqrt[3]{x}) d x=0$",
"$ \\int {0}^{\\infty} \\frac{d}{d x}\\left[e^{1+x-x^{2}}\\right] d x=-e$",
```

The Claude model Opus—and later, Sonnet 3.5 Turbo—was chosen arbitrarily to be the ultimate grader, who would parse "C" or "NC" for each question, response, and MIT solution set.

*5. The process is repeated for each model in the study, so that each model gets a chance to generate solutions, have every model reply to those solutions, and be graded "C" or "NC"*

*Methodology 1.2: Dataset Structure & Research Assumptions*

The column space of the dataset is given below:


doi.org/10.32388/GGTCEY

1. **Question**
    - The LaTeX formatted question from each MIT Integration Bee qualifier test
2. **Response**
    - The LLM-generated response to the string in ‘Question’ with this additional prompt text: “Provide the solution only for the following integral written in LaTeX format:”
3. **Qualifier Year**
    - The year of the MIT qualifier test used (there is only one qualifier test per year)
4. **Model**
    - The model that produced the ‘Response’ value in the same row
5. **Question Number**
    - The associated question number from the test
6. **Assessment GPT 3.5 Turbo**
    - An LLM-generated assessment of responses’ correctness to the question of the format “True/False, Prediction X” Where “True/False” is the GPT 3.5 Turbo’s designation of incorrect or correct of the question-response pair in the same row, and “Confidence: X” is the model’s own confidence in producing its own assessment. An engineered prompt, with the ‘Question’, ‘Response’, and instructional text is sent together to produce this value.
7. **Assessment 4o Mini**
    - All other columns that have ‘assessment’ follow the same prompt-and-return pattern described above.
8. **Assessment 4o**
9. **Assessment 4o -High Temp**
10. **Assessment 4o -Low Temp**
11. **Haiku Assessment**
12. **Sonnet Assessment**
13. **Sonnet 3.5 Assessment**
14. **Opus Assessment**
15. **GPT Assist 4o Mini Assessment**
16. **GPT Assist Optimized Book Assessment**
17. **MIT Provided Solution**
    - The LaTeX formatted solutions from each MIT Integration Bee qualifier test
18. **Response Correct?**
    - A final grade. The models ‘Claude 3 Opus’ and ‘Claude 3.5 Sonnet’ were used for this purpose. These final graders received

- This dataset will be made
    - Across all MIT Integration Bee qualifier tests
- Across each LLM model considered

- Different levels of training instances were considered within the experiment—would an AI chatbot do better solving the integrals if it were equipped with a calculus textbook from MIT?

For this, Professor Gilbert Strang's text *Calculus v1* was used[8]. On initial testing, the token cost of sending prompts to a full GPT-4 assistant equipped with the full MIT text was severe, averaging 30-40,000 tokens used per send. Here is a snippet of my conversation with ChatGPT-4 about how many computational costs in tokens were used by the assistant with a full textbook:

> Critically, many rows experienced some message like this: tokens per min (TPM): Limit 30000, Used 15833, Requested 21101. you can see how the limit was surpassed.

Given the token cost of ChatGPT 4o in April of 2024, this was deemed too expensive for this experiment. A more efficient AI product was needed to see this through, or a rethought way of using the textbook. An optimized version of Strang's textbook achieved this. The optimized textbook can be found in the associated GitHub repository, as can the ChatGPT conversation that helped make it. This token-reduced version of the textbook led to a great reduction in computational cost.

Two assistant types were settled on:

- a ChatGPT-4o assistant with the optimized version of the textbook
- a ChatGPT-4o Mini assistant trained with the full textbook.

The GPT-4o mini assistant used far fewer tokens with the full textbook when compared to its larger version. The compressed version of the book is a 37-page, dense PDF of highly tokenized information, about 18,824 separable tokens. This snippet shows how the book appears:

Optimized Calculus Textbook

CALCULUS GilbertStrang Massac huset t sInst it ut eofTechnology WELLES LEY- CAMBRIDGE PRESS Box 812060 WellesleyMA 02482 CALCULUS T h irdEdition GILBERTSTRANG Mass achusettsInstituteofTechnology WELLESLEY-CAMBRIDGEPRESS Box812060WellesleyMA02482 P re fa c e M y g oalistohelpyou learncalculus.Itisabeautifulsubjectan d itscentralideas arenotsohard.Everythingcomesfrom therelationbetweentwodifferentfunctions. Herearetwoimportantexamples: Function .1/ The distance acartravels Function .2/ Its speed Function .1/ The height ofagraphFunction .2/ Its slope Function(2)istellingus howquicklyFunction(1)is changing. Thedistancewill changequicklyorslowlybasedonthespeed. Theheightchanges

Which is interpretable to a human reader, but not informative. This compressed textbook is available in the GitHub Repository as "optimized_calculus_textbook_comprehensive.pdf".

### *Methodology 1.3: Assumptions*

For each new training instance to be fair, it is necessary for the models to not 'learn' from previous instances when sending each new instance to the API. In this, the AI models must be 'stateless': no previous conversation information or user data can be a part of the query, and each new query must be sent separately from the other. In investigating this, I directly ask a model from OpenAI's API this question:

In [4]:

```
### OpenAI API
## Note: using GPT-4o as comparison model as it is far cheaper than GPT-4 and GPT-4-turbo

OPENAI_API_KEY = "###"

open_client = OpenAI(
    # This is the default and can be omitted
    api_key=OPENAI_API_KEY,
)

chat_completion = open_client.chat.completions.create(
    messages=[
        {
            "role": "user",
            "content": "Hi ChatGPT, do you know anything from my previous query to you here in the API?",
        }
    ],
    model="gpt-4o-mini",
)

# Extract and print response
response_text = chat_completion.choices[0].message.content
print(response_text)

```

```
No, I don't have the ability to recall previous interactions or maintain any memory of past conversations. Each session is
stateless, meaning I don't retain any information once the conversation ends. How can I assist you today?
```

Here it confirms that it does not know any previous knowledge from previous conversations and achieves this ‘stateless’ quality that is desirable in this experiment. This query can be found at the beginning of the training notebook called ‘Responses_3.5_Turbo.ipynb’.

This evidence exists, too, for GPT assistants and Anthropic models. Here is confirmation from the ChatGPT 4o Mini assistant, grabbed in April of 2024:

- GPT Assistant:

**GPT_4o_OptimizedTextbook**

No, I don't have access to any previous queries or interactions. Each session is independent, so I can only respond based on the information provided in the current session. How can I assist you today?

- Anthropic Model (Claude 3 Opus):

In [7]:

```
### Anthropic API

import anthropic

anthro_client = anthropic.Anthropic(
    # defaults to os.environ.get("ANTHROPIC_API_KEY")
    api_key="sk-ant-api03-q4RTAziV_-uIYiuwN7N8OGU2N4WmQhl_QHNq-bzEApvsedMAnVNojQc1GLVW_k3VsGr17_7a5LfMoYOxdDndZQ-zkpFLgA
)
message = anthro_client.messages.create(
    model="claude-3-opus-20240229",
    max_tokens=1024,
    messages=[
        {"role": "user", "content": "Hi Claude, do you know anything from my previous query to you here in the API?"}
    ]
)
print(message.content[0].text) #.content[0].text permits the text to be called without 'ContentBlock'.
```

```
I don't have any memory of previous conversations or queries. Each interaction with me starts fresh, without any context car
ried over from before. So I'm afraid I don't know anything about your previous queries to me via this API.
```

This experiment also makes assumptions that temperature can be a predictor for a model’s ability to complete a task. The OpenAI models accept values from 0 to 2, but “1” is considered high temperature: “The sampling temperature, between 0 and 1. Higher values like 0.8 will make the output more random, while lower values like 0.2 will make it more focused and deterministic. If set to 0, the model will use log probability to automatically increase the temperature until certain thresholds are hit” (OpenAI). Given the documentation, these are the parameters used:

- For the low temperature ChatGPT model, temperature = 0.1 was selected.
- For the high temperature ChatGPT model, temperature = 0.9 was selected.

Only OpenAI’s ChatGPT 4o model was used, as it serves as a baseline model for many other points of comparison across this experiment.


doi.org/10.32388/GGTCEY

*Methodology 1.4: Communication of Results*

An associated GitHub Repository and Jupyter notebook will accompany the publication for others to explore the methods used to obtain these results. The following Jupyter files exist:

- LaTeX conversion notebook
- Response Generating Notebooks, named 'Response_{model name}.ipynb'.
- The data generated from each Response notebook, named 'df_{model name}.csv'
- An IDA & EDA file which makes most of the results section
- The.pdf of the calculus textbook for training purposes
- The.pdf of the optimized calculus textbook
- Links to the webpage where the qualifying tests from the MIT Integration Bee are found
- Links in the GitHub repository are provided to used products, to MIT, according to their licensure and use of Gilbert Strang's textbook and their usage of MIT Integration Bee materials, like the qualifier test.
- Disclosure: the Python code was generated or edited in part by OpenAI models ChatGPT-3.5 Turbo, ChatGPT-4, ChatGPT-4o; and Anthropic models Claude Sonnet 3.5 and Claude 3 Opus. The generations span a timeframe from January of 2024 to November of 2024.

## Results

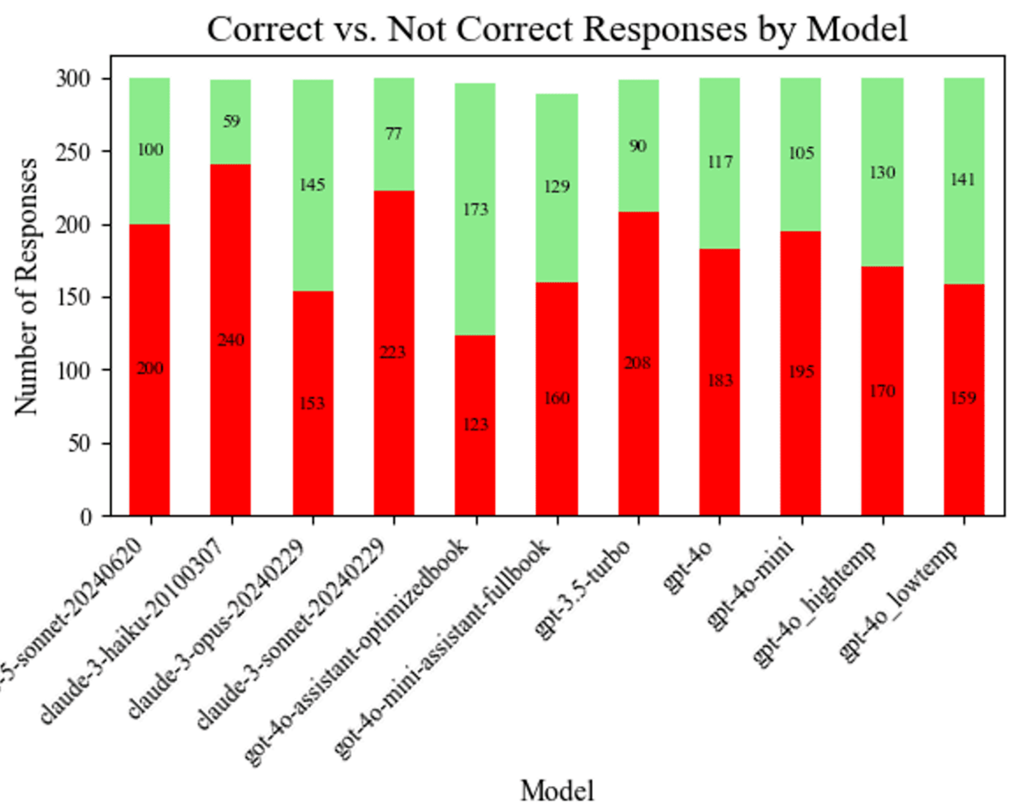
The results represented here are meant to be retestable and have the capacity—the calling—to be reapproached with new questions and insights. This is encouraged for readers to do, and all code/work is available in the GitHub Repository supplied with this paper, which contains the dataset, Jupyter notebook files, and accompanied LaTeX converted questions and answers from the MIT Integration Bee. Early experiments began in March of 2024, and results were produced over a period beginning in July of 2024 and were finalized in August of 2024. Each dataset was timestamped immediately after it was produced in the Jupyter notebook by the researcher. Results are given as a set of confusion matrices, plots, graphs, and that emphasize the binary "Correct/Not Correct" parts of the data. I will also follow a spot-checking process that checks the true-grading model—Claude 3.5 Sonnet, which was supplied the LaTeX solutions for each question on each qualifier test.

Each model's original answer was graded using the labels "Correct (C)" and "Not Correct (NC)". Considering a model deposited this label, it is important, too, that we sample the grades and ensure the grade it gave is consistent with a human labeler, which is given towards the end of the results section. The production code is available with the GitHub Repository. Note: much of the code was produced with the help or design of OpenAI or Claude products over this period. The range of models used was ChatGPT 4, ChatGPT4o, Claude Opus, and Claude 3.5 Sonnet.

### *IDA (Introductory Data Analysis)*

There are many ways to approach the data collected—and a series of questions that come with it. It might be best to examine the questions as points of curiosity.

### *Results, Topic 1: Models as Competitors in the Bee*




### *Results 1.1: which model identified the correct answer most often?*

The figure above shows that model ChatGPT GPT 4o Assistant, trained with an optimized textbook, received the best score at 53% (173/300) rate of earning the grade ‘Correct’. Both trained assistants were high performers, including the ChatGPT GPT 4o-mini assistant, which got 43% (129/300) correct. Outside of textbook training, Claude 3 Opus slightly won with a 48% (145/300) over ChatGPT

4o with a low temperature setting at 47% (141/300). Interestingly, *both* ChatGPT GPT 4o models with adjusted temperatures performed ~6% better (an average of 45% for both models) than the basic ChatGPT 4o model at 39% (117/300). Across all models, the average score gained was 38% (115/300).

Smaller and older models tended to do poorer: Claude 3 Haiku achieved 19% (59/300) correct, followed by Claude 3 Sonnet at 26% (79/300), and then GPT 3.5 Turbo at 30% (90/300). This suggests that larger models tended to be better, but only slightly, for the GPT-4o mini at 35% (105/300) only incorrectly 7 less observations than the full-size GPT-4o model.

To be sure if temperature is a worthy training parameter, retesting and investigating over more iterations to confirm is required, as there are too many exogenous variables between the time of training, existing user data in the API, and the possible temperature range. A range of temperatures across all models is a worthy point of study, but this approach is not central to this experiment.

The selected grader model, Claude 3.5 Sonnet, did not seem to appeal to its own output in giving 'final' grades, as Claude 3.5 Sonnet received a subpar grade of 33% (100/300). Some investigation was done by the researcher to confirm that the answers this final grader model gave were correct. At random, 25 rows were spot-checked to ensure the grade "C" or "NC" sensibly matched the MIT solution key's answer. 24 out of the 25 rows appear to have the appropriate label., with one directly incorrect label produced.

## *Results, Topic 2: Models as Graders of the Bee*

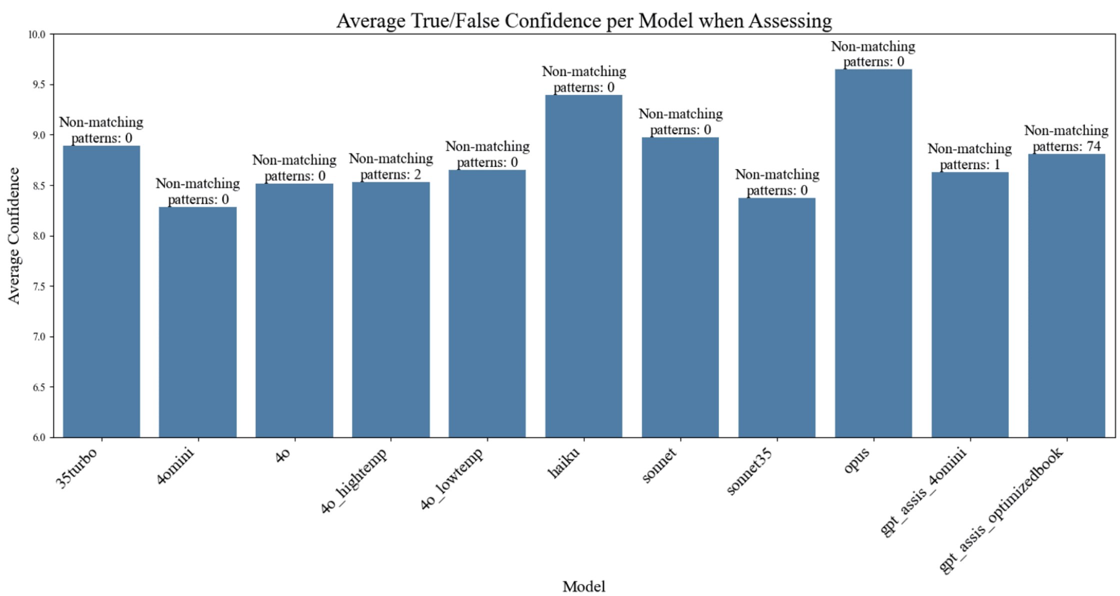
### *Results 2.1: Which model was most confident in its predicted assessment?*

After predicting if the given solution was True—correct—or False—incorrect—each assessing model was asked to give its confidence in making that prediction. This can be seen in the figure below. Claude Opus was the most confident in its assessment, giving on average a confidence of 9.64. The least confident assessor was ChatGPT 4o Mini, with an average of 8.28. This does not necessarily imply that larger models are more confident graders: Claude Haiku, one of the smallest and cheapest Anthropic models, had an average confidence of 9.40 in its replies. Temperature does not appear to impact confidence significantly; the trained assistants seem to have elevated confidence scores, but a significant increase over their respective base models is not observed.


doi.org/10.32388/GGTCEY

*Results 2.2: Do models tend to increase in confidence when assessing their own handiwork?*

Confidence was not significantly impacted whether or not a model was grading an answer it had previously generated. ChatGPT 3.5 Turbo was the least confident in its own work and had the greatest negative difference in confidence, dropping 0.225 in self-assessment. Claude Sonnet was the most confident in its own answers, giving itself a 0.245 higher average score over other models' answers. Important Note: the model is not aware of who produced the original answer. The assessment queries are sent to separate instances, but it's interesting to think that confidence does not rise with previously generated answers from the same model, given the expectation that the LLM's training dataset should largely be the same from instance to instance: in short, it appears it is hard for any of these models to recognize its own handiwork.




*Results 2.3, Confusion Matrices: How well did assessing models predict the correct assessment?*

Confusion matrices are used to represent the accuracy of the predictions. To do this, the produced label of 'True' or 'False' by the assessment task was compared to the 'Response Correct?' label 'C' or 'NC'. If the labels matched, a correct solution was recorded; if the labels did not match, Type I and Type II errors were recorded respectively.

 doi.org/10.32388/GGTCEY 

The matrix with the highest accuracy is the ChatGPT 4o Mini, at 60.43%, followed closely by the Optimized Textbook ChatGPT4o Assistant at 60.34% accuracy. This 4o-mini's performance is surprising, given the relatively small size and low training of the 4o-mini compared to other models: of all assessor models, it was the best predictor of the 'Not Correct' label. The worst-performing model was Claude Haiku, at 39.02%, and generally, the ChatGPT models outperformed the Claude models, with Claude Opus being the best at 56.98%. The Claude Haiku model, interestingly, was serially optimistic about the correctness of each solution, predicting 1,994 false 'correct' labels. The most 'pessimistic' grader was also the best, ChatGPT 4o Mini, predicting 733 false 'incorrect' labels across the dataset. The results of this experiment show that an untrained, smaller ChatGPT model was negligibly different in performance from a trained ChatGPT 4o Assistant at the task of assessing the correctness of a response without an answer key. All confusion matrices are shown below.


## *Results, Topic 3: Cost*

To do this experiment, more than 117 million tokens were consumed across both OpenAI and Anthropic products. Of this pool, about 5.3 million tokens belong to Anthropic, and about 112 million


doi.org/10.32388/GGTCEY

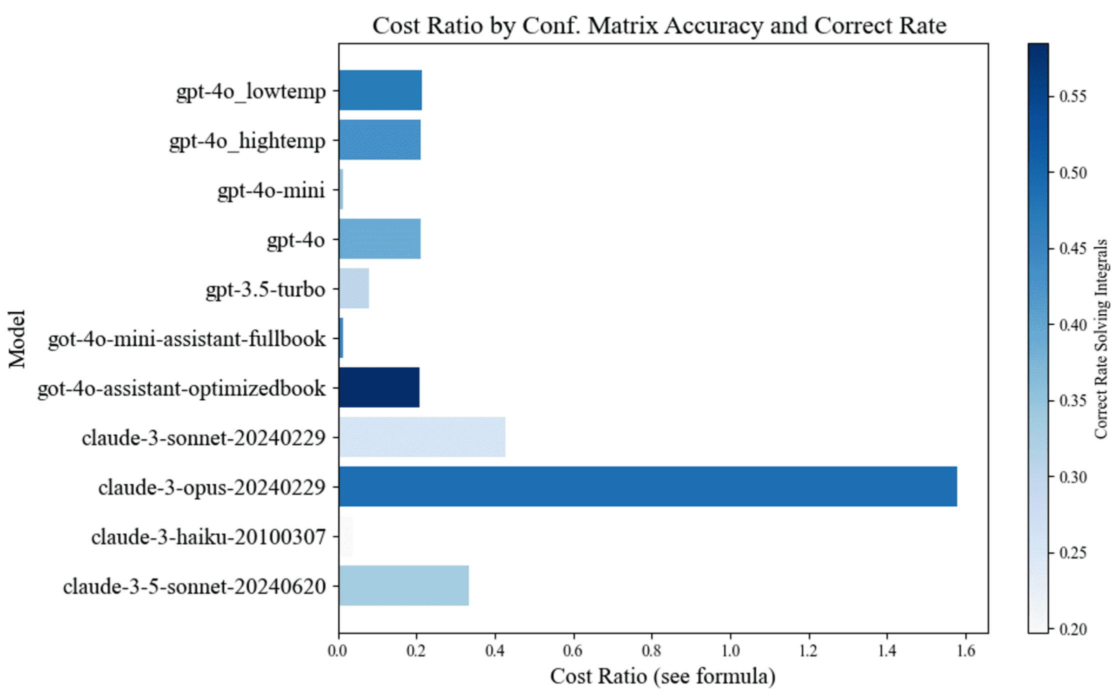
tokens belong to OpenAI. This distribution is representative of the bias in the number and types of GPT models used in the paper to test assistants and temperature. Of these totals, a staggering 94,000,000 tokens were used by ChatGPT 4o Mini alone—this is attributed to two things—the repeated use of the model as a 'tester' for experiment design and the token cost of searching the full-sized textbook to which the model had access. This elevated token cost is part of why a full-sized text wasn't with the GPT 4o Assistant.

In financial burden, $125.00 USD was consumed through the APIs to parse these tokens. Larger and more complex models certainly carried a higher price tag[9][10], which means we can compare the token cost to the success metrics of the models to evaluate the integrals.

### *Results, Topic 3.1: Cost Analysis: What was the most cost-effective model at the task (most success per dollar?)*

The figure below shows that smaller models significantly outperform larger ones at being cost-effective at this task:

The ratio seen in the figure was calculated using the formula below:




The ratio plotted here is shown in the equation below:


doi.org/10.32388/GGTCEY

$$\text{Model Cost Ratio } = \frac{\text{Input Cost per Million Tokens } + \text{ Output Cost per Million Tokens}}{\left(\text{Mean Correct Rate (\%)} \times \frac{1}{11}\right) + \textit{ Mean Conf.} \text{ Matrix Acc. (\%)}}$$

Where both input and output costs are summed, and the correct rate is properly reduced to $1/11^{\text{th}}$ the size of the conf. matrix values, as there are 3300 rows of generated assessments and 300 generated answers per model. It is unlikely that the use of input and output tokens is equal, and as this can vary greatly per query, these were directly summed. It is easy to see that Claude 3 Opus was much more expensive than other models, and generally, Anthropic API offerings were more expensive than the OpenAI counterparts. Temperature, nor Assistant training, greatly impacted these metrics, but one can see that the correct rate is slightly elevated for the low-temperature GPT 4o model over other GPT 4o's tested for nearly the same cost. This Correct Rate percentage color bar is the amount correct from solving the original integrals and does not account for the confusion matrices' performances in the color. Unaccounted here is the total token usage per model; while some models may be much more expensive, the number of tokens used is not documented. It could be inferred that larger and more expensive models correlate to a higher use of tokens, but this is not directly tested.

## Discussion and Future Work

This experiment aimed to be practical with no research budget, generate a unique and interesting dataset, and address many facets of how AI is as an assessor, at answering questions, and behaving as it pertains to education, mathematics, and the enterprise of using AI for solutions to mathematical questions. There do not exist yet robust practice schemes to validate or test AI for an application, and the methods to best validate AI as part of educational schemes are undeveloped across the discipline. The experiment here provides some ability to compare, contrast, and test AI as a preface to deploying its use in a task-oriented process in education.

The metric used to test the AI was the MIT Integration Bee, which proved to be a challenging and unique opponent for many AIs and was exciting to see the generated responses to its questions. A major limit to the experiment's quality is the directed query replies. Larger, more lenient, and complex queries that ask for a Tree of Thoughts reply style could improve the quality of the responses generated by AI[7]. This would have been a financially and computationally expensive addition to the research goal and would require additional resources, funding, and support. This would be an exciting opportunity for future work.

There are two findings that stood out above others: smaller, less computationally expensive models outperformed larger models at certain tasks in this experiment, and the mean reported confidence of AI did not rise significantly when grading its previously generated, and in some cases, fell. This alludes to the idea that AI does not recognize work it has previously generated, as it relates to solving or identifying a mathematical solution. This lack of mean confidence change may not hold with longer generations, where more language and syntax can be permitted. Interestingly, too, a theme was observed that overly confident models had a higher likelihood of a poor assessment of True/False; this theme is weakly supported and requires further exploration.

Using Confusion Matrices to evaluate the binary True/False metrics against the true Correct/Not Correct column is novel and fits a task where AI models must reply with a correct answer or predict a correct answer, and allowed us to examine pessimism and optimism in replies—seen in Claude 3 Haiku's affinity for the label 'True'—and works well with a prompt scheme that asked for a confidence value.

There are other implications and findings still available in the dataset, and it is worth exploring, adding to, and replicating. There is an identified weakness with this data, and that the exact time and place when the prompt was generated was not recorded per query. This could permit longitudinal work to explore model growth of this task over time.

## Statements and Declarations

### *GitHub Reference*

All data, snapshots, and materials here can be accessed at the following GitHub Repository link: https://github.com/othergandalf/Could-an-LLM-qualify-for-the-MIT-Integration-Bee-An-AI-Learning-Experiment_Repo?tab=readme-ov-file

### *Acknowledgements*

**Professor Micheal S. Murillo, at Michigan State University**: For your critical and valuable encouragement, and your insightful suggestions of how to design and produce this experiment.

**Yuan Yao, Massachusetts Institute of Technology:** Providing early-on LaTeX materials and letting me graciously use materials from the MIT Integration Bee qualifier questions for the work in this study.

## References


1. [^]MIT Integration Bee. (2024, March 24). "43rd Annual MIT Integration Bee [Webpage]." MIT. https://math.mit.edu/~yyao1/integrationbee.html.
2. [^]MIT Integration Bee. (2023, February 03). "2023 MIT Integration Bee - Finals [Video]." YouTube. https://www.youtube.com/watch?v=guRoWTYfxMs.
3. [a, b]Liang Z, Yu W, Rajpurohit T, Clark P, Zhang X, Kaylan A (2023). "Let gpt be a math tutor: Teaching math word problem solvers with customized exercise generation." arXiv preprint arXiv:2305.14386.
4. [a, b]Shankar S, Zamfirescu-Pereira JD, Hartmann B, Parameswaran AG, Arawjo I (2024). "Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences." arXiv preprint arXiv:2404.12272.
5. [a, b]Nilsson F, Tuvstedt J (2023). "GPT-4 as an Automatic Grader: The accuracy of grades set by GPT-4 on introductory programming assignments."
6. [^]Abedi M, Alshybani I, Shahadat MRB, Murillo M (2023). "Beyond traditional teaching: The potential of large language models and chatbots in graduate engineering education." Qeios. doi:10.32388/MD04B0.2.
7. [a, b, c]Yao S, Yu D, Zhao J, Shafran I, Griffiths TL, Cao Y, Narasimhan K (2023). "Tree of thoughts: Deliberate problem solving with large language models." arXiv. https://arxiv.org/abs/2305.10601.
8. [^]Strang G (2020). "Calculus v1." OpenStax.
9. [^]Anthropic. (n.d.). Models. Retrieved January 21, 2025, from https://docs.anthropic.com/en/docs/about-claude/models
10. [^]OpenAI. (n.d.). Pricing. Retrieved January 21, 2025, from https://openai.com/api/pricing/


**Supplementary data:** available at https://doi.org/10.32388/GGTCEY

## Declarations


**Funding:** No specific funding was received for this work.

**Potential competing interests:** No potential competing interests to declare.