Fine-tuning for Better Few Shot Prompting: An Empirical Comparison for Short Answer Grading

Joel Walsh[1,†], Siddarth Mamidanna[2,†], Benjamin Nye[1], Mark Core[1] and Daniel Auerbach[1]

[1]University of Southern California - Institute for Creative Technologies, Los Angeles, CA USA
[2]University of California, Santa Cruz, Santa Cruz, CA, USA

Abstract

Research to improve Automated Short Answer Grading has recently focused on Large Language Models (LLMs) with prompt engineering and no- or few-shot prompting to achieve best results. This is in contrast to the fine-tuning approach, which has historically required large-scale compute clusters inaccessible to most users. New closed-model approaches such as OpenAIs fine-tuning service promise results with as few as 100 examples, while methods using open weights such as quantized low-rank adaptive (QLORA) can be used to fine-tune models on consumer GPUs. We evaluate both of these fine-tuning methods, measuring their interaction with few-shot prompting for automated short answer grading (ASAG) with structured (JSON) outputs. Our results show that finetuning with small amounts of data has limited utility for Llama open-weight models, but that fine-tuning methods can outperform few-shot baseline instruction-tuned LLMs for OpenAI's closed models. While our evaluation set is limited, we find some evidence that the observed benefits of finetuning may be impacted by the domain subject matter. Lastly, we observed dramatic improvement with the LLama3.1 8B-Instruct open-weight model by seeding the initial training examples with a significant amount of cheaply generated synthetic training data.

1. Introduction

The widespread adoption of Massive Open Online Courses (MOOCs) and Learning Management Systems has created an increasing amount of learning assessments in online, machine-readable formats. Due to the volume of assessment responses and sometimes teacher-less nature of these courses, Automated Short Answer Grading (ASAG) has become a robust research target. Most attempts have used few or no shot learning with baseline LLMs. The practice of fine-tuning LLMs offers some promise as a way to increase the performance of LLMs on specific tasks, but the cost of compute and data collection budget are significant constraints. This paper explores finetuning under realistic constraints. Specifically, given a modest set of labeled examples (N=148) and a singleGPU training envelope, when does parameterefficient finetuning (QLoRA or OpenAIs inhouse tuning) yield statistically and practically meaningful gains over fewshot prompting for multiconcept ASAG? We find that the answer to this question is somewhat nuanced, as some methods of fine-tuning modestly improve ASAG in this context, while other methods do not.

Rather than train the models on a specific content domain, this fine-tuning approach trains across a varied set of domains and with different numbers of few-shot prompts, with the goal to tune a model that uses few-shot prompts more effectively for new content areas. With this specific task the model must assign binary labels for demonstrated understanding of user-defined concepts, or learning objectives. The evaluation set consists of expert human-graded responses from different domain areas. This particular type of structured JSON output is particularly

EvalLAC'25: 2nd Workshop on Automatic Evaluation of Learning and Assessment Content, July 26, 2025, Palermo, Italy

[†]These authors contributed equally to this work.

jwalsh@ict.usc.edu (J. Walsh); spmamida@ucsc.edu (S. Mamidanna)
0000-0002-5902-9196 (B. Nye); 0000-0002-0438-3868 (M. Core)

useful as agents and multi-agent systems have shown their usefulness at linking LLM outputs to software decisions.

1.1. Related Work

Deep learning for automated text scoring has been the focus of numerous public dataset challenges, but the approaches that have come from these competitions often focus on essay-length text and often ignore issues such as textual entailment [1]. Since the widespread adoption of neural methods in Natural Language Processing, there have been many attempts to utilize the latest network architectures for Short Answer Grading (SAG). These approaches included using LSTMs [2], mixing sentence level and token level embeddings [3], and a variety of transformers-based approaches [4] [5].

As LLMs began to perform well on benchmark tasks similar to ASAG, such as Question Answering [6], several groups began to experiment with leveraging the no-shot and few-shot capabilities of LLMs for ASAG [7] [8] [9]. Since the release of industry open weight models (e.g., Meta's Llama family), there has been limited research comparing ASAG performance with fine-tuned closed models (i.e., different sizes of Open AI's GPT-4) [10].

Supervised finetuning (SFT) and Instruction-tuned models [11] like ChatGPT have no doubt changed the world in the past few years; but training often requires a lot of gas (i.e., large numbers of examples). In the case of SFT, optimal prompt-response pairs is the gas. Since the proliferation of instruction tuned models, researchers have had success mitigating this need by using existing human-annotated data to drive reinforcement learning [12] and using high-quality examples as a seed to create additional SFT data [13].

Although our approach to these empirical studies on ASAG are similar, we expand these comparisons in a few areas. In this study, we also fine-tune a large open weight model for ASAG. We feel that this is an important departure as closed models do not disclose details about the parameter count, hyperparameters, or techniques used in fine-tuning. As many organizations are compute-limited, we focused on training quantized 4-bit models that can fit on one NVIDIA A40 GPU. We also study the effect of varying the amount N of few shot examples (i.e., N-shot) used to prompt each model at test time. We focus on producing JSON-structured outputs, which play an important role in agentic and LLM-based software. Lastly, we seed synthetic data using a small number of examples created by real subject matter experts and learners. This approach will enable low-resource educational organizations to create supervised finetuning data on a scale that makes the technology viable.

2. Methods

2.1. Fine-tuning and baseline models

For the closed models, we used OpenAI's GPT-4o-mini. OpenAI does not publish parameter counts for either of these models, or key information about the architecture. OpenAI has offered fine-tuning services starting in August of 2023[14]. The documentation for the fine-tuning API states that, "We typically see improvements from fine-tuning on 50 to 100 training examples" [15]. However, OpenAI does not allow you to download the fine-tuned models, or any of their models.

For the open models, we used LLama3.1 8B-Instruct [16]. Like GPT-4o-mini, this model is already instruction-tuned. The model size is 8 billion parameters, which is the largest model that can be finetuned on one NVIDIA A40 GPU with 48 GB of RAM using a Quantized Low Rank Adapter (QLoRA) [17] approach. Each model was evaluated on a set of 148 distinct short-answer labeled examples, which was effectively extended to 17820 prompts (Sec. 2.5), which took approximately 40-50 hours to run on our university's NVIDIA A40 GPUs. Due to

these compute limitations at test time we were not able to run extensive ablation tests on model architecture, or on hyperparameters like LoRa rank or number of hidden dimensions.

2.2. Data collection

Data was gathered with permission from the OpenTutor project [18]. OpenTutor allows curriculum creators to author tutoring dialogues, where assessment of concept understanding is conducted via multi-concept ASAG on student responses. Subject matter experts created the dialogues, and also graded student responses in order to improve the system. The subject matter of the training set pulled from technical subjects for which the models certainly had exposure (biology and computer science) to subjects where the models likely had much less exposure; or where best-practices may be changed over time, such as military leadership or culture. Each authored dialogue contains several concepts, or key points that the responses were supposed to address. For example, a dialogue on invasive species contains three concepts:

1. Monitoring and collecting data on the invasive species is key to fighting them,
2. Rapid or a quick response is another key to fighting invasive species, and
3. AI and machine learning can be used to find and track the spread of invasive species faster.

When a subject matter expert grades the answer, they provide an either 0 or 1 label for each concept; a 0 if the answer does not demonstrate knowledge of the concept and a 1 if it does. Some answers contain correct statements about one concept without mentioning the others; this is reflected in the grades as a skipped concept.

All Short Answer questions and training examples were generated as a byproduct of user interaction with the OpenTutor dialogue-based tutoring system. The corpus of graded responses for this analysis comes from multiple studies conducted with adult learners, to include both students and Mechanical Turk workers. The MTurk workers were screened for appropriate study behavior (e.g., enough time spent, expert human review of answers) Subject matter experts were then asked to grade each answer, specifying whether or not it addressed the concepts dictated by the expert.

2.3. Fine-Tuning Training Data

Subject matter experts labeled only a portion of user responses from OpenTutor dialogues; only labeled responses were used in the current research. The training data included all labeled responses from 42 lessons, excluding three lessons that were set aside for evaluation. Lessons without graded responses were omitted.

Due to varying levels of use and grading by experts, each lesson had a different number of graded responses (ranging from as few as 4 to over 100). To generate training data, multiple subsets of graded responses were randomly selected from each lesson, with each subset containing between 1 and 40 examples. Each subset was then individually paired with another distinct graded response from the same lesson, forming multiple complete training examples. Each of these pairings represented an n-shot example, where n denotes the number of responses in the subset.

Initially, the graded responses contained only binary labels (true/false) for each concept, without confidence scores or justifications. To enhance the quality and informativeness of the data, we used GPT-4o to generate justifications and assign confidence scores for each concept. Specifically, each concept label within a response was individually passed to GPT-4o along with the corresponding student response, generating a justification and confidence rating. To better calibrate the confidence ratings provided by GPT-4o, we averaged the confidence scores with those obtained from an existing logistic regression classifier previously trained on the same dataset [19]. This resulted in a more accurate reflection of the true confidence. All samples (a total of 148 training examples) were manually reviewed afterward to ensure their quality and correctness.

2.4. Generating Synthetic data

To extend this research beyond this small amount of hand-labeled examples, we also investigated the impact of training LLama3.1 8B-Instruct using synthetic data. To generate synthetic data, the training data was split into a 90:10 train-validation split. Google Gemini's 1.5 Flash model was used for generation. Our process involved randomly choosing 1-3 examples from either the train or validation, and appending them to a prompt to "generate one additional example from an academic, corporate, or military training domain". If the example did not form a valid JSON, it was thrown out. These one thousand examples fortified the existing training and validation sets. The test set remained entirely real data. For 1.5 Flash, this process cost 45 cents (US), offering a relatively cost-effective process. Surprisingly the latest Gemini thinking model, 2.5 Flash, could not create valid JSONs with any regularity. A limitation of this study is that the amount of synthetic data was chosen somewhat arbitrarily, as we did not have the test-time compute resources to test models trained on several different amounts of synthetic data. Also, we were unable to fine-tune GPT 4o-mini with synthetic data to see how it would affect GPT 4o-mini. This will be explored in future work.

2.5. Generating Evaluation Data

Evaluation data consisted of three gold lessons withheld from training, each covering a distinct topic:

1. Diode Breakdown (technical)
2. Reaching Out (leadership-focused military context)
3. Suicide Prevention (general risk factor awareness)

Each gold lesson had manually graded responses to ensure high evaluation quality: Diode Breakdown (50 responses), Reaching Out (50 responses), and Suicide Prevention (120 responses).

To simulate realistic scenarios of incremental data annotation and model updating, we evaluated the system using an N-shot strategy at different numbers of examples. For example, when N=5, five graded responses were appended to the prompt as context examples. To ensure robust evaluation despite limited available data, we structured the evaluation as follows (see Fig. 1):

1. Chunking. Responses from each gold lesson were grouped into test sets containing 10 responses each. This followed an approach similar to cross-validation, so for a lesson with 50 responses, there were 5 test sets.
2. Creating N-shot Examples. For each test set, multiple N-shot contexts were created by drawing graded examples from the remaining responses within that lesson. For instance, if a lesson contained 50 responses grouped into 5 test sets (each containing 10 items), each test set would have 8 distinct N-shot context sets. The total number of N-shot sets was calculated as (50 - 10) / 5 = 8 sets. This process was repeated for N=0, N=5, N=10, ..., and while the N-shot contexts sometimes overlapped with each other, test items never overlapped with any N-shot examples. Consequently, each test response was evaluated multiple times, each time across different values of N and N-shot contexts.
3. Multiple Trials for Robustness. When $N > 0$, we generated multiple evaluation trials (typically 10 trials per set) to ensure stability and reliability in our assessment. For each trial, we reshuffled the selected N-shot examples, not just altering their order but also simulating realistic variability by randomly ablating certain labeled concepts. This meant that, across trials, some contexts were richly annotated, while others were sparsely annotatedmimicking real-world annotation inconsistencies.
4. Comprehensive Evaluation. Every response in the test sets was evaluated across all trials and corresponding N-shot contexts. This extensive evaluation approach ensured a thorough and robust assessment, enabling confident observations and reliable conclusions. As we

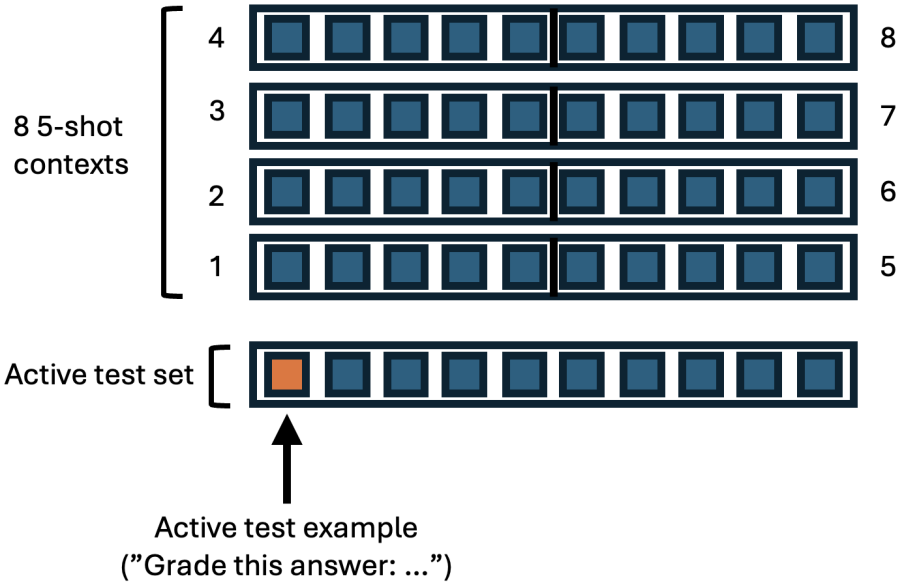

Figure 1: Visualization of evaluation structure for N=5 and 50 examples. All examples in the active test set consist of only the answer, while the n-shot examples contain the graded answers.

mentioned earlier the n-shot prompts also made evaluation somewhat slow (approximate 50 hours) for the open weight models, as self-attention inference scales quadratically with context window length.

Ultimately, this expanded our final test size across all lessons to be n=17820 examples, providing a rigorous and thorough evaluation procedure.

3. Results

Our analysis finds that, for this particular type of structured output, fine-tuning GPT 4o-mini on just ~ 150 examples was impactful, resulting in an F1 increase from 0.68 to 0.73. The largest improvement occurs in the Reaching Out and Suicide Prevention domains, which are based on highly specific content domains and whose evaluations require using labeled examples rather than prior knowledge. The objective of fine-tuning the model on a series of few-shot examples is to allow it to adapt far quicker to the grading criteria and knowledge in the few-shot examples, as opposed to relying on prior knowledge and idiosyncratic judgment. This alignment is not just about boosting raw scores; it also makes the models output more interpretable and consistent with human feedback loops. By internalizing each specific lesson's grading framework, the fine-tuned model better generalizes to new topics with lower shot counts, delivering reliable, rubric-compliant assessments.

Table 1
Metrics for Different Baseline and Fine-Tuned Model Configurations

Model	Accuracy	Precision	Recall	F1 Score
Base GPT-4o-mini	**0.812**	**0.936**	0.550	0.681
Fine-Tuned GPT-4o-mini	0.789	0.749	**0.741**	**0.735**
Baseline Llama-3-8B	0.587	0.493	0.106	0.165
Fine-tuned Llama-3-8B (3 Epochs)	0.586	0.287	0.046	0.078
Fine-tuned Llama-3-8B (6 Epochs)	0.571	0.459	0.373	0.408
+ (w/ Gemini 1.5 synthetic training data)	0.721	0.663	0.650	0.653
Fine-tuned Llama-3-8B (9 Epochs)	0.554	0.435	0.336	0.376

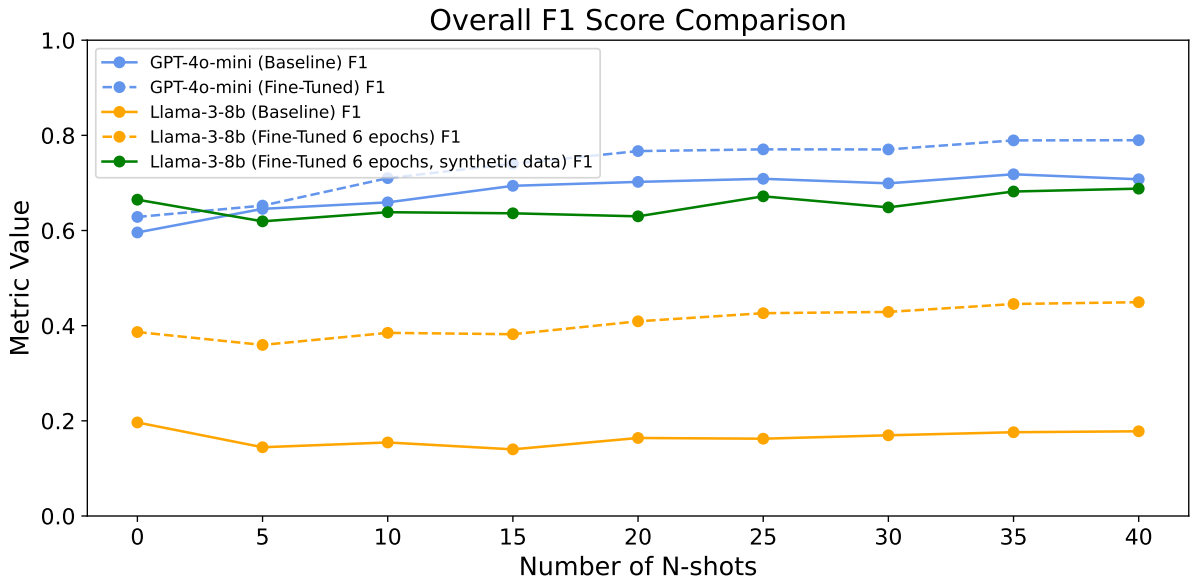

Figure 2: F1 scores for baseline and fine-tuned models (Using best Llama model, 6 epoch)

Additionally, the fine-tuned GPT 4o-mini model's F1 score improved as the number of N-shots increased. The F1 score graph for the Diodes lesson is impressive (Fig. 3), as it shows that while the base model had a higher initial F1, it did not improve with increasing N-shots. However, the F1 for the fine-tuned model started significantly lower than the base and progressively improved with the number of N-shots, eventually matching the performance of the base model. Looking at the overall F1 score comparison (Fig. 2) also displays the fine-tuned model adapting to new data more effectively than the base model, as the F1 score increases at a slightly higher rate as the number of N-shots increase for the fine-tuned model.

On the other hand, fine-tuning LLama3.1 8B-Instruct using QLORA on the initial data was not as successful. The baseline and 1 epoch fine-tuned model sometimes failed to generate stopping points, entered phases of repeating, and mainly predicted one class - False. While the model did show some signs of life at epoch 6, the F1 score was still only 0.408. This still however, represents a large gain on the baseline model (see Table 1). The fine-tuning (for 6 epochs) ultimately provided a significant increase in performance over the base model. At 9 epochs, the model's performance began to degrade, suggesting that the threshold for overfitting had been reached. Adding in the synthetic data caused the most dramatic improvement. The best model in both instances was 6 epochs, and boosted the F1 score from 0.408 to 0.653, nearly matching the baseline GPT 4o-mini model. This would suggest that synthetic data can be incredibly effective, and also that this family of models requires much more data than 50-100 examples in order to grasp structured ASAG tasks like this.

4. Discussion and Future Research

While the Llama QLORA fine-tuning did not work as well as the GPT 4o mini fine-tuning, there are some major advantages to tuning and serving open-weight models. For one, OpenAI will never let a user download the model weights or architecture. Any system built on OpenAI's technology will have vendor lock-in, and a dependence on an internet connection. LLama3.1 8B-Instruct with synthetic data was close to GPT 4o-mini performance and trending enough in a positive direction to justify serving this model as a viable alternative.

Recent research into structured LLM outputs has shown that constraining LLMs to structured

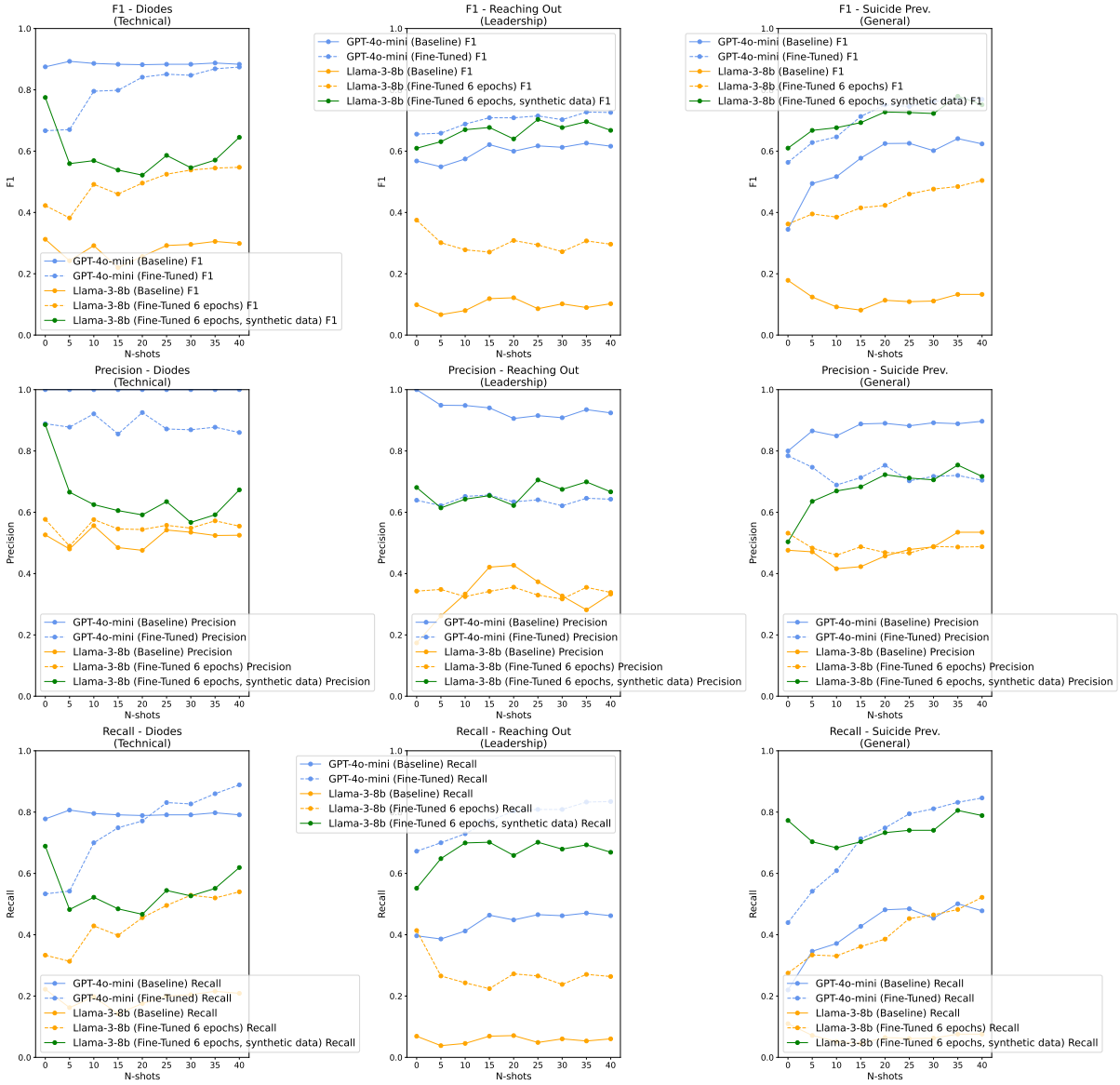

Figure 3: Performance metrics across different text domains within the training data

outputs can have a deleterious effect on model reasoning and domain knowledge capabilities [20]. Future research will explore using an agentic process, whereby a model is prompted to provide a free-form justification and a grade, and then a agent is prompted to parse the information into a JSON using some combination of handwritten rules or additional LLM calls. Building learning engineering systems with structured outputs is a constant battle with the noisy output of LLMs. Agents built on handwritten rules or additional LLM calls can also serve as means to tweak "almost there" responses that might be off by a data type, or a slightly incorrect key value.

Other levers for improvement include prompt complexity and the size of both synthetic and real training datasets. The real dataset required content creation, human short answers, and a round of subject matter expert grading. This is time consuming and expensive to collect. As an alternative to collecting thousands of real training examples at great cost or generating synthetic data, a potential alternative could be to use the Deepseek approach of “cold start” supervised fine-tuning followed by Group Relative Policy Optimization (GRPO) reinforcement learning to further refine the model [21]. This process is thought to be much more sample efficient than supervised finetuning.

In terms of training with synthetic data, this process can optimized substantially. The prompt could be optimized, the amount of data could be increased, and much more advanced models than Gemini 1.5 Flash now exist. However, the iterative nature of data annotation could lend itself quite well to simulation by multi-agent system, where teacher and student agents with different profiles, prior knowledge, and directives work in conjunction to create new examples. This could create higher-quality synthetic examples.

5. Conclusion

This study explored realistic approaches to supervised fine-tuning of LLMs for automated short answer grading tasks. Our findings have shown that fine-tuning large commercial models (such as GPT-4o-mini) with relatively small amounts of data enhances performance over baseline few-shot prompting methods, even in highly context-dependent domains. Meanwhile, QLORA finetuning LLama-3-8B models initially showed poor performance for this particular task, but additional infusions of synthetic data made them competitive with GPT models.

Overall, the effectiveness of this synthetic data-infused training approach holds great promise for developing reliable ASAG systems that can function without relying on large corporations, server-scale GPUs, or the presence of an Internet connection. Continued exploration into synthetic data generation and hybrid agentic pipelines suggest further performance gains. Collectively, these approaches can enable resource-limited educational organizations to deploy and use strong ASAG technologies, helping democratize access to reliable AI-driven assessment methods.

6. Acknowledgements

The project or effort depicted was or is sponsored by the U.S. Government under contract number W912CG-24-D-0001 and W911NF-14-D-0005, as part of the USC ICT UARC and the AI Research Center of Excellence in Education (AIRCOEE). The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

References

[1] D. Alikaniotis, H. Yannakoudakis, M. Rei, Automatic text scoring using neural networks, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, 2016, p. 715725. URL: https://www.aclweb.org/anthology/P16-1068.

[2] S. Kumar, S. Chakrabarti, S. Roy, Earth mover's distance pooling over siamese LSTMs for automatic short answer grading, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2017, p. 20462052. URL: https://aclanthology.org/D17-1217.

[3] S. Saha, T. I. Dhamecha, S. Marvaniya, R. Sindhgatta, B. Sengupta, Sentence level or token level features for automatic short answer grading?: Use both, in: Natural Language Processing and Information Systems, volume 10859 of Lecture Notes in Computer Science, Springer International Publishing, Cham, 2018, pp. 35–47. doi:10.1007/978-3-319-91947-8_3.

[4] T. Liu, W. Ding, Z. Wang, J. Tang, G. Y. Huang, Z. Liu, Automatic short answer grading via multiway attention networks, arXiv preprint arXiv:1909.10166 (2019). URL: http://arxiv.org/abs/1909.10166.

[5] Z. Wang, A. S. Lan, A. E. Waters, P. J. Grimaldi, R. G. Baraniuk, A meta-learning augmented bidirectional transformer model for automatic short answer grading, in: Proceedings of the 12th International Conference on Educational Data Mining (EDM 2019), International Educational Data Mining Society (IEDMS), 2019, pp. 156–163. URL: https://educationaldatamining.org/files/conferences/EDM2019/papers/paper_156.pdf.

[6] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, arXiv preprint arXiv:2005.14165 (2020). URL: http://arxiv.org/abs/2005.14165.

[7] S.-Y. Yoon, Short answer grading using one-shot prompting and text similarity scoring model, arXiv preprint arXiv:2305.18638 (2023). URL: http://arxiv.org/abs/2305.18638.

[8] J. Schneider, B. Schenk, C. Niklaus, Towards LLM-based autograding for short textual answers, arXiv preprint arXiv:2309.11508 (2024). URL: http://arxiv.org/abs/2309.11508.

[9] R. Ivanova, S. Handschuh, Evaluating LLMs' performance at automatic short-answer grading, in: Proceedings of the Workshop on Automatic Evaluation of Learning and Assessment Content (EvalLAC 2024), volume 3772 of CEUR Workshop Proceedings, CEUR-WS.org, Recife, Brazil, 2024, pp. 79–86. URL: https://ceur-ws.org/Vol-3772/paper10short.pdf.

[10] I. Chamieh, T. Zesch, K. Giebermann, LLMs in short answer scoring: Limitations and promise of zero-shot and few-shot approaches, in: Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024), Association for Computational Linguistics, Mexico City, Mexico, 2024, p. 309315. URL: https://aclanthology.org/2024.bea-1.28.

[11] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, Q. V. Le, Finetuned language models are zero-shot learners, arXiv preprint arXiv:2109.01652 (2022). URL: http://arxiv.org/abs/2109.01652.

[12] Z. Chen, Y. Deng, H. Yuan, K. Ji, Q. Gu, Self-play fine-tuning converts weak language models to strong language models, arXiv preprint arXiv:2401.01335 (2024). URL: http://arxiv.org/abs/2401.01335.

[13] A. Zhu, P. Asawa, J. Q. Davis, L. Chen, L. Hanin, I. Stoica, J. E. Gonzalez, M. Zaharia, BARE: Leveraging base language models for few-shot synthetic data generation, arXiv preprint arXiv:2502.01697 (2025). URL: http://arxiv.org/abs/2502.01697.

[14] OpenAI, Introducing improvements to the fine-tuning api and ex-

panding our custom models program, https://openai.com/index/introducing-improvements-to-the-fine-tuning-api-and-expanding-our-custom-models-program/, 2023. Accessed on 2025-07-08.

[15] OpenAI platform, https://platform.openai.com, ???? Accessed on 2025-07-08.

[16] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, ..., Z. Ma, The llama 3 herd of models, arXiv preprint arXiv:2407.21783 (2024). URL: http://arxiv.org/abs/2407.21783.

[17] T. Dettmers, A. Pagnoni, A. Holtzman, L. Zettlemoyer, QLoRA: Efficient finetuning of quantized LLMs, arXiv preprint arXiv:2305.14314 (2023). URL: http://arxiv.org/abs/2305.14314.

[18] B. D. Nye, R. Sanghrajka, V. Bodhwani, M. Acob, D. Budziwojski, K. Carr, W. R. Swartout, Opentutor: Designing a rapid-authored tutor that learns as you grade, in: The International FLAIRS Conference Proceedings, volume 34, 2021. URL: https://doi.org/10.32473/flairs.v34i1.128576. doi:10.32473/flairs.v34i1.128576.

[19] B. D. Nye, R. Sanghrajka, V. Bodhwani, M. Acob, D. Budziwojski, K. Carr, W. R. Swartout, Opentutor: Designing a rapid-authored tutor that learns as you grade, in: The International FLAIRS Conference Proceedings, volume 34, 2021. URL: https://doi.org/10.32473/flairs.v34i1.128576. doi:10.32473/flairs.v34i1.128576.

[20] Z. R. Tam, C.-K. Wu, Y.-L. Tsai, C.-Y. Lin, H. yi Lee, Y.-N. Chen, Let me speak freely? a study on the impact of format restrictions on performance of large language models, arXiv preprint arXiv:2408.02442 (2024). URL: http://arxiv.org/abs/2408.02442.

[21] DeepSeek-AI, D. Guo, D. Yang, H. Zhang, et al., Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, arXiv preprint arXiv:2501.12948 (2025). URL: http://arxiv.org/abs/2501.12948.

A. Prompt template, N=30

Listing 1: Abridged sample of 30-shot prompt

```
{

The user provided an answer to a tutoring question. The answer is provided in JSON
    format. You are a tutor who is evaluating if the answer is sufficient to show
    that the user knows a one or more "concepts" which will be labeled "concept_1"
    to "concept_N". For each concept you evaluate, you must also express how
    confident you are in your evaluation of how well the answer shows knowledge of
    each concept. You will also provide a brief justification.

These are the concepts for this lesson.
{
  "concept_1": "They're much less likely to commit suicide now or in the future,
      because suicidal urges last less than an hour.",
  "concept_2": "The person is still at risk compared to other people, because they
      still have suicidal thoughts."
}
This is my answer provided in JSON to be evaluated.
{
  "answer_text": "it lowers"
}
Please respond in the following format:
{
  "answer": {
    "answer_text": "string // State the text of the particular answer being
        classified.",
    "concepts": {
      "concept_N": {
        "is_known": "string // true or false. If the input answer implies that the
            concept is known, the classification should be true. Otherwise it should
             be false.",
        "confidence": "float // A 0 to 1 score indicating certainty that a
            classification is correct. Confidence scores closer to 1 represent
            higher certainty, and confidence scores closer to 0 represent lower
            certainty.",
        "justification": "string // Why you believe the user answer is or is not
            sufficient to determine if they know the concepts."
      }
    }
  }
}
Only respond with the JSON output in the exact format of the template and no other
    words or symbols. The output must be valid JSON. Check that the output is valid
    JSON.

Here are some examples that have already been labeled (although they may not be
    fully labeled). They are presented in JSON format, where the answer is given,
    followed by a concept and a true or false label. Consider these to be ground
    truth examples.
{
  "answer_1": {
    "answer": "It'll deter them from wanting to commit suicide.",
    "concept_1": "true"
  },
  "answer_2": {
    "answer": "they're still more likely as they have suicidal ideation",
    "concept_1": "false"
  },
  .
  .
  .
```

```
  "answer_30": {
    "answer": "they may not have anything else to use",
    "concept_1": "false"
  }
}
```

B. Synthetic Content Generation Algorithm

Algorithm 1 Synthetic Content Generation Algorithm

```
function GeneratePrompt(few_shot_examples)
    prompt ← "You are an expert AI assistant tasked with generating high-quality, diverse educational content.", "Your goal is to create new examples similar in structure and intent to the provided examples, but tailored for specific domains.", "Please generate ONE new example. The new example MUST be in valid JSON format, mirroring the structure of the examples shown below.", "The content of the new example should belong to one of the following domains: academic education (e.g., K-12, university courses, research), corporate training (e.g., employee onboarding, skill development, compliance), or military education (e.g., tactical training, leadership development, technical skills for defense).", "Ensure the generated examples are distinct and cover a wide range of topics and styles appropriate for these domains.", "Here are some examples to learn from (pay attention to their JSON structure and content style):"
    for each example in few_shot_examples do
    prompt ← prompt + "\n— Example —\n" + ToJson(example)
    prompt ← prompt + "\nNow, generate a new, unique example..."
    return prompt
function GenerateFromModel(model, prompt)
    response ← model.generate(prompt, GenerationConfig, SafetySettings)
    json_object ← ExtractJson(response.text)
    return json_object
procedure Main
    originalExamples ← LoadExamples(InputFile)
    WriteExamples(OutputFile, originalExamples)
    generatedCount ← 0
    failedAttempts ← 0
    model ← new GenerativeModel(ModelName)
    while generatedCount < TargetCount and failedAttempts < MaxFailedAttempts do
        k ← RandomInt(1, 3)
        fewShotSamples ← RandomSample(originalExamples, k)
        promptText ← GeneratePrompt(fewShotSamples)
        newExample ← GenerateFromModel(model, promptText)
        if newExample is valid then
            AppendExample(OutputFile, newExample)
            generatedCount ← generatedCount + 1
            failedAttempts ← 0
        else
            failedAttempts ← failedAttempts + 1
    Print("Process finished. Total generated: " + generatedCount)
```