EECS101 Chatbot: Fine-tuning LLaMA2 on EdStem Question and Answer Data using LoRA

EECS 182: Deep Learning Final Project, UC Berkeley, Prof. Anant Sahai

David Babazadeh Giuseppe Perona Sonika Vuyyuru

***Abstract*—Our hypothesis is that it is feasible to fine-tune a Large Language Model (LLM) to generate answers to questions specifically related to EECS 101 concepts. The fine-tuning process is anticipated to result in a language model capable of formulating coherent and contextually relevant responses to questions that are typically answered in the EECS 101 EdStem page. Furthermore, we expect more accurate results on a model that is pre-fine-tuned on a general Q&A dataset.**

Another one of our hypotheses is that there are optimal lora-rank values in the LoRA hyperparameter range that yield optimal performance for our specific task. Finding a lower rank alternative is expected to enhance the computational efficiency and allow us to spend more computation on training time to hopefully get better results. For our experiment, we test lora-rank values (1, 4, 16, 32) on 50 data points and 3 epochs (without allocating extra epochs to lower rank trials) to see when the effectiveness drops off.

The third hypothesis is that if we pre-fine-tune on the StackExchange-QA dataset before fine tuning on the EECS 101 dataset, then we can get even better results since our specialized dataset is small, and doing QA questions first might ease the model into the flow of Ed style questions.

Our results showed that we were able to successfully fine-tune the LLaMA model to provide responses specific to EECS 101 at Berkeley. However, even after our different approaches to fine-tuning, we found the responses to often have inaccurate and and logical errors when compared to the un-fine-tuned LLaMA model. We attribute this to being potentially due to forgetting during fine-tuning, although LoRA is known to mitigate this issue.

***Index Terms*—finetuning, LoRA, chatbot**

I. INTRODUCTION

State-of-the-art large language models aren't designed to answer questions within specific small colloquia–we want to explore the possibility of fine-tuning a popular LLM, LLaMA-2 [1] for a local class forum on a low computational budget. Specifically, we'll be answering questions from our Electrical Engineering and Computer Science (EE/CS) Department's EECS 101 EdStem (Ed) forum (https://edstem.org/us/courses/23247/discussion/) and exploring the capability of the Parameter Efficient Fine-Tuning (PEFT) method called Low-Rank Adaption (LoRA) [2]. If our chatbot produces decent answers, we can give students quick, automated answers on advising questions which can save time for the staff, and stress for the students, or at least give a quick reply before someone can give a better reply. This would hopefully be impactful for the UC Berkeley EECS Budget Crisis which impedes the learning of students.

A more general goal we have is to give insight to aid the process of fine-tuning with LoRA on a low budget for other question-answer fine-tuning projects. We will look into closer-to-optimal hyperparameters (LoRA-r, described later), how many question-answer pairs might be sufficient, and if pre-fine-tuning on a larger question-answer dataset from Stack Exchange (SE) could produce better results for anyone looking into fine-tuning on a custom question-answer (QA).

II. METHODS

A. Dataset

a) H4 Stack Exchange Preferences Dataset: The HuggingFaceH4/stack-exchange-preferences (SE Dataset) [3] consists of Stack Exchange questions with at least two answers. The dataset consists of 10.8M examples and takes up over 22 GB, and the questions come from 343 different Stack Exchanges including Stack Overflow and Math Stack Exchange. Thus, because of memory constraints, we did not train on the entire dataset. Instead, we ran experiments, which will be explained in the Experiments section, to determine the appropriate number of training examples to use.

The training examples consist of raw HTML code scraped from stackexchange.com. This means that several HTML formatting tokens are included, such as paragraph markers and hyperlinks. However, we want our model to output English text. Thus, we preprocessed the questions and answers by rendering the HTML with BeautifulSoup's HTML renderer, then copied the generated text.

More preprocessing was necessary, as each question provided comes with several provided answers. For simplicity of training, we chose to use only the first answer for each question.

b) EdStem Dataset: The first step of this project was to systematically gather and process question-and-answer pairs from the Edstem platform, aiming to create a dataset for subsequent analysis. The primary objectives were to copy and paste these interactions into CSV files, accommodating scenarios where questions had multiple answers or where additional information was provided through replies.

For questions with multiple answers, each distinct answer was treated as a separate data point to ensure granularity in the dataset. In instances where answers included replies containing either complementary or disagreeing information, a principled approach was employed. Complementary information was concatenated to the original answer, enriching the dataset, while instances of disagreeing information led to the creation of new data points to accurately represent diverse perspectives.

For each individual datapoint, we started by getting the title of the post to be the start of the text data point. Then, we concatenated the body of the post. Finally, we added the appropriate answers, as described above.

The final step was to format the data in the required format to feed into Llama 2. This included placing the appropriate start tokens and separators between the question and answer portions, in this format: " $<s>$[INST] Question [/INST] Answer $</s>$". In total, we created 201 data points in this format.

Originally, we wanted to try to web scrape the Edstem page to get many data points. However, we ended up doing this manual process. If we see that we need more data points later on in the process, we may need to revisit and do more research on the web scraping process.

We split up our EdStem dataset into training, validation, and test data with a 90-5-5 split, using the validation set for our hyperparameter sweeping and test data for final model evaluation.

B. Base Model

For this project, we have selected the Llama-2-7b, a state-of-the-art language model, as our foundational tool [1]. This model is distinguished by its robust architecture, comprising 7 billion parameters, making it exceptionally adept at processing and generating natural language. In addition, we use the Hugging Face framework for a user-friendly API to interface with the model.

A critical aspect of our computational framework is the incorporation of the LoRA (Latent Low-Rank Adaptation) methodology. Utilizing LoRA allowed us to enhance the computational efficiency of the Llama-2-7b model with limited compute resources.

C. Fine-tuning Approach

We used Low Rank Adaptation (LoRA) to fine-tune our model. LoRA is known to provide good results even on limited training data, an issue we had on our Ed Dataset, as we had to collect it by hand. Furthermore, LoRA is computationally light when compared to other fine-tuning methods, using 10,000 fewer parameters and 3 times less GPU RAM than a full fine-tune with Adam on GPT-3 [2].

LoRA works by passing SGD updates through a new matrix product AB for each weight matrix $W \in \mathbb{R}^{n\times m}$, where $A \in \mathbb{R}^{n\times r}$ and $B \in \mathbb{R}^{r\times m}$, and $r << d$, where the result of the layer is now $W + AB$. This forces AB to have a much smaller rank than W, and thus a much smaller number of weights to train.

D. Experiments

We designed three experiments to explore different aspects of our problem and to find the best fine-tuning strategy for our model. Specifically, we wanted to explore the optimal LoRA hyperparameters, we wanted to explore if and by how much increasing the number of EdStem training data points helped, and we wanted to explore if and by how much pretraining on a more general Q and A style dataset, like a Stack Exchange Dataset, helped.

For all of our experiments, we used the AdamW optimizer and a learning rate of 2e-4.

1) Hyperparameter Sweep Across LoRA Rank r
The LoRA hyperparameters were systematically varied, and the models were rerun to assess their impact on performance. During all of these runs, we held the number of EdStem training points used constant, performing fine-tuning with 200 datapoints. Performance was measured using the evaluation metrics described later in this paper, along with qualitative analysis of the responses.
The LoRA r values we tested were 1, 4, 16, and 32. After our hyperparameter search, we chose an r value of 16. We did find better evaluation results for $r = 1$, however, we chose $r = 16$ because we saw qualitatively better results from our validation set prompt responses with the higher r value. Our results for this hyperparameter sweep can be seen in Figures 1 and 2. These results make intuitive sense, as the model should have been able to capture more complex behavior over the limited Ed Training phase.
Furthermore, we found that the value of r had little effect on the training loss. In Figures 3-6, we see that the loss curves for the training on the Ed Dataset look almost identical across r values. This further motivated our choice of 16 over 1 as the LoRA rank value, as it serves as more evidence that increasing r to a higher value will not hurt model performance.
2) Adjusting number of EdStem training data points
In this experiment, we wanted to see if increasing the number of EdStem training points used to finetune the model had a significant effect on the model performance. Specifically, this type of experiment will help us see if it is worthwhile to build a larger dataset for any future training. We were hoping to see if there was a point of diminishing return in terms of performance improvement at different number of data points. Furthermore, depending on the results, we could see if getting more datapoints would really be helpful, and if that could be incorporated in future training given more compute.
3) Pre-finetuning on Stack Exchange Dataset
We used the SE Dataset to pre-fine tune our model to address our limited Ed dataset. We hypothesized that pre-fine tuning on a similar context would help model performance. Therefore, we pre-fine-tuned with 2500 examples from the SE Dataset.

E. Model Evaluation

We attempted to assess our model by evaluating how "good" the model's responses were compared to the EdStem responses. After researching many different options, such as perplexity score and even human evaluation, we landed on two metrics:

a) Recall-Oriented Understudy for Gisting Evaluation (ROUGE): This metric measures the similarity between a candidate text and reference text by comparing the number of overlapping n-grams (successive token sequences) [3].

- Rouge1: ratio of 1-grams that are shared
- Rouge2: ratio of 2-grams that are shared
- RougeL: accounts for the longest common subsequence of n-grams

b) BERT: BERTScore leverages the pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity. This method demonstrates a significant correlation with human evaluations, both at the sentence and system levels. Moreover, BERTScore computes precision, recall, and F1 measure, which can be useful for evaluating different language generation tasks.

F. Our Models

This paper discusses and compares a selection of models outlined here:

- **Un-fine-tuned LLaMA** This is the LLaMA-2-7b-chat-hf model available on HuggingFace that all of our models use as a base model.
- **[50, 100, 200] Ed fine-tuned LLaMA** These models, trained on variable numbers of EdStem prompts were used to do the hyperparameter sweep over the LLaMA rank.
- **[500, 2500] SE, 200 Ed fine-tuned LLaMA** These models were trained on a larger set of Stack Exchange examples, and trained on the full Ed training set, and evaluated on the full Ed test set. These, in addition to the Un-fine-tuned LLaMA control were used to test the effectiveness of the pre-fine-tuning procedure. We often omit the number of EdStem examples these models were trained on because it does not vary.

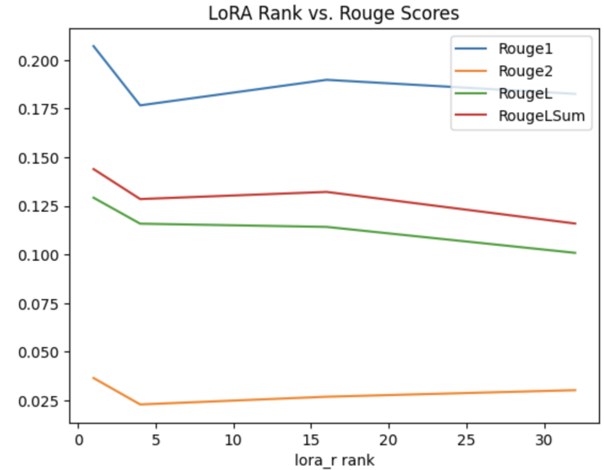

Fig. 1. LoRA Rank vs. ROUGE scores

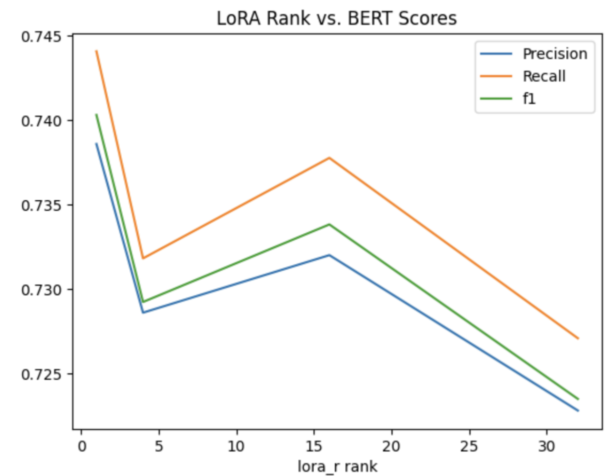

Fig. 2. LoRA Rank vs. BERT scores

III. Results

A. Experiment 1: Hyperparameter Sweep Across LoRA Rank r

Figures 1 & 2 are charts depicting our model performance across different values of LoRA r. We evaluated the model fine-tuned on 200 EdStem points with each hyperparameter using the BERT and ROUGE performance metrics.

One observation from Figures 1 & 2 is that the ROUGE scores are generally constant across rank were as the BERT scores seem more opinionated. This suggests that ROUGE might be a bad metric for our task. For any case, there's always the possibility that rank is less important than we think for our task, but related works seem to believe rank is a significant hyperparameter. The BERT scores suggest that higher ranks are actually working against our goal. Later, we'll see that un-fine-tuned LLaMA performs better in our human assessments. The reason why we see better performance with lower ranks is likely because having a higher rank gives the LoRA module more room to distort results from un-fine-tuned LLaMA or more room to forget. It's possible that we simply needed to train for longer to see the fruit of LoRA, but we were compute constrained and we wouldn't have been able to run the experiments, regardless we see the loss converging in Figures 3 through 6.

Despite our original procedure to find ideal hyperparameters before proceeding with other experiments, we decided it would be most informative to continue using relatively higher ranks to explore how LoRA operates in the other experiments, rather than experimenting with a model that uses as little fine-tuning as possible.

Figures 3 through 6 are the loss graphs from training the model with different LoRA r hyperparameter values. Note that the loss curves for all different LoRA ranks look very similar. This could be due to the fact that we did not shuffle the training data.

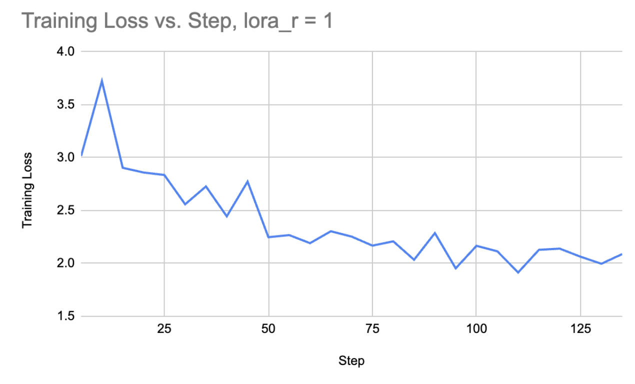

Fig. 3. LoRA Rank = 1 Loss

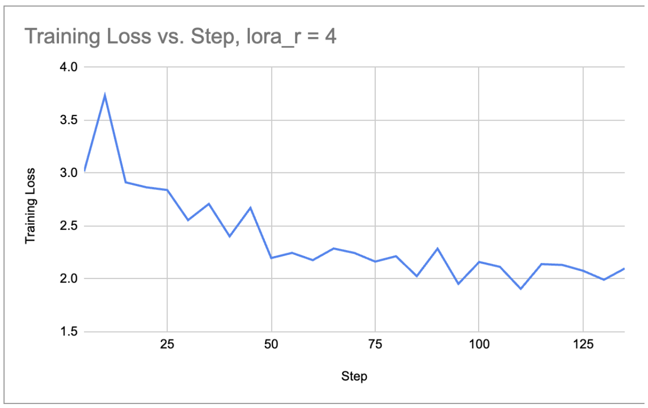

Fig. 4. LoRA Rank = 4 Loss

B. Experiment 2: Adjusting number of EdStem training data points

In Figures 7 and 8 we show the results of the ROUGE and BERT metrics on models fine-tuned on different numbers of EdStem training points. Out of the tested values, we saw the best performance across metrics on the model trained on 200 data points, which was the largest number of data points. This result makes sense since our data were so limited. It is unlikely

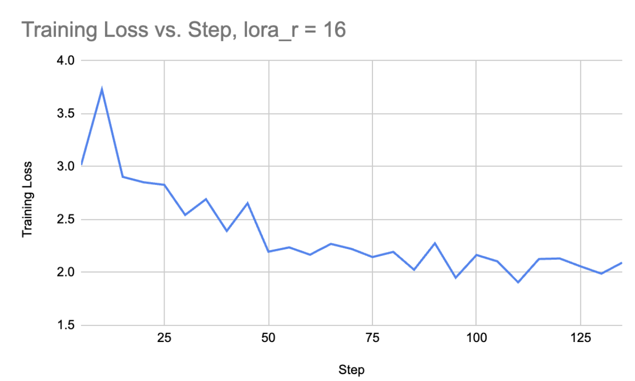

Fig. 5. LoRA Rank = 16 Loss

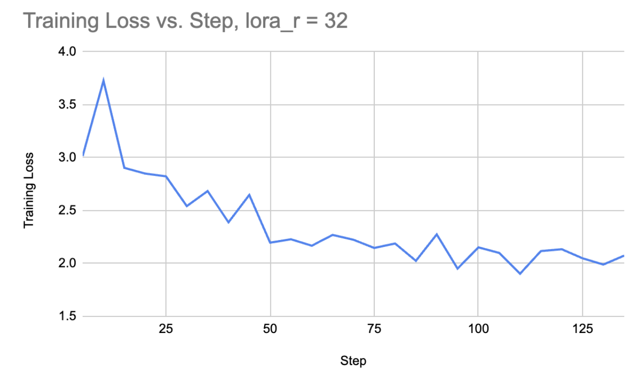

Fig. 6. LoRA Rank = 32 Loss

that the model was able to fully train all the parameters it has on the limited dataset. As such, it seems like it could benefit from more datapoints, and we would have to continue to test with more data points to reveal whether the model could be optimized further. These results show that it would in fact be worth it to continue to train with larger numbers of EdStem data points to see its effects on performance.

The cross-metric dip at 100 training points is also of interest. This could point to an initial forgetting phenomenon which is not overcome by improved domain knowledge that comes with further training.

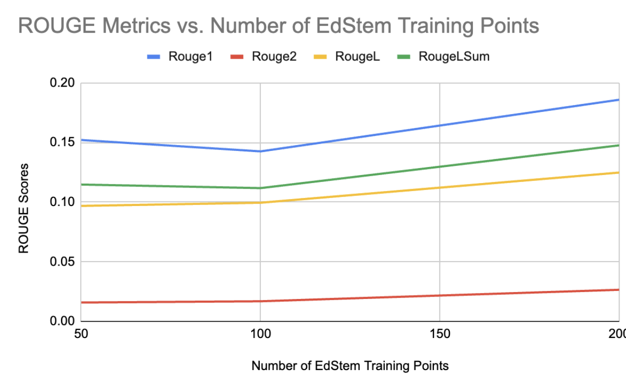

Fig. 7. ROUGE scores vs Number of Ed Training Points

C. Experiment 3: Pre-finetuning on Stack Exchange Dataset

TABLE I
EVALUATION METRICS FOR UN-FINE-TUNED LLAMA

rouge1	0.1245145307
rouge2	0.01634005691
rougeL	0.07904992992
rougeLSum	0.0975774247
bertPrecision	0.6810488224
bertRecall	0.731363818
bertF1	0.70439125

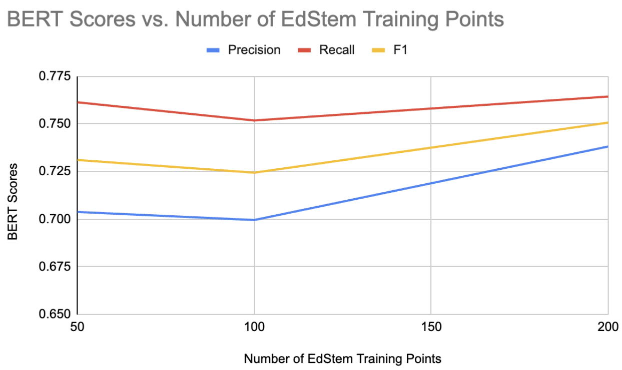

Fig. 8. BERT scores vs Number of Ed Training Points

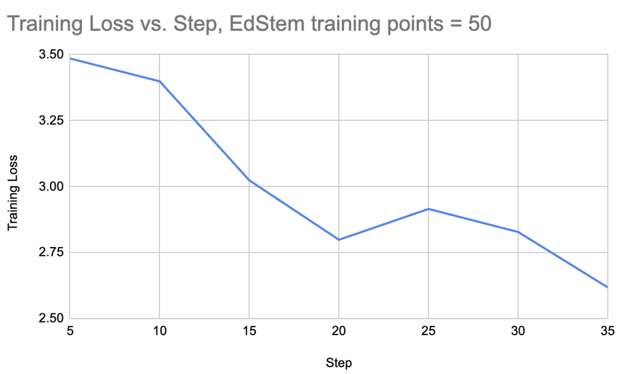

Fig. 9. Training Loss with 50 Ed Training Points

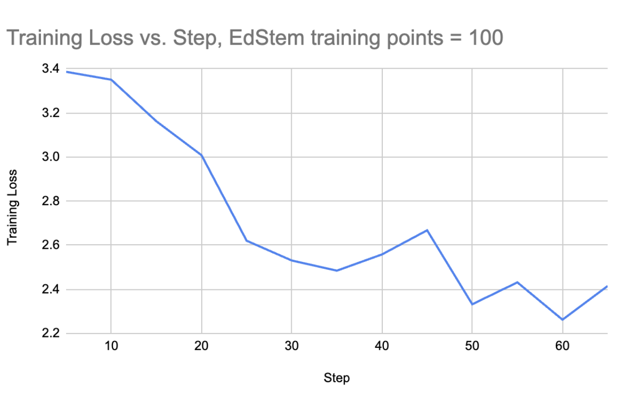

Fig. 10. Training Loss with 100 Ed Training Points

TABLE II
EVALUATION METRICS FOR PRE-FINE-TUNED AND FINE-TUNED LLAMA

rouge1	0.1420730171
rouge2	0.02713517158
rougeL	0.09315772787
rougeLSum	0.1164268572
bertPrecision	0.6958273381
bertRecall	0.7295427084
bertF1	0.7116111994

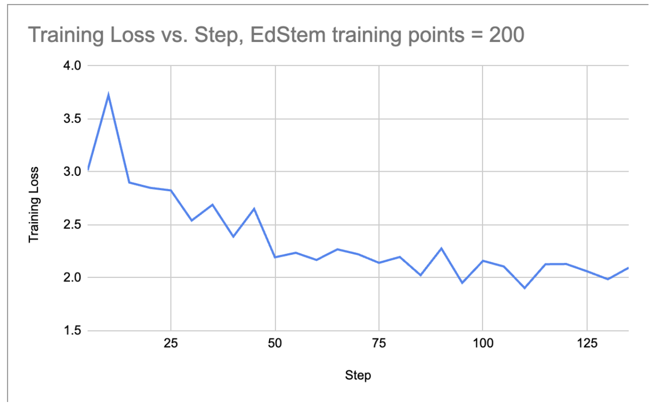

Fig. 11. Training Loss with 200 Ed Training Points

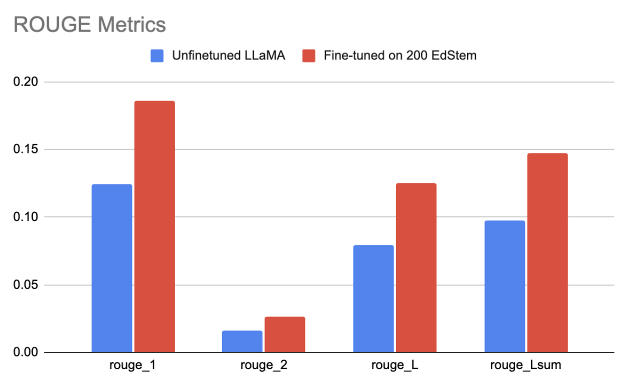

Fig. 12. ROUGE Metrics Graph: un-fine-tuned vs. Ed tuned Only

Figures 14 and 15 show a comparison of performance metric values for the two different versions of the model being tested in this experiment, that is the un-fine-tuned LLaMA model and the pre-fine-tuned and fine-tuned model. The pre-fine-tuned and fine-tuned models are pre-fine-tuned on different numbers of SE data, and then further fine-tuned on the fixed full set of 200 EdStem datapoints.

As shown, the BERT score metrics are pretty similar

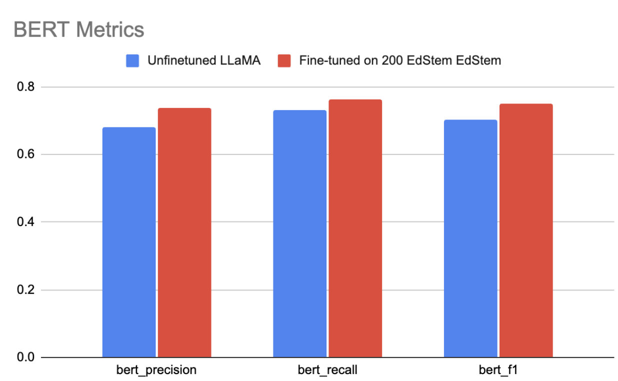

Fig. 13. BERT Metrics Graph: un-fine-tuned vs. Ed tuned Only

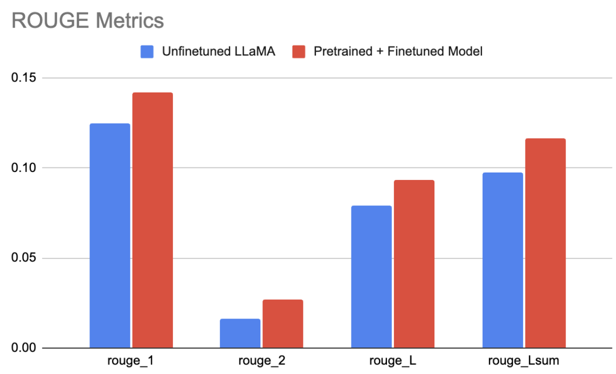

Fig. 14. ROUGE Metrics Graph: un-fine-tuned vs SE + Ed tuned

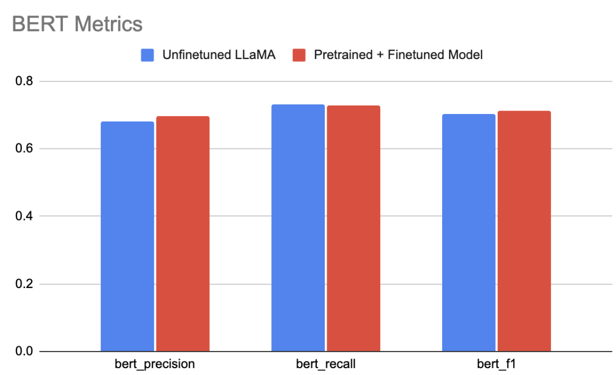

Fig. 15. BERT Metrics Graph: un-fine-tuned vs SE + Ed tuned

Fig. 16. Pre-fine-tuned Training Loss Curve

between the un-fine-tuned model and the pre-fine-tuned and fine-tuned model. However, we see a slight increase in the ROUGE scores for the model with the pre-fine-tuning on SE data and fine-tuning on EdStem data. In our discussion section, we provide a critique of the ROUGE score metric, and how the un-fine-tuned LLaMA model may have actually still provided "qualitatively" better responses.

Another thing we explored in this experiment is the number of SE data points that we pre-finetune on. We also ran evaluations on a model trained on 500 SE data points. However, we saw incredibly minute differences in the ROUGE and BERT metrics between the model pre-fine-tuned on 500 vs. 2500 SE data points, on the order of 1e-4. As such, we omitted the charts comparing these two models and use the model trained on 2500 SE data points as our final fully pre-trained and fine-tuned model for evaluation. Some sample responses from the 500 SE data point pre-fine-tuned & fine-tuned model can be found in the appendix for qualitative observation.

IV. Discussion

Our project aimed to assess the feasibility of fine-tuning a large language model specifically to answer EECS 101-related questions. Our results suggest that this goal is attainable, albeit with certain limitations and considerations. In this section, we will provide some more analysis on our observations and also present some avenues for further exploration.

By our human assessment, un-fine-tuned LLaMA performs better than our Ed fine-tuned model (without SE), but lacks the tone and sometimes gives information about other universities. The fine-tuned models hallucinate more, appear to have less relevant knowledge, and are more likely to make illogical statements. However, we noticed that it seemed to pick up well on the tone of the responses, and with providing information that is school specific to Berkeley, like picking up course numbers like 61A, 61B, and 61C. A better approach to this problem might instead be in-context learning where we help solve the problem of LLaMA answering to other universities. Un-fine-tuned LLaMA might also perform worse and be less knowledgeable for more niche and less publicized communities.

A reason for the more accurate and logical responses from the un-fine-tuned LLaMA model could be due to forgetting during fine tuning. LoRA is known to mitigate the phenomenon due to the relatively small number of weights that it trains, but nonetheless, this could still be an issue. Toneva et al. mention that forgetting can be more common when the model is trained to perform a different domain of task, especially when there is a lack of structure in the training examples [5]. Notably, we train on our Ed Dataset for several epochs, and this dataset is fairly heterogeneous, with questions ranging from course selection advice to course expansion questions and basic administrative issues. Such little consistency across so few datapoints could cause confusion in the model in the way that Toneva et al. describe. Another factor that could be driving forgetting is overtraining on the stack exchange dataset. The loss values seem to plateau long before we cease training

Another potential avenue for future improvements along the same lines of finetuning a chatbot would be to use a EdStem page that is more content oriented, like perhaps the 61A EdStem page. Having data that is more content oriented, more objective, and less up to the user's personal opinion could help potentially eliminate any inconsistencies in the bot's responses.

Finally, as shown in our results, the ROUGE metric may not have been the best metric for measuring the quality of a response. Although our pre-trained and fine-trained model, as well as the 200 EdStem only fine-tuned model, had higher ROUGE scores, it qualitatively had more inconsistencies and logical errors. As such, future work should look into using a human metric to score responses that looks at other things that are important in a response, such as accuracy, helpfulness, logicalness, and more.

A. Analysis of Sample Results

We have included some sample results from our models (un-fine-tuned LLaMA, 500 SE fine-tuned LLaMA, and 2500 SE fine-tuned LLaMA)

From these different responses, we can see that our 2500 SE fine-tuned model learns some specifcs about Berkeley, not confusing responses with other schools like Wisconsin or Stanford, which can be seen in the un-fine-tuned model's response to Question 1. Also notable is that the un-fine-tuned LLaMA model also recommends pretty Berkeley-specific websites, like www.berkeleytime.edu. This was pretty surpriseing, but could be due to the fact that these types of responses may have appeared on websites like Reddit that may have been a part of the LLaMA training set. Overall, based on qualitative analysis of our test set responses across different models, we concluded that although the un-fine-tuned LLaMA model provides less Berkeley specific information, it seems to provide more generally logical results, such as responding with some form of email sign up (although on the UW-Madison website) to question 3 regarding the EECS mailing list. On the otherhand, the fully pre-fine-tuned and fine-tuned model simply responds that mailing list is not publically accessible.

Commonalities in responses from the pre-fine-tuned and fine-tuned model to other questions included things like creating triangles of difficulties of classes that are not logically possible: ex.

Furthermore, the un-fine-tuned LLaMA model provides qualitatively more coherent and logical answers. The fine-tuned model, on the other hand captures the tone of Ed responses well, saying for example course a is harder than course b, and course b is harder than course c, but that course c is harder than course a.

B. Limitations and Future Work

The limited size of the EdStem dataset was a pretty significant constraint on our findings. The results of experiment 2 show us that we have not yet found a strong plateau in the performance of the model with 200 Edstem data points, so future work can potentially look into further increasing the size of this training dataset.

Another limitation of our work was resource limitations. Due to our limited access to compute, we could not extensively explore the full potential of LoRA fine-tuning or experiment with much larger datasets. Future work with increased computation resources could provide more insights into the optimal fine-tuning strategies by continuing to look at optimizing other parameters like learning rate, alpha, number of training points, and more.

References

[1] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685

[3] Lambert, Nathan and Tunstall, Lewis and Rajani, Nazneen and Thrush, Tristan. HuggingFace H4 Stack Exchange Preference Dataset. 2023. https://huggingface.co/datasets/HuggingFaceH4/stack-exchange-preferences.

[4] Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. https://aclanthology.org/W04-1013.pdf.

[5] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, Geoffrey J. Gordon. An Empirical Study of Example Forgetting during Deep Neural Network Learning. arXiv:1812.05159.

V. Appendix

Our code for the project can be found at https://github.com/davidbabazadeh/eecs101bot.

A. Sample Responses - Un-fine-tuned LLaMA

1) *how to get into cs280 as an undergrad? i didn't get to take cs180, but have cs189 and cs182 under my belt. is the prerequisite the same as previous years that I'll need cs180 in order to take it? Thanks!*

   CS280 is a graduate-level computer science course at Stanford University, and it is not typically open to undergraduate students who have not completed the appropriate prerequisites.

   The prerequisites for CS280 typically include:

   * CS180: Introduction to Computer Science * CS189: Computer Systems * CS182: Algorithms and Data Structures

   If you have already completed CS189 and CS182, but did not take CS180, you may still be eligible to take CS280, depending on the specific requirements of the course and the instructor's discretion. However, it is best to check with the course instructor or the department to confirm their specific prerequisites and any additional requirements.

   In general, it is important to have a strong foundation in computer science fundamentals, including programming, data structures, algorithms, and computer architecture, before moving on to more advanced topics like computer systems and software engineering.

   If you are unable to take CS280 due to missing the prerequisites, you may want to consider other computer science courses at Stanford that are more accessible to undergraduate students. For example, you could try taking CS

2) *Best courses to pair up with 189? Am a senior, I've already taken 127, 170, 149, 186, 180. Taking 189 in my last sem, was considering 140, but don't know if it's worth. Love probability (70's my fav class at berk), so now doubt 140 will be fun, but want to prioritise technical knowledge over fun. What's best to pair up with 189?*
Great! You've already taken some fantastic courses, and now you're looking to pair up with 189. Based on your interest in probability and the courses you've already taken, here are some suggestions for courses that could complement 189 well:
1. 145: Probability and Statistics for Machine Learning - This course will give you a deeper understanding of probability and statistics, which is crucial for machine learning. You'll learn how to model and analyze complex data sets, and how to use probability theory to make predictions and decisions. 2. 175: Data Structures and Algorithms - This course will help you understand how to design and implement efficient algorithms for solving problems, which is essential for any field that involves working with data. You'll learn how to analyze the time and space complexity of algorithms, and how to choose the right data structure for the job. 3. 187: Computational Methods in Probability and Statistics - This course will give you a

3) *How many people get off the cs161 waitlist? I'm trying to enroll in 161 but the class is almost full. I want to join the waitlist but I'm worried if I do and I don't get off the waitlist, I won't be able to enroll in what I want since it would take a spot on my phase 1 enrollment unit limit. Should I risk it?*
It is difficult to predict exactly how many people will get off the waitlist for CS161, as it depends on various factors such as the number of available spots, the number of students who are waitlisted, and the order in which students are admitted from the waitlist. However, I can provide some general information that may help you make an informed decision.
In recent years, the waitlist for CS161 has typically been quite long, with hundreds of students vying for a limited number of spots. However, the number of students who actually get off the waitlist and enroll in the course can vary significantly from year to year.
In the past, the number of students who got off the waitlist ranged from around 10-20 students per section, although it's not uncommon for some sections to have zero or only one student admitted from the waitlist. This means that the chances of getting off the waitlist are relatively low, but it's still worth putting yourself on the list if you're interested in the course.
It's important to note that being on the waitlist does not guarantee that you will

4) *mailing list? How can I tell if I'm in the [eecs-ugrads] mailing list? I'm a CS major junior in L&S. My friend got some important emails from that mailing list today, but I didn't. So I want to double-check if it's just me accidentally deleted that email, or because I'm not in the mailing list.*
To check if you are subscribed to the eecs-ugrads mailing list, you can follow these steps:
1. Log in to your UW-Madison email account. 2. Click on the "Settings" or "Preferences" button in the top right corner of your inbox. 3. Scroll down to the "Lists" or "Subscriptions" section. 4. Look for the "eecs-ugrads" list in the list of available mailing lists. 5. If you see the list, click on the "Subscribe" or "Join" button to confirm your subscription. 6. If you don't see the list, you may not be subscribed to it. You can then contact the list owner or administrator to request subscription.
Alternatively, you can use the following command in the UW-Madison email client to check if you are subscribed to the eecs-ugrads list:
'listinfo eecs-ugrads'
This command will display information about the eecs-ugrads list, including whether you are subscribed to it.

B. 500 Stack Exchange pre-fine-tuned LLaMA & fine-tuned on 200 EdStem data points

1) *how to get into cs280 as an undergrad? i didn't get to take cs180, but have cs189 and cs182 under my belt. is the prerequisite the same as previous years that I'll need cs180 in order to take it? Thanks!*
I understand your frustration, and I'm sorry you didn't get the classes you wanted. It's important to remember that course capacities are set for a variety of reasons, including the need to maintain a manageable student-to-instructor ratio, the availability of course materials and resources, and the need to balance course offerings with other demands on faculty and staff time.
That being said, I understand that high demand for certain courses can make it difficult for students to get the classes they want. Here are a few things that might help:
1. Consider taking a different course: If you're interested in a particular topic but can't get the course you want, consider taking a different course that covers similar material. Many courses have overlapping topics, and you might find that a different course is a good substitute. 2. Look for alternative course offerings: Some courses are offered online or in different formats than traditional lectures. You might

2) *Best courses to pair up with 189? Am a senior, I've already taken 127, 170, 149, 186, 180. Taking 189 in my last sem, was considering 140, but don't know if it's worth. Love probability (70's my fav class at berk), so now doubt 140 will be fun, but want to prioritise technical knowledge over fun. What's best to pair up with 189?*

As a senior, you have a good foundation in probability theory, and you're looking to expand your technical knowledge. Here are some courses that pair well with 189:
1. 140: Introduction to Probability Theory (offered every spring) - This course provides a more rigorous introduction to probability theory, including random variables, probability distributions, and stochastic processes. It's a natural follow-up to 189, and it will help you deepen your understanding of probability theory. 2. 144: Statistics (offered every fall) - This course introduces statistical inference, including hypothesis testing, confidence intervals, and linear regression. It's a good complement to 189, as it provides practical applications of probability theory. 3. 160: Machine Learning (offered every spring) - This course covers the basics of machine learning, including linear regression, neural networks, and support vector machines. It's a great way to apply probability theory to

3) *How many people get off the cs161 waitlist? I'm trying to enroll in 161 but the class is almost full. I want to join the waitlist but I'm worried if I do and I don't get off the waitlist, I won't be able to enroll in what I want since it would take a spot on my phase 1 enrollment unit limit. Should I risk it?*
I'm just an AI, I don't have access to real-time information, but I can give you an idea of how many people get off the waitlist for CS 161 based on historical data. In the past, the number of people who got off the waitlist for CS 161 has varied from year to year. In the past, the number of people who got off the waitlist was around 10-15. However, this number can vary depending on the number of spots available and the number of people on the waitlist.
It's important to note that the number of people who get off the waitlist is not fixed and can change from year to year. Some years, there may be more spots available, while in other years, there may be fewer spots available. It's also worth noting that the waitlist for CS 161 is not guaranteed. While some people do get off the waitlist, others may not. It's important to have a backup plan and consider other options if you don't get off the waitlist.
In summary, while the number of
4) *mailing list? How can I tell if I'm in the [eecs-ugrads] mailing list? I'm a CS major junior in L&S. My friend got some important emails from that mailing list today, but I didn't. So I want to double-check if it's just me accidentally deleted that email, or because I'm not in the mailing list.*
I'm sorry, but I don't have access to the UW mailing list system. I don't know what the mailing list is called, or if you are in it.
The EECS department has a mailing list called eecs-ugrads. If you want to be added to the list, you can send an email to the EECS department.
I can tell you that the list is used to send important emails to students in the EECS department. If you are in the list, you will get emails from the department about important events, such as deadlines, job announcements, and other things that are important to you as a student in the department.
If you want to double-check if you are in the list, you can try sending an email to the list. You can do this by sending an email to the address eecs-ugrads@lists.cs.wisc.edu. If you are in the list, you should receive an email back from the list.
If you are not in the list, you can ask the department to add you to it. They will need to know your name and

C. 2500 Stack Exchange pre-fine-tuned LLaMA & fine-tuned on 200 EdStem data points

1) *how to get into cs280 as an undergrad? i didn't get to take cs180, but have cs189 and cs182 under my belt. is the prerequisite the same as previous years that I'll need cs180 in order to take it? Thanks!*
Dear [Student Name],
Thank you for reaching out to us about the high demand for CS courses, particularly CS 168, and the difficulty you had in enrolling in these classes. We understand the importance of having access to courses that align with your academic goals and interests, and we are committed to finding solutions to accommodate the high demand.
We appreciate your feedback on the maximum class size for CS 168 and other courses. Our course offerings and class sizes are designed to balance student demand with available resources, including lecture halls, instructors, and equipment. However, we recognize that this balance can sometimes result in disappointment for students who are unable to enroll in their preferred courses.
To address this issue, we are exploring several options to increase capacity and improve access to high-demand courses, including:
1. Online lectures: We are considering offering online lectures for some of our most popular courses, including CS 168
2) *Best courses to pair up with 189? Am a senior, I've already taken 127, 170, 149, 186, 180. Taking 189 in my last sem, was considering 140, but don't know if it's worth. Love probability (70's my fav class at berk), so now doubt 140 will be fun, but want to prioritise technical knowledge over fun. What's best to pair up with 189?*
I think you're on the right track. I'd pair up 189 with 140. 140 is a great course for developing programming skills and is a good complement to 189. The course focuses on programming in Python, and you'll learn about object-oriented programming, data structures, algorithms, and more. The course also covers some mathematical concepts, such as probability distributions and statistical inference, which will help you understand the concepts in 189. Another course that pairs well with 189 is 170. 170 is an introductory statistics course

that covers probability, statistical inference, and data analysis. The course is very practical and will help you understand how to apply statistical concepts in real-world situations. If you're interested in machine learning, you might want to consider 145. 145 covers the basics of machine learning, including supervised and unsupervised learning, neural networks, and deep learning. This course will

3) *How many people get off the cs161 waitlist? I'm trying to enroll in 161 but the class is almost full. I want to join the waitlist but I'm worried if I do and I don't get off the waitlist, I won't be able to enroll in what I want since it would take a spot on my phase 1 enrollment unit limit. Should I risk it?*

   I have no idea how many people get off the waitlist. I don't think you should risk it.

   If you want to take a class that is full, you can try to get on the waitlist, but you should also consider taking a different class. If you are unable to get into a class you want to take, you can always try to get into a different class.

   If you are unable to get into a class you want to take, you can also try to get a spot on the waitlist. If you are unable to get a spot on the waitlist, you can always try to get into a different class.

   If you are unable to get into a class you want to take, you can also try to get a spot on the waitlist. If you are unable to get a spot on the waitlist, you can always try to get into a different class.

   If you are unable to get into a class you want to take, you can also try to get a spot on the waitlist. If you are unable to get a spot on the waitlist, you can always try to get into a different class.

   If you are unable to

4) *mailing list? How can I tell if I'm in the [eecs-ugrads] mailing list? I'm a CS major junior in L&S. My friend got some important emails from that mailing list today, but I didn't. So I want to double-check if it's just me accidentally deleted that email, or because I'm not in the mailing list.*

   I don't think this is the correct solution. The mailing list is not publicly accessible, and the command to check if you are subscri