Fine-tuning Open-Source LLMs for Automated Research Approach Generation

Name: Soochan Andrew Lee

1. Introduction

Drafting the 'Approach' section of a research paper is one of the most intellectually demanding tasks in scientific writing. Authors must translate an abstract research problem into a coherent sequence of hypotheses, design choices, and evaluation strategies, all while adhering to disciplinary standards of clarity, novelty, and rigor. This process demands deep domain expertise, creativity, and writing agility, posing particular challenges for early-stage researchers attempting to structure their ideas into methodologically sound narratives. Recent advances in Large Language Models (LLMs) offer the potential to assist writers by generating high-quality text continuations, but most current approaches rely heavily on prompt engineering; manually crafting complex prompts to steer frozen models. Despite achieving impressive results, prompt-based methods suffer from critical limitations: models remain contextually blind to niche academic conventions, outputs often lack the structure and coherence expected in academic writing, and the ethical provenance of generated text remains opaque without fine-tuning.

To overcome the limitations of prompt-based generation, this project reframes research approach drafting as a short-to-long text generation task. Given a single-sentence research problem as input, the model is trained to generate a coherent, multi-paragraph research approach proposing hypotheses, methods, and evaluation strategies. Unlike prior work that relies solely on prompting, this project fine-tunes a small, open-source LLM directly on a curated dataset of problem-approach pairs, enabling the model to internalize domain-specific writing patterns without the need for elaborate prompts. Scientifically, the goal is to demonstrate that a compact LLM can be trained, not merely prompted, to produce domain-plausible research approaches. Pedagogically, the project provides hands-on experience in dataset construction, parameter-efficient fine-tuning, evaluation, and critical analysis; all performed on consumer-grade hardware.

Automating research approach generation offers practical benefits, supporting early-stage researchers during ideation, assisting educators with model-generated drafts for comparison exercises, and helping editors and reviewers identify methodological patterns across submissions. However, the task presents several challenges: maintaining long-form coherence, adapting to domain-specific academic language, and managing resource constraints. To address these, the project fine-tunes TinyLlama-1.1B-Chat-v1.0, selected for its balance between size and

capability, using LoRA (Low-Rank Adaptation) to update only a small fraction of parameters efficiently. The model is trained on approximately 200,000 arXiv abstracts and evaluated using BLEU, ROUGE-L, and BERTScore metrics, supplemented by qualitative assessments. Through lightweight fine-tuning on a purpose-built dataset, this project demonstrates that meaningful specialization of an LLM for structured scientific writing tasks is not only possible, but practically achievable on modest consumer hardware.

2. Dataset Construction

The raw material for this task is the publicly available scientific_papers / arxiv corpus on Hugging Face, which contains 203,037 LaTeX-cleaned abstracts. Each abstract was heuristically split into two segments: the first sentence was labelled problem and the remaining sentences were concatenated as approach. Although coarse, this rule captures the common scientific writing style where authors open with a gap statement and follow up with their method.

To keep the training signal focused, three filters were applied: (1) problems longer than 50 tokens were discarded because they usually bundle problem with results; (2) approaches shorter than 90 tokens were removed since they seldom provide methodological detail; and (3) single-sentence abstracts were ignored because they lack an approach part altogether.

After filtering, the remaining dataset was divided into training (90%), validation (5%), and test (5%) splits. This proportional split ensures that model development, tuning, and final evaluation are conducted on disjoint sets, preserving the integrity of the experimental workflow and avoiding data leakage during training.

Final Dataset Size:

Split	Instances	Ratio(%)
Train	103,663	90%
Validation	5,759	5%
Test	5,759	5%

During preprocessing, a total of 87,856 abstracts were excluded based on heuristic filtering rules designed to improve dataset quality. Specifically, 38,686 were skipped due to the problem being too long, 46,035 were skipped due to the approach being too short, and 3,135 were skipped for having too few sentences. These filters helped ensure that the remaining data consisted of relatively clean, high-signal examples suitable for training. However, it is important to note that treating the first sentence of an abstract as the "problem" is a heuristic that does not always yield a genuine or well-defined problem statement. The limitations of this approach, and its potential impact on model performance.

3. Methodology

3.1 Model Selection and Adaptation (LoRA)

Given the constraints of consumer-grade hardware, the base model selected for this project was TinyLlama-1.1B-Chat-v1.0, an open-source LLM with approximately 1.1 billion parameters and a 4096-token context window. The model strikes a balance between size and expressiveness, making it a suitable candidate for experimentation on mid-tier GPUs. To fine-tune the model without exceeding memory limits, the project employed Low-Rank Adaptation (LoRA); a parameter-efficient training method that introduces trainable low-rank matrices into specific layers of the model. LoRA was configured with a rank of 8, a scaling factor (lora_alpha) of 16, and a dropout rate of 0.05. The fine-tuning targeted the key projection layers: q_proj, k_proj, v_proj, o_proj, and gate_proj.

3.2 Tokenization and Input Formatting

To format training samples, four special tokens; <problem>, </problem>, <approach>, and </approach> were added to the tokenizer vocabulary. These tokens were used to wrap the problem and approach sections of each example, allowing for precise segmentation during both training and inference. A label masking strategy was implemented to compute loss only on tokens following the <approach> tag. This ensured that the model treated the problem text purely as context and focused learning on generating the approach portion.

3.3 Training Configuration and Hardware

Training was conducted with a batch size of 2 and a gradient accumulation step size of 2, yielding an effective batch size of 4. The maximum sequence length was capped at 512 tokens, and training proceeded for 100 steps. The learning rate was set to 1e-4, and optimization was performed using the AdamW optimizer. Due to VRAM limitations on the target hardware - an NVIDIA GTX 1660 Ti GPU with 6 GB of memory - the training was conducted entirely in FP32 precision. The full training pipeline was run on a Windows 11 machine equipped with an Intel Core i7-9750H CPU and 16 GB of RAM. The total wall-time for the training run was approximately two hours.

3.4 Model Export and Deployment

Upon completion of training, LoRA adapters were merged into the base TinyLlama weights using the PEFT library. This process produced a single, unified model checkpoint suitable for evaluation, inference, and practical deployment. The resulting checkpoint required no additional adapter modules during inference, streamlining the deployment process significantly.

3.5 Modular and Configurable Pipeline Design

Importantly, the training pipeline is designed to be modular and adaptable. Key parameters such as the number of training steps, maximum sequence length, learning rate, batch size, gradient accumulation steps, and the percentage of the training data used can all be customized via command-line arguments. This allows users to scale the fine-tuning process according to their own hardware and experimentation needs, making the system highly portable and reproducible across a range of computing environments.

4. Experimental Results and Evaluation

4.1 Evaluation Setup

After fine-tuning, the model was evaluated using automatic metrics on a subset of 100 examples from the test split. Evaluation focused solely on the fine-tuned model since no separate zero-shot baseline run was performed. The evaluation was conducted using the evaluate library with the sacrebleu, rouge, and bertscore packages.

4.2 Evaluation Results and Interpretation of Metrics

The metric scores for the fine-tuned TinyLlama model are summarized below:

Metric	Fine-tuned TinyLlama
BLEU	3.50
ROUGE-L	0.1673
BERTScore-F1	0.8189

The BLEU score of 3.50 reflects the n-gram overlap between generated and reference approaches, suggesting that while the model captures key phrases, it does not perform rote copying - an expected characteristic for open-ended generation tasks. The ROUGE-L score of 0.1673, measuring the longest common subsequence overlap, highlights the moderate alignment in structure between model outputs and human-written approaches. The BERTScore F1 of 0.8189, which evaluates semantic similarity based on contextualized embeddings, shows that even when surface-level token overlap is limited, the model maintains a strong conceptual alignment with reference answers. Overall, these results indicate that the fine-tuned TinyLlama model produces fluent, semantically relevant outputs, though there is room for improvement in capturing detailed structure and specificity.

4.3 Loss Progression

The training progression was tracked by monitoring the training loss at regular intervals. Key checkpoints during training are summarized below:

Step	Loss

0	8.57
10	4.44
50	1.86

At Step 0, the loss was initially high at 8.57, reflecting the random initialization of the LoRA-adapted parameters. A steep decline followed immediately, with loss dropping to 4.44 by Step 10, demonstrating that the model quickly learned basic patterns and associations between problems and approaches. By Step 50, the loss further decreased to 1.86, indicating continued, steady improvement as the model assimilated more examples.

4.5 Convergence Behavior and Limitations

After approximately Step 60 onward (beyond what is shown in the snapshot above), the loss curve began to stabilize, fluctuating gently between 1.0 and 1.5, suggesting that the model was entering a steady convergence phase. Importantly, there were no signs of overfitting throughout training: the loss trajectory remained smooth without sudden spikes or erratic behavior, and gradients remained well-behaved.

However, it must be noted that training encompassed only about 0.4% of a full epoch, given the large dataset size (over 100,000 examples) and the small effective batch size (4 samples per optimization step). As a result, although the model's learning curve was healthy and trending positively, the model remains under-trained relative to the full data distribution. It had not yet seen sufficient variety to fully generalize across the diversity of research problems and writing styles present in the dataset.

Nevertheless, the observed dynamics confirm that the fine-tuning setup, data curation, and optimization strategy were well-designed. The model was effectively learning from the limited data it encountered, and extending training to additional steps or full epochs would likely yield further significant gains in output quality and task specialization.

5. Challenges, Insights, and Lessons Learned

The project successfully built an end-to-end fine-tuning pipeline for an open-source LLM, achieving measurable progress despite computational and time limitations. Several important observations and limitations emerged.

5.1 Strengths

- Working pipeline: Data preprocessing, model adaptation, fine-tuning, evaluation, and inference worked as expected.

- Loss reduction: Training loss decreased steadily from 8.6 to 1.86 within just 100 steps, indicating effective optimization.
- Meaningful outputs: The model produced reasonable, fluent approaches aligned with input research problems.

5.2 Challenges

- Data Noise: The heuristic of using the first sentence as the problem and the rest as the approach introduced label noise. Sometimes, the "problem" was background information, and the "approach" included results or discussion rather than methodology.
- Underfitting: Due to time and hardware constraints, only 100 steps of training were performed, representing a tiny fraction of an epoch over the dataset. More extensive training would likely further improve results.
- Metric Limitations: BLEU and ROUGE-L mainly capture token overlap, but generating approaches is a creative task where lexical overlap is not the best indicator of quality. BERTScore helped better capture semantic similarity, but a human evaluation would have provided deeper insights.
- Lack of Baseline: No zero-shot TinyLlama baseline was evaluated. While the LoRA-fine-tuned model outputs appeared fluent and topic-aligned, a direct comparison would have strengthened the evaluation.

6. Lessons & Experience Learned

This project provided practical insights into fine-tuning open-source LLMs for structured writing tasks. One of the key takeaways was the critical role of data quality: while the heuristic splitting of abstracts into "problem" and "approach" sections allowed for scalable dataset creation, it also introduced noise. Many "problems" were actually background statements, and some "approaches" mixed in results or discussions. This highlighted that even large datasets require careful design to provide truly effective supervision.

From a technical standpoint, the use of LoRA proved highly effective. It enabled fine-tuning a 1.1B parameter model within a 6 GB VRAM constraint without requiring full model retraining or quantization. The model converged quickly during training, showing meaningful loss reduction within a short time frame. This confirmed that parameter-efficient fine-tuning strategies make large model adaptation feasible even on modest consumer hardware.

Another important lesson was in evaluation. Standard metrics like BLEU and ROUGE were useful for baseline assessment but fell short in fully capturing the quality of the generated outputs, especially in open-ended tasks like writing research approaches. BERTScore offered better insights into semantic alignment, but manual qualitative review remained essential to judge coherence, technical relevance, and creativity.

Finally, the modular design of the training pipeline - allowing easy adjustment of training steps, learning rate, batch size, and other hyperparameters - proved invaluable. It enabled the system to adapt to hardware limitations without compromising the overall experimentation process.

Overall, the project reinforced that successful LLM fine-tuning is not just about model architecture, but equally about thoughtful data preparation, efficient adaptation methods, and comprehensive evaluation. The experience gained will directly inform future work in training domain-specialized language models under real-world hardware constraints.