Coursework: Improving Basic Mathematical Reasoning in Language Models

Advanced Topics in Natural Language Processing

Yifu Qiu and Shay Cohen

Semester 2, 2025-26

Overview

While large language models (LLMs) are groundbreaking technology, they often struggle with reasoning tasks, particularly mathematical reasoning. In this assignment, you will explore LLM mathematical reasoning using simple maths word problems.

This assignment has two parts. The first part uses LLMs as blackboxes, and analyses a prompt engineering technique to improve mathematical reasoning for word problems. The second part requires deeper coding with LLMs, and makes use of finetuning and reinforcement learning to train models with small datasets.

Note that both parts require working with code and writing new code. The assignment comes with two directories: *partI/* and *partII/*, and for each part you should work independently over the relevant directory.

You will work in pairs to complete this coursework. You are strongly encouraged to partner with a student from a different academic or technical background, as the assignment is designed to benefit from diverse perspectives. Teams with similar backgrounds may find certain questions significantly more challenging.

Part I: Using Explicit Chain-of-thought Steps

Part I is based on the paper Chain of Mathematically Annotated Thought *(link)*, referred to as CoMAT. Before you begin, please skim the paper to familiarise yourself with its key concepts and details. To make the assignment self-contained, we provide the important details below.

How CoMAT Works: CoMAT relies only on the predictions of the LLM to solve the question. It is a relatively straight-forward *prompt engineering technique*. Its steps are detailed below, with each step corresponding to an instruction in the prompt used to solve the question.

Symbolic conversion: The LLM is asked to formalise the natural language question Q in four intuitive steps that you might have internalised in some form in a maths class. The four steps include *identification and definition*, *structural logic translation*, *explicit factual representation*, and *question formalisation*.

Hence, given the following question MQ0:

MQ0

Taylor Swift is planning a concert tour. The venue can hold 50,000 fans. VIP tickets cost $250 each, and regular tickets cost $100 each. If the total revenue from ticket sales is $6,500,000 and all tickets are sold, how many VIP tickets were sold?

s_1 . **Identification and Definition.** CoMAT identifies and defines the relevant variables and constants in MQ0.

- **Variables**: v (number of VIP tickets), r (number of regular tickets)
- **Constants**: T (total capacity of 50,000), P_v (price of VIP tickets, 250), P_r (price of regular tickets, 100), R (total revenue, $6,500,000$)

s_2 . **Structural Logic Translation**: Then, CoMAT extracts the key variables and translates the problem into formal rules that define their relationships:

- $v + r = T$
- $P_v \cdot v + P_r \cdot r = R$
- $v \geq 0, r \geq 0$ (non-negative constraints)

s_3 **Explicit Factual Representation**: CoMAT then integrates all relevant facts into the logical structure:

- $T = 50,000$
- $P_v = 250$
- $P_r = 100$
- $R = 6,500,000$

s_4 **Question Formalisation**: CoMAT formalises the question based on the previous steps:

In our example, we are tasked with finding v, the number of VIP tickets:
Find $v : (v + r = T) \wedge (P_v \cdot v + P_r \cdot r = R)$

Reasoning Execution: During the second part, the problem is solved step-by-step using previous passages, as demonstrated:

Step 1: Express r in terms of v using $v + r = T$:

$$r = T - v = 50,000 - v$$

Step 2: Substitute into the revenue equation $P_v \cdot v + P_r \cdot r = R$:

$$250v + 100(50,000 - v) = 6,500,000$$

Step 3: Simplify:

$$250v + 5,000,000 - 100v = 6,500,000$$

Step 4: Solve for v:

$$v = \frac{1,500,000}{150} = 10,000$$

6. **Derivation of Final Answer**: The final answer is then derived. In this case:

The number of VIP tickets sold is 10,000.

In the next set of questions, we will explore this prompt engineering both programmatically and conceptually. You will learn what makes the case for such step-by-step chain-of-thought style prompting and whether it can be improved.

Q1: Formalising questions

In the first part of the assignment, you will be required to formalise two questions yourself, as if you were the LLM. This should provide you with an idea of the kind of skill a language model needs to possess in order to derive the solution.

The two questions you will work on formalising are:

MQ1

In a neighbourhood, the number of rabbits pets is twelve less than the combined number of pet dogs and cats. If there are two cats for every dog, and the number of dogs is 60, how many pets in total are in the neighbourhood?

MQ2

Company X shipped 5 computer chips, 1 of which was defective, and Company Y shipped 4 computer chips, 2 of which were defective. One computer chip is to be chosen uniformly at random from the 9 chips shipped by the companies. If the chosen chip is found to be defective, what is the probability that the chip came from Company Y?
Choices:
a. 2/9
b. 4/9
c. 1/2
d. 2/3

1. Solve MQ1 and MQ2 using any method or approach of your choice. Show your calculations clearly. *Do not use an LLM to solve MQ1 and MQ2.*

2. Now, follow the steps of Identification and Definition, Structural Logic Translation, Explicit Factual Representation and Question Formalisation for MQ1 and MQ2. Use the example at the beginning of the assignment as a basis for this formalisation. *These steps should be followed manually, rather than with an LLM. Do not use an LLM to solve this question.*

3. Now, based on the formalisation, solve the questions and get an answer ("Derivation of Final Answer"). You do not have to follow the reasoning execution steps.

4. Explain how the structured approach in 2-3 helped you arrive at an answer more mechanically, and critically assess whether it made reaching a solution easier. Discuss both the advantages and limitations of this approach, reflecting on its effectiveness in guiding your reasoning process. Note that there is no right or wrong answer; the goal is to thoughtfully evaluate the process.

5. Following subquestion 2, do you think additional formalisation steps could improve the process? Provide an example of such a step to support your argument. Alternatively, you may argue that no further formalisation steps are necessary and that the current sequence is complete. A strong answer should include specific examples and justification to support your answer.

Q2: Analysis of steps through Shapley values

In this question, we will study the contribution of each step by analysing the different components in the step-wise procedure of formalisation. The analysis is based on the *Shapley value* of the different steps. The Shapley value defines a way to measure the contribution of a "player" to a payoff in a "game." Formally, assume there are n players (in our case, each player corresponds to a step in CoMAT) and that there is a way to measure the level of success of a subset of these n players, $S \subseteq \{1, \ldots, n\}$, in the game, denoted by $v(S)$. Let π be a permutation over the set of n players (meaning, an ordering of the n players). We define the payoff difference for player i as:

$$\Delta_i(\pi) = v(S_i \cup \{i\}) - v(S_i), \tag{1}$$

where S_i is the set of players that precede i in the ordering π. The Shapley value is then defined for each player $i \in \{1, \ldots, n\}$ as:

$$\phi_i = \frac{1}{|\Pi|} \sum_{\pi \in \Pi} \Delta_i(\pi), \tag{2}$$

where Π is the set of permutations over the n players (all orderings).

In this question, we let $v(S)$ be the fraction of questions (over the full evaluation dataset) that are answered correctly when using only the steps in S (the others are omitted). This fraction is calculated on some held-out validation set of questions and answers (number of correctly answered questions from the held-out set divided by the total number of questions in the held-out set).

To make it more concrete, if $\pi = [1, 4, 2, 3]$, and we are focusing on player 2, then $S_2 = 1, 4$. If the question at hand is answered correctly using only steps 1 and 4, then for

that question, we count a correctness of 1; otherwise, 0. Similarly, if the same question is answered correctly using only steps 1, 4 and 2, then we again count it as 1 for that question, otherwise 0. We can compute $\Delta_2([1, 4, 2, 3])$ based on these values. To compute ϕ_2, the Shapley value for step 2, we will range π over all possible permutations, each time creating S_i, which are the steps that precede step 2.

1. Fill in the template provided with this assignment, which consists of two major sections: evaluating the *mmlu-redux* dataset using GPT and calculating the Shapley value. Please follow the instructions carefully.

Code instructions

You will complete 7 questions within the provided code files (excluding sub-questions). Ensure that the completed code is included in the final document. You can grep for the questions by typing *grep Question *.py* on a Terminal. To download the required dependencies, execute the following command:

```
pip install -r requirements.txt
```

Note: Please avoid downgrading or altering the dependency versions.

Now, answer the questions based on the instructions provided below:

(a) *Questions 1 and 2:* These are in *utils.py*, where you will create a function for GPT-based predictions. Follow the provided configuration carefully and ensure it is explicitly passed when calling the OpenAI function.

(b) *Questions 3:* This question is located in *CoMAT_Instruction.py*, where you are instructed to fill in the prompt instructions.

(c) *Questions 4 to 7:* These are in *shapley_value_evaluation.py*, where you will implement functions to evaluate the Shapley analysis.

Note: Helpful tips and explanations have been included in the code. Please retain them for clarity and guidance.

Please carefully follow the instructions in the code and the configurations below to answer the questions.

Configuration:

- `temperature` = 0.0
- `model` = gpt-4o-mini
- `max_tokens` = 2000

To evaluate *main.py* , execute the following command:

```
python main.py --dataset mmlu-redux-college_mathematics
               --method comat --model gpt
```

Note: A higher accuracy does not lead to higher marks, so please *do not* use a more advanced model than gpt-4o-mini. We ask you to use gpt-4o-mini to balance the costs, so you have the ability to evaluate the full dataset multiple times. Each student is allocated approximately $8 in total for this assignment, and a full evaluation of the dataset costs about $0.05. We estimate you could run the evaluation at most 100-150 times from beginning to end. Far fewer runs are actually needed. Budget your tokens!

2. Answer the following questions based on the code for prompting the LLM:

   (a) Evaluate the model using the given configuration with temperature 0.0. What is the accuracy of the model under this setting?

   (b) Adjust the temperature setting to 0.7 and evaluate the model again. What is the accuracy of the model this time?

   (c) Compare the accuracy under this configuration to the original evaluation. Explain why the accuracy is higher or lower when the temperature is set to 0.7 compared to 0.0.

   (d) What alternative configurations other than adjusting temperature could affect performance, and why? Note: Changing the model itself is not a valid answer; focus only on configuration adjustments within the same model. (Consider what other hyperparameters control the inference of the model, for example, the top-p parameter.)

3. You might ask yourself: How much does each step contribute to the success of solving a given question?

   To do this, calculate the Shapley value of each step, with total of 4 steps based on the code you have written. Base your calculations on the file *evaluation_with_steps.csv*, which includes 0/1 values indicating correctness for different configurations of the steps. Do not answer the question based the execution of the different steps yourself. Your calculations will be done by completing the code in *shapley_value_evaluation.py*, which you are required to submit.

   (a) What are the Shapley Values for each step s_1, s_2, s_3, s_4? Include the contribution of each step in the Gradescope form.

(**Note:** Negative Δ values do exist, so please do not assume apriori that the result cannot contain negative Δ values when evaluating the Shapley Value.)

(b) Which step receives the highest Shapley value? Argue why this might be the case.

(c) (Follow-up) Order the steps by their Shapley value. What do you notice about the relationship of the Shapley values to this order? Argue why this relationship is surprising or not.

Part II: Post-Training a Small Model

In this part, you will train a tiny language model (Qwen-2.5-0.5B) to solve grade-school maths problems (taken from the GSM8K dataset). You will first establish a baseline using supervised finetuning (SFT) and then attempt to improve reasoning capabilities using reinforcement learning with Group Relative Policy Optimisation (GRPO). You can use the notebook environment with Google Colab, as in ANLP, which provides access to a T4 GPU (with 16GB VRAM). Detailed instructions for setting up the Colab and Conda environments are provided in `README.md` under *partII/*. Please read these instructions carefully before starting the assignment.

Q3: Supervised fine-tuning

In the first part of the assignment, you will fine-tune two variations of the Qwen model: `Qwen-2.5-0.5B` (Base)[1] and `Qwen-2.5-0.5B-Instruct` (Instruct)[2]. They are open-sourced models, meaning that you can freely run the model locally in Google Colab, by simply set `model_signature` to be either `Qwen/Qwen2.5-0.5B-Instruct` or `Qwen/Qwen2.5-0.5B`. To reduce computational cost, we use a subset of 3,000 samples from the GSM8K training set. From these training samples, 10% are held out as a validation set to monitor training progress. All models are evaluated on 100 samples from the test set.

1. **Code Review:** You might find yourself often working with code generated by AI tools (as an anecdote, AI is already writing 70-90% of the code at Anthropic[3]). An important principle is to always review AI-generated code before using it. AI-generated code frequently contains bugs that compile and execute, but are difficult to detect and may lead to unexpected or even risky code behaviour.

   Before running any experiments, perform a code review of `finetuning/main.py` and identify a bug in the code. Explain how it could affect the experiment. Fix the bug so you can continue with the next questions.

2. **Zero-shot Baselines:** Before training, evaluate both the Base model and the Instruct model on the GSM8K test set in a zero-shot setting. Report the accuracy for both.

   How to run the code

   The code for evaluation is *evaluation/main.py*, simply execute the following command:

   ```
   python evaluation/main.py \
   --model_signature Qwen/Qwen2.5-0.5B-Instruct \
   --output_path ./outputs/Qwen/Qwen2.5-0.5B-Instruct-zero-shot
   ```

[1] https://huggingface.co/Qwen/Qwen2.5-0.5B

[2] https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct

[3] https://www.linkedin.com/posts/noamsp_ai-is-already-writing-70-90-of-the-code-activity-7375151780319617024-kgcG/

3. **Manually Check One SFT Training Sample:** For SFT, we must teach the model a specific format to encourage Chain-of-Thought (CoT) reasoning. Looking into the provided source code (specifically `prompt.py` or the dataset processing function), copy and paste one SFT sample (including prompt and completion; pay attention we do not require code here, but rather the SFT sample itself) used for training (into the Gradescope form). Ensure you include the special tokens (e.g., `<|im_start|>`).

4. **SFT Performance and Training Dynamics:** After fine-tuning both models for the specified number of steps (2 epochs), evaluate them again on the test set.

   (a) Summarise the results: Compare the SFT accuracy of the Base model versus the Instruct model.

   (b) Download the training and validation loss curve from your Weights & Biases (WandB) run and include the plot in your report. Briefly describe the convergence behaviour.

How to run the code

The code for SFT is *finetuning/main.py*, you can execute the following command:

```
python finetuning/main.py \
--model_signature Qwen/Qwen2.5-0.5B-Instruct \
--output_path ./checkpoints/Qwen/Qwen2.5-0.5B-Instruct-sft \
--wandb_token <YOUR_WANDB_API_KEY> \
```

Note that you will need to pass your own Wandb API. You should already use Wandb in ANLP. If you have not done this, following the instructions:

- Go to https://wandb.ai and log in.
- Click your profile picture in the top-right corner.
- Click the "API Key" tab.
- Copy the key.

5. **Technical Analysis:** You likely observed a performance gap between the SFT-Base and SFT-Instruct models, even though they share the same architecture and were fine-tuned on the same GSM8K data. Refer to the *Qwen2.5 Technical Report* [4] (especially Section 4.1). Argue why the Instruct model performs better even after we re-train it?

Q4: Reinforcement learning with verifiable rewards (GRPO)

In this section, you will use Group Relative Policy Optimisation (GRPO) to align the model further. Unlike standard RLHF which requires a learned Reward Model, maths

[4] https://arxiv.org/abs/2412.15115

problems allow us to use **Verifiable Rewards** (rules and heuristics).

What is GRPO? We asked ChatGPT.

Group Relative Policy Optimization (GRPO) is a reinforcement learning method for improving large language models that focuses on *comparing answers rather than scoring them in isolation.* Instead of training an extra "critic" model to judge how good an output is (as in PPO), GRPO asks the model to generate several candidate answers to the same prompt and then learns from which answers are better than others. When **verifiable rewards** are available—such as checking whether a math answer is correct or whether code passes tests—these checks are used to rank the candidates. This makes training simpler and more stable, because the model only needs to learn "which answer is better," not "how good is this answer in absolute terms."

1. **Reward Function Implementation:** We do not use a neural reward model. Instead, we define two Python functions: `format_reward_func` and `correctness_reward_func`.

   - `format_reward_func`: (Look back to the SFT sample you copied in Q3.2.) Checks if the model's outputs (`completions`) adheres to the answer format defined in the prompt template used for SFT (i.e., check if `"the answer is"` is in the completion).
   - `correctness_reward_func`: Use string pattern matching commands (such as the regular expression library) to parse the numeric answer and check whether the ground truth (`answer`) is in model's outputs (`completions`).

   Copy and paste your implementation of these two functions below. *Note: Set the reward value to 0.5 for format compliance and 2.0 for a correct answer.*

   Implement this code

   ```
   def format_reward_func(completions, **kwargs):
       # Your code here
       pass

   def correctness_reward_func(prompts, completions, answer, **kwargs):
       # Your code here
       pass
   ```

2. **Training Observations:** Continue training your best SFT model using the GRPO trainer with the rewards defined above. Report the final accuracy compared to the SFT baseline. Additionally, report observations with the qualitative cases: How did the text generation style change? Did the frequency of the phrase `"the answer is"` increase?

How to run the code

The code for GRPO is *grpo/main.py*, you can execute the following command:

```
python grpo/main.py \
--model_signature Qwen/Qwen2.5-0.5B-Instruct \
--adapter_path ./checkpoints/Qwen/Qwen2.5-0.5B-Instruct-sft \
--output_path ./checkpoints/Qwen/Qwen2.5-0.5B-Instruct-sft_grpo \
--wandb_token <YOUR_WANDB_API_KEY>
```

Just login to your Wandb, and you will find an automatically created new workspace, `nlu-gsm8k-grpo`, for you to monitor the training curves.

3. **Failure Analysis:** Analyse the training curves (Reward vs. Steps). You may notice that the `format_reward` increases rapidly, but the `correctness_reward` oscillates or the overall accuracy drops/stagnates. Explain this phenomenon and search the relevant literature and find what this phenomenon is called. Why might the model prioritise formatting over solving the math problem?

Q5: Open-ended exploration

With the help of LLMs (such as ChatGPT/Gemini/Claude) and the compute resources available to you, propose and implement one significant improvement to the above pipeline. The evaluation of this section is based on the maturity of your short report and the rationale behind your idea, rather than raw performance. The idea does not have to be very complex or require demanding coding/compute.

1. Describe your proposed method and why you believe it should improve performance.
2. Implement the method, run the experiment, and report the results. Discuss whether the hypothesis was validated.

Submission Procedure

Timeline We recommend starting early to work on the assignment. The deadline is March 13, 2026 (Friday) at 12:00.

Usage of Edinburgh's access to language models As part of this assignment, you were provided an OpenAI key to use with ELM. You will need to use this key as part of your code (Part I). Note that you should *not* use this key for any other reasons. In addition, note that you should not run your code through the whole dataset more than 100 times or so (which should be plenty to complete the code). Your budget is $8. If in doubt, run the code on a small part of the dataset until you have a stable version of the code. You should *only* use the gpt-4o-mini model. If you have not yet received your key (search for an email with the string "ATNLP Coursework" in the subject from scohen@inf.ed.ac.uk), please contact us as soon as possible.

Submission entry point Your group's solution to the assignment should be submitted on Gradescope. The Gradescope submission entry is a *form* where you will need to put your answers in the specific rubric as detailed in the above script. We highly recommend to:

- Write down your answers on a scratch pad or in Overleaf, and then copy paste your answers into Gradescope (possibly formatting them if needed) when you feel ready. Gradescope does have a "Save answer" button, where you can enter your answers gradually, but keeping a separate document could save your group a lot of trouble.
- Not leave the submission of the assignment to the last minute! Make sure you are familiar with the form beforehand, and that you have enough time to enter your solution properly.

If you used maths equations in your answer, and used Overleaf or LaTeX for that, you may need to convert them into Gradescope. The most likely thing is that you would need to change $ to $$. See more information here: https://guides.gradescope.com/hc/en-us/articles/21591530268429-Can-I-use-LaTeX-on-Gradescope.

Good scholarly practice This assignment is intended to be done in pairs. You may freely discuss and share work within your pair, but may not share or make available solutions or code outside your group. As usual, please use care when posting questions to ensure you don't give away parts of the solution. In addition, you are responsible for following the University academic misconduct policies regarding all assessed work for credit. Details and advice about this can be found at https://informatics.ed.ac.uk/taught-students/all-students/your-studies/academic-misconduct.

We are aware that some of the code required for this assignment may be available online. You are expected to write the code on your own – and there are tools to identify code similarity that indicate a suspect copying of a code compared to a baseline code.

Disclaimer Writing assignments can be challenging, and we are aware of that. In designing this coursework, we aimed to balance between developing critical thinking skills about LLMs as well as help you with skills that directly relevant in industry. We also had to carefully consider the use of computational resources, ensuring that the scale of the assignment is appropriate. In addition, the assignment has been designed so that, even if a student chooses to use GenAI tools to solve certain parts (contrary to our policy), there would still be barriers that require self-thinking. If you would like to provide *any* anonymous feedback on the assignment, especially feedback you may not feel comfortable sharing on Piazza, we would be very grateful. This will help us improve this assignment or design better ones in the future. You can use the feedback form linked here: https://forms.office.com/e/2sucwepQCr.

 **Bug hunt**  As this is a largely new assignment, despite extensive testing there may still be some undetected issues or bugs. To encourage early feedback and work on the assignment, the first student or pair to identify a significant bug, within the first week after the assignment is officially released, and judged important by the lecturers, will receive a bonus of up to 5 marks on the assignment. Bugs reported within the first two weeks may still be eligible for a slightly smaller bonus. Please send bugs directly to `scohen@inf.ed.ac.uk`. (Note this does not include Q3.1, where the bug is intentional.)