Start Date: 30.10.2025
Due Date: 20.11.2025, *23:55*

CENG467
PROJECT 1

Overview for CoMAT:

While large language models (LLMs) are groundbreaking technology, they often struggle with reasoning tasks, particularly mathematical reasoning. In this assignment, you will explore prompting techniques for mathematical reasoning using simple math word problems and analyze LLMs in this context. The work is based on the paper Chain of Mathematically Annotated Thought (https://arxiv.org/pdf/2410.10336), referred to as CoMAT. Before you begin, please skim the paper to familiarize 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 large language model 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 formalize the natural language question Q in four intuitive steps that you might have internalized in some form in a math class. The four steps include identification and definition, structural logic translation, explicit factual representation, and question formalization.

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?

Step 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, $6,500,000$)

Step 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)

Step 3 - Explicit Factual Representation: CoMAT then integrates all relevant facts into the logical structure.

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

Step 4 - Question Formalization: CoMAT formalizes the question based on the previous steps.

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

Step 5 - Reasoning Execution: During the second half of the process, 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:

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

Step 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 set of questions below, we will explore CoMAT 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.

Overview for Shapley Analysis:

In this project, we will also study the contribution of each step by analyzing the different components in the step-wise procedure of formalization. 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:

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

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

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

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

In this project, 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 preceded step 2.

Project Questions:

The questions below are designed to help you become familiar with solving math problems methodically, following an analysis of solving such problems with an LLM.

All questions (that require mathematical operations or writing down answers/comments) MUST BE HANDWRITTEN ON AN A4 PAPER. We will not accept any homeworks written digitally on Word or any similar program.

Q1 – Formalizing math questions:

In the first part of the assignment, you will be required to formalize a math question 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 math question you will work on formalizing is:

MQ1
An empty fuel tank with a capacity of 218 gallons was filled partially with fuel "A" and then to capacity with fuel "B". Fuel "A" contains 12% ethanol by volume and fuel "B" contains 16% ethanol by volume. If the full fuel tank contains 30 gallons of ethanol, how many gallons of fuel "A" were added? *Choices:* *A. 122* *B. 150* *C. 100* *D. 80* *E. 50*

Q1.a. (2 points) Solve MQ1 using any method or approach of your choice. Show your calculations clearly. Do not use an LLM to solve MQ1. It should be an approach you preferred to use for solving similar questions when you were in high school or earlier.

Q1.b. (5 points) Now, follow the steps of Identification and Definition, Structural Logic Translation, Explicit Factual Representation and Question Formalization for MQ1. Use the example at the beginning of this document as a basis for this formalization. These steps should be followed manually, rather than with an LLM. Do not use an LLM to solve this question. Instead, assume that you are an LLM.

Q1.c. (3 points) Now, based on the formalization, solve the question and get an answer ("Derivation of Final Answer"). You do not have to follow the exact Reasoning Execution steps shown in Step 5. It simply needs to be like the Reasoning Execution steps while being based on your work from Step 1 to 4.

Q1.d. (5 points) Explain how the structured approach in Q1.b and Q1.c 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.

Q1.e. (5 points) Following Q1.b, do you think additional formalization steps could improve the process? Provide an example of such a step to support your argument. Alternatively, you may argue that no further formalization steps are necessary and that the current sequence is complete. A strong answer should include specific examples and justification to support your answer. Note that there is no right or wrong answer; the goal is to thoughtfully explain your opinions.

Q2 – Running evaluations & Analysis of steps through Shapley values:

Q2.a. (10 points) Fill in the code template provided with this assignment, which consists of two major sections: evaluating the mmlu-redux dataset using QWEN model (from Huggingface Transformers) and calculating the Shapley value. Please follow the instructions written inside the code files carefully. You will complete multiple STUB sections within the provided code files. Do not change anything else on the files including the comment sections. Ensure that the completed code is included in your final submission.

To download the required dependencies, execute the following command: "`pip install -r requirements.txt`". Please avoid downgrading or altering the dependency versions. You are recommended to use Python 3.12. Afterwards, please go to the https://pytorch.org/get-started/locally/ website to download an appropriate PyTorch version for your own computer. If possible, make sure you can use your GPU with PyTorch with this code line "torch.cuda.is_available()". (It should return True when run on the Python console after "import torch".) There are many online resources available if you need help using your GPU with torch.

In case your PC is unable to run this code, follow the instructions in this link to run your entire code on Google Colab: https://medium.com/swlh/leverage-google-colab-gpu-runtime-for-your-non-notebook-python-project-d13840c932eb

In Colab, make sure to set your runtime environment to GPU device. You can simply use the free T4 GPU option for this project. Other paid GPU options are also acceptable.

Q2.a.I. There is one STUB section in utils.py, where you will create a function for QWEN-based predictions. Follow the provided configuration (placed in the comments) carefully and ensure it is explicitly passed when calling the relevant Transformers function. (Hint: To fully understand what is being performed by the code, carefully examine main.py and mmlu_redux.py, even though they do not have STUB sections.)

Q2.a.II. There is a very small STUB section in CoMAT_Instruction.py, where you need to fill in the prompt instructions. (Hint: This step is much simpler than it initially seems.)

Q2.a.III. There are 8 small STUB sections 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 in Q2.b - Q2.j.

Configurations:

- temperature = 0.1 OR 0.7
- max_token_limit = 2000 OR 4000
- model = qwen2 OR qwen3

To evaluate main.py, execute the following command:

```
python main.py --dataset mmlu-redux-college_mathematics
               --method comat --model qwen2 --temperature 0.1
               --max_token_limit 2000
```

Q2.b. (5 points) Evaluate the model using the given configuration with temperature=0.1 and max_token_limit=2000. Use this specific QWEN2 model:

https://huggingface.co/Qwen/Qwen2-1.5B-Instruct?library=transformers

If this model is too large for your system, you can use https://huggingface.co/Qwen/Qwen2-0.5B-Instruct.

It may take a considerably long time to run the code for the entire mmlu-redux dataset (100 examples). If it is taking too long for you, you can simply let it run for a few hours and observe the results yourselves.

What is the accuracy of the model under this setting? Note that you can view the output log for each example in real time in the following file: "<your project folder>\final_results\mmlu-redux-college_mathematics\comat\qwen2\comat_qwen2.json". If you didn't run for the entire dataset, you can give your accuracy results from this file by manually counting the correct predictions.

Additionally, the code that checks whether the predicted answer is option A, B, C, or D is very basic (simple regex matching in the mmlu_redux.py file). Even if a question's answer is correct, the regex check may fail to find A, B, C, or D. This causes it to accept the final answer as -1. So, you should carefully check the file to see if a case like this happened or not. If so, readjust your accuracy calculations.

Q2.c. (5 points) Evaluate the model again with a single difference from before: temperature=0.7. What is the accuracy of the model under this setting? Do not forget about the additional details mentioned in Q2.b.

Q2.d. (5 points) Compare the accuracy under both configurations above. Explain why the accuracy is higher or lower when the temperature is set to 0.7 compared to 0.1. What is the effect of the temperature parameter?

Q2.e. (5 points) CoMAT is a robust method with consistently strong and well-documented results. Do you believe it may have underperformed in the previous experiments? If so, what potential factors might explain this outcome?

Q2.f. (5 points) Now, repeat the evaluation in the Q2.b with the more advanced **QWEN3** model (https://huggingface.co/Qwen/Qwen3-1.7B?library=transformers or https://huggingface.co/Qwen/Qwen3-0.6B). Only experimenting with temperature=0.1 is enough. Also, you do not need to run the code for the entire dataset as well. Explain what you observed during the experiment. How long did the code run for compared to QWEN2 and what results did it achieve? Explain what the reasons behind the difference in results can be.

Q2.g. (5 points) Repeat the experiment from Q2.f, but increase the max_token_limit to 4000. Observe the results similar to before. Is there any change in the results and what could be their cause?

Q2.h. (5 points) 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 (from "Identification and Definition" to "Question Formalization") 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 manually based on the execution of the different steps yourself. Your calculations will be done by completing and running the code in shapley_value_evaluation.py. Keep in mind that the Shapley Value Analysis part of this project is independent from the rest of the codes you filled in and executed. The evaluation_with_steps.csv contains a toy example for you to easily understand Shapley Value Analysis.

What are the Shapley Values for each step s_1, s_2, s_3, s_4? Write down the contribution of each step. (Note: Negative Δ values do exist, so please do not assume that the result cannot contain negative Δ values when evaluating the Shapley Value.)

Q2.i. (5 points) Which step receives the highest Shapley value? Argue why this might be the case.

Q2.j. (5 points) Based on Appendix D in the CoMAT paper, removing both Steps 1 and 2 results in a smaller accuracy drop than removing Step 1 alone (4.48% higher accuracy). Argue what this tells about the dependence between the steps.

Q3 – Analyzing a case in which CoMAT does not work:

In this question, you will identify an example math question for which the steps in CoMAT fail to provide the correct answer. You will have to manually investigate the dataset provided and analyze at least a few questions. Note that you should answer this question by comparing results from Q2.b and Q2.g. Since CoMAT is far from being perfect, it should not take you too long to find a case in which it fails. Aim to find a question where QWEN2 fails, but QWEN3 succeeds. Hence, after finding a failed math question Q_f:

Q3.a. (2 points) Write down the full set of steps CoMAT provides through the large language model for Q_f in which the final answer is incorrect. If it has been solved correctly by QWEN3, write down its correct solution as well.

Q3.b. (3 points) Analyze the steps CoMAT follows and identify where it fails.

Q3.c. (5 points) Can you now suggest an additional step that would help fix this failure (other than using a more advanced model)? Note that there is no right or wrong answer; the goal is to thoughtfully evaluate the process and suggest your ideas.

Q4 – Finetuning with GRPO Reinforcement Learning Algorithm:

The **GRPO (Generalized Reinforcement Policy Optimization)** algorithm is a reinforcement learning method designed to improve upon traditional policy optimization techniques like PPO (Proximal Policy Optimization). GRPO generalizes the optimization objective to provide a more flexible and stable training framework by adaptively balancing exploration and exploitation, ensuring smoother updates to the policy network.

In the **GRPO** approach — particularly as applied to LLM alignment — the model generates **multiple completions (responses)** for each prompt. Instead of evaluating each completion in isolation, GRPO **compares them *relatively***. This means that for a given prompt, the algorithm looks at all the sampled completions and assigns **relative preference scores** (e.g., ranking or pairwise comparisons) rather than relying on an absolute numeric reward. The optimization then updates the policy to increase the likelihood of producing responses that score higher *relative to others* for the same prompt.

This relative comparison mechanism helps stabilize training and reduces sensitivity to noisy or inconsistent reward signals, since the model learns primarily from *which completions are better* rather than *how good they are in absolute terms*.

In this question, we will experiment with a simpler example using CoMAT prompts. You are provided with a code that finetunes on 80 examples from mmlu-redux dataset and evaluates on the remaining 20 examples. We are using a simple reward function that only rewards for correct answers in the last 20 characters of a model's completion. For example, if the option letter " B " is in the last 20 characters in the model completion outputs, and the correct answer is B; then, the reward will be 1.0. Otherwise, reward will be 0.0. Keep in mind that we denote options A to D as indices 0 to 3 (in the code).

When running the finetuning in the code, we will only use the default parameters (except output directory) of the GRPO classes from the "trl" library (from HuggingFace). This speeds up finetuning significantly at the cost of acquiring less reliable finetuned models. However, for this toy example, using default parameters is enough. By default, the GRPO algorithm compares 8 different completions per math question. So, remember that many parts of the code are written to handle lists of 8 elements.

In the code, an evaluation that saves generation results for test dataset have been performed before and after the finetuning step. The results are saved in their corresponding json files in the previously stated output directory. You can also view the change in finetuning and evaluation loss while running the code.

The model we will finetune on is the smallest QWEN2 model, which is https://huggingface.co/Qwen/Qwen2-0.5B-Instruct.

Q4.a. (5 points) You are also given a code file named "grpo_finetune.py". There are 2 STUB sections in this file that you are expected to complete. The code already has an example scenario to check the reward function. Use this section to make sure reward function works correctly. Its output should be [1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0]. Remember to follow the additional instructions provided in the comments of the file.

After filling in the STUB sections, execute the code. After executing the code, find and observe the output json files for before and after the finetuning step. ***Select an example question and write down its outputs for both versions.*** Your chosen question does not need to be correct. Try to choose a question with a noticeable improvement when observed by a human.

You will notice that the model will not be able to achieve success comparable to our previous experiments. ***Try to explain what kinds of reasons may lead to this outcome.*** For example, what can you say about our reward function and train set size?

Q4.b. (5 points) Try to find any improvements in the outputs due to GRPO algorithm in the json files. You can discuss using multiple examples. Explain how GRPO algorithm can improve CoMAT's answers even if it isn't a substantial improvement in answer quality.

Q4.c. (5 points) Can you suggest a better reward function compared to the one in the code. There is no wrong answer. Simply, discuss how the reward function can be improved.

Submission Rules:

- If you are using Colab, we recommend that you move your entire code to Google Drive and mount the corresponding Drive disk to run the code. Do not try to copy and paste the entire code on a Colab project as it may cause issues.
- All questions (that require mathematical operations or writing down answers/comments) MUST BE HANDWRITTEN ON AN A4 PAPER. We will not accept any project reports written digitally on Word, etc.
- You are expected to submit a project report named "CENG467_project1_report_<groupnumber>.pdf". The contents of this report should be handwritten and scanned using an app like CamScanner. Make sure that the top-right corner of the first page has the group no, names and numbers of your group members.
- You should also submit the entire codebase (after you have filled the STUB sections) in a zip file named "CENG467_project1_code_<groupnumber>.zip". Make sure that your group no, your names and numbers are written (in comments) on the top of each code file you performed modifications on.
- In Teams, it is enough for only a single group member to upload these 2 files. (For example, "*CENG467_project1_report_G07.pdf*" and "*CENG467_project1_code_G07.zip*")
- Any collaborations between different groups are not allowed.